Matrox Graphics Inc.

Matrox MGA-G100 Specification

Document Number 10534-MS-0110

February 4, 1998

Trademark Acknowledgements

MGA,™ MGA-1064SG,™ MGA-1164SG,™ MGA-G100,™ 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.

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

Contents

Chapter 6: Hardware Designer's Notes

Appendix A: Technical Information

Appendix B: Changes

List of Figures

Chapter 1: MGA Overview

Chapter 2: MGA-G100 Overview

Chapter 3: Resource Mapping

Chapter 4: Register Descriptions

Chapter 5: Programmer's Specification

Chapter 6: Hardware Designer's Notes

Appendix A: Technical Information

List of Figures

Appendix B: Changes

List of Tables

Chapter 1: MGA Overview

Chapter 2: MGA-G100 Overview

Chapter 3: Resource Mapping

Chapter 4: Register Descriptions

Chapter 5: Programmer's Specification

Chapter 6: Hardware Designer's Notes

Appendix A: Technical Information

List of Tables

Appendix B: Changes

Chapter 1: MGA Overview

This chapter includes:

1.1 Introduction

The Matrox MGA-G100 is a next generation 3D graphics, multimedia and Windows accelerator. In one low-cost package, the MGA-G100:

- 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
- Includes digital video input port
- Includes hardware CODEC interface port
- Connects to fast SSTL SGRAM or LVTTL SGRAM

The Matrox MGA-G100 has special features specifically designed to provide superior 3D game performance in a 2 MByte frame buffer. Matrox MGA-G100 is intended to provide a complete solution for home PC users who are interested in top performance Windows 95 and DOS 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 environments such as Windows NT, IBM OS/2 PM, Unix X-Windows, and AutoCAD.

The MGA-G100 series has an improved 3D acceleration core over the Matrox MGA-1064SG, key video capabilities of the MGA-VC064FB video engine and a significantly faster frame buffer interface for applications requiring a high display bandwidth. It controls up to 16 megabytes of SSTL SGRAM.

The integrated DAC in MGA-G100 eliminates the need for an external DAC. This substantially lowers the cost and space required for the graphics sub-system.

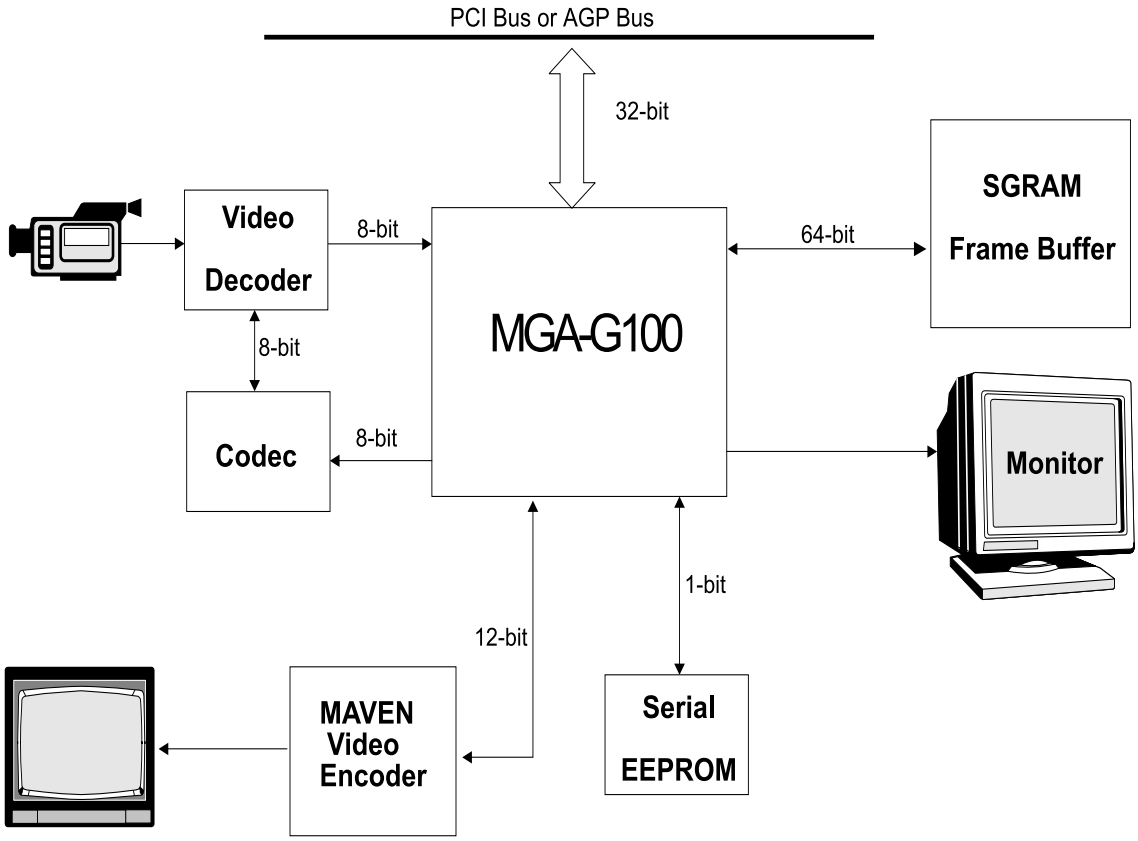
A full-featured 3D rendering engine, the Matrox Fast Texture Architecture, is the centerpiece of the MGA-G100. This 3D engine is an advanced renderer with full perspective correct texture mapping, lighting, Gouraud shading, fogging, stipple alpha blending, optional 16-bit or 32 bit buffering, Z buffering, capable of bus mastering and o keying on texture color or texel alpha key bit. 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-G100 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-G100 can be upgraded with the Matrox Video Encoder (MAVEN) which provides high quality output to a TV or VCR. MAVEN includes interpolated downscaling to the safe title area and 3 tap filtering to produce a superior TV picture.

This specification covers two devices : the MGA-G100-PCI that connects to the PCI bus and the MGA-G100-AGP that connects to the AGP bus. This specification applies the term MGA-G100 to both chips. For PCI specific information, the term MGA-G100-PCI will apply, while the term MGA-G100-AGP will apply to AGP specific information.

1.2 System Block Diagram

Figure 1-1: System Block Diagram



1.3 Application Areas

- Windows accelerator with high performance levels, (ideal for mid-range system requirements). The MGA-G100 will complement the MGA-1 family by delivering a strong price/performance point for users who need top performance at high resolution and color depths.
- 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-G100 3D engine. In addition, all other types of games will be accelerated by a combination of the MGA-G100'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-G100's ultra-fast 32-bit VGA core.
- Video capture.
- DVD playback
- Video editing.
- Video out to a TV with MAVEN.

1.4 Typical Implementation

MGA-G100 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
- Professional multimedia PC markets
- Desktop publishing

1.5 Features

1.5.1 Core GUI Accelerator

- Based on the current award-winning MGA-1064SG 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
- 71 MHz drawing engine
- 143 MHz operation for the SSTL SGRAM memory interface

1.5.2 3D Texture Mapping Engine

- Perspective Correct Texture Mapping
- True color lighting of textures
- Hardware dithering of LUT textures
- 16-bit or 32 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 (bits/pixel), 8bpp.
    - 3:3:2, 5:6:5, 1:5:5:5, 4:4:4:4.
- 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
- Fogging
- Stipple Alpha Blending

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 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 16 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
  - Bilinear interpolation as well as nearest neighbor
  - 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
- Fogging and depth cueing
- Stipple Alpha Blending

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

1.5.6 Integrated DAC

- 230 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 128Kword x 32bit x 2bank and 256Kword x 32bit x 2bank. Supports block write and write per bit for added performance.
- Supports from 2 to 16 MBytes of memory
  - Up to 4 banks of 128K x 32 x 2 SGRAM
  - Up to 4 banks of 256K x 32 x 2 SGRAM

1.5.8 Miscellaneous

- Feature connector interface
  - 8-bit VGA mode
  - 12 bit output mode for the MAVEN video encoder (24 bpp multiplexed to 12 bit bus).
- Host interface
  - PCI 2.1 compliant (MGA-G100-PCI) or AGP 1.0 including side band addresses MGA-G100-AGP)
  - 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.
  - PC 97 1.0 compliant

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 5-28.	**trans** # # # #

- 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-G100 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-G100 registers is found in Chapter 5, while information relating to hardware design is located in Chapter 6. Appendix A contains pinout, timing, and other miscellaneous information.

Appendix B lists the significant changes since the last revision of this document.

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.

This page left blank intentionally.

Chapter 2: MGA-G100 Overview

This chapter includes:

2.1 Introduction

The MGA-G100 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-G100 chip are listed below and described in detail in the remainder of this chapter.

- HOST 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 HOST Bus interface

This section of the MGA-G100 chip implements the interface with the host processor. It includes:

- Decoding of all resources
- Configuration registers
- Bus mastering circuitry

2.3 VGA Graphics Controller

This section of the MGA-G100 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.

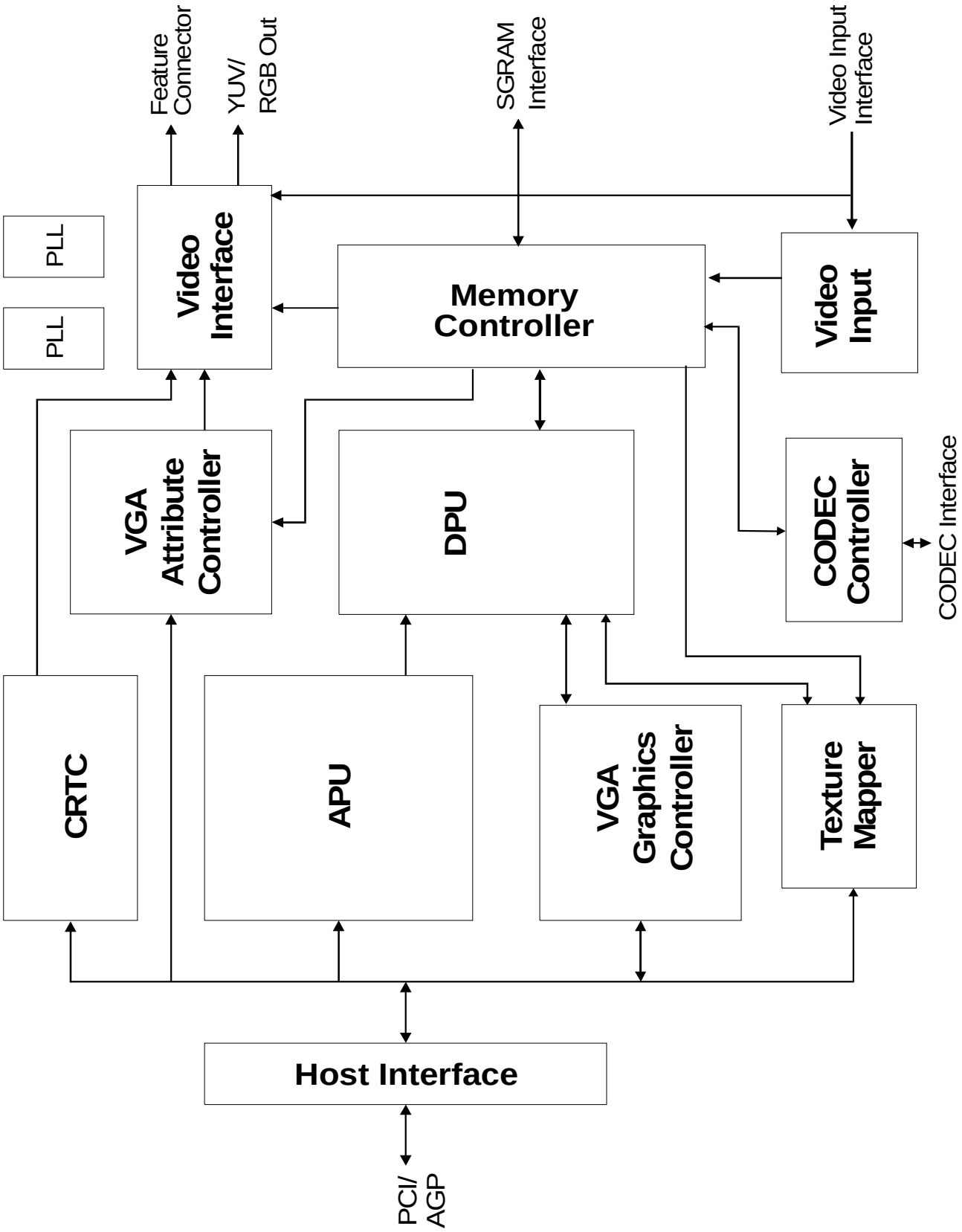
2.5 CRTC

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 12 port, the DAC registers, the DAC, the system clock PLL, and the pixel clock PLL.

Figure 2-1: MGA-G100 Block Diagram



2.7 Address Processing Unit (APU)

This section of the MGA-G100 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
- Fogging circuitry
- Stipple Alpha blending circuitry

2.9 Texture Mapper

This section implements the perspective-correct texture mapping feature of the MGA-G100. It includes:

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

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 SGRAM
- Arbitration of internal requests to the frame buffer

The MGA-G100 chip can interface directly with the SGRAM chips. A frame buffer of up to 16 MBytes is supported.

2.11 Video Input Interface

This section implements the 8 bit video capture interface that is compatible with ITU-656 with encoded sync signals. It includes:

- Double buffered configuration registers
- VBI capture circuitry
- Filtered UV upsampler
- YUV -> RGB color space converter

2.12 CODEC Interface

This section interfaces to various VMI- like hardware CODEC devices. It includes:

- Control registers
- CODEC Interface circuitry

This page is left blank intentionally.

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-G100 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	**MGABASE2**	MGA Frame Buffer Aperture Address
14h-17h	**MGABASE1**	MGA Control Aperture Base
18h-1Bh	**MGABASE3**	MGA ILOAD Aperture Base Address
1Ch-2Bh	Reserved(1)	
2Ch-2Fh	**SUBSYSID**	Subsystem ID. Writing has no effect.
30h-33h	**ROMBASE**	ROM Base Address
34h-37h	**CAP_PTR**	Capabilities Pointer
38h-3Bh	Reserved(1)	
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-53h	**OPTION2**	Option2
54h-DBh	Reserved(1)	
DCh-DFh	**PM_IDENT**	PCI Power Management Capability Identifier
E0h-E3h	**PM_CSR**	PCI Power Management Control/Status
E4h-EFh	Reserved(1)	
F0h-F3h	**AGP_IDENT**(2)	AGP Capability Identifier
F4h-F7h	**AGP_STS**(2)	AGP Status
F8h-FBh	**AGP_CMD**(2)	AGP Command
FCh-FFh	Reserved(1)	

(1) 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.

(2) These locations exist only for the MGA-G100-AGP. For the MGA-G100-PCI, all these locations are reserved and '0' will be returned when read.

3.1.2 MGA General Map

Table 3-2: MGA General Map

Address	*Condition*	*Name/Notes*
000A0000h-000BFFFFh	**GCTL6**<3:2> = '00', **MISC**<1> = '1'	VGA frame buffer (1) (2)
000A0000h-000AFFFFh	**GCTL6**<3:2> = '01', **MISC**<1> = '1'	
000B0000h-000B7FFFh	**GCTL6**<3:2> = '10', **MISC**<1> = '1'	
000B8000h-000BFFFFh	**GCTL6**<3:2> = '11', **MISC**<1> = '1'	
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** + FFFFFFh	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)
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	EXP	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) Reserved locations are not decoded. Accesses outside the defined 32 bit registers will cause unpredictable behavior.

3.2 Register Mapping

2Note: 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 10)

Register Mnemonic Name	*Access*	*Memory Address(1)*	*I/O Address(2)*	*Index*	*Description/Comments*	*Page*
DWGCTL (3)	WO	1C00h(4)	-	-	Drawing Control	4-63
MACCESS (3)	WO	1C04h(4)	-	-	Memory Access	4-82
MCTLWTST	WO	1C08h(4)	-	-	Memory Control Wait State	4-84
ZORG	WO	1C0Ch(4)	-	-	Z-Depth Origin	4-128
PAT0	WO	1C10h(4)	-	-	Pattern	4-87
PAT1	WO	1C14h(4)	-	-	Pattern	4-87
-	WO	1C18h(4)	-	-	Reserved	-
PLNWT (3)	WO	1C1Ch(4)	-	-	Plane Write Mask	4-89
BCOL	WO	1C20h(4)	-	-	Background Color / Blit Color Mask	4-38
FCOL	WO	1C24h(4)	-	-	Foreground Color / Blit Color Key	4-70
-	WO	1C28h(4)	-	-	Reserved	-
-	WO	1C2Ch(4)	-	-	Reserved (SRCBLT)	-
SRC0	WO	1C30h(4)	-	-	Source	4-98
SRC1	WO	1C34h(4)	-	-	Source	4-98
SRC2	WO	1C38h(4)	-	-	Source	4-98
SRC3	WO	1C3Ch(4)	-	-	Source	4-98
XYSTRT (5)	WO	1C40h(4)	-	-	XY Start Address	4-122
XYEND (5)	WO	1C44h(4)	-	-	XY End Address	4-121
-	1C48h-1C4Ch(4)		-	-	Reserved	-
SHIFT (5)	WO	1C50h(4)	-	-	Funnel Shifter Control	4-96
DMAPAD (5)	WO	1C54h(4)	-	-	DMA Padding	4-46
SGN (5)	WO	1C58h(4)	-	-	Sign	4-95
LEN (5)	WO	1C5Ch(4)	-	-	Length	4-81
AR0 (5)	WO	1C60h(4)	-	-	Multi-Purpose Address 0	4-31
AR1 (5)	WO	1C64h(4)	-	-	Multi-Purpose Address 1	4-32
AR2 (5)	WO	1C68h(4)	-	-	Multi-Purpose Address 2	4-33
AR3 (5)	WO	1C6Ch(4)	-	-	Multi-Purpose Address 3	4-34
AR4 (5)	WO	1C70h(4)	-	-	Multi-Purpose Address 4	4-35

Table 3-4: Register Map (Part 2 of 10)

Register Mnemonic Name	*Access*	*Memory Address(1)*	*I/O Address(2)*	*Index*	*Description/Comments*	*Page*
AR5 (5)	WO	1C74h(4)	-	-	Multi-Purpose Address 5	4-36
AR6 (5)	WO	1C78h(4)	-	-	Multi-Purpose Address 6	4-37
-	WO	1C7Ch(4)	-	-	Reserved	-
CXBNDRY (5)	WO	1C80h(4)	-	-	Clipper X Boundary	4-39
FXBNDRY (5)	WO	1C84h(4)	-	-	X Address (Boundary)	4-76
YDSTLEN (5)	WO	1C88h(4)	-	-	Y Destination and Length	4-125
PITCH (5)	WO	1C8Ch(4)	-	-	Memory Pitch	4-88
YDST (5)	WO	1C90h(4)	-	-	Y Address	4-124
YDSTORG (5)	WO	1C94h(4)	-	-	Memory Origin	4-126
YTOP (5)	WO	1C98h(4)	-	-	Clipper Y Top Boundary	4-127
YBOT (5)	WO	1C9Ch(4)	-	-	Clipper Y Bottom Boundary	4-123
CXLEFT (5)	WO	1CA0h(4)	-	-	Clipper X Minimum Boundary	4-40
CXRIGHT (5)	WO	1CA4h(4)	-	-	Clipper X Maximum Boundary	4-41
FXLEFT (5)	WO	1CA8h(4)	-	-	X Address (Left)	4-77
FXRIGHT (5)	WO	1CACh(4)	-	-	X Address (Right)	4-78
XDST (5)	WO	1CB0h(4)	-	-	X Destination Address	4-120
-	1CB4h-1CBCh(4)		-	-	Reserved	-
DR0	WO	1CC0h(4)	-	-	Data ALU 0	4-53
FOGSTART	WO	1CC4h(4)	-	-	Fog Start	4-73
DR2	WO	1CC8h(4)	-	-	Data ALU 2	4-51
DR3	WO	1CCCh(4)	-	-	Data ALU 3	4-52
DR4	WO	1CD0h(4)	-	-	Data ALU 4	4-53
FOGXINC	WO	1CD4h(4)	-	-	Fog X Inc	4-74
DR6	WO	1CD8h(4)	-	-	Data ALU 6	4-54
DR7	WO	1CDCh(4)	-	-	Data ALU 7	4-55
DR8	WO	1CE0h(4)	-	-	Data ALU 8	4-56
FOGYINC	WO	1CE4h(4)	-	-	Fog Y Inc	4-75
DR10	WO	1CE8h(4)	-	-	Data ALU 10	4-57
DR11	WO	1CECh(4)	-	-	Data ALU 11	4-58
DR12	WO	1CF0h(4)	-	-	Data ALU 12	4-59
FOGCOL	WO	1CF4h(4)	-	-	Fog Color	4-72

Table 3-4: Register Map (Part 3 of 10)

Register Mnemonic Name	*Access*	*Memory Address(1)*	*I/O Address(2)*	*Index*	*Description/Comments*	*Page*
DR14	WO	1CF8h(4)	-	-	Data ALU 14	4-60
DR15	WO	1CFCh(4)	-	-	Data ALU 15	4-61
-	1D00h-1DFFh(4)		-	-	Same mapping as 1C00h-1CFCh (6)	-
-	1E00h - 1E0Fh		-	-	Reserved	-
FIFOSTATUS	RO	1E10h	-	-	Bus FIFO Status	4-71
STATUS	RO	1E14h	-	-	Status	4-99
ICLEAR	WO	1E18h	-	-	Interrupt Clear	4-79
IEN	R/W	1E1Ch	-	-	Interrupt Enable	4-80
VCOUNT	RO	1E20h	-	-	Vertical Count	4-119
-	1E24h - 1E2Fh		-	-	Reserved	-
DMAMAP30	R/W	1E30h	-	-	DMA Map 3h to 0h	4-42
DMAMAP74	R/W	1E34h	-	-	DMA Map 7h to 4h	4-43
DMAMAPB8	R/W	1E38h	-	-	DMA Map Bh to 8h	4-44
DMAMAPFC	R/W	1E3Ch	-	-	DMA Map Fh to Ch	4-45
RST	R/W	1E40h	-	-	Reset	4-93
-	1E44h - 1E53h		-	-	Reserved	-
OPMODE	R/W	1E54h	-	-	Operating Mode	4-86
PRIMADDRESS	R/W	1E58h	-	-	Primary DMA Current Address	4-90
PRIMEND	R/W	1E5Ch	-	-	Primary DMA End Address	4-91
-	1E60h - 1E7Fh		-	-	Reserved	-
DWG_INDIR_WT<0>	WO	1E80h(4)	-	-	Drawing Register Indirect Write 0	4-62
DWG_INDIR_WT<1>	WO	1E84h(4)	-	-	Drawing Register Indirect Write 1	4-62
DWG_INDIR_WT<2>	WO	1E88h(4)	-	-	Drawing Register Indirect Write 2	4-62
DWG_INDIR_WT<3>	WO	1E8Ch(4)	-	-	Drawing Register Indirect Write 3	4-62
DWG_INDIR_WT<4>	WO	1E90h(4)	-	-	Drawing Register Indirect Write 4	4-62
DWG_INDIR_WT<5>	WO	1E94h(4)	-	-	Drawing Register Indirect Write 5	4-62
DWG_INDIR_WT<6>	WO	1E98h(4)	-	-	Drawing Register Indirect Write 6	4-62
DWG_INDIR_WT<7>	WO	1E9Ch(4)	-	-	Drawing Register Indirect Write 7	4-62
DWG_INDIR_WT<8>	WO	1EA0h(4)	-	-	Drawing Register Indirect Write 8	4-62
DWG_INDIR_WT<9>	WO	1EA4h(4)	-	-	Drawing Register Indirect Write 9	4-62
DWG_INDIR_WT<10>	WO	1EA8h(4)	-	-	Drawing Register Indirect Write 10	4-62
DWG_INDIR_WT<11>	WO	1EACh(4)	-	-	Drawing Register Indirect Write 11	4-62
DWG_INDIR_WT<12>	WO	1EB0h(4)	-	-	Drawing Register Indirect Write 12	4-62
DWG_INDIR_WT<13>	WO	1EB4h(4)	-	-	Drawing Register Indirect Write 13	4-62

