Matrox Graphics Inc.

Matrox MGA-1064SG Developer Specification

Document Number 10524-MS-0100
February 10, 1997

Trademark Acknowledgements

MGA,™ MGA-1064SG,™ MGA-1164SG,™ MGA-2064W,™ MGA-2164W,™ MGA-VC064SFB,™ MGA-VC164SFB,™ MGA Marvel,™ MGA Millennium,™ MGA Mystique,™ MGA Rainbow Runner,™ MGA DynaView,™ PixelTOUCH,™ MGA Control Panel,™ and Instant ModeSWITCH,™ are trademarks of Matrox Graphics Inc.

Matrox® is a registered trademark of Matrox Electronic Systems Ltd.

VGA,® is a registered trademark of International Business Machines Corporation; Micro Channel™ is a trademark of International Business Machines Corporation.

Intel® is a registered trademark, and 386,™ 486,™ Pentium,™ and 80387™ are trademarks of Intel Corporation.

Windows™ is a trademark of Microsoft Corporation; Microsoft,® and MS-DOS® are registered trademarks of Microsoft Corporation.

AutoCAD® is a registered trademark of Autodesk Inc.

Unix™ is a trademark of AT&T Bell Laboratories.

X-Windows™ is a trademark of the Massachusetts Institute of Technology.

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

Contents

Chapter 1: MGA Overview

Chapter 2: MGA-1064SG Overview

Chapter 3: Resource Mapping

Chapter 4: Register Descriptions

Chapter 5: Programmer’s Specification

Chapter 6: Hardware Designer’s Notes

List of Figures

Chapter 1: MGA Overview

Chapter 2: MGA-1064SG Overview

Chapter 3: Resource Mapping

Not Applicable

Chapter 4: Register Descriptions

Not Applicable

Chapter 5: Programmer’s Specification

Chapter 6: Hardware Designer’s Notes

List of Tables

Chapter 1: MGA Overview

Chapter 2: MGA-1064SG Overview

Not Applicable

Chapter 3: Resource Mapping

Chapter 4: Register Descriptions

Not Applicable

Chapter 5: Programmer’s Specification

Chapter 6: Hardware Designer’s Notes

Chapter 1: MGA Overview

This chapter includes:

1.1 Introduction

The Matrox MGA-1064SG is a next generation 3D graphics, multimedia and Windows accelerator. In one low-cost package, the MGA-1064SG:

- Provides superior Windows performance
- Accelerates 3D texture mapped consumer applications such as PC games with the Matrox Fast Texture Architecture
- Is fully Microsoft DirectDraw and Direct 3D compliant
- Accelerates digital video including software MPEG
- Has fast VGA acceleration
- Includes an integrated DAC.

The Matrox MGA-1064SG has special features specifically designed to provide superior 3D game performance in a 2 MByte frame buffer. Matrox MGA-1064SG is intended to provide a complete solution for home PC users who are interested in top performance in 3D game and multimedia applications, but who are also interested in leveraging their home PC as a home office and education center. It is also suitable for GUI environments such as Windows NT, IBM OS/2 PM, Unix X-Windows, and AutoCAD.

The MGA-1064SG series has the same Windows acceleration core as the award-winning Matrox MGA-2064W used on the MGA Millennium graphics card, but works with either SGRAM or SDRAM rather than WRAM. The MGA-1064SG is capable of supporting frame buffer sizes from 2 to 8 MBytes. A key design feature of MGA-1064SG is the memory/graphics clock generator. This supports a ratio of 1.5:1 which optimizes memory bandwidth and the graphics engine clock.

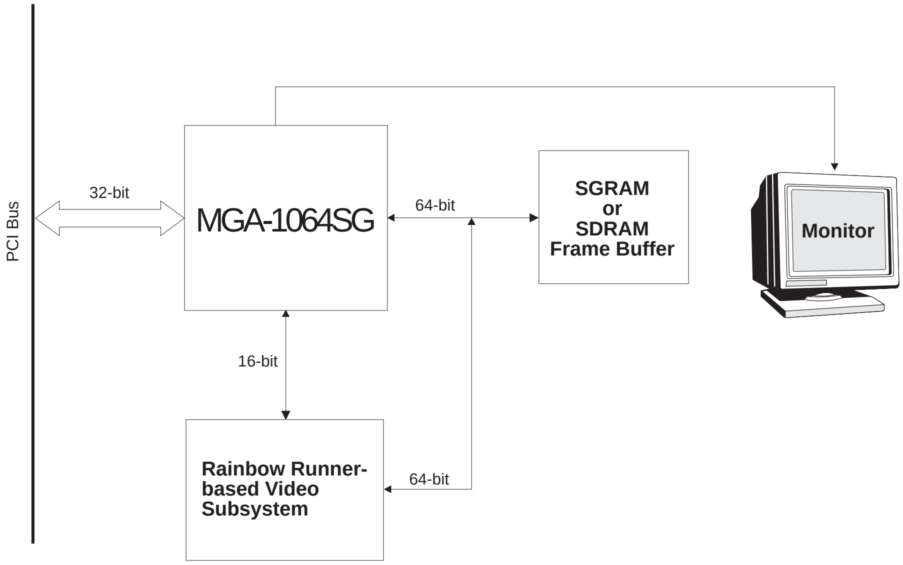
The integrated DAC in MGA-1064SG eliminates the need for an external DAC. This substantially lowers the cost and space required for the graphics sub-system.

The centerpiece of MGA-1064SG is the Matrox Fast Texture Architecture, a full-featured 3D rendering engine, the capabilities of which are aimed squarely at optimizing 3D texture mapped games. This 3D engine is an advanced renderer with full perspective correct texture mapping, lighting, Gouraud shading, optional 16-bit Z-buffering, bus mastering, efficient use of texture memory, keying on textures, high color output (16 bpp), and the ability to work in conjunction with the video engine to use video as a source for texturing. The key feature of the Matrox Fast Texture Architecture is excellent cost/performance. Matrox’s texture compression model saves on memory usage, enabling low cost and high performance in a 2 MByte frame buffer

The MGA-1064SG core engine fully implements the Matrox Video Architecture with its integrated digital video scaling, filtering and color space conversion engine. This architecture supports both shared frame buffer and split frame buffer (overlay) modes of operation to provide maximum flexibility in combining video with graphics. This architecture supports video sprites, video texture maps, graphics overlay, and many other methods of combining video with graphics. The MGA-1064SG can be upgraded with the Matrox Rainbow Runner proprietary video engine to achieve a comprehensive set of high quality video capabilities.

1.2 System Block Diagram

Figure 1-1: System Block Diagram



1.3 Application Areas

- Windows accelerator with high performance levels, which is ideal for mid-range system requirements. The MGA-1064SG will broaden the MGA family's market penetration by delivering a strong price/performance point for users who do not need the top performance at ultra high resolution and color depths provided by MGA Millennium.
- Full acceleration of the next generation of Windows multimedia and game applications. Specifically, 3D texture mapped games achieve a significant boost in performance and image quality with the MGA-1064SG 3D engine. In addition, all other types of games will be accelerated by a combination of the MGA-1064SG's DirectDraw and Direct Video engine.
- Digital video playback is accelerated to full screen, full motion, with high-quality scaling.The architecture supports all of today's popular CODECs including Indeo and software MPEG. OM-1 and Quartz compatibility is provided.
- Full acceleration of all MS-DOS applications via MGA-1064SG's ultra-fast 32-bit VGA core.

1.4 Typical Implementation

MGA-1064SG is ideal for use as an add-in graphics card, or on the motherboard with frame buffer RAM.

1.4.1 Target Markets

- The home, SOHO, and multimedia PC markets
- Mainstream business markets
- The computer gaming market

1.5 Features

1.5.1 Core GUI Accelerator

- Based on the current award-winning MGA-2064W core
- Line draw engine with patterning
- 3D polygons with Gouraud shading
- Optional Z-buffer
- 2D polygons with patterning capabilities
- BITBLT engine
- Sync reset input for video genlock and overlay
- DPMS and Green PC support
- Hardware pan and zoom
- DDC level 2B compliant
- 44 MHz drawing engine
- 66 MHz operation for the memory interface

1.5.2 3D Texture Mapping Engine

- Perspective Correct Texture Mapping
- True color lighting of textures
- Hardware dithering of lit textures
- 16-bit Z-buffer (optionally enabled or disabled)
- Double buffering
- Screen transparency
- Storage of source textures in off-screen frame buffer memory
- Source textures may be in following formats:
  - Color Look Up Table (compressed) 4 bpp, 8bpp.
  - 3:3:2, 5:6:5, 5:5:5.
- Look-up table translations from CLUT4 and CLUT8 to 5:6:5 on-the-fly
- Command list processing via bus mastering
- Keying on textures is supported
- Digital video as source for texturing is supported

1.5.3 Digital Video Engine

- True linear interpolation scaling filter in both X and Y
- Hardware color space conversion engine
- 8 MByte window for ILOAD and IDUMP operations
- Split frame buffer support for true graphics overlay (graphics and video are in separate sections of the frame buffer, with keying in the DAC):
  - 2G8/V16 : graphics 8-bit pseudo color or 3:3:2, video 5:5:5 dithered
  - G16/V16 : graphics 5:6:5 & 5:5:5, video 5:5:5 dithered
  - Synchronized video/graphics updates (no tearing) are supported
  - Supports any number of video windows/sprites simultaneously
  - Split frame buffer is supported simultaneously with shared frame buffer mode layering
  - Direct frame buffer access sees each buffer linearly
- Shared frame buffer mode supports graphics and video written to a shared surface through layering
  - Supports 8, 16, 24, or 32 bit/pixel configurations
  - Graphics and video pixels must have the same pixel depth

1.5.4 DirectDraw Support

- Hardware scaling and color space conversion engine fully accelerates digital video
- Support BITBLT & ILOAD functions with color key for full transparent blit support
- Full 8 MByte window on linear mapping of frame buffer
- Equality compare with plane masking for transparent blits
- Single register page flip
- Programmable blitter stride
- Ability to read the current scan line
- Ability to tell when the vertical blank begins
- Interrupt generated on VSYNC

1.5.5 Direct 3D Support

- Texture mapping
  - Perspective correct
  - Monochrome and true color lighting
  - Decal
  - Texture wrapping and clamping
  - 16-bit true color or 8- or 4-bit palettized
- Gouraud shading
- Optional Z-buffer and Z-test
- Color and Z-masking
- Dithering

- Sub-pixel positioning
- Transparency
- Flat alpha stipple
- Bus mastering

1.5.6 Integrated DAC

- 135 MHz operation
- Supports shared memory and graphic overlay modes
- 3 x 256 x 8 lookup table
- Hardware cursor
- VGA compatible

1.5.7 Synchronous Memory Interface

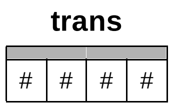
- SGRAM 256K x 32. Supports block write and write per bit for added performance
- SDRAM 256K x 16, 1M x 16, 2M x 8
- Supports from 2 to 8 MBytes of memory
    - Up to 4 banks of 256K x 32 SGRAM
    - Up to 2 banks of 256K x 16 SDRAM
    - 1 bank of 1M x 16 SDRAM
    - 1 bank of 2M x 8 SDRAM.

1.5.8 Miscellaneous

- Feature connector interface
    - 8-bit VGA mode
    - 16 bit output mode for the Rainbow Runner video encoder (32 bpp multiplexed to 16 bit bus).
- Host interface
    - PCI 2.1 compliant.
    - PCI bus master capable. Primarily used to increase 3D performance by off-loading the CPU.
- VESA 2.0-compliant DOS applications running at a resolution of 320 x 200 can be scaled up to 640 x 480.
    - Higher resolutions are possible without changing the frame rate.
    - Filtering in the X-direction is supported.

1.6 Typographical Conventions Used

Table 1-1: Typographical Conventions

Description	***Example***
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.	**DEVID**
Fields within registers are indicated by lower-case bold sans-serif letters.	**vendor**
Bits within a field appear within angle brackets, separated by a colon.	**vendor**<15:0>
Hexadecimal values are indicated by a trailing letter ‘h’.	CFFFh
Binary values are indicated by a trailing letter ‘b’ or are enclosed in single quotes, as: ‘00’ or ‘1’. Also, in a bulleted list in a register description field, 0: and 1: are assumed to be binary.	0000 0010b
Special conventions are used for the register descriptions. Refer to the sample register description pages in Sections 4.1.1, 4.1.2, 4.2.1, and 4.3.1.	
In a table, X = “don’t care” (the value doesn’t matter)	1X = Register Set C
Emphasized text and table column titles are set in bold italics.	This bit ***must be set***.
In Chapter 5’s **DWGCTL** illustrations, the ‘+’ and ‘#’ symbols have a special meaning. This is explained in ‘Overview’ on page 5-26.	

- A vertical bar in the margin (as seen here on the right) indicates a change made since the release of Revision 1 of this manual, which was dated April 5, 1996.

1.7 Locating Information

The MGA-1064SG register descriptions are located in Chapter 4. They are divided into several sections, and arranged in alphabetical order within each section.

- To look up a register by name when you know which section it’s in, go to that section and search the running headers at the top of the page for the register you want. (The sections are identified in ‘Contents’ at the front of the manual, on page 4-1, and within the page footers of Chapter 4.)
- If you don’t know which section it’s in, look the register up in the Index in the back.
- To look up a register by its index or address, refer to the tables in Chapter 3. Indirect access register indexes are also duplicated on the description page of the direct access register that they refer to.
- To look up a particular field within a register, look in the Alphabetical List of Register Fields near the back of the manual.

Information on how to program the MGA-1064SG registers is found in Chapter 5, while information relating to hardware design is located in Chapter 6.

At the beginning of this manual you’ll find a complete table of Contents, followed by a List of (major) Figures, and a List of (major) Tables.

Chapter 2: MGA-1064SG Overview

This chapter includes:

2.1 Introduction

The MGA-1064SG 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-1064SG chip are listed below and described in detail in the remainder of this chapter.

- PCI bus interface
- VGA graphics controller
- VGA attributes controller
- CRTC
- Video Interface
- Address Processing Unit (APU)
- Data Processing Unit (DPU)
- Texture Mapper
- Memory Controller

2.2 PCI Bus interface

This section of the MGA-1064SG 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
- Bus mastering circuitry

2.3 VGA Graphics Controller

This section of the MGA-1064SG 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.

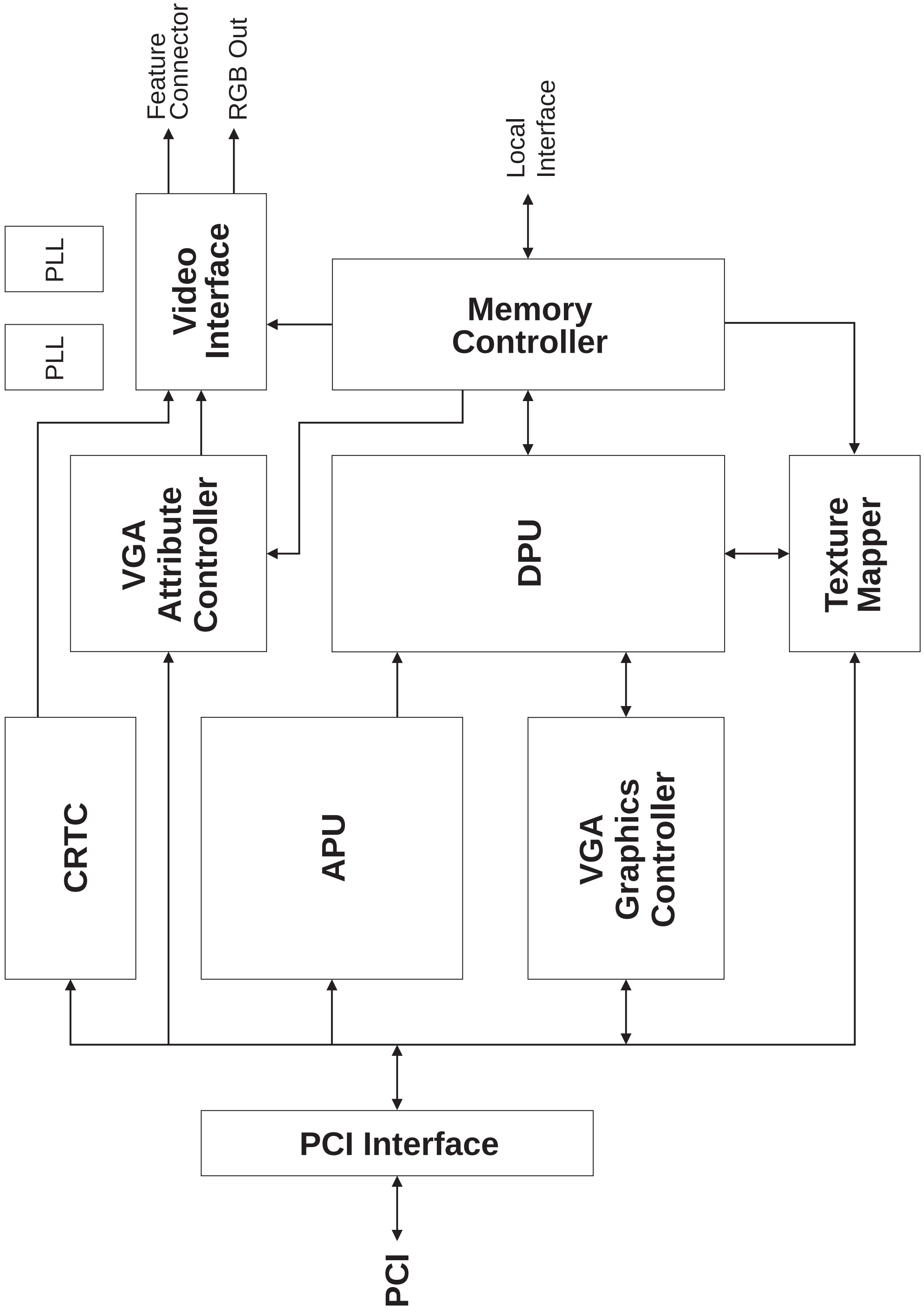
2.5 CRTC

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.

2.6 Video Interface

The video interface converts display pixels from the frame buffer into analog signals that are sent to the CRT monitor. It includes the color LUT, cursor generation, keying logic, the MAFC port, the DAC registers, the DAC, the system clock PLL, and the pixel clock PLL.

Figure 2-1: MGA-1064SG Block Diagram



2.7 Address Processing Unit (APU)

This section of the MGA-1064SG 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

2.8 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 commands to the memory controller. The DPU includes the:

- Generation of the sequences for every drawing operation
- Funnel shifter for data alignment
- Boolean ALU
- Patterning circuitry
- Color space converter
- Dithering circuitry
- Data FIFO for blit operations
- Color expansion circuitry for character drawing
- Gouraud shading generator
- Depth generation circuitry

2.9 Texture Mapper

This section implements the perspective-correct texture mapping feature of the MGA-1064SG. It includes:

- Texture parameter interpolation (STQ)
- Perspective correction circuitry
- Texel address FIFO
- Transparency circuitry
- Texture LUT
- Lighting module

2.10 Memory Controller

This section converts the read and write commands issued by internal modules into memory cycles that are sent to the frame buffer. Its functions include:

- Generation of memory cycles
- Interface to the SDRAM/SGRAM
- Arbitration of internal requests to the frame buffer
- Depth comparison circuitry
- All control circuitry for external devices

The MGA-1064SG chip can interface directly with the SDRAM/SGRAM chips. A frame buffer of up to 8 MBytes is supported.

Chapter 3: Resource Mapping

This chapter includes:

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-1064SG Configuration Space Mapping

Address	Name/Note	Description
00h-03h	**DEVID**	Device Identification
04h-07h	**DEVCTRL**	Device Control
08h-0Bh	**CLASS**	Class Code
0Ch-0Fh	**HEADER**	Header
10h-13h	**MGABASE1**	MGA Control Aperture Base
14h-17h	**MGABASE2**	MGA Frame Buffer Aperture Address
18h-1Bh	**MGABASE3**	MGA ILOAD Aperture Base Address
1Ch-2Bh	Reserved	
2Ch-2Fh	**SUBSYSID**	Subsystem ID. Writing has no effect.
30h-33h	**ROMBASE**	ROM Base Address
34h-3Bh	Reserved	
3Ch-3Fh	**INTCTRL**	Interrupt Control
40h-43h	**OPTION**	Option
44h-47h	**MGA_INDEX**	MGA Indirect Access Index
48h-4Bh	**MGA_DATA**	MGA Indirect Access Data
4Ch-4Fh	**SUBSYSID**	Subsystem ID. Reading will give 0’s.
50h-FFh	Reserved	

✍ Writing to a reserved location has no effect. Reading from a reserved location will give 0’s. Access to any location (including a reserved one) will be decoded.

3.1.2 MGA General Map

Table 3-2: MGA General Map

Address	*Condition*	*Name/Notes*
000A0000h-000BFFFFh 000A0000h-000AFFFFh 000B0000h-000B7FFFh 000B8000h-000BFFFFh	**GCTL6**<3:2> = ‘00’, **MISC**<1> = ‘1’ **GCTL6**<3:2> = ‘01’, **MISC**<1> = ‘1’ **GCTL6**<3:2> = ‘10’, **MISC**<1> = ‘1’ **GCTL6**<3:2> = ‘11’, **MISC**<1> = ‘1’	VGA frame buffer (1) (2)
ROMBASE + 0000h to **ROMBASE** + FFFFh	**biosen** = 1 (see **OPTION**) **romen** = 1 (see **ROMBASE**)	BIOS EPROM (1)
MGABASE1 + 0000h to **MGABASE1** + 3FFFh	MGA control aperture (see Table 3-3)	(1)
MGABASE2 + 000000h to **MGABASE2** + 7FFFFFh	Direct frame buffer access aperture	(1)(2)(3)
MGABASE3 + 000000h to **MGABASE3** + 7FFFFFh	8 MByte Pseudo-DMA window	(1)(4)

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

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

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

(4) Hardware swapping for big endian support is performed in accordance with the settings of the **OPMODE** register’s **dmaDataSiz** bits.

3.1.3 MGA Control Aperture

Table 3-3: MGA Control Aperture (extension of Table 3-2)

MGABASE1 +	*Attr.*	*Mnemonic*	*Device name*
0000h-1BFFh	W	DMAWIN (ILOAD)	7KByte Pseudo-DMA window (1)
	R	DMAWIN (IDUMP)	7KByte Pseudo-DMA window (1)
1C00h-1DFFh	W	DWGREG0	First drawing registers (2)(3)(4)
1E00h-1EFFh	R/W	HSTREG	Host registers (2)(3)
1F00h-1FFFh	R/W	VGAREG	VGA registers (5)(3)
2000h-2BFFh		------	Reserved (6)
2C00h-2DFFh	W	DWGREG1	Second drawing registers(2)(3)(4)
2E00h-3BFFh		------	Reserved (6)
3C00h-3C0Fh	R/W	DAC	DAC(3)
3C10h-3DFFh		------	Reserved (6)
3E00h-3FFFh	R/W	EXPDEV	Expansion (7)

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

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

(3) See the register map in Table 3-4 for a more detailed view of this memory space

(4) Reads of these locations are not decoded.

(5) 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.

(6) Reserved locations are decoded. The returned values are unknown.

(7) The exact mapping within this range depends on the external connections and on the external devices used.

3.2 Register Mapping

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.

Table 3-4: Register Map (Part 1 of 9)

Register Mnemonic Name	*Access*	*Memory Address(1)*	*I/O Address(2)*	*Index*	*Description/Comments*	*Page*
DWGCTL (3)	WO	1C00h(4)	-	-	Drawing Control	4-49
MACCESS (3)	WO	1C04h(4)	-	-	Memory Access	4-64
MCTLWTST	WO	1C08h(4)	-	-	Memory Control Wait State	4-65
ZORG	WO	1C0Ch(4)	-	-	Z-Depth Origin	4-84
PAT0	WO	1C10h(4)	-	-	Pattern	4-67
PAT1	WO	1C14h(4)	-	-	Pattern	4-67
-	WO	1C18h(4)	-	-	Reserved	-
PLNWT (3)	WO	1C1Ch(4)	-	-	Plane Write Mask	4-69
BCOL	WO	1C20h(4)	-	-	Background Color / Blit Color Mask	4-27
FCOL	WO	1C24h(4)	-	-	Foreground Color / Blit Color Key	4-56
-	WO	1C28h(4)	-	-	Reserved	-
-	WO	1C2Ch(4)	-	-	Reserved (SRCBLT)	-
SRC0	WO	1C30h(4)	-	-	Source	4-73
SRC1	WO	1C34h(4)	-	-	Source	4-73
SRC2	WO	1C38h(4)	-	-	Source	4-73
SRC3	WO	1C3Ch(4)	-	-	Source	4-73
XYSTRT (5)	WO	1C40h(4)	-	-	XY Start Address	4-78
XYEND (5)	WO	1C44h(4)	-	-	XY End Address	4-77
-	1C48h-1C4Ch(4)		-	-	Reserved	-
SHIFT (5)	WO	1C50h(4)	-	-	Funnel Shifter Control	4-72
DMAPAD (5)	WO	1C54h(4)	-	-	DMA Padding	4-35
SGN (5)	WO	1C58h(4)	-	-	Sign	4-71
LEN (5)	WO	1C5Ch(4)	-	-	Length	4-63
AR0 (5)	WO	1C60h(4)	-	-	Multi-Purpose Address 0	4-20
AR1 (5)	WO	1C64h(4)	-	-	Multi-Purpose Address 1	4-21
AR2 (5)	WO	1C68h(4)	-	-	Multi-Purpose Address 2	4-22
AR3 (5)	WO	1C6Ch(4)	-	-	Multi-Purpose Address 3	4-23
AR4 (5)	WO	1C70h(4)	-	-	Multi-Purpose Address 4	4-24

Table 3-4: Register Map (Part 2 of 9)

Register Mnemonic Name	Access	Memory Address(1)	I/O Address(2)	Index	Description/Comments	Page
AR5 (5)	WO	1C74h(4)	-	-	Multi-Purpose Address 5	4-25
AR6 (5)	WO	1C78h(4)	-	-	Multi-Purpose Address 6	4-26
-	WO	1C7Ch(4)	-	-	Reserved	-
CXBNDRY (5)	WO	1C80h(4)	-	-	Clipper X Boundary	4-28
FXBNDRY (5)	WO	1C84h(4)	-	-	X Address (Boundary)	4-58
YDSTLEN (5)	WO	1C88h(4)	-	-	Y Destination and Length	4-81
PITCH (5)	WO	1C8Ch(4)	-	-	Memory Pitch	4-68
YDST (5)	WO	1C90h(4)	-	-	Y Address	4-80
YDSTORG (5)	WO	1C94h(4)	-	-	Memory Origin	4-82
YTOP (5)	WO	1C98h(4)	-	-	Clipper Y Top Boundary	4-83
YBOT (5)	WO	1C9Ch(4)	-	-	Clipper Y Bottom Boundary	4-79
CXLEFT (5)	WO	1CA0h(4)	-	-	Clipper X Minimum Boundary	4-29
CXRIGHT (5)	WO	1CA4h(4)	-	-	Clipper X Maximum Boundary	4-30
FXLEFT (5)	WO	1CA8h(4)	-	-	X Address (Left)	4-59
FXRIGHT (5)	WO	1CACh(4)	-	-	X Address (Right)	4-60
XDST (5)	WO	1CB0h(4)	-	-	X Destination Address	4-76
-	1CB4h-1CBCh(4)		-	-	Reserved	-
DR0	WO	1CC0h(4)	-	-	Data ALU 0	4-36
-	WO	1CC4h(4)	-	-	Reserved (DR1)	-
DR2	WO	1CC8h(4)	-	-	Data ALU 2	4-37
DR3	WO	1CCCh(4)	-	-	Data ALU 3	4-38
DR4	WO	1CD0h(4)	-	-	Data ALU 4	4-39
-	WO	1CD4h(4)	-	-	Reserved (DR5)	-
DR6	WO	1CD8h(4)	-	-	Data ALU 6	4-40
DR7	WO	1CDCh(4)	-	-	Data ALU 7	4-41
DR8	WO	1CE0h(4)	-	-	Data ALU 8	4-42
	WO	1CE4h(4)	-	-	Reserved (DR9)	-
DR10	WO	1CE8h(4)	-	-	Data ALU 10	4-43
DR11	WO	1CECh(4)	-	-	Data ALU 11	4-44
DR12	WO	1CF0h(4)	-	-	Data ALU 12	4-45
-	WO	1CF4h(4)	-	-	Reserved (DR13)	-

Table 3-4: Register Map (Part 3 of 9)

Register Mnemonic Name	*Access*	*Memory Address(1)*	*I/O Address(2)*	*Index*	*Description/Comments*	*Page*
DR14	WO	1CF8h(4)	-	-	Data ALU 14	4-46
DR15	WO	1CFCh(4)	-	-	Data ALU 15	4-47
-		1D00h-1DFFh(4)	-	-	Same mapping as 1C00h-1CFCh (6)	-
-		1E00h - 1E0Fh	-	-	Reserved	-
FIFOSTATUS	RO	1E10h	-	-	Bus FIFO Status	4-57
STATUS	RO	1E14h	-	-	Status	4-74
ICLEAR	WO	1E18h	-	-	Interrupt Clear	4-61
IEN	R/W	1E1Ch	-	-	Interrupt Enable	4-62
VCOUNT	RO	1E20h	-	-	Vertical Count	4-75
-		1E24h - 1E2Fh	-	-	Reserved	-
DMAMAP30	R/W	1E30h	-	-	DMA Map 3h to 0h	4-31
DMAMAP74	R/W	1E34h	-	-	DMA Map 7h to 4h	4-32
DMAMAPB8	R/W	1E38h	-	-	DMA Map Bh to 8h	4-33
DMAMAPFC	R/W	1E3Ch	-	-	DMA Map Fh to Ch	4-34
RST	R/W	1E40h	-	-	Reset	4-70
-		1E44h - 1E53h	-	-	Reserved	-
OPMODE	R/W	1E54h	-	-	Operating Mode	4-66
-		1E60h - 1E7Fh	-	-	Reserved	-
DWG_INDIR_WT<0>	WO	1E80h(4)	-	-	Drawing Register Indirect Write 0	4-48
DWG_INDIR_WT<1>	WO	1E84h(4)	-	-	Drawing Register Indirect Write 1	4-48
DWG_INDIR_WT<2>	WO	1E88h(4)	-	-	Drawing Register Indirect Write 2	4-48
DWG_INDIR_WT<3>	WO	1E8Ch(4)	-	-	Drawing Register Indirect Write 3	4-48
DWG_INDIR_WT<4>	WO	1E90h(4)	-	-	Drawing Register Indirect Write 4	4-48
DWG_INDIR_WT<5>	WO	1E94h(4)	-	-	Drawing Register Indirect Write 5	4-48
DWG_INDIR_WT<6>	WO	1E98h(4)	-	-	Drawing Register Indirect Write 6	4-48
DWG_INDIR_WT<7>	WO	1E9Ch(4)	-	-	Drawing Register Indirect Write 7	4-48
DWG_INDIR_WT<8>	WO	1EA0h(4)	-	-	Drawing Register Indirect Write 8	4-48
DWG_INDIR_WT<9>	WO	1EA4h(4)	-	-	Drawing Register Indirect Write 9	4-48
DWG_INDIR_WT<10>	WO	1EA8h(4)	-	-	Drawing Register Indirect Write 10	4-48
DWG_INDIR_WT<11>	WO	1EACh(4)	-	-	Drawing Register Indirect Write 11	4-48
DWG_INDIR_WT<12>	WO	1EB0h(4)	-	-	Drawing Register Indirect Write 12	4-48
DWG_INDIR_WT<13>	WO	1EB4h(4)	-	-	Drawing Register Indirect Write 13	4-48
DWG_INDIR_WT<14>	WO	1EB8h(4)	-	-	Drawing Register Indirect Write 14	4-48

Table 3-4: Register Map (Part 4 of 9)

Register Mnemonic Name	*Access*	*Memory Address[(1)]*	*I/O Address[(2)]*	*Index*	*Description/Comments*	*Page*
DWG_INDIR_WT<15>	WO	1EBCh(4)	-	-	Drawing Register Indirect Write 15	4-48
-		1EC0h - 1FBFh	-	-	Reserved	-
ATTR (Index)	R/W	1FC0h	3C0h	-	Attribute Controller	4-86
ATTR (Data)	WO	1FC0h	3C0h	-	Attribute Controller	
ATTR (Data)	RO	1FC1h	3C1h	-	Attribute Controller	-
-	WO	1FC1h	3C1h	-	Reserved	-
ATTR0	R/W	-	-	00h	Palette entry 0	4-88
ATTR1	R/W	-	-	01h	Palette entry 1	4-88
ATTR2	R/W	-	-	02h	Palette entry 2	4-88
ATTR3	R/W	-	-	03h	Palette entry 3	4-88
ATTR4	R/W	-	-	04h	Palette entry 4	4-88
ATTR5	R/W	-	-	05h	Palette entry 5	4-88
ATTR6	R/W	-	-	06h	Palette entry 6	4-88
ATTR7	R/W	-	-	07h	Palette entry 7	4-88
ATTR8	R/W	-	-	08h	Palette entry 8	4-88
ATTR9	R/W	-	-	09h	Palette entry 9	4-88
ATTRA	R/W	-	-	0Ah	Palette entry A	4-88
ATTRB	R/W	-	-	0Bh	Palette entry B	4-88
ATTRC	R/W	-	-	0Ch	Palette entry C	4-88
ATTRD	R/W	-	-	0Dh	Palette entry D	4-88
ATTRE	R/W	-	-	0Eh	Palette entry E	4-88
ATTRF	R/W	-	-	0Fh	Palette entry F	4-88
ATTR10	R/W	-	-	10h	Attribute Mode Control	4-89
ATTR11	R/W	-	-	11h	Overscan Color	4-91
ATTR12	R/W	-	-	12h	Color Plane Enable	4-92
ATTR13	R/W	-	-	13h	Horizontal Pel Panning	4-93
ATTR14	R/W	-	-	14h	Color Select	4-94
-	-	-	-	15h - 1Fh: Reserved		-
INSTS0	RO	1FC2h	3C2h	-	Input Status 0	4-149
MISC	WO	1FC2h	3C2h	-	Miscellaneous Output	4-151
-	R/W	1FC3h	3C3h(7)	-	Reserved, not decoded for I/O	
SEQ (Index)	R/W	1FC4h	3C4h	-	Sequencer	4-153
SEQ (Data)	R/W	1FC5h	3C5h	-	Sequencer	-
SEQ0	R/W	-	-	00h	Reset	4-154
SEQ1	R/W	-	-	01h	Clocking Mode	4-155
SEQ2	R/W	-	-	02h	Map Mask	4-156

Table 3-4: Register Map (Part 5 of 9)

Register Mnemonic Name	*Access*	*Memory Address$^{(1)}$*	*I/O Address$^{(2)}$*	*Index*	*Description/Comments*	*Page*
SEQ3	R/W	-	-	03h	Character Map Select	4-157
SEQ4	R/W	-	-	04h	Memory Mode	4-158
-	R/W	-	-	05h - 07h: Reserved		-
-	-	1FC6h	-	-	Reserved	-
DACSTAT	RO	1FC7h	3C7h	-	DAC Status (requires a byte access)	4-136
-	WO	1FC7h	-	-	Reserved	-
-	1FC8h-1FC9h		-	-	Reserved	-
FEAT	RO	1FCAh	3CAh	-	Feature Control	4-137
	WO	1FCAh	3CAh	-	Reserved	-
-	-	1FCBh	3CBh$^{(7)}$	-	Reserved, not decoded for I/O	-
MISC	RO	1FCCh	3CCh	-	Miscellaneous Output	4-151
-	WO	1FCCh	3CCh	-	Reserved	-
-	-	1FCDh	3CDh$^{(7)}$	-	Reserved, not decoded for I/O	-
GCTL (Index)	R/W	1FCEh	3CEh	-	Graphic Controller	4-138
GCTL (Data)	R/W	1FCFh	3CFh	-	Graphic Controller	-
GCTL0	R/W	-	-	00h	Set/Reset	4-139
GCTL1	R/W	-	-	01h	Enable Set/Reset	4-140
GCTL2	R/W	-	-	02h	Color Compare	4-141
GCTL3	R/W	-	-	03h	Data Rotate	4-142
GCTL4	R/W	-	-	04h	Read Map Select	4-143
GCTL5	R/W	-	-	05h	Graphics Mode	4-144
GCTL6	R/W	-	-	06h	Miscellaneous	4-146
GCTL7	R/W	-	-	07h	Color Don’t Care	4-147
GCTL8	R/W	-	-	08h	Bit Mask	4-148
-	-	-	-	09h - 0Fh: Reserved		-
-	1FD0h-1FD3h		-	-	Reserved	-
CRTC (Index)	R/W	1FD4h	3D4h	-	CRTC Registers (or 3B4h$^{(8)}$)	4-96
CRTC (Data)	R/W	1FD5h	3D5h	-	CRTC Registers (or 3B5h$^{(8)}$)	-
CRTC0	R/W	-	-	00h	Horizontal Total	4-98
CRTC1	R/W	-	-	01h	Horizontal Display Enable End	4-99
CRTC2	R/W	-	-	02h	Start Horizontal Blanking	4-100
CRTC3	R/W	-	-	03h	End Horizontal Blanking	4-101
CRTC4	R/W	-	-	04h	Start Horizontal Retrace Pulse	4-102
CRTC5	R/W	-	-	05h	End Horizontal Retrace	4-103
CRTC6	R/W	-	-	06h	Vertical Total	4-104
CRTC7	R/W	-	-	07h	Overflow	4-105

Table 3-4: Register Map (Part 6 of 9)

Register Mnemonic Name	Access	Memory Address(1)	I/O Address(2)	Index	Description/Comments	Page
CRTC8	R/W	-	-	08h	Preset Row Scan	4-106
CRTC9	R/W	-	-	09h	Maximum Scan Line	4-107
CRTCA	R/W	-	-	0Ah	Cursor Start	4-108
CRTCB	R/W	-	-	0Bh	Cursor End	4-109
CRTCC	R/W	-	-	0Ch	Start Address High	4-110
CRTCD	R/W	-	-	0Dh	Start Address Low	4-111
CRTCE	R/W	-	-	0Eh	Cursor Location High	4-112
CRTCF	R/W	-	-	0Fh	Cursor Location Low	4-113
CRTC10	R/W	-	-	10h	Vertical Retrace Start	4-114
CRTC11	R/W	-	-	11h	Vertical Retrace End	4-115
CRTC12	R/W	-	-	12h	Vertical Display Enable End	4-116
CRTC13	R/W	-	-	13h	Offset	4-117
CRTC14	R/W	-	-	14h	Underline Location	4-118
CRTC15	R/W	-	-	15h	Start Vertical Blank	4-119
CRTC16	R/W	-	-	16h	End Vertical Blank	4-120
CRTC17	R/W	-	-	17h	CRTC Mode Control	4-121
CRTC18	R/W	-	-	18h	Line Compare	4-125
-	-	-	-	19h - 21h: Reserved		-
CRTC22	R/W	-	-	22h	CPU Read Latch	4-126
-	-	-	-	23h	Reserved	-
CRTC24	R/W	-	-	24h	Attributes Address/Data Select	4-127
-	-	-	-	25h	Reserved	-
CRTC26	R/W	-	-	26h	Attributes Address	4-128
-	-	-	-	27h - 3Fh: Reserved		-
-	-	1FD6h	3D6h(7)	-	Reserved, not decoded for I/O (or 3B6h(8))	-
-	-	1FD7h	3D7h(7)	-	Reserved, not decoded for I/O (or 3B7h(8))	-
-	1FD8h-1FD9h		-	-	Reserved	-
INSTS1	RO	1FDAh	3DAh	-	Input Status 1 (or 3BAh(8))	4-150
FEAT	WO	1FDAh	3DAh	-	Feature Control (or 3BAh(8))	4-137
-	-	1FDBh	3DBh(7)	-	Reserved, not decoded for I/O (or 3BBh(8))	-
-	1FDCh-1FDDh		-	-	Reserved	-
CRTCEXT (Index)	R/W	1FDEh	3DEh	-	CRTC Extension	4-129
CRTCEXT (Data)	R/W	1FDFh	3DFh	-	CRTC Extension	-
CRTCEXT0	R/W	-	-	00h	Address Generator Extensions	4-130

Table 3-4: Register Map (Part 7 of 9)

Register Mnemonic Name	*Access*	*Memory Address(1)*	*I/O Address(2)*	*Index*	*Description/Comments*	*Page*
CRTCEXT1	R/W	-	-	01h	Horizontal Counter Extensions	4-131
CRTCEXT2	R/W	-	-	02h	Vertical Counter Extensions	4-132
CRTCEXT3	R/W	-	-	03h	Miscellaneous	4-133
CRTCEXT4	R/W	-	-	04h	Memory Page	4-134
CRTCEXT5	R/W	-	-	05h	Horizontal Video Half Count	4-135
-	1FE0h - 1FFEh		-	-	Reserved	-
CACHEFLUSH	R/W	1FFFh	-	-	Cache Flush	4-95
-	2C4Ch-2CFFh(4)		-	-	Reserved	-
-	2D00h-2DFFh(4)		-	-	Same mapping as 2C00-2CFC (6)	-
PALWTADD	R/W	3C00h	3C8h	-	Palette RAM Addr. Write/Load Index	4-163
PALDATA	R/W	3C01h	3C9h	-	Palette RAM Data Register	4-161
PIXRDMSK	R/W	3C02h	3C6h	-	Pixel Read Mask	4-164
PALRDADD	R/W	3C03h	3C7h	-	Palette RAM Address - Read. This register is WO for I/O accesses.	4-162
-	3C04h - 3C09h		-	-	Reserved	-
X_DATAREG	R/W	3C0Ah	-	-	Indexed Data Register	4-165
-	-	-	-	00h - 03h: Reserved		-
XCURADDL	R/W	-	-	04h	Cursor Base Address, Low	4-175
XCURADDH	R/W	-	-	05h	Cursor Base Address, High	4-174
XCURCTRL	R/W	-	-	06h	Cursor Control	4-177
-	-	-	-	07h	Reserved	-
XCURCOL0RED	R/W	-	-	08h	Cursor color 0 Red	4-176
XCURCOL0GREEN	R/W	-	-	09h	Cursor color 0 Green	4-176
XCURCOL0BLUE	R/W	-	-	0Ah	Cursor color 0 Blue	4-176
-	-	-	-	0Bh	Reserved	-
XCURCOL1RED	R/W	-	-	0Ch	Cursor color 1 Red	4-176
XCURCOL1GREEN	R/W	-	-	0Dh	Cursor color 1 Green	4-176
XCURCOL1BLUE	R/W	-	-	0Eh	Cursor Color 1 Blue	4-176
-	-	-	-	0Fh	Reserved	-
XCURCOL2RED	R/W	-	-	10h	Cursor Color 2 Red	4-176
XCURCOL2GREEN	R/W	-	-	11h	Cursor Color 2 Green	4-176
XCURCOL2BLUE	R/W	-	-	12h	Cursor Color 2 Blue	4-176
-	-	-	-	13h - 17h: Reserved		-
XVREFCTRL	R/W			18h	Voltage Reference Control	4-193
XMULCTRL	R/W	-	-	19h	Multiplex Control	4-182
XPIXCLKCTRL	R/W	-	-	1Ah	Pixel Clock Control	4-183

Table 3-4: Register Map (Part 8 of 9)

Register Mnemonic Name	*Access*	*Memory Address(1)*	*I/O Address(2)*	*Index*	*Description/Comments*	*Page*
-	-	-	-	1Bh - 1Ch: Reserved		-
XGENCTRL	R/W	-	-	1Dh	General Control	4-178
XMISCCTRL	R/W	-	-	1Eh	Miscellaneous Control	4-181
-	-	-	-	1Fh - 29h: Reserved		-
XGENIOCTRL	R/W	-	-	2Ah	General Purpose I/O Control	4-179
XGENIODATA	R/W	-	-	2Bh	General Purpose I/O Data	4-180
XSYSPLLM	R/W	-	-	2Ch	SYSPLL M Value Register	4-189
XSYSPLLN	R/W	-	-	2Dh	SYSPLL N Value Register	4-190
XSYSPLLP	R/W	-	-	2Eh	SYSPLL P Value Register	4-191
XSYSPLLSTAT	RO	-	-	2Fh	SYSPLL Status	4-192
-	-	-	-	30h - 37h: Reserved		-
XZOOMCTRL	R/W	-	-	38h	Zoom Control	4-194
-	-	-	-	39h	Reserved	-
XSENSETEST	R/W	-	-	3Ah	Sense Test	4-188
-	-	-	-	3Bh	Reserved	-
XCRCREML	RO	-	-	3Ch	CRC Remainder Low	4-173
XCRCREMH	RO	-	-	3Dh	CRC Remainder High	4-172
XCRCBITSEL	R/W	-	-	3Eh	CRC Bit Select	4-171
-	-	-	-	3Fh	Reserved	
XCOLKEYMSKL	R/W	-	-	40h	Color Key Mask, Low	4-170
XCOLKEYMSKH	R/W	-	-	41h	Color Key Mask, High	4-169
XCOLKEYL	R/W	-	-	42h	Color Key, Low	4-168
XCOLKEYH	R/W	-	-	43h	Color Key, High	4-167
XPIXPLLAM	R/W	-	-	44h	PIXPLL M Value Register Set A	4-184
XPIXPLLAN	R/W	-	-	45h	PIXPLL N Value Register Set A	4-185
XPIXPLLAP	R/W	-	-	46h	PIXPLL P Value Register Set A	4-186
-	-	-	-	47h	Reserved	-
XPIXPLLBM	R/W	-	-	48h	PIXPLL M Value Register Set B	4-184
XPIXPLLBN	R/W	-	-	49h	PIXPLL N Value Register Set B	4-185
XPIXPLLBP	R/W	-	-	4Ah	PIXPLL P Value Register Set B	4-186
-	-	-	-	4Bh	Reserved	-

Table 3-4: Register Map (Part 9 of 9)

Register Mnemonic Name	Access	Memory Address[1]	I/O Address[2]	Index	Description/Comments	Page
XPIXPLLCM	R/W	-	-	4Ch	PIXPLL M Value Register Set C	4-184
XPIXPLLCN	R/W	-	-	4Dh	PIXPLL N Value Register Set C	4-185
XPIXPLLCP	R/W	-	-	4Eh	PIXPLL P Value Register Set C	4-186
XPIXPLLSTAT	RO	-	-	4Fh	PIXPLL Status	4-187
-	-	-	-	50h - FFh: Reserved		-
-	-	3C0Bh	-	-	Reserved	-
CURPOSXL	R/W	3C0Ch	-	-	Cursor Position X LSB	4-160
CURPOSXH	R/W	3C0Dh	-	-	Cursor Position X MSB	4-160
CURPOSYL	R/W	3C0Eh	-	-	Cursor Position Y LSB	4-160
CURPOSYH	R/W	3C0Fh	-	-	Cursor Position Y MSB	4-160

(1) The Memory Address for the direct access registers is a byte address offset from **MGABASE1**.

(2) 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).

(3) The memory controller may become idle after the data processor; therefore, we recommend that all other drawing registers be initialized before these registers in order to maximize performance.

(4) Reads of these locations are not decoded.

(5) Since the address processor can become idle before the data processor, we recommend that you initialize these registers first, in order to take advantage of this idle time.

(6) Accessing a register in this range instructs the drawing engine to start a drawing cycle.

(7) 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.)

(8) VGA I/O addresses in the 3DXh range are for CGA emulation (the **MISC**<0> register (**ioaddsel** field) is ‘1’). VGA I/O addresses in the 3BXh range are for monochrome (MDA) emulation (the **ioaddsel** field is ‘0’). Exception: for **CRTCEXT**, the 3BEh and 3BFh I/O addresses are reserved, not decoded.

Chapter 4: Register Descriptions

This chapter includes:

Note: The registers within each section (and sub-section, for the Power graphic mode registers) of this chapter are arranged in alphabetical order of mnemonic name. For more tips on finding registers, refer to ‘Locating Information’ on page 1-7.

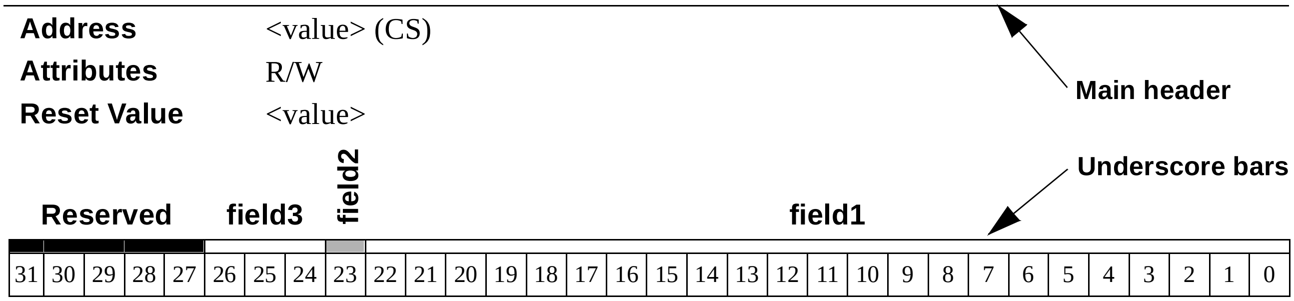
4.1 Power Graphic Mode Register Descriptions

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 memory 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.

Sample Power Graphic Mode Config. Space Register **SAMPLE_CS**

Address	<value> (CS)
Attributes	R/W
Reset Value	<value>



field1 <22:0>	Field 1. Detailed description of the **field1** field of the **SAMPLE_CS** 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_CS**, which is bit 23.
field3 <26:24>	Field 3. Detailed description of the **field3** field of the **SAMPLE_CS** register, which comprises bits 26 to 24.
Reserved <31:27>	Reserved. When writing to this register, the bits in this field must be set to ‘0’. (Reserved registers always appear at the end of a register description.)

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

Here are some of the symbols that appear as part of a register’s reset value:

- `000? 0000 000S ???? 1101 0000 S000 0000`b
  (b = binary, ? = unknown, S = bit’s reset value is affected by a strap setting, N/A = not applicable)

Class Code CLASS

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

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 (00h).

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 vgaboot strap, which is sampled on hard reset.

vgaboot strap	*Value*	*Meaning*
‘0’	038000h	Non-Super VGA display controller
‘1’	030000h	Super VGA compatible controller

The sampled state of the vgaboot strap (pin VD<13>, described on page A-4) can be read through this register.

Address 04h (CS)

Attributes R/W, BYTE/WORD/DWORD, STATIC

Reset Value `0000 0010 1000 0000 0000 0000 1000 0000b`

detparerr	sigsyserr	recmastab	rectargab	sigtargab	devseltim		Reserved	fastbackcap	udfsup	cap66Mhz	Reserved												SERRenable	waitcycle	resparerr	vgasnoop	memwrien	specialcycle	busmaster	memspace	iospace
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

iospace
R/W <0>

I/O space. Controls device response to I/O SPACE accesses (VGA registers).

- 0: disable the device response
- 1: enable the device response

memspace
R/W <1>

Memory space. Controls device response to memory accesses (EPROM, VGA frame buffer, MGA control aperture, MGA direct access aperture, and 8 MByte Pseudo-DMA window).

- 0: disable the device response
- 1: enable the device response

busmaster
R/W <2>

Bus master. Controls a device’s ability to act as a master on the PCI bus (used to access system memory):

- 0: prevents the device from generating PCI accesses
- 1: allows the device to behave as a bus master

specialcycle
RO <3>

The hard coded ‘0’ indicates that the MGA will not respond to a special cycle.

memwrien
RO <4>

The hard coded ‘0’ indicates that an MGA acting as a bus master will never generate the write and invalidate command.

vgasnoop
R/W <5>

Controls how the chip handles I/O accesses to the VGA DAC locations.
The **vgasnoop** field is only used when **vgaioen** (see **OPTION on page 4-14)** is ‘1’.

- ‘0’: The chip will reply to read and write accesses at VGA locations 3C6h, 3C7h, 3C8h, and 3C9h.
- ‘1’: The chip will snoop writes to VGA DAC locations. It will not assert PTRDY/, PSTOP/, and PDEVSEL/, but will internally decode the access and program the on-board DAC. In situations where the chip is not ready to snoop the access, it will acknowledge the cycle by asserting PDEVSEL/, and force a retry cycle by asserting PSTOP/. Read accesses to VGA DAC locations are not affected by **vgasnoop**.

resparerr
RO <6>

The hard coded ‘0’ indicates that the MGA will not detect and signal parity errors (MGA does generate parity information as per the PCI specification requirement). Writing has no effect.

waitcycle **RO <7>**	This bit reads as ‘1’, indicating that continuous address/data stepping is performed for read accesses in the target (data stepping) and the master (address stepping). Stepping lasts one pclk. Writing has no effect.
SERRenable **RO <8>**	This hard coded ‘0’ indicates that MGA does not generate SERR interrupts. Writing has no effect.
cap66Mhz **RO <21>**	The hard coded ‘0’ indicates that the MGA is running at 33 MHz or lower clock rates.
udfsup **RO <22>**	The hard coded ‘0’ indicates that the MGA does not support user-definable features.
fastbackcap **RO <23>**	The hard coded ‘1’ indicates that the MGA supports fast back-to-back transactions when part of the transaction targets a different agent. Writing has no effect.
devseltim **RO <26:25>**	Device select timing. Specifies the timing of devsel. It is read as ‘01’.
sigtargab **R/W <27>**	Signaled target abort. Set to ‘1’ when the MGA terminates a transaction in target mode with target-abort. This bit is cleared to ‘0’ when written with ‘1’.
rectargab **R/W <28>**	Received target abort. Set to ‘1’ when the MGA is a master and a transaction is terminated with target-abort. This bit is cleared to ‘0’ when written with ‘1’.
recmastab **R/W <29>**	Received master abort. Set to ‘1’ when a transaction is terminated with master-abort by the MGA. This bit is cleared to ‘0’ when written with ‘1’.
sigsyserr **RO <30>**	MGA does not assert SERR/. Writing has no effect. Reading will give ‘0’s.
detparerr **RO <31>**	MGA does not detect parity errors. Writing has no effect. Reading will give ‘0’s.
Reserved:	**<20:9> <24>** Reserved. When writing to this register, the bits in these fields must be set to ‘0’. Reading will give ‘0’s.

DEVID Device Identification

Address	00h (CS)
Attributes	RO, BYTE/WORD/DWORD, STATIC
Reset Value	`0000 0101 0001 1010 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 <15:0>	This field contains the Matrox manufacturer identifier for PCI: 102Bh.
device <31:16>	This field contains the Matrox device identifier for this product: 051Ah.

Header HEADER

Address 0Ch (CS)

Attributes R/W, BYTE/WORD/DWORD, STATIC

Reset Value `0000 0000 0000 0000 0000 0000 0000 0000b`

Reserved								**header**								**latentim**								**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

latentim
R/W <15:11>
RO <10:8>

Value of the latency timer in PCI clocks. The count starts when PFRAME/ is asserted. Once the count expires, the master must initiate transaction termination as soon as its PGNT/ signal is removed.

header
RO <23:16>

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

Reserved: **<7:0> <31:24>**

Reserved. When writing to this register, the bits in these fields must be set to ‘0’. 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 1111b`

maxlat								**mingnt**								**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).

mingnt
RO <23:16>

This field specifies the PCI device’s required burst length, assuming a clock rate of 33 MHz.

Values of ‘0’ indicate that the PCI device (the MGA-1064SG board) has no major requirements for setting the latency timer.

maxlat
RO <31:24>

This field specifies how often the PCI device must gain access to the PCI bus.

Values of ‘0’ indicate that the PCI device (the MGA-1064SG board) has no major requirements for setting the latency timer.

Address 48h (CS)

Attributes R/W, BYTE/WORD/DWORD, STATIC

Reset Value 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**.

The **MGA_INDEX** and **MGA_DATA** registers cannot be used in Pseudo-DMA mode (see page 5-25).

MGA_INDEX MGA Indirect Access Index

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

Reserved																		**index**												**Res.**	
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).

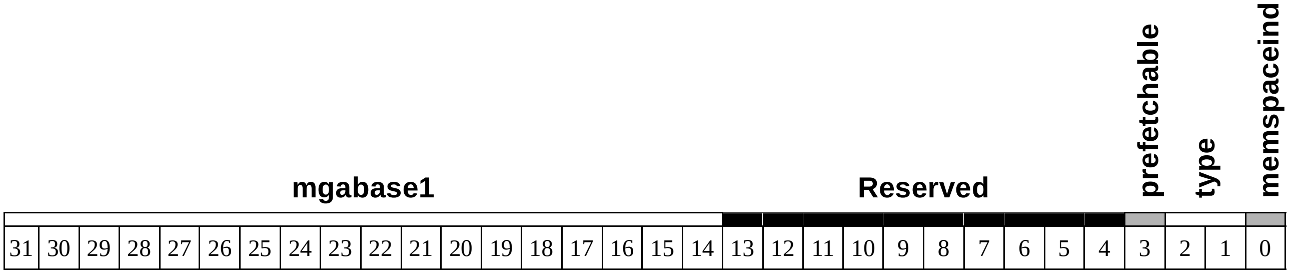
Reserved **<1:0> <31:14>**

Reserved. When writing to this register, the bits in this field must be set to ‘0’. Reading will give ‘0’s.

The **MGA_INDEX** and **MGA_DATA** registers cannot be used in Pseudo-DMA mode (see page 5-25).

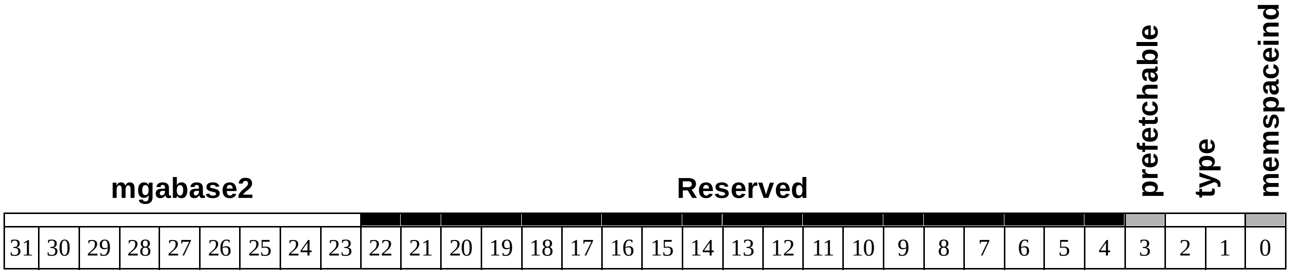
MGA Control Aperture Base MGABASE1

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



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.
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. 8 MByte Pseudo-DMA window 4. VGA frame buffer aperture 5. MGA frame buffer aperture (lowest precedence)
Reserved <13:4>	Reserved. When writing to this register, the bits in this field must be set to ‘0’. Reading will give ‘0’s.

Address	14h (CS)
Attributes	R/W, BYTE/WORD/DWORD, STATIC
Reset Value	`0000 0000 0000 0000 0000 0000 0000 S000b`



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>

A ‘1’ indicates that this space can be prefetchable (better system performance can be achieved when the bridge enables prefetching into that range).

The state of this field depends on the unimem strap, as shown below:

unimem	prefetchable
‘0’	‘1’
‘1’	‘0’

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. 8 MByte Pseudo-DMA window
4. VGA frame buffer aperture
5. MGA frame buffer aperture (lowest precedence)

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

Reserved <22:4>

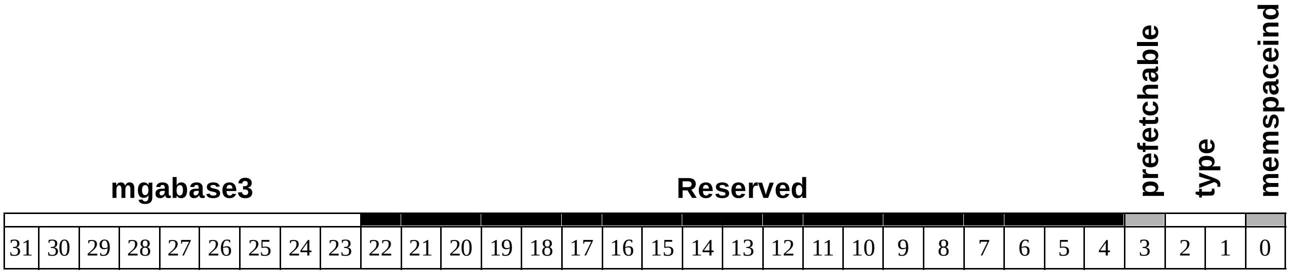
Reserved. When writing to this register, the bits in this field must be set to ‘0’. Reading will give ‘0’s.

MGA ILOAD Aperture Base Address

Address 18h (CS)

Attributes R/W, BYTE/WORD/DWORD, STATIC

Reset Value `0000 0000 0000 0000 0000 0000 0000 0000b`

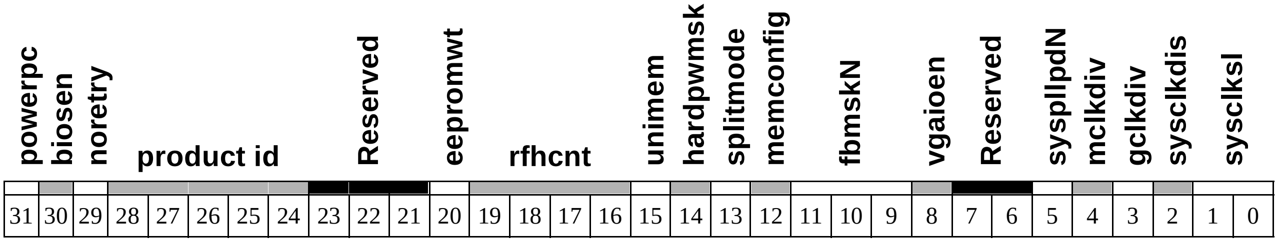


memspace ind RO <0>	The hard coded ‘0’ indicates that the map is in the memory space.
type RO <2:1>	The hard coded ‘00’ instructs the configuration program to locate the aperture anywhere within the 32-bit address space.
prefetchable RO <3>	The hard coded ‘0’ indicates that this space cannot be prefetchable.
mgabase3 <31:23>	Specifies the base address of the 8 MByte Pseudo-DMA window. 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. 8 MByte Pseudo-DMA window 4. VGA frame buffer aperture 5. MGA frame buffer aperture (lowest precedence)
Reserved <22:4>	Reserved. When writing to this register, the bits in this field must be set to ‘0’. Reading will give ‘0’s.

Address 40h (CS)

Attributes R/W, BYTE/WORD/DWORD, STATIC

Reset Value `0S0S SSSS 0000 0000 S000 000S 0000 0000b`



sysclksl <1:0> System clock selection. These bits select the source of the system clock:

- 00: select the PCI clock
- 01: select the output of the system clock PLL
- 10: selects an external source from the MCLK pin (permitted only if MCLK has been configured as an input)
- 11: Reserved

sysclkdis <2> System clock disable. This bit controls the system clock output:

- 0: enable system clock oscillations
- 1: disable system clock oscillations

gclkdiv <3> Graphics clock divider select. Selects the ratio by which the system clock is divided in order to produce the graphics clock when **sysclksl** = ‘01’.

- 0: divide by 3
- 1: divide by 1

mclkdiv <4> Memory clock divider select. Selects the ratio by which the system clock is divided in order to produce the memory clock when **sysclksl** = ‘01’.

- 0: divide by 2
- 1: divide by 1

syspllpdN <5> System PLL power down.

- 0: power down
- 1: power up

vgaioen <8> VGA I/O map enable.

vgaioen	***Status***
‘0’	VGA I/O locations are not decoded (hard reset mode if vgaboot = 0)
‘1’	VGA I/O locations are decoded (hard reset mode if vgaboot = 1)

On hard reset, the sampled vgaboot strap (VD<13>) will replace the **vgaioen** value.

Note that the MGA control registers and MGA frame buffer map are always enabled for all modes.

fbmskN <11:9>

VGA frame buffer mask. This field allows re-mapping of the VGA frame buffer to the top of memory (values not shown in the table are reserved):

fbmskN	*Frame buffer size shared with the host CPU*	*VGA frame buffer location*
‘000’	1 MBytes	700000h - 7FFFFFh
‘001’	2 MBytes	600000h - 7FFFFFh
‘011’	4 MBytes	400000h - 7FFFFFh
‘111’	8 MBytes	000000h - 7FFFFFh

memconfig <12>

Memory configuration. This bit indicates the configuration of the memory chips which comprise the frame buffer (refer to ‘SDRAM/SGRAM Configurations’ on page 6-5 for more information regarding pin configurations). It is used by the memory controller to map the addresses according to the following table:

memconfig	*Internal chip configuration*			
	No. of Banks	*Bank Size*	*Word Size*	*Total*
‘0’	2	128k	32	8 Mb
	2	128k	16	4 Mb
‘1’	2	512k	16	16 Mb
	2	1 M	8	16 Mb
	2	2 M	4	16 Mb

splitmode <13>

Split frame buffer mode. When this field is ‘1’, the 8 MByte frame buffer is divided into two sections:

- 0 MBytes to (4 MBytes - 1) is the graphics buffer
- 4 MBytes to (8 MBytes - 1) is the video buffer

hardpwmsk <14>

Hardware plane write mask. This field is used to enable SGRAM special functions. This field must always be set to ‘0’ when SDRAM is used. (when SGRAM is used, software must set **hardpwmsk** to ‘1’ in order to take advantage of special SGRAM functions).

- 0: Special SGRAM functions are not available; however, a plane write mask cycle will be emulated in the MGA-1064SG at a reduced performance level.
- 1: Special SGRAM functions are enabled, so plane write mask operations will be performed by the memory (with optimal performance) and block mode operations are available. Note that **hardpwmsk** must ***never be set to ‘1’*** when the memory does not consist of ***SGRAM***.

unimem RO <15>

Unified memory. On hard reset, the sampled unimem strap (VD<15>) value will replace the value of **unimem**. The unimem strap must always be set to ‘0’.

rfhcnt <19:16>

Refresh counter. Defines the rate of the MGA-1064SG’s memory refresh. Page cycles and co-processor acknowledges will not be interrupted by a refresh request unless a second request is queued (in this case, the refresh request becomes the highest priority after the screen refresh). Since all banks have to be precharged, both queued refreshes will keep this new highest priority.

When programming the **rfhcnt** register, the following rule must be respected:

ram refresh period >= (**rfhcnt**<3:1> * 256 + **rfhcnt**<0> * 64 + 1) * MCLK period

✎ Note that setting **rfhcnt** to zero halts the memory refresh.

eepromwt **<20>**	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.
productid **RO <28:24>**	Product ID. Sampled state of the VD<12:8> pins 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, an indication if a writable ROM is present, and so on). These bits do not control hardware within the chip.
noretry **<29>**	Retry disable. A ‘1’ disables generation of the retry sequence on the PCI bus (except during a VGA snoop cycle). At this setting, violation of the PCI latency rules may occur.
biosen **<30>**	BIOS enable. On hard reset, the sampled biosen strap (VD<14>) is loaded into this field. • 0: The **ROMBASE** space is automatically disabled. • 1: The **ROMBASE** space is enabled **- rombase** must be correctly initialized since it contains unpredictable data.
powerpc **<31>**	Power PC mode. • 0: No special swapping is performed. The host processor is assumed to be of little endian type. • 1: Enables byte swapping for the memory range **MGABASE1** + 1C00h to **MGABASE1** + 1EFFh, as well as **MGABASE1** + 2C00h to **MGABASE1** + 2DFFh. This swapping allows a big endian processor to access the information in the same manner as a little endian processor.
Reserved:	**<7:6> <23:21>** Reserved. When writing to this register, the bits in these fields must be set to ‘0’.

ROM Base Address ROMBASE

Address 30h (CS)

Attributes R/W, BYTE/WORD/DWORD, STATIC

Reset Value `0000 0000 0000 0000 0000 0000 0000 0000b`

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).

biosen	**romen** *attribute*
‘0’	RO (read as 0)
‘1’	R/W

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**.

biosen	**rombase** *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. 8 MByte Pseudo-DMA window
4. VGA frame buffer aperture
5. 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.

Reserved <15:1> Reserved. When writing to this register, the bits in this field must be set to ‘0’. Reading will give ‘0’s.

SUBSYSID Subsystem ID

Address 2Ch (CS) RO; 4Ch (CS) WO

Attributes BYTE/WORD/DWORD, STATIC

Reset Value `0000 0000 0000 0000 0000 0000 0000 0000b`

subsysid																subsysvid															
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

subsysvid <15:0> Subsystem vendor ID. This field is reset with the value that is found in word location 7FF8h of the BIOS ROM (32K ROM used), or at word location FFF8h of the BIOS ROM (64K ROM used). It indicates a subsystem vendor ID as provided by the PCI Special Interest Group to the manufacturer of the add-in board which contains the MGA-1064SG chip.

subsysid <31:16> Subsystem ID. This field is reset with the value that is found in word location 7FFAh of the BIOS ROM (32K ROM used), or at word location FFFAh of the BIOS ROM (64K ROM used). It indicates a subsystem ID as determined by the manufacturer of the add-in board which contains the MGA-1064SG chip.

Note: If the biosen strap is ‘0’, the ROM will not be read and the value found in the register will be 00000000h. In this case, the driver ***must write the correct values*** to this register after power-up.

Note: This register must contain all zeros if the manufacturer of the add-in board does not have a subsystem vendor ID, or if the manufacturer does not wish to support the **SUBSYSID** register.

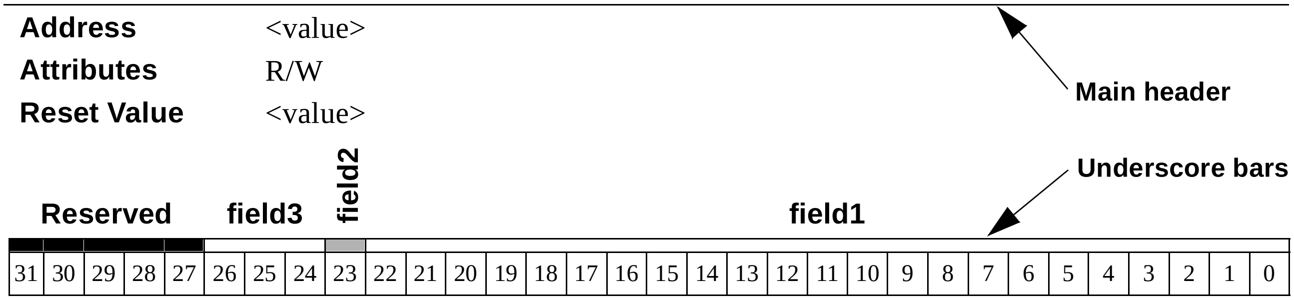
Note: There may be a delay of up to 500 PCLKs following a hard reset before this register is initialized.

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. When writing to this register, the bits in this field must be set to '0'. (Reserved registers always appear at the end of a register description.)

Memory Address

The addresses of all the Power Graphic mode registers are provided in Chapter 3. Note: MEM indicates that the address lies in the memory space; IO indicates that 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

Here are some of the symbols that appear as part of a register's 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 0000 ???? 1101 0000 0000 0000b`
  (b = binary, ? = unknown, _ = reset on soft/hard reset (see above), N/A = not applicable)

Address **MGABASE1** + 1C60h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value 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**.

ar0 <17:0>

Address register 0. The ar0 field is an 18-bit signed value in two's complement notation.

- For AUTOLINE, this register holds the x end address (in pixels). See the **XYEND register on page 4-77.**
- For LINE, it holds 2 x 'b'.
- For a filled trapezoid, it holds 'dYl'.
- For a BLIT, **ar0** holds the line end source address (in pixels).
- For an ILOAD_SCALE or ILOAD_FILTER, **ar0** holds the destination end address (in pixels) minus one line.

Reserved <31:18>

Reserved. When writing to this register, the bits in this field must be set to '0'.

Multi-Purpose Address1 — AR1

Address	**MGABASE1** + 1C64h (MEM)
Attributes	WO, FIFO, DYNAMIC, DWORD
Reset Value	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 x ‘b’ - ‘a’ -[**sdy**]).
- 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:

  ‘errl’ = [**sdxl**] ? ‘dXl’ + ‘dYl’ - 1 : -‘dXl’
- 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. When writing to this register, the bits in this field must be set to ‘0’.

Address **MGABASE1** + 1C68h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value 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 4-77.**
- For LINE, it holds the minor axis error increment (initially 2 x 'b' - 2 x 'a').
- For a filled trapezoid, it holds the minor axis increment (-|dXl|).
- 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)
- For ILOAD_HIQH and ILOAD_HIQHV, it holds:

$$\frac{(\mathrm{SRC\_X\_DIMEN} - 1) << 16}{(\mathrm{DST\_X\_DIMEN} - 1)} + 1$$

This register is not used for BLIT operations without scaling.

Reserved <31:18> Reserved. When writing to this register, the bits in this field must be set to '0'.

Address	**MGABASE1** + 1C6Ch (MEM)
Attributes	WO, FIFO, DYNAMIC, DWORD
Reset Value	Unknown

<table>
<tr><th colspan="6">Reserved</th><th colspan="2">spage</th><th colspan="24">ar3</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

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. When writing to this register, the bits in this field must be set to '0'.

AR4 Multi-Purpose Address 4

Address	**MGABASE1** + 1C70h (MEM)
Attributes	WO, FIFO, DYNAMIC, DWORD
Reset Value	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:

  'errr' = [**sdxr**] ? 'dXr' + 'dYr' - 1 : -'dXr'

- This register is used during AUTOLINE, but doesn't need to be initialized.
- This register is not used for LINE or BLIT operations without scaling.
- For the ILOAD_SCALE, ILOAD_FILTER, ILOAD_HIQH, and ILOAD_HIQHV, it holds the error term, but it doesn't need to be initialized.

Reserved <31:18>

Reserved. When writing to this register, the bits in this field must be set to '0'.

Multi-Purpose Address 5 AR5

Address MGABASE1 + 1C74h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value 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 4-78.** 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. When writing to this register, the bits in this field must be set to '0'.

Address	**MGABASE1** + 1C78h (MEM)
Attributes	WO, FIFO, DYNAMIC, DWORD
Reset Value	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 4-78.** 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)
- 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)

✎ For ILOAD_SCALE and ILOAD_FILTER, **ar6** must be less than or equal to zero.

- For ILOAD_HIQH and ILOAD_HIQHV, it holds:

$$\frac{(\mathrm{SRC\_X\_DIMEN} - \mathrm{DST\_X\_DIMEN}) << 16}{(\mathrm{DST\_X\_DIMEN} - 1)}$$

This register is not used for BLIT (without scaling) or IDUMP operations.

Reserved <31:18>

Reserved. When writing to this register, the bits in this field must be set to '0'.

Background Color / Blit Color Mask

BCOL

Address	**MGABASE1** + 1C20h (MEM)
Attributes	WO, FIFO, STATIC, DWORD
Reset Value	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

bltcmsk

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 5-18 for the the definition of the slice in each mode.

bltcmsk <31:0> Blit color mask. This field enables blit transparency comparison on a planar basis (‘0’ indicates a masked bit). Refer to the description of the **transc** field of **DWGCTL** for the transparency equation.

In 8 and 16 bit/pixel configurations, all bits in **bltcmsk** are used, so the mask information must be replicated on all bytes.

CXBNDRY Clipper X Boundary

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 4-29.**
cxright <26:16>	Clipper x right boundary. See the **CXRIGHT register on page 4-30.**
Reserved:	**<15:11> <31:27>** Reserved. When writing to this register, the bits in these fields must be set to ‘0’.

Clipper X Minimum Boundary CXLEFT

Address	**MGABASE1** + 1CA0h (MEM)
Attributes	WO, FIFO, STATIC, DWORD
Reset Value	Unknown

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																					**cxleft**										

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 **YDST on page 4-80)**. 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. When writing to this register, the bits in this field must be set to ‘0’.

Address	**MGABASE1** + 1CA4h (MEM)
Attributes	WO, FIFO, STATIC, DWORD
Reset Value	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 **YDST on page 4-80)**. 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. When writing to this register, the bits in this field must be set to ‘0’.

DMA Map 3h to 0h DMAMAP30

Address	**MGABASE1** + 1E30h (MEM)
Attributes	R/W, STATIC, BYTE/WORD/DWORD
Reset Value	Unknown

map_reg3								map_reg2								map_reg1								map_reg0							
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

map_regN <31:0>	Map register N. The 16 8-bit map registers form a look-up table used when addressing through the range of **MGABASE1** + 1E80h to **MGABASE1** + 1EBFh. The **DMAMAP30** register contains entries 0h to 3h of this lookup table. Refer to **DWG_INDIR_WT<15:0>** for more information.

DMAMAP74 DMA Map 7h to 4h

Address **MGABASE1** + 1E34h (MEM)

Attributes R/W, STATIC, BYTE/WORD/DWORD

Reset Value Unknown

map_reg7								**map_reg6**								**map_reg5**								**map_reg4**							
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

map_regN <31:0> Map register N. The 16 8-bit map registers form a look-up table used when addressing through the range of **MGABASE1** + 1E80h to **MGABASE1** + 1EBFh. The **DMAMAP74** register contains entries 4h to 7h of this lookup table. Refer to **DWG_INDIR_WT<15:0>** for more information.

DMA Map Bh to 8h DMAMAPB8

Address	**MGABASE1** + 1E38h (MEM)
Attributes	R/W, STATIC, BYTE/WORD/DWORD
Reset Value	Unknown

map_regb								**map_rega**								**map_reg9**								**map_reg8**							
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

map_regN <31:0>	Map register N. The 16 8-bit map registers form a look-up table used when addressing through the range of **MGABASE1** + 1E80h to **MGABASE1** + 1EBFh. The **DMAMAPB8** register contains entries 8h to Bh of this lookup table. Refer to **DWG_INDIR_WT<15:0>** for more information.

DMAMAPFC — DMA Map Fh to Ch

Address	**MGABASE1** + 1E3Ch (MEM)
Attributes	R/W, STATIC, BYTE/WORD/DWORD
Reset Value	Unknown

map_regf								**map_rege**								**map_regd**								**map_regc**							
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

map_regN <31:0>

Map register N. The 16 8-bit map registers form a look-up table used when addressing through the range of **MGABASE1** + 1E80h to **MGABASE1** + 1EBFh. The **DMAMAPFC** register contains entries Ch to Fh of this lookup table. Refer to **DWG_INDIR_WT<15:0>** for more information.

Address MGABASE1 + 1C54h (MEM)
Attributes WO, FIFO, STATIC, DWORD
Reset Value Unknown

dmapad

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

dmapad <31:0> DMA Padding. Writes to this register, which have no effect on the drawing engine, can be used to pad display lists. Padding should be used only when necessary, since it may impact drawing performance.

Address **MGABASE1** + 1CC0h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value 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.

Address	**MGABASE1** + 1CC8h (MEM)
Attributes	WO, FIFO, STATIC, DWORD
Reset Value	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

Address	**MGABASE1** + 1CCCh (MEM)
Attributes	WO, FIFO, STATIC, DWORD
Reset Value	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
<23:0>

Data ALU register 4. This field holds a signed 9.15 value in two's complement notation.

- 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 TRAP_ILOAD, 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
<31:24>

Reserved. When writing to this register, the bits in this field must be set to '0'.

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
<23:0>

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

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

Reserved
<31:24>

Reserved. When writing to this register, the bits in this field must be set to ‘0’.

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

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

Reserved <31:24> Reserved. When writing to this register, the bits in this field must be set to '0'.

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 TRAP_ILOAD, 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. When writing to this register, the bits in this field must be set to '0'.

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 TRAP_ILOAD, 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. When writing to this register, the bits in this field must be set to ‘0’.

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 TRAP_ILOAD, 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. When writing to this register, the bits in this field must be set to '0'.

Address **MGABASE1** + 1CF0h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value 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 TRAP_ILOAD, 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. When writing to this register, the bits in this field must be set to '0'.

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 TRAP_ILOAD, 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. When writing to this register, the bits in this field must be set to '0'.

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 TRAP_ILOAD, 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. When writing to this register, the bits in this field must be set to '0'.

Address	**MGABASE1** + 1E80h (MEM) (entry 0)
	...
	MGABASE1 + 1EBCh (MEM) (entry 15)
Attributes	WO, DWORD
Reset Value	N/A

lut entry N

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

lutentry N <31:0> These 16 registers are a lookup table that can be used in conjunction with the **DMAMAP** registers. Writing to these locations address the register that is programmed in the Nth byte of the **DMAMAP**. This indirect write register provides a means to access non-sequential drawing registers sequentially.

Address	**DWG_INDIR_WT** *Register*
MGABASE1 + 1C00h + **map_reg0**	**DWG_INDIR_WT**<0>
MGABASE1 + 1C00h + **map_reg1**	**DWG_INDIR_WT**<1>
MGABASE1 + 1C00h + **map_reg2**	**DWG_INDIR_WT**<2>
MGABASE1 + 1C00h + **map_reg3**	**DWG_INDIR_WT**<3>
MGABASE1 + 1C00h + **map_reg4**	**DWG_INDIR_WT**<4>
MGABASE1 + 1C00h + **map_reg5**	**DWG_INDIR_WT**<5>
MGABASE1 + 1C00h + **map_reg6**	**DWG_INDIR_WT**<6>
MGABASE1 + 1C00h + **map_reg7**	**DWG_INDIR_WT**<7>
MGABASE1 + 1C00h + **map_reg8**	**DWG_INDIR_WT**<8>
MGABASE1 + 1C00h + **map_reg9**	**DWG_INDIR_WT**<9>
MGABASE1 + 1C00h + **map_rega**	**DWG_INDIR_WT**<10>
MGABASE1 + 1C00h + **map_regb**	**DWG_INDIR_WT**<11>
MGABASE1 + 1C00h + **map_regc**	**DWG_INDIR_WT**<12>
MGABASE1 + 1C00h + **map_regd**	**DWG_INDIR_WT**<13>
MGABASE1 + 1C00h + **map_rege**	**DWG_INDIR_WT**<14>
MGABASE1 + 1C00h + **map_regf**	**DWG_INDIR_WT**<15>

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**			**opcod**			
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

opcod **<3:0>** Operation code. The **opcod** field defines the operation that is selected by the drawing engine.

		opcod	
Function	***Sub-Function***	***Value***	***Mnemonic***
Lines		‘0000’	LINE_OPEN
	AUTO	‘0001’	AUTOLINE_OPEN
	WRITE LAST	‘0010’	LINE_CLOSE
	AUTO, WRITE LAST	‘0011’	AUTOLINE_CLOSE
Trapezoid		‘0100’	TRAP
	Data from host	‘0101’	TRAP_ILOAD
Blit	RAM -> RAM	‘1000’	BITBLT
	HOST -> RAM	‘1001’	ILOAD
	HOST -> RAM scale	‘1101’	ILOAD_SCALE
	HOST -> RAM scale, filter	‘1111’	ILOAD_FILTER
	RAM -> HOST	‘1010’	IDUMP
	HOST -> RAM scale, high-quality filter	‘0111’	ILOAD_HIQH
	HOST -> RAM horizontal and vertical scale, high-quality filter	‘1110’	ILOAD_HIQHV
	Reserved	‘1011’	
	”	‘1100’	

atype <6:4>

Access type. The **atype** field is used to define the type of access performed to the RAM.

atype		RAM Access
Value	*Mnemonic*	
‘000’	RPL	Write (replace)
‘001’	RSTR	Read-modify-write (raster)
‘010’		Reserved
‘011’	ZI	Depth mode with Gouraud
‘100’	BLK	Block write mode (1) (2)
‘101’		Reserved
‘110’		Reserved
‘111’	I	Gouraud (with depth compare) (3)

(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.

(2) The **hardpwmsk** field of the **OPTION** register must be set to ‘1’.

(3) Depth comparison works according to the **zmode** setting (same as ‘ZI’); however, the depth is never updated.

linear <7>

Linear mode. Specifies whether the blit is linear or xy.

- 0: xy blit
- 1: linear blit

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.

zmode		Pixel Update
Value	*Mnemonic*	
‘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 >=

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 4-73).

- 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. Writing to any of the **SRC0, SRC1, SRC2, SRC3** or **PAT0, PAT1** registers while **solid** is ‘1’ may produce unpredicatable results.

arzero
<12>

AR register at zero. The **arzero** field provides an alternate way to set certain **AR** registers (see descriptions starting on page 4-20).

- 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 **AR**x registers when **arzero** = 1 will produce unpredictable results.

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** = 0 Horizontal scan right
sdxl = 0 Left edge in increment mode
sdxr = 0 Right edge in increment mode
sdy = 0 **iy** (see **PITCH on page 4-68)** is added to
ydst (see **YDST on page 4-80)**

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

Writing to the **SGN** register when **sgnzero** = 1 will produce unpredictable results.

shftzero <14>

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

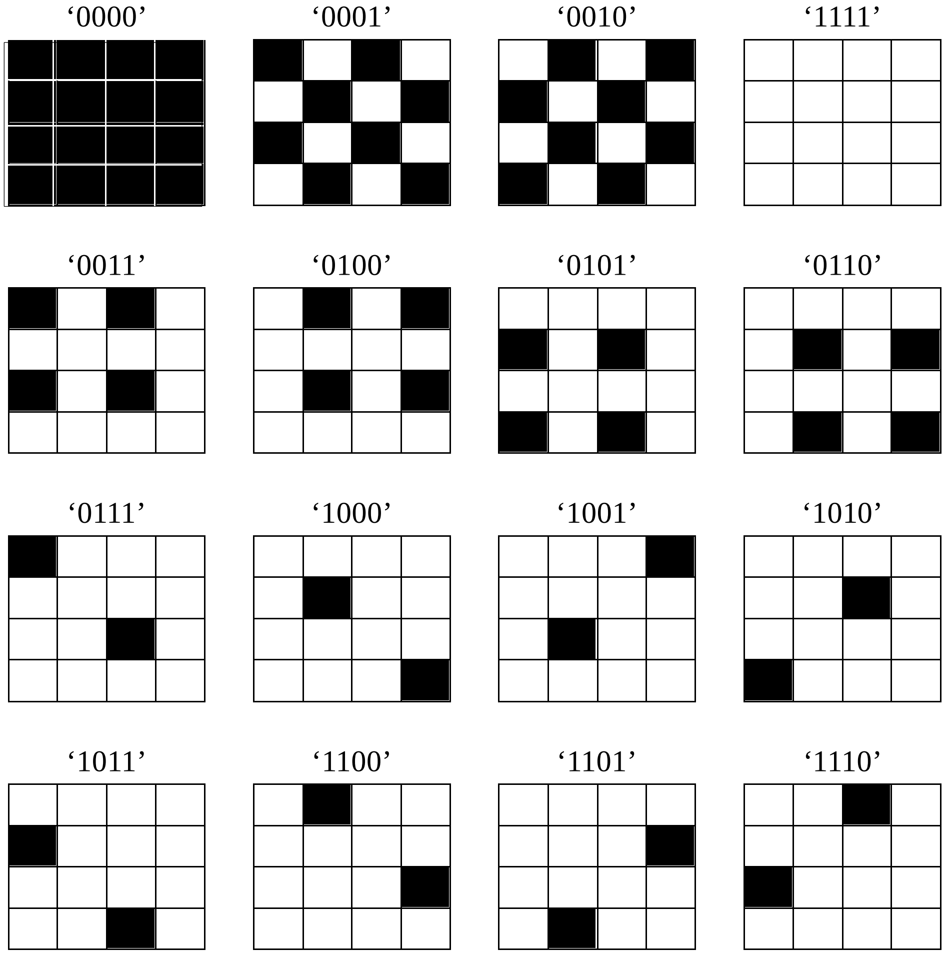
- 0: No effect
- 1: **SHIFT** <= 0h

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.

bop	***Function***
‘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):



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

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 RAM-to-RAM BITBLT operation, hardware fast clipping will be enabled if BFCOL is specified.
- The field is also used for the IDUMP and TRAP_ILOAD operations.