Table 3-4: Register Map (Part 4 of 10)

Register Mnemonic Name	*Access*	*Memory Address(1)*	*I/O Address(2)*	*Index*	*Description/Comments*	*Page*
DWG_INDIR_WT<14>	WO	1EB8h(4)	-	-	Drawing Register Indirect Write 14	4-62
DWG_INDIR_WT<15>	WO	1EBCh(4)	-	-	Drawing Register Indirect Write 15	4-62
-	1EC0h - 1FBFh		-	-	Reserved	-
ATTR (Index)	R/W	1FC0h	3C0h	-	Attribute Controller	4-130
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	4-132
ATTR1	R/W	-	-	01h	Palette entry 1	4-132
ATTR2	R/W	-	-	02h	Palette entry 2	4-132
ATTR3	R/W	-	-	03h	Palette entry 3	4-132
ATTR4	R/W	-	-	04h	Palette entry 4	4-132
ATTR5	R/W	-	-	05h	Palette entry 5	4-132
ATTR6	R/W	-	-	06h	Palette entry 6	4-132
ATTR7	R/W	-	-	07h	Palette entry 7	4-132
ATTR8	R/W	-	-	08h	Palette entry 8	4-132
ATTR9	R/W	-	-	09h	Palette entry 9	4-132
ATTRA	R/W	-	-	0Ah	Palette entry A	4-132
ATTRB	R/W	-	-	0Bh	Palette entry B	4-132
ATTRC	R/W	-	-	0Ch	Palette entry C	4-132
ATTRD	R/W	-	-	0Dh	Palette entry D	4-132
ATTRE	R/W	-	-	0Eh	Palette entry E	4-132
ATTRF	R/W	-	-	0Fh	Palette entry F	4-132
ATTR10	R/W	-	-	10h	Attribute Mode Control	4-133
ATTR11	R/W	-	-	11h	Overscan Color	4-135
ATTR12	R/W	-	-	12h	Color Plane Enable	4-136
ATTR13	R/W	-	-	13h	Horizontal Pel Panning	4-137
ATTR14	R/W	-	-	14h	Color Select	4-138
-	-	-	-	15h - 1Fh: Reserved		-
INSTS0	RO	1FC2h	3C2h	-	Input Status 0	4-194
MISC	WO	1FC2h	3C2h	-	Miscellaneous Output	4-196
-	R/W	1FC3h	3C3h(7)	-	Reserved, not decoded for I/O	
SEQ (Index)	R/W	1FC4h	3C4h	-	Sequencer	4-198
SEQ (Data)	R/W	1FC5h	3C5h	-	Sequencer	-
SEQ0	R/W	-	-	00h	Reset	4-199
SEQ1	R/W	-	-	01h	Clocking Mode	4-200

Table 3-4: Register Map (Part 5 of 10)

Register Mnemonic Name	*Access*	*Memory Address(1)*	*I/O Address(2)*	*Index*	*Description/Comments*	*Page*
SEQ2	R/W	-	-	02h	Map Mask	4-201
SEQ3	R/W	-	-	03h	Character Map Select	4-202
SEQ4	R/W	-	-	04h	Memory Mode	4-203
-	R/W	-	-	05h - 07h: Reserved		-
-	-	1FC6h	-	-	Reserved	-
DACSTAT	RO	1FC7h	3C7h	-	DAC Status (requires a byte access)	4-181
-	WO	1FC7h	-	-	Reserved	-
-	1FC8h-1FC9h		-	-	Reserved	-
FEAT	RO	1FCAh	3CAh	-	Feature Control	4-182
	WO	1FCAh	3CAh	-	Reserved	-
-	-	1FCBh	3CBh(7)	-	Reserved, not decoded for I/O	-
MISC	RO	1FCCh	3CCh	-	Miscellaneous Output	4-196
-	WO	1FCCh	3CCh	-	Reserved	-
-	-	1FCDh	3CDh(7)	-	Reserved, not decoded for I/O	-
GCTL (Index)	R/W	1FCEh	3CEh	-	Graphic Controller	4-183
GCTL (Data)	R/W	1FCFh	3CFh	-	Graphic Controller	-
GCTL0	R/W	-	-	00h	Set/Reset	4-184
GCTL1	R/W	-	-	01h	Enable Set/Reset	4-185
GCTL2	R/W	-	-	02h	Color Compare	4-186
GCTL3	R/W	-	-	03h	Data Rotate	4-187
GCTL4	R/W	-	-	04h	Read Map Select	4-188
GCTL5	R/W	-	-	05h	Graphics Mode	4-189
GCTL6	R/W	-	-	06h	Miscellaneous	4-191
GCTL7	R/W	-	-	07h	Color Don’t Care	4-192
GCTL8	R/W	-	-	08h	Bit Mask	4-193
-	-	-	-	09h - 0Fh: Reserved		-
-	1FD0h-1FD3h		-	-	Reserved	-
CRTC (Index)	R/W	1FD4h	3D4h	-	CRTC Registers (or 3B4h(8))	4-140
CRTC (Data)	R/W	1FD5h	3D5h	-	CRTC Registers (or 3B5h(8))	-
CRTC0	R/W	-	-	00h	Horizontal Total	4-142
CRTC1	R/W	-	-	01h	Horizontal Display Enable End	4-143
CRTC2	R/W	-	-	02h	Start Horizontal Blanking	4-144
CRTC3	R/W	-	-	03h	End Horizontal Blanking	4-145
CRTC4	R/W	-	-	04h	Start Horizontal Retrace Pulse	4-146
CRTC5	R/W	-	-	05h	End Horizontal Retrace	4-147
CRTC6	R/W	-	-	06h	Vertical Total	4-148

Table 3-4: Register Map (Part 6 of 10)

Register Mnemonic Name	*Access*	*Memory Address(1)*	*I/O Address(2)*	*Index*	*Description/Comments*	*Page*
CRTC7	R/W	-	-	07h	Overflow	4-149
CRTC8	R/W	-	-	08h	Preset Row Scan	4-150
CRTC9	R/W	-	-	09h	Maximum Scan Line	4-151
CRTCA	R/W	-	-	0Ah	Cursor Start	4-152
CRTCB	R/W	-	-	0Bh	Cursor End	4-153
CRTCC	R/W	-	-	0Ch	Start Address High	4-154
CRTCD	R/W	-	-	0Dh	Start Address Low	4-155
CRTCE	R/W	-	-	0Eh	Cursor Location High	4-156
CRTCF	R/W	-	-	0Fh	Cursor Location Low	4-157
CRTC10	R/W	-	-	10h	Vertical Retrace Start	4-158
CRTC11	R/W	-	-	11h	Vertical Retrace End	4-159
CRTC12	R/W	-	-	12h	Vertical Display Enable End	4-160
CRTC13	R/W	-	-	13h	Offset	4-161
CRTC14	R/W	-	-	14h	Underline Location	4-162
CRTC15	R/W	-	-	15h	Start Vertical Blank	4-163
CRTC16	R/W	-	-	16h	End Vertical Blank	4-164
CRTC17	R/W	-	-	17h	CRTC Mode Control	4-165
CRTC18	R/W	-	-	18h	Line Compare	4-169
-	-	-	-	19h - 21h: Reserved		-
CRTC22	R/W	-	-	22h	CPU Read Latch	4-170
-	-	-	-	23h	Reserved	-
CRTC24	R/W	-	-	24h	Attributes Address/Data Select	4-171
-	-	-	-	25h	Reserved	-
CRTC26	R/W	-	-	26h	Attributes Address	4-172
-	-	-	-	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))	4-195
FEAT	WO	1FDAh	3DAh	-	Feature Control (or 3BAh(8))	4-182
-	-	1FDBh	3DBh(7)	-	Reserved, not decoded for I/O (or 3BBh(8))	-
-	1FDCh-1FDDh		-	-	Reserved	-
CRTCEXT (Index)	R/W	1FDEh	3DEh	-	CRTC Extension	4-173
CRTCEXT (Data)	R/W	1FDFh	3DFh	-	CRTC Extension	-

Table 3-4: Register Map (Part 7 of 10)

Register Mnemonic Name	*Access*	*Memory Address(1)*	*I/O Address(2)*	*Index*	*Description/Comments*	*Page*
CRTCEXT0	R/W	-	-	00h	Address Generator Extensions	4-174
CRTCEXT1	R/W	-	-	01h	Horizontal Counter Extensions	4-175
CRTCEXT2	R/W	-	-	02h	Vertical Counter Extensions	4-176
CRTCEXT3	R/W	-	-	03h	Miscellaneous	4-177
CRTCEXT4	R/W	-	-	04h	Memory Page	4-178
CRTCEXT5	R/W	-	-	05h	Horizontal Video Half Count	4-179
CRTCEXT6	R/W	-	-	06h	Priority Request Control	4-180
-	1FE0h - 1FFEh		-	-	Reserved	-
CACHEFLUSH	R/W	1FFFh	-	-	Cache Flush	4-139
TMR0	WO	2C00h(4)	-	-	Texture Mapping ALU 0	4-**110**
TMR1	WO	2C04h(4)	-	-	Texture Mapping ALU 1	4-**111**
TMR2	WO	2C08h(4)	-	-	Texture Mapping ALU 2	4-**112**
TMR3	WO	2C0Ch(4)	-	-	Texture Mapping ALU 3	4-**113**
TMR4	WO	2C10h(4)	-	-	Texture Mapping ALU 4	4-**114**
TMR5	WO	2C14h(4)	-	-	Texture Mapping ALU 5	4-**115**
TMR6	WO	2C18h(4)	-	-	Texture Mapping ALU 6	4-**116**
TMR7	WO	2C1Ch(4)	-	-	Texture Mapping ALU 7	4-**117**
TMR8	WO	2C20h(4)	-	-	Texture Mapping ALU 8	4-**118**
TEXORG	WO	2C24h(4)	-	-	Texture Origin	4-107
TEXWIDTH	WO	2C28h(4)	-	-	Texture Width	4-**109**
TEXHEIGHT	WO	2C2Ch(4)	-	-	Texture Height	4-**105**
TEXCTL	WO	2C30h(4)	-	-	Texture Map Control	4-**101**
TEXTRANS	WO	2C34h(4)	-	-	Texture Transparency	4-**108**
-	2C38h-2C3Ch(4)		-	-	Reserved	-
SECADDRESS	R/W	2C40h	-	-	Secondary DMA Current Address (9)	4-**93**
SECEND	R/W	2C44h	-	-	Secondary DMA End Address (9)	4-**94**
SOFTRAP	R/W	2C48h	-	-	Soft Trap Handle (9)	4-**97**
-	-	2C4Ch(4)	-	-	Reserved	-
DR0_Z32LSB, DR0_Z32MSB	WO	2C50h, 2C54h(4)	-	-	Extended Data ALU 0	4-47
TEXFILTER	WO	2C58h(4)	-	-	Texture Filtering	4-106
-	-	2C5Ch	-	-	Reserved	-
DR2_Z32LSB, DR2_Z32MSB	WO	2C60h, 2C64h(4)	-	-	Extended Data ALU 2	4-48

Table 3-4: Register Map (Part 8 of 10)

Register Mnemonic Name	*Access*	*Memory Address(1)*	*I/O Address(2)*	*Index*	*Description/Comments*	*Page*
DR3_Z32LSB, DR3_Z32MSB	WO	2C68h, 2C6Ch(4)	-	-	Extended Data ALU 3	4-49
ALPHASTART	WO	2C70h(4)	-	-	Alpha Start	4-28
ALPHAXINC	WO	2C74h(4)	-	-	Alpha X Inc	4-29
ALPHAYINC	WO	2C78h(4)	-	-	Alpha Y Inc	4-30
ALPHACTRL	WO	2C7Ch(4)	-	-	Alpha CTRL	4-27
-	2C80h-2CFFh(4)		-	-	Reserved	-
-	2D00h-2DFFh(4)		-	-	Same mapping as 2C00-2CFC (6)	-
PALWTADD	R/W	3C00h	3C8h	-	Palette RAM Addr. Write/Load Index	4-209
PALDATA	R/W	3C01h	3C9h	-	Palette RAM Data Register	4-207
PIXRDMSK	R/W	3C02h	3C6h	-	Pixel Read Mask	4-210
PALRDADD	R/W	3C03h	3C7h	-	Palette RAM Address - Read. This register is WO for I/O accesses.	4-208
-	3C04h - 3C09h		-	-	Reserved	-
X_DATAREG	R/W	3C0Ah	-	-	Indexed Data Register	4-211
-	-	-	-	00h - 03h: Reserved		-
XCURADDL	R/W	-	-	04h	Cursor Base Address, Low	4-221
XCURADDH	R/W	-	-	05h	Cursor Base Address, High	4-220
XCURCTRL	R/W	-	-	06h	Cursor Control	4-223
-	-	-	-	07h	Reserved	-
XCURCOL0RED	R/W	-	-	08h	Cursor color 0 Red	4-222
XCURCOL0GREEN	R/W	-	-	09h	Cursor color 0 Green	4-222
XCURCOL0BLUE	R/W	-	-	0Ah	Cursor color 0 Blue	4-222
-	-	-	-	0Bh	Reserved	-
XCURCOL1RED	R/W	-	-	0Ch	Cursor color 1 Red	4-222
XCURCOL1GREEN	R/W	-	-	0Dh	Cursor color 1 Green	4-222
XCURCOL1BLUE	R/W	-	-	0Eh	Cursor Color 1 Blue	4-222
-	-	-	-	0Fh	Reserved	-
XCURCOL2RED	R/W	-	-	10h	Cursor Color 2 Red	4-222
XCURCOL2GREEN	R/W	-	-	11h	Cursor Color 2 Green	4-222
XCURCOL2BLUE	R/W	-	-	12h	Cursor Color 2 Blue	4-222
-	-	-	-	13h - 17h: Reserved		-
XVREFCTRL	R/W			18h	Voltage Reference Control	4-240
XMULCTRL	R/W	-	-	19h	Multiplex Control	4-229
XPIXCLKCTRL	R/W	-	-	1Ah	Pixel Clock Control	4-230
-	-	-	-	1Bh - 1Ch: Reserved		-
XGENCTRL	R/W	-	-	1Dh	General Control	4-224

Table 3-4: Register Map (Part 9 of 10)

Register Mnemonic Name	*Access*	*Memory Address(1)*	*I/O Address(2)*	*Index*	*Description/Comments*	*Page*
XMISCCTRL	R/W	-	-	1Eh	Miscellaneous Control	4-227
-	-	-	-	1Fh - 29h: Reserved		-
XGENIOCTRL	R/W	-	-	2Ah	General Purpose I/O Control	4-225
XGENIODATA	R/W	-	-	2Bh	General Purpose I/O Data	4-226
XSYSPLLM	R/W	-	-	2Ch	SYSPLL M Value Register	4-236
XSYSPLLN	R/W	-	-	2Dh	SYSPLL N Value Register	4-237
XSYSPLLP	R/W	-	-	2Eh	SYSPLL P Value Register	4-238
XSYSPLLSTAT	RO	-	-	2Fh	SYSPLL Status	4-239
-	-	-	-	30h - 37h: Reserved		-
XZOOMCTRL	R/W	-	-	38h	Zoom Control	4-241
-	-	-	-	39h	Reserved	-
XSENSETEST	R/W	-	-	3Ah	Sense Test	4-235
-	-	-	-	3Bh	Reserved	-
XCRCREML	RO	-	-	3Ch	CRC Remainder Low	4-219
XCRCREMH	RO	-	-	3Dh	CRC Remainder High	4-218
XCRCBITSEL	R/W	-	-	3Eh	CRC Bit Select	4-217
-	-	-	-	3Fh	Reserved	
XCOLKEYMSKL	R/W	-	-	40h	Color Key Mask, Low	4-216
XCOLKEYMSKH	R/W	-	-	41h	Color Key Mask, High	4-215
XCOLKEYL	R/W	-	-	42h	Color Key, Low	4-214
XCOLKEYH	R/W	-	-	43h	Color Key, High	4-213
XPIXPLLAM	R/W	-	-	44h	PIXPLL M Value Register Set A	4-231
XPIXPLLAN	R/W	-	-	45h	PIXPLL N Value Register Set A	4-232
XPIXPLLAP	R/W	-	-	46h	PIXPLL P Value Register Set A	4-233
-	-	-	-	47h	Reserved	-
XPIXPLLBM	R/W	-	-	48h	PIXPLL M Value Register Set B	4-231
XPIXPLLBN	R/W	-	-	49h	PIXPLL N Value Register Set B	4-232
XPIXPLLBP	R/W	-	-	4Ah	PIXPLL P Value Register Set B	4-233
-	-	-	-	4Bh	Reserved	-
XPIXPLLCM	R/W	-	-	4Ch	PIXPLL M Value Register Set C	4-231
XPIXPLLCN	R/W	-	-	4Dh	PIXPLL N Value Register Set C	4-232
XPIXPLLCP	R/W	-	-	4Eh	PIXPLL P Value Register Set C	4-233
XPIXPLLSTAT	RO	-	-	4Fh	PIXPLL Status	4-234
-	-	-	-	50h - FFh: Reserved		-
-	-	3C0Bh	-	-	Reserved	-
CURPOSXL	R/W	3C0Ch	-	-	Cursor Position X LSB	4-206
CURPOSXH	R/W	3C0Dh	-	-	Cursor Position X MSB	4-206

Table 3-4: Register Map (Part 10 of 10)

Register Mnemonic Name	*Access*	*Memory Address*[1]	*I/O Address*[2]	*Index*	*Description/Comments*	*Page*
CURPOSYL	R/W	3C0Eh	-	-	Cursor Position Y LSB	4-206
CURPOSYH	R/W	3C0Fh	-	-	Cursor Position Y MSB	4-206
-	3C10h - 3DFFh		-	-	Reserved	-
VINCTL0	WO	3E00h[4]	-	-	Video Input Control Window 0	4-256
VINCTL1	WO	3E04h[4]	-	-	Video Input Control Window 1	4-257
VBIADDR0	WO	3E08h[4]	-	-	VBI Address Window 0	4-249
VBIADDR1	WO	3E0Ch[4]	-	-	VBI Address Window 1	4-250
VINADDR0	WO	3E10h[4]	-	-	Video Input Address Window 0	4-253
VINADDR1	WO	3E14h[4]	-	-	Video Input Address Window 1	4-254
VINNEXTWIN	WO	3E18h[4]	-	-	Video Input Next Window	4-258
VINCTL	WO	3E1Ch[4]	-	-	Video Input Control	4-255
-	3E20h - 3E2Ch		-	-	Reserved	-
VSTATUS	RO	3E30h	-	-	Video Status	4-259
VICLEAR	WO	3E34h[4]	-	-	Video Interrupt Clear	4-251
VIEN	WO	3E38h[4]	-	-	Video Interrupt Enable	4-252
-		3E3Ch	-	-	Reserved	-
CODECCTL	WO	3E40h[4]	-	-	CODEC Control	4-244
CODECADDR	WO	3E44h[4]	-	-	CODEC Buffer Start Address	4-243
CODECHOSTPTR	WO	3E48h[4]	-	-	CODEC Host Pointer	4-248
CODECHARDPTR	RO	3E4Ch	-	-	CODEC Hard Pointer	4-247
CODECLCODE	RO	3E50h	-	-	CODEC LCODE Pointer	4-246
-	3E54h - 3FFFh		-	-	Reserved	-

(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.

(9) These registers are not writable through **MGABASE1** + 2C4xh. They can only be written via bus mastering operations from the MGA-G100. They ***can*** be read through **MGABASE1** + 2C4xh.

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.

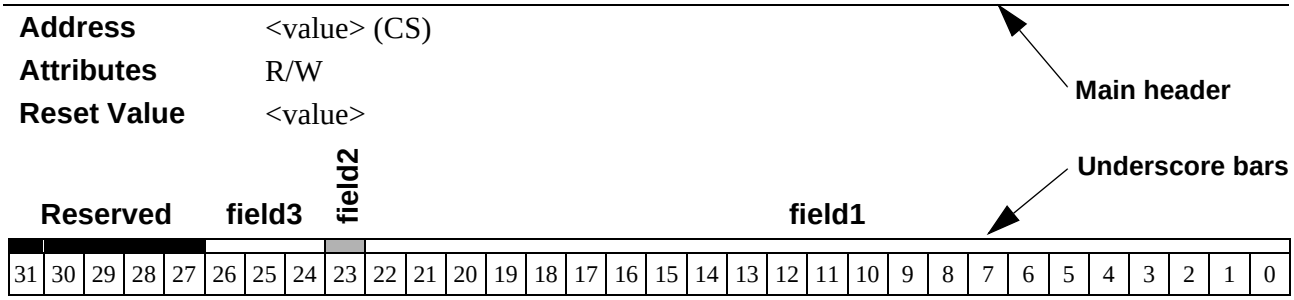
4.1 Power Graphic Mode Register Descriptions

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:

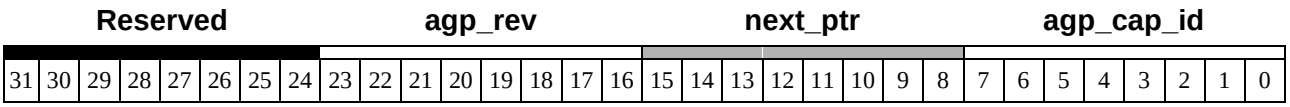
n 000? 0000 000S ???? 1101 0000 S000 0000b

(b = binary, ? = unknown, S = bit's reset value is affected by a strap setting, N/A = not applicable)

Address F0h (CS) for MGA-G100-AGP only

Attributes RO, BYTE/WORD/DWORD, STATIC

Reset Value 0000 0000 0001 0000 0000 0000 0000 0010b



agp_cap_id <7:0> This field contains the AGP capabilities identifier: 02h, which describes the information contained in the capability entry (F0h-F8h)

next_ptr <15:8> This field contains the hard coded value of 00h, which indicates that it is the last capability in the list.

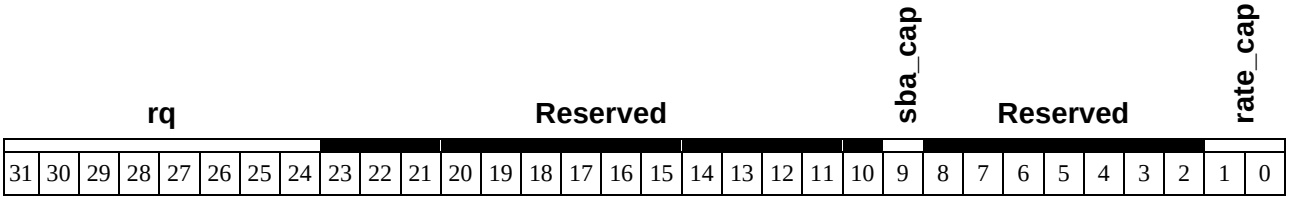
agp_rev <23:16> This field contains the AGP specification revision to which this device complies: 10h (as in 1.0)

Reserved <31:24> Reserved. When writing to this register, the bits in these fields must be set to ‘0’. Reading will give ‘0’s.

Address F4h (CS) for MGA-G100-AGP only

Attributes RO, BYTE/WORD/DWORD, STATIC

Reset Value 0000 00001 0000 0000 0000 0010 0000 0001b



rate_cap <1:0> The hard coded “01b” indicates that the device supports AGP transfer rate of 1x.

sba_cap <9> The hard coded “1b” indicates that the device does support AGP Sideboard addressing.

rq <31:24> The hard coded “01h” indicates that the device can manage a maximum number of 2 AGP requests.