Refer to the subsections contained in ‘Drawing in Power Graphic Mode’ on page 5-25 for more information on how to use this field. That section also presents the definition of the various pixel formats.

pattern <29> Patterning enable. This bit specifies if the patterning is enabled when performing BITBLT operations.

- 0: Patterning is disabled.
- 1: Patterning is enabled.

transc **<30>**	Transparency color enabled. This field can be enabled for blits, vectors that have a linestyle, and trapezoids with patterning. For operations with color expansion, this bit specifies if the background color is used.

- 0: Background color is opaque.
- 1: Background color is transparent.

For other types of blit, this field enables the transparent blit feature, based on a comparison with a transparent color key. This transparency is defined by the following equation:

```
if ( transc==1 && (source & bltcmsk==bltckey) )
   do not update the destination
else
   update the destination with the source
```

Refer to the **FCOL** and **BCOL** register descriptions for the definitions of the **bltckey** and **bltcmsk** fields, respectively.

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

Reserved. When writing to this register, the bits in these fields must be set to ‘0’.

FCOL — Foreground Color / Blit Color Key

Address	**MGABASE1** + 1C24h (MEM)
Attributes	WO, FIFO, STATIC, DWORD
Reset Value	Unknown

forcol

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

bltckey

forcol <31:0>

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

Refer to ‘Pixel Format’ on page 5-18 for the the definition of the slice in each mode.

Part of the **forcol** register is also used for Gouraud shading to generate the alpha bits. In 32 bpp, bits 31 to 24 originate from **forcol**<31:24>. In 16 bpp, when 5:5:5 mode is selected, bit 15 originates from **forcol**<31>.

bltckey <31:0>

Blit color key. This field specifies the value of the color that is defined as the ‘transparent’ color. Planes that are not used must be set to ‘0’. Refer to the description of the **transc** field of **DWGCTL** for the transparency equation

In 8 and 16 bit/pixel configurations, all bits in **bltckey** are used, so the color information must be replicated on all bytes.

Address **MGABASE1** + 1E10h (MEM)

Attributes RO, DYNAMIC, BYTE/WORD/DWORD

Reset Value `0000 0000 0000 0000 0000 0010 0010 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>.

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).

Reserved: **<7:6> <31:10>**

Reserved. When writing to this register, the bits in these fields must be set to ‘0’. Reading will give ‘0’s.

Address	**MGABASE1** + 1C84h (MEM)
Attributes	WO, FIFO, DYNAMIC, DWORD
Reset Value	Unknown

fxright																**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

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 4-60.**

X Address (Left) FXLEFT

Address MGABASE1 + 1CA8h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value 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. When writing to this register, the bits in this field must be set to '0'.

Address MGABASE1 + 1CACh (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value 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.

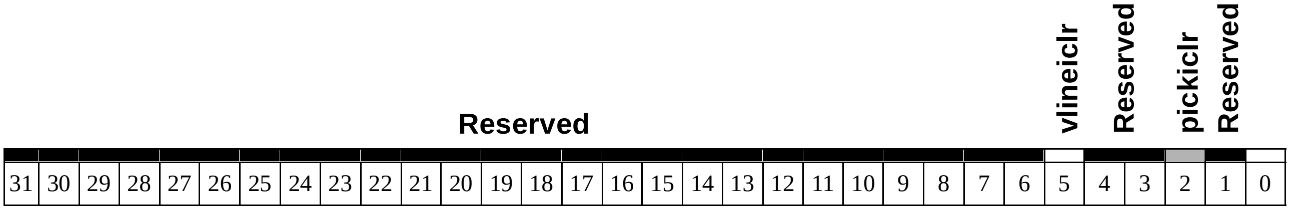
Reserved <31:16>

Reserved. When writing to this register, the bits in this field must be set to ‘0’.

Address MGABASE1 + 1E18h (MEM)

Attributes WO, DYNAMIC, BYTE/WORD/DWORD

Reset Value `0000 0000 0000 0000 0000 0000 0000 0000b`



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

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

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

Reserved. When writing to this register, the bits in these fields must be set to ‘0’. Reading will give ‘0’s.

IEN Interrupt Enable

Address **MGABASE1** + 1E1Ch (MEM)

Attributes R/W, STATIC, BYTE/WORD/DWORD

Reset Value `0000 0000 0000 0000 0000 0000 0000 0000b`

Reserved																									extien	vlineien	Reserved		pickien	Reserved	softrapien
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

pickien <2> Picking interrupt enable. When set to ‘1’, enables interrupts if a picking interrupt occurs.

vlineien <5> Vertical line interrupt enable. When set to ‘1’, an interrupt will be generated when the vertical line counter equals the vertical line interrupt count.

extien <6> External interrupt enable. When set to ‘1’, an external interrupt will contribute to the generation of a PCI interrupt on the PINTA/ line.

Reserved: **<1> <4:3> <31:7>**

Reserved. When writing to this register, the bits in these fields must be set to ‘0’. Reading will give ‘0’s.

Length LEN

Address	**MGABASE1** + 1C5Ch (MEM)
Attributes	WO, FIFO, DYNAMIC, DWORD
Reset Value	`0000 0000 0000 0000 0000 0000 0000 0000b`

beta				**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 **<15:0>**	Length. The length field is a 16-bit unsigned value. • 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 blitted. • To load the texture color palette, **length** is programmed with the number of locations in the LUT to be filled.
beta **<31:28>**	Beta factor. This field is used to drive the vertical scaling in ILOAD_HIQHV (it is not used for other opcodes). The **beta** field represents the four least significant bits of a value between 1 and 16 (16 is 0000b), which represents a beta factor of 1/16 through 16/16.
Reserved **<27:16>**	Reserved. When writing to this register, the bits in this field must be set to '0'.

Address	**MGABASE1** + 1C04h (MEM)
Attributes	WO, FIFO, STATIC, DWORD
Reset Value	`0000 0000 0000 0000 0000 0000 0000 0000`b

dit555	**nodither**	**tlutload**	**Reserved**													**memreset**	**jedecrst**	**Reserved**												**pwidth**	
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

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

pwidth		
Value	***Mnemonic***	***Mode***
‘00’	PW8	8 bpp
‘01’	PW16	16 bpp
‘10’	PW32	32 bpp
‘11’	PW24	24 bpp

jedecrst <14> JEDEC power-up sequence.

- 0: Memory sequencer performs the mode register set, followed by eight refreshes.
- 1: Memory sequencer performs the JEDEC sequence (eight refreshes followed by the mode register set).

memreset <15> Resets the RAM. When this bit is set to ‘1’, the memory sequencer will generate a power-up cycle to the RAM.

Caution: Refer to Section 5.3.3 on page 5-21 for instructions on when to use this field. The **memreset** field must always be set to ‘0’ except under specific conditions which occur during the reset sequence.

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

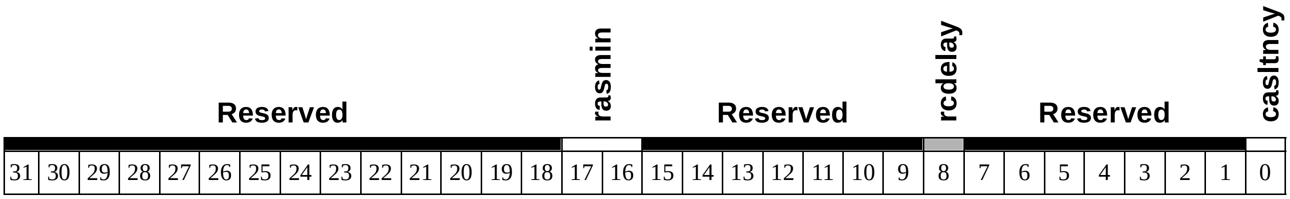
Reserved **<13:2> <28:16>**

Reserved. When writing to this register, the bits in these fields must be set to ‘0’.

Address **MGABASE1** + 1C08h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value `0000 0000 0000 0011 0000 0001 0000 0001`b



casltncy <0> CAS latency. This bit must be programmed prior to the memory power-up sequence. Refer to ‘Power Up Sequence’ on page 5-20 for more details.

casltncy	*CAS Latency (mclk)*
‘0’	2
‘1’	3

rcdelay <8> RAS to CAS delay. This bit selects one of two RAS to CAS delay values, as shown below:

rcdelay	*RAS to CAS Delay (mclk)*
‘0’	2
‘1’	3

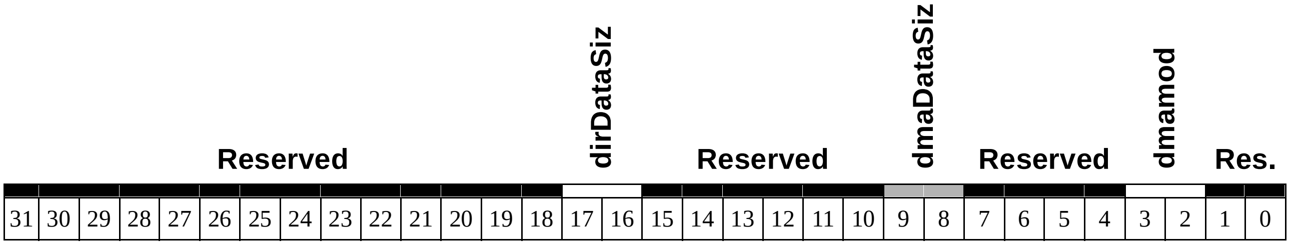
rasmin <17:16> RAS minimum active time. The valid values are shown below:

rasmin	*RAS Minimum (mclk)*
‘00’	4
‘01’	5
‘10’	6
‘11’	7

Reserved **<7:1> <15:9> <31:18>**

Reserved. When writing to this register, the bits in these fields must be set to ‘0’.

Address	**MGABASE1** + 1E54h (MEM)
Attributes	R/W, STATIC BYTE/WORD/DWORD
Reset Value	0000 0000 0000 0000 0000 0000 0000 0000b



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 or to the 8 MByte Pseudo-DMA window. 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				

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.

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

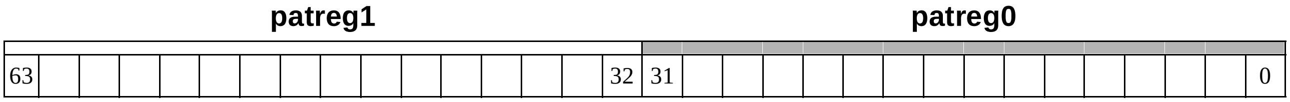
Reserved. When writing to this register, the bits in these fields must be set to ‘0’. Reading will give ‘0’s.

Pattern PAT0, PAT1

Address **MGABASE1** + 1C10h **MGABASE1** + 1C14h (MEM)

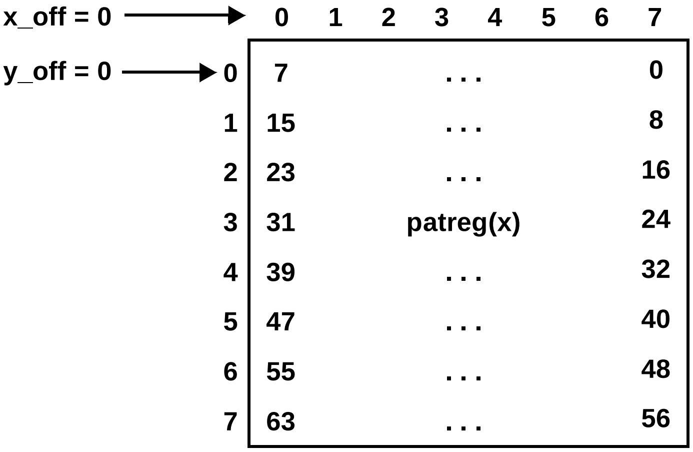
Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown



patreg <63:0> Pattern register. The **PAT** registers are not physical registers. They simply provide an alternate way to load the **SRC** registers with a Windows format 8 x 8 pattern.

The following illustration shows how the data written to the **PAT** registers is mapped into the frame buffer. The screen representation is shown below:



The pattern-pixel pinning can be changed using the **x_off** and **y_off** fields of the **SHIFT** register. See the **SRC0, SRC1, SRC2, SRC3 register on page 4-73.**

PITCH — Memory Pitch

Address MGABASE1 + 1C8Ch (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value Unknown

Reserved																ylin	Res.			iy											
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 4-80)** between two destination lines. The **iy** field is also used as the multiplicator 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:

Pitch	**iy**	*Pitch*	**iy**
512	001000000000b	1152	010010000000b
640	001010000000b	1280	010100000000b
768	001100000000b	1600	011001000000b
800	001100100000b	1664	011010000000b
832	001101000000b	1920	011110000000b
960	001111000000b	2048	100000000000b
1024	010000000000b		

This register must be loaded with a value that is a multiple of 32 or 64, due to a restriction involving block mode, according to the table below. See 'Constant Shaded Trapezoids / Rectangle Fills' on page 5-35. See page 4-50 for additional restrictions that apply to block mode (**atype** = BLK).

pwidth	*Value*
PW8	64
PW16	32
PW24	64
PW32	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: **<14:12> <31:16>**

Reserved. When writing to this register, the bits in these fields must be set to '0'.

Plane Write Mask PLNWT

Address **MGABASE1** + 1C1Ch (MEM)
Attributes WO, FIFO, STATIC, DWORD
Reset Value 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. The plane write mask is not used for z cycles, or for direct write access (all planes are written in this case).

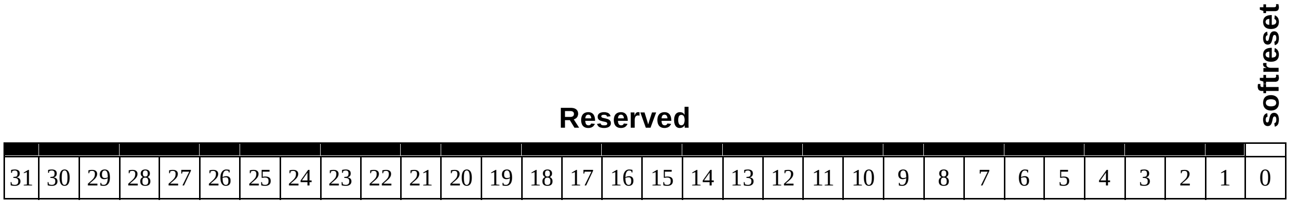
- 0 = inhibit write
- 1 = permit write

The bits from the **plnwrmsk**<31:0> register 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**<31:0> 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 5-18 for the the definition of the slice in each mode.

RST Reset

Address	**MGABASE1** + 1E40h (MEM)
Attributes	R/W, STATIC, BYTE/WORD/DWORD
Reset Value	`0000 0000 0000 0000 0000 0000 0000 0000`b



softreset
<0>

Soft reset. When set to ‘1’, this resets all bits that permit software resets. This has the effect of flushing the BFIFO and aborting the current drawing instruction. A soft reset will not generate invalid memory cycles, and memory contents are preserved. 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 5.3.3 on page 5-25 for instructions on when to use this field.

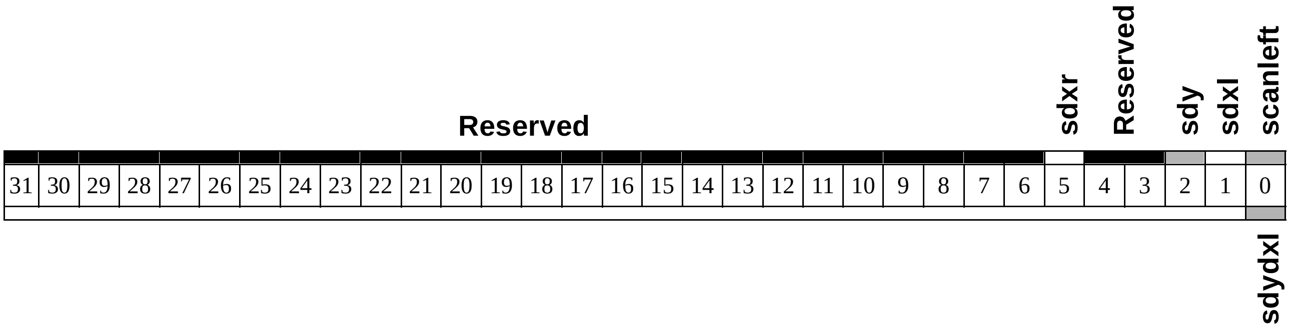
WARNING! A soft reset will not re-read the chip strapping.

Reserved
<31:1>

Reserved. When writing to this register, the bits in this field must be set to ‘0’. Reading will give ‘0’s.

Sign SGN

Address MGABASE1 + 1C58h (MEM)
Attributes WO, FIFO, DYNAMIC, DWORD
Reset Value 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**.

sdydxl <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 **sdydxl** 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 4-49)**. 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

sdy <2> Sign of delta y. The **sdy** 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. When writing to this register, the bits in these fields must be set to '0'.

SHIFT

Funnel Shifter Control

Address	**MGABASE1** + 1C50h (MEM)
Attributes	WO, FIFO, DYNAMIC, DWORD
Reset Value	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**			

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 4-73), 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. When writing to this register, the bits in these fields must be set to ‘0’.

Source

SRC0, SRC1, SRC2, SRC3

Address	**MGABASE1** + 1C30h, + 1C34h, + 1C38h, + 1C3Ch (MEM)
Attributes	WO, FIFO, DYNAMIC, DWORD
Reset Value	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.

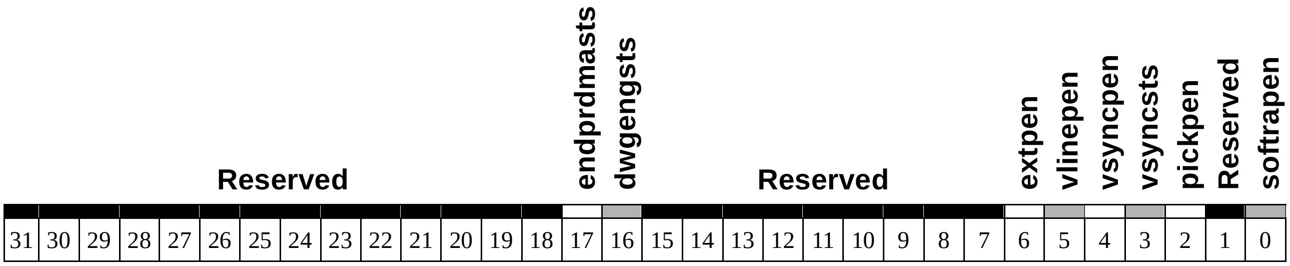
For LINE with the RPL or RSTR attribute, the source register is used to store the line style. The **funcnt** field of the **SHIFT** register points to the selected source register bit being used as the linestyle for the current pixel. Refer to Section 5.5.4.3 on page 5-30 for more details.

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 5.5.5.3 on page 5-36 for more details.

For all BLIT 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 4-67) will load the **SRC** registers.

Address	**MGABASE1** + 1E14h (MEM)
Attributes	RO, DYNAMIC, BYTE/WORD/DWORD
Reset Value	`0000 0000 0000 0000 0000 0000 0?00 0000b`



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 4-61)** 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 **CRTC11** 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 4-125.** This bit is cleared through the **vlineiclr** bit (see **ICLEAR on page 4-61)** 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> <15:7> <31:18>**

Reserved. When writing to this register, the bits in these fields must be set to ‘0’. 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).

Vertical Count VCOUNT

Address **MGABASE1** + 1E20h (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 <11:0> Vertical counter value. Writing has no effect. Reading will give the current vertical count value.

✎ 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.

Reserved <31:12> Reserved. When writing to this register, the bits in this field must be set to '0'. Reading will give '0's.

Address	**MGABASE1** + 1CB0h (MEM)
Attributes	WO, FIFO, DYNAMIC, DWORD
Reset Value	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 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 BLITs, this register is automatically loaded from **fxleft** (see **FXLEFT on page 4-59)** and **fxright** (see **FXRIGHT on page 4-60)**, and no initial value must be loaded. • For trapezoids with depth, this register is automatically loaded from **fxleft**. For trapezoids without depth, **xdst** will be loaded with the larger of **fxleft** or **cxleft**, and an initial value must not be loaded. (See **CXLEFT** on page 4-29.)
Reserved **<31:16>**	Reserved. When writing to this register, the bits in this field must be set to '0'.

Address **MGABASE1** + 1C44h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value 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.

Address	**MGABASE1** + 1C40h (MEM)
Attributes	WO, FIFO, DYNAMIC, DWORD
Reset Value	Unknown

y_start																**x_start**															
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.

Clipper Y Maximum Boundary YBOT

Address	**MGABASE1** + 1C9Ch (MEM)
Attributes	WO, FIFO, STATIC, DWORD
Reset Value	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 4-80)**. 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: **cybot** = (bottom line number) × **PITCH** + **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. When writing to this register, the bits in this field must be set to ‘0’.

YDST — Y Address

Address MGABASE1 + 1C90h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value 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 4-68**).

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:

ydst <-- (Y coordinate) * **PITCH** >> 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** must be 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.
- To load the texture color palette, **ydst** must be loaded with the position in the color palette (0 to 255) at which the texture color palette will begin loading.

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.

Reserved **<28:22>**

Reserved. When writing to this register, the bits in this field must be set to '0'.

Y Destination and Length YDSTLEN

Address	MGABASE1 + 1C88h (MEM)
Attributes	WO, FIFO, STATIC, DWORD
Reset Value	Unknown

yval																**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

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 4-63.**

yval <31:16>

The y destination value. See the **YDST register on page 4-80.** 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.

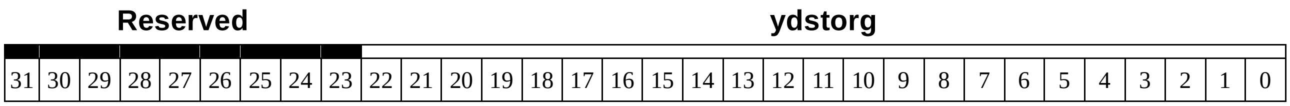
ydst<21:0> <= sign extension (**yval**<31:16>)

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

YDSTORG

Memory Origin

Address	**MGABASE1** + 1C94h (MEM)
Attributes	WO, FIFO, STATIC, DWORD
Reset Value	Unknown



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 value that is a multiple of 32 or 64, according to the table below, due to a restriction involving block mode. See 'Constant Shaded Trapezoids / Rectangle Fills' on page 5-35. See page 4-50 for additional restrictions that apply to block mode (**atype** = BLK).

pwidth	***Value***
PW8	64
PW16	32
PW24	64
PW32	32

Reserved
<31:23>

Reserved. When writing to this register, the bits in this field must be set to '0'.

Clipper Y Top Boundary YTOP

Address MGABASE1 + 1C98h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value Unknown

<table>
<tr><th colspan="9">Reserved</th><th colspan="23">cytop</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

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 4-80)**. 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:

cytop = (top line number) × **PITCH** + **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. When writing to this register, the bits in this field must be set to ‘0’.

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 <22:0>	Z-depth origin. The **zorg** field is a 23-bit unsigned value which provides an offset value (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). **zorg** = **ydstorg** + 2048 + n * 4096, where n is any integer that does not cause an overlap between the intensity and depth buffers.
Reserved <31:23>	Reserved. When writing to this register, the bits in this field must be set to ‘0’.

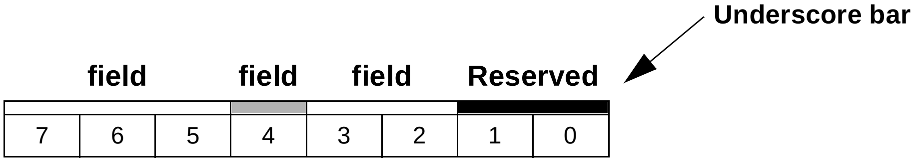
4.2 VGA Mode Registers

4.2.1 VGA Mode Register Descriptions

The MGA-1064SG 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>



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

- `000? 0000`b (b = binary, ? = unknown, N/A = not applicable)

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 0000b`

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

attrx <4:0>

Attribute controller index register. VGA.

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

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

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

- A read from port 3BAh/3DAh resets this port to the attributes address register. The first write at 3C0h after a 3BAh/3DAh reset accesses the attribute index. The next write at 3C0h accesses the palette. Subsequent writes at 3C0h toggle between the index and the palette.
- A read at port 3C1h does not toggle the index/data pointer.

Example of a palette write:

Reset pointer:	read at port 3BAh
Write index:	write at port 3C0h
Write color:	write at port 3C0h

Example of a palette read:

Reset pointer:	read at port 3BAh
Write index:	write at port 3C0h
Read color:	read at port 3C1h

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.
attrd **<15:8>**	**ATTR** data register. Retrieve or write the contents of the register pointed to by the **attrx** field.
Reserved **<7:6>**	Reserved. When writing to this register, the bits in this field must be set to 0. Reading will give 0’s.

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>

Reserved. When writing to this register, the bits in this field must be set to 0.

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> Graphics/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).

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 DAC. • 1: When it is set to ‘1’, **colsel54** (**ATTR14**<1:0>) will be transmitted to the DAC. See the **ATTR14** register on page 4-94.
Reserved **<4>**	Reserved. When writing to this register, this field must be set to 0.

Overscan Color

Index **attrx** = 11h

Reset Value `0000 0000`b

<table>
<tr><td colspan="8">ovscol</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

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 DAC. 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 4-150.

Reserved <7:6> Reserved. When writing to this register, the bits in this field must be set to 0.

Index **attrx** = 13h

Reset Value `0000 0000b`

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):

hpelcnt	8 dot mode pixel shifted **dotmode** (**SEQ1**<0>) = ‘1’	9 dot mode pixel shifted **dotmode** = ‘0’	**mode256** (**GCTL5**<6>) = ‘1’
‘0000’	0	1	0
‘0001’	1	2	-
‘0010’	2	3	1
‘0011’	3	4	-
‘0100’	4	5	2
‘0101’	5	6	-
‘0110’	6	7	3
‘0111’	7	8	-
‘1000’	-	0	-

Reserved <7:4>

Reserved. When writing to this register, the bits in this field must be set to 0.

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 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>

Reserved. When writing to this register, the bits in this field must be set to 0.

Address	**MGABASE1** + 1FFFh (MEM)
Attributes	R/W, BYTE, STATIC
Reset Value	Unknown

cacheflush

7	6	5	4	3	2	1	0

cacheflush <7:0>

Flush the cache. Writes to this register will flush the cache. For additional details, refer to 'Direct Access Read Cache' on page 5-4.

Even though this register can be read, its data has no significance, and may not be consistent. When writing to this register, ***all bits must be set to '0'***.

Address	03B4h (I/O), (**MISC**<0> == 0: MDA emulation) 03D4h (I/O), (**MISC**<0> == 1: CGA emulation) **MGABASE1** + 1FD4h (MEM)
Attributes	R/W, BYTE/WORD, STATIC
Reset Value	`nnnn nnnn 0000 0000b`

crtcd								**Reserved**		**crtcx**					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

crtcx <5:0> CRTC index register.

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

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

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

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

crtcd <15:8> CRTC data register.

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

Reserved <7:6> Reserved. When writing to this register, the bits in this field must be set to 0. Reading will give 0’s.

CRTC0 Horizontal Total

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.

Horizontal Display Enable End CRTC1

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.

CRTC2 Start Horizontal Blanking

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

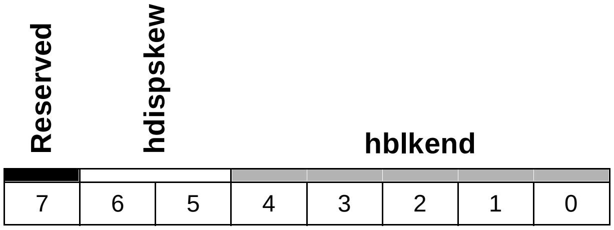
hblkstr

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.

End Horizontal Blanking CRTC3

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.

hdispskew	***Skew***
‘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.

CRTC4 Start Horizontal Retrace Pulse

Index **crtcx** = 04h

Reset Value `0000 0000`b

hsyncstr

7	6	5	4	3	2	1	0

hsyncstr <7:0>

Start horizontal retrace pulse. VGA/MGA.

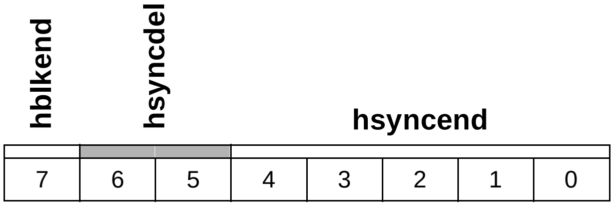
These are the low-order eight bits of a 9-bit register. Bit 8 is contained in **hsyncstr** (**CRTCEXT1**<2>). The horizontal sync signal becomes active when the horizontal character counter reaches this value.

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

End Horizontal Retrace CRTC5

Index **crtcx** = 05h

Reset Value `0000 0000b`



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.

hsyncdel	***Skew***
‘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 4-101.

Index	**crtcx** = 06h
Reset Value	`0000 0000`b

vtotal

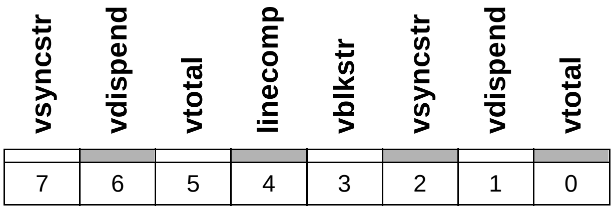
7	6	5	4	3	2	1	0

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).

Overflow

CRTC7

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



vtotal **<0>**	Vertical total bit 8. VGA/MGA. Contains bit 8 of the Vertical Total. See the **CRTC6** register on page 4-104. 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 4-116.
vsyncstr **<2>**	Vertical retrace start bit 8. VGA/MGA. Contains bit 8 of the Vertical Retrace Start. See the **CRTC10** register on page 4-114.
vblkstr **<3>**	Start vertical blank bit 8. VGA/MGA. Contains bit 8 of the Start Vertical Blank. See the **CRTC15** register on page 4-119.
linecomp **<4>**	Line compare bit 8. VGA/MGA. Line compare bit 8. See the **CRTC18** register on page 4-125. 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 4-104.
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 4-116.
vsyncstr **<7>**	Vertical retrace start bit 9. VGA/MGA. Contains bit 9 of the Vertical Retrace Start. See the **CRTC10** register on page 4-114.

CRTC8 Preset Row Scan

Index **crtcx** = 08h

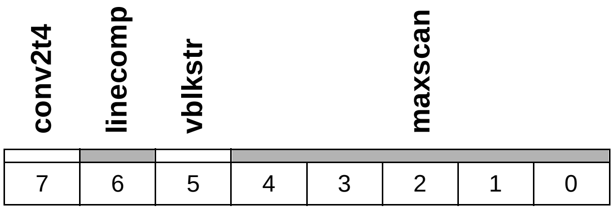
Reset Value `0000 0000`b

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>	Reserved. When writing to this register, this field must be set to 0. Reading will give 0’s.

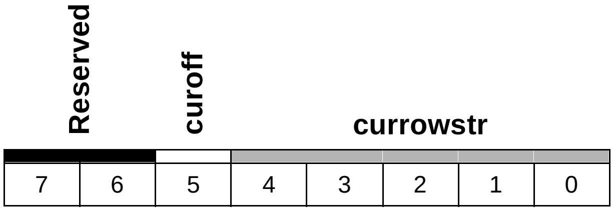
Index **crtcx** = 09h

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 4-119.
linecomp <6>	Line compare bit 9. VGA/MGA. Bit 9 of the Line Compare register. See the **CRTC18** register on page 4-125.
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.

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

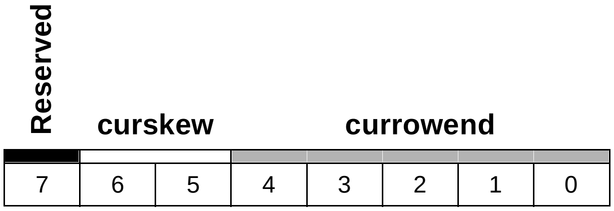


currowstr <4:0>	Row scan cursor begins. VGA. These bits specify the row scan of a character line where the cursor is to begin. When the cursor start register is programmed with a value greater than the cursor end register, no cursor is generated.
curoff<5>	Cursor off. VGA. • Logical ‘1’: turn off the cursor • Logical ‘0’: turn on the cursor
Reserved <7:6>	Reserved. When writing to this register, the bits in this field must be set to 0.

Cursor End CRTCB

Index **crtcx** = 0Bh

Reset Value `0000 0000`b



currowend <4:0>

Row scan cursor ends. VGA.

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

curskew <6:5>

Cursor skew control. VGA.

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

curskew	***Skew***
‘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>

Reserved. When writing to this register, this field must be set to 0.

Index **crtcx** = 0Ch

Reset Value `0000 0000`b

startadd							
7	6	5	4	3	2	1	0

startadd <7:0>

Start address, bits<15:8>. 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 (**CRTCD**<7:0>) 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 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.

Power Graphic Mode

Special considerations for Power Graphic mode programming are presented in the Note on page 5-67 in the ‘Programming in Power Graphic Mode’ section of Chapter 5.

Start Address Low CRTCD

Index **crtcx** = 0Dh

Reset Value `0000 0000`b

startadd

7	6	5	4	3	2	1	0

startadd <7:0>	Start address, bits<7:0>. VGA/MGA. These are the low-order eight bits of the start address. See the **CRTCC** register on page 4-110.

CRTCE Cursor Location High

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.

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 4-112.

CRTC10 Vertical Retrace Start

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.

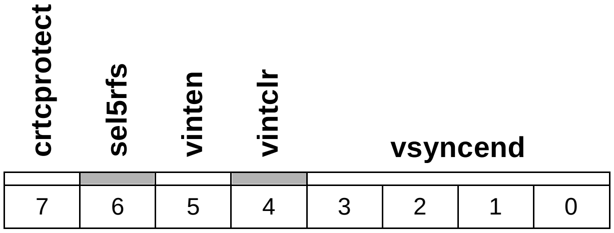
The vertical sync signal becomes active when the vertical line counter reaches the vertical retrace start value (a 12-bit value). The lower eight bits are located here. Bit 8 is in **CRTC7**<2>, bit 9 is in **CRTC7**<7>, and bits 10 and 11 are in **CRTCEXT2**<6:5>.

The units can be one or two scan lines:
- **hsyncsel** (**CRTC17**<2>) = 0: 1 scan line
- **hsyncsel** = 1: 2 scan lines

Vertical Retrace End CRTC11

Index **crtcx** = 11h

Reset Value `0000 0000b`



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.

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>.

Offset CRTC13

Index **crtcx** = 13h

Reset Value `0000 0000`b

offset

7	6	5	4	3	2	1	0

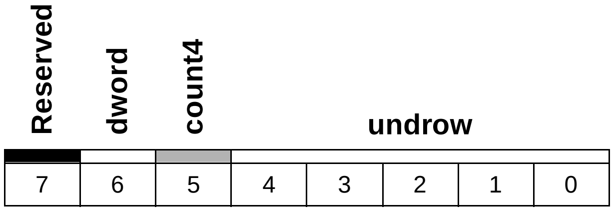
offset <7:0>

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.

CRTC14 Underline Location

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



undrow <4:0>

Horizontal row scan where the underline will occur. VGA.

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>

Reserved. When writing to this register, this field must be set to 0.

Start Vertical Blank CRTC15

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>.

CRTC16 End Vertical Blank

Index **crtcx** = 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.

Index **crtcx** = 17h

Reset Value `0000 0000`b

crtcrstN	wbmode	addwrap	Reserved	count2	hsyncsel	selrowscan	cms
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*

	{addwrap, selrowscan: cms}			
Output	*‘X00’*	*‘X01’*	*‘X10’*	*‘X11’*
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

*Word access {***dword, wbmode***} = 00*

	{addwrap, selrowscan: cms}							
Output	*‘000’*	*‘001’*	*‘010’*	*‘011’*	*‘100’*	*‘101’*	*‘110’*	*‘111’*
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, wbmode***} = 01*

	{addwrap, selrowscan: cms}			
Output	***‘X00’***	***‘X01’***	***‘X10’***	***‘X11’***
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

selrowscan **<1>**	Select row scan counter. VGA. • 0: Select the row scan counter bit 1 to be output instead of memory counter address 14. • 1: Select memory address 14 to be output. See the tables in the **cms** field's description.
hsyncsel **<2>**	Horizontal retrace select. VGA/MGA. • 0: The vertical counter is clocked on every horizontal retrace. • 1: The vertical counter is clocked on every horizontal retrace divided by 2. 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.
count2 **<3>**	Count by 2. VGA/MGA. • 0: The **count4** field (**CRTC14**<5>) dictates if the character clock is divided by 4 (**count4** = 1) or by 1 (**count4** = 0). • 1: The memory address counter is clocked with the character clock divided by 2 (**count4** is 'don't care' in this case).
addwrap **<5>**	Address wrap. VGA. • 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. • 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 **cms** field's description.
wbmode **<6>**	Word/byte mode. VGA. • 0: When not in double word mode (**dword** (**CRTC14**<6>) = 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 **cms** field's description. • 1: Select byte mode. The memory address counter bits are applied directly to the video memory.
crtcrstN **<7>**	**CRTC** reset. VGA/MGA. • 0: Force the horizontal and vertical sync to be inactive. • 1: Allow the horizontal and vertical sync to run.
Reserved **<4>**	Reserved. When writing to this register, this field must be set to 0. Reading will give 0's.

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.

Index **crtcx** = 22h

Reset Value `0000 0000`b

cpudata							
7	6	5	4	3	2	1	0

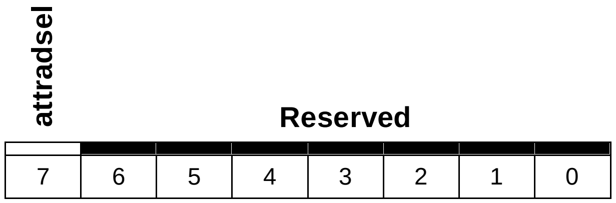
cpudata <7:0>

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 **rdmapsl** 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.

Index **crtcx** = 24h

Reset Value `0000 0000`b



attradsel **<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.
Reserved **<6:0>**	Reserved. When writing to this register, the bits in this field must be set to 0.

Index **crtcx** = 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>	Reserved. When writing to this register, the bits in this field must be set to 0.

- See the **ATTR** register on page 4-86.

CRTC Extension CRTCEXT

Address 03DEh (I/O), **MGABASE1** + 1FDEh (MEM)

Attributes R/W, BYTE/WORD, STATIC

Reset Value `nnnn nnnn 0000 0000`b

crtcextd								Reserved					crtcextx		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

crtcextx <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 **crtcextd** field is accessed.

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

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

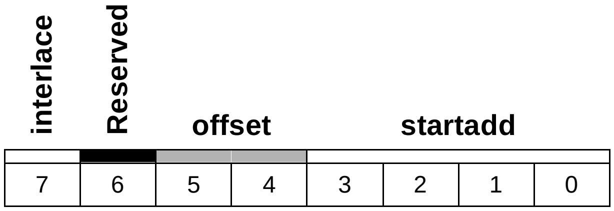
crtcextd <15:8> CRTC extension data register.

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

Reserved <7:3> Reserved. When writing to this register, the bits in this field must be set to 0.

Index **crtcextx** = 00h

Reset Value `0000 0000b`



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 4-110.
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 4-117.
interlace <7>	Interlace enable. Indicates if interlace mode is enabled. • 0: Not in interlace mode. • 1: Interlace mode.
Reserved <6>	Reserved. When writing to this register, this field must be set to 0.

Index **crtcextx** = 01h

Reset Value `0000 0000b`

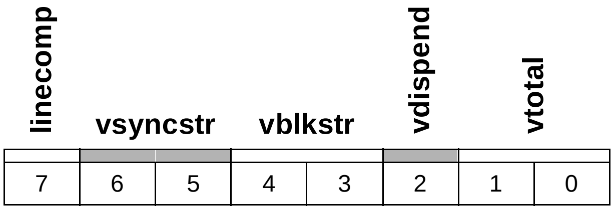
vrsten	hblkend	vsyncoff	hsyncoff	hrsten	hsyncstr	hblkstr	htotal
7	6	5	4	3	2	1	0

htotal **<0>**	Horizontal total bit 8. This is the most significant bit of the **htotal** (horizontal total) register. See the **CRTC0** register on page 4-98.
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 4-100.
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 4-102.
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 4-101.
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

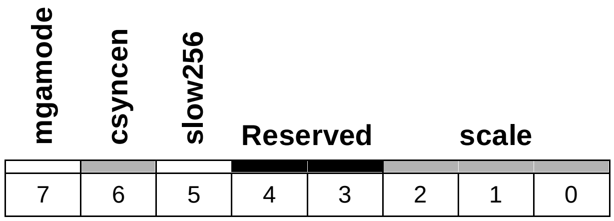
Reset Value `0000 0000b`



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 4-104.
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 4-116.
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 4-119.
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 4-114.
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 4-125.

Index **crtcextx** = 03h

Reset Value `0000 0000`b



scale<2:0> Video clock scaling factor. Specifies the video clock division factor in MGA mode.

Scale	*Division Factor*
‘000’	/1
‘001’	/2
‘010’	/3
‘011’	/4
‘100’	Reserved
‘101’	/6
‘110’	Reserved
‘111’	/8

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 or IOG output.

- 0: Horizontal sync.
- 1: Composite sync (block sync).

mgamode <7> MGA mode enable.

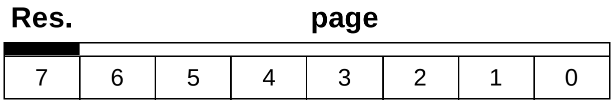
- 0: Select VGA compatibility mode. In this mode, VGA data is sent to the DAC via the VGA attribute controller. The memory address counter clock will be selected by the **count2** (**CRTC17**<3> and **count4** (**CRTC14**<5>) bits. This mode should be used for all VGA modes up to mode 13, and for all Super VGA alpha modes. When **mgamode** = ‘0’, the full frame buffer aperture mapped to **MGABASE2** is unusable.
- 1: Select MGA mode. In this mode, the graphics engine data is sent directly to the DAC. The memory address counter is clocked, depending on the state of the **hzoom** field of the **XZOOMCTRL** register. This mode should be used for all Super VGA graphics modes and all accelerated graphics modes.

Reserved <4:3> Reserved. When writing to this register, the bits in this field must be set to 0.

CRTCEXT4 Memory Page

Index **crtcextx** = 04h

Reset Value `0000 0000`b



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.

GCTL6<3:2>	*Bits used to address RAM*	*Comment*
‘00’	**CRTCEXT4**<6:1>, CPUA<16:0>	128K window
‘01’	**CRTCEXT4**<6:0>, CPUA<15:0>	64K window
‘1X’	Undefined	Window is too small

Reserved <7>

Reserved. When writing to this register, this field must be set to 0.