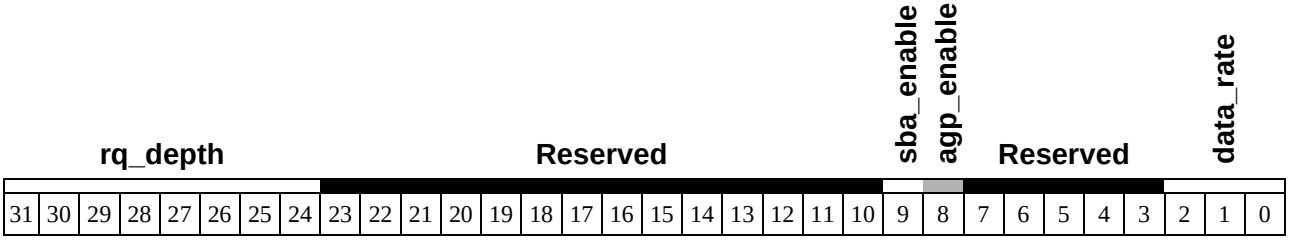
Reserved: <23:10> <8:2>

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

Address F8h (CS) for MGA-G100-AGP only

Attributes RW, BYTE/WORD/DWORD, STATIC only

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



data_rate <2:0> Indicates the operational data rate of the device. Only one bit in this field must be set:

- 001: 1 x data rate
- xx0: Reserved
- 1xx: Reserved
- x1x: Reserved

agp_enable <8> When set, this bit enables the master (this device) to initiate AGP operation.

sba_enable <9> When set, the side address mechanism of the device is enabled.

rq_depth <31:24> This should be programmed with the maximum number of pipelined operations that the master (this device) is allowed to queue. This value should be equal, or less, than the value reported in the target rq field of AGP_STS register.

Reserved: <23:10> <7:3>

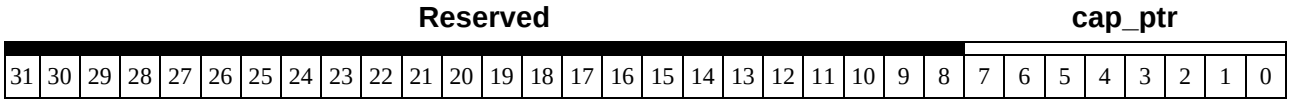
Reserved. When writing to this register, the bits in these fields must be set to '0'. Reading will give '0's.

Note: **agp-enable** and **sba-enable** must be '1' and **data_rate<2:0>** must be programmed to "001b" for AGP cycles to be initiated since the only AGP cycles are those with sideband signals.

Address 34h (CS)

Attributes RO, BYTE/WORD/DWORD, STATIC

Reset Value 0000 0000 0000 0000 0000 0000 1101 1100b



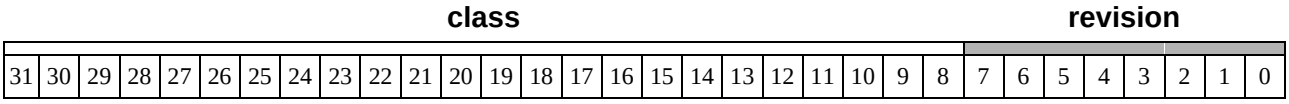
cap_ptr RO<7:0> This field contains the hard-coded offset byte (DCh) within the device configuration space of the PCI Power Management Capability Identifier register.

Reserved <31:8> Reserved. When writing to this register, the bits in these fields must be set to ‘0’. Reading will give ‘0’s.

Address 08h (CS)

Attributes RO, BYTE/WORD/DWORD, STATIC

Reset Value 0000 0011 S000 0000 0000 0000 0000 0000b



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.

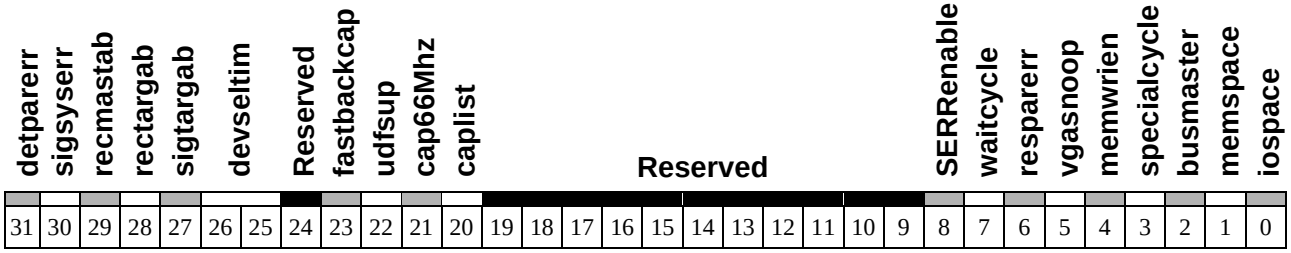
vgaboot strap	*Value*	*Meaning*
'0'	038000h	Non-Super VGA display controller
'1'	030000h	Super VGA compatible controller

The sampled state of the vgaboot strap (pin HDATA<0> described on page A-5) can be read through this register.

Address 04h (CS)

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

Reset Value 0000 0010 1001 0000 0000 0000 1000 0000b



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 4-18)** 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 address/data stepping is performed for read accesses in the target (data stepping) and the master (address stepping). Writing has no effect.

SERRenable
RO <8>
This hard coded ‘0’ indicates that MGA-G100 does not generate SERR interrupts. Writing has no effect.

caplist
RO <20>
The hard coded ‘1’ for MGA-G100 indicates that the device has a capability list in the configuration space. The list is located at the offset in the **CAP_PTR** register.

cap66Mhz
RO <21>
The hard coded ‘0’ indicates that the MGA-G100-PCI is running at 33 MHz or lower clock rates.Reserved for MGA-G100-AGP.

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.

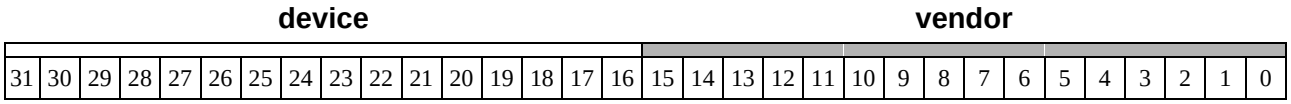
Reserved: <19:9> <24>

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

Address 00h (CS)

Attributes RO, BYTE/WORD/DWORD, STATIC

Reset Value 0001 0000 0000 000? 0001 0000 0010 1011b



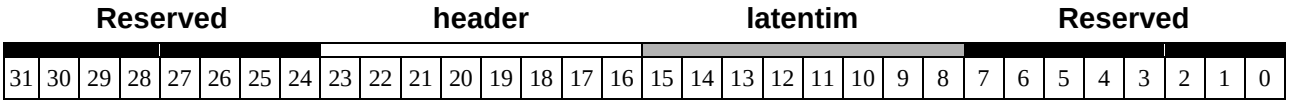
vendor <15:0> This field contains the Matrox manufacturer identifier for PCI: 102Bh.

device <31:16> This field contains the Matrox device identifier, for the MGA-G100- PCI is 1000h; for the MGA-G100-AGP is 1001h.

Address 0Ch (CS)

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

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



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.

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-G100 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-G100 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

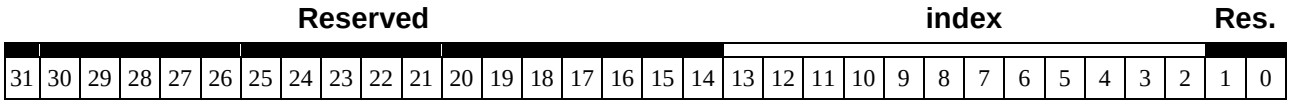
data <31:0> Data. Will read or write data at the control register address provided by **MGA_INDEX**.

The **MGA_INDEX** and **MGA_DATA** registers cannot be used in Pseudo-DMA mode (see page 5-27).

Address 44h (CS)

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

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



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>

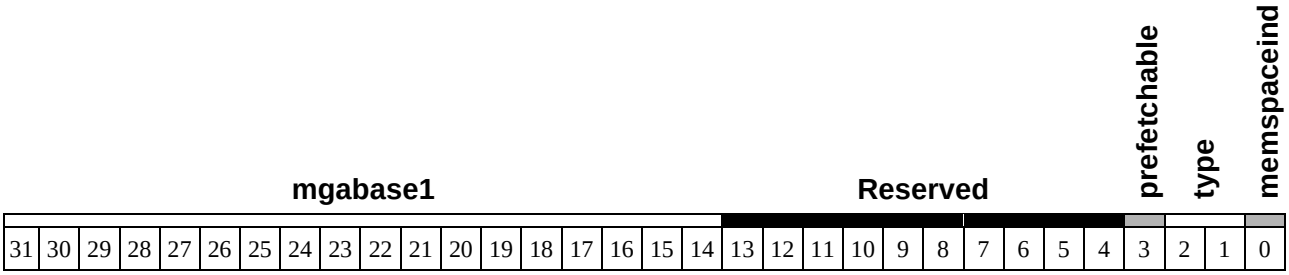
Reserved. When writing to this register, the bits in this field must be set to ‘0’. Reading will give ‘0’s.

The **MGA_INDEX** and **MGA_DATA** registers cannot be used in Pseudo-DMA mode (see page 5-27).

Address 14h (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.

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 10h (CS)

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

Reset Value 0000 0000 0000 0000 0000 0000 0000 1000b

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).

mgabase2 <31:24> Specifies the PCI start address of the 16 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)

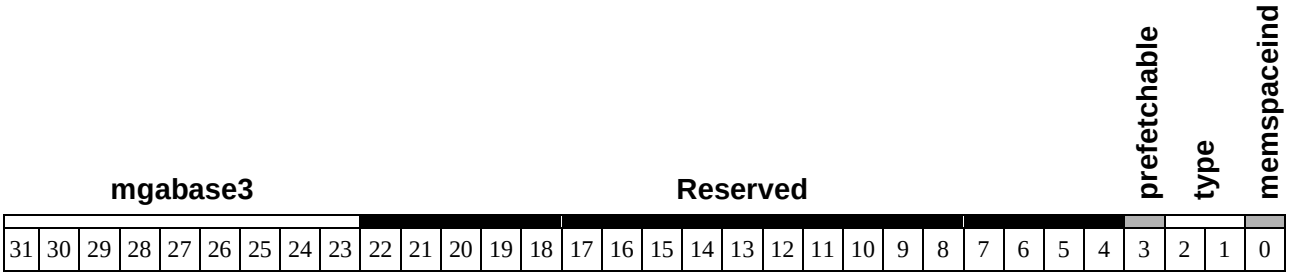
When **mgamode** = 0 (**CRTCEXT3**<7>), the full frame buffer aperture is not available.

Reserved <23:4> Reserved. When writing to this register, the bits in this field must be set to ‘0’. Reading will give ‘0’s.

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:

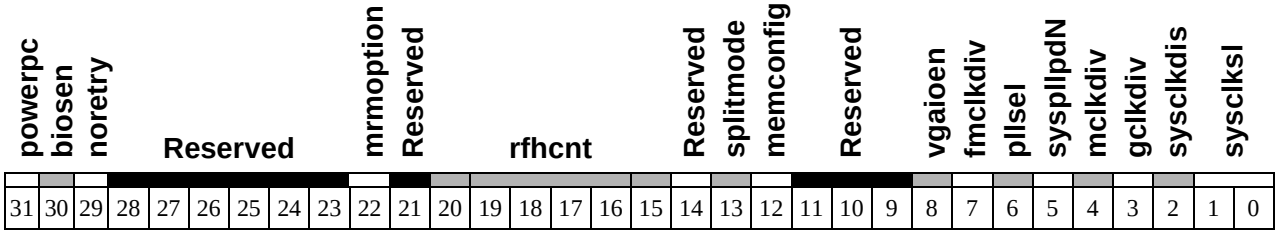
1. BIOS EPROM (highest precedence)
2. MGA control aperture
3. 8 MByte Pseudo-DMA window
4. VGA frame buffer aperture
5. MGA frame buffer aperture (lowest precedence)

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 0S00 0000 0000 0000 0000 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

➧ Note: The system clock selection affects gclk, mclk, fmclk.

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

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

➧ Note: The system clock disabled affects gclk, mclk, fmclk.

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 2
- 1: divide by 3

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: Reserved

syspllpdN <5> System PLL power down.

- 0: power down
- 1: power up

pllsel <6> PLL select.

- 0: PLL0 will be the system PLL
  PLL1 will be the pixel PLL
- 1: PLL0 will be the pixel PLL
  PLL1 will be the system PLL

fmclkdiv <7> Fast memory clock divider select. Selects the ratio by which the system clock is divided in order to produce the fast memory clock when **sysclksl** = '01'.

- 0: divide by 1

• 1: divide by 2

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 HDATA <0> will replace the **vgaioen** value.

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

memconfig <12>

Memory configuration. This bit indicates the configuration of the memory chips which comprise the frame buffer (refer to 'SGRAM Configurations' on page 6-4 for more information regarding pin configurations). It is used by the memory controller to map the addresses according to the following table:

	Internal chip configuration			
memconfig	*No. of Banks*	*Bank Size*	*Word Size*	*Total*
'0'	2	128k	32	8 Mb
'1'	2	256k	32	16 Mb

splitmode <13>

Split frame buffer mode. When this field is '1', the 16 MByte frame buffer is divided into two sections:

• 0 MBytes to (8 MBytes - 1) is the graphics buffer
• 8 MBytes to (16 MBytes - 1) is the video buffer

rfhcnt <20:15>

Refresh counter. Defines the rate of the MGA-G100's memory refresh. Page cycles will not be interrupted by a refresh request unless a second refresh request is queued (in this case, the refresh request becomes the highest priority after the screen refresh).

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

ram refresh period >= (**rfhcnt**<5:0> * 64 + 1) * MCLK period

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

mrmoption <22>

Memory read multiple option

• 0: PCI read command will be : memory read (0110b)
• 1: PCI read command will be : memory read multiple(1100b)

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 (HDATA <1>) 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.

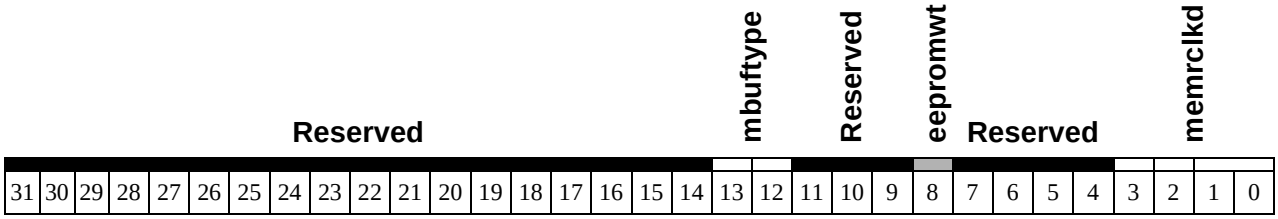
Reserved: <11:9> <14> <21> <28:23>

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

Address 50h (CS)

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

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



memrclkd <3:0> Memory read clock delay. This field is used to adjust the delay on the clock used to register the read-back data, MDQ. **memrclkd** is a pointer which determines where the delay line for the read-back clock will be tapped. Each increment of **memrclkd** adds approximately 0.2ns to the delay on the read-back clock.

memrclkd

"0000" = minumum delay added.

"1111" = maximum delay added.

- *Note:* This field MUST be loaded after initiating a memory reset and before attempting any other access to the frame buffer.
- *Note:* This field should be INVISIBLE to the user. Software will set it based on read/write trials which vary this field.

eepromwt <8> 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.

mbuftype <13:12> Memory buffer type. This field determines the type of input buffer selected for the SGRAM bus.

00: LVTTL input buffers

01: SSTL input buffers

1X: Reserved

- *Note:* This field MUST be loaded before initiating a memory reset.

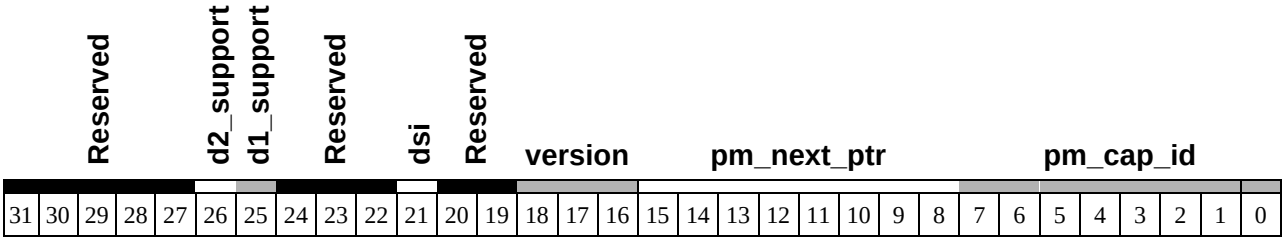
Reserved: <7:4> <11:9> <15:14> <31:14>

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

Address DCh (CS)

Attributes RO, BYTE/WORD/DWORD, STATIC

Reset Value 0000 0000 0010 0001 1111 0000 0000 0001b



pm_cap_id <7:0> This field contains the PCI Power Management Capability Identifier 01h, which describes the information contained in the capability entry (DCh-E3h)

pm_next_ptr <15:8> This field in the MGA-G100-AGP will contain the hard coded offset byte (“F0h”) within the device configuration space of the AGP Capability Identifier register. Reserved for the MGA-G100-PCI.

version <18:16> The hard coded ‘001b’ indicates that MGA-G100 complies with revision 1.0 of the PCI Bus Power Management Interface Specification.

dsi <21> The hard coded ‘1’ indicates that the MGA requires special initialization.

d1_support <25> The hard coded ‘0’ indicates that the MGA does not support the D1 power state.

d2_support <26> The hard coded ‘0’ indicates that the MGA does not support the D2 power state.

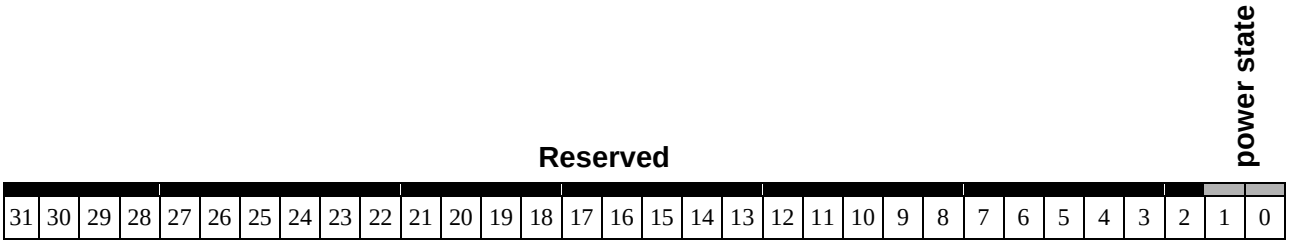
Reserved: <31:27> <24:22> <20:19>

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

Address E0h (CS)

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

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



power state <1:0> Power State. Indicates the status of current power state.

- 00: D0 “On” State

  Back-end : On

  Video Controller Context: Preserved

  Video Memory Contents: Preserved

  Actions to Function: Any PCI Transaction

  Actions from Function: Any PCI Transaction or Interrupt

- 01: Reserved

- 10: Reserved

- 11: D3 “Off” State

  Back-end : Off

  Video Controller Context: Lost (Power removed)

  Video Memory Contents: Lost (Power removed)

  Actions to Function: PCI Config Cycles

  Actions from Function: None

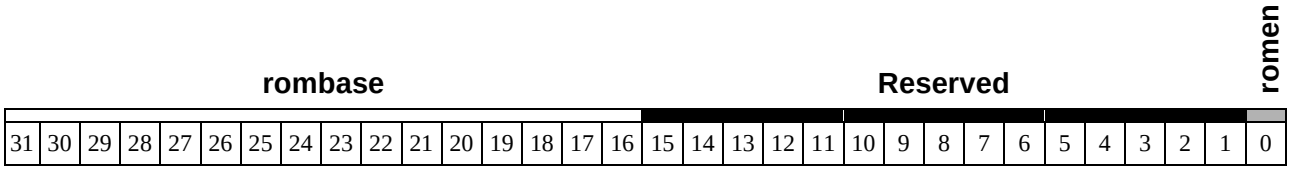
➽ *Note:* Software is required to comply with the defintions for the appropriate power state including meeting latency times. The exceptions are ‘Access to Function’ and ‘Access from Function’ which are controlled by hardware.

Reserved <31:02> Reserved. When writing to this register, the bits in these fields must be set to ‘0’.

Address 30h (CS)

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

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



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 BIOS 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 1-bit-wide serial 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.

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) or at word location 38h of the VPD ROM (128 byte 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-G100 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) or at word location 3Ah of the VPD ROM (128 byte ROM used). It indicates a subsystem ID as determined by the manufacturer of the add-in board which contains the MGA-G100 chip.

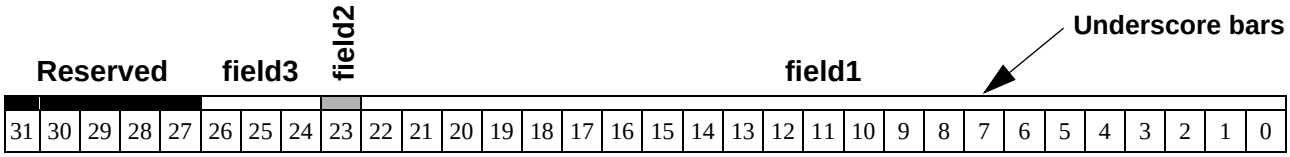
- Note: If the bios strap is ‘0’, then a 128 byte ROM is used for VPD.
- Note: This register must contain all zeros if the manufacturer of the add-in board does not have a subsystem vendor ID, or if the manufacturer does not wish to support the **SUBSYSID** register.
- Note: There may be a delay of up to 2,250 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

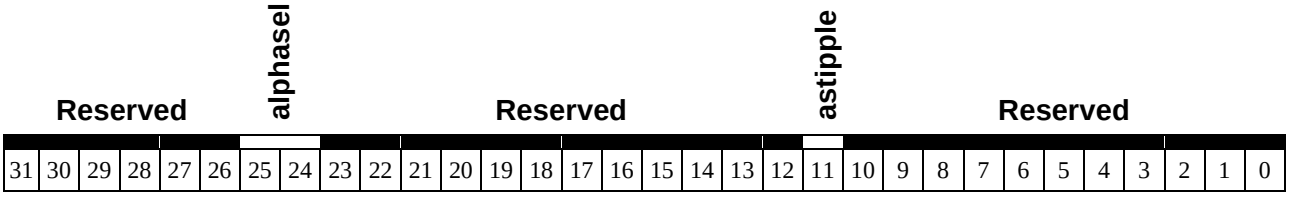
Here are some of the symbols that appear as part of a register's reset value. Most bits are reset on hard reset. Some bits are also reset on soft reset, and they are underlined when they appear in the register description headers.

n 000X 0000 0000 ???? 1101 0000 0000 0000b
(b = binary, ? = unknown, _ = reset on soft/hard reset (see above), N/A = not applicable)

Address **MGABASE1** + 2C7Ch (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



astipple <11> Alpha Stipple mode. Approximation of alpha blending using a dithring matrix.

alphasel <25:24> Alpha Select. Determine the alpha for the pixel.

- 00 alpha from texture
- 01 interpolated alpha
- 10 modulated alpha
  The resulting alpha is the product of the texture alpha and the interpolated alpha.
- 11 Reserved

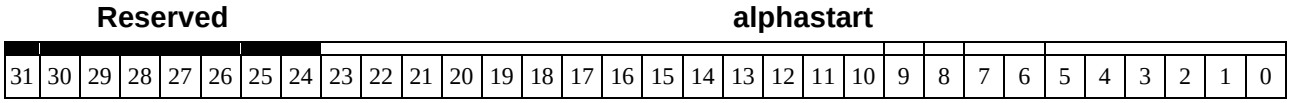
Reserved: <10:0> <23:12> <31:26>

Reserved. When writing to this register, the bits in this field must be set to ‘0’, except for <7:0> which must be set to ‘01010100b’ for compatibility with future products.

Address **MGABASE1** + 2C70h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown



alphastart <23:0> Alpha Start register. This field holds a signed 9.15 value in two's complement notation.

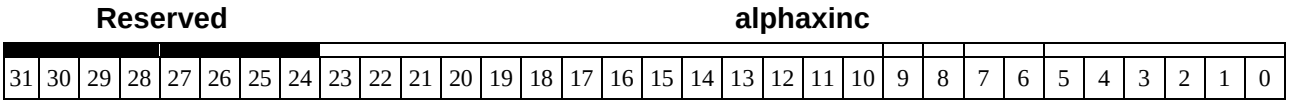
For texture primitives, the **alphastart** field is used to scan the left edge of the trapezoid for the alpha component of the source (when alpha blending is enabled). This register must be initialized with its starting alpha value.

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

Address **MGABASE1** + 2C74h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown



alphaxinc <23:0> Alpha X Increment register. This field holds a signed 9.15 value in two's complement notation.