Index **crtcextx** = 05h

Reset Value `0000 0000`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:

$$\frac{\text{Start Horizontal Retrace + End Horizontal Retrace - Horizontal Total}}{2} - 1$$

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.

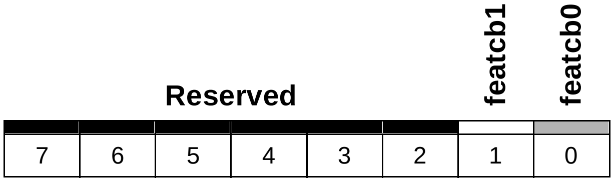
- 00: Write palette cycle
- 11: Read palette cycle

Reserved <7:2> This field returns all zeroes when read.

Refer to the **PALRDADD** register description on page 4-162 for information on writes to 03C7h.

Feature Control FEAT

Address	03BAh (I/O), Write (**MISC**<0> == 0: MDA emulation) 03DAh (I/O), Write (**MISC**<0> == 1: CGA emulation) 03CAh (I/O) Read **MGABASE1** + 1FDAh (MEM)
Attributes	R/W, BYTE, STATIC
Reset Value	`0000 0000b`



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>	Reserved. When writing to this register, the bits in this field must be set to 0.

GCTL Graphics Controller

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 <3:0> Graphics controller index register.

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.

Register name	*Mnemonic*	**gctlx** *address*
Set/Reset	**GCTL0**	00h
Enable Set/Reset	**GCTL1**	01h
Color Compare	**GCTL2**	02h
Data Rotate	**GCTL3**	03h
Read Map Select	**GCTL4**	04h
Graphic Mode	**GCTL5**	05h
Miscellaneous	**GCTL6**	06h
Color Don’t Care	**ATTR13**	07h
Bit Mask	**GCTL8**	08h
Reserved ((1))	---	09h - 0Fh

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

gctld <15:8> Graphics controller data register.

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

Reserved <7:4> Reserved. When writing to this register, the bits in these fields must be set to 0.

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.

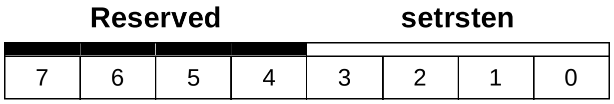
Reserved <7:4>

Reserved. When writing to this register, the bits in these fields must be set to 0.

GCTL1 Enable Set/Reset

Index **gctlx** = 01h

Reset Value `0000 0000`b



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> Reserved. When writing to this register, the bits in these fields must be set to 0.

Color Compare GCTL2

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 4-147) register will be tested.

Reserved <7:4> Reserved. When writing to this register, the bits in these fields must be set to 0.

GCTL3 Data Rotate

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 4-148) 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).

funsel	*Function*
'00'	Source unmodified
'01'	Source AND latched data
'10'	Source OR latched data
'11'	Source XOR latched data

Reserved <7:5>

Reserved. When writing to this register, the bits in these fields must be set to 0.

Read Map Select

GCTL4

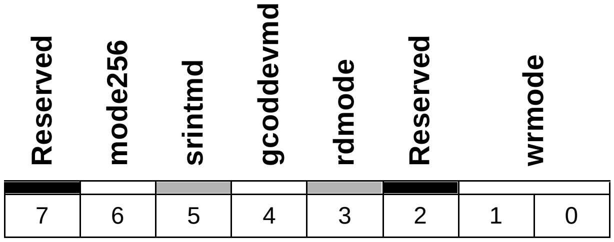
Index	**gctlx** = 04h
Reset Value	`0000 0000`b

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>**	Reserved. When writing to this register, the bits in these fields must be set to 0.

Index **gctlx** = 05h

Reset Value `0000 0000b`



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 4-148) 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>

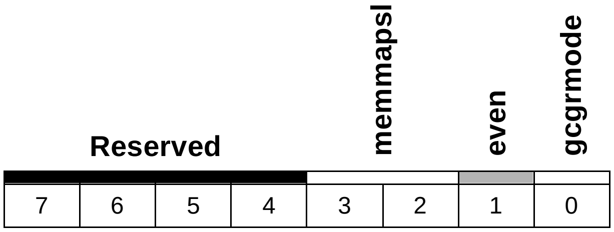
Shift register interleave mode. VGA.

- 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.

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	**<2> <7>** Reserved. When writing to this register, the bits in these fields must be set to 0.. These fields return all zeroes when read.

Index **gctlx** = 06h

Reset Value `0000 0000b`



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 4-151).

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’	A0000h - BFFFFh
‘01’	A0000h - AFFFFh
‘10’	B0000h - B7FFFh
‘11’	B8000h - BFFFFh

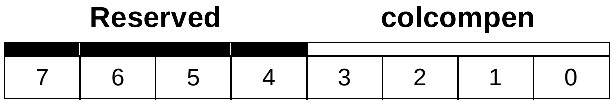
Reserved <7:4>

Reserved. When writing to this register, the bits in these fields must be set to 0.

Color Don’t Care GCTL7

Index **gctlx** = 07h

Reset Value `0000 0000`b



colcompen <3:0>	Color enable comparison for planes 3 to 0. VGA. When any of these bits are set to ‘1’, the associated plane is included in the color compare read cycle.
Reserved <7:4>	Reserved. When writing to this register, the bits in these fields must be set to 0.

GCTL8 Bit Mask

Index **gctlx** = 08h

Reset Value `0000 0000`b

wrmask

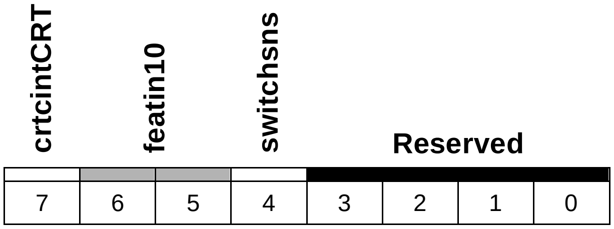
7	6	5	4	3	2	1	0

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.

Input Status INSTS0

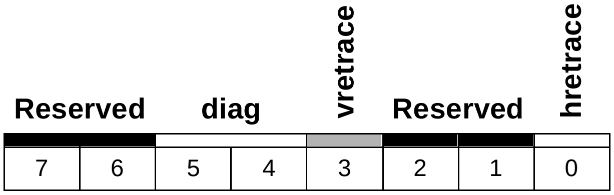
Address	03C2h (I/O), **MGABASE1** + 1FC2h (MEM) Read
Attributes	RO, BYTE, STATIC
Reset Value	`?111 0000b`



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.
Reserved <3:0>	Reserved. When writing to this register, the bits in these fields must be set to 0.

INSTS1 Input Status 1

Address	03BAh (I/O), Read (**MISC**<0> == 0: MDA emulation) 03DAh (I/O), Read (**MISC**<0> == 1: CGA emulation) **MGABASE1** + 1FDAh (MEM)
Attributes	RO, BYTE, DYNAMIC
Reset Value	Unknown



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.

vidstmx		diag	
5	*4*	*5*	*4*
‘0’	‘0’	PD2	PD0
‘0’	‘1’	PD5	PD4
‘1’	‘0’	PD3	PD1
‘1’	‘1’	PD7	PD6

Reserved **<2:1> <7:6>**

Reserved. When writing to this register, the bits in these fields must be set to 0. These fields return all zeroes when read.

Miscellaneous Output MISC

Address 03C2h (I/O), **MGABASE1** + 1FC2h (MEM) Write 03CCh (I/O)
MGABASE1 + 1FCCh (MEM) Read

Attributes R/W, BYTE, STATIC

Reset Value `0000 0000b`

vsyncpol	hsyncpol	hpgoddev	videodis	clksel		rammapen	ioaddsel
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 03DAh 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.

clksel <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. Selects the MGA pixel clock (see **XPIXPLLM**, **XPIXPLLN**, and **XPIXPLLP**).

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.

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.

VSYNC	*HSYNC*	*Description*
+	+	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

Sequencer SEQ

Address 03C4h (I/O), **MGABASE1** + 1FC4h (MEM)

Attributes R/W, BYTE/WORD, STATIC

Reset Value `nnnn nnnn 0000 0000`b

seqd								**Reserved**					**seqx**		
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.

Register name	*Mnemonic*	**seqx** *address*
Reset	**SEQ0**	00h
Clocking Mode	**SEQ1**	01h
Map Mask	**SEQ2**	02h
Character Map Select	**SEQ3**	03h
Memory Mode	**SEQ4**	04h
Reserved (⁽¹⁾)	---	05h - 07h

(1) When writing to a reserved register, all fields must be set to 0. Reading from a reserved index will give 0’s.

seqd <15:8> Sequencer data register.

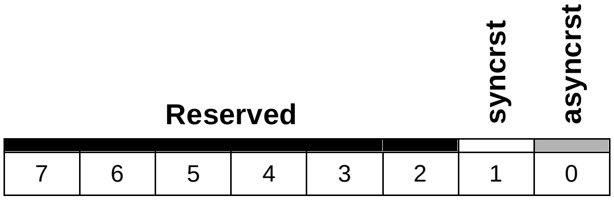
Retrieve or write the contents of the register that is pointed to by the **seqx** field.

Reserved <7:3> Reserved. When writing to this register, the bits in these fields must be set to 0.

SEQ0 Reset

Index **seqx** = 00h

Reset Value `0000 0000`b



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-1064SG 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> Reserved. When writing to this register, the bits in these fields must be set to 0.

Index **seqx** = 01h

Reset Value `0000 0000`b

Reserved		scroff	shiftfour	dotclkrt	shftldrt	Reserved	dotmode
7	6	5	4	3	2	1	0

dotmode <0> 9/8 dot mode. VGA.

- 0: The sequencer generates a 9-dot character clock.
- 1: The sequencer generates an 8-dot character clock.

shftldrt <2> Shift/load rate. VGA.

- 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 <3> Dot clock rate. VGA.

- 0: The dot clock rate is the same as the clock at the VDCLK pin.
- 1: The dot clock rate is slowed to one-half the clock at the VDCLK pin. The character clock and shift/load signals are also slowed to half their normal speed.

shiftfour <4> Shift four. VGA.

- 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 <5> Screen off. VGA/MGA.

- 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.

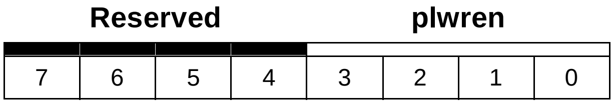
Reserved **<1> <7:6>**

Reserved. When writing to this register, the bits in these fields must be set to 0. These fields return all zeroes when read.

SEQ2 Map Mask

Index **seqx** = 02h

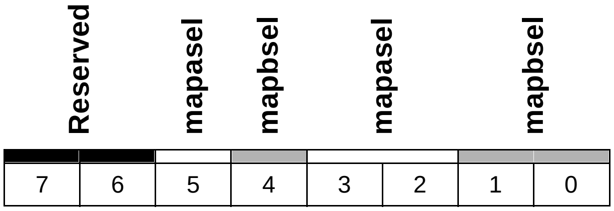
Reset Value `0000 0000`b



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>	Reserved. When writing to this register, the bits in these fields must be set to 0.

Index **seqx** = 03h

Reset Value `0000 0000`b



This register is reset by the reset pin (PRST/), or by the **asyncrst** 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:

mapbsel	*Map#*	*Map location*
'000'	0	1st 8 KBytes of Map 2
'001'	1	3rd 8 KBytes of Map 2
'010'	2	5th 8 KBytes of Map 2
'011'	3	7th 8 KBytes of Map 2
'100'	4	2nd 8 KBytes of Map 2
'101'	5	4th 8 KBytes of Map 2
'110'	6	6th 8 KBytes of Map 2
'111'	7	8th 8 KBytes 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:

mapasel	*Map#*	*Map location*
'000'	0	1st 8 KBytes of Map 2
'001'	1	3rd 8 KBytes of Map 2
'010'	2	5th 8 KBytes of Map 2
'011'	3	7th 8 KBytes of Map 2
'100'	4	2nd 8 KBytes of Map 2
'101'	5	4th 8 KBytes of Map 2
'110'	6	6th 8 KBytes of Map 2
'111'	7	8th 8 KBytes of Map 2

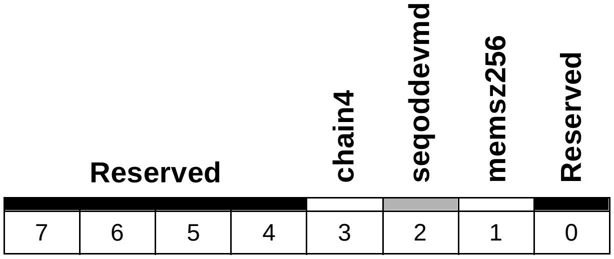
Reserved <7:6>

Reserved. When writing to this register, the bits in these fields must be set to 0.

SEQ4 Memory Mode

Index **seqx** = 04h

Reset Value `0000 0000b`



memsz256 <1> 256K memory size.

- 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 <2> Odd/Even mode. VGA.

- 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 <3> Chain four. VGA.

- 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 **<0> <7:4>**

Reserved. When writing to this register, the bits in these fields must be set to 0. These fields return all zeroes when read.

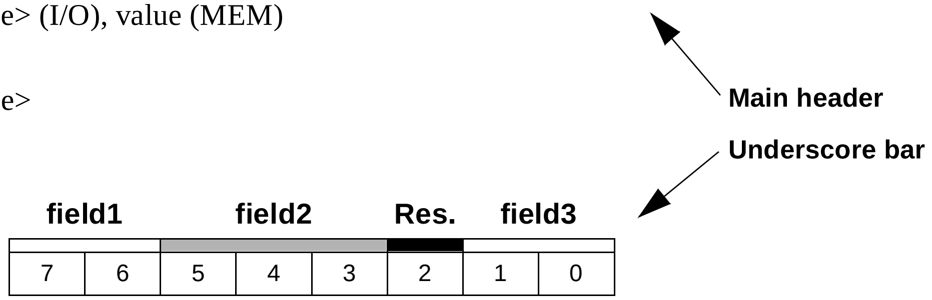
4.3 DAC Registers

4.3.1 DAC Register Descriptions

The MGA-1064SG DAC register descriptions contain a (gray 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 DAC Register Description SAMPLE_DAC

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



field1 **<7:6>**	Field 1. Detailed description of the **field1** field of the **SAMPLE_DAC** register, which comprises bits 7 to 6. ***Note the font and case changes which indicate a register or field in the text.***
field2 **<5:3>**	Field 2. Detailed description of the **field2** field of **SAMPLE_DAC**, which comprises bits 5 to 3.
field3 **<1:0>**	Field 3. Detailed description of the **field3** field of **SAMPLE_DAC**, which comprises bits 1 to 0.
Reserved<2>	Reserved. When writing to this register, this field must be set to 0. (Reserved registers always appear at the end of a register description.)

Address

This address is an offset from the Power Graphic mode base memory address.

Index

The index is an offset from the starting address of the indirect access register (**X_DATAREG**).

Attributes

The DAC register attributes are:

- RO: There are no writable bits.
- WO: There are no readable 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.

Reset Value

Here are some of the symbols that appear as part of a register's reset value.

- `000? 0?00b`
  (b = binary, ? = unknown, N/A = not applicable)

Address	**CURPOSXL** **MGABASE1** + 3C0Ch (MEM)
	CURPOSXH **MGABASE1** + 3C0Dh (MEM)
	CURPOSYL **MGABASE1** + 3C0Eh (MEM)
	CURPOSYH **MGABASE1** + 3C0Fh (MEM)
Attributes	R/W, BYTE, WORD, DWORD
Reset Value	Unknown

Reserved				**curposy**												**Reserved**				**curposx**											
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

curposx <11:0>

X Cursor position. Determines the position of the last column of the hardware cursor on the display screen.

In order to avoid either noise or a split cursor within a frame, the software must ensure that a cursor update never occurs during a retrace period (when the **vsyncsts** status is 1 - see the **STATUS** register). Cursor update can take place at any other time.

Cursor repositioning will only take effect on the next activation of the vertical retrace.

Cursor position (1,1) corresponds to the top left corner of the screen (it is the first displayed pixel following a vertical retrace).

Specifically, the programmed cursor x position is the number of pixels from the end of the blank signal at which the bottom right hand corner of the cursor is located. Therefore, loading 001h into **curposx** causes the rightmost pixel of the cursor to be displayed on the leftmost pixel of the screen.

Likewise, the programmed cursor y position is the number of scanlines from the end of vertical blanking at which the bottom right hand corner of the cursor is located. Therefore, loading 001h into **curposy** causes the bottommost pixel of the cursor to be displayed on the top scanline of the screen. If 000h is written to either of the cursor position registers, the cursor will be located off-screen.

The hardware cursor is operational in both non-interlace and interlaced display modes (see the **interlace** bit of the **CRTCEXT0** VGA register).

curposy <27:16>

Y Cursor position. Determines the position of the last row of the hardware cursor on the display screen.

Reserved **<5:12> <31:28>**

Reserved. When writing to this register, the bits in these fields must be set to 0.

Palette RAM Data PALDATA

Address 03C9h (I/O), **MGABASE1** + 3C01h (MEM)

Attributes R/W, BYTE

Reset Value Unknown

paldata

7	6	5	4	3	2	1	0

paldata <7:0>

Palette RAM data. This register is used to load data into and read data from the palette RAM. Since the palette RAM is 24 bits wide, three writes are required to this register in order to write one complete location in the RAM. The address in the palette RAM to be written is determined by the value of the **PALWTADD** register.

Likewise, three reads are required to obtain all three bytes of data in one entry. The address in the palette RAM to be read is determined by the value of the **PALRDADD** register.

The **vga8dac** bit (see the **XMISCCTRL** register) controls how the data is written into the palette.

- In 6-bit mode, the host data is shifted left by two to compensate for the lack of a sufficient bit width (zeros are shifted in). When reading data from the palette RAM in 6-bit mode, the data will be shifted right and the two most significant bits filled with 0’s to be compatible with VGA.
- In 8-bit mode, no shifting or zero padding occurs; the full 8 bit host data is transferred.

The palette RAM is dual-ported, so reading or writing will not cause any noticeable disturbance of the display.

PALRDADD Palette RAM Read Address

Address	03C7h (I/O, W), **MGABASE1** + 3C03h (MEM, R/W)
Attributes	BYTE
Reset Value	Unknown

palrdadd

7	6	5	4	3	2	1	0

palrdadd <7:0> Palette address register for LUT read. This register is used to address the palette RAM during a read operation. It is incremented for every three bytes read from the **PALDATA** port (the palette RAM is 24 bits wide). When the address increments beyond the last palette location, it will wrap around to location 0. Writing this register resets the modulo 3 counter to 0.

Note: This location stores the same physical register as location **PALWTADD**.

Palette RAM Write Address PALWTADD

Address	03C8h (I/O), **MGABASE1** + 3C00h (MEM)
Attributes	R/W, BYTE
Reset Value	Unknown

palwtadd							
7	6	5	4	3	2	1	0

palwtadd <7:0> Palette address register for LUT write. This register has two functions:

- It is used to address the palette RAM during a write operation. This register is incremented for every 3 bytes written to the **PALDATA** port (the palette RAM is 24 bits wide). When the address increments beyond the last palette location, it will wrap around to location 0. Writing this register resets the modulo 3 counter to 0.
- When used as the index register, this register is loaded with the index of the indirect register which is to be accessed through the **X_DATAREG** port.

Note: This location stores the same physical register as the **PALRDADD** location.

Address	03C6h (I/O), **MGABASE1** + 3C02h (MEM)
Attributes	R/W, BYTE
Reset Value	`1111 1111`h

pixrdmsk							
7	6	5	4	3	2	1	0

pixrdmsk <7:0>

Pixel read mask. The pixel read mask register is used to enable or disable a bit plane from addressing the palette RAM. This mask is used in all modes which access the palette RAM (not just the 8 bit/pixel modes).

Each palette address bit is logically ANDed with the corresponding bit from the read mask register before going to the palette RAM. Note that direct pixels (pixels that do not go through the palette RAM) are not masked in any mode.

Indexed Data X_DATAREG

Address **MGABASE1** + 3C0Ah (MEM)
Attributes R/W, BYTE
Reset Value Unknown

indd							
7	6	5	4	3	2	1	0

indd <7:0> Indexed data register. This is the register that is used to read from or write to any of the valid indexed (indirect) registers. See the following register descriptions. The address which is accessed is determined by the Index Register (**PALWTADD**).

Locations marked as ‘Reserved’ return unknown information; writing to any such reserved location may affect other indexed registers.

Index	*Register Addressed*	*Mnemonic*
00h-03h	Reserved	
04h	Cursor Base Address Low	**XCURADDL**
05h	Cursor Base Address High	**XCURADDH**
06h	Cursor Control	**XCURCTRL**
07h	Reserved	
08h	Cursor Color 0 RED	**XCURCOL0RED**
09h	Cursor Color 0 GREEN	**XCURCOL0GREEN**
0Ah	Cursor Color 0 BLUE	**XCURCOL0BLUE**
0Bh	Reserved	
0Ch	Cursor Color 1 RED	**XCURCOL1RED**
0Dh	Cursor Color 1 GREEN	**XCURCOL1GREEN**
0Eh	Cursor Color 1 BLUE	**XCURCOL1BLUE**
0Fh	Reserved	
10h	Cursor Color 2 RED	**XCURCOL2RED**
11h	Cursor Color 2 GREEN	**XCURCOL2GREEN**
12h	Cursor Color 2 BLUE	**XCURCOL2BLUE**
13h-17h	Reserved	
18h	Voltage Reference Control	**XVREFCTRL**
19h	Multiplex Control	**XMULCTRL**
1Ah	Pixel Clock Control	**XPIXCLKCTRL**
1Bh-1Ch	Reserved	
1Dh	General Control	**XGENCTRL**
1Eh	Miscellaneous Control	**XMISCCTRL**
1Fh-29h	Reserved	
2Ah	General Purpose I/O Control	**XGENIOCTRL**
2Bh	General Purpose I/O Data	**XGENIODATA**
2Ch	SYSPLL M Value	**XSYSPLLM**
2Dh	SYSPLL N Value	**XSYSPLLN**
2Eh	SYSPLL P Value	**XSYSPLLP**
2Fh	SYSPLL Status	**XSYSPLLSTAT**

indd *Address*	*Register Addressed*	*Mnemonic*
30h-37h	Reserved	
38h	Zoom Control	**XZOOMCTRL**
39h	Reserved	
3Ah	Sense Test	**XSENSETEST**
3Bh	Reserved	
3Ch	CRC Remainder LSB	**XCRCREML**
3Dh	CRC Remainder MSB	**XCRCREMH**
3Eh	CRC Bit Select	**XCRCBITSEL**
3Fh	Reserved	
40h	Color Key Mask Low	**XCOLKEYMSKL**
41h	Color Key Mask High	**XCOLKEYMSKH**
42h	Color Key Low	**XCOLKEYL**
43h	Color Key High	**XCOLKEYH**
44h	PIXPLL M Value Set A	**XPIXPLLAM**
45h	PIXPLL N Value Set A	**XPIXPLLAN**
46h	PIXPLL P Value Set A	**XPIXPLLAP**
47h	Reserved	
48h	PIXPLL M Value Set B	**XPIXPLLBM**
49h	PIXPLL N Value Set B	**XPIXPLLBN**
4Ah	PIXPLL P Value Set B	**XPIXPLLBP**
4Bh	Reserved	
4Ch	PIXPLL M Value Set C	**XPIXPLLCM**
4Dh	PIXPLL N Value Set C	**XPIXPLLCN**
4Eh	PIXPLL P Value Set C	**XPIXPLLCP**
4Fh	PIXPLL Status	**XPIXPLLSTAT**
50h-FFh	Reserved	

Color Key High XCOLKEYH

Index 43h

Attributes R/W, BYTE

Reset Value Unknown

colkey							
7	6	5	4	3	2	1	0

colkey <7:0> Color key bits 15 to 8. The color key is used to perform color keying between graphics and the video buffer.

Note: In 2G8V16 and 32 bits/pixel single frame buffer modes (**depth** = ‘100’), the contents of this register should be set to all zeroes. (The **depth** field is located in the **XMULCTRL** register.)

The switching for keying is performed according to the following equation:

```
if ((ALPHAEN & VP<15>) == VS)      ;show graphics buffer
elsif ((COLKEYMSK & GP) == COLKEY) ;show video buffer
else                               ;show graphics buffer
```

where:

`VP<15>`	is bit 15 of the video stream
`GP`	is the graphics stream
vs	is the video select polarity (see the **XGENCTRL** register)
alphaen	is the alpha plane enable bit (see the **XGENCTRL** register)
COLKEYMSK	is the 16-bit key mask from the **XCOLKEYMSKH** and **XCOLKEYMSKL** registers
COLKEY	is the 16-bit key from the **XCOLKEYH** and **XCOLKEYL** registers

✎ Note: To always choose the graphics stream when in split frame buffer mode, program **vs** = 0 and **alphaen** = 0. To always choose the video stream when in split frame buffer mode, program **vs** = 1, **alphaen** = 0, **XCOLKEY** = 0, and **XCOLKEYMSK** = 0.

The color key is also used for overlay keying in 32 bits/pixel single frame buffer mode (**depth** = ‘100’) to specify the transparent color. The equation is:

```
if (COLKEYMSK & ALPHA == COLKEY) ;show the 24-bit direct stream
else                             ;show the overlay color LUT(ALPHA))
```

where:

ALPHA is the address of the overlay register (in that mode, the palette is used as 256 overlay registers) and LUT(ALPHA) is the overlay color.

The overlay can be disabled by programming COLKEYMSK = 0 and COLKEY = 0.

XCOLKEYL Color Key Low

Index	42h
Attributes	R/W, BYTE
Reset Value	Unknown

colkey

7	6	5	4	3	2	1	0

colkey <7:0>	Color key bits 7 to 0. The color key is used to perform color keying between graphics and the video buffer (see **XCOLKEYH** for more information on keying). It is also used in 32 bits/pixel single frame buffer mode (**depth** = ‘100’) to specify the transparent color. See the **XCOLKEYH** register description for more information.

Color Key Mask High XCOLKEYMSKH

Index 41h

Attributes R/W, BYTE

Reset Value Unknown

colkeymsk

7	6	5	4	3	2	1	0

colkeymsk <7:0> Color key mask bits 15 to 8. To prevent a particular bit plane from participating in a keying comparison, the corresponding color key mask bit should be set to 0b.

✒ Note: In 2G8V16 and 32 bits/pixel single frame buffer modes (**depth** = ‘100’), this register must be set to all zeroes.

Index	40h
Attributes	R/W, BYTE
Reset Value	Unknown

colkeymsk

7	6	5	4	3	2	1	0

colkeymsk <7:0>

Color key mask bits 7 to 0. To prevent a particular bit plane from participating in a keying comparison, the corresponding color key mask bit should be set to 0b.

The mask is also used in 32 bits/pixel single frame buffer modes (**depth** = ‘100’) for overlay enable/disable. See the **XCOLKEYH** register description for more information.

CRC Bit Select XCRCBITSEL

Index 3Eh

Attributes R/W, BYTE

Reset Value Unknown

Reserved			crcsel				
7	6	5	4	3	2	1	0

crcsel <4:0> CRC bit selection. This register determines which of the 24 DAC data lines the 16-bit CRC should be calculated on. Valid values are 0h-17h:

Value	*DAC Data Lines to Use*
00h-07h	blue0 - blue7
08h-0Fh	green0 - green7
10h-17h	red0 - red7

Reserved <7:5> Reserved. When writing to this register, the bits in this field must be set to 0.

XCRCREMH CRC Remainder High

Index 3Dh

Attributes RO, BYTE

Reset Value Unknown

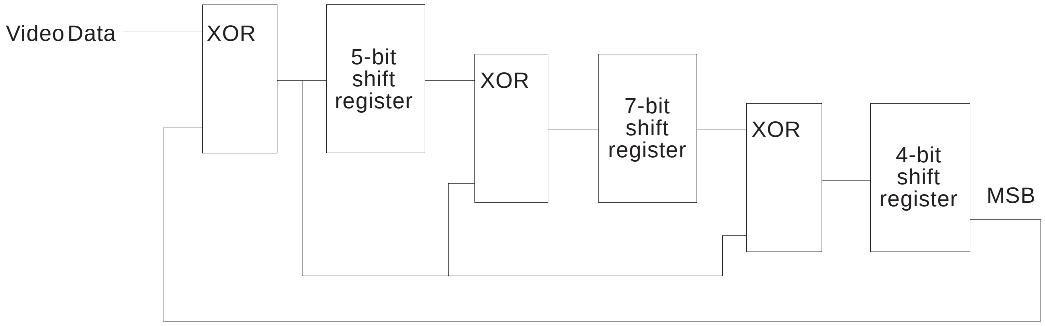
crcdata							
7	6	5	4	3	2	1	0

crcdata <7:0>

High-order CRC remainder. This register is used to read the results of the 16-bit CRC calculation. **XCRCREMH** corresponds to bits 15:8 of the 16-bit CRC.

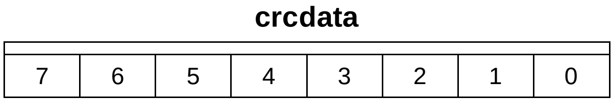
A 16-bit cyclic redundancy check (CRC) is provided so that the video data’s integrity can be verified at the input of the DACs. The **XCRCBITSEL** register indicates which video line is checked. The **XCRCREMH** and **XCRCREML** registers accept video data when the screen is not in the blank period. The CRC Remainder register is reset to 0 at the end of vertical sync period and must be read at the beginning of the next vertical sync period (when VSYNC status goes to 1).

The CRC is calculated as follows:



CRC Remainder Low XCRCREML

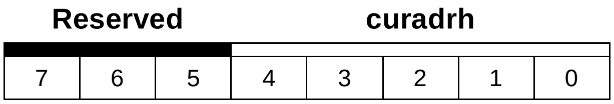
Index	3Ch
Attributes	RO, BYTE
Reset Value	Unknown



crcdata <7:0>	Low-order CRC remainder. This register is used to read the results of the 16-bit CRC calculation. **XCRCREML** corresponds to bits 7:0 of the 16-bit CRC. See **XCRCREMH**.

XCURADDH Cursor Base Address High

Index	05h
Attributes	R/W, BYTE
Reset Value	Unknown



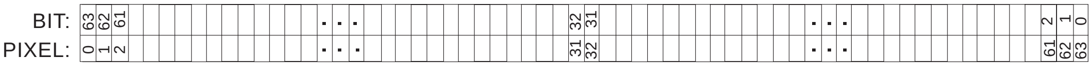
curadrh <4:0>

Cursor address high. These are the high-order bits of the cursor map address.

The 13-bit value from the high and low order cursor address locations is the base address (bits 22:10) of the frame buffer where the cursor maps are located. The cursor maps must be aligned on a 1 KByte boundary.

When **XCURADDL** or **XCURADDH** are written, the values take effect immediately. This may result in temporarily invalid cursor pixel values, if the cursor map is being fetched simultaneously.

The 64 x 64 x 2 cursor map is used to define the pixel pattern within the 64 x 64 pixel cursor window. Each pixel of the cursor is defined by two bits, referred to as bit plane 1 and bit plane 0. The cursor data is stored in 64-bit slices in memory (each slice contains all of the data for one plane of one scanline of the cursor). One plane of one scanline is stored in memory as follows:



Assuming that the entire scanline of the cursor is displayed, cursor data is displayed from bit 63 to bit 0. To facilitate fetching of cursor data from memory, slices alternate between plane 0 and plane 1. The cursor data is organized in memory as follows:

64-bit Address (Qword)	*Data*
Base + 0	Line 0, Plane 0
Base + 1	Line 0, Plane 1
Base + 2	Line 1, Plane 0
Base + 3	Line 1, Plane 1
.	.
.	.
Base + 126	Line 63, Plane 0
Base + 127	Line 63, Plane 1

Reserved <7:5>

Reserved. When writing to this register, the bits in this field must be set to 0.

Cursor Base Address Low XCURADDL

Index	04h
Attributes	R/W, BYTE
Reset Value	Unknown

curadrl							
7	6	5	4	3	2	1	0

curadrl <7:0>	Cursor address low. These are the low-order bits of the cursor map address. See the **XCURADDH** register description for more details.

XCURCOL Cursor Color 0, 1, 2

Index	08h	**XCURCOL0RED**
	09h	**XCURCOL0GREEN**
	0Ah	**XCURCOL0BLUE**
	0Ch	**XCURCOL1RED**
	0Dh	**XCURCOL1GREEN**
	0Eh	**XCURCOL1BLUE**
	10h	**XCURCOL2RED**
	11h	**XCURCOL2GREEN**
	12h	**XCURCOL2BLUE**
Attributes	R/W, BYTE	
Reset Value	Unknown	

curcol							
7	6	5	4	3	2	1	0

curcol <7:0> Cursor color register. The desired color register (0, 1, or 2) is chosen according to both the cursor mode and cursor map information. (See the **XCURCTRL** register for more information.) Each color register is 24 bits wide and contains an 8-bit red, 8-bit green, and 8-bit blue field.

Index 06h

Attributes R/W, BYTE

Reset Value `0000 0000`b

Reserved						curmode	
7	6	5	4	3	2	1	0

curmode <1:0> Cursor mode select. This field is used to disable or select the cursor mode, as shown below:

- 00 = cursor disabled (default)
- 01 = three-color cursor
- 10 = XGA cursor
- 11 = X-Windows cursor

Since the cursor maps (located in the frame buffer at the location pointed to by **XCURADDH** and **XCURADDL**) use two bits to represent each pixel of the 64 x 64 cursor, there are four possible ways to display each pixel of the cursor. The following table shows how the encoded pixel data is decoded, based on the cursor mode (set by **curmode**):

RAM		*Cursor Mode*		
Plane 1	*Plane 0*	*Three-Color*	*XGA*	*X-Windows*
‘0’	‘0’	Transparent(1)	Cursor Color 0	Transparent
‘0’	‘1’	Cursor Color 0	Cursor Color 1	Transparent
‘1’	‘0’	Cursor Color 1	Transparent	Cursor Color 0
‘1’	‘1’	Cursor Color 2	Complement(2)	Cursor Color 1

(1) The underlying pixel is displayed (that is, the cursor has no effect on the display).

(2) Each bit of the underlying pixel is inverted, then displayed.

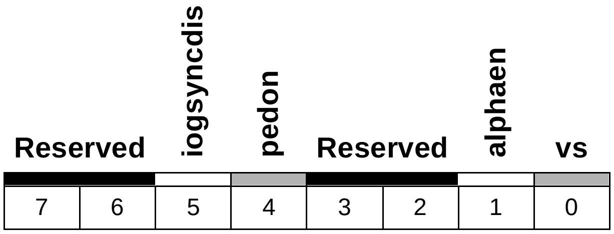
Reserved <7:2> Reserved. When writing to this register, the bits in this field must be set to 0.

XGENCTRL General Control

Index 1Dh

Attributes R/W, BYTE

Reset Value `0000 0000b`



vs <0>

Video select. This bit is used by the keying function to select the polarity of the alpha comparison in the keying equation (see the **XCOLKEYH** register).

This field must be set to 0 when not in split frame buffer mode. (that is, not in mode G16V16 or 2G8V16).

- 0: show graphics stream if alpha is 0 or masked.
- 1: show graphics stream if alpha is 1 and not masked.

alphaen <1>

Video alpha bit enable. This bit is used by the keying function to enable or disable the alpha bits in the equation for split frame buffer modes (mode G16V16 or 2G8V16). It is also used in 15-bit single frame buffer mode to enable or disable the 1-bit overlay.

- 0: disabled (forces the effective value of all alpha bits to 0b) or overlay disable
- 1: enabled (alpha bits are used for color keying) or overlay enable

pedon <4>

Pedestal control. This field specifies whether a 0 or 7.5 IRE blanking pedestal is to be generated on the video outputs.

- 0: 0 IRE (default)
- 1: 7.5 IRE

iogsyncdis <5>

Green channel sync disable. This field specifies if sync (from the internal signal HSYNC) information is to be sent to the output of the green DAC.

- 0: enable (default)
- 1: disable

Note: The HSYNC can be programmed to be either horizontal sync only, or composite (block) sync. See the **csyncen** bit of the **CRTCEXT3** VGA register.

Reserved **<3:2> <7:6>**

Reserved. When writing to this register, the bits in these fields must be set to 0.

General Purpose I/O Control XGENIOCTRL

Index 2Ah

Attributes R/W, BYTE

Reset Value `0000 0000`b

Reserved		miscoe		ddcoe			
7	6	5	4	3	2	1	0

ddcoe <3:0>	DDC pin output control. Controls the output enable of the driver on pins DDC<3:0>, respectively. • 0: disable the output driver • 1: enable the output driver
miscoe <5:4>	MISC pin output control. Controls the output enable of the driver on pins MISC<1:0>, respectively. • 0: disable the output driver • 1: enable the output driver
Reserved <7:6>	Reserved. When writing to this register, the bits in this field must be set to 0.

XGENIODATA General Purpose I/O Data

Index 2Bh

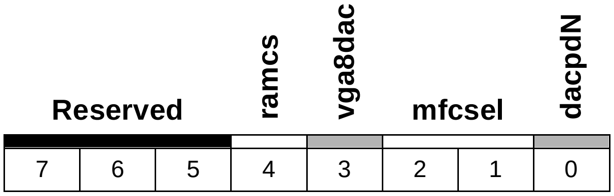
Attributes R/W, BYTE

Reset Value `0000 0000`b

Reserved		miscdata		ddcdata			
7	6	5	4	3	2	1	0

ddcdata <3:0>	DDC pin output state. Controls the output state of the driver on pins DDC<3:0>, respectively. On read, this field returns the state of the DDC<3:0> pins.
miscdata <5:4>	MISC pin output state. Controls the output state of the driver on pins MISC<1:0> during a write operation. On read, this field returns the state of the MISC<1:0> pins.
Reserved <7:6>	Reserved. When writing to this register, the bits in this field must be set to 0.

Index	1Eh
Attributes	R/W, BYTE
Reset Value	`0000 0000`b



dacpdN <0>

DAC power down. This field is used to remove power from the DACs, to conserve power.

- 0: DAC disabled (default)
- 1: DAC enabled

mfcsel <2:1>

Matrox Advanced Feature Connector (MAFC) function select. Selects the mode in which the MAFC will operate.

- 00: standard VGA connector (output only)

  In standard VGA mode (this is the reset mode), the VD<15:8> outputs are tri-stated, and straps on these pins can be input to the chip.

- 01: Matrox Advanced Feature Connector

  In MAFC mode, the output just before the DAC, plus the 8-bit alpha channel are output on the 16-bit feature connector using both edges of the clock. This effectively transfers one 32-bit pixel RGBA/8888 per clock. The alpha portion is only valid in 32 bit/pixel display modes (**depth** = ‘100’). (See the **depth** field in **XMULCTRL**.)

  Note: The alpha portion does not pass through the LUT.

- 10: reserved
- 11: disable feature connector

  When the feature connector is disabled, the VD<15:8> outputs are tri-stated, and the VD<7:0>, VBLANK/, and VDCLK pins are driven low, regardless of the state of the VEVIDEO, VESYNC, and VEDCLK pins (that is, it is not possible to tri-state VD<7:0>, VBLANK/, and VDCLK when **mfcsel** = ‘11’).

vga8dac <3>

VGA 8-bit DAC. This field is used for compatibility with standard VGA, which uses a 6-bit DAC.

- 0: 6 bit palette (default)
- 1: 8 bit palette

ramcs <4>

LUT RAM chip select. Used to power up the LUT.

- 0: LUT disabled
- 1: LUT enabled

Reserved <7:5>

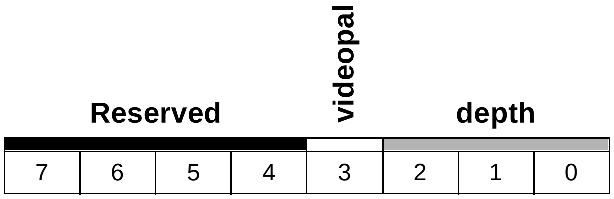
Reserved. When writing to this register, the bits in this field must be set to 0.

XMULCTRL — Multiplex Control

Index 19h

Attributes R/W, BYTE

Reset Value `0000 0000`b



depth <2:0> Color depth. The following table shows the available color depths and their properties:

Value	*Color Depth Used*	
‘000’	8 bits/pixel	(palettized) (default)
‘001’	15 bits/pixel	(palettized) + 1-bit overlay
‘010’	16 bits/pixel	(palettized)
‘011’	24 bits/pixel	(packed, palettized)
‘100’	32 bits/pixel	(24 bpp direct, 8 bpp overlay palettized)
‘101’	16 bits/pixel 2G8V16	(15 bpp video direct, 8 bpp graphics palettized, video half-resolution)
‘110’	32 bits/pixel G16V16	(15 bpp video, 15 bpp graphics) One of the pixels is palettized (refer to **videopal**).
‘111’	32 bits/pixel	(24 bpp palettized, 8 bpp unused)

The **mgamode** field of the **CRTCEXT3** VGA register overrides the **depth** field and sets the DAC to VGA mode (when at ‘0’).

videopal <3> Palette source in G16V16 mode. In G16V16 mode, this bit indicates the source that goes through the palette:

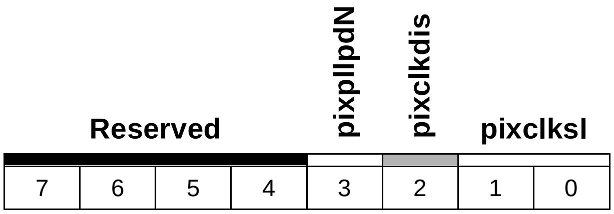
- 0: graphics go through the palette
- 1: video goes through the palette

Reserved <7:4> Reserved. When writing to this register, the bits in this field must be set to 0.

Note: The **depth** and **mgamode** fields also control the VCLK division factor.

Pixel Clock Control XPIXCLKCTRL

Index	1Ah
Attributes	R/W, BYTE
Reset Value	`0000 0000`b



pixclksl <1:0> Pixel clock selection. These bits select the source of the pixel clock:

- 00: selects the output of the PCI clock
- 01: selects the output of the pixel clock PLL
- 10: selects external source (from the VDCLK pin)
- 11: reserved

pixclkdis <2> Pixel clock disable. This bit controls the pixel clock output:

- 0: enable pixel clock oscillations.
- 1: stop pixel clock oscillations.

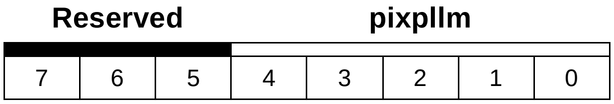
pixpllpdN<3> Pixel PLL power down.

- 0: power down
- 1: power up

Reserved <7:4> Reserved. When writing to this register, the bits in this field must be set to 0.

See ‘Programming the PLLs on page 5-77 for information on modifying the clock parameters.

Index	44h	**XPIXPLLAM**
	48h	**XPIXPLLBM**
	4Ch	**XPIXPLLCM**
Attributes	R/W, BYTE	
Reset Value	1Eh	**XPIXPLLAM**
	19h	**XPIXPLLBM**
	Unknown	**XPIXPLLCM**



pixpllm <4:0> Pixel PLL M value register. The ‘m’ value is used by the reference clock prescaler circuit.

There are three sets of PIXPLL registers:

Set A	*Set B*	*Set C*
XPIXPLLAM	**XPIXPLLBM**	**XPIXPLLCM**
XPIXPLLAN	**XPIXPLLBN**	**XPIXPLLCN**
XPIXPLLAP	**XPIXPLLBP**	**XPIXPLLCP**

The **pixpllm** field can be programmed from any of the ‘m’ registers in Set A, B, or C: **XPIXPLLAM**, **XPIXPLLBM**, or **XPIXPLLCM**. The register set which defines the pixel PLL operation is selected by the **clksel** field of the **MISC** VGA register as shown in the following table:

clksel	*Pixel Clock PLL Frequency*	*Reset Value*
‘00’	Register Set A (25.172 MHz)	M = 30
‘01’	Register Set B (28.361 MHz)	M = 25
‘1X’	Register Set C	Unknown

Reserved <7:5> Reserved. When writing to this register, the bits in this field must be set to 0.