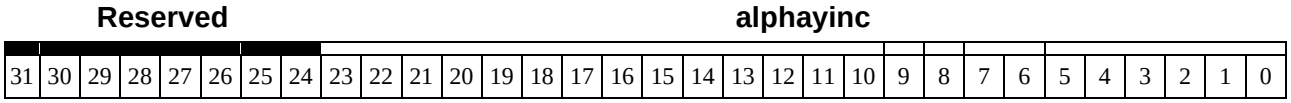
For texture primitives, the **alphaxinc** fielddf holds the alpha increment along the x axis.

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

Address **MGABASE1** + 2C78h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown



alphayinc <23:0> Alpha Y Increment register. This field holds a signed 9.15 value in two's complement notation.

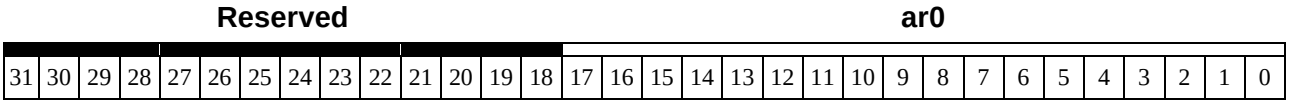
For texture primitives, the **alphayinc** field holds the alpha increment along the y axis.

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

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

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown



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**.

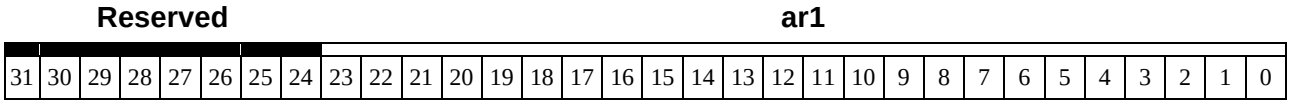
ar0 <17:0> Address register 0. The ar0 field is an 18-bit signed value in two's complement notation.
- For AUTOLINE, this register holds the x end address (in pixels). See the **XYEND register on page 4-121.**
- For LINE, it holds 2 x 'b'.
- For a filled trapezoid, it holds 'dYl'.
- For a BLIT, **ar0** 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'.

Address **MGABASE1** + 1C64h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown



➽ 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 4-121.**
- 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:

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

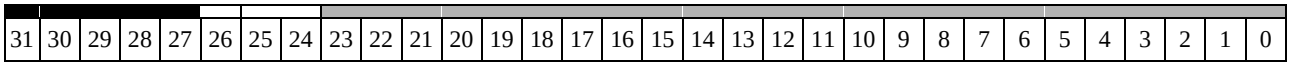
This register is **not** used for BLIT operations without scaling.

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



➽ 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 <26:24> These three bits are used as an extension to **ar3** in order to generate a 27-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 <31:27> Reserved. When writing to this register, the bits in this field must be set to '0'.

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'

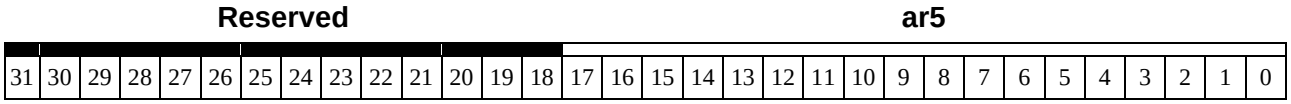
• This register is used during AUTOLINE, but doesn't need to be initialized.
• This register is not used for LINE or BLIT operations without scaling.
• For the ILOAD_SCALE, ILOAD_FILTER, ILOAD_HIQH, and ILOAD_HIQHV, it holds the error term, but it doesn't need to be initialized.

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

Address **MGABASE1** + 1C74h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown



➽ 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 4-122.** 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 4-122.** 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:

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

This register is **not** used for BLIT (without scaling) .

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

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 5-21 for 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.

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 a more efficient way to load the **CXRIGHT** and **CXLEFT** registers.

cxleft <10:0> Clipper x left boundary. See the **CXLEFT register on page 4-40.**

cxright <26:16> Clipper x right boundary. See the **CXRIGHT register on page 4-41.**

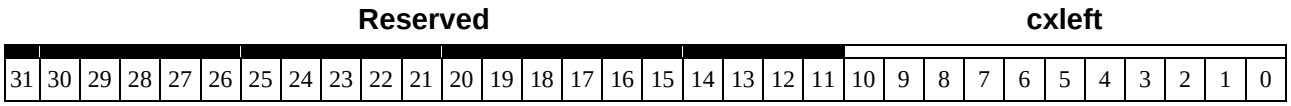
Reserved: <15:11> <31:27>

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

Address **MGABASE1** + 1CA0h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value Unknown



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 4-124)**. 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 4-124)**. 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’.

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.

The value to place in a **map_reg** field is determined as follows;

```
if (address is within the DWGRE0 range)
   map_reg? = (drawing_reg byte address >> 2)
   & 0x7F
else if (address is withing DWGREG1 range)
   map_reg? = (drawing byte address >> 2)
   & 0x7F | 0x80
else
   error, can't use indirect mapping
```

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.

The value to place in a **map_reg** field is determined as follows;

```
if (address is within the DWGRE0 range)
   map_reg? = (drawing_reg byte address >> 2)
   & 0x7F
else if (address is withing DWGREG1 range)
   map_reg? = (drawing byte address >> 2)
   & 0x7F | 0x80
else
   error, can't use indirect mapping
```

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.

The value to place in a **map_reg** field is determined as follows;

if (address is within the DWGRE0 range)

map_reg? = (drawing_reg byte address >> 2)

& 0x7F

else if (address is withing DWGREG1 range)

map_reg? = (drawing byte address >> 2)

& 0x7F | 0x80

else

error, can’t use indirect mapping

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.

The value to place in a **map_reg** field is determined as follows;

if (address is within the DWGRE0 range)

map_reg? = (drawing_reg byte address >> 2)

& 0x7F

else if (address is withing DWGREG1 range)

map_reg? = (drawing byte address >> 2)

& 0x7F | 0x80

else

error, can’t use indirect mapping

Address **MGABASE1** + 1C54h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value Unknown

dmapad

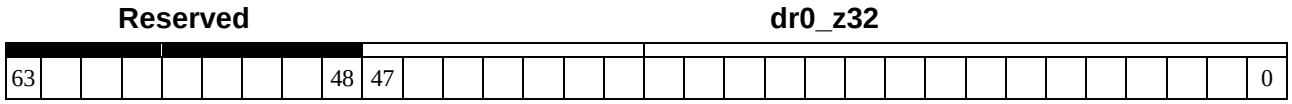
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

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** + 2C50h MGABASE1 + 2C54h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown



dr0_z32 <47:0> Data ALU 0

- For TRAP or TEXTURE_TRAP with z, the DR0_Z32 register is used to scan the left edge of the trapezoid and must be initialized with its starting z value. In this case, DR0_Z32 is signed 33.15 in two's complement notation.
- For LINE with z, the DR0_Z32 register holds the z value for the current drawn pixel and must be initialized with the starting z value. In this case, DR0_Z32 is a signed 33.15 value in two's complement notation.

Reserved <63:48> Reserved. When writing to this register, bits 63 to 48 are completely ignored.

Note:

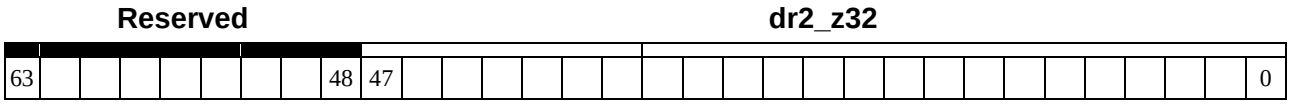
DR0_Z32<63:32> = DR0_Z32MSB<31:0>

DR0_Z32<31:0> = DR0_Z32LSB<31:0>

Address **MGABASE1** + 2C60h MGABASE1 + 2C64h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown



dr2_z32 <47:0> Data ALU 2

- For TRAP or TEXTURE_TRAP with z, the DR2_Z32 register is used to scan the left edge of the trapezoid and must be initialized with its starting z value. In this case, DR2_Z32 is signed 33.15 in two's complement notation.
- For LINE with z, the DR2_Z32 register holds the z value for the current drawn pixel and must be initialized with the starting z value. In this case, DR2_Z32 is a signed 33.15 value in two's complement notation.

Reserved <63:48> Reserved. When writing to this register, bits 63 to 48 are completely ignored.

Note:

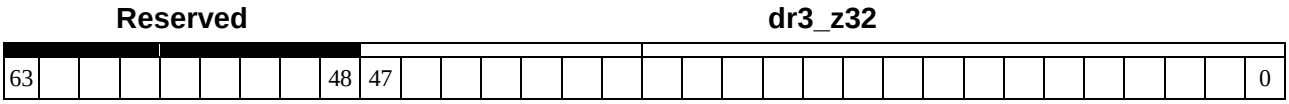
DR2_Z32<63:32> = DR2_Z32MSB<31:0>

DR2_Z32<31:0> = DR2_Z32LSB<31:0>

Address **MGABASE1** + 2C68h MGABASE1 + 2C6Ch (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown



dr3_z32 <47:0> Data ALU 3.

- For TRAP or TEXTURE_TRAP with z, the DR3_Z32 register is used to scan the left edge of the trapezoid and must be initialized with its starting z value. In this case, DR3_Z32 is signed 33.15 in two's complement notation.
- For LINE with z, the DR3_Z32 register holds the z value for the current drawn pixel and must be initialized with the starting z value. In this case, DR3_Z32 is a signed 33.15 value in two's complement notation.

Reserved <63:48> Reserved. When writing to this register, bits 63 to 48 are completely ignored.

Note:

DR3_Z32<63:32> = DR3_Z32MSB<31:0>

DR3_Z32<31:0> = DR3_Z32LSB<31:0>

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 or TEXTURE_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.

➧ Note: Bits 31 to 16 of DR0 map to bits 15 to 0 of DR0_32MSB; bits 15 to 0 of DR0 map to bits 31 to 16 of DR0_32LSB. Writing to this register clears bits 15 to 0 of DR0_32LSB.

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 or TEXTURE_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.

Note: Bits 31 to 16 of DR2 map to bits 15 to 0 of DR2_32MSB; bits 15 to 0 of DR2 map to bits 31 to 16 of DR2_32LSB. Writing to this register clears bits 15 to 0 of DR2_32LSB.

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.

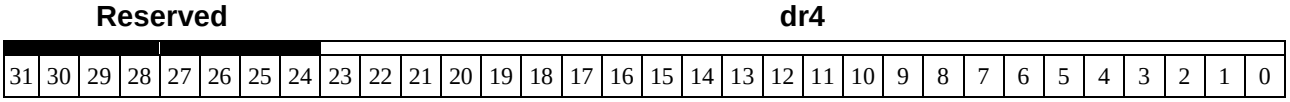
- For TRAP or TEXTURE_TRAP with z, **DR3** register holds the z increment value along the y axis. In this case, **DR3** is a signed 17.15 value in two's complement notation.
- For LINE with z, **DR3** register holds the z increment value along the diagonal axis. In this case, **DR3** is a signed 17.15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

➽ Note: Bits 31 to 16 of DR3 map to bits 15 to 0 of DR3_32MSB; bits 15 to 0 of DR3 map to bits 31 to 16 of DR3_32LSB. Writing to this register clears bits 15 to 0 of DR3_32LSB.

Address **MGABASE1** + 1CD0h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown



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

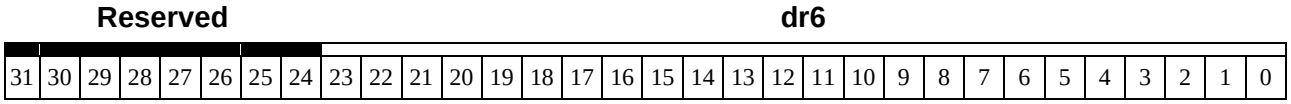
- For TRAP with z, the **DR4** register is used to scan the left edge of the trapezoid for the red color (Gouraud shading). This register must be initialized with its starting red color value.
- For TRAP_ILOAD, this register is not used, and will be corrupted.
- For LINE with z, the **DR4** register holds the current red color value for the currently drawn pixel. This register must be initialized with the starting red color.
- For TEXTURE_TRAP with texture modulation (**tmodulate** = '1', see **TEXCTL**), the **DR4** register is used to scan the left edge of the trapezoid for the red modulation factor. This register must be initialized with its starting red modulation factor value.
- For TEXTURE_TRAP using the decal feature (**tmodulate** = '0'), the **DR4** register is used to scan the left edge of the trapezoid for the red (Gouraud shaded surface) color. This register must be initialized with its starting red color value.

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



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.
- For TEXTURE_TRAP with modulation or decal, the **DR6** register holds the red increment value along the x axis.

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

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.

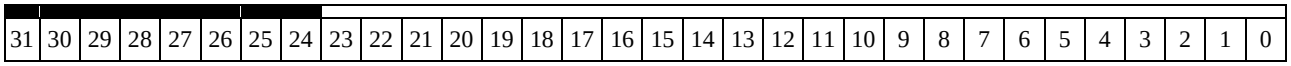
- For TRAP with z, the **DR7** register holds the red increment value along the y axis.
- For TRAP_ILOAD, this register is not used.
- For LINE with z, the **DR7** register holds the red increment value along the diagonal axis.
- For TEXTURE_TRAP with modulation or decal, the **DR7** register holds the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



dr8 <23:0> Data ALU register 8. This field holds a signed 9.15 value in two's complement notation.

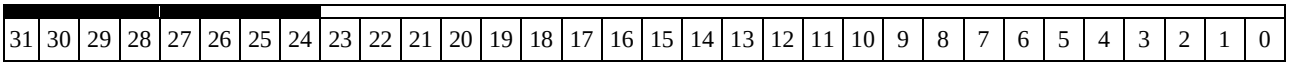
- For TRAP with z, the **DR8** register is used to scan the left edge of the trapezoid for the green color (Gouraud shading). This register must be initialized with its starting green color value.
- For TRAP_ILOAD, this register is not used, but will be corrupted.
- For LINE with z, the **DR8** register holds the current green color value for the currently drawn pixel. This register must be initialized with the starting green color.
- For TEXTURE_TRAP with texture modulation (**tmodulate** = '1', see **TEXCTL**), the **DR8** register is used to scan the left edge of the trapezoid for the green modulation factor. This register must be initialized with its starting green modulation factor value.
- For TEXTURE_TRAP using the decal feature (**tmodulate** = '0'), the **DR8** register is used to scan the left edge of the trapezoid for the green (Gouraud shaded surface) color. This register must be initialized with its starting green color value.

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



dr10 <23:0> Data ALU register 10. This field holds a signed 9.15 value in two's complement notation.

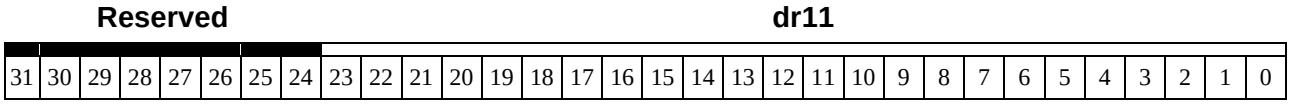
- For TRAP with z, the **DR10** register holds the green increment value along the x axis.
- For TRAP_ILOAD, this register is not used.
- For LINE with z, the **DR10** register holds the green increment value along the major axis.
- For TEXTURE_TRAP with modulation or decal, the **DR10** register holds the green increment value along the x axis.

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

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

Attributes WO, FIFO, STATIC, DWORD

Reset Value Unknown



dr11 <23:0> Data ALU register 11. This field holds a signed 9.15 value in two's complement notation.

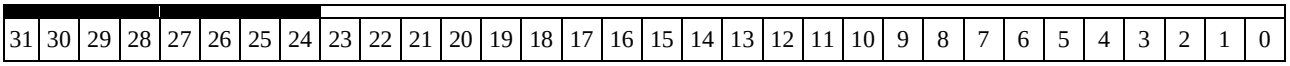
- For TRAP with z, the **DR11** register holds the green increment value along the y axis.
- For TRAP_ILOAD, this register is not used.
- For LINE with z, the **DR11** register holds the green increment value along the diagonal axis.
- For TEXTURE_TRAP with modulation or decal, the **DR11** register holds the green increment value along the y 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



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

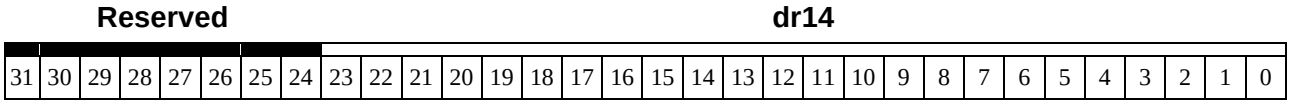
- For TRAP with z, the **DR12** register is used to scan the left edge of the trapezoid for the blue color (Gouraud shading). This register must be initialized with its starting blue color value.
- For TRAP_ILOAD, this register is not used, but will be corrupted.
- For LINE with z, the **DR12** register holds the blue color value for the currently drawn pixel. This register must be initialized with the starting blue color.
- For TEXTURE_TRAP with texture modulation (**tmodulate** = '1', see **TEXCTL**), the **DR12** register is used to scan the left edge of the trapezoid for the blue modulation factor. This register must be initialized with its starting blue modulation factor value.
- For TEXTURE_TRAP using the decal feature (**tmodulate** = '0'), the **DR12** register is used to scan the left edge of the trapezoid for the blue (Gouraud shaded surface) color. This register must be initialized with its starting blue color value.

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

Address **MGABASE1** + 1CF8h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value Unknown



dr14 <23:0> Data ALU register 14. This field holds a signed 9.15 value in two's complement notation.

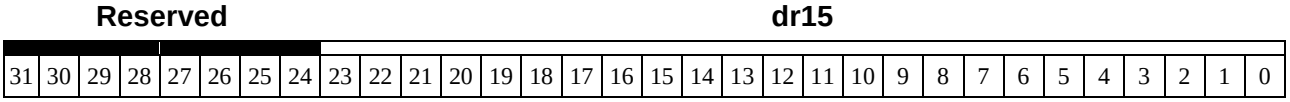
- For TRAP with z, the **DR14** register holds the blue increment value along the x axis.
- For TRAP_ILOAD, this register is not used.
- For LINE with z, the **DR14** register holds the blue increment value along the major axis.
- For TEXTURE_TRAP with modulation or decal, the **DR14** register holds the blue increment value along the x 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



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.
- For TEXTURE_TRAP with modulation or decal, the **DR15** register holds the blue increment value along the y 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

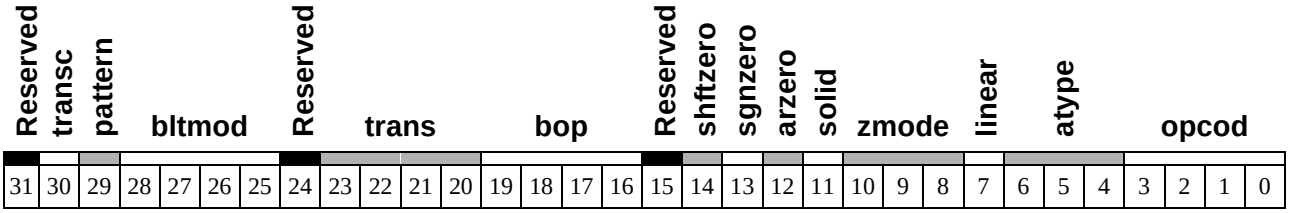
lutentry N <31:0> These 16 registers are a lookup table that can be used in conjunction with the **DMAMAP** registers. Writing to these locations address the register that is programmed in the Nth byte of the **DMAMAP**. This indirect write register provides a means to access non-sequential drawing registers 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 0000b



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
	Texture mapping	‘0110’	TEXTURE_TRAP
Blit	RAM -> RAM	‘1000’	BITBLT
	HOST -> RAM	‘1001’	ILOAD
	HOST -> RAM scale	‘1101’	ILOAD_SCALE
	HOST -> RAM scale, filter	‘1111’	ILOAD_FILTER
	HOST -> RAM scale, high-quality filter	‘0111’	ILOAD_HIQH
	HOST -> RAM horizontal and vertical scale, high-quality filter	‘1110’	ILOAD_HIQHV
	Reserved	‘1010’	
	”	‘1011’	
	”	‘1100’	

atype <6:4>

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

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

(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) 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		
Value	***Mnemonic***	***Pixel Update***
'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 4-98).

- 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 unpredictable results.

arzero <12>

AR register at zero. The **arzero** field provides an alternate way to set certain **AR** registers (see descriptions starting on page 4-31).

- 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 4-88)** is added to **ydst** (see **YDST on page 4-124)**

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

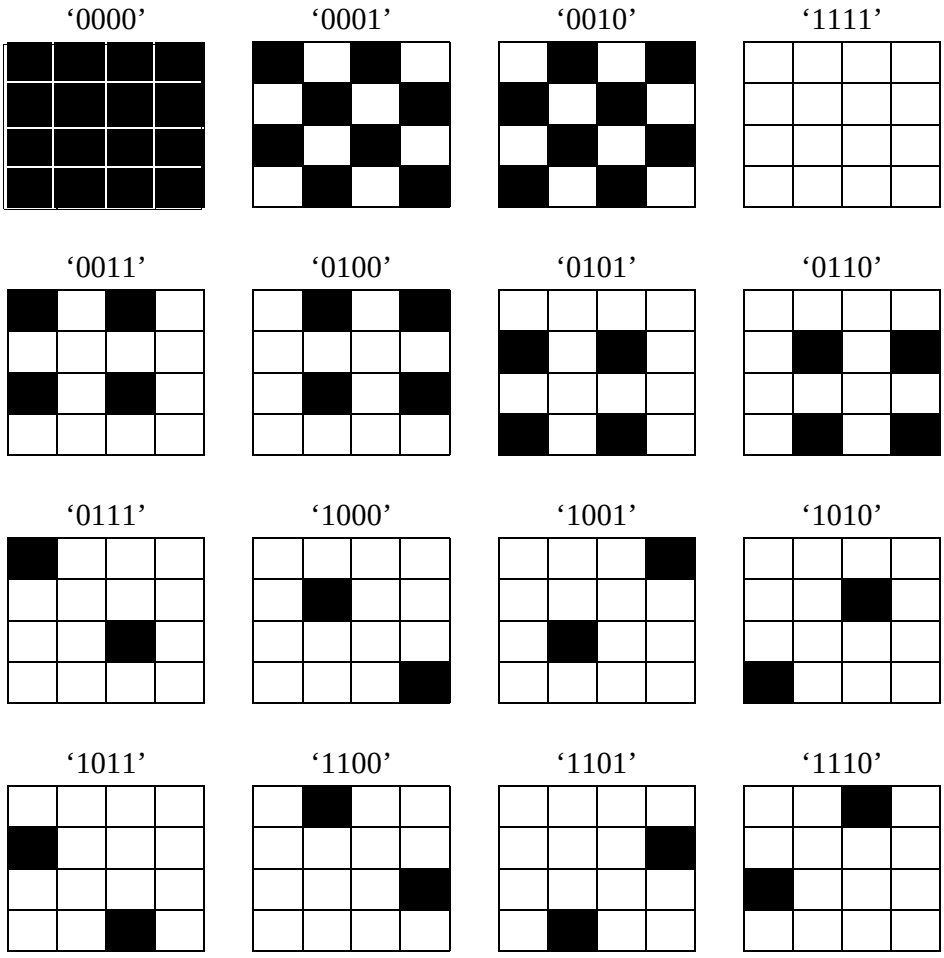
bop <19:16>

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 TRAP_ILOAD operations.

Refer to the subsections contained in 'Drawing in Power Graphic Mode' on page 5-27 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’.

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 5-21 for 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>.