Index	45h	**XPIXPLLAN**
	49h	**XPIXPLLBN**
	4Dh	**XPIXPLLCN**
Attributes	R/W, BYTE	
Reset Value	6Ch	**XPIXPLLAN**
	66h	**XPIXPLLBN**
	Unknown	**XPIXPLLCN**

Res.	**pixplln**						
7	6	5	4	3	2	1	0

pixplln <6:0>

Pixel PLL N value register. The ‘n’ value is used by the VCO feedback divider circuit.

The **pixplln** field can be programmed from any of the ‘n’ registers in Set A, B, or C: **XPIXPLLAN**, **XPIXPLLBN**, or **XPIXPLLCN**. The register set which defines the pixel PLL operation is selected by the **clksel** field of the **MISC** VGA register as shown in the following table:

clksel	***Pixel Clock PLL Frequency***	***Reset Value***
‘00’	Register Set A (25.172 MHz)	N = 108
‘01’	Register Set B (28.361 MHz)	N = 102
‘1X’	Register Set C	Unknown

Note: The **pixplln** value must be in the range of 100 (64h) to 127 (7Fh) inclusive.

Reserved <7>

Reserved. When writing to this register, this field must be set to 0.

XPIXPLLP PIXPLL P Value

Index	46h	**XPIXPLLAP**
	4Ah	**XPIXPLLBP**
	4Eh	**XPIXPLLCP**
Attributes	R/W, BYTE	
Reset Value	01h	**XPIXPLLAP**
	01h	**XPIXPLLBP**
	Unknown	**XPIXPLLCP**

Reserved			pixplls		pixpllp		
7	6	5	4	3	2	1	0

pixpllp <2:0>

Pixel PLL P value register. The ‘p’ value is used by the VCO clock divider circuit. The permitted values are:

P = 0 -> Fo = Fvco/1
P = 1 -> Fo = Fvco/2
P = 3 -> Fo = Fvco/4
P = 7 -> Fo = Fvco/8

pixplls <4:3>

Pixel PLL S value register. The ‘s’ value controls the loop filter bandwidth.

50 MHz <= Fvco < 100 MHz S=0
100 MHz <= Fvco < 140 MHz S=1
140 MHz <= Fvco < 180 MHz S=2
180 MHz <= Fvco < 220 MHz S=3

The **pixpllp** and **pixplls** fields can be programmed from any of the ‘p’ registers in Set A, B, or C: **XPIXPLLAP**, **XPIXPLLBP**, or **XPIXPLLCP**. The register set which defines the pixel PLL operation is selected by the **clksel** field of the **MISC** VGA register as shown in the following table:

clksel	***Pixel Clock PLL Frequency***	***Reset Value***
‘00’	Register Set A (25.172 MHz)	P = 1, S = 0
‘01’	Register Set B (28.361 MHz)	P = 1, S = 0
‘1X’	Register Set C	Unknown

Reserved <7:5>

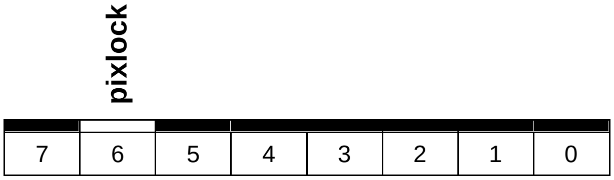
Reserved. When writing to this register, the bits in this field must be set to 0.

PIXPLL Status XPIXPLLSTAT

Index 4Fh

Attributes RO, BYTE

Reset Value Unknown



pixlock <6> Pixel PLL lock status.

- 1: indicates that the pixel PLL has locked to the selected frequency defined by Set A, B, or C
- 0: indicates that lock has not yet been achieved

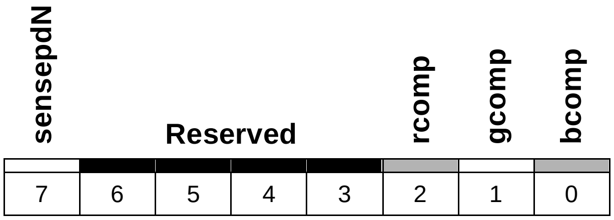
Reserved **<5:0> <7>**

Reserved. When writing to this register, the bits in these fields must be set to 0.

Index 3Ah

Attributes R/W, BYTE

Reset Value `0XXX XXXX`b



bcomp <0>

Sampled blue compare. Verifies that the blue termination is correct.

- 0: blue DAC output is below 350 mV
- 1: blue DAC output exceeds 350 mV

gcomp <1>

Sampled green compare. Verifies that the green termination is correct.

- 0: green DAC output is below 350 mV
- 1: green DAC output exceeds 350 mV

rcomp <2>

Sampled red compare. Verifies that the red termination is correct.

- 0: red DAC output is below 350 mV
- 1: red DAC output exceeds 350 mV

sensepdN <7>

Sense comparator power down

- 0: power down
- 1: power up

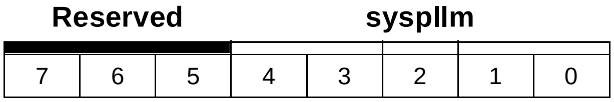
Reserved <6:3>

Reserved. When writing to this register, the bits in this field must be set to 0.

This register reports the sense comparison function, which determines the presence of the CRT monitor and if the termination is correct. The output of the comparator is sampled at the end of every active line. When doing a sense test, the software should program a uniform color for the entire screen.

SYSPLL M Value XSYSPLLM

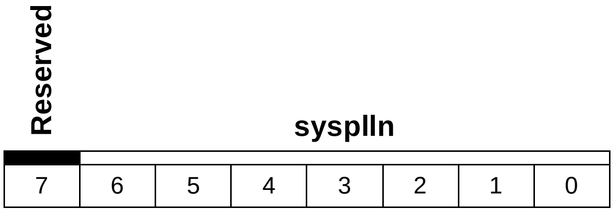
Index	2Ch
Attributes	R/W, BYTE
Reset Value	`0000 1100`b



syspllm <4:0>	System PLL M value register. The ‘m’ value is used by the reference clock prescaler circuit.
Reserved <7:5>	Reserved. When writing to this register, the bits in this field must be set to 0.

XSYSPLLN SYSPLL N Value

Index	2Dh
Attributes	R/W, BYTE
Reset Value	`0111 1000`b



sysplln <6:0> System PLL N value register. The ‘n’ value is used by the VCO feedback divider circuit.

Note: The **sysplln** value must be in the range of 100 (64h) to 127 (7Fh) inclusive.

Reserved <7> Reserved. When writing to this register, this field must be set to 0.

Index 2Eh

Attributes R/W, BYTE

Reset Value `0000 1000`b

Reserved			sysplls		syspllp		
7	6	5	4	3	2	1	0

syspllp <2:0>

System PLL P value register. The ‘p’ value is used by the VCO post-divider circuit.

The permitted values are:

P=0 -> Fo = Fvco/1
P=1 -> Fo = Fvco/2
P=3 -> Fo = Fvco/4
P=7 -> Fo = Fvco/8

Other values are reserved.

sysplls <4:3>

System PLL S value register. The ‘s’ value controls the loop filter bandwidth.

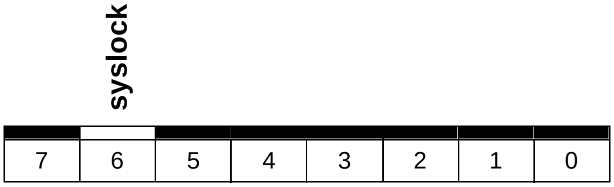
50 MHz <= Fvco < 100 MHz -> S=0
100 MHz <= Fvco < 140 MHz -> S=1
140 MHz <= Fvco < 180 MHz -> S=2
180 MHz <= Fvco < 220 MHz -> S=3

Reserved <7:5>

Reserved. When writing to this register, the bits in this field must be set to 0.

XSYSPLLSTAT SYSPLL Status

Index	2Fh
Attributes	RO, BYTE
Reset Value	Unknown



syslock <6> System PLL lock status.

- 1: indicates that the system PLL has locked to the selected frequency
- 0: indicates that lock has not yet been achieved

Reserved **<5:0> <7>**

Reserved. When writing to this register, the bits in these fields must be set to 0.

Index	18h
Attributes	R/W, BYTE
Reset Value	`0000 0000`b

Reserved		dacbgen	dacbgpdN	pixpllbgen	pixpllbgpdN	syspllbgen	syspllbgpdN
7	6	5	4	3	2	1	0

syspllbgpdN <0>

System PLL voltage reference block power down.

- 0: power down
- 1: power up

syspllbgen <1>

System PLL voltage reference enable.

- 0: use external voltage reference
- 1: use PLL voltage reference block

pixpllbgpdN <2>

Pixel PLL voltage reference block power down.

- 0: power down
- 1: power up

pixpllbgen <3>

Pixel PLL voltage reference enable.

- 0: use external voltage reference
- 1: use PLL voltage reference block

dacbgpdN <4>

DAC voltage reference block power down.

- 0: power down
- 1: power up

dacbgen <5>

DAC voltage reference enable.

- 0: use external voltage reference
- 1: use DAC voltage reference block

Reserved <7:6>

Reserved. When writing to this register, the bits in this field must be set to 0.

To select an off-chip voltage reference, ***all*** enables must be set to ‘0’. To select the internal voltage references, all enables must be set to ‘1’, and all voltage reference blocks must be powered up (write ‘0011 1111’).

XZOOMCTRL Zoom Control

Index 38h

Attributes R/W, BYTE

Reset Value `0000 0000`b

Reserved						hzoom	
7	6	5	4	3	2	1	0

hzoom <1:0> Horizontal zoom factor. Specifies the (zoom) factor used to replicate pixels in the horizontal display line. The following factors are supported:

- 00: 1x (default)
- 01: 2x
- 10: reserved
- 11: 4x

Note: The cursor is not affected by the **hzoom** bits (the cursor is never zoomed).

Reserved <7:2> Reserved. When writing to this register, the bits in this field must be set to 0.

Chapter 5: Programmer’s Specification

This chapter includes:

5.1 PCI Interface

5.1.1 Introduction

The MGA-1064SG chip interacts directly with the PCI interface. We have exploited certain features and characteristics of the PCI interface in order to improve the performance of the graphics subsystem. To this end, the following buffering has been provided:

BFIFO	This is a 32-entry FIFO which is used to interface with the drawing engine registers. All the registers that are accessed through the BFIFO are identified in the register descriptions in Chapter 4 with the ‘FIFO’ attribute. The BFIFO is also used for the data by ILOAD operations.
MIFIFO	This is an 8-entry FIFO which is used for direct frame buffer VGA/MGA accesses, for accesses to the DAC, and for accesses to external devices.
MOFIFO	This is a 4-entry FIFO which is used for IDUMP operations.
CACHE	This is a 4-location cache, which is used for direct frame buffer VGA/MGA read accesses, for accesses to the DAC, or for accesses to external devices.

The following table shows when the BFIFO, MIFIFO, MOFIFO, or CACHE are used for different classes of access.

Access	*Type*	*BFIFO*	*MIFIFO*	*CACHE*	*MOFIFO*
Configuration registers	R W				
ROM	R W		W W	R	
DMAWIN or **MGABASE3**	R W	W			R
Drawing registers	R W	W			
Host registers	R W				
Host registers +DRWI(1)	R W	W			
VGA registers (I/O, MEM)	R W				
DAC (I/O, MEM, Snooping)	R W		W W	R	
Expansion devices	R W		W W	R	
VGA frame buffer	R W		W W	R	
MGABASE2	R W		W W	R	

(1) DRWI: Drawing Register Window Indirect access

5.1.2 PCI Retry Handling

In some situations the chip may not be able to respond to a PCI access immediately, so a certain number of retry cycles will be generated. A retry will be asserted when:

- The BFIFO is written to when it is full.
- The MIFIFO is written to when it is full.
- The MOFIFO is read when it is empty.
- The CACHE is read when the MIFIFO is not empty or when the data in the cache is not ready.
- The VGA registers are written to when the MIFIFO is not empty.

In some situations, retries can be used to increase efficiency and for software simplification. For example, there is no need to poll the bfull flag of the BFIFO before writing to it. If the BFIFO is full, a retry cycle will be generated until a location becomes free. At that point the access can be completed, and the program will proceed to the next instruction.

✐ Note: Some systems generate an error after very few retries. In this case, you must check the BFIFO flag (thereby limiting the number of retries) to prevent a system error.

5.1.3 PCI Burst Support

The chip uses PCI burst mode in all situations where performance is critical. The following table summarizes when bursting is and is not used:

Access	*Access Type*	*Burst*
MGABASE1 + DMAWIN range	R/W	Yes
MGABASE1 + drawing register range	W	Yes
MGABASE1 + host reg. range +DRWI range	W	Yes
MGABASE3 range	R/W	Yes
VGA frame buffer range	W	Yes
VGA frame buffer range (mgamode = 0)	R	No (1)
VGA frame buffer range (mgamode = 1)	R (cache hit)	Yes
VGA frame buffer range (mgamode = 1)	R (cache miss)	No (1)
MGABASE2 range	W	Yes
MGABASE2 range	R (cache hit)	Yes
MGABASE2 range	R (cache miss)	No (1)
Configuration register range	R/W	No
I/O range	R/W	No
ROMBASE range	R/W	No (1)
MGABASE1 + host register range	R/W	No
MGABASE1 + VGA register range	R/W	No
MGABASE1 + DAC range	R/W	No (1)
MGABASE1 + expansion device range	R/W	No (1)

(1) The *PCI Specification* (Rev. 2.1) states that a target is required to complete the initial data phase within 16 PCLKs. In order to meet this specification, a read of a location within one of these ranges will activate the delayed transaction mechanism (when the **noretry** field of **OPTION** = ‘0’).

Accesses that are not supported in burst mode always generate a target disconnect when they are accessed in burst mode. Refer to Section 3.1.3 on page 3-4 for the exact addresses.

Burst mode is supported for reads of the MOFIFO, which is read in the DMAWIN or 8 MByte Pseudo-DMA window range. Disconnection will occur when the MOFIFO becomes empty (such a situation can happen when the drawing engine is busy with a memory or screen refresh cycle).

5.1.4 PCI Target-Abort Generation

The MGA-1064SG generates a target-abort in two cases, as stated in the PCI Specification. The target-abort is generated only for I/O accesses, since they are the only types of access that apply to each case.

Case A: PCBE<3:0>/ and PAD<1:0> are Inconsistent

The only exception, mentioned in the PCI Specification, is when PCBE<3:0>/ = ‘1111’. The following table shows the combinations of PAD<1:0> and PCBE<3:0>/ which result in the generation of a target-abort by the MGA-1064SG.

PAD<1:0>	*PCBE<3:0>/*
‘00‘	‘0XX1’
	‘X0X1’
	‘XX01’
‘01‘	‘XXX0’
	‘X011’
	‘0111’
‘10‘	‘XXX0’
	‘XX01’
	‘0111’
‘11‘	‘XXX0’
	‘XX01’
	‘X011’

CASE B: PCBE<3:0>/ Addresses More Than One Device

For example, if a write access is performed at 3C5h with PCBE<3:0>/ = ‘0101’, both the VGA **SEQ** (Data) register and the DAC **PALRDADD** register are addressed. All of these accesses are terminated with a target-abort, after which the **sigtargab** bit of the **DEVCTRL** register is set to ‘1’.

5.1.5 Transaction Ordering

The order of the transactions is extremely important for the VGA and the DAC for either I/O or memory mapped accesses. This means that a read to a VGA register must be completed before a write to the same VGA register can be initiated (especially when there is an address/data pointer that toggles when the register is accessed). In fact, this limits to one the number of PCI devices that are allowed to access the MGA-1064SG’s VGA or DAC.

5.1.6 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 four-dword cache entry. After a hard or soft reset, no cache hit is possible and the first direct read from the frame buffer fills the cache. When the data is available in the cache, the data phase of the access will be completed in 2 pclks.

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 to the EPROM, DAC, 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. The **CACHEFLUSH** register can be used, since it occupies a reserved address in the memory mapped VGA register space (**MGABASE1** + 1FFFh).

5.1.7 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-1064SG to correct this situation.

The exact action that will be taken depends on the data size (the MGA-1064SG 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. Keep in mind that the MGA-1064SG 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-1064SG 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 processors 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-1064SG 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-1064SG 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:

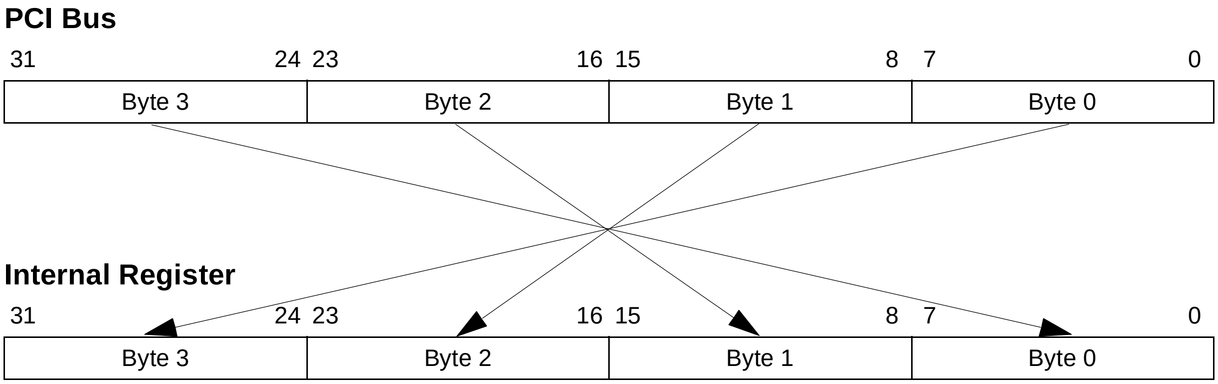
- The VGA compatibility section. Data in this section is 8 bits wide.
- The DAC. Data in this section is 8 bits wide.
- External devices. In this case, the width of the data cannot be known in advance.

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.

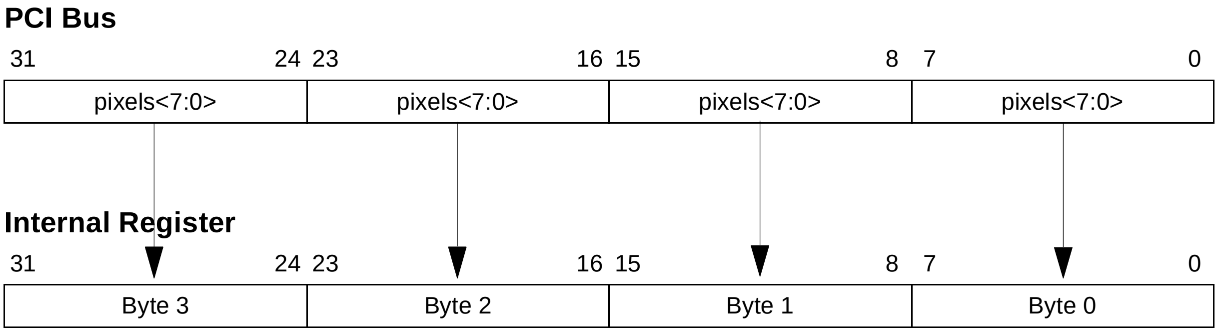
Setting the **powerpc** bit ensures that a 32-bit access by a big endian processor will load the correct data into a 32-bit register. In other words, when data is treated as 32 bit-quantities, it will appear in the identical way to both little and big endian processors. Note however that byte and word 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



powerpc = 0

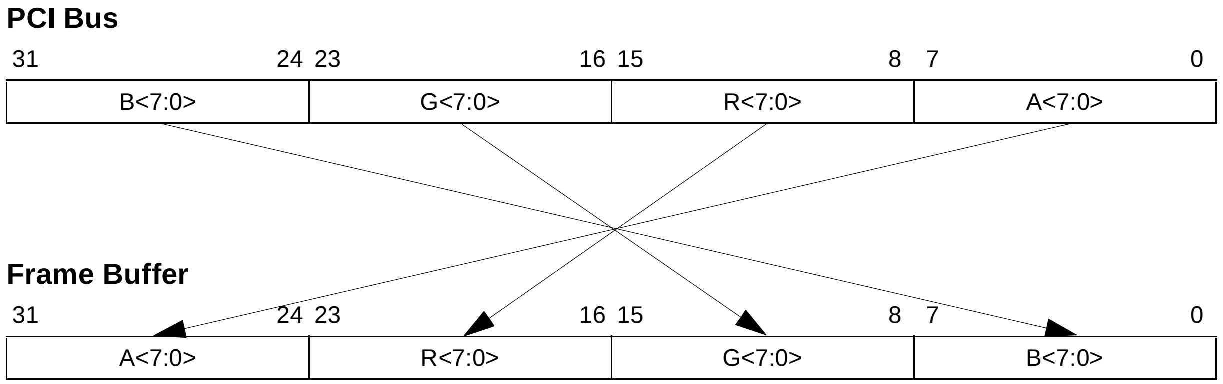


Frame Buffer Space

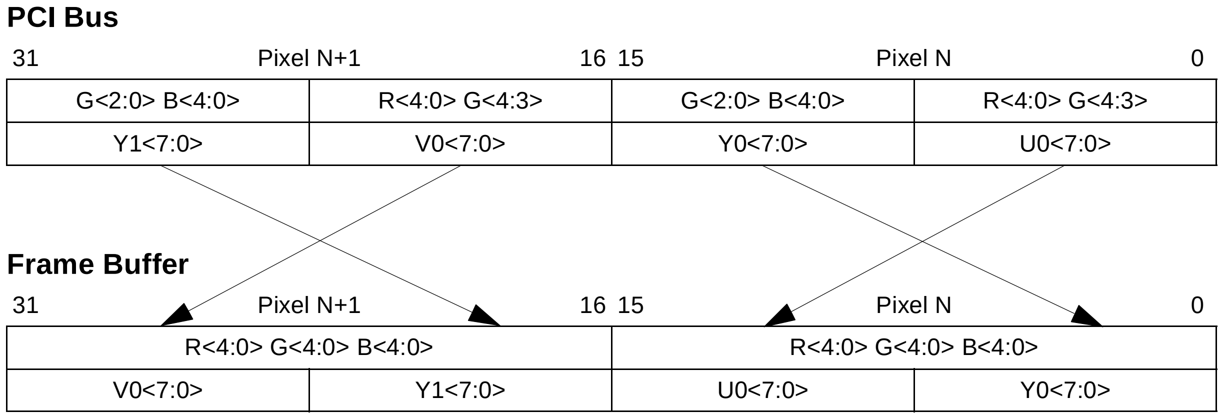
The frame buffer is organized in little endian format, and byte swapping depends on the size of the pixel. As usual, addresses are not modified.

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.

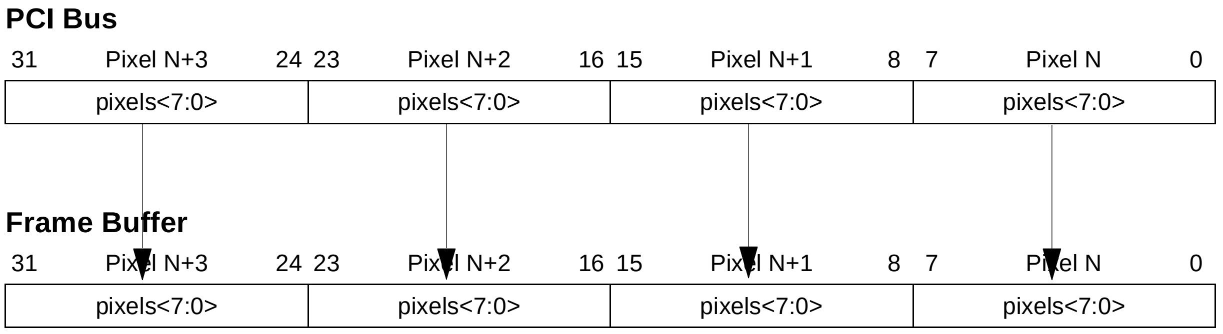
32 bits/pixel, dirDataSiz = 10



16 bits/pixel, dirDataSiz = 01



8 bits/pixel, dirDataSiz = 00



ILOAD & IDUMP Space (DMAWIN or 8 MByte Pseudo-DMA Window))

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’.

5.1.8 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 Little endian 8-bit (see the **powerpc** field of **OPTION**): used in ILOAD and IDUMP operations. Refer to Table 5-3 on page 5-49 and Table 5-10 on page 5-61.

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 Big endian 8-bit (see the **powerpc** field of **OPTION**): used in ILOAD and IDUMP operations. Refer to Table 5-3 on page 5-49 and Table 5-10 on page 5-61.

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 Little endian 16-bit (see the **powerpc** field of **OPTION**): used in ILOAD and IDUMP operations. Refer to Table 5-3 on page 5-49 and Table 5-10 on page 5-61.

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 Big endian 16-bit (see the **powerpc** field of **OPTION**): used in ILOAD and IDUMP operations. Refer to Table 5-3 on page 5-49 and Table 5-10 on page 5-61.

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 32-bit RGB, used in ILOAD and IDUMP operations. Refer to Table 5-3 on page 5-49, Table 5-5 on page 5-51, and Table 5-10 on page 5-61.

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 32-bit BGR used in ILOAD and IDUMP operations. Refer to Table 5-3 on page 5-49, Table 5-5 on page 5-51, and Table 5-10 on page 5-61.

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	:	:	:	:

32-bit C 32-bit RGB used in ILOAD_HIGHV operations. Refer to Table 5-6 on page 5-51.

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	Green: Line 0, Pixel 0	Red: Line 1, Pixel 0	Green: Line 1, Pixel 0	Blue: Line 1, Pixel 0
1	Green: Line 0, Pixel 1	Red: Line 1, Pixel 1	Green: Line 1, Pixel 1	Blue: Line 1, Pixel 1
2	Green: Line 0, Pixel 2	Red: Line 1, Pixel 2	Green: Line 1, Pixel 2	Blue: Line 1, Pixel 2
3	:	:	:	:

32-bit D 32-bit BGR used in ILOAD_HIGHV operations. Refer to Table 5-6 on page 5-51.

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	Green: Line 0, Pixel 0	Blue: Line 1, Pixel 0	Green: Line 1, Pixel 0	Red: Line 1, Pixel 0
1	Green: Line 0, Pixel 1	Blue: Line 1, Pixel 1	Green: Line 1, Pixel 1	Red: Line 1, Pixel 1
2	Green: Line 0, Pixel 2	Blue: Line 1, Pixel 2	Green: Line 1, Pixel 2	Red: Line 1, Pixel 2
3	:	:	:	:

24-bit A 24-bit RGB packed pixel, used in ILOAD and IDUMP operations. Refer to Table 5-3 on page 5-49, Table 5-5 on page 5-51, and Table 5-10 on page 5-61.

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 24-bit BGR packed pixel, used in ILOAD and IDUMP operations. Refer to Table 5-3 on page 5-49, Table 5-5 on page 5-51, and Table 5-10 on page 5-61.

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 Little endian, single-buffer YUV, used in ILOAD operations. Refer to Table 5-3 on page 5-49 and Table 5-5 on page 5-51.

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 Little endian, single-buffer YUV with byte swap, used in ILOAD operations. Refer to Table 5-3 on page 5-49 and Table 5-5 on page 5-51.

	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 Big endian, single-buffer YUV, used in ILOAD operations. Refer to Table 5-3 on page 5-49 and Table 5-5 on page 5-51.

	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 Big endian, single-buffer YUV with byte swap, used in ILOAD operations. Refer to Table 5-3 on page 5-49 and Table 5-5 on page 5-51.

	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	:	:	:	:

YUV E(1) Little endian, double-buffer YUV, used in ILOAD_HIGHV operations. Refer to Table 5-6 on page 5-51.

	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	V10	Y11	U10	Y10
1	V00	Y01	U00	Y00
2	V12	Y13	U12	Y12
3	V02	Y03	U02	Y02
4	V14	Y15	U14	Y14
5	V04	Y05	U04	Y04
6	:	:	:	:

---

(1) Yij | Uij | Vij, where i = line, j = pixel. For example: Y10 = Y for pixel 0 on line 1.

YUV F(1) Little endian, double-buffer YUV with byte swap, used in ILOAD_HIGHV operations. Refer to Table 5-6 on page 5-51.

	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	Y11	V10	Y10	U10
1	Y01	V00	Y00	U00
2	Y13	V12	Y12	U12
3	Y03	V02	Y02	U02
4	Y15	V14	Y14	U14
5	Y05	V04	Y04	U04
6	:	:	:	:

YUV G(1) Big endian, double-buffer YUV used in ILOAD_HIGHV operations. Refer to Table 5-6 on page 5-51.

	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	Y10	U10	Y11	V10
1	Y00	U00	Y01	V00
2	Y12	U12	Y13	V12
3	Y02	U02	Y03	V02
4	Y14	U14	Y15	V14
5	Y04	U04	Y05	V04
6	:	:	:	:

YUV H(1) Big endian, double-buffer YUV with byte swap, used in ILOAD_HIGHV operations. Refer to Table 5-6 on page 5-51.

	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	U10	Y10	V10	Y11
1	U00	Y00	V00	Y01
2	U12	Y12	V12	Y13
3	U02	Y02	V02	Y03
4	U14	Y14	V14	Y15
5	U04	Y04	V04	Y05
6	:	:	:	:

---

(1) Yij | Uij | Vij, where i = line, j = pixel. For example: Y10 = Y for pixel 0 on line 1.

MONO A Little endian 1-bit used in ILOAD and BITBLT operations. Refer to Table 5-4 on page 5-50.

	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 Little endian 1-bit Windows format, used in ILOAD and BITBLT operations. Refer to Table 5-4 on page 5-50.

	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 Big endian 1-bit Windows format, used in ILOAD and BITBLT operations. Refer to Table 5-4 on page 5-50.

	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 Memory Interface

5.2.1 Frame Buffer Organization

The MGA-1064SG supports up to four 2 MByte banks of memory (using SGRAM) or one 8 MByte bank of memory (using SDRAM). Using this configuration, it is possible to design a 2, 4, or 8 MByte product.

There are two different frame buffer organizations, described below:

- VGA mode
- Power Graphic mode

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.

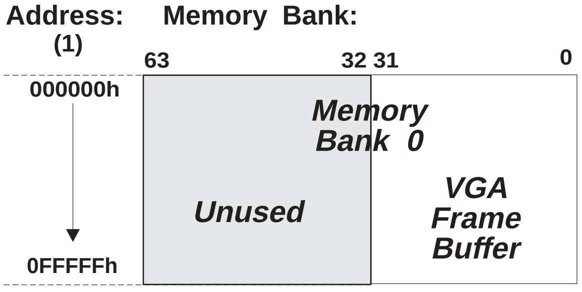
Supported Resolutions

Resolution	*8-bit*	*16-bit*	*24-bit*	*32-bit*
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	-

The memory type must be properly initialized at power up. In normal operation, the **hardpwmsk** and **memconfig** fields should not change, as they're used to determine the ***type*** of memory. However, **splitmode** and **fbmskN** can be changed to reflect a new memory ***organization***.

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

The possible memory configurations are described in the subsections which follow. Note that:

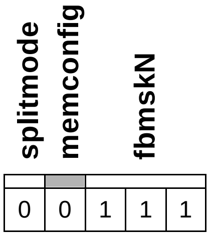
- All addresses are hexadecimal
- In split mode, the graphics and video pixels are processed in parallel by the internal DAC.

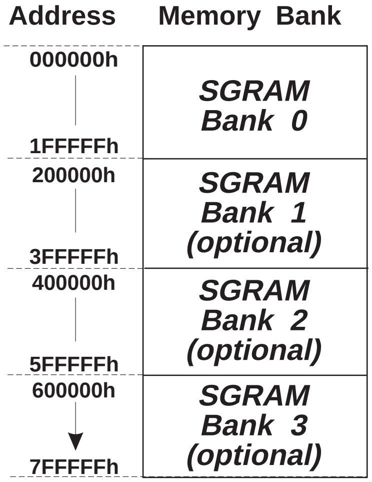
- The **depth** field of the **XMULCTRL** DAC register chooses between 2G8V16 and G16V16 when the **mgamode** field of **CRTCEXT3** is ‘1’. When SG8V16 is selected for split mode, the **pwidth** field of **MACCESS** is usually set to PW8 for any graphics access, and to PW16 for any video access. When G16V16 is selected, **pwidth** should be set as PW16 for both graphics and video accesses.

✎ In each memory configuration subsection which follows, the fields within **OPTION<13:9>** are set according to the illustration on the left side to produce the configuration depicted on the right side.

■ SGRAM

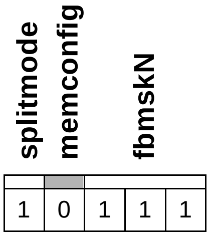
OPTION<13:9>

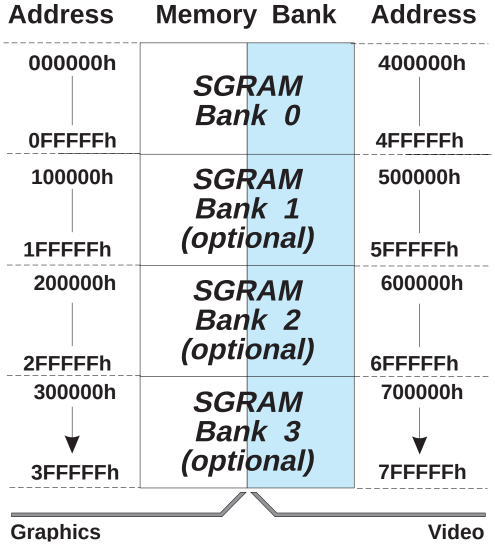




■ SGRAM, split mode

OPTION<13:9>





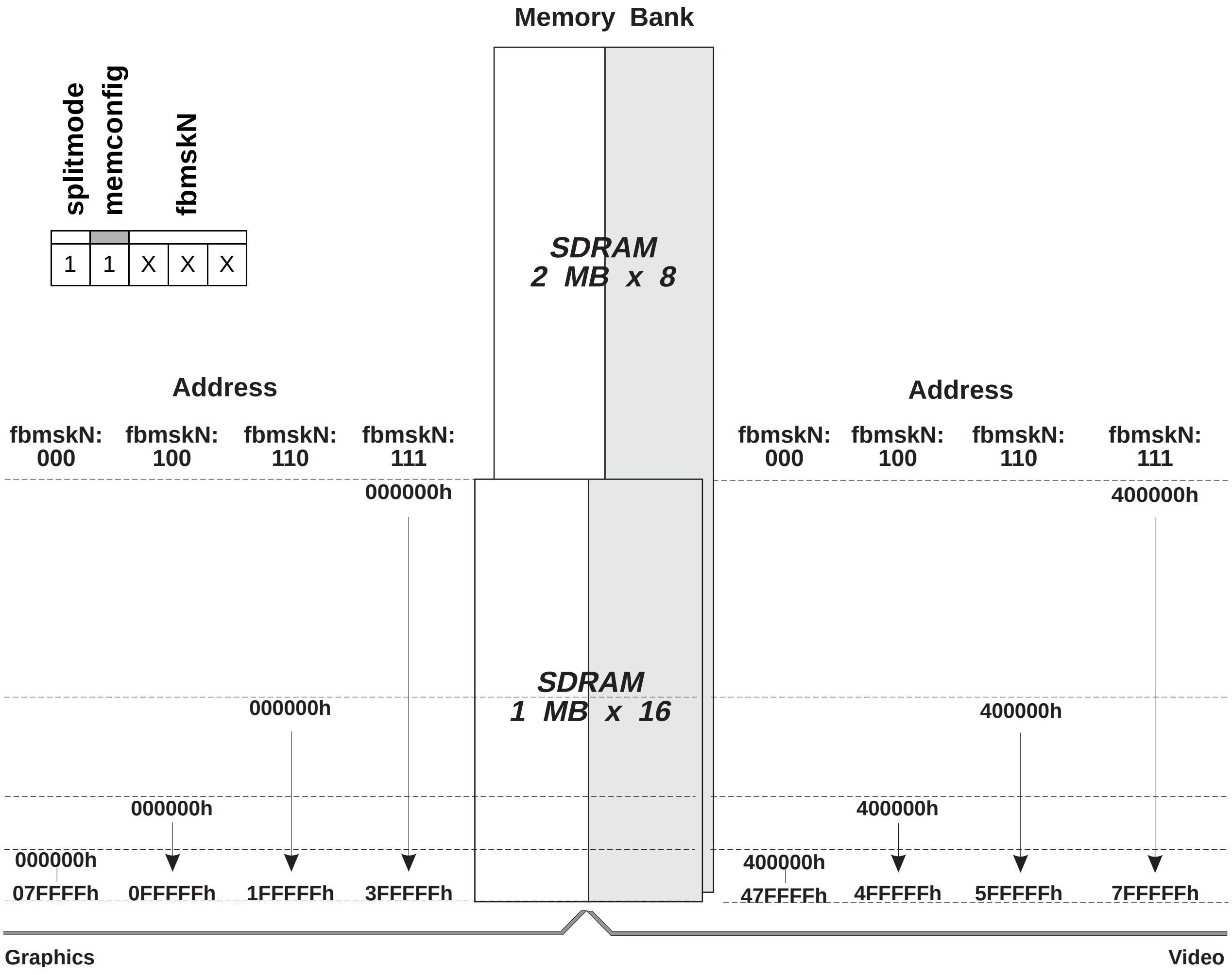
SDRAM

OPTION<13:9>

splitmode	memconfig	fbmskN		
0	1	X	X	X

SDRAM, split mode

OPTION<13:9>



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 DAC. 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.

32 bits/pixel

63 32	31 0
P1	P0

24 bits/pixel

63 56	55 48	47 40	39 32	31 24	23 16	15 8	7 0
P2		P1			P0		
P5	P4			P3			P2
P7			P6			P5	

16 bits/pixel

63 48	47 32	31 16	15 0
P3	P2	P1	P0

8 bits/pixel

63 56	55 48	47 40	39 32	31 24	23 16	15 8	7 0
P7	P6	P5	P4	P3	P2	P1	P0

Monochrome

63	0
P63	P0

For each of these modes, the pixels are arranged as follows:

32 bits/pixel

31 24	23 16	15 8	7 0
7 Alpha 0	7 Red 0	7 Green 0	7 Blue 0

24 bits/pixel

23 16	15 8	7 0
7 Red 0	7 Green 0	7 Blue 0

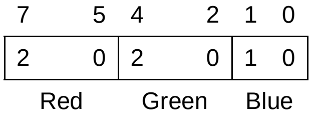
16 bits/pixel (5:5:5)

15	14 10	9 5	4 0
0	4 Red 0	4 Green 0	4 Blue 0
Alpha			

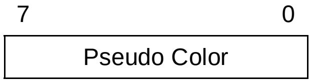
16 bits/pixel (5:6:5)

15 11	10 5	4 0
4 Red 0	5 Green 0	4 Blue 0

8 bits/pixel



8 bits/pixel



5.3 Chip Configuration and Initialization

5.3.1 Reset

The MGA-1064SG can be both hard and soft reset. Hard reset is achieved by activating the PRST/ pin. There is no need for the PRST/ pin to be synchronous with any clock.

- 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.

The state of the straps are read and registered internally upon hard reset. A soft reset will not re-read the external straps, nor will it change the state of the bits of the **OPTION** register.

Strap Name	*Pins*	*Description*
biosen	VD<14>	Indicates whether a ROM is installed ('1') or not ('0'). The biosen strap also controls the **biosen** field of the **OPTION** register.
pid<4:0>	VD<12:8>	User-defined. Undefined bits should be strapped ***high*** by default. These bits are loaded into the **productid** field of the **OPTION** register.
unimem	VD<15>	Reserved. Must be pulled down.
vgaboot	VD<13>	Indicates whether the VGA I/O locations are decoded ('1') or not ('0') only if the **vgaioen** bit has not been written. The vgaboot strap also controls bit 23 of the **CLASS** register, setting the **class** 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 4-70 for the details).

The soft reset should be interpreted as a drawing engine reset more than as a general soft reset. The video circuitry, VGA registers, and frame buffer memory accesses, 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 Operations After Hard Reset

- After a hard reset, the DAC will be in a VGA-compatible state. All of the internal clocks (GCLK, VCLK, and PIXCLK) will be based on the PCI bus clock and enabled. The MCLK will be based on the PCI bus clock.
- The two internal PLLs will be bypassed and powered down, and the analog DAC will also be powered down. Refer to the **pixpllpdN** field of the **XPIXCLKCTRL** register, the **syspllpdN** field of the **OPTION** register, and the **dacpdN** field of the **XMISCCTRL** register.
- The three internal voltage reference blocks will be powered down to avoid contention on the REF pins.
- The **clksel** field of the VGA **MISC** register will select register set A for the pixel PLL so that the frequency of the pixel PLL will be at 25.175 MHZ when the PLL is powered up.
- The system PLL registers will program the system PLL to oscillate at 133.333 MHz when it is powered up.
- The internal data path of the DAC will be configured in VGA mode, so the pixel data will come from the MGA-1064SG's Attribute Controller.
- The palette defaults to 6-bit operation.
- Register bits that do not have a reset value will wake up with unknown values. In particular, the palette RAM (LUT) will be undefined, and must be programmed before being used.
- The feature connector will be in 8-bit VGA output mode.
- Frame buffer memory refreshing is not running.

5.3.3 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*
eepromwt	'0'	To be set to '1' if a FLASH ROM is used, and writes are to be done to the ROM.
powerpc	'0'	To be set to '1' to support big endian processor accesses.
rfhcnt	'0000'	The refresh counter defines the rate of MGA memory refresh. For a typical 75 MHz MCLK, a value of 9 would be programmed.
vgaioen	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.

5.3.3.1 MGA-1064SG and RAM Reset Sequence

After a reset, the clocks must be initialized first. Observe the following sequences to ensure proper behavior of the chip and to properly initialize the RAM.

Analog Macro Power Up Sequence

Step 1. If an ‘off-chip’ voltage reference is not used, then:

(i) Program **XVREFCTRL** (refer to the register description and its associated note).

(ii) Wait 100 mS for the reference to become stable.

Step 2. Power up the system PLL by setting the **syspllpdN** field of **OPTION** to ‘1’.

Step 3. Wait for the system PLL to lock (indicated when the **syslock** field of the **XSYSPLLSTAT** register is ‘1’).

Step 4. Power up the pixel PLL by setting the **pixpllpdN** field of **XPIXCLKCTRL** to ‘1’.

Step 5. Wait for the pixel PLL to lock (indicated when the **pixlock** field of **XPIXPLLSTAT** is ‘1’).

Step 6. Power up the LUT by setting the **ramcs** field of **XMISCCTRL** to ‘1’.

Step 7. Power up the DAC by setting the **dacpdN** field of **XMISCCTRL** to ‘1’.

The PLLs are now set up and oscillating at their reset frequencies, but they are not selected. The following steps will set MCLK to 66 MHz, GCLK to 44 MHz, and PIXCLK to 25.175 MHz. Refer to Section 5.7.8.3 on page 5-77.

1. Disable the system clocks by setting **sysclkdis** (**OPTION** register) to ‘1’.
2. Select the system PLL by setting **sysclksl** to ‘01’.
3. Enable the system clocks by setting **sysclkdis** to ‘0’.
4. Disable the pixel clock and video clock by setting **pixclkdis** (**XPIXCLKCTRL** register) to ‘1’.
5. Select the pixel PLL by setting **pixclksl** to ‘01’.
6. Enable the pixel clock and video clock by setting **pixclkdis** to ‘0’.