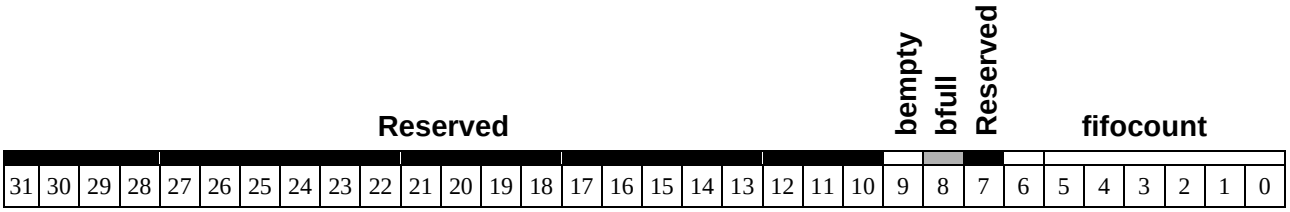
bltckey <31:0> Blit color key. This field specifies the value of the color that is defined as the ‘transparent’ color. Planes that are not used must be set to ‘0’. Refer to the description of the **transc** field of **DWGCTL** for the transparency equation

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 0100 0000b



fifocount <6:0> 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 64.

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**<6>.

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).

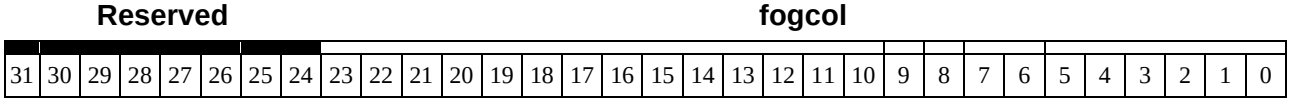
Reserved: <7> <31:10>

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

Address **MGABASE1** + 1CF4h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown



fogcol <23:0> Fog Color. The **fogcol** field is used by the color that will be blended (using the fog blending factor) with the current rasterized fragment's color when fogging is enabled.

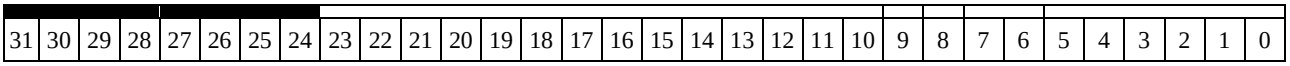
fogcol is in true color (RGB 888) format.

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

Address **MGABASE1** + 1CC4h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown



fogstart <23:0> The **fogstart** register. This field holds a signed 9.15 value in two's complement notation.

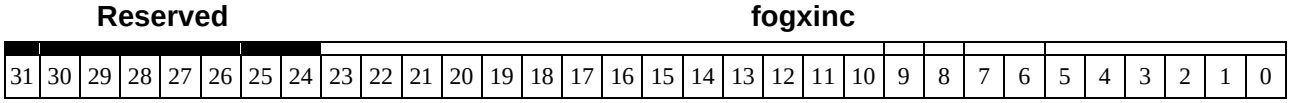
For texture primitives, the **fogstart** field is used to scan the left edge of the trapezoid for the fog blending factor (when fogging is enabled). This register must be initialized with its starting for factor value.

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

Address **MGABASE1** + 1CD4h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown



fogxinc <23:0> The Fog X Increment register. This field holds a signed 9.15 value in two's complement notation.

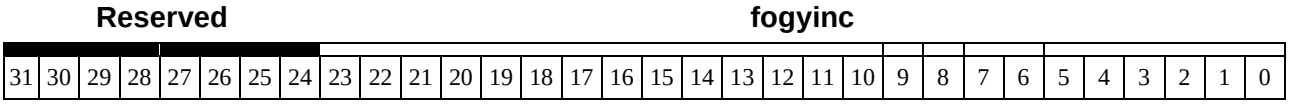
For texture primitives, the **fogxinc** field holds the fog blending factor increment along the X axis.

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

Address **MGABASE1** + 1CE4h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown



fogyinc <23:0> The Fog Y Increment register. This field holds a signed 9.15 value in two's complement notation.

For texture primitives, the **fogyinc** field holds the fog blending factor increment along the y axis.

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

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 a more efficient way to load the **FXRIGHT** and **FXLEFT** registers.

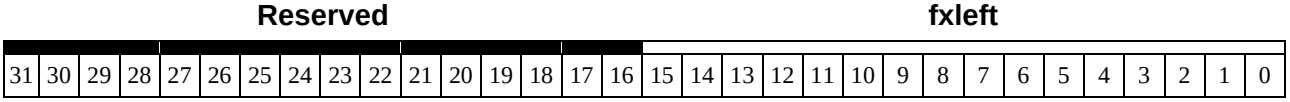
fxleft <15:0> Filled object x left coordinate. Refer to the **FXLEFT** register for a detailed description.

fxright <31:16> Filled object x right coordinate. See the **FXRIGHT register on page 4-78.**

Address **MGABASE1** + 1CA8h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown



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.

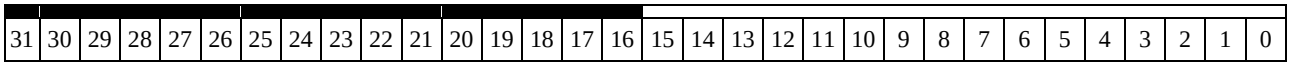
- The **fxleft** field is not used for line drawing.
- During filled trapezoid drawing, **fxleft** is updated during the left edge scan.
- During a BLIT operation, **fxleft** is static, and specifies the left pixel boundary of the area being written to.

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



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.

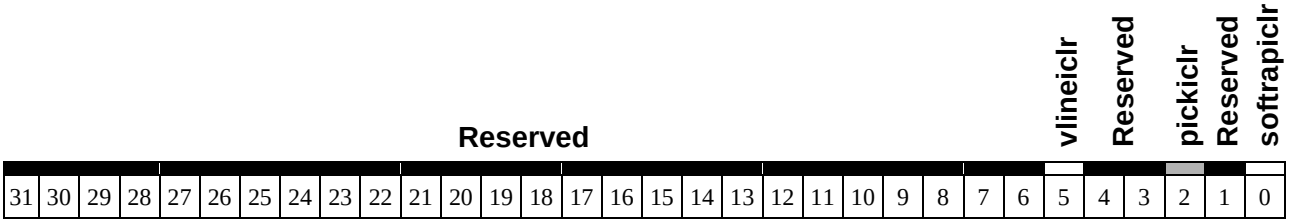
- The **fxright** field is not used for line drawing.
- During filled trapezoid drawing, **fxright** is updated during the right edge scan.
- During a BLIT operation, **fxright** is static, and specifies the right pixel boundary of the area being written to.

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



softrapiclr <0> Soft trap interrupt clear. When a ‘1’ is written to this bit, the soft trap interrupt pending flag is cleared.

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.

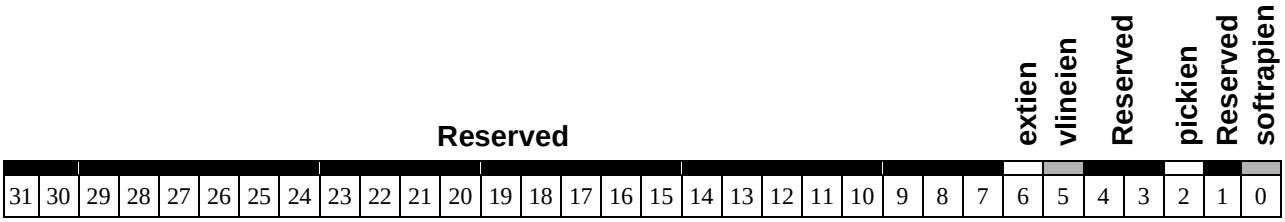
Reserved: <1> <4:3> <31:6>

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

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

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

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



softrapien <0> Soft trap interrupt enable. When this field is set to '1', the PCI interrupt is enabled on the PINTA/ line when the **SOFTRAP** register is written to.

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.

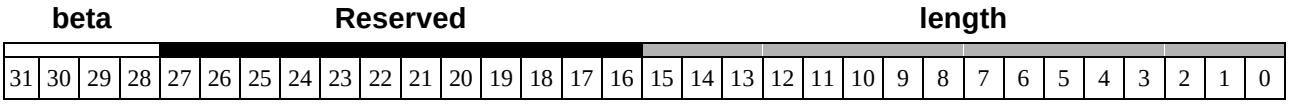
Reserved: <1> <4:3> <31:7>

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

Address **MGABASE1** + 1C5Ch (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



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.

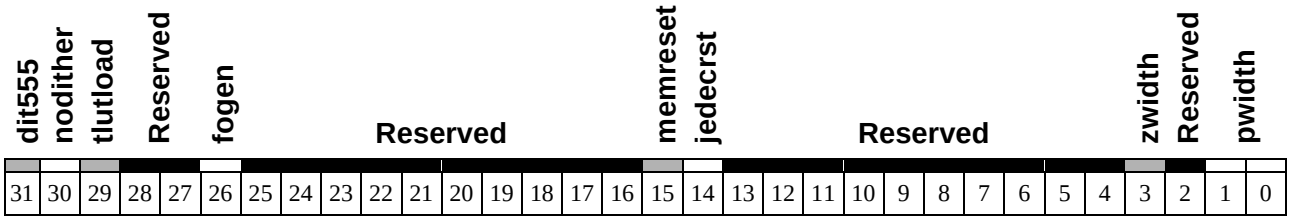
beta <31:28> Beta factor. This field is used to drive the vertical scaling in ILOAD_HIQHV (it is not used for other opcodes). The **beta** field represents the four least significant bits of a value between 1 and 16 (16 is 0000b), which represents a beta factor of 1/16 through
16/16.

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 0000b



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

zwidth <3> Z depth width. Specifies the size of Z values :

zwidth		
Value	*Mnemonic*	*Mode*
'0'	ZW16	16 bit Z
'1'	ZW32	32 bit Z

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).

➧ *Note:* Always program this to '1'.

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 5-24 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.

fogen <26> fogenable. Fogging can be performed on any texture trap operation.

tlutload <29> Texture LUT load. When this bit is set to '1' during an ILOAD or BITBLT operation, the destination becomes the texture LUT rather than the frame buffer.

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

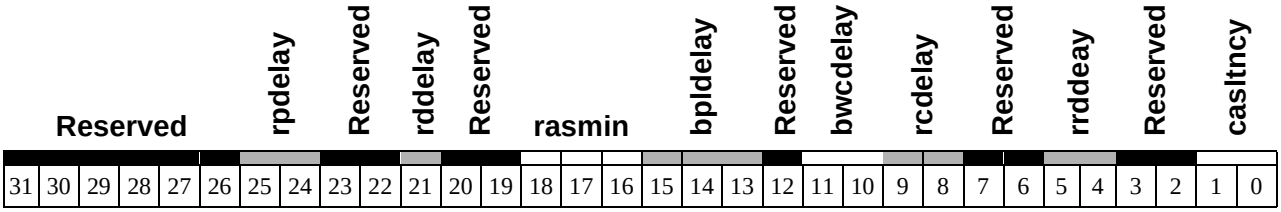
Reserved : <2> <13:4> <25:16> <28:27>

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 0001b



casltncy <1:0> CAS latency. This bit must be programmed prior to the memory power-up sequence. Refer to 'Power Up Sequence' on page 5-23 for more details.

casltncy	*CAS Latency (fmclk)*
'00'	2
'01'	3
'10'	4
'11'	Reserved

rrddelay <5:4> Minimum RAS to RAS delay. This field MUST be loaded before attempting to read or write to the frame buffer.

rrddelay	*RAS to RAS Delay (fmclk)*
'00'	1 cycle
'01'	2cycles
'10'	3 cycles
'11'	4 cycles

rcdelay <9:8> RAS to CAS delay. This bit selects a RAS to CAS delay, as shown below:

rcdelay	*RAS to CAS Delay (fmclk)*
'00'	2
'01'	3
'10'	4
'11'	Reserved

bwcdelay <11:10> Block write cycle delay

bwcdelay	*Block write cycle Delay (fmclk)*
00	1
01	2
10	3
11	Reserved

bpldelay <15:13> Block write to precharge delay

bpldelay	*Block write to precharge Delay (fmclk)*	bpldelay	*Block write to precharge Delay (fmclk)*
000	1	100	5
001	2	101	Reserved
010	3	110	Reserved
011	4	111	Reserved

rasmin <18:16> RAS minimum active time. The valid values are shown below:

rasmin	*RAS Minimum (fmclk)*	rasmin	*RAS Minimum (fmclk)*
'000'	3	'100'	7
'001'	4	'101'	8
010	5	'110'	Reserved
'011'	6	'111'	Reserved

rddelay <21> Minimum Read to precharge delay. This field MUST be loaded before attempting to read or write to frame buffer

rddelay	*Minimum read to precharege delay (fmclk)*
'0'	n cycles
'1'	n + (CL - 2) cycles

Where

n = # of data to be read

CL = cas latency (2,3,4,5)

rpdelay <25:24> Minimum row precharge. The valid values are shown below:

rpdelay	*Minimum row precharge (fmclk)*
'00'	2
'01'	3
'10'	4
'11'	5

Note: Row cycle time is the addition of **rasmin** the and **rpdelay**.

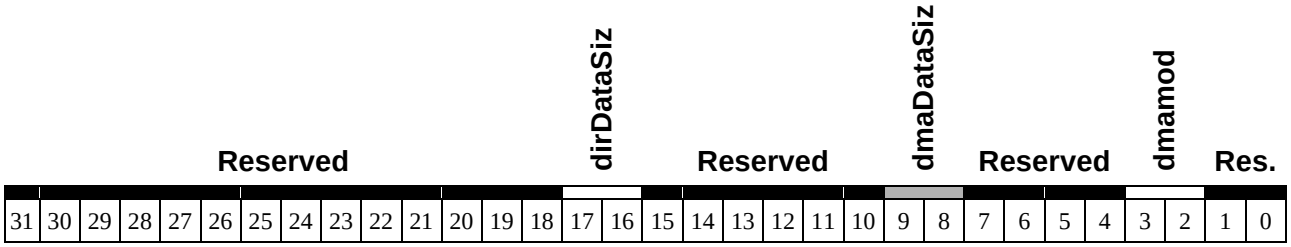
Reserved **<31:26> <23:22> < 20:19> <15> <12> <9> <7:1>**

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.

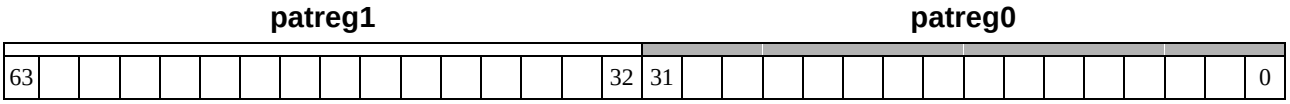
Reserved: <1:0> <7:4> <15:10> <31:18>

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

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

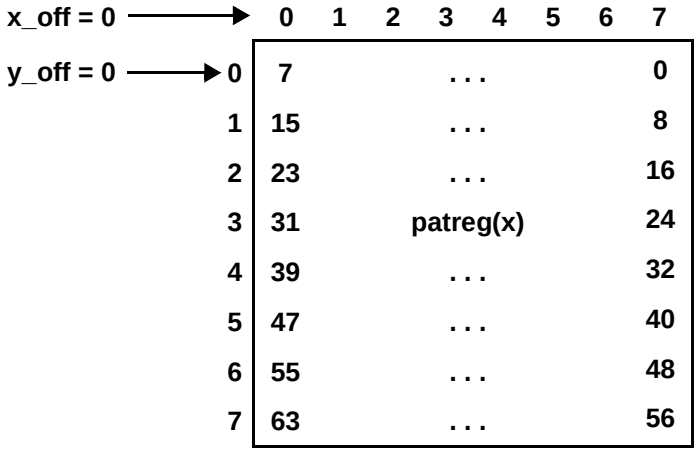
Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown



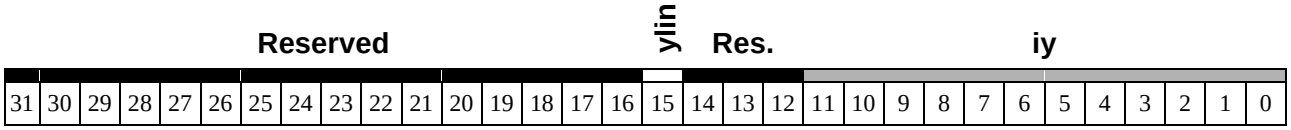
patreg <63:0> Pattern register. The **PAT** registers are not physical registers. They simply provide an alternate way to load the **SRC** registers with a Windows format 8 x 8 pattern.

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 4-98.**

Address **MGABASE1** + 1C8Ch (MEM)
Attributes WO, FIFO, STATIC, DWORD
Reset Value Unknown



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 4-124)** between two destination lines. The **iy** field is also used as the multiplication factor for linearizing the **ydst** register.

All pitches that are multiple of 32 to 2K inclusively are supported for the linearization in hardware.

Block mode restictions:

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 5-37. See page 4-64 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’.

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

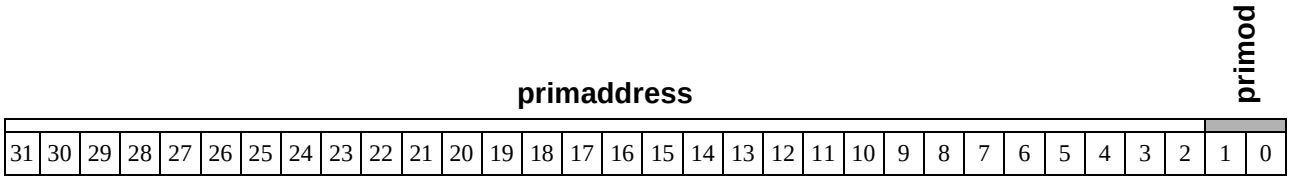
The bits from the **plnwrmsk**<31:0> register are output on the MDQ<31:0> signal and also on MDQ<63:32>. In 8 and 16 bit/pixel configurations, all bits in **plnwrmsk**<31:0> are used, so the mask information must be replicated on all bytes. In 24 bits/pixel, the plane masking feature is limited to the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

Refer to ‘Pixel Format’ on page 5-21 for the definition of the slice in each mode.

Address **MGABASE1** + 1E58h (MEM)

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

Reset Value Unknown



primod <1:0> Primary Pseudo-DMA mode. This static field indicates the Pseudo-DMA mode to be used to transfer data from system memory to the MGA in mastering mode through the primary DMA channel.

primod	*DMA Transfer Mode*
'00'	DMA General Purpose Write
'01'	DMA Blit Write
'10'	DMA Vector Write
'11'	Reserved

primaddress <31:2> Primary current address. This field indicates the address to be used to access the primary DMA channel in the system memory when the MGA-G100 is performing bus mastering.

The start address value of the primary display list must be written to this register before **PRIMEND** is written to.

The primaddress field is increased by one every time the MGA-G100 terminates a read access at primaddress in the system memory.

If, when increased, **primaddress** becomes equal to **primend**, the primary channel is empty. Bus mastering stops, and **endprdmasts** is set (refer to the **STATUS** register description).

Refer to 'Programming Bus Mastering for DMA Transfers' on page 5-14 for more details.

Address **MGABASE1** + 1E5Ch (MEM)

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

Reset Value Unknown

primend																												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

primend <31:2> Primary end address. The **primend** field holds the end address + 1 of the primary display list in the system memory, when doing bus mastering.

Writing to this field will ***start*** the transfers from the primary DMA channel in the MGA-G100. This means that the **PRIMEND** register must always be written to last.

The MGA-G100 generates an end status (see **endprdmasts** in the **STATUS** register) every time **primaddress** equals **primend** and **secaddress** equals **secend**, or when a soft trap interrupt occurs.

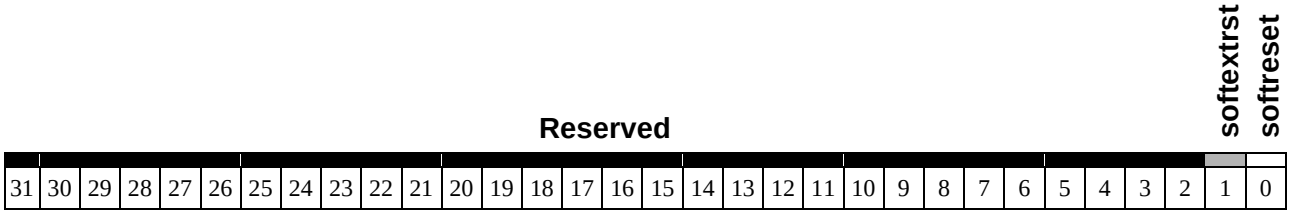
Refer to ‘Programming Bus Mastering for DMA Transfers’ on page 5-14 for more details.

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

Address **MGABASE1** + 1E40h (MEM)

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

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



softreset <0> Soft reset. When set to ‘1’, resets all software-resettable bits. This has the effect of flushing the BFIFO, and the direct access read cache, 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.

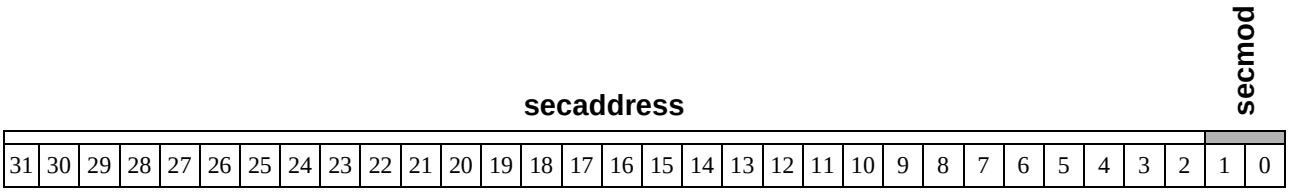
Refer to Section 5.3.3 on page 5-24 for instructions on when to use this field.

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

softextrst <1> External reset by software. When set to ‘1’, this forces low the external reset pin (EXTSTN). The external reset is maintained until this bit is reset to ‘0’.

Reserved <31:2> Reserved. When writing to this register, the bits in this field must be set to ‘0’. Reading will give ‘0’s.

Address **MGABASE1** + 2C40h (MEM)
Attributes R/W, FIFO, DYNAMIC, DWORD
Reset Value Unknown



secmod <1:0> Secondary Pseudo-DMA mode. This static field indicates the Pseudo-DMA mode to be used to transfer data from system memory to the MGA in mastering mode through the secondary DMA channel.

secmod	*DMA Transfer Mode*
'00'	DMA General Purpose Write
'01'	DMA Blit Write
'10'	DMA Vector Write
'11'	Reserved

secaddress <31:2> Secondary DMA Address. This field indicates the address to be used to access the secondary DMA channel in the system memory when the MGA-G100 is performing bus mastering.

The start address value of the secondary DMA channel must be written to this register before **SECEND** is written to.

The field secaddress is increased by one every time the MGA-G100 terminates a read access at **secaddress** in the system memory.

If, when incremented, **secaddress** becomes equal to **secend**, the secondary channel is empty. Bus mastering then continues, using the primary channel.

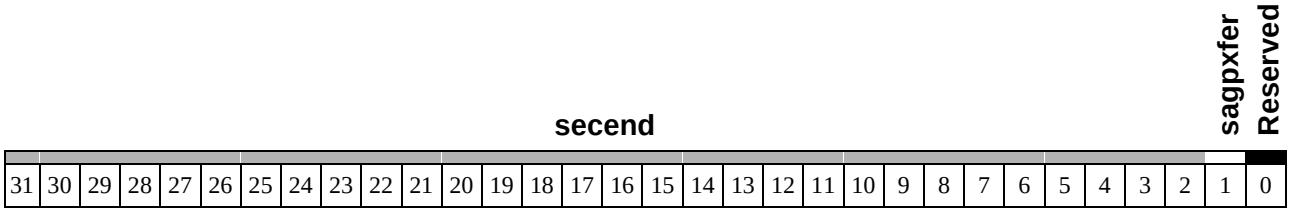
The data that is written to the **SOFTRAP** register will also be loaded into the **secaddress** field.

Note: It is not possible to write to this register directly. Write access must ***absolutely*** be performed through mastering mode. That is, a primary display list ***must be programmed***. Refer to 'Programming Bus Mastering for DMA Transfers' on page 5-14 for more details.

Address **MGABASE1** + 2C44h (MEM)

Attributes R/W, FIFO, STATIC, DWORD

Reset Value Unknown



secend <31:2> Secondary End address. The **secend** field holds the end address + 1 of the secondary DMA channel in the system memory, when doing bus mastering.

Writing to this field will ***start*** the secondary DMA transfers by the MGA-G100 using bus mastering. The **SECEND** register must always be written to after **SECADDRESS**.

The MGA-G100 generates an end status (see **endprdmasts** in the **STATUS** register) every time **primaddress** equals **primend** and **secaddress** equals **secend**, or when a soft trap interrupt occurs.

➧ Note: It is not possible to write to this register directly. Write access must ***absolutely*** be performed through mastering mode. That is, a primary display list ***must be programmed***. Refer to 'Programming Bus Mastering for DMA Transfers' on page 5-14 for more details.

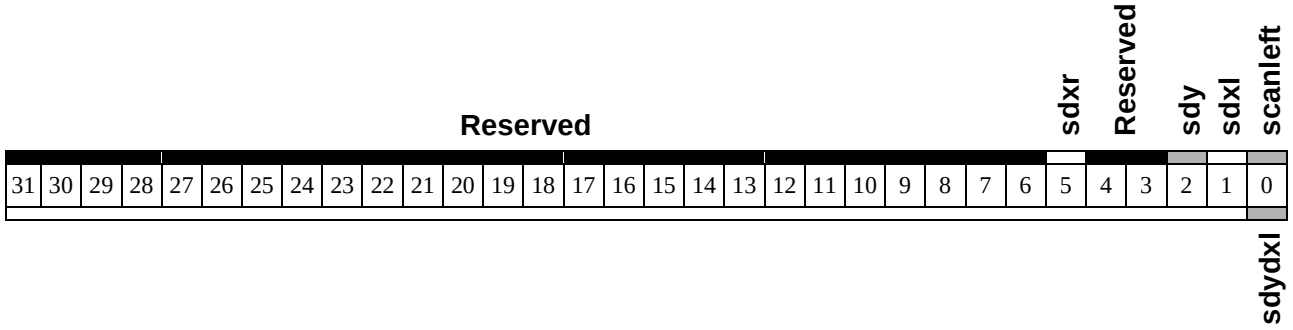
sagpxfer <1> Secondary AGP Transfer. This bit indicates whether the secondry DMA transfer will use AGP protocol or PCI.

- 0: transfer will be done as PCI
- 1: transfer will be done as AGP

➧ Note: For the transfer to use AGP protocol, the **agp_enable** and **sba_enable** must be '1' and **data_rate<2:0>** in the **AGP_CMD** register must be '001b'.

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

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



➽ Note: Witing 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 4-63)**. 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.

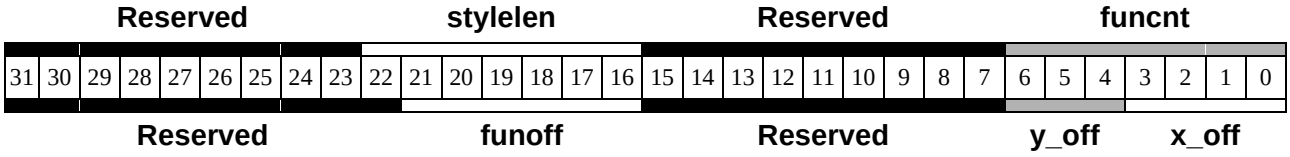
- 0: delta x is positive
- 1: delta x is negative

Reserved: <4:3> <31:6>
Reserved. When writing to this register, the bits in these fields must be set to '0'.

Address **MGABASE1** + 1C50h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown



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 4-98), so its value is the number of bits in the complete pattern minus one. For 3D vectors, this field must be initialized to 0.

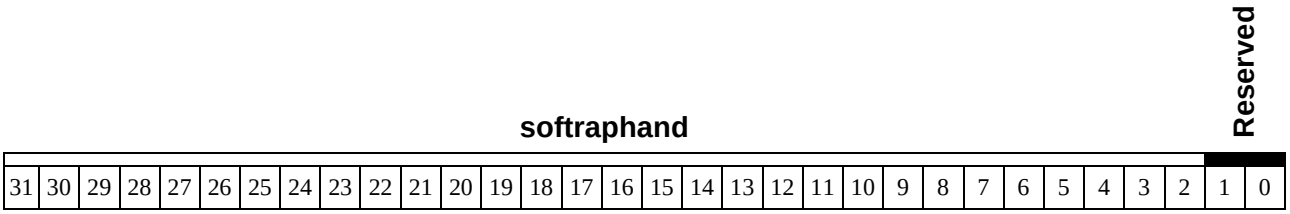
Reserved: <15:7> <31:23/22>

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

Address **MGABASE1** + 2C48h (MEM)

Attributes R/W, FIFO, DYNAMIC, DWORD

Reset Value Unknown



softraphand <31:2> Soft trap handle. When this field is written, a soft trap interrupt is generated, and the primary DMA channel is stopped (the **softrapen** and **endprdmasts** fields of **STATUS** are set to '1'). To restart the primary DMA channel, **PRIMEND** must be written. It is not permitted to access the soft trap handle within a seconday DMA.

Data written to the **softraphand** field is actually loaded into the **secaddress** field of **SECADDRESS** (which is temporarily borrowed for use by this function). This same mechanism could be used by software to transfer information to the interrupt handler.

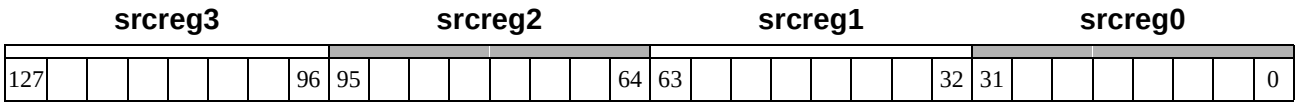
Note: It is not possible to write to this register directly. Write access must ***absolutely*** be performed through mastering mode. That is, a primary display list ***must be programmed***. Refer to 'Programming Bus Mastering for DMA Transfers' on page 5-14 for more details.

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

Address **MGABASE1** + 1C30h, + 1C34h, + 1C38h, + 1C3Ch (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown



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 5-32 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 5-38 for more details.

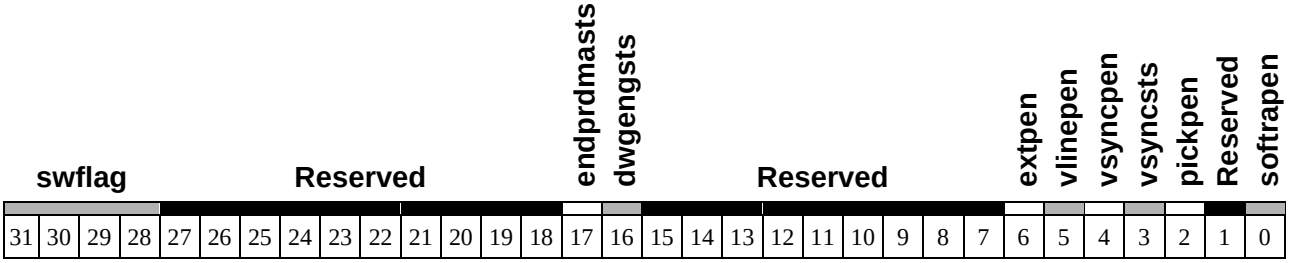
For all BLIT operations, and for TRAP or LINE using depth mode, the source register is used internally for intermediate data.

A write to the **PAT** registers (see page 4-87) 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



softrapen <0> Soft trap interrupt pending. When set to '1', this field indicates that the MGA-G100 has stopped reading through the primary channel.

This field is set to '1' when the **SOFTRAP** register is written. This field is cleared through the **softrapiclr** field (see **ICLEAR** on page 4-79) or upon soft or hard reset.

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 4-79)** 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 4-169.** This bit is cleared through the **vlineiclr** bit (see **ICLEAR on page 4-79)** 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).

endprdmasts <17> End of primary DMA channel status. When set to ‘1’, this bit indicates that the MGA-G100 has completed its DMA transfers (**primaddress** equals **primend** and **secaddress** equals **secend**), or when a soft trap interrupt occurs. Writing to **PRIMEND** will restart the DMA transfers, and **endprdmasts** will be ‘0’.

swflag <31:28> Software Flags. Read/Write bits for software protocols applications.

Reserved: <1> <15:7> <27:18>

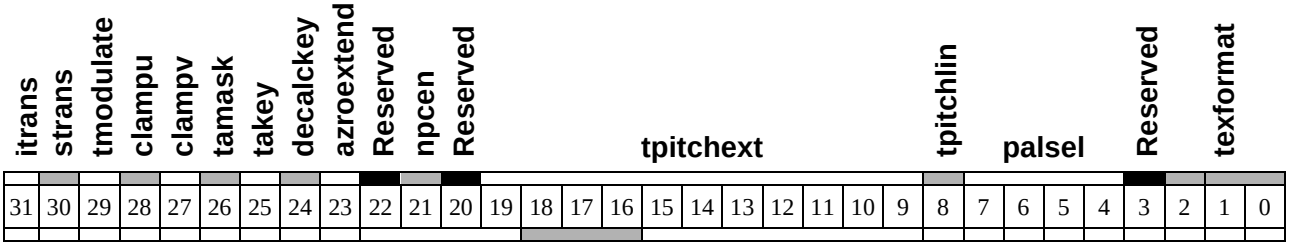
Reserved. When writing to this register, the bits in these fields must be set to ‘0’. Reading will give ‘0’s.

➾ Note: 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).

Address **MGABASE1** + 2C30h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value Unknown



texformat <2:0> Texel format. Specifies the texel format for the texture.

texformat		*Format*	
Value	*Mnemonic*		
'000'	TW4	4 bits/texel	4-bit LUT index (1)
'001'	TW8	8 bits/texel	8-bit LUT index
'010'	TW15	1:5:5:5	(A:R:G:B)
'011'	TW16	5:6:5	(R:G:B)
'100'	TW12	4:4:4:4	(A:R:G:B)

(1) LUT selected by **palsel**.

palsel <7:4> Palette select. This field selects which of the 16 palettes to use for TW4.

tpitchlin <8> Texture Pitch Linear. Indicates if the value in the **tpitch/ tpitchext** field is programmed with its linear value or not

- 0: bit <18:16> of register **TEXCTL** are used to set the pitch.
  pitch = 8 << **TEXCTL** (18:16)
- 1: bit <19:9> of register **TEXCTL** are used to set the pitch
  pitch = **TEXCTL** (19:9)

tpitchext <19:9> Texture Pitch Extended. If the **tpitchlin** bit is set to '1', the **tpitchext** field is programmed with the pitch of the texture which can vary from 1 to 2048 where a value of ' 0' represents a pitch of 2048.

tpitch <18:16> When the **tpitchlin** bit is set to '0', the **tpitch** is set according to the table below

Pitch	tpicth
8	‘000’
16	‘001’
32	‘010’
64	‘011’
128	‘100’
256	‘101’
512	‘110’
1024	‘111’

npcen <21> Non-perspective correction enable. Writing a ‘1’ to this bit disables the perspective correction normally done in the texture mapping engine.

azeroextend <23> Alpha Zero Extend. Widen the alpha mask and key (**tamask** and **takey)** up to the actual texel alpha component size.

0: Replicate tamask and takey
1: Zero extend

decalckey <24> Decal with color key. This bit indicates whether texel color keying or texel alpha keying will control the decal feature.

- 0: Alpha keying controls the decal feature. The surface transparency feature is available (see **strans**, below), based on alpha keying. Color keying is used to prevent frame buffer updates for transparent texels (independent of **strans**).
- 1: Alpha keying must be disabled. Color keying controls the decal feature (it does not automatically prevent frame buffer updates for transparent texels). The surface transparency feature is available (see **strans**), based on color keying.

takey <25> Texture alpha key. This field indicates which polarity is defined as transparent.

tamask <26> Texture alpha mask. This field enables alpha transparency. To disable transparency (that is, to make the texture opaque), set **takey** to ‘1’ and **tamask** to ‘0’.

clampv <27> Clamp mode enable for V. This bit specifies if the texture is clamped or repeated over the surface.

- 0: repeat
- 1: clamp

clampu <28> Clamp mode enable for U. This bit specifies if the texture is clamped or repeated over the surface.

- 0: repeat
- 1: clamp

tmodulate <29> Texture modulate enable. This bit enables the multiplication of the texture with the I ALU on a color-by-color basis. When modulation is disabled, decal mode is used. The decal function selects between the texel and the surface color (I ALU), based on the **decalckey** field and the transparency information from the texture (the ‘ctransp’ and ‘atransp’ values - see **itrans**).

strans <30> Surface transparency enable. When 1, this bit enables control of the frame buffer update on a per-pixel basis. Only opaque pixels are updated (when **itrans** = 0). Transparency is determined by either alpha keying or color keying, according to the setting of the **decalckey** field.

itrans <31> Invert transparency enable. When 1, the transparency decision is inverted to allow two-pass algorithms when **strans** is active.

Only the following field value combinations are allowed (all others are reserved):

tmodulate	strans	itrans	decalckey	*ctransp*	*atransp*	*Pixel Result*
'0'	'0'	'0'	'0'	'0'	'0'	texel
'0'	'0'	'0'	'0'	'0'	'1'	I
'0'	'0'	'0'	'0'	'1'	X	Not written
'0'	'0'	'0'	'1'	'0'	'0'	texel
'0'	'0'	'0'	'1'	'1'	'0'	I
'0'	'1'	'0'	'0'	'0'	'0'	texel
'0'	'1'	'0'	'0'	'0'	'1'	Not written
'0'	'1'	'0'	'0'	'1'	X	Not written
'0'	'1'	'0'	'1'	'0'	'0'	texel
'0'	'1'	'0'	'1'	'1'	'0'	Not written
'0'	'1'	'1'	'0'	'0'	'0'	Not written
'0'	'1'	'1'	'0'	'0'	'1'	I
'0'	'1'	'1'	'0'	'1'	X	Not written
'0'	'1'	'1'	'1'	'0'	'0'	Not written
'0'	'1'	'1'	'1'	'1'	'0'	I
'1'	'0'	'0'	'0'	'0'	'0'	texel * I
'1'	'0'	'0'	'0'	'1'	'0'	Not written
'1'	'1'	'0'	'0'	'0'	'0'	texel * I
'1'	'1'	'0'	'0'	'0'	'1'	Not written
'1'	'1'	'0'	'0'	'1'	X	Not written
'1'	'1'	'0'	'1'	'0'	'0'	texel * I
'1'	'1'	'0'	'1'	'1'	'0'	Not written

In the preceding table, 'ctransp' indicates the color keying result as defined by:

```
if ( texel & tkmask == tckey )
    ctransp = 1
else
    ctransp = 0
```

In the preceding table, 'atransp' indicates the alpha keying result as defined by:

alpha = texel<15> when texformat = TW15 or

alpha = texel<15:12> when texformat = TW12

```
if ( alpha & tamask == takey )
    atransp = 1
else
    atransp = 0
```

The **tkmask** and **tckey** fields are located in the **TEXTRANS** register.

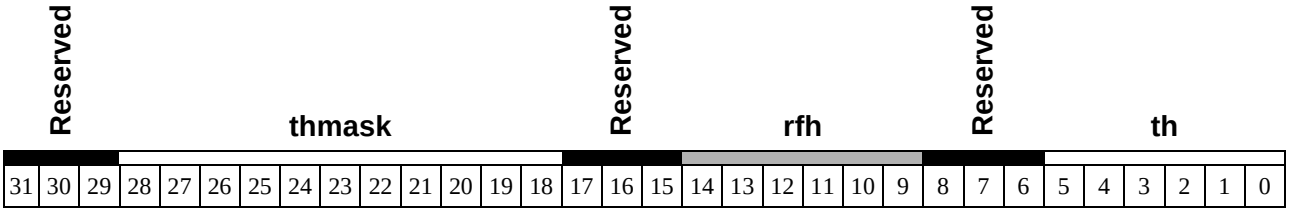
Reserved: <3> <20> <22>

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

Address **MGABASE1** + 2C2Ch (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value Unknown



th <5:0>

Represents log2 of the texture height combined with an adjustment factor. This field holds a 6-bit signed value in two's complement notation, set as follows:

th = log2(texture height) +(4 - t_fractional_bits + q_fractional_bits)

Typically, **th** = log2(texture height) + (4 - 20 + 16).

rfh <14:9>

Height round-up factor. This factor is used to improve the generated image quality by assuring coherence in texel choice. This field holds a 6-bit signed value in two's complement notation, usually set to 8 - log2 (texture height) - (q_fractional_bits - 16).

thmask <28:18>

Height mask. Determines the usable height of the texture (select on which texel the repeat or clamping will be performed). This field holds a 11 bit unsigned value, usually set to (texture height)-1.

Note: The repeat mode will not work properly if **thmask** is set to a value that is not a power of two minus one.

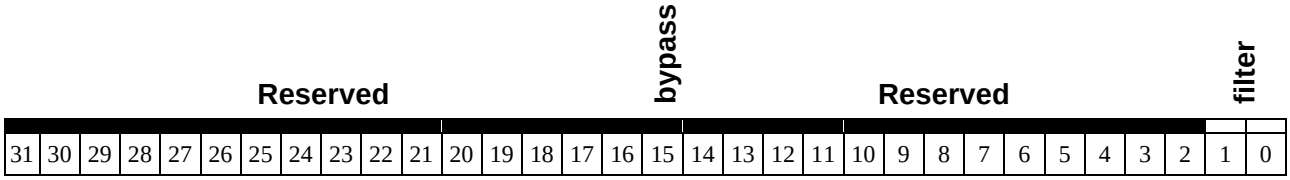
Reserved: <8:6> <17:15> <31:29>

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

Address **MGABASE1** + 2C58h (MEM)

Attributes WO, BYTE/WORD/DWORD, STATIC

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



filter <1:0> Filtering Mode

Table 4-1:

filter	mnemonic	filter mode
00	NRST	Nearest
01		RSVD
10	BILIN	Bi-linear
11	CNST	Constant Bi-linear (.25)

bypass <15> Bypass texture cache lookup. Fetch all required texels, ignoring the cache index table.

Reserved **<31:16> <14:2>**

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

Address **MGABASE1** + 2C24h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value Unknown

Reserved								torg																							
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

torg <23:0> Texture origin. The **torg** field holds a 24-bit unsigned value which provides an offset value (the base address), in order to position the first texel in the texture.

The **torg** field corresponds to a byte address in memory. This register must be set to have no overlap with either the frame buffer or the Z-depth buffer.

This field must be loaded with a multiple of 32. The five LSBs must be set to ‘00000’.

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

Address **MGABASE1** + 2C34h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value Unknown

tkmask																tckey															
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

tckey <15:0> Texture transparency color key. This field holds the 16-bit unsigned value of the color that is defined as the ‘transparent’ color. Planes that are not used must be set to 0.

tkmask <31:16> Texture color keying mask. This field enables texture transparency comparison on a planar basis (0 indicates mask). The mask setting must be based on the **twidth** value, so that unused bits are masked as shown in the table below:

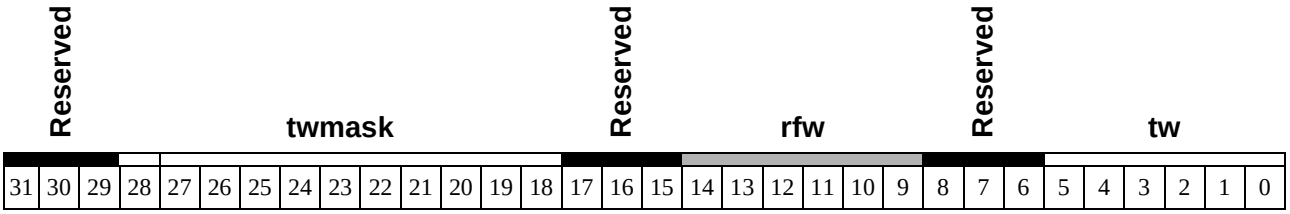
twidth	15 tkmask 0
TW4	0 0 0 0 0 0 0 0 0 0 0 0 m a s k
TW8	0 0 0 0 0 0 0 0 - - m a s k - -
TW12	- - - - - - m a s k - - - - - -
TW15	- - - - - - m a s k - - - - - -
TW16	- - - - - - m a s k - - - - - -

To disable transparency (that is, to make the texture opaque), set **tckey** to FFFFh and **tkmask** to 0000h.

Address **MGABASE1** + 2C28h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value Unknown



tw <5:0> Represents log2 of the texture width, combined with an adjustment factor. This field holds a 6-bit signed value in two's complement notation, set as follows:

tw = log2(texture width) + (4 - s_fractional_bits + q_fractional_bits)

Typically, **tw** = log2(texture width) + (4 - 20 + 16).

rfw <14:9> Width round-up factor. This factor is used to improve the generated image quality by assuring coherence in texel choice. This field holds a 6-bit signed value in two's complement notation, usually set to 8 - log2 (texture width) - (q_fractional_bits - 16).

twmask <28:18> Width mask. Determines the usable width of the texture (select on which texel the repeat or clamping will be performed). This field holds a 11-bit unsigned value, usually set to (texture width) - 1.

➽ Note: The repeat mode will not work properly if **twmask** is set to a value that is not a power of two minus one.

Reserved: <8:6> <17:15> <31:29>

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

Address **MGABASE1** + 2C00h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value Unknown

tmr0

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

tmr0 <31:0> Texture mapping ALU register 0. This field holds a signed 12.20 value in two's complement notation.

For texture mapping, the **TMR0** register holds the s/wc increment value along the x axis.

Address **MGABASE1** + 2C04h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value Unknown

tmr1

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

tmr1 <31:0> Texture mapping ALU register 1. This field holds a signed 12.20 value in two's complement notation.

For texture mapping, the **TMR1** register holds the s/wc increment value along the y axis.

Address **MGABASE1** + 2C08h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value Unknown

tmr2

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

tmr2 <31:0> Texture mapping ALU register 2. This field holds a signed 12.20 value in two's complement notation.

For texture mapping, the **TMR2** register holds the t/wc increment value along the x axis.

Address **MGABASE1** + 2C0Ch (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value Unknown

tmr3

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

tmr3 <31:0> Texture mapping ALU register 3. This field holds a signed 12.20 value in two's complement notation.

For texture mapping, the **TMR3** register holds the t/wc increment value along the y axis.

Address **MGABASE1** + 2C10h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value Unknown

tmr4

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

tmr4 <31:0> Texture mapping ALU register 4. This field holds a signed 16.16 value in two's complement notation.

For texture mapping, the **TMR4** register holds the q/wc increment value along the x axis.

Address **MGABASE1** + 2C14h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value Unknown

tmr5

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

tmr5 <31:0> Texture mapping ALU register 5. This field holds a signed 16.16 value in two's complement notation.

For texture mapping, the **TMR5** register holds the q/wc increment value along the y axis.

Address **MGABASE1** + 2C18h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown

tmr6

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

tmr6 <31:0> Texture mapping ALU register 6. This field holds a signed 12.20 value in two's complement notation.

For texture mapping, the **TMR6** register is used to scan the left edge of the trapezoid for the s/wc parameter. This register must be initialized with the starting s/wc value.

Address **MGABASE1** + 2C1Ch (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown

tmr7

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

tmr7 <31:0> Texture mapping ALU register 7. This field holds a signed 12.20 value in two's complement notation.

For texture mapping, the **TMR7** register is used to scan the left edge of the trapezoid for the t/wc parameter. This register must be initialized with the starting t/wc value.

Address **MGABASE1** + 2C20h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown

tmr8

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

tmr8 <31:0> Texture mapping ALU register 8. This field holds a signed 16.16 value in two's complement notation.

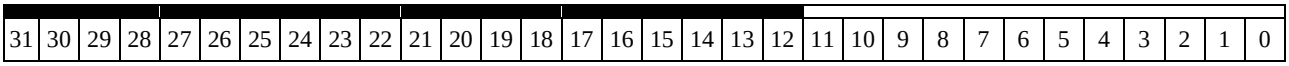
For texture mapping, the **TMR8** register is used to scan the left edge of the trapezoid for the q/wc parameter. This register must be initialized with the starting q/wc value.

➾ Note: Cases where q/wc is less than or equal to 0 will be processed as exceptions. Software should ensure that q remains positive to avoid an overflow.