✎ Each of the preceding six steps ***must*** be done as a single PCI access. They cannot be combined.

SGRAM/SDRAM Initialization

Step 1. Set the **scroff** blanking field (**SEQ1**<5>) to prevent any transfer.

Step 2. Program the **casltncy** field of the **MCTLWTST** register.

Step 3. Program the **memconfig** field of the **OPTION** register.

Step 4. Wait a minimum of 200 µs.

Step 5. Set the **memreset** field and clear the **jedecrst** field of **MACCESS**.

Step 6. Wait a minimum of (100 × the current MCLK period).

Step 7. Set the **memreset** and **jedecrst** fields of **MACCESS**.

Step 8. Start the refresh by programming the **rfhcnt** field of the **OPTION** register.

5.3.4 Operation Mode Selection

The MGA-1064SG provides three different display modes: text (VGA or SVGA), VGA graphics, and SVGA graphics. Table 5-1 lists all of the display modes which are available through BIOS calls.

- 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. This results in very high performance, with high resolution and a greater number of pixel depths.

Table 5-1: Display Modes (Part 1 of 3)

Mode	*Type*	*Organization*	*Resolution*	*No. of colors*
0	VGA	40x25 Text	360x400	16
1	VGA	40x25 Text	360x400	16
2	VGA	80x25 Text	720x400	16
3	VGA	80x25 Text	720x400	16
4	VGA	Packed-pixel 2 bpp	320x200	4
5	VGA	Packed-pixel 2 bpp	320x200	4
6	VGA	Packed-pixel 1 bpp	640x200	2
7	VGA	80x25 Text	720x400	2
D	VGA	Multi-plane 4 bpp	320x200	16
E	VGA	Multi-plane 4 bpp	640x200	16
F	VGA	Multi-plane 1 bpp	640x350	2
10	VGA	Multi-plane 4 bpp	640x350	16
11	VGA	Multi-plane 1 bpp	640x480	2
12	VGA	Multi-plane 4 bpp	640x480	16
13	VGA	Packed-pixel 8 bpp	320x200	256
?	VGA	90x25 Text	810x400	2
?	VGA	90x28 Text	810x400	2
?	VGA	90x50 Text	810x400	2
?	VGA	120x25 Text	960x400	2
?	VGA	120x28 Text	960x400	2
?	VGA	120x50 Text	960x400	2
?	VGA	132x43 Text	1056x350	2
?	VGA	132x25 Text	1056x400	2
?	VGA	132x28 Text	1056x400	2
?	VGA	132x50 Text	1056x400	2
?	VGA	132x60 Text	1056x480	2
?	VGA	90x25 Text	810x400	16
?	VGA	90x28 Text	810x400	16
?	VGA	90x50 Text	810x400	16
?	VGA	120x25 Text	960x400	16
?	VGA	120x28 Text	960x400	16

Table 5-1: Display Modes (Part 2 of 3)

Mode	*Type*	*Organization*	*Resolution*	*No. of colors*
?	VGA	120x50 Text	960x400	16
?	VGA	132x43 Text	1056x350	16
?	VGA	132x25 Text	1056x400	16
?	VGA	132x28 Text	1056x400	16
?	VGA	132x50 Text	1056x400	16
?	VGA	132x60 Text	1056x480	16
?	SVGA	Packed-pixel 8 bpp	640x480	256
?	SVGA	Packed-pixel 16 bpp	640x480	32K
?	SVGA	Packed-pixel 16 bpp	640x480	64K
?	SVGA	Packed-pixel 24 bpp	640x480	16M
?	SVGA	Packed-pixel 32 bpp	640x480	16M
?	SVGA	Packed-pixel 8 bpp	768x576	256
?	SVGA	Packed-pixel 16 bpp	768x576	32K
?	SVGA	Packed-pixel 16 bpp	768x576	64K
?	SVGA	Packed-pixel 24 bpp	768x576	16M
?	SVGA	Packed-pixel 32 bpp	768x576	16M
?	SVGA	Packed-pixel 8 bpp	800x600	256
?	SVGA	Packed-pixel 16 bpp	800x600	32K
?	SVGA	Packed-pixel 16 bpp	800x600	64K
?	SVGA	Packed-pixel 24 bpp	800x600	16M
?	SVGA	Packed-pixel 32 bpp	800x600	16M
?	SVGA	Packed-pixel 24 bpp	960x720	16M
?	SVGA	Packed-pixel 8 bpp	1024x768	256
?	SVGA	Packed-pixel 16 bpp	1024x768	32K
?	SVGA	Packed-pixel 16 bpp	1024x768	64K
? (1)	SVGA	Packed-pixel 24 bpp	1024x768	16M
? (1)	SVGA	Packed-pixel 32 bpp	1024x768	16M
?	SVGA	Packed-pixel 8 bpp	1152x882	256
?	SVGA	Packed-pixel 16 bpp	1152x882	32K
?	SVGA	Packed-pixel 16 bpp	1152x882	64K
? (1)	SVGA	Packed-pixel 24 bpp	1152x882	16M
? (1)	SVGA	Packed-pixel 32 bpp	1152x882	16M
?	SVGA	Packed-pixel 8 bpp	1280x1024	256
? (1)	SVGA	Packed-pixel 16 bpp	1280x1024	32K
? (1)	SVGA	Packed-pixel 16 bpp	1280x1024	64K
? (1)	SVGA	Packed-pixel 24 bpp	1280x1024	16M
? (2)	SVGA	Packed-pixel 32 bpp	1280x1024	16M
?	SVGA	Packed-pixel 8 bpp	1600x1200	256
? (1)	SVGA	Packed-pixel 16 bpp	1600x1200	32K
? (1)	SVGA	Packed-pixel 16 bpp	1600x1200	64K

Table 5-1: Display Modes (Part 3 of 3)

Mode	Type	Organization	Resolution	No. of colors
? (2)	SVGA	Packed-pixel 24 bpp	1600x1200	16M
?	SVGA	Packed-pixel 8 bpp	1920x1024	256
? (1)	SVGA	Packed-pixel 16 bpp	1920x1024	32K
? (1)	SVGA	Packed-pixel 16 bpp	1920x1024	64K
? (2)	SVGA	Packed-pixel 24 bpp	1920x1024	16M

(1) Possible only with a frame buffer of at least 4 MBytes.

(2) Possible only with a frame buffer of 8 MBytes.

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 DAC 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. Note that the MGA-1064SG has the capacity to 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-1064SG's registers to perform various graphics functions. The following two methods can be used:

- Direct access to the register. In this case all registers are accessed directly by the host, using the address as specified in the register descriptions found in Chapter 4.
- Pseudo-DMA. In this case, the addresses of the individual registers to be accessed are embedded in the data stream. Pseudo-DMA can be used in four different ways:
  - The General Purpose Pseudo-DMA mode can used with any command.
  - The DMA Vector Write mode is specifically dedicated to polyline operations.
  - ILOAD and IDUMP operations always use Pseudo-DMA transfers for exchanging data with the frame buffer.

Note: Only ***dword*** accesses can be used when initializing the drawing engine. This is true for both direct register access and for Pseudo-DMA operation.

5.5.1 Drawing Register Initialization Using General Purpose Pseudo-DMA

The general purpose Pseudo-DMA operations are performed through the DMAWIN aperture in the MGA control register space, or in the 8 MByte Pseudo-DMA 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).

General Purpose Pseudo-DMA mode is entered when either the DMAWIN space or the 8 MByte Pseudo-DMA window is written to or read from. The DMA sequence can be interrupted by writing to byte 0 of the **OPMODE** register; this mechanism can be used when the last packet is incomplete.

The first double word written to the DMA window is loaded into the Address Generator. This double word contains indices to the next four drawing registers to be written, and the next four double word transfers contain the data that is to be written to the four registers specified.

When each double word of data is transferred, the Address Generator sends the appropriate 8-bit index to the Bus FIFO. This 8-bit address corresponds to bits 13 and 8:2. Bit 13 represents the DWGREG1 range (refer to Table 3-3 on page 3-4). Bits 1:0 are omitted, since each register is a double word. All registers marked with the FIFO attribute in the register descriptions in Chapter 4 can be initialized in General Purpose Pseudo-DMA mode. When the fourth (final) index has been used, the next double word transfer reloads the Address Generator.

DMA General Purpose Transfer Buffer Structure

	31 24	23 16	15 8	7 0
0	indx3	indx2	indx1	indx0
1	data 0			
2	data 1			
3	data 2			
4	data 3			
5	indx3	indx2	indx1	indx0
6	data 0			
7	data 1			
8	data 2			
	. . .			

5.5.2 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 5.5.4.2.
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 4-49).

   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.

5.5.3 Global Initialization (All Operations)

You must initialize the following registers for all graphics operations:

Register	*Function*	*Comment / Alternate Function*
PITCH	Set pitch	Specify destination address linearization (**iy** field)
YDSTORG	Determine screen origin	
MACCESS	Set pixel format (8, 16, 24, 32 bpp)	Some limitations apply
CXBNDRY	Left/right clipping limits	Can use **CXLEFT** and **CXRIGHT** instead
YTOP	Top clipping limit	
YBOT	Bottom clipping limit	
PLNWT	Plane write mask	
ZORG	Z origin position	Only required for depth operations

5.5.4 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 5.5.3 and subsection 5.5.4.1 first. Also, ***the last register you program must be accessed in the 1D00h-1DFFh range in order to start the drawing engine***.

5.5.4.1 Slope Initialization

Non Auto-init Lines

This type of line is initiated when the **DWGCTL** register's opcod 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*
AR0	2b (1)	
AR1	Error term: 2b - a - sdy	
AR2	Minor axis increment: 2b - 2a	
SGN	Vector quadrant (2)	
XDST	The x start position	
YDSTLEN	The y start position and vector length	Can use **YDST** and **LEN** instead; **must** use **YDST** and **LEN** when destination address is linear (i.e.. **ylin** = 1, see **PITCH**)

(1) Definitions: a = max (|dY|, |dX|), b = min (|dY|, |dX|).

(2) 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 **opcod** 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*
XYSTRT	The x and y starting position	Can use **AR5**, **AR6**, **XDST**, and **YDST** instead
XYEND	The x and y ending position	Can use **AR0** and **AR2** instead

5.5.4.2 Solid Lines

DWGCTL:

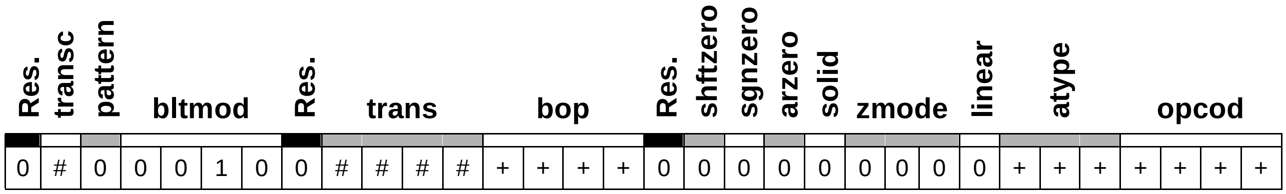
Res.	transc	pattern	bltmod				Res.	trans				bop				Res.	shftzero	sgnzero	arzero	solid	zmode			linear	atype			opcod			
0	0	0	0	0	1	0	0	#	#	#	#	+	+	+	+	0	1	0	0	1	0	0	0	0	+	+	+	+	+	+	+

- **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
- **opcod**: must be set to LINE_OPEN, LINE_CLOSE, AUTOLINE_OPEN, or AUTOLINE_CLOSE

Register	*Function*	*Comment / Alternate Function*
FCOL	Foreground color	

5.5.4.3 Lines That Use a Linestyle

DWGCTL:



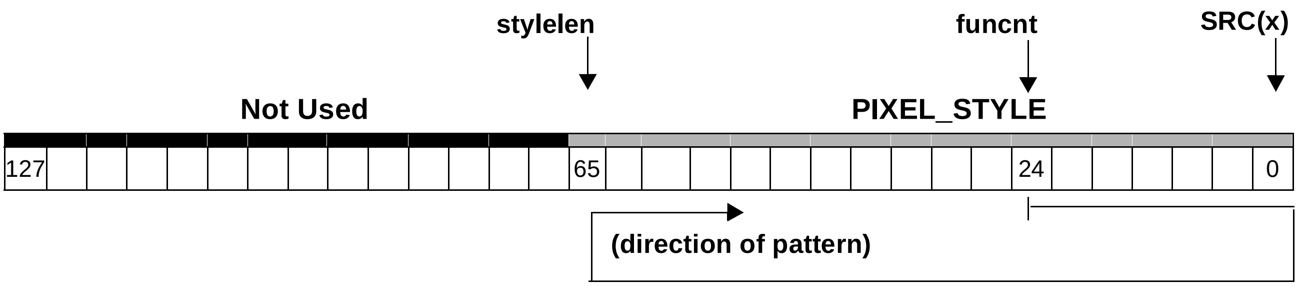
- **bop**: uses any Boolean operation if **atype** is RSTR; if **atype** is RPL, **bop** must be loaded with ‘0000’, ‘0011’, ‘1100’, or ‘1111’
- **atype**: can only be RPL or RSTR
- **opcod**: must be LINE_OPEN, LINE_CLOSE, AUTOLINE_OPEN, or AUTOLINE_CLOSE

Register	*Function*	*Comment / Alternate Function*
SHIFT	Linestyle length (stylelen), linestyle start point within the pattern (funcnt)	
SRC0	Linestyle pattern storage	
SRC1	Linestyle pattern storage	If **stylelen** is from 32-63
SRC2	Linestyle pattern storage	If **stylelen** is from 64-95
SRC3	Linestyle pattern storage	If **stylelen** is from 96-127
BCOL	Background color	If **transc** = 0
FCOL	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’

5.5.4.4 Lines with Depth

DWGCTL:

Res.	transc	pattern	bltmod				Res.	trans				bop				Res.	shftzero	sgnzero	arzero	solid	zmode			linear	atype			opcod			
0	0	0	0	0	1	0	0	#	#	#	#	1	1	0	0	0	0	0	0	0	#	#	#	0	+	+	+	+	+	+	+

- **atype**: must be either ZI or I
- **opcod**: 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 **zmode** <> NOZCMP or **atype** = ZI
DR2	The z major increment	Only if **zmode** <> NOZCMP or **atype** = ZI
DR3	The z diagonal increment	Only if **zmode** <> NOZCMP or **atype** = 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	
FCOL	Alpha value	Only if **pwidth** = 32, or **pwidth** = 16 and **dit555** = 1

✍ Note that the **MACCESS** register's **pwidth** field must not be set to 24 bits per pixel (PW24) when drawing lines with depth.

5.5.4.5 Polyline/Polysegment Using Vector Pseudo-DMA mode

The sequence for this operation is slightly different than the sequence for the other lines. First, the polyline primitive must be initialized:

- The global initialization registers (see section 5.5.3) must be set.
- Solid lines can be selected by initializing the registers as explained in subsection 5.5.4.2. Lines with linestyle can be selected by initializing the registers as explained in subsection 5.5.4.3. In both cases, AUTOLINE_OPEN or AUTOLINE_CLOSE must be selected.
- Bits 15-0 of the **OPMODE** register must be initialized to 0008h (for little endian processors) or 0208h (for big endian processors). It is important to access the **OPMODE** register (at least byte 0) since this will reset the state of the address generator. A 16-bit access is required (to prevent modification of the **dirDataSiz** field).

The polyline/polysegment will begin when either the DMAWIN space or the 8 MByte Pseudo-DMA window is written to.

The first double word that is transferred is loaded into the Address Generator. This double word contains one bit of ‘address select’ for each of the next 32 vector vertices to be sent to the drawing registers. These 32 bits are called the vector tags. The next 32 double word transfers contain the xy address data to be written to the drawing registers.

When a tag bit is set to zero (0), the address generator will force the index to the one of the **XYSTRT** registers without setting the bit to start the drawing engine. When the tag bit is set to one (1), the address generator will force the index to the one of the **XYEND** registers with the flag set to start the drawing engine.

When each double word of data is transferred, the Address Generator checks the associated tag bit and sends the appropriate 8-bit index to the Bus FIFO. When the 32nd (final) tag has been used, the next double word transfer reloads the Address Generator with the next 32 vector tags.

The Pseudo-DMA sequence can be interrupted by writing to byte 0 of the **OPMODE** register; this mechanism can be used when the last packet is incomplete.

DMA Vector Transfer Buffer Structure

	31 16	15 0
0	V31 . . .	Vn . . . V0
1	Y0	X0
2	Y1	X1
3	Y2	X2
:	:	
n	Yn + 1	Xn + 1
:	:	
31	Y30	X30
32	Y31	X31
33	V31 . . .	Vn . . . V0
34	Y0	X0
35	Y1	X1
36	Y2	X2
:	:	

5.5.5 Trapezoid / Rectangle Fill Programming

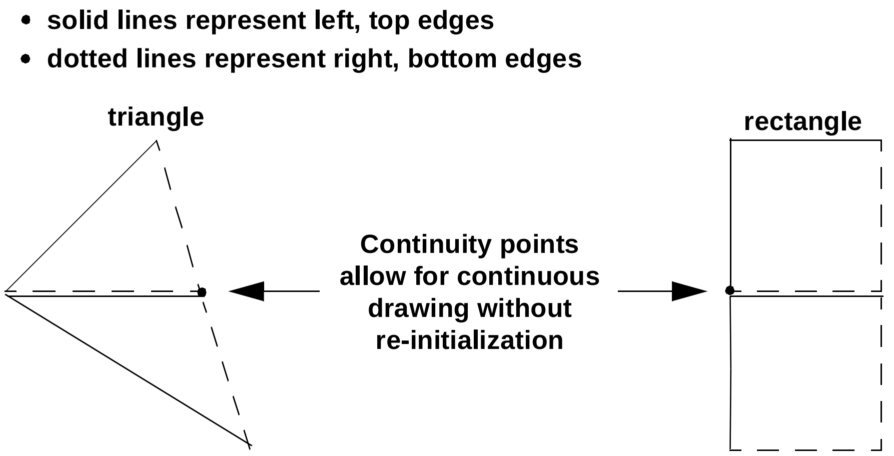
The following subsections list the registers that must be specifically programmed for constant and Gouraud shaded, patterned, and textured trapezoids, including rectangle and span line fills. Remember to program the registers listed in section 5.5.3 and in the tables in subsection 5.5.5.1 first. Also, ***the last register you program must be accessed in the 1D00h-1DFFh range in order to start the drawing engine***.

5.5.5.1 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.

Figure 5-1: Drawing Multiple Primitives



Trapezoids

The following registers must be initialized for trapezoid drawing:

Register	*Function*	*Comment / Alternate Function*
AR0	Left edge major axis increment: dYl yl_end - yl_start	
AR1	Left edge error term: errl (**sdxl** == XL_NEG) ? dXl + dYl - 1 : - dXl	
AR2	Left edge minor axis increment: -|dXl| -|xl_end - xl_start|	
AR4	Right edge error term: errr (**sdxr** == XR_NEG) ? dXr + dYr - 1 : - dXr	
AR5	Right edge minor axis increment: -|dXr| -|xr_end - xr_start|	
AR6	Right edge major axis increment: dYr yr_end - yr_start	
SGN	Vector quadrant	
FXBNDRY	Filled object x left and right coordinates	Can use **FXRIGHT** and **FXLEFT**
YDSTLEN	The y start position and number of lines	Can use **YDST** and **LEN** instead; **must** use **YDST** and **LEN** when destination address is linear (i.e.. **ylin** = 1, see **PITCH**)

Rectangles and Span Lines

The following registers must be initialized for rectangle and span line drawing:

Register	*Function*	*Comment / Alternate Function*
FXBNDRY	Filled object x left and right coordinates	Can use **FXRIGHT** and **FXLEFT**
YDSTLEN	The y start position and number of lines	Can use **YDST** and **LEN** instead; **must** use **YDST** and **LEN** when destination address is linear (i.e.. **ylin** = 1, see **PITCH**)

5.5.5.2 Constant Shaded Trapezoids / Rectangle Fills

DWGCTL:

	Res.	transc	pattern	bltmod				Res.	trans				bop				Res.	shftzero	sgnzero	arzero	solid	zmode			linear	atype			opcod			
TRAP	0	0	0	0	0	0	0	0	+	+	+	+	+	+	+	+	0	1	0	0	1	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	1	0	0

- **trans**: if **atype** is BLK (block mode[1]), 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 BLK, **bop** must be loaded with ‘1100’
- **atype**: can be RPL, RSTR, or BLK

Register	*Function*	*Comment / Alternate Function*
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

---

(1) ‘Block mode’ refers to the high bandwidth block mode function of SGRAM. It should be used whenever possible for the fastest performance, although certain restrictions apply (see the **atype** field of the **DWGCTL** register on page 4-49).

5.5.5.3 Patterned Trapezoids / Rectangle Fills

DWGCTL:

	Res.	transc	pattern	bltmod				Res.	trans				bop				Res.	shftzero	sgnzero	arzero	solid	zmode			linear	atype			opcod			
TRAP	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	1	0	0

- **transc:** if **atype** is BLK, an opaque background is not supported- the value of **transc** must be '1'
- **trans**: if **atype** is BLK, 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 BLK, **bop** must be loaded with '1100'
- **atype**: Can be RPL, RSTR, or BLK

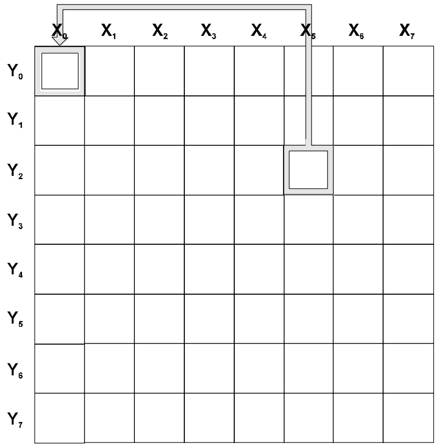
Register	*Function*	*Comment / Alternate Function*
PAT0	Pattern storage in Windows format	Use **SRC0, SRC1, SRC2, SRC3** for pattern storage in little endian format
PAT1		
SHIFT	Pattern origin offset	Only if **shftzero** = 0
BCOL	Background color	Only if **transc** = 0
FCOL	Foreground color	

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.

Offset = 0,0 Windows

X coordinates

Y coordinates

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	7	6	5	4	3	2	1	0
1	15	14	13	12	11	10	9	8
2	23	22	21	20	19	18	17	16
3	31	30	29	28	27	26	25	24
4	39	38	37	36	35	34	33	32
5	47	46	45	44	43	42	41	40
6	55	54	53	52	51	50	49	48
7	63	62	61	60	59	58	57	56

Offset = 5,2 Windows

X coordinates

Y coordinates

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	18	17	16	23	22	21	20	19
1	26	25	24	31	30	29	28	27
2	34	33	32	39	38	37	36	35
3	42	41	40	47	46	45	44	43
4	50	49	48	55	54	53	52	51
5	58	57	56	63	62	61	60	59
6	2	1	0	7	6	5	4	3
7	10	9	8	15	14	13	12	11

- 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.

Offset = 0,0 Little Endian

X coordinates

Y coordinates

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0	1	2	3	4	5	6	7
1	16	17	18	19	20	21	22	23
2	32	33	34	35	36	37	38	39
3	48	49	50	51	52	53	54	55
4	64	65	66	67	68	69	70	71
5	80	81	82	83	84	85	86	87
6	96	97	98	99	100	101	102	103
7	112	113	114	115	116	117	118	119

Offset = 5,2 Little Endian

X coordinates

Y coordinates

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	37	38	39	40	41	42	43	44
1	53	54	55	56	57	58	59	60
2	69	70	71	72	73	74	75	76
3	85	86	87	88	89	90	91	92
4	101	102	103	104	105	106	107	108
5	117	118	119	120	121	122	123	124
6	5	6	7	8	9	10	11	12
7	21	22	23	24	25	26	27	28

- 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’

5.5.5.4 Gouraud Shaded Trapezoids / Rectangle Fills

DWGCTL:

	Res.	transc	pattern	bltmod				Res.	trans				bop				Res.	shftzero	sgnzero	arzero	solid	zmode			linear	atype			opcod			
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*
DR0	The z start position	Only if **zmode** <> NOZCMP or **atype** = ZI
DR2	The z increment for x	Only if **zmode** <> NOZCMP or **atype** = ZI
DR3	The z increment for y	Only if **zmode** <> NOZCMP or **atype** = ZI
DR4	Red start position	
DR6	Red increment on x axis	
DR7	Red increment on y axis	
DR8	Green start position	
DR10	Green increment on x axis	
DR11	Green increment on y axis	
DR12	Blue start position	
DR14	Blue increment on x axis	
DR15	Blue increment on y axis	
FCOL	Alpha value	Only if **pwidth** = 32, or **pwidth** = 16 and **dit555** = 1.

Note that the **MACCESS** register’s **pwidth** field must not be set to 24 bits per pixel (PW24) when drawing Gouraud shaded trapezoids.

5.5.5.5 Trapezoids / Rectangle Fills Using Host Data

DWGCTL:

	Res.	transc	pattern	bltmod				Res.	trans				bop				Res.	shftzero	sgnzero	arzero	solid	zmode			linear	atype			opcod			
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*
OPMODE	Select DMA BLIT Write	
DR0	The z start position	Only if zmode <> NOZCMP or atype = ZI
DR2	The z increment for x	Only if zmode <> NOZCMP or atype = ZI
DR3	The z increment for y	Only if zmode <> NOZCMP or atype = ZI
FCOL	Alpha value	Only if **pwidth** = 32, or **pwidth** = 16 and **dit555** = 1.

Note that the **MACCESS** register's **pwidth** field must not be set to 24 bits per pixel (PW24) when drawing this type of trapezoid.

This type of primitive (TRAP_ILOAD) employs the same algorithm as Gouraud shaded trapezoids, with the exception that the pixel data comes from the host by means of an ILOAD operation.

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.

The complete steps to take for ILOAD (image load: Host -> RAM) operations are listed in 'ILOAD Programming' on page 5-46.

5.5.6 Bitblt Programming

The following subsections list the registers that must be specifically programmed for Bitblt operations. Remember to program the registers listed in section 5.5.3 and subsection 5.5.6.1 first. Also, ***the last register you program must be accessed in the 1D00h-1DFFh range in order to start the drawing engine***.

5.5.6.1 Address Initialization

XY Source Addresses

Register	*Function*	*Comment / Alternate Function*
AR0	Source end address	The last pixel of the first line
AR3	Source start address	
AR5	Source y increment	
FXBNDRY	Destination boundary (left and right)	Can use **FXRIGHT** and **FXLEFT**
YDSTLEN	The y start position and number of lines	Can use **YDST** and **LEN** instead

Linear Source Addresses

Register	*Function*	*Comment / Alternate Function*
AR0	Source end address	The last pixel of the source
AR3	Source start address	
FXBNDRY	Destination boundary (left and right)	Can use **FXRIGHT** and **FXLEFT**
YDSTLEN	The y start position and number of lines	Must use **YDST** and **LEN** when destination address is linear (i.e.. **ylin** = 1, see **PITCH**)

AR0 comprises 18 bits, so a maximum of 256 Kpixels can be blitted.

Patterning Operations

Register	*Function*	*Comment / Alternate Function*
FXBNDRY	Destination boundary (left and right)	Can use **FXRIGHT** and **FXLEFT**
YDSTLEN	The y start position and number of lines	Can use **YDST** and **LEN** instead; ***must*** use **YDST** and **LEN** when destination address is linear (i.e.. **ylin** = 1, see **PITCH**)

5.5.6.2 Two-operand Bitblts

DWGCTL:

	Res.	transc	pattern	bltmod				Res.	trans				bop				Res.	shftzero	sgnzero	arzero	solid	zmode			linear	atype			opcod			
XY	0	+	0	0	0	1	0	0	#	#	#	#	+	+	+	+	0	1	+	0	0	0	0	0	0	+	+	+	1	0	0	0
LIN.	0	+	0	0	1	1	1	0	#	#	#	#	+	+	+	+	0	1	1	0	0	0	0	0	1	+	+	+	1	0	0	0

- **transc:** must be ‘0’ if the **MACCESS** register’s **pwidth** field is set to 24 bits/pixel (PW24)
- **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*
SGN	Vector quadrant (1)	Only needs to be set when **sgnzero** = ‘0’
FCOL	Transparency color key	Only when **transc** = ‘1’
BCOL	Color key plane mask	Only when **transc** = ‘1’

(1) Sets major or minor axis and positive or negative direction for x and y.

5.5.6.3 Color Patterning 8 x 8

DWGCTL:

Res.	transc	pattern	bltmod				Res.	trans				bop				Res.	shftzero	sgnzero	arzero	solid	zmode			linear	atype			opcod			
0	+	1	0	0	1	0	0	#	#	#	#	+	+	+	+	0	1	1	0	0	0	0	0	0	+	+	+	1	0	0	0

- **transc:** must be ‘0’ if the **MACCESS** register’s **pwidth** field is set to 24 bits/pixel (PW24)
- **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*
AR0	When **pwidth** = PW8, PW16, or PW32: **AR0**<17:3> = **AR3**<17:3> When **pwidth** = PW8: **AR0**<2:0> = **AR3**<2:0> + 2 When **pwidth** = PW16: **AR0**<2:0> = **AR3**<2:0> + 4 When **pwidth** = PW32: **AR0**<2:0> = **AR3**<2:0> + 6 When **pwidth** = PW24: **AR0**<17:0> = **AR3**<17:0>+ 7	
AR3	Pattern address + x offset + (y offset * 32)	
AR5	32	
FCOL	Transparency color key	Only when **transc** = ‘1’
BCOL	Color key plane mask	Only when **transc** = ‘1’

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 5-36, 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 module 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):

		Pattern 0																***Pattern 1***															
	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

5.5.6.4 BitBlts With Expansion (Character Drawing) 1 bpp

DWGCTL:

Res.	transc	pattern	bltmod				Res.	trans				bop				Res.	shftzero	sgnzero	arzero	solid	zmode			linear	atype			opcod			
0	+	0	0	0	0	0	0	+	+	+	+	+	+	+	+	0	1	1	0	0	0	0	0	#	+	+	+	1	0	0	0

- **transc:** if **atype** is BLK, an opaque background is not supported- the value of **transc** must be ‘1’
- **trans**: if **atype** is BLK, 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 BLK, must be loaded with ‘1100’
- **atype**: can be RPL, RSTR, or BLK

Register	*Function*	*Comment / Alternate Function*
BCOL	Background color	Only when transc = ‘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.

5.5.6.5 BitBlts With Expansion (Character Drawing) 1 bpp Planar

DWGCTL:

Res.	transc	pattern	bltmod				Res.	trans				bop				Res.	shftzero	sgnzero	arzero	solid	zmode			linear	atype			opcod			
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*
SHIFT	Plane selection	
BCOL	Background color	Only when **transc** = ‘0’
FCOL	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 -> RAM) operations. You must take the following steps:

Step 1. Initialize the registers. Remember to program the registers listed in section 5.5.3 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 1D00h-1DFFh or 2000h-2DFFh range in order to start the drawing engine.

Step 3. Write the data in the appropriate format to either the DMAWIN or 8 MByte Pseudo-DMA memory ranges.

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 or 8 MByte Pseudo-DMA 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.

> ✎ 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.

  Total = INT((psiz * width * Nlines + 31) / 32)

- When the source is ***not linear***: the data is padded at the end of every line.

  Total = INT((psiz * width + 31) / 32) * 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*
OPMODE	Data format	A 16-bit access is required to prevent modification of the **dirDataSiz** field (bits 17:16), since direct frame buffer access may be concurrent
AR0	Total number of source pixels - 1	
AR3	Must be 0	
FXBNDRY	Destination boundary (left and right)	Can use **FXLEFT** and **FXRIGHT**
YDSTLEN	The y start position and length	Can use **YDST** and **LEN** instead; ***must*** use **YDST** and **LEN** when destination address is linear (i.e.. **ylin** = 1, see **PITCH**)

XY Addresses

Register	*Function*	*Comment / Alternate Function*
OPMODE	Data format	A 16-bit access is required to prevent modification of the **dirDataSiz** field (bits 17:16).
AR0	Number of pixels per line - 1	
AR3	Must be 0	
AR5	Must be 0	
FXBNDRY	Destination boundary (left and right)	Can use **FXLEFT** and **FXRIGHT**
YDSTLEN	The y start position and length	Can use **YDST** and **LEN** instead; ***must*** use **YDST** and **LEN** when destination address is linear (i.e.. **ylin** = 1, see **PITCH**)

5.5.7.2 ILOAD of Two-operand Bitblts

DWGCTL:

Res.	transc	pattern	bltmod				Res.	trans				bop				Res.	shftzero	sgnzero	arzero	solid	zmode			linear	atype			opcod			
0	+	0	+	+	+	+	0	#	#	#	#	+	+	+	+	0	1	+	0	0	0	0	0	+	+	+	+	1	0	0	1

- **transc:** must be ‘0’ if the **MACCESS** register’s **pwidth** field is set to 24 bits/pixel (PW24); must be ‘0’ when the **bltmod** field is anything other than BFCOL
- **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*
FCOL	Foreground color	For the BU32BGR and BU32RGB formats, depending on the **MACCESS** register’s **pwidth** setting, the following bits from **FCOL** are used: PW32: Bits 31:24 originate from **forcol**<31:24> PW16: Bit 15 originates from **forcol**<15> when **dit555** = 1
SGN	Scanning direction	Must be set only when **sgnzero** = 0
FCOL	Transparency color key	Only when **transc** = ‘1’
BCOL	Color key plane mask	Only when **transc** = ‘1’

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 Formats’ illustrations starting on page 5-8).

Table 5-3: ILOAD Supported Formats

Processor Type	bltmod	dmaDataSiz	pwidth	*Data Format*
Little endian	BFCOL	‘00’	PW8 PW16 PW24 PW32	8-bit A 16-bit A 24-bit A 32-bit A
	BU24RGB	‘00’	PW8 PW16 PW32	24-bit A 24-bit A 24-bit A
	BU24BGR	‘00’	PW8 PW16 PW32	24-bit B 24-bit B 24-bit B
	BU32RGB	‘00’	PW8 PW16 PW32	32-bit A 32-bit A 32-bit A
	BU32BGR	‘00’	PW8 PW16 PW32	32-bit B 32-bit B 32-bit B
	BUYUV	‘00’	PW8 PW16 PW32	YUV A YUV A YUV A
		‘01’	PW8 PW16 PW32	YUV B YUV B YUV B
Big endian	BFCOL	‘00’ ‘01’ ‘10’	PW8 PW16 PW32	8-bit B 16-bit B 32-bit A
	BU32RGB	‘10’	PW8 PW16 PW32	32-bit A 32-bit A 32-bit A
	BU32BGR	‘10’	PW8 PW16 PW32	32-bit B 32-bit B 32-bit B
	BUYUV	‘00’	PW8 PW16 PW32	YUV C YUV C YUV C
		‘01’	PW8 PW16 PW32	YUV D YUV D 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			opcod			
0	+	0	+	+	+	+	0	+	+	+	+	+	+	+	+	0	1	1	0	0	0	0	0	1	+	+	+	1	0	0	1

- **transc:** if **atype** is BLK, an opaque background is not supported- the value of **transc** must be '1'
- **bltmod**: must be set to either BMONOLEF or BMONOWF
- **trans**: if atype is BLK, 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 BLK, **bop** must be loaded with '1100'
- **atype**: must be set to either RPL, RSTR, or BLK

Register	*Function*	*Comment / Alternate Function*
BCOL	Background color	Only when **transc** = '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 Formats' illustrations starting on page 5-8).

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 Operations

The MGA-1064SG supports various scaling operations:

ILOAD_SCALE	Horizontal scaling by pixel replication
ILOAD_FILTER	Horizontal scaling with simple filtering
ILOAD_HIQH	Horizontal scaling with high quality filtering using linear interpolation
ILOAD_HIQHV	Horizontal and vertical scaling with high quality filtering using linear interpolation

5.5.8.1 Horizontal scaling

Horizontal scaling uses ILOAD_SCALE (pixel replication) or ILOAD_FILTER (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. For ILOAD_HIQH and ILOAD_HIQHV, the maximum horizontal factor is 8x, and the SRC_X_DIMEN must be 2 or higher.
- 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 for ILOAD_SCALE, ILOAD_FILTER, and ILOAD_HIQH. Table 5-6 shows all the valid format combinations for ILOAD_HIQHV. In all cases, **pwidth** may be set to PW8, PW16, or PW32 (but not PW24). The structure of the buffers to be transferred is defined for each data format (as shown the ‘Pixel Formats’ illustrations starting on page 5-8).

Table 5-5: Scaling Supported Formats: ILOAD_SCALE, ILOAD_FILTER, and ILOAD_HIQH

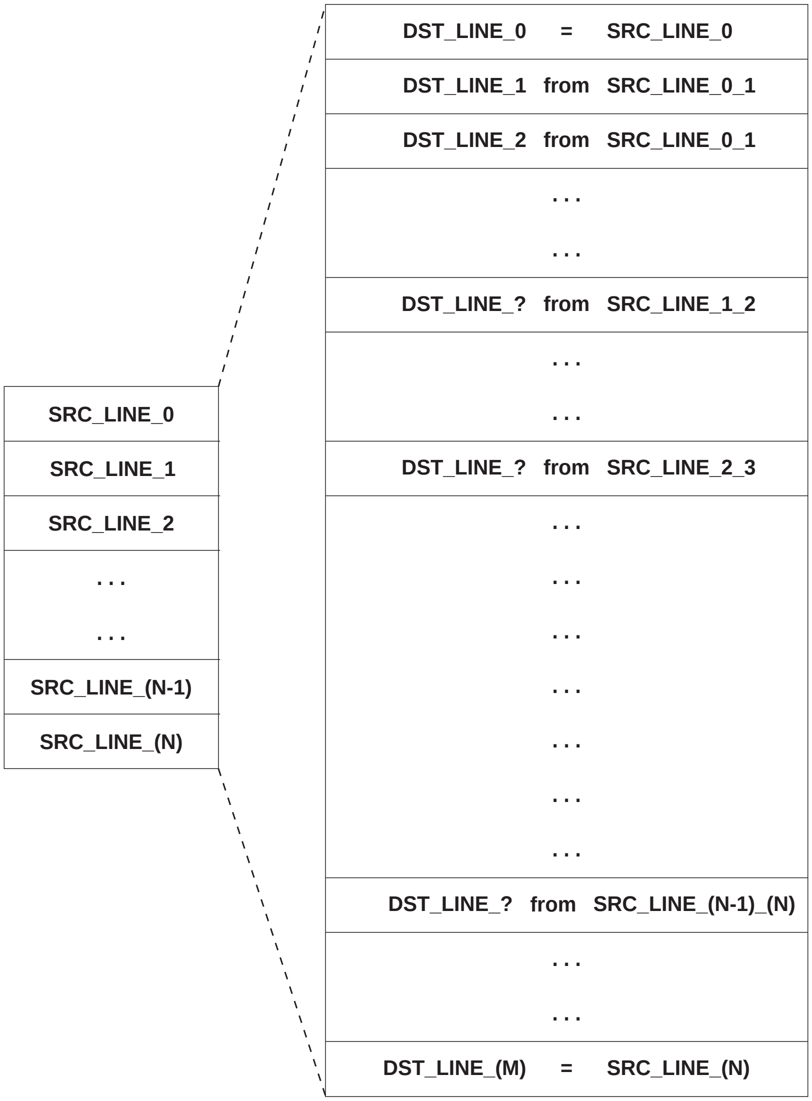
Processor Type	**bltmod**	**dmaDataSiz**	*Data Format*
Little endian	BU24RGB	00	24-bit A
	BU24BGR	00	24-bit B
	BU32RGB	00	32-bit A
	BU32BGR	00	32-bit B
	BUYUV	00	YUV A
	BUYUV	01	YUV B
Big endian	BU32RGB	10	32-bit A
	BU32BGR	10	32-bit B
	BUYUV	00	YUV C
	BUYUV	01	YUV D

Table 5-6: Scaling Supported Formats: ILOAD_HIQHV

Processor Type	**bltmod**	**dmaDataSiz**	*Data Format*
Little endian	BU32RGB	00	32-bit C
	BU32BGR	00	32-bit D
	BUYUV	00	YUV E
	BUYUV	01	YUV F
Big endian	BU32RGB	10	32-bit C
	BU32BGR	10	32-bit D
	BUYUV	00	YUV G
	BUYUV	01	YUV H

(1) The data is transferred as shown on the next page:

Figure 5-2: ILOAD_HIQHV Beta Programming and Data Transfer to the Chip



$$\text{DST\_LINE\_?} = \frac{\text{SRC\_BUF\_0}\,(16 - \text{beta}') + \text{SRC\_BUF\_1}\,(\text{beta}')}{16}$$

Where:

SRC_BUF_0 represents SRC_LINE_(X-1)

SRC_BUF_1 represents SRC_LINE_(X)

(X depends on the current scan source position.)

beta	***beta′***
0	16
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15

To produce:

- DST_LINE_0 = SRC_LINE_0

  beta = 0
  SRC_BUF_0 = ‘don’t care’ (but must be present)
  SRC_BUF_1 = SRC_LINE_0

- DST_LINE_1 from SRC_LINE_0

  beta = from 1 to 15
  SRC_BUF_0 = SRC_LINE_0
  SRC_BUF_1 = SRC_LINE_1

- DST_LINE_? from SRC_LINE_(X-1)_(X)

  beta = from 0 to 15
  SRC_BUF_0 = SRC_LINE_(X-1)
  SRC_BUF_1 = SRC_LINE_(X)

- DST_LINE_(M) = SRC_LINE_(N)

  beta = 0
  SRC_BUF_0 = ‘don’t care’ (but must be present)
  SRC_BUF_1 = SRC_LINE_(N)

BU32RGB (32-bit C):

	MSB			LSB	
SRC_BUF_0=	A00	R00	G00	B00	Pixel 0
	A01	R01	G01	B01	Pixel 1
	A02	R02	G02	B02	Pixel 2
	. . .	. . .	. . .	. . .	. . .

	MSB			LSB	
SRC_BUF_1=	A10	R10	G10	B10	Pixel 0
	A11	R11	G11	B11	Pixel 1
	A12	R12	G12	B12	Pixel 2
	. . .	. . .	. . .	. . .	. . .

	MSB			LSB	
DW to Send to the Chip	G00	R10	G10	B10	DW0
	G01	R11	G11	B11	DW1
	G02	R12	G12	B12	DW2
	. . .	. . .	. . .	. . .	. . .

BU32BGR (32-bit D):

	MSB			LSB	
SRC_BUF_0=	A00	B00	G00	R00	Pixel 0
	A01	B01	G01	R01	Pixel 1
	A02	B02	G02	R02	Pixel 2
	. . .	. . .	. . .	. . .	. . .

	MSB			LSB	
SRC_BUF_1=	A10	B10	G10	R10	Pixel 0
	A11	B11	G11	R11	Pixel 1
	A12	B12	G12	R12	Pixel 2
	. . .	. . .	. . .	. . .	. . .

	MSB			LSB	
DW to Send to the Chip	G00	B10	G10	R10	DW0
	G01	B11	G11	R11	DW1
	G02	B12	G12	R12	DW2
	. . .	. . .	. . .	. . .	. . .

BUYUV (YUV E):

SRC_BUF_0=

MSB			LSB	
V00	Y01	U00	Y00	Pixel 0_1
V02	Y03	U02	Y02	Pixel 2_3
V04	Y05	U04	Y04	Pixel 4_5
. . .	. . .	. . .	. . .	. . .

SRC_BUF_1=

MSB			LSB	
V10	Y11	U10	Y10	Pixel 0_1
V12	Y13	U12	Y12	Pixel 2_3
V14	Y15	U14	Y14	Pixel 4_5
. . .	. . .	. . .	. . .	. . .

DW to Send to the Chip

MSB			LSB	
V10	Y11	U10	Y10	DW0
V00	Y01	U00	Y00	DW1
V12	Y13	U12	Y12	DW2
V02	Y03	U02	Y02	DW3
V14	Y15	U14	Y14	DW4
V04	Y05	U04	Y04	DW5
. . .	. . .	. . .	. . .	. . .

BUYUV (YUV F):

SRC_BUF_0=

MSB			LSB	
Y01	V00	Y00	U00	Pixel 0_1
Y03	V02	Y02	U02	Pixel 2_3
Y05	V04	Y04	U04	Pixel 4_5
. . .	. . .	. . .	. . .	. . .

SRC_BUF_1=

MSB			LSB	
Y11	V10	Y10	U10	Pixel 0_1
Y13	V12	Y12	U12	Pixel 2_3
Y15	V14	Y14	U14	Pixel 4_5
. . .	. . .	. . .	. . .	. . .

DW to Send to the Chip

MSB			LSB	
Y11	V10	Y10	U10	DW0
Y01	V00	Y00	U00	DW1
Y13	V12	Y12	U12	DW2
Y03	V02	Y02	U02	DW3
Y15	V14	Y14	U14	DW4
Y05	V04	Y04	U04	DW5
. . .	. . .	. . .	. . .	. . .

BUYUV (YUV G):

	MSB			LSB	
SRC_BUF_0=	Y00	U00	Y01	V00	Pixel 0_1
	Y02	U02	Y03	V02	Pixel 2_3
	Y04	U04	Y05	V04	Pixel 4_5
	. . .	. . .	. . .	. . .	. . .

	MSB			LSB	
SRC_BUF_1=	Y10	U10	Y11	V10	Pixel 0_1
	Y12	U12	Y13	V12	Pixel 2_3
	Y14	U14	Y15	V14	Pixel 4_5
	. . .	. . .	. . .	. . .	. . .

	MSB			LSB	
DW to Send to the Chip	Y10	U10	Y11	V10	DW0
	Y00	U00	Y01	V00	DW1
	Y12	U12	Y13	V12	DW2
	Y02	U02	Y03	V02	DW3
	Y14	U14	Y15	V14	DW4
	Y04	U04	Y05	V04	DW5
	. . .	. . .	. . .	. . .	. . .

BUYUV (YUV H):

	MSB			LSB	
SRC_BUF_0=	U00	Y00	V00	Y01	Pixel 0_1
	U02	Y02	V02	Y03	Pixel 2_3
	U04	Y04	V04	Y05	Pixel 4_5
	. . .	. . .	. . .	. . .	. . .

	MSB			LSB	
SRC_BUF_1=	U10	Y10	V10	Y11	Pixel 0_1
	U12	Y12	V12	Y13	Pixel 2_3
	U14	Y14	V14	Y15	Pixel 4_5
	. . .	. . .	. . .	. . .	. . .

	MSB			LSB	
DW to Send to the Chip	U10	Y10	V10	Y11	DW0
	U00	Y00	V00	Y01	DW1
	U12	Y12	V12	Y13	DW2
	U02	Y02	V02	Y03	DW3
	U14	Y14	V14	Y15	DW4
	U04	Y04	V04	Y05	DW5
	. . .	. . .	. . .	. . .	. . .

5.5.8.2 Vertical Scaling

- For ILOAD_SCALE, ILOAD_FILTER, and ILOAD_HIQH, vertical scaling is performed using the BITBLT function to do line replication. This type of scaling operation is divided into two phases one for horizontal scaling and the other for vertical scaling.
- In ILOAD_HIQHV horizontal and vertical scaling is done in a single phase. For line drawn, two source lines must be transferred. Multiple lines can be drawn with the same **beta** factor.

5.5.8.3 Scaling Steps

The following steps must be executed for scaling:

Step 1. Initialize the scaling engine as specified in subsection 5.5.8.4. Also, remember to program the registers listed in section 5.5.3. ***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 1D00h-1DFFh range in order to start the drawing engine.

DWGCTL:

Res.	transc	pattern	bltmod				Res.	trans				bop				Res.	shftzero	sgnzero	arzero	solid	zmode			linear	atype			opcod			
0	0	0	+	+	+	+	0	0	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	+	+	+	+

- **bltmod**: Can be set to BUYUV, BU32RGB, BU32BGR, BU24BGR, BU24RGB, or BU24GBR for ILOAD_SCALE, ILOAD_FILTER and ILOAD_HIQH. Can be set to BUYUV, BU32RGB, or BU32BGR for ILOAD_HIQHV.
- **opcod**: can be set to ILOAD_SCALE, ILOAD_FILTER, ILOAD_HIQH or ILOAD_HIQHV

Register / Space	*Field*	*Comment / Alternate Function*
LEN	Number of lines to draw and beta factor.	Without line replication: When ILOAD_HIQHV, **length** must be set to 1, and **beta** must be programmed. When not ILOAD_HIQHV, **beta** must be set to 0.

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 5-46), 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):

Total = INT((psiz * width + 31) / 32) * factor * Nlines

Legend:

Total:	The number of dwords to transfer
width:	The number of pixels per line to write
factor:	The factor operator, according to Table 5-7
Nlines:	The number of lines to write
psiz:	The source size, according to Table 5-8

Table 5-7: Source Factor

opcod	bltmod	*Factor*
ILOAD_SCALE ILOAD_FILTER ILOAD_HIQH		1
ILOAD_HIQHV	BUYUV	2
	BU32RGB BU32BGR	1

Table 5-8: Source Size

bltmod	psiz
BUYUV	16
BU24RGB	24
BU24BGR	24
BU32RGB	32
BU32BGR	32

Step 4. For ILOAD_HIQHV, skip this step. Initialize the drawing engine for vertical scaling. The last register you program must be accessed in the 1D00h-1DFFh range in order to start the drawing engine.

Register / Space	*Function*	*Comment / Alternate Function*
LEN	Number of lines	Replicated lines
DWGCTL	040C6008h (BITBLT)	

Step 5. Repeat Steps 2 to 4 until the end of the scaling sequence.

5.5.8.4 Scaling Initialization

ILOAD_SCALE

Register	*Function*	*Comment / Alternate Function*
OPMODE	Data format	A 16-bit access is required to prevent modification of the **dirDataSiz** field (bits 17:16).
AR0	DST_END_ADDRESS - DST_Y_INCREMENT	
AR2	SRC_X_DIMENSION	
AR3	DST_START_ADDRESS - DST_Y_INC	
AR5	DST_Y_INC	Only required if vertical scaling is used
AR6	SRC_X_DIMEN - DST_X_DIMEN	
FXBNDRY	Destination boundary (left and right)	Can use **FXLEFT** and **FXRIGHT**
YDST	Y start position	

ILOAD_FILTER

Register	*Function*	*Comment / Alternate Function*
OPMODE	Data format	A 16-bit access is required to prevent modification of the **dirDataSiz** field (bits 17:16).
AR0	DST_END_ADDRESS - DST_Y_INC	
AR2	(2 * SOURCE_X_DIMEN - 1)	
AR3	DST_START_ADDRESS - DST_Y_INC	
AR5	DST_Y_INC	Only required if vertical scaling is used
AR6	(2 * SRC_X_DIMEN - 1) - DST_X_DIMEN	
FXBNDRY	Destination boundary (left and right)	Can use **FXLEFT** and **FXRIGHT**
YDST	Y start position	

ILOAD_HIQH and ILOAD_HIQHV

Register	*Function*	*Comment / Alternate Function*
OPMODE	Data format	A 16-bit access is required to prevent modification of the **dirDataSiz** field (bits 17:16).
AR0	DST_END_ADDRESS - DST_Y_INC	
AR2	$\frac{(SRC\_X\_DIMEN - 1) << 16}{(DST\_X\_DIMEN - 1)} + 1$	
AR3	DST_START_ADDRESS - DST_Y_INC	
AR5	DST_Y_INC	Only required if performing vertical scaling using the BITBLT function.
AR6	$\frac{(SRC\_X\_DIMEN - DST\_X\_DIMEN) << 16}{(DST\_X\_DIMEN - 1)}$	
FXBNDRY	Destination boundary (left and right)	Can use **FXLEFT** and **FXRIGHT**
YDST	Y start position	

5.5.9 IDUMP Programming

The following subsections list the registers that must be specifically programmed for IDUMP (image dump: SD/SGRAM -> Host) operations. You must take the following steps:

Step 1. Initialize the registers. Remember to program the registers listed in section 5.5.3.

DWGCTL:

Res.	transc	pattern	bltmod				Res.	trans				bop				Res.	shftzero	sgnzero	arzero	solid	zmode			linear	atype			opcod			
0	0	0	+	+	+	+	0	0	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	#	0	0	0	1	0	1	0

- **bltmod**: can be BU32BGR, BU32RGB, BU24BGR, or BU24RGB. See Table 5-10.

Register	*Function*	*Comment / Alternate Function*
OPMODE	Data format	A 16-bit access is required to prevent modification of the **dirDataSiz** field (bits 17:16). There is no need to program the **dmamod** field of the **OPMODE** register - reading the DMAWIN or the 8 MByte Pseudo-DMA window is sufficient to trigger the IDUMP.
AR0	Source end address	
AR3	Source start address	
AR5	Source y increment	Not required for a linear source
FXBNDRY	Destination boundary. Left = 0; Right = number of pixels per line minus 1	Can use **FXLEFT** and **FXRIGHT**
YDSTLEN	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 1D00h-1DFFh range in order to start the drawing engine.

Step 3. Read the data in the appropriate format from either the DMAWIN or 8 MByte Pseudo-DMA memory ranges.

Since the IDUMP operation generates the addresses for the destination, the addresses of the data are not used while accessing either the DMAWIN or 8 MByte Pseudo-DMA window. Subsequently, move string instructions can be used through the 7KByte space of either the DMAWIN or 8 MByte Pseudo-DMA window to read the data from the MGA-1064SG. 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 5-9

Table 5-9: IDUMP Source Size

bltmod	pwidth	psiz
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 5-10 shows all the valid format combinations. The structure of the buffers to be transferred is defined for each data format (as shown the ‘Pixel Formats’ illustrations starting on page 5-8).

Table 5-10: IDUMP Supported Formats

Processor Type	bltmod	dmaDataSiz	pwidth	*Data Format*
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

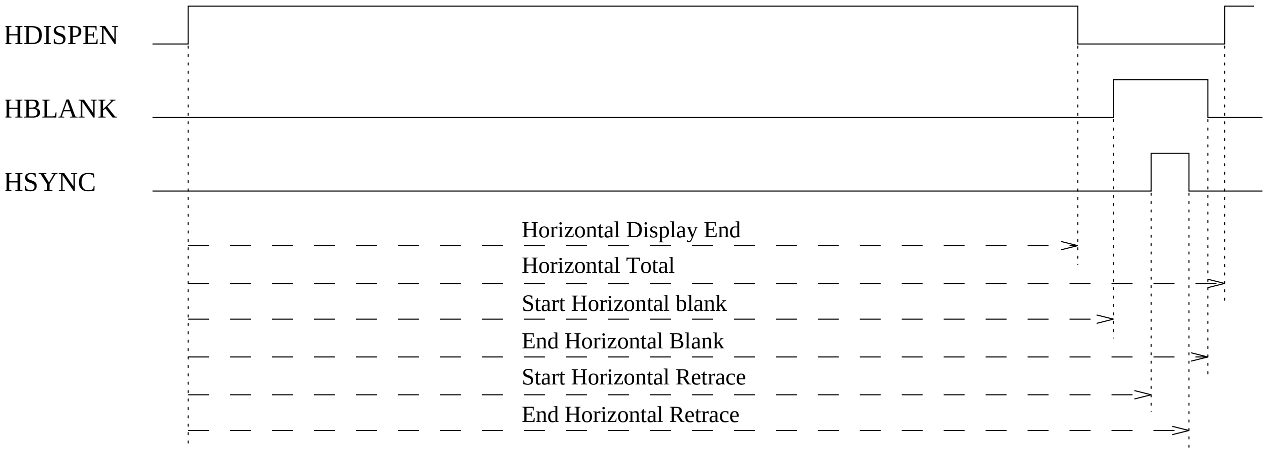
The CRTC can be programmed in one of two modes: VGA mode or Power Graphic mode. The **mgamode** field of the **CRTCEXT3** register is used to select the operating mode.

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 set to ‘0’. The **page** field of **CRTCEXT4** can be used to select a different page of RAM in which to write pixels.

5.6.1 Horizontal Timing

Figure 5-3: CRTC Horizontal Timing



In VGA mode, the horizontal timings are defined by the following VGA register fields:

htotal<7:0>	Horizontal total. Should be programmed with the total number of displayed characters plus the non-displayed characters minus 5.
hdispend<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’.

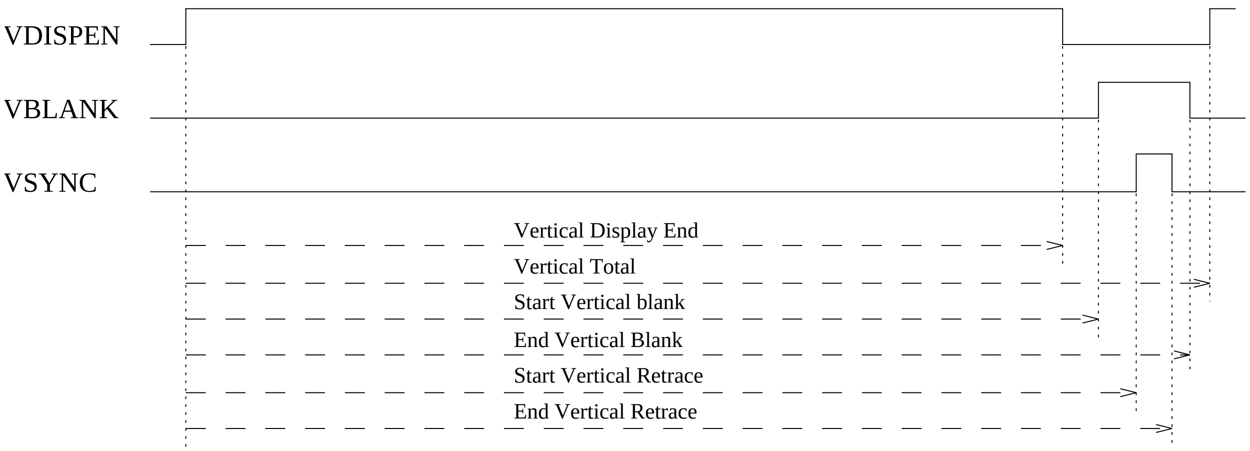
The units of the horizontal counter are ‘character clocks’ for VGA mode, or 8 pixels in Power Graphic mode. The **scale** field of the **CRTCEXT3** register is used to bring the VCLK clock down to an ‘8 pixel’ clock.

The suggested scale factor settings are shown in the following table:

Bits/Pixel	**scale**
8	‘000’
16	‘001’
24	‘010’
32	‘011’
2G8V16	‘001’
G16V16	‘011’

5.6.2 Vertical Timing

Figure 5-4: CRTC Vertical Timing



In VGA mode, the vertical timings are defined by the following VGA register fields:

vtotal<9:0>	Vertical total. Should be programmed with the total number of displayed lines plus the non-displayed lines minus 2.
vdispend<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 fields are extended to support a larger display area:

vtotal<11:0>	Vertical total
vdispend<10:0>	Vertical display end
vblkstr<11:0>	Vertical blanking start
vsyncstr<11:0>	Start vertical retrace
linecomp<10:0>	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:

startadd<15:0>	Start address
offset<7:0>	Logical line width of the screen. This is programmed with the number of double or single words in one character line.
curpos<15:0>	Cursor position
prowscan<4:0>	Preset row scan
maxscan<4:0>	Maximum scan line
currowstr<4:0>	Row scan cursor begins
currowend<4:0>	Row scan cursor ends
curoff<4:0>	Cursor off
undrow<4:0>	Horizontal row scan where underline will occur
curskew<1:0>	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 fields are extended in order to support both a larger display, and up to 8 Mbytes of memory.

startadd<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**, **prowscan**, **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 setting the **hzoom** field of the **XZOOMCTRL** register.

5.6.4 Programming in VGA Mode

The VGA CRTC of the MGA-1064SG 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-1064SG’s CRTC.

In the formulas which follow, ‘cc’ represents the number of video clocks 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:

Display Mode	dotmode	dotclkrt	pelwidth	*cc*
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

In VGA mode, Tvclk is equivalent to Tpixclk.

The following factors (in MCLKs) must be applied to the formulas which follow, according to whether text or graphics are being displayed:

Variable	*VGA Text*	*VGA Graphics*
A	54	33
B	1	1
C	11	8
D	120	51

Using these values, we can determine the following rules:

1. (cc * ((H_total - Byte_pan) - (H_dispend + MAX(H_dispskew + 2, H_syncstr - H_dispend)) + 1) - 3) * Tvclk >= A * Tmclk
2. (cc * 4 - 2) * Tvclk >= A * Tmclk
3. cc * Tvclk >= B * Tmclk
4. (cc * ((H_total - Byte_pan) - H_dispend + 2) - 2) * Tvclk >= (A + C) * Tmclk
5. (cc * ((H_total - Byte_pan) - (H_dispend + MAX(H_dispskew + 2, H_syncstr - H_dispend)) + 2) - 3) * Tvclk >= (A + C) * Tmclk
6. (cc * ((H_total - Byte_pan) - H_dispend + 3) - 2) * Tvclk >= (D + C) * Tmclk

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 formula:

$$\textbf{offset} = \frac{\text{video pitch * bpp * fsplit}}{128}$$

Where:
- ‘bpp’ is the pixel width, expressed in bits per pixel, and
- ‘video pitch’ is the number of pixels per line in the frame buffer (including pixels that are not visible).
- fsplit = 2 in split mode; 1 in all other modes

For example, with a 16 bit/pixel frame buffer at a resolution of 1280 x 1024:

offset = (1280 x 16)/128 = 160

Depending on the pixel width (bpp), the video pitch must be a multiple of one of the following:

bpp	*Multiple of*
8	16
16	8
24	16
32	4

The **startadd** field represents the number of pixels to offset the start of the display by.

$$\textbf{startadd} = \frac{\text{address of the first pixel to display * fsplit}}{\textit{factor}}$$

Depending on the pixel depth, the following *factor*s must be used:

bpp	*Factor*
8	8
16	4
24	8
32	2

For example, to program **startadd** to use an offset of 64 with a 16 bit/pixel frame buffer, **startadd** = 64/4 = 16. With a 24 bit/pixel frame buffer, **startadd** = 64/8 = 8.

Note that when accessing the three-part **startadd** field, the portion which is located in **CRTCEXT0** must ***always*** be written, and it must always be written ***last*** (that is, written ***after*** the other portions of **startadd**, which are located in **CRTCC** and **CRTCD**). The change of start address will take effect at the beginning of the next horizontal retrace following the write to **CRTCEXT0**. Display will continue at the next line, using the new **startadd** value. This arrangement permits page flipping at any line, with no tearing occuring within the line.

To avoid tearing between lines within a frame, software can poll either **vcount** or the **vretrace** field of **INSTS1**, or use the VSYNC interrupt to update **CRTCEXT0** between frames.

Note that the Attributes Controller (ATC) is not available in Power Graphic mode.

There is no overscan in Power Graphic mode, therefore:

htotal+5 == **hblkend**+1
hdispend+1 == **hblkstr**+1

The End Horizontal Blank value must always be greater that **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 can also be incorporated into the IOG signal, if the **iogsyncdis** field of the **XGENCTRL** register is set to ‘0’.

The composite sync is always active low. Note that the following values must be programmed in Power Graphic mode.

- **hsyncdel** = 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

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 using the **hzoom** field of the **XZOOMCTRL** register. To implement the zoom function, pixels are duplicated within the DAC, and the memory address generator advances at a reduced rate.

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.

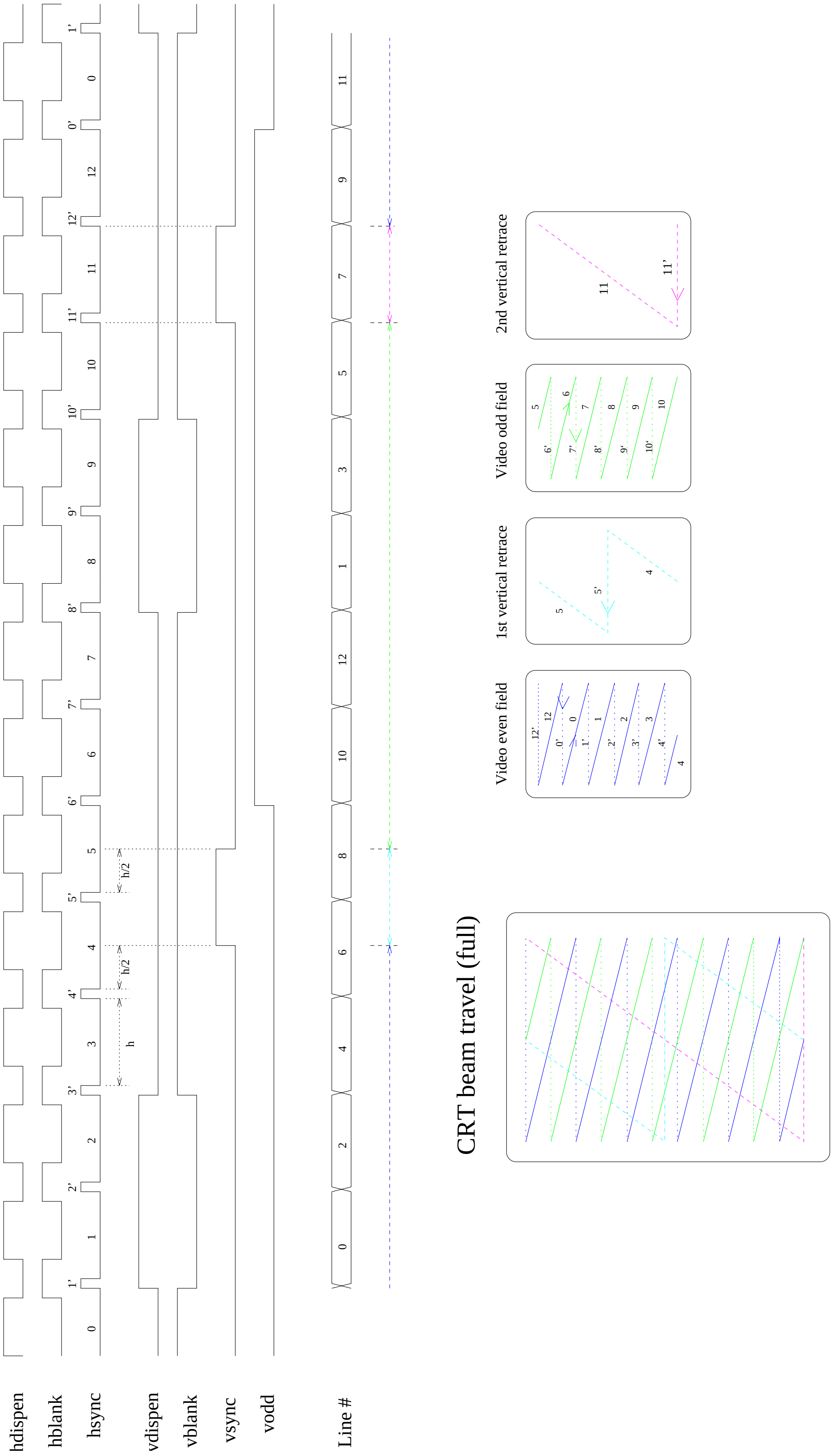
CRTC Latency Formulas

This section presents several rules that must be followed in Power graphic mode in order to adhere to the latency constraints of the MGA-1064SG’s CRTC.

In the formulas below, ‘VC’ represents the number of PIXCLKs per video clock. VC = 8/(SCALE + 1).

1. ((8 * (H_total - H_syncstr) - 2 (VC)) + 3) * Tpixclk >= 95 * Tmclk
2. ((31 * VC) + 3) * Tpixclk >= 95 * Tmclk
3. VC >= 1 * Tmclk
4. ((8 * (H_total - H_dispend) + 7 * (VC)) + 3) * Tpixclk >= 141 * Tmclk
5. ((8 * (H_total - H_syncstr + 1) - 2 (VC)) + 3) * Tpixclk >= 103 * Tmclk
6. ((8 * (H_total - H_dispend + 1) + 7 (VC)) + 3) * Tpixclk >= 149 * Tmclk

Figure 5-5: Video Timing in Interlace Mode



5.7 Video Interface

5.7.1 Operation Modes

The MGA-1064SG’s DAC can operate in one of five modes, depending on the values of the **mgamode** field of the **CRTCEXT3** register and the **depth** field of the **XMULCTRL** DAC register, as shown below:

mgamode	**depth**	***Mode Selected***
0	XXX	VGA
1	000	Pseudo Color (BPP8)
1	001	True Color (BPP15)
1	010	True Color (BPP16)
1	011	True Color (BPP24)
1	100	Direct Color (BPP32DIR)
1	101	Split Mode (2G8V16)
1	110	Split Mode (G16V16)
1	111	True Color (BPP32PAL)

5.7.1.1 VGA Mode

In VGA mode, the data to be displayed comes from the MGA-1064SG’s VGA Attribute Controller (ATC). The data from the ATC is used as an address for the three-LUT RAM. The pixel read mask can be applied to the data before it passes through the palette. The hardware cursor and keying functions are not supported in VGA mode.

Red LUT	P7	P6	P5	P4	P3	P2	P1	P0
Green LUT	P7	P6	P5	P4	P3	P2	P1	P0
Blue LUT	P7	P6	P5	P4	P3	P2	P1	P0

- The frequency of the pixel PLL is determined by the **clksel** field of the VGA **MISC** register, which selects the proper set of registers for the PLL. The frequency can also be changed via the **XPIXPLLM**, **XPIXPLLN**, and **XPIXPLLP** registers.
- Horizontal zooming is not supported in VGA mode.

5.7.1.2 Pseudo Color Mode

In Pseudo Color mode (BPP8), the data from the memory is sent to the DAC’s internal FIFO via the memory controller. The data is then used as an address for the three-LUT RAM (as for VGA mode). No keying is supported in BPP8, but a hardware cursor is available.

- The pixel PLL should use register set ‘C’ register set, so as to not change the frequency of the VGA PLL sets.
- Horizontal zooming is supported in pseudo color mode.

5.7.1.3 True Color Mode

The four true color modes supported by MGA-1064SG are BPP15, BPP16, BPP24, and BPP32PAL. In these modes, the pixel data from the internal DAC's FIFO is mapped to the LUT addresses as shown in the following illustrations:

15-Bit True Color (BPP15)

Red LUT	P15	0	0	P14	P13	P12	P11	P10
Green LUT	P15	0	0	P9	P8	P7	P6	P5
Blue LUT	P15	0	0	P4	P3	P2	P1	P0

- Bit 15 can be used as an overlay color (it selects another LUT table in the RAM) and can be masked out by the **alphaen** field of the **XGENCTRL** register (when at '0').

16-bit True Color (BPP16)

Red LUT	0	0	0	P15	P14	P13	P12	P11
Green LUT	0	0	P10	P9	P8	P7	P6	P5
Blue LUT	0	0	0	P4	P3	P2	P1	P0

24-bit True Color (BPP24 and BPP32PAL)

Red LUT	P23	P22	P21	P20	P19	P18	P17	P16
Green LUT	P15	P14	P13	P12	P11	P10	P9	P8
Blue LUT	P7	P6	P5	P4	P3	P2	P1	P0

In BPP24, the pixel data in the FIFO is unpacked before it enters the pixel pipeline, since each slice contains 2 2/3 24-bit pixels. In BPP32PAL, each pixel is 32 bits wide, but the eight MSBs are not used since they do not contain any color information.

- Keying is not available for true color modes, the but hardware cursor and horizontal zooming are supported.
- Register set 'C' should be used to program the pixel PLL.

5.7.1.4 Direct Color Mode (BPP32DIR)

In direct color mode, each pixel in the FIFO is composed of a 24-bit color portion and an 8-bit alpha portion. The 24-bit portion is sent directly to the DACs (that is, each color is directly applied on each DAC input). The alpha portion of the pixel can be used for color keying (refer to the **XCOLKEYH** register description) and may be displayed as a pseudo color pixel, depending on the outcome of the color comparison.

- As in all non-VGA modes, the hardware cursor and horizontal zooming are available.
- Register set 'C' should be used to program the pixel PLL.

5.7.1.5 Split modes (2G8V16 and G16V16)

Two split modes are supported by the DAC: 2G8V16 and G16V16.

In 2G8V16 mode, the video resolution is only half of the graphics resolution, so the DAC must average adjacent pixels to create the same effective resolution. The averaging is done on a color-by-color basis. For example: the red component of V0 with the red component of V2, the green component of V0 with the green component of V2, and so on.

Pixel	*Video*		*Graphics*
0	alpha0	V0	G0
1	alpha0	(V0+V2)/2	G1
2	alpha2	V2	G2
3	alpha2	(V2+V4)/2	G3
4	alpha4	V4	G4
5	alpha4	(V4+V6)/2	G5
6	alpha6	V6	G6
7	alpha6	(V6+V8)/2	G7

If the last pixel of a line is a video pixel, it is replicated since it cannot be averaged. The format of the video pixel is ‘5:5:5 + alpha’ and is sent directly to the DACs, since the graphics pixel must use the LUT (in 8-bit pseudo color). The video bits are mapped to the DACs as follows:

Red LUT	P14	P13	P12	P11	P10	P14	P13	P12
Green LUT	P9	P8	P7	P6	P5	P9	P8	P7
Blue LUT	P4	P3	P2	P1	P0	P4	P3	P2

Unlike the case for 2G8V16, in G16V16 mode the video and graphics resolution are the same, so no averaging is required. The graphics information is in 15-bit format, and the video is in the same format as for 2G8V16 mode. The video pixel can pass through the LUT or it can go directly to the DAC, depending on the **videopal** field the **XMULCTRL** DAC register. The pixel (video or graphics) that does not pass through the palette is sent directly to the DACs like a 2G8V16 video pixel. If the video pixel passes through the palette, the bits are then mapped on the LUT as follows:

Red LUT	0	0	1	P14	P13	P12	P11	P10
Green LUT	0	0	1	P9	P8	P7	P6	P5
Blue LUT	0	0	1	P4	P3	P2	P1	P0

- When a graphics pixel passes through the LUT, it uses the same format as a BPP15 pixel.
- The selection of the pixel to display is done via the keying mechanism (refer to the **XCOLKEYH** register description).
- The hardware cursor and horizontal zooming are available.
- Register set ‘C’ should be used to program the pixel PLL.

5.7.2 Palette RAM (LUT)

The MGA-1064SG's DAC uses three 256x8 dual-ported RAM chips for its color LUT. The use of a dual-ported RAM allows for asynchronous operation of the RAM, regardless of the current display state. The RAM is addressed by an 8-bit register/counter (**PALWTADD**) and selection among the three LUTs is done using a modulo 3 counter.

To write the red, green, and blue components of a pixel to a location in the RAM, three writes to the **PALDATA** DAC register must occur. Each byte will be transferred to the RAM when it is written. The modulo 3 counter will track the color being written. When the last byte (the blue component) of a RAM location is written, the address register is incremented, the modulo 3 counter is cleared, and the circuit is ready to write the red component of the next location. This allows the entire RAM to be updated with only one access to the **PALWTADD** register.

To read a complete location in the palette RAM, three reads of the **PALDATA** DAC register must occur. The palette address register will then be incremented to the next location. As with writes, the RAM can be completely read with only one write to the **PALRDADD**.

Note: When changing the **ramcs** bit of the **XMISCCTRL** DAC register, the pixel clock ***must*** be disabled (that is, **pixclkdis** = '1').

5.7.3 Hardware Cursor

A hardware cursor has been defined for all non-VGA modes. This cursor will be displayed over any other display information on the screen, either video or graphics.

The cursor position is relative to the end of the blanking period. Refer to the **CURPOSX** and **CURPOSY** register descriptions. The cursor is not zoomed when horizontal and/or vertical zooming is selected. The cursor pattern is stored in the off-screen memory of the frame buffer at the location defined by the **XCURADDH** and **XCURADDL** DAC registers.

The **CURPOSX** and **CURPOSY** registers are double-buffered (that is, they are updated at the end of the vertical retrace period). They ***must not*** be programmed when the **vsyncsts** field of the **STATUS** register is '1'. (They can be updated at any other time.) The **XCURADDH** and **XCURADDL** registers are ***not double buffered***, so changes to this register may produce unwanted artifacts on the screen.

In interlaced mode, if the cursor Y position is greater than 64, the first line of the cursor to appear on the screen will depend on the state of the internal field signal.

- If the value of **CURPOSY** is an odd number, the data for row 0 of the cursor will be displayed in the odd field. Rows 2, 4, ... 62 will then be displayed on the subsequent lines. The data for row 1 of the cursor will be displayed in the even field, followed by rows 3, 5, ... 63.
- If the value of **CURPOSY** is an even number, the data for row 0 of the cursor will be displayed in the even field. Rows 2, 4, ... 62 will then be displayed on the subsequent lines. The data for row 1 of the cursor will be displayed in the odd field, followed by rows 3, 5, ... 63.
- If the value of **CURPOSY** is less than 64, the cursor is partially located off the top of the screen. The first cursor row (row N) to be displayed will always be on scan line 0, which is the first line of the even field, and therefore the topmost scan line of the screen. Rows N+2, N+4, and so on will follow. The data in cursor row N+1 will be displayed on the first line of the odd field, followed by row N+3, N+5, and so on.

In order for the cursor to function properly, the following rules must be respected:

Hblank_width (ns) >= 6 * Tmclk + A * Tmclk, where:
A = 93 (the memory controller's cursor request latency), Tmclk = MCLK cycle time (ns)

5.7.4 Keying Functions

Keying can occur in the two split modes and in direct color mode. Refer to the **XCOLKEYH** DAC register description for more information. Color keying is performed only on graphics pixels. In 2G8V16 and BPP32DIR modes, only the LSBs of the **XCOLKEYx** and **XCOLKEYMSKx** registers are used because the graphics or overlay pixel is only 8 bits wide. In G16V16 mode, the entire **XCOLKEYx** and **KEYCOLMSKx** registers are used since the graphics pixel is 15 bits wide. Bit 15 of the video pixel can also be used in the keying equation when in split mode.

5.7.5 Zooming

Horizontal zooming is achieved by changing the **hzoom** field of the **XZOOMCTRL** register. The CRTC Memory Address Counter clock will automatically be changed accordingly. No other CRTC register need be changed. The supported zoom factors are x1 (no zoom), x2, and x4.

Vertical zooming is performed by the CRTC and nothing need be done in the DAC section of the MGA-1064SG.

5.7.6 Feature Connector

The MGA-1064SG supports two kinds of feature connectors: the standard 8-bit VGA output only connector, and a special Matrox 32-bit MAFC connector for a video encoder. The feature connector is 16 bits wide and also provides the VHSYNC/, VVSYNC/, VBLANK/, and VDCLK signals. All of the pins of the connector can be disabled (reset to ‘0’) except for the sync signals, which must always be active for the monitor.

The standard 8-bit connector is used only in VGA mode to output the data. The pixel to be output is taken from the input of the DAC section, so this data has not gone through the pixel pipeline or the color LUT. The data can be found on the eight lower bits of the connector.

The Matrox feature connector (MAFC) takes 24 bits of data at the input of the DAC and multiplexes them on both edges of the pixel clock. When in BPP32DIR mode, the alpha data is also available on the connector. In other modes, the alpha data will not be valid.

Feature Connector Pin	VDCLK ***High***	VDCLK ***Low***
D<7:0>	Blue	Red
D<15:8>	Green	Alpha

When using the MAFC connector, VDCLK must be set to input mode (**vedclk** = ‘0’).

5.7.7 Test Functions

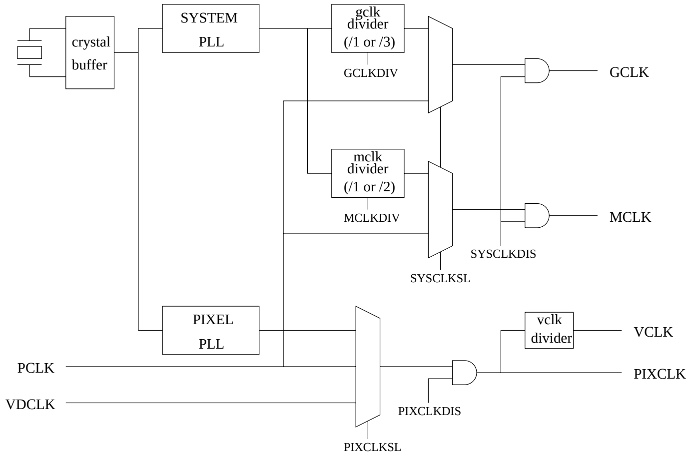
A 16-bit CRC is provided to verify video integrity at the input of the DAC. The CRC can be read via the **XCRCREMH** and **XCRCREML** registers when the vertical sync is active. The CRC is cleared at the end of the vertical retrace period, and calculated only when the video is active. The **crcsel** field determines which of the 24 bits will be used in the calculation.

The output of the sense comparator can be read via the **XSENSETEST** DAC register. This provides a means to check for the presence of the CRT monitor and determine if the termination is correct. The sense bits are latched at the start of the blanking interval. In order to ensure a stable value at the input of the comparator, the input of the DACs should remain constant during the visible display period. The sense amplifiers can be powered down by setting the **sensepdN** bit to ‘0’.

5.7.8 PLL Clock Generators

The MGA-1064SG's DAC has two independent programmable Phase Lock Loops (PLLs). The first one is the system PLL, which is used for the system clocks: the memory clock (MCLK) and the graphics engine clock (GCLK). The second one is the pixel PLL, which is responsible for generating the pixel clock that is used by the DAC (PIXCLK) and the video clock that is used by the CRTC (VCLK).

Figure 5-6: Clock Scheme



5.7.8.1 System PLL

The system PLL is programmed through the **XSYSPLLM**, **XSYSPLLN** and **XSYSPLLP** registers. The frequency of the Voltage Controlled Oscillator (VCO) is defined by:

Fvco = Fref * (**XSYSPLLN** + 1) / (**XSYSPLLM** + 1)

Where Fref = 14.31818 MHz.

100 <= N <= 127 (feedback divider)

1 <= M <= 31 (input divider)

P = {0,1,3,7} (post-divider)

0 <= S <= 3

The PLL output frequency is then:

Fo = Fvco / (syspllp + 1)

On reset, the system PLL is bypassed and the system clock is derived from the PCI bus clock. This permits the MGA-1064SG to boot-up properly. The system PLL resets to its oscillating frequency when the **syspllpdN** bit is set to '1'. The system PLL clock can then be divided down to provide the 66 MHz MCLK and 44 MHz GCLK when the **gclkdiv** and **mclkdiv** fields are set to '0'. When these two fields are set to '1', the memory and graphics clocks have the same frequency as their sources.

The memory clock (MCLK) can be selected to be the PCI bus clock (on boot-up), the MCLK pin, or the system PLL clock output. The graphics clock is the PCI bus clock or the system PLL. Refer to the **sysclksl** field of the **OPTION** register for more details. The graphics clock can also be gated off when **sysclkdis** is '1', when changing the characteristics of MCLK or GCLK (see Section 5.7.8.3 on page 5-77). To further lower power consumption, **syspllpdN** can be reset to '0' to shut off the PLL. The contents of the memory will be lost.

5.7.8.2 Pixel PLL

The pixel PLL contains three independent sets of registers: sets A, B, and C. The **clksel** field of the VGA MISC register will determine which set will define the operating frequency of the pixel PLL (see the pixpllan register description). The frequency of the Voltage Controlled Oscillator (VCO) is defines by the following formula:

Fvco = Fref * (**XPIXPLLN** + 1) / (**XPIXPLLM** + 1)

Where Fref = 14.31818 MHz.

100 <= N <= 127 (feedback divider)

1 <= M <= 31 (input divider)

P = {0,1,3,7} (post-divider)

0 <= S <= 3

The PLL output frequency is then:

Fo = Fvco / (**XPIXPLLP** + 1)

On reset, the pixel clock (PIXCLK) is generated from the PCI bus clock. The pixel PLL will run with the register set that is selected by the **clksel** field when the **pixpllpdN** field is set to ‘1’. After a reset, **clksel** is ‘00’, so the pixel PLL will oscillate at 25.175 MHz and VCLK will be the same frequency (since the DAC wakes up in VGA mode).

The video clock (VCLK) is function of the display mode of the DAC:

mgamode	depth	*Video Clock*
0	xxx	PIXCLK
1	000	PIXCLK/8
1	001	PIXCLK/4
1	010	PIXCLK/4
1	011	PIXCLK*3/8
1	100	PIXCLK/2
1	101	PIXCLK/4
1	110	PIXCLK/2
1	111	PIXCLK/2

The maximum supported pixel clock frequency is 135 MHz (1280 x 1024 resolution at a 75 Hz refresh rate). The minimum period of the VCLK signal is 14.8 ns (1280 x 1024, 24-bit packed pixel at a 75 Hz vertical refresh rate).

The pixel clock can obtain its source from three different places: the Pixel PLL (normal operation), the PCI bus clock (at boot-up), or the VDCLK pin (when slaving the MGA-1064SG to an external video source). The selection is done via the **pixclksl** field of the **XPIXCLKCTRL** DAC register. PIXCLK and VCLK can also be shut off by setting the **pixclkdis** bit to ‘1’. Again, as for the system PLL, the pixel PLL can be powered down by resetting the **pixpllpdN** bit to ‘0’ to lower power consumption.

5.7.8.3 Programming the PLLs

To change the frequency of one of the PLLs or the source of a clock, the following procedure ***must*** be followed:

(A) Changing the Pixel Clock Frequency or Source

To program any of the **XPIXPLLM**, **XPIXPLLN**, **XPIXPLLP**, or **XPIXCLKCTRL** registers, the memory clock ***must*** be running and enabled (**sysclkdis** = ‘0’).

1. Force the screen off.
2. Set **pixclkdis** to ‘1’ (disable the pixel and video clocks).
3. Re-program the desired pixel PLL registers by changing the values of the registers, by changing the **clksel** field of the VGA **MISC** register, or by selecting another source for the pixel clock.
4. Wait until the clock source is locked onto its new frequency (the **pixlock** bit is ‘1’) for the pixel PLL, or for the VDCLK pin to become stable.
5. Set **pixclkdis** to ‘0’ (enable the pixel and video clocks).
6. Resume normal operations (re-enable the screen display).

No special procedures need to be followed when changing the frequency of the video clock since the MGA-1064SG’s hardware will not generate glitches on the video clock when the **mgamode** or **depth** fields are changed.

(B) Changing the System PLL Frequency

Special care must be taken when changing the frequency of the system PLL. Since the **XSYSPLLM**, **XSYSPLLN**, and **XSYSPLLP** registers are clocked on the memory clock, the system PLL must always be running.

1. Set **sysclkdis** to ‘1’ (disable the system clocks).
2. Select the PCI bus clock for the system clocks (**sysclksl** = ‘00’).
3. Set **sysclkdis** to ‘0’ (enable the system clocks).
4. Re-program the desired system PLL registers.
5. Wait until the **syslock** bit is ‘1’.
6. Set **sysclkdis** to ‘1’ (disable the system clocks).
7. Select the system PLL clock for the system clocks (**sysclksl** = ‘01’).
8. Set **sysclkdis** TO ‘0’ (enable the system clocks).
9. Resume normal operations.