Address **MGABASE1** + 1E20h (MEM)

Attributes RO, DYNAMIC, WORD/DWORD

Reset Value Unknown



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

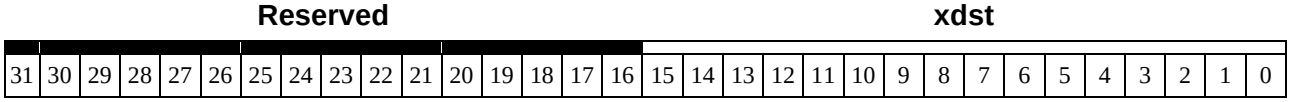
- This register must be read using a word or dword access, because the value might change between two byte accesses. A sample and hold circuit will ensure a stable value for the duration of one PCI read access.

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



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.

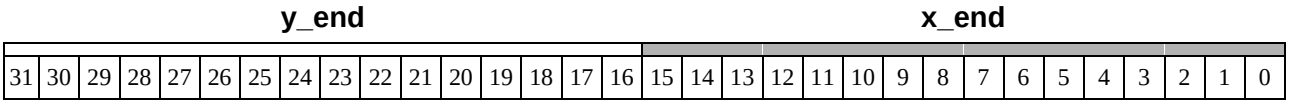
- Before starting a vector draw, **xdst** must be loaded with the x coordinate of the starting point of the vector. At the end of a vector, **xdst** contains the address of the last pixel of the vector. This can also be done by accessing the **XYSTRT** register.
- This register does not require initialization for polyline operations.
- For BLITs, this register is automatically loaded from **fxleft** (see **FXLEFT on page 4-77)** and **fxright** (see **FXRIGHT on page 4-78)**, 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 4-40.)

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



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.

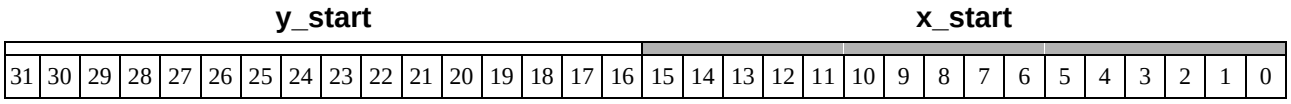
When **XYEND** is written, the following registers are affected:

- **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



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**<22:0> (sign extended) 0 --> **sellin**
- **y_start**<15:0> --> **ar6**<17:0> (sign extended)

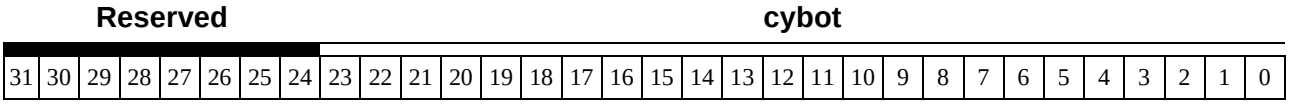
x_start <15:0> The **x_start** field contains the x coordinate of the starting point of the vector. It is a 16-bit signed value in two's complement notation.

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.

Address **MGABASE1** + 1C9Ch (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value Unknown



cybot <23:0> Clipper y bottom boundary. The **cybot** field contains an unsigned 24-bit value which is interpreted as a positive pixel address and compared with the current **ydst** (see **YDST on page 4-124)**. 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**

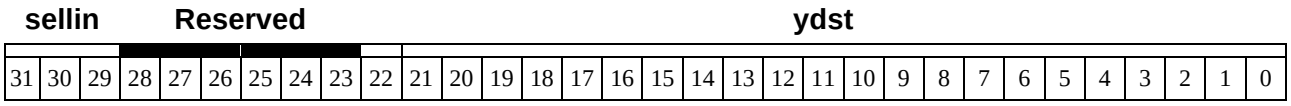
The **YBOT** register must be loaded with a multiple of 32 (the five LSBs = 0). There is no way to disable clipping.

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

Address **MGABASE1** + 1C90h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown



ydst <22:0>

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 4-88)**.

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 <28:23>

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

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 4-81.**

yval <31:16> The y destination value. See the **YDST register on page 4-124.** 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<22:0> <= sign extension (**yval**<31:16>)

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

Address **MGABASE1** + 1C94h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value Unknown

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

ydstorg <23:0> Destination y origin. The **ydstorg** field is a 24-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 5-37. See page 4-64 for additional restrictions that apply to block mode (**atype** = BLK).

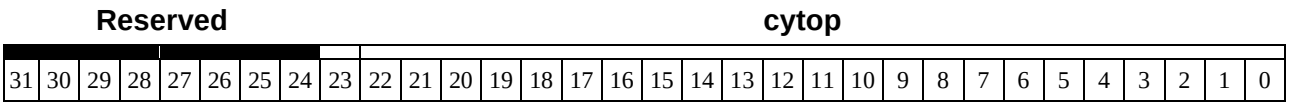
pwidth	*Value*
PW8	64
PW16	32
PW24	64
PW32	32

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

Address **MGABASE1** + 1C98h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value Unknown



cytop <23:0>

Clipper y top boundary. The **cytop** field contains an unsigned 24-bit value which is interpreted as a positive pixel address and compared with the current **ydst** (see **YDST on page 4-124)**. 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.

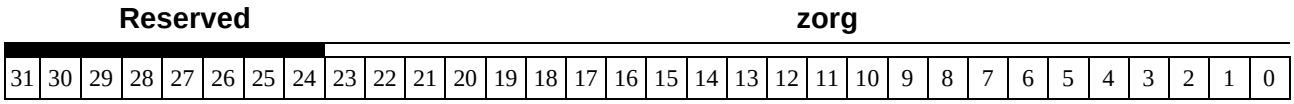
Reserved <31:24>

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



zorg <23:0>

Z-depth origin. The **zorg** field is a 24-bit unsigned value used as an offset from the Intensity buffer to position the first Z value of the depth buffer.

The **zorg** field is a byte address in memory. This register must be set so that there is no overlap with the Intensity 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 <31:24>

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

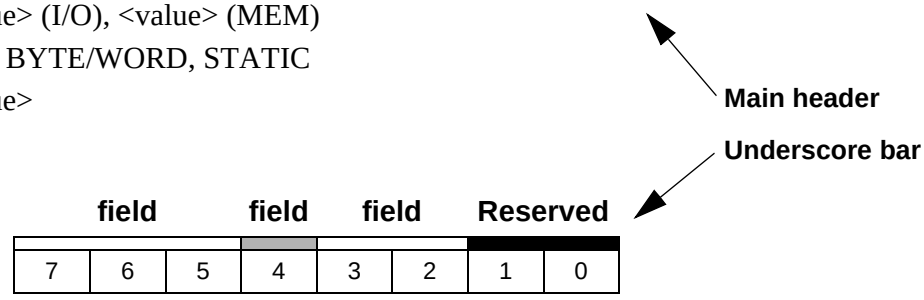
4.2 VGA Mode Registers

4.2.1 VGA Mode Register Descriptions

The MGA-G100 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? 0000b (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.

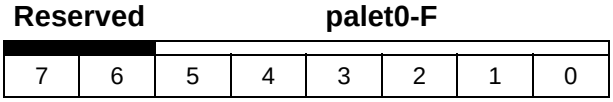
Retrieve or write the contents of the register pointed to by the **attrx** field.

Reserved <7:6>

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

Index **attrx** = 00h to **attrx** = 0Fh

Reset Value 0000 0000b



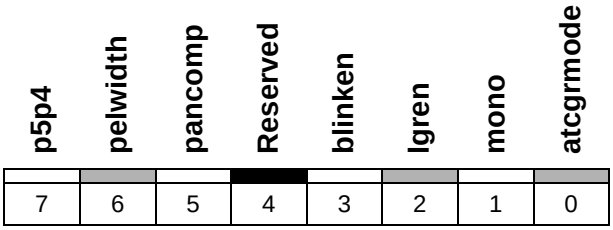
palet0-F <5:0> Internal palette data. VGA.

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 0000b



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 4-138.

Reserved <4> Reserved. When writing to this register, this field must be set to 0.

Index **attrx** = 11h

Reset Value 0000 0000b

ovscol

7	6	5	4	3	2	1	0

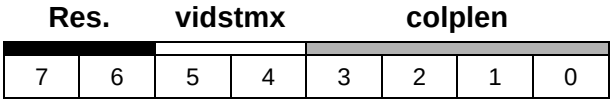
ovscol <7:0>

Overscan color. VGA.

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

Index **attrx** = 12h

Reset Value 0000 0000b



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

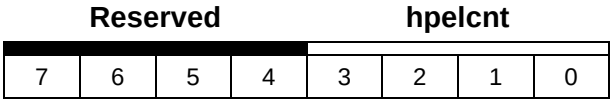
vidstmx <5:4> Video status multiplexer (MUX). VGA.

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 4-195.

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



hpelcnt <3:0> Horizontal pel count. VGA.

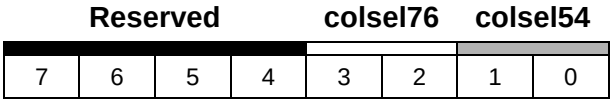
This 4-bit value specifies the number of picture elements to shift the video data horizontally to the left, according to the following table (values 9 to 15 are reserved):

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.

Index **attrx** = 14h

Reset Value 0000 0000b



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

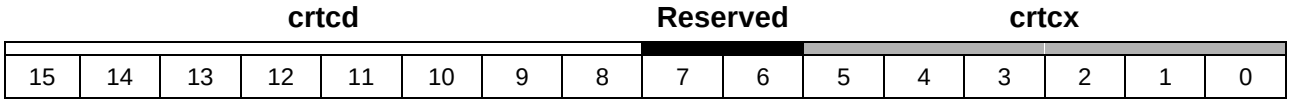
cacheflush <7:0> Flush the cache. Writes to this register will flush the cache. For additional details, refer to ‘Direct Access Read Cache’ on page 5-4.

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



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.

Index **crtcx** = 00h

Reset Value 0000 0000b

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.

Index **crtcx** = 01h

Reset Value 0000 0000b

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.

Index **crtcx** = 02h

Reset Value 0000 0000b

hblkstr

7	6	5	4	3	2	1	0

hblkstr <7:0>

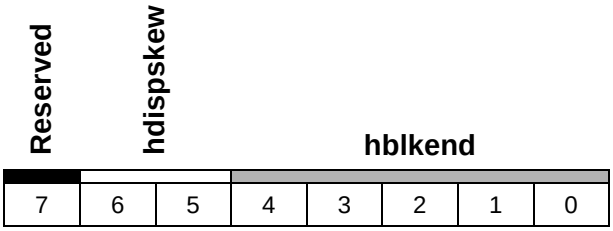
Start horizontal blanking. VGA/MGA.

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.

Index **crtcx** = 03h

Reset Value 1000 0000b



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.

Index **crtcx** = 04h

Reset Value 0000 0000b

hsyncstr

7	6	5	4	3	2	1	0

hsyncstr <7:0>

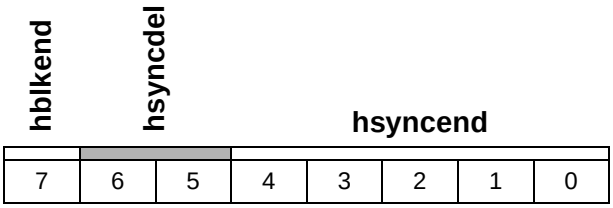
Start horizontal retrace pulse. VGA/MGA.

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.

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 4-145.

Index **crtcx** = 06h

Reset Value 0000 0000b

vtotal

7	6	5	4	3	2	1	0

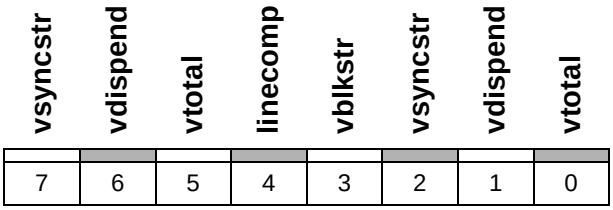
vtotal <7:0> Vertical total. VGA/MGA.

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).

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 4-148.

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 4-160.

vsyncstr <2> Vertical retrace start bit 8. VGA/MGA.
Contains bit 8 of the Vertical Retrace Start. See the **CRTC10** register on page 4-158.

vblkstr <3> Start vertical blank bit 8. VGA/MGA.
Contains bit 8 of the Start Vertical Blank. See the **CRTC15** register on page 4-163.

linecomp <4> Line compare bit 8. VGA/MGA.
Line compare bit 8. See the **CRTC18** register on page 4-169. This bit is not write-protected by **crtcprotect** (**CRTC11**<7>).

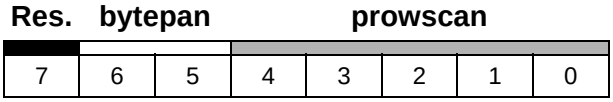
vtotal <5> Vertical total bit 9. VGA/MGA.
Contains bit 9 of the Vertical Total. See the **CRTC6** register on page 4-148.

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 4-160.

vsyncstr <7> Vertical retrace start bit 9. VGA/MGA.
Contains bit 9 of the Vertical Retrace Start. See the **CRTC10** register on page 4-158.

Index **crtcx** = 08h

Reset Value 0000 0000b



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

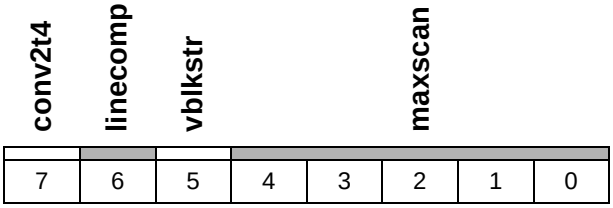
bytepan <6:5> Byte panning control. VGA/MGA.

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 0000b



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 4-163.

linecomp <6> Line compare bit 9. VGA/MGA.

Bit 9 of the Line Compare register. See the **CRTC18** register on page 4-169.

conv2t4 <7> 200 to 400 line conversion. VGA/MGA.

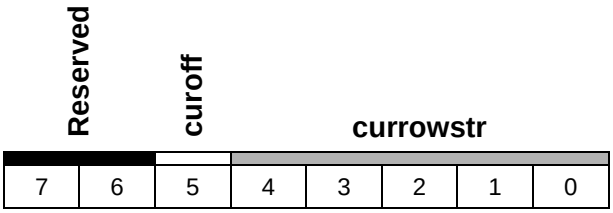
Controls the row scan counter clock and the time when the start address latch loads a new memory address:

- **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 0000b



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.

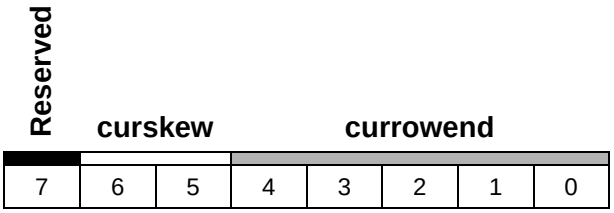
curoff <5> Cursor off. VGA.

- 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.

Index **crtcx** = 0Bh

Reset Value 0000 0000b



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 0000b

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 21-bit value from the **startadd** (**CRTCEXT0**<6, 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.

Power Graphic Mode

Special considerations for Power Graphic mode programming are presented in the Note on page 5-72 in the ‘Programming in Power Graphic Mode’ section of Chapter 5.

Index **crtcx** = 0Dh

Reset Value 0000 0000b

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 4-154.

Index **crtcx** = 0Eh

Reset Value 0000 0000b

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.

Index **crtcx** = 0Fh

Reset Value 0000 0000b

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 4-156.

Index **crtcx** = 10h

Reset Value 0000 0000b

vsyncstr

7	6	5	4	3	2	1	0

vsyncstr <7:0> Vertical retrace start bits 7 to 0. VGA/MGA.

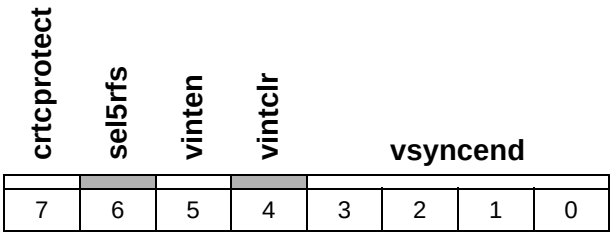
The vertical sync signal becomes active when the vertical line counter reaches the vertical retrace start value (a 12-bit value). The lower eight bits are located here. Bit 8 is in **CRTC7**<2>, bit 9 is in **CRTC7**<7>, and bits 10 and 11 are in **CRTCEXT2**<6:5>.

The units can be one or two scan lines:

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

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 0000b

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>.

Index **crtcx** = 13h

Reset Value 0000 0000b

offset

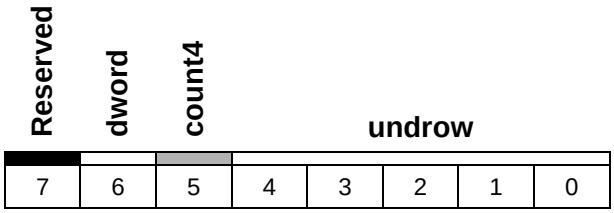
7	6	5	4	3	2	1	0

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>. In VGA mode the value is the number of double words (**dword** (**CRTC14**<6>) = 1) or single words (**dword** = 0) in one line. In MGA mode the offset is in double slices (a single slice is 8 bytes)

Index **crtcx** = 14h

Reset Value 0000 0000b



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.

Index **crtcx** = 15h

Reset Value 0000 0000b

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>.

Index **crtcx** = 16h

Reset Value 0000 0000b

vblkend

7	6	5	4	3	2	1	0

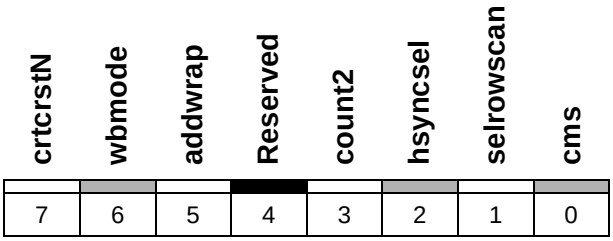
vblkend <7:0>

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 0000b



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.

Index **crtcx** = 18h

Reset Value 0000 0000b

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 0000b

cpudata

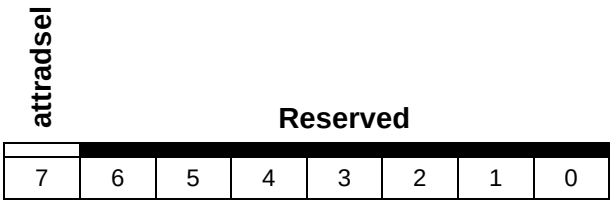
7	6	5	4	3	2	1	0

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 0000b



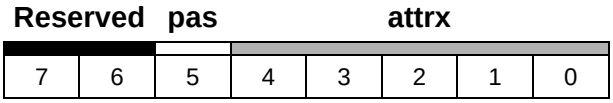
attradsel <7> Attributes address/data select. VGA.

- 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 0000b



attrx <4:0> VGA attributes address

pas <5> VGA palette enable.

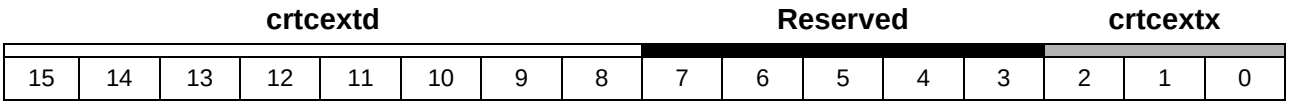
Reserved <7:6> Reserved. When writing to this register, the bits in this field must be set to 0.

• See the **ATTR** register on page 4-130.

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

Attributes R/W, BYTE/WORD, STATIC

Reset Value nnnn nnnn 0000 0000b



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
Priority Request Control	**CRTCEXT6**	06h
Reserved (1)	---	06h - 07h

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

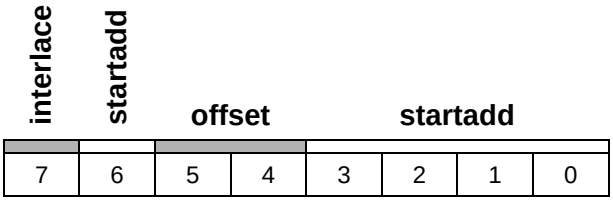
crtcextd <15:8> CRTC extension data register.

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 <6, 3:0> Start address bits 20, 19, 18, 17, and 16.

These are the five most significant bits of the start address. See the **CRTCC** register on page 4-154.

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 4-161.

interlace <7> Interlace enable.

Indicates if interlace mode is enabled.
- 0: Not in interlace mode.
- 1: Interlace mode.

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 4-142.

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 4-144.

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 4-146.

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**).

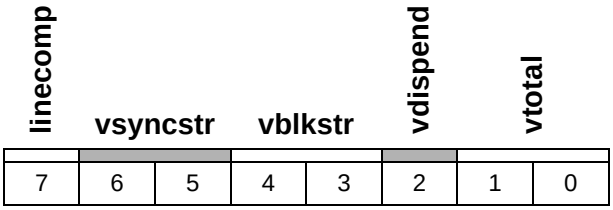
Bit 6 of the End Horizontal Blanking value. See the **CRTC3** register on page 4-145.

vrsten <7>
Vertical reset enable.

When at '1', the vertical counter can be reset by the VIDRST pin.

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 4-148.

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 4-160.

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 4-163.

vsyncstr <6:5> Vertical retrace start bits 11 and 10.

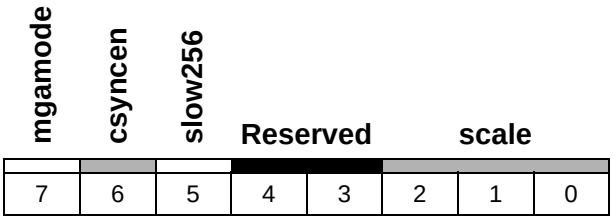
These are the two most significant bits of the **vsyncstr** (vertical retrace start) register (the vertical retrace start is then 12 bits wide). See the **CRTC10** register on page 4-158.

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 4-169.

Index **crtcextx** = 03h

Reset Value 0000 0000b



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).

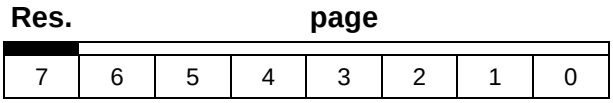
mgamode <7> MGA mode enable.

- 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.

Index **crtcextx** = 04h

Reset Value 0000 0000b



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 16 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 0000b

hvidmid

7	6	5	4	3	2	1	0

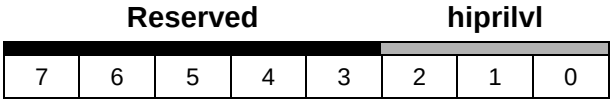
hvidmid <7:0> Horizontal video half count.

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:

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

Index **crtcextx** = 06h

Reset Value 0111 0000b



hiprilvl <2:0> High priority request level. This field indicates the number of 8 quadword requests in the CRTC fifo when the request to the memory controller changes from low to high priority.

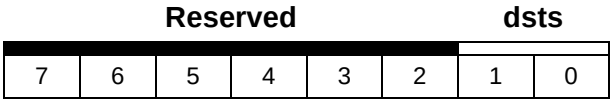
hiprilvl	*high priority request level*
'000'	1
'001'	2
'010'	3
'011'	4
'100'	5
'101'	6
'110'	Reserved
'111'	Reserved

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

Address 03C7h (I/O), **MGABASE1** + 1FC7h (MEM)

Attributes RO, BYTE, STATIC

Reset Value 0000 0000b



dsts <1:0> This port returns the last access cycle to the palette.

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

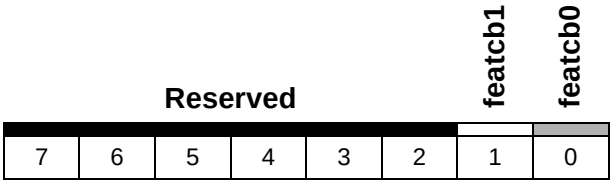
Reserved <7:2> This field returns all zeroes when read.

Refer to the **PALRDADD** register description on page 4-208 for information on writes to 03C7h.

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.

Address 03CEh (I/O), **MGABASE1** + 1FCEh (MEM)

Attributes R/W, BYTE/WORD, STATIC

Reset Value nnnn nnnn 0000 0000b

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.

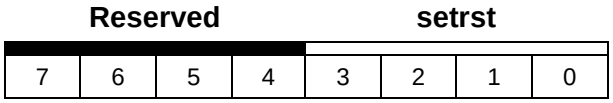
Retrieve or write the contents of the register pointed to by the **gctlx** field.