(C) Changing the System Clock Source, MCLK, or GCLK Division Factor

1. Set **sysclkdis** to ‘1’ (disable the system clocks).
2. Select the new clock source or change the **mclkdiv** and/or **gclkdiv** fields. Make sure that the new clock source is stable before continuing.
3. Set **sysclkdis** to ‘0’ (enable the system clocks).
4. Resume normal operations.

DAC external components:

The magnitude of the full scale current can be controlled by a resistor using the following calculation:

R (ohm) = K * 1000 * REF(V) / Iout (mA)

Pedestal	*K factor* *With sync*	*No sync*
7.5 IRE	3.927	2.805
0.0 IRE	3.716	2.593

This resistor should be placed between the RSET pin and the analog GND.

A 0.1 uF capacitor should be placed between the COMP pin and the analog VDD.

The voltage applied to the Vref pins is 1.235 V.

5.8 Interrupt Programming

The MGA-1064SG 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.

Interrupt	*STATUS*	*EVENT*	*ENABLE*	*CLEAR*
Pick	- -	**pickpen** STATUS<2>	**pickien** IEN<2>	**pickiclr** ICLEAR<2>
Vertical sync	**vsyncsts** STATUS<3>	**vsyncpen** STATUS<4>	**vinten** CRTC11<5>	**vintclr** CRTC11<4>
Vertical line	- -	**vlinepen** STATUS<5>	**vlineien** IEN<5>	**vlineiclr** ICLEAR<5>
External	**extpen** STATUS<6>	-	**extien** IEN<6>	-

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	Each 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

5.9 Power Saving Features

The MGA-1064SG supports two power conservation features:

- DPMS is supported directly, through the following control bits:
  - Video can be disabled using **scroff** blanking bit (**SEQ1**<5>)
  - Vertical sync can be forced inactive using **vsyncoff** (**CRTCEXT1**)
  - Horizontal sync can be forced inactive using **hsyncoff** (**CRTCEXT1**)
- The power consumption of the chip can be reduced by slowing down the system clocks and stopping the video clocks.
  - Program the memory clock to be 6.66 MHz (10% of the normal operating frequency): The system PLL should oscillate at 6.66 MHz and the **mclkdiv** field should be set to '1'.
  - Program the **gclkdiv** field to '0'. The graphics clock is now running at 2.22 MHz.
  - If the contents of the frame buffer must be preserved, MCLK must be running and the **rfhcnt** field of the **OPTION** register must be re-programmed according to the new MCLK frequency.
  - The video section can be powered via the following steps:

    - Disable the cursor (set the **curmode** field to `00').

    - Set the **pixclkdis** field of **XPIXCLKCTRL** to `1'.

    - Power down the DAC.

    - Power down the LUT.

    - Power down the pixel PLL.

Chapter 6: Hardware Designer's Notes

This chapter includes:

6.1 Introduction

The MGA-1064SG chip has been designed in such a way as to minimize the amount of external logic required to implement a board. Included among its features are:

- Direct interface to the PCI bus
- All necessary support for external devices such as ROM, video co-processor, and others
- Direct connection to the RAM
- Support for a video co-processor which can share the frame buffer
- Direct interface to the video and feature connectors

6.2 PCI Interface

The MGA-1064SG interfaces directly with PCI as shown in Figure 6-1. The MGA-1064SG is a medium-speed (target) device which will respond with PDEVSEL/ during the second clock after PFRAME/ is asserted.

In order to optimize performance on the PCI bus, burst mode, disconnect, and retry are used as much as possible rather than the insertion of wait states. Only a linearly-incrementing burst mode is supported. Because a 5-bit counter is used, a disconnect will be generated every 32 aligned dwords. Refer to Sections 5.1.2 and 5.1.3 for more information.

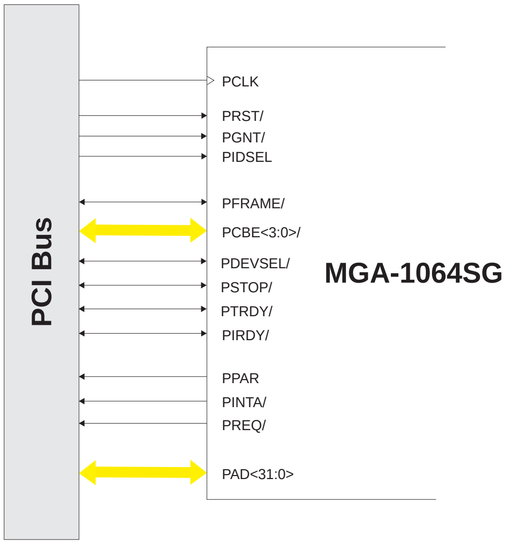
6.2.1 Snooping

The MGA-1064SG performs snooping when VGA I/O is enabled and snooping is turned on. In this specific case, two things may occur when the DAC is written to:

1. If the MGA-1064SG is unable to process the access immediately, it takes control of the bus, and a retry cycle is performed.
2. If the MGA-1064SG is able to process the access, the access is snooped, and the MGA-1064SG processes it as soon as the transaction is completed on the PCI bus.

Under normal conditions, only a subtractive agent will respond to the access. There could also be no agent at all (all devices are set to snoop, so a master-abort occurs). In these cases, the snoop mechanism will function correctly. If there is another device on the PCI bus that responds to this mapping, or if another device performs the snoop mechanism with retry capabilities, the result will be contention on the PCI bus.

Figure 6-1: PCI Interface



6.3 External Devices

The MGA-1064SG supports a few external devices (the EPROM is a standard expansion device that is supported by the MGA-1064SG). Other devices can also be added by using the MGA-1064SG’s EXTCS/ strobe.

Figure 6-2 shows how to connect the standard expansion devices to the MGA-1064SG. It should be noted that the local bus interface shares pins with the RAM. This limits the load on the MDQ bus to 10 pF (1 load) per bit, which is automatically the case when there are no extra external devices.

EPROM

The MGA-1064SG supports both 256K x 8 and 512K x 8 EPROMs, as well as flash memory. Flash memory provides the capability to modify the BIOS ‘on the fly’. The following table lists specific EPROM and flash memory devices that have been verified to work with the MGA-1064SG:

	Flash Memory		***EPROM***	
Manufacturer	***256K x 8***	***512K x 8***	***256K x 8***	***512K x 8***
AMD	AM28F256-150	AM28F512-150	AM27256-200	AM27C512-200
Atmel	AT29C257-12 AT29C257-15 AT29C257-90	AT29C512-90 AT29C512-120 AT29C512-150		
SGS	M28F256-15			
Intel	N28F256A-150	N28F512-150		
Toshiba			TC57256AD-20	TMM27512AD-20
Texas Instruments		TMS28F512A-10 TMS28F512A-12 TMS28F512A-15	TMS27C256-2	
National			NM27C256Q200	NMC27C512AQ200
Microchip			27C256-20	27C512-20

A write cycle to the EPROM has been defined in order to support flash memory. Another bit which locks write accesses to the EPROM has also been added in order to prevent unexpected writes.

Note, however, that sequencing of operations to erase and write the memory must be performed by software. Additionally, some timing parameters (tWR, tWH1, tWH2) must be guaranteed by software using programming loops (refer to the device specification).

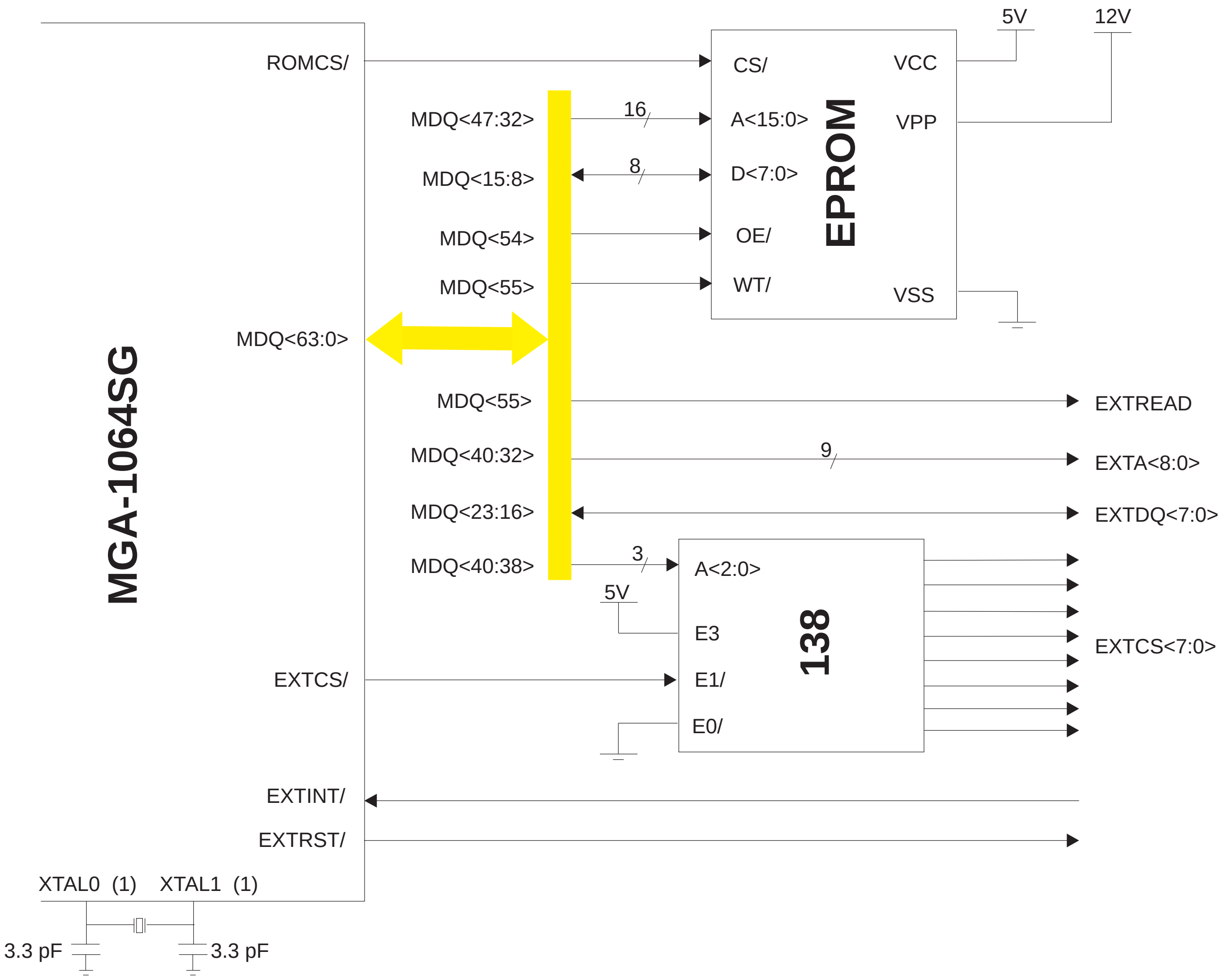
Finally, if a 12 V power supply is required for flash memory, it will have to be provided on the board, and the MGA-1064SG will have no ability to control it.

Other Devices

Extra devices can be added to the MGA-1064SG (in addition to the standard expansion devices mentioned above). If a video co-processor or any other extra device is required, a decoder (as shown in Figure 6-2) can be used to generate multiple CS/ signals. However, in order to respect load constraints on the MDQ bus, the following rules must be respected:

- Read strobes and addresses that are used for both the EPROM and the external devices (including the decoder) must be buffered.
- If multiple devices are added, the data bus to those external devices must be buffered.

Figure 6-2: External Device Connections



(1) If a local oscillator is used instead of a crystal, it is connected to XTAL0, and XTAL1 is left unconnected.

6.4 Memory Interface

6.4.1 SDRAM/SGRAM Configurations

The principal characteristics of the MGA-1064SG's SDRAM/SGRAM interface are provided in Table 6-1, which identifies the cycles that are supported by the chip, and lists all of the commands generated by the MGA-1064SG.

Table 6-1: Supported SGRAM/SDRAM Commands

Command (1)	*Mnemonic*	*MCS/*	*MRAS/*	*MCAS/*	*MWE/*	*MDSF*	*MDQMi*	*BS*	*AP*	*Address*
Mode Register Set	MRS	L	L	L	L	L	X	L	L	opcode1 (2)
Special Mode Register Set	SMRS	L	L	L	L	H	X	L	L	opcode2 (3) (4)
Auto Refresh	REF	L	L	L	H	L	X	X	X	X
Bank Activate / Row Address (Mask Disabled)	ACTV	L	L	H	H	L	X	V	Row Address (5)	
Bank Activate / Row Address (Mask Enabled)	ACTM	L	L	H	H	H	X	V	Row Address (4) (5)	
Read / Column Address (Auto-Precharge Disabled)	READ	L	H	L	H	L	X	V	L	Column Address (6)
Write / Column Address (Auto-Precharge Disabled)	WRITE	L	H	L	L	L	X	V	L	Column Address(6)
Block Write / Column Address (Auto-precharge Disabled)	BWRIT	L	H	L	L	H	X	V	L	Column Address (4) (6) (7)
Precharge (***Single*** Bank)	PRE	L	L	H	L	L	X	V	L	X
Precharge (***Both*** Banks)	PALL	L	L	H	L	L	X	X	H	X
No Operation	NOP	L	H	H	H	L	X	X	X	X
Device De-select	DESL	H	X	X	X	X	X	X	X	X
Mask Write Data / Disable Read Output	X	X	X	X	X	X	H	X	X	X (8)
Write Data / Enable Read Output	X	X	X	X	X	X	L	X	X	X (8)

✎ Legend: H = Logical High, L = Logical Low, V = Valid, X = "Don't Care", '/' indicates an active low signal.

(1) For 8 MBytes SGRAM / 4 MBytes SDRAM:
MCS = MA<11:10> MCS<1:0>, BS=MA<9>, AP=MA<8>, Address = MA<7:0>.
For 16 MBytes SDRAM:
MCS = MCS<0>, BS = MA<11>, AP = MA<10>, Address = MA<9:0>.

(2) The MGA-1064SG supports CAS Latency (CL=2, CL= 3); burst type = sequential; burst length = 1.
For 8 MBytes SGRAM / 4 MBytes SDRAM: opcode1 = '001':CLbit:'0000'.
For 16 MBytes SDRAM: opcode1 = '00001': CLbit = '0000' where CLbit = 0 for CL = 2 and CLbit = 1 for CL = 3.

(3) A5 = 1 for a mask register access, and A6 = 1 for a color register access. Both registers ***cannot*** be accessed simultaneously.
opcode2 = '0000100000' <- load mask register
opcode2 = '0001000000' <- load color register

(4) Valid only for SGRAM.

(5) For 8 MBytes SGRAM / 4 MBytes SDRAM: Row Address = MA<8:0>.
For 16 MBytes SDRAM: Row Address = MA<10:0>.

(6) The MGA-1064SG does not support the auto-precharge function, so AP will always be forced low for READ/WRITE/BWRIT commands.
For 8 MBytes SGRAM / 4 MBytes SDRAM: Column Address = MA<7:0>.
For 16 MBytes SDRAM: Column Address = MA<9:0>.

(7) MA<2:0> are ‘don’t care’ for block write commands.

(8) Not a command - ‘MDQ mask enable’.

✍ Note that the MGA-1064SG does ***not*** drive CKE: it should be driven high externally.

✍ Both shared and split address generation are supported (selected by the **splitmode** field of **OPTION**). The number of address bits also depends on the memory type (selected by the **memconfig** field of **OPTION**). The addresses are mapped as follows:

Table 6-2: 12-bit Address Configuration (**memconfig** ***= 1)***

splitmode		***MA***											
		11(BS)	***10(AP)***	***9***	***8***	***7***	***6***	***5***	***4***	***3***	***2***	***1***	***0***
0	Row	A11	A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	A12
	Column	A11	‘0’	‘1’	‘1’	A10	A9	A8	A7	A6	A5	A4	A3
1	Row	A10	A21	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11
	Column	A10	‘0’	‘1’	‘1’	A9	A8	A7	A6	A22	A5	A4	A3

Table 6-3: 10-bit Address Configuration (**memconfig** ***= 0)***

splitmode		***MA***									
		9(BS)	***8(AP)***	***7***	***6***	***5***	***4***	***3***	***2***	***1***	***0***
0	Row	A11	A20	A19	A18	A17	A16	A15	A14	A13	A12
	Column	A11	‘0’	A10	A9	A8	A7	A6	A5	A4	A3
1	Row	A10	A19	A18	A17	A16	A15	A14	A13	A12	A11
	Column	A10	‘0’	A9	A8	A7	A6	A22	A5	A4	A3

Figure 6-3: SGRAM Connection

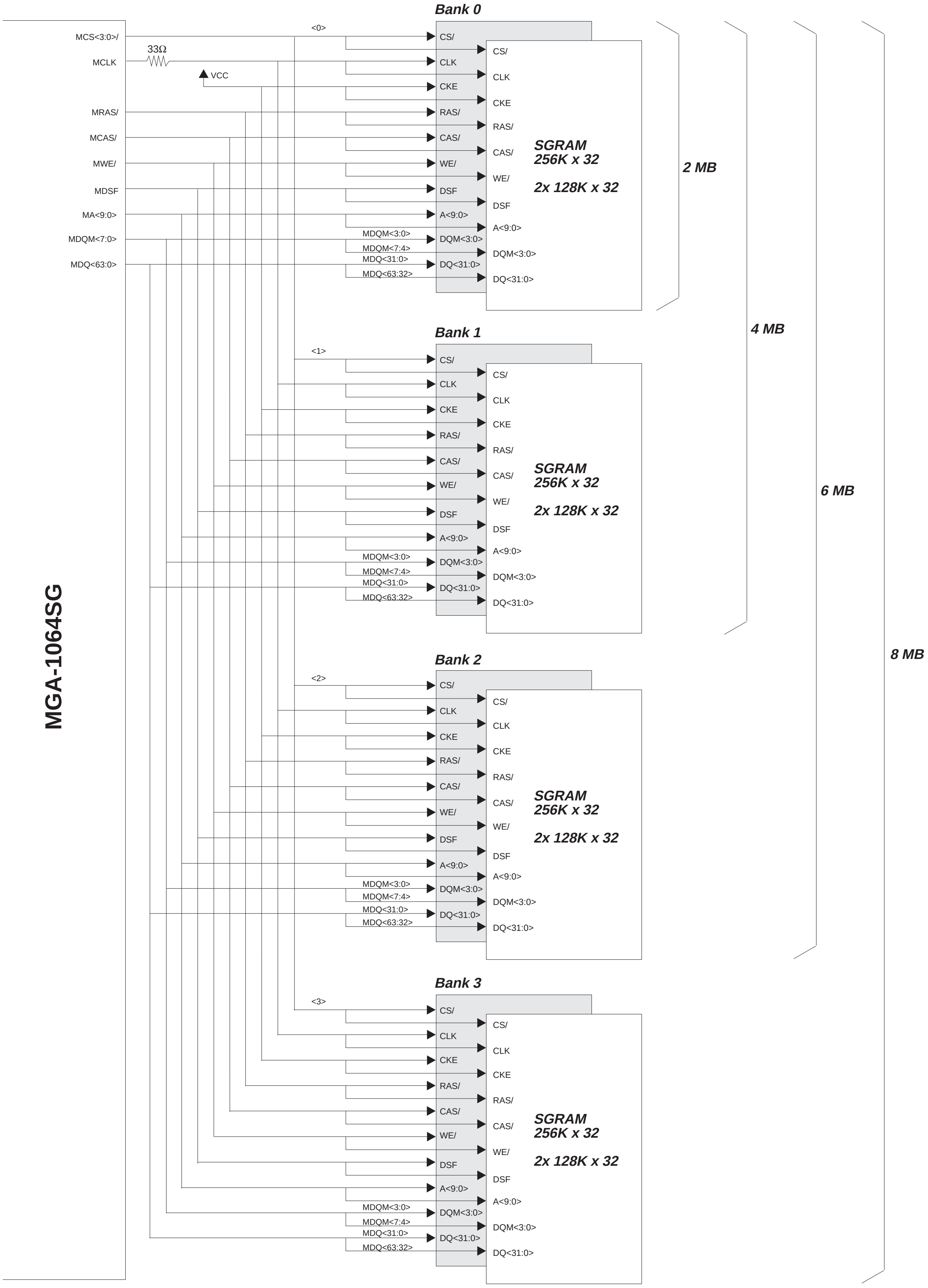


Figure 6-4: SDRAM Connection (256Kx16)

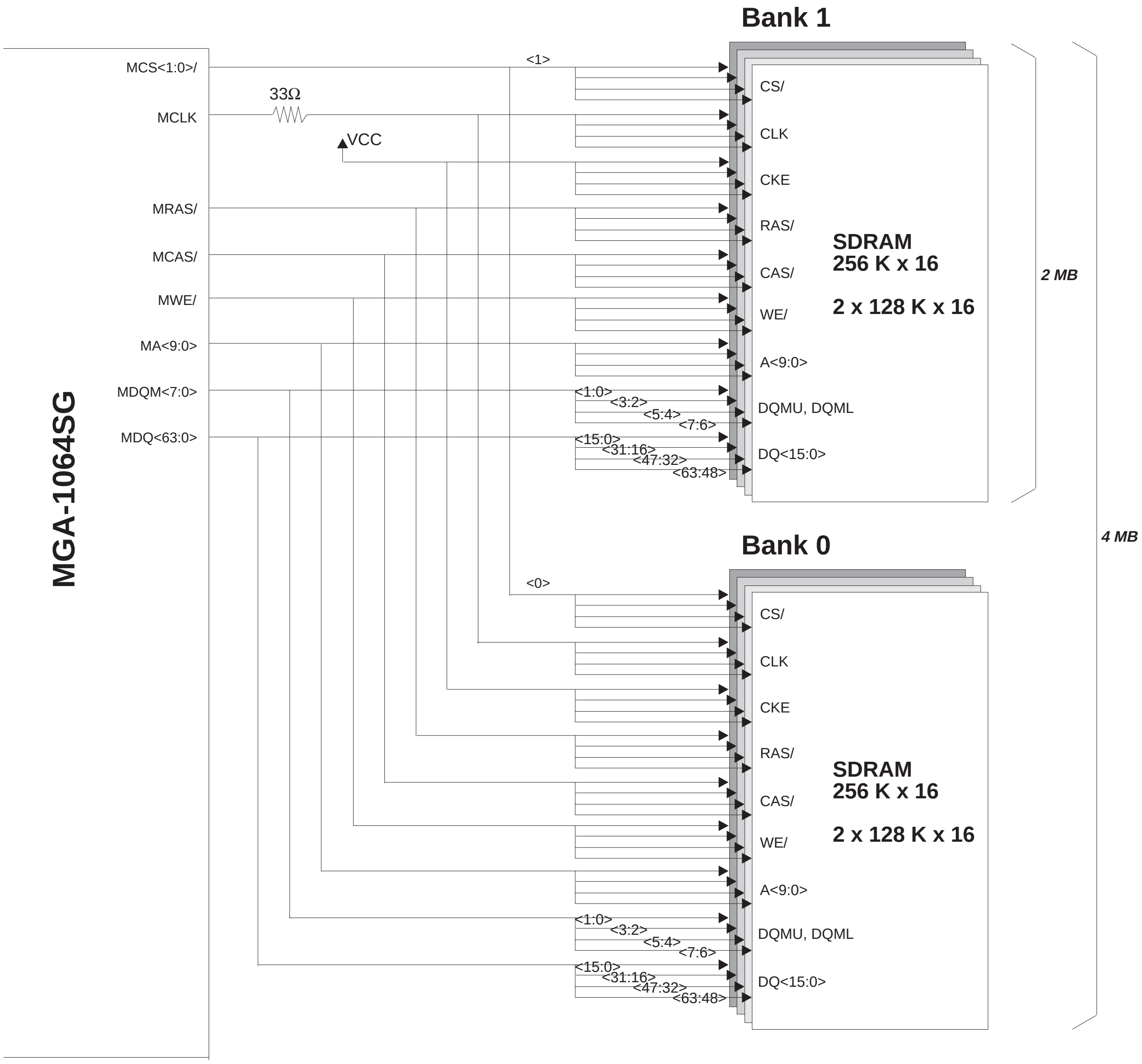
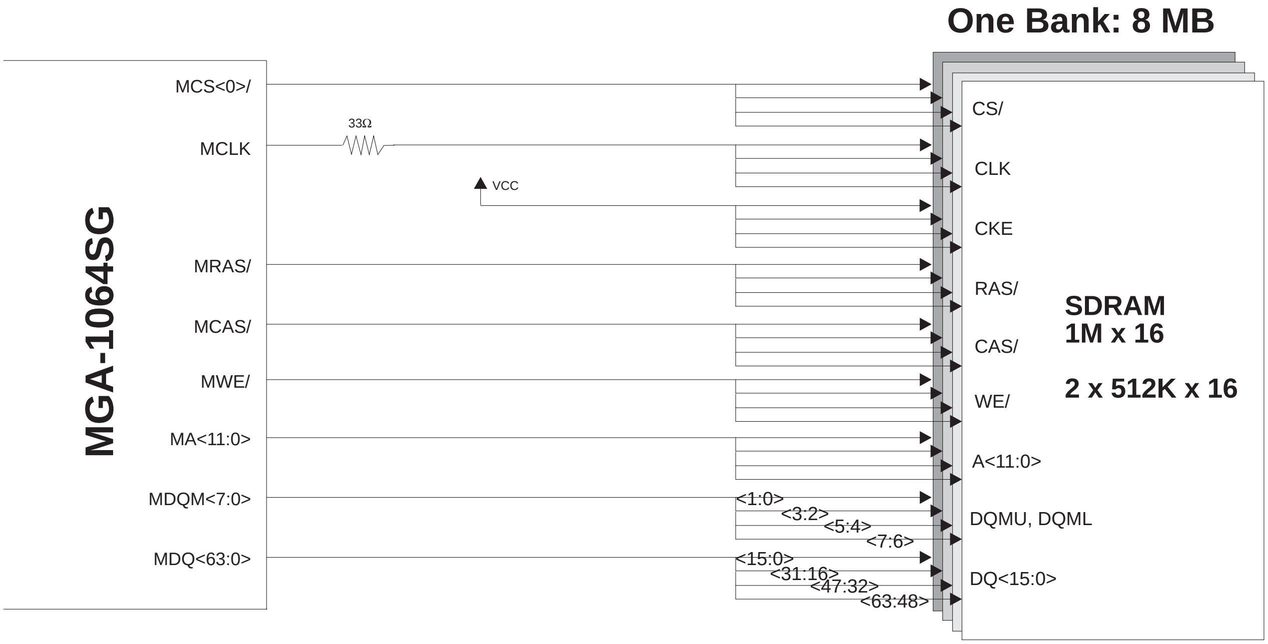
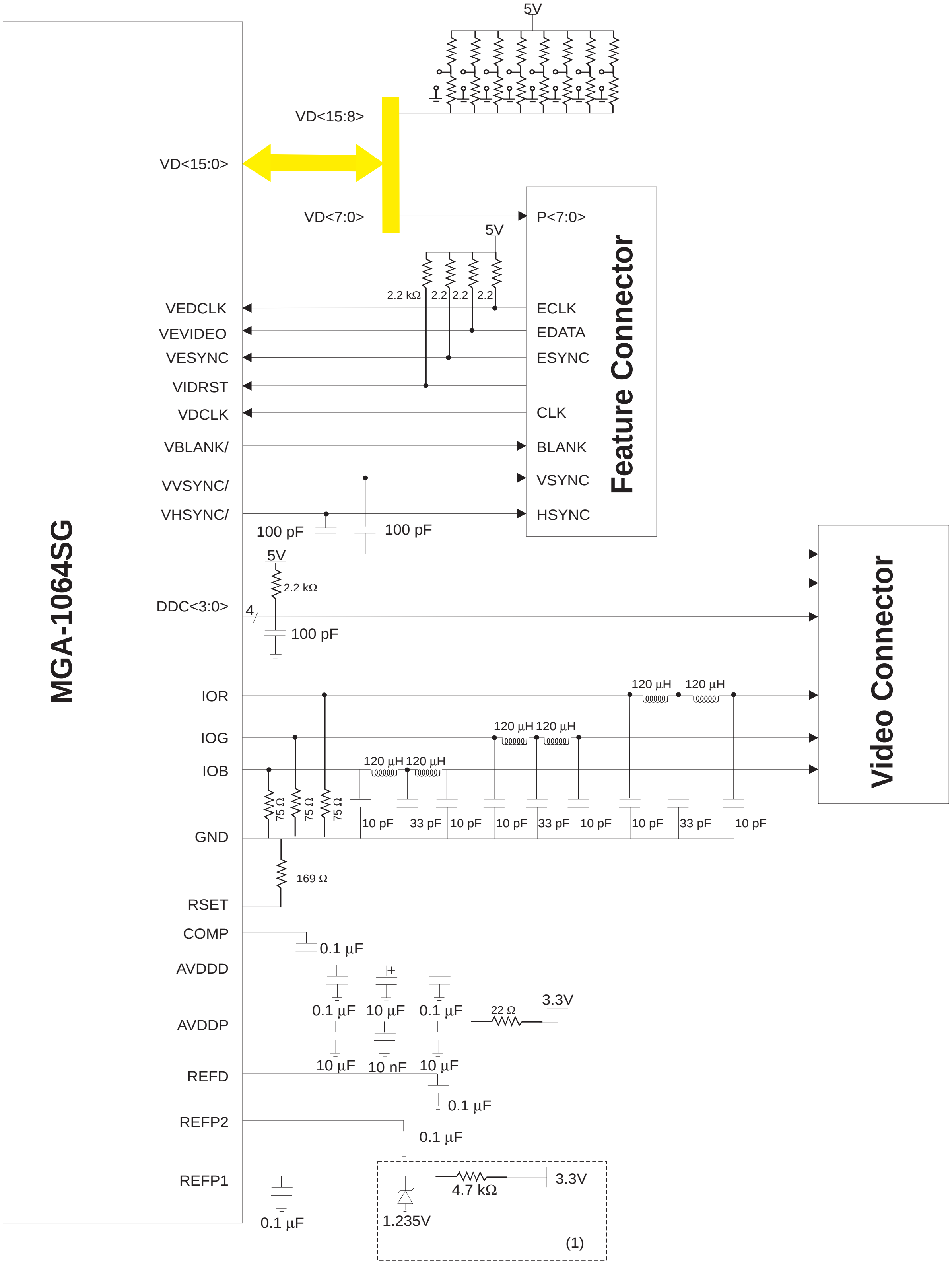


Figure 6-5: SDRAM Connection (1Mx16)



6.5 Video interface

Figure 6-6: Feature Connector, Video Connector



(1) Remove this circuitry when the internal reference is used.

6.5.1 Slaving the MGA-1064SG

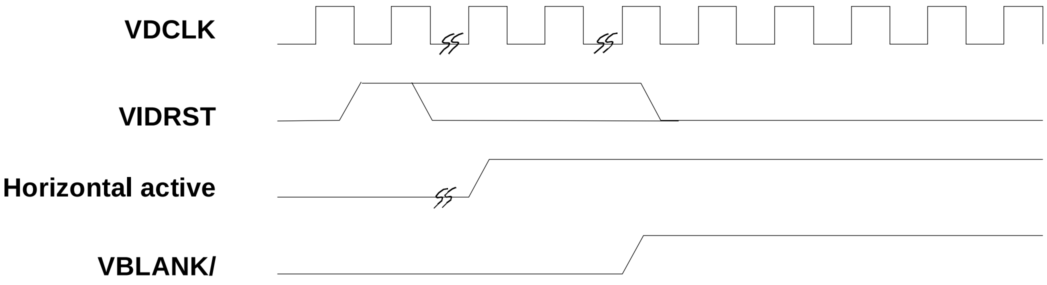
This section describes the operations of the VIDRST (video reset input) signal. A VIDRST is detected on the first rising edge of VDCLK where VIDRST is high. The video reset can affect both the horizontal and/or vertical circuitry.

The first time that the MGA-1064SG's CRTC is synchronized, the data may be corrupted for up to one complete frame. However, when the CRTC is already synchronous and a reset occurs, the CRTC will behave as if there was no VIDRST.

> ✎ Note: In order for the MGA-1064SG to be synchronous with any other source, the MGA-1064SG CRTC must be programmed with the same video parameters as that other source. VDCLK can also be modulated in order to align both CRTCs.

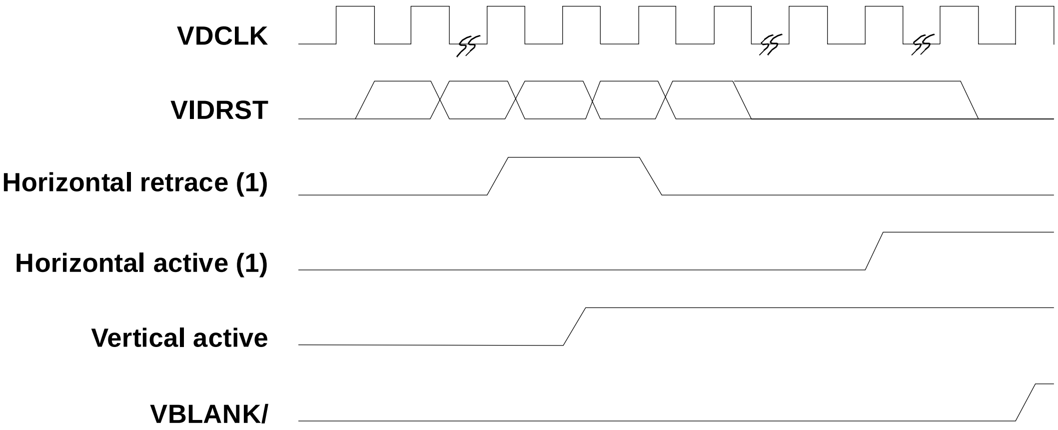
The **hrsten** field of the **CRTCEXT1** register is used to enable the horizontal reset, which sets the horizontal and character counters to the beginning of the horizontal active video. Figure 6-7 shows the relationship between VIDRST, the internal horizontal active, and VBLANK/ when the MGA-1064SG is already synchronized.

Figure 6-7: VIDRST, Internal Horizontal Active, VBLANK/



The **vrsten** field of the **CRTCEXT1** register is used to enable the vertical reset, which sets the vertical counter to the beginning of the vertical active video in the even field. Figure 6-8 shows the relationship between VIDRST, the internal horizontal retrace, the internal vertical active signal, and VBLANK/ when only the vertical counter is reset.

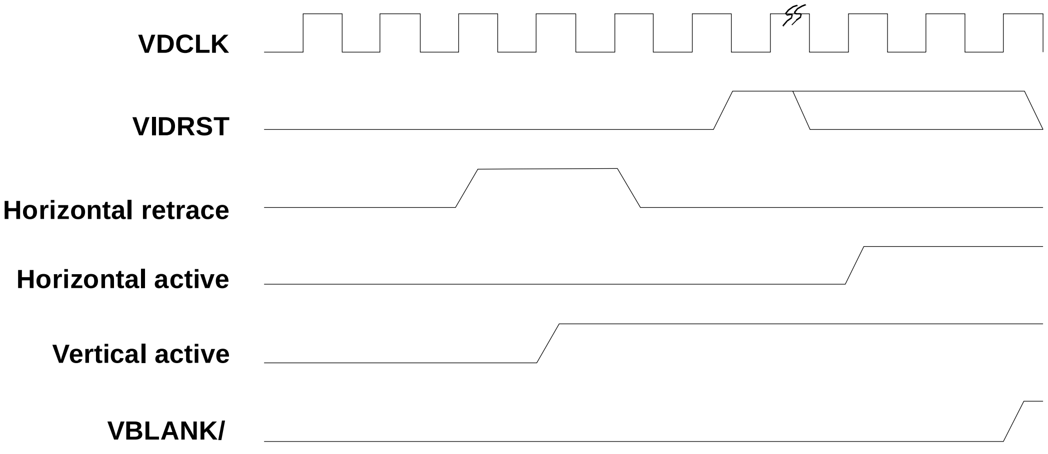
Figure 6-8: VIDRST, Internal Horizontal Retrace/Vertical Active, VBLANK/



(1) Horizontal active and horizontal retrace are not affected by VIDRST when only the vertical reset is active. They are shown in the waveform as a reference to the location where VIDRST can be active in steady state.

Figure 6-9 shows the relationship between VIDRST, the internal horizontal retrace, the internal horizontal and vertical active signals, and VBLANK/ when both the horizontal and vertical counters are reset.

Figure 6-9: VIDRST, Internal Horizontal/Vertical Active, and VBLANK/



6.5.2 Genlock Mode

The VIDRST pin can be used to reset the CRTC horizontal and/or vertical counters. VIDRST must be maintained high for at least 1 VDCLK cycle for the reset to take effect in the MGA-1064SG. When it is not used, the VIDRST pin ***must*** be maintained low (there is no enable/disable control bit for the VIDRST pin).

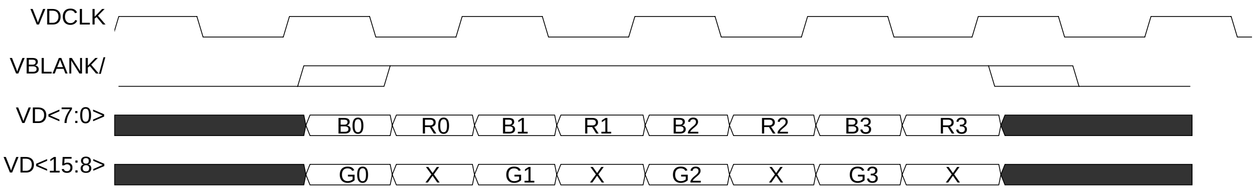
If the timing on the VIDRST pin is respected, the reset operation on the chip will be completed (the VBLANK/ pin is set to ‘1’), according to the number of VDCLKs shown in the following table:

Pixel Width	*Delay to Video Pin (VDCLKs)*
BPP8	28
BPP15	18
BPP16	18
BPP24	14
BPP32	13

Genlocking is not supported in VGA mode.

6.5.3 MAFC Data Sequence

Figure 6-10: MAFC Waveform



6.6 Co-processor Interface

Two pins permit sharing of the RAM bus:

- MVREQ/ (generated by the co-processor)
- MVGNT/ (generated by the MGA-1064SG)

When it releases the bus to the co-processor, the MGA-1064SG chip brings all memory chip select signals high before placing them in tri-state. The co-processor should do the same when releasing the bus. This procedure will guarantee that no false access will be performed on the memory. Pull-up resistors must be connected to the memory chip select signals when a co-processor is used.

Figure 6-11 and Figure 6-12 show the normal sequence when the co-processor requests and releases the bus.

The priority of operations in the MGA-1064SG is organized in such a way that the MGA-1064SG will notify the co-processor when it requires the bus in order to perform screen refresh or memory refresh cycles. When this is the case, the co-processor must return the bus to the MGA-1064SG as shown in Figure 6-12.

The MGA-1064SG's priorities are as follows:

1. Screen refresh
2. Second memory refresh request
3. Co-processor requests
4. Direct frame buffer or external device access
5. Drawing engine
6. First memory refresh request

The co-processor is not permitted to change the plane write mask that is stored in the SGRAM chips. The MGA-1064SG does not re-program the mask when it recovers the memory bus.

Figure 6-11: MGA-1064SG to Co-Processor Bus Transfer

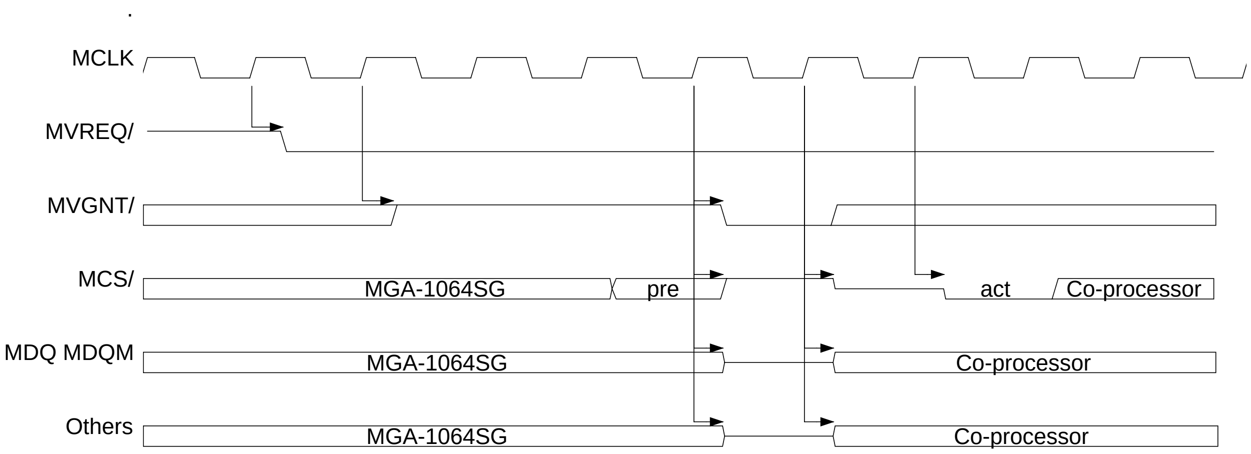
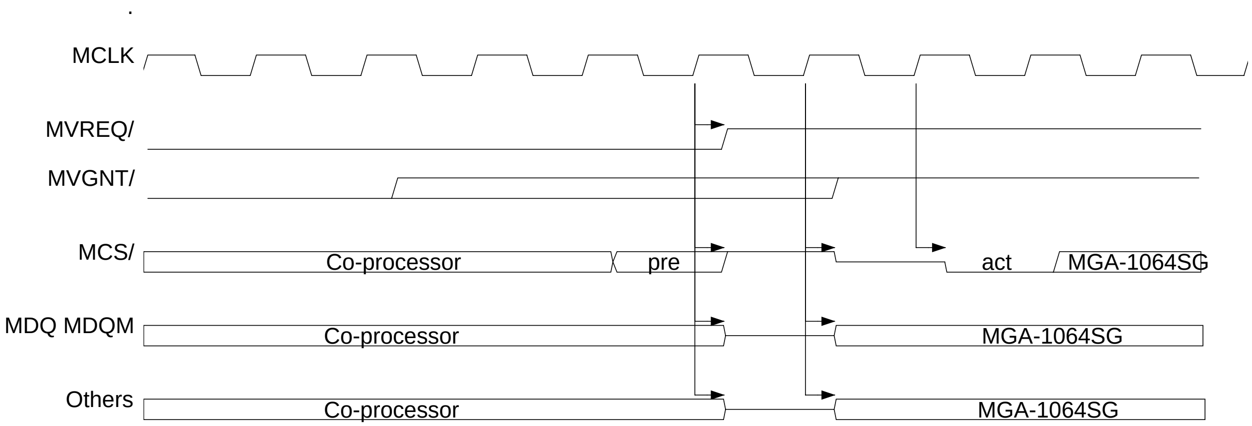


Figure 6-12: Co-Processor to MGA-1064SG Bus Transfer



6.7 Crystal Resonator Specification

Frequency	14.31818 MHz (+/- 50 ppm)
Equivalent series resistance (Rs)	35 - 200 Ω
Load capacitance (Cl)	18 or 20 pF (series ***or*** parallel)
Shunt capacitance (Co)	7 pF max.
Drive level	100 - 1,000 µW
Temperature stability	50 ppm

Alphabetical List of Register Fields

Power Graphic Mode Register Fields (includes configuration space and memory space register fields)

Alphabetical List of Register Fields

Alphabetical List of Register Fields

VGA Mode Register Fields

Alphabetical List of Register Fields

Alphabetical List of Register Fields

RAMDAC Register Fields

Notes

Notes