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 0000b



setrst <3:0> Set/reset. VGA.

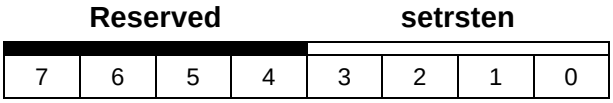
These bits allow setting or resetting byte values in the four video maps:
- 1: Set the byte, assuming the corresponding set/reset enable bit is ‘1’.
- 0: Reset the byte, assuming the corresponding set/reset enable bit is ‘0’.

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.

Index **gctlx** = 01h

Reset Value 0000 0000b



setrsten <3:0> Enable set/reset planes 3 to 0. VGA.

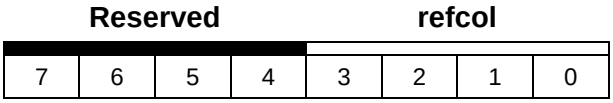
When a set/reset plane is enabled (the corresponding bit is '1') and the write mode is 0 (**wrmode** (**GCTL5**<1:0>) = 00), the value written to all eight bits of that plane represents the contents of the set/reset register. Otherwise, the rotated CPU data is used.

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.

Index **gctlx** = 02h

Reset Value 0000 0000b



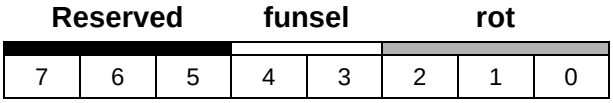
refcol <3:0> Reference color. VGA.

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 4-192) register will be tested.

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

Index **gctlx** = 03h

Reset Value 0000 0000b



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 4-193) register to select which pixel is written.

funsel <4:3> Function select. VGA.

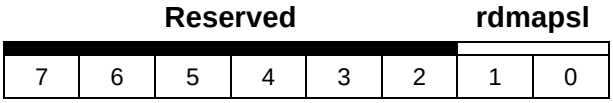
Specifies one of four logical operations between the video memory data latches and any data (the source depends on the write mode).

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.

Index **gctlx** = 04h

Reset Value 0000 0000b



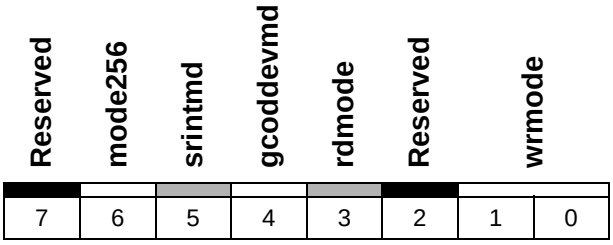
rdmapsl <1:0> Read map select. VGA.

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 4-193) 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.

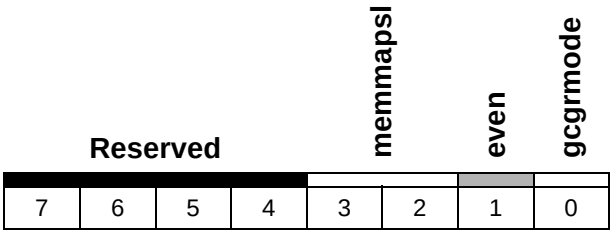
- 0: The loading of the shift registers is controlled by the **srintmd** field.
- 1: The shift registers are loaded in a manner which supports 256-color mode.

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 4-196).

memmapsl <3:2> Memory map select bits 1 and 0. VGA.

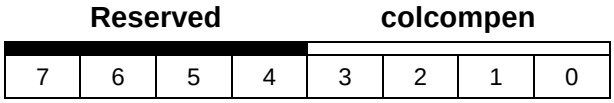
These bits select where the video memory is mapped, as shown below:

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.

Index **gctlx** = 07h

Reset Value 0000 0000b



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.

Index **gctlx** = 08h

Reset Value 0000 0000b

wrmask

7	6	5	4	3	2	1	0

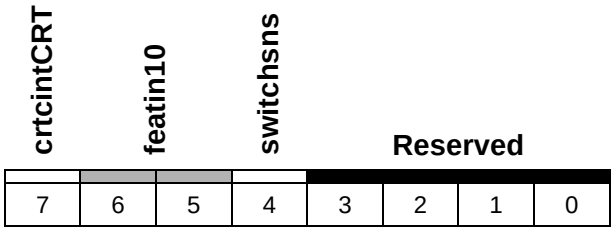
wrmask <7:0> Data write mask for pixels 7 to 0. VGA.

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.

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.

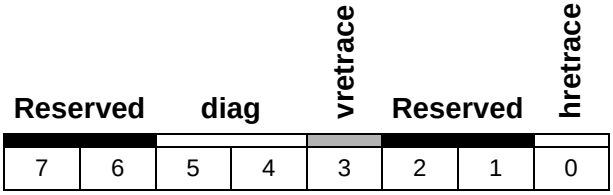
- 0: Vertical retrace interrupt is cleared.
- 1: Vertical retrace interrupt is pending.

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

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 **vidstmx** field (**ATTR12**<5:4>) 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

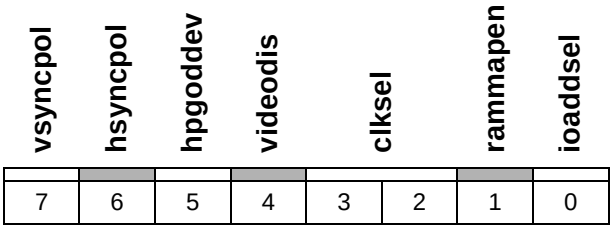
Reserved: <2:1> <7:6>

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

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

Attributes R/W, BYTE, STATIC

Reset Value 0000 0000b



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

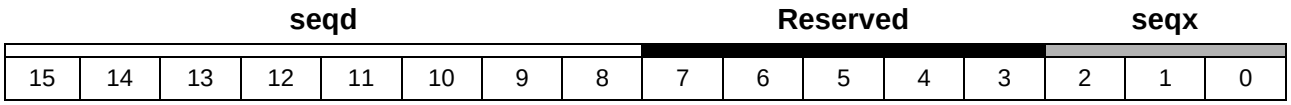
vsyncpol <7> Vertical sync polarity. VGA/MGA.

- Logical ‘0’: active high vertical sync pulse
- Logical ‘1’: active low vertical sync pulse

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

Attributes R/W, BYTE/WORD, STATIC

Reset Value nnnn nnnn 0000 0000b



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.

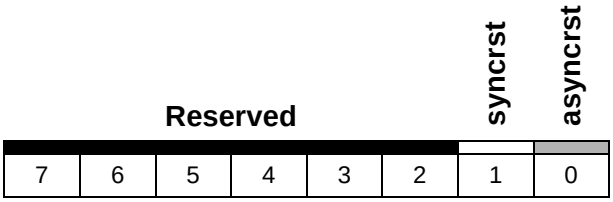
seqd <15:8> Sequencer data register.

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.

Index **seqx** = 00h

Reset Value 0000 0000b



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.

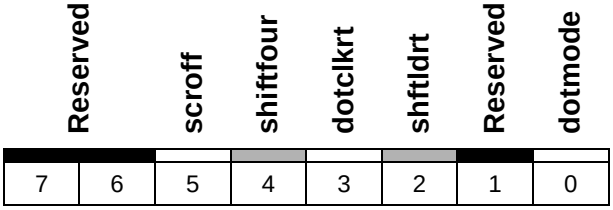
syncrst <1> Synchronous reset. VGA.

- 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-G100 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 0000b



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.

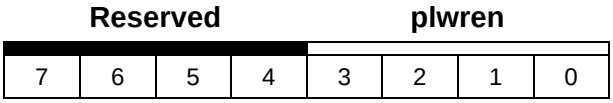
- 0: Normal video operation.
- 1: Turns off the video, and maximum memory bandwidth is assigned to the system. The display is blanked, however all sync pulses are generated normally.

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.

Index **seqx** = 02h

Reset Value 0000 0000b



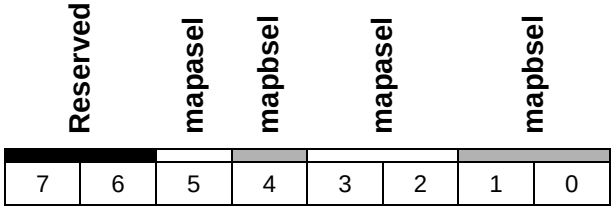
plwren <3:0> Map 3, 2, 1 and 0 write enable. VGA.

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 0000b



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.

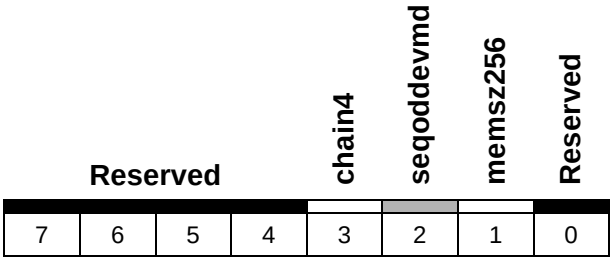
These bits are used for alpha character generation when the character's attribute bit 3 is '1', according to the following table:

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.

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.

This page is left blank intentionally.

4.3 DAC Registers

4.3.1 DAC Register Descriptions

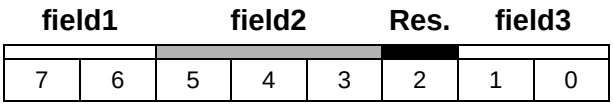
The MGA-G100 DAC register descriptions contain a (gray single-underlined) main header which indicates the register's name and mnemonic. Below the main header, the memory address or index, attributes, and reset value are indicated. Next, an illustration of the register identifies the bit fields, which 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>

Main header

Underscore bar



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

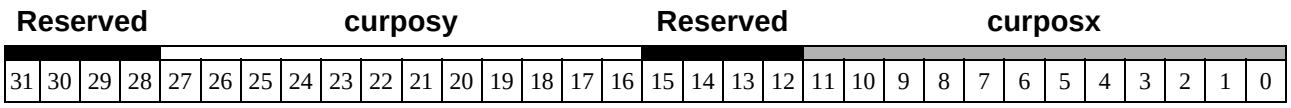
Here are some of the symbols that appear as part of a register's reset value.

- `000? 0?00`b
  (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



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.

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.

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 increased 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**.

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 1111h

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.

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	

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.

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.

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

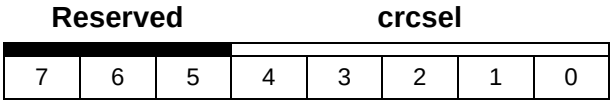
colkeymsk

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.

Index 3Eh
Attributes R/W, BYTE
Reset Value Unknown



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.

Index 3Dh
Attributes RO, BYTE
Reset Value Unknown

crcdata

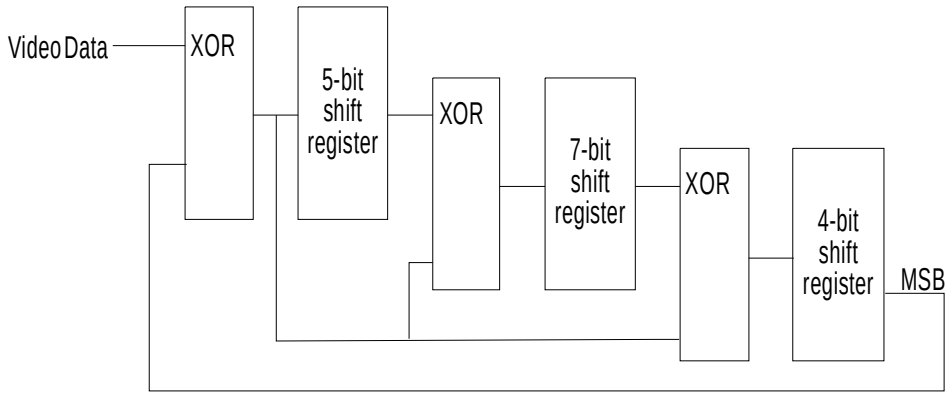
7	6	5	4	3	2	1	0

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:



Index 3Ch
Attributes RO, BYTE
Reset Value Unknown

crcdata

7	6	5	4	3	2	1	0

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**.

Index 05h

Attributes R/W, BYTE

Reset Value Unknown

Reserved			curadrh				
7	6	5	4	3	2	1	0

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.

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.

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

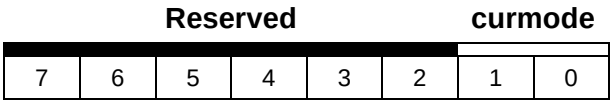
7	6	5	4	3	2	1	0

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 0000b



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).

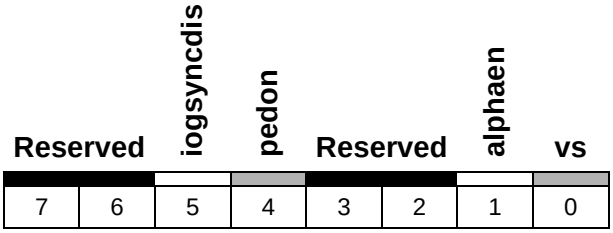
(2) Each bit of the underlying pixel is inverted, then displayed.

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

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.

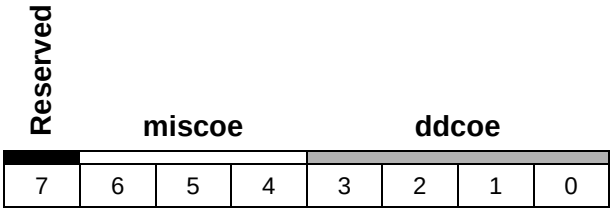
Reserved: <3:2> <7:6>

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

Index 2Ah

Attributes R/W, BYTE

Reset Value 0000 0000b



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 <6:4> MISC pin output control. Controls the output enable of the driver on pins MISC<2:0>, respectively.

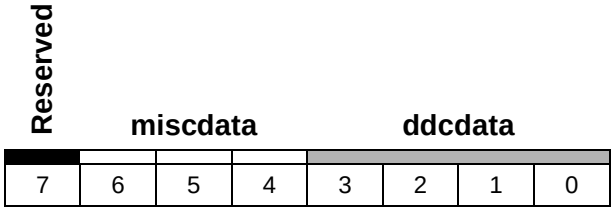
- 0: disable the output driver
- 1: enable the output driver

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

Index 2Bh

Attributes R/W, BYTE

Reset Value 0000 0000b



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.

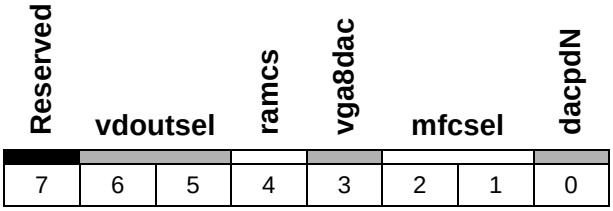
miscdata <6:4> MISC pin output state. Controls the output state of the driver on pins MISC<2:0> during a write operation. On read, this field returns the state of the MISC<2:0> pins.

Reserved <7> 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 0110b



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> This field selects the mode in which the VDOUT interface will operate. Notice that this interface is independent of the video-in interface that utilizes the VD bus and VDCLK pins. The video-in iterface is always an input to MGA-G100 and it is not affected by the configuration of the video-out fields **mfcsel** and **vdoutsel.**

- 00: Reserved

- 01: VDOUT mode

  In VDOUT mode, the interface drives data through the VDOUT bus to provide video-out functionality and behaves according to the selection in the **XMISCCTRL** field **vdoutsel** <6:5>.

  VDOCLK - taken as an input

  VDOUT bus - driven with MAFC multiplexed 24-bit data (output).

  VOBLANKN - driven with the gated clock/blank (output).

- 10: PanelLink mode

  In PanelLink mode, the VDOUT interface behaves just like in VDOUT mode (“01”) except:

  VDOUT bus is mapped appropriately for the panel link device.

  VDOCLK is driven with internal pixel PLL instead of taken as an input.

  VOBLANKN is driven with blank instead of gated with the clock.

  The **XMISCCTRL** field **vdoutsel** <6:5> has no effect in this mode.

- 11: Disable mode

  In disable mode, VDOUT bus <11:0> and VOBLANKN, are driven low.

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

vdoutsel <6:5>

VDOUT Interface Selection. This field selects the behavior of the VDOUT interface and it only has an effect when the interface is active (**XMISCCTRL** field **mfcsel** <2:1> = “01”)

- 00: MAFC mode A

  In this mode the VDOUT interface is compatible with the Matrox MAVEN Encoder. Output just before the DAC is multiplexed on the 12 bit VDOUT bus using bothe edges of the clock (VDOCLK). This effectively transfers one 24-bit RGB pixel per clock.

  VDOCLK - talken as an input

  VDOUT bus - driven with multiplexed 24-bit data (output).

  VOBLANKN - driven with the gated clock/blank (output).

  ➾ *Note*: It is important that the field **pixclksl** <1:0> in register **XPIXCLKCTRL** is set to “10” to select the external source for the pixel clock.

- 01: MAFC mode B

  In this mode data is maped differently on VDOUT <11:0>. The VOBLANKN pin is driven with the input VDOCLK that is sourced by the encoder. That is VDOCLK is passed through to VOBLANKN instead of being gated with the blank. This clock is used to provide a delayed clock that is synchronized with the data being driven out. Refer to Section Figure 6-8: on page 6-11 for interface details.

- 10: bypass656 mode

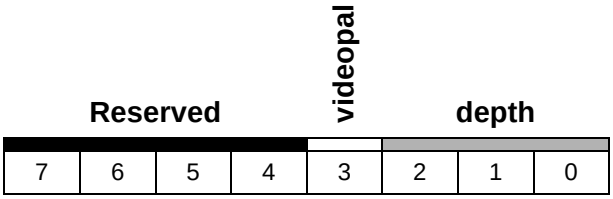
  In this mode the VD<7:0> bus input data is passed through MGA-G100 to the VDOUT<7:0> bus, delayed by one VDCLK clock cycle. The data coming out of VDOUT <7:0> is synchronized to the VDCLK. Bits <11:8> of VDOUT bus are driven low. VOBLANKN is a delayed version of VDCLK input that is synchronized with data

- 11: Reserved

Reserved <7:5>

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

Index 19h
Attributes R/W, BYTE
Reset Value 0000 0000b



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 when at '0' sets the DAC to VGA mode. The **depth** field should be programmed to "000" when in VGA mode.

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

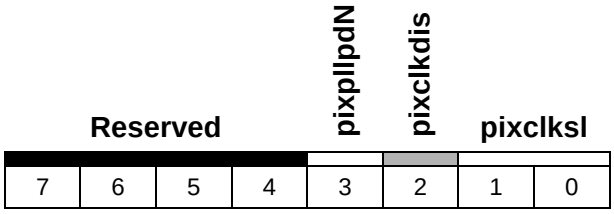
Reserved <7:4> Reserved. When writing to this register, the bits in this field must be set to 0.

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

Index 1Ah

Attributes R/W, BYTE

Reset Value 0000 0000b



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 VDOCLK 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

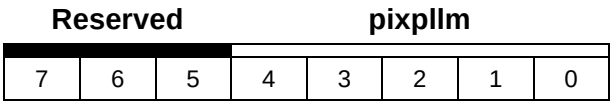
Reserved <7:4> Reserved. When writing to this register, the bits in this field must be set to 0.

See 'Programming the PLLs on page 5-82 for information on modifying the clock parameters.

Index 44h **XPIXPLLAM**
48h **XPIXPLLBM**
4Ch **XPIXPLLCM**

Attributes R/W, BYTE

Reset Value 15h **XPIXPLLAM**
1Eh **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 (25.159 MHz)	M = 21
'01'	Register Set B (28.306 MHz)	M = 30
'1X'	Register Set C	Unknown

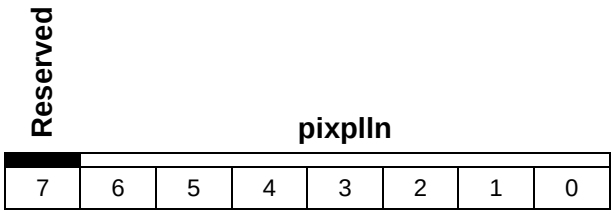
Note: The **pixpllm** value must be in the range of 1 to 6 inclusive.

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 28h **XPIXPLLAN**
40h **XPIXPLLBN**
Unknown **XPIXPLLCN**



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 (25.159 MHz)	N = 40
'01'	Register Set B (28.306 MHz)	N = 64
'1X'	Register Set C	Unknown

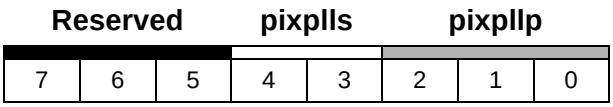
Note: The **pixplln** value must be in the range of 7 to 127 (7Fh) inclusive.

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

Index 46h **XPIXPLLAP**
4Ah **XPIXPLLBP**
4Eh **XPIXPLLCP**

Attributes R/W, BYTE

Reset Value 01h **XPIXPLLAP**
01h **XPIXPLLBP**
Unknown **XPIXPLLCP**



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.

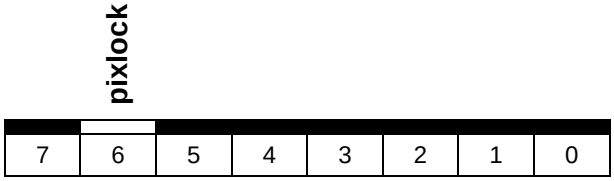
50 MHz <= Fvco < 100 MHz S=0
100 MHz <= Fvco < 140 MHz S=1
140 MHz <= Fvco < 180 MHz S=2
180 MHz <= Fvco < 220 MHz S=3

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 (25.159 MHz)	P = 1, S = 0
‘01’	Register Set B (28.306 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.

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

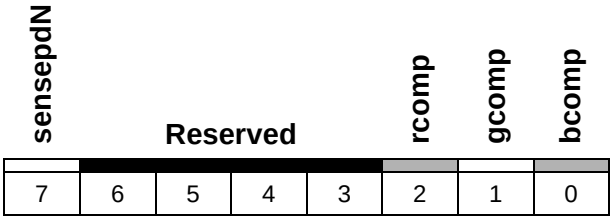
Reserved: <5:0> <7>

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 XXXXb



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

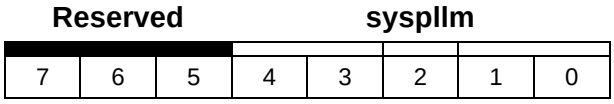
Reserved <6:3> Reserved. When writing to this register, the bits in this field must be set to 0.

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.

Index 2Ch

Attributes R/W, BYTE

Reset Value 0000 0110b



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

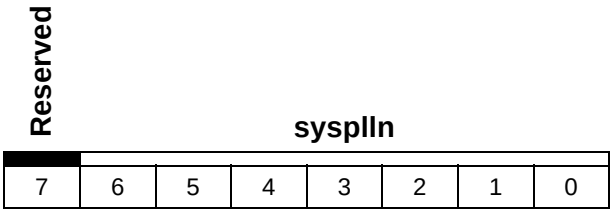
Note: The **pixpllm** value must be in the range of 1 to 6 inclusive.

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

Index 2Dh

Attributes R/W, BYTE

Reset Value 0010 0100b



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

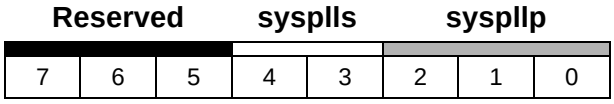
Note: The **sysplln** value must be in the range of 7 to 127 (7Fh) inclusive.

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

Index 2Eh

Attributes R/W, BYTE

Reset Value 0001 0000b



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.

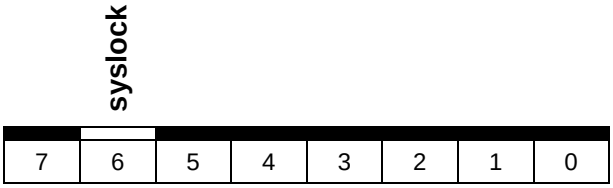
50 MHz <= Fvco < 100 MHz -> S=0
100 MHz <= Fvco < 140 MHz -> S=1
140 MHz <= Fvco < 180 MHz -> S=2
180 MHz <= Fvco < 220 MHz -> S=3

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

Index 2Fh

Attributes RO, BYTE

Reset Value Unknown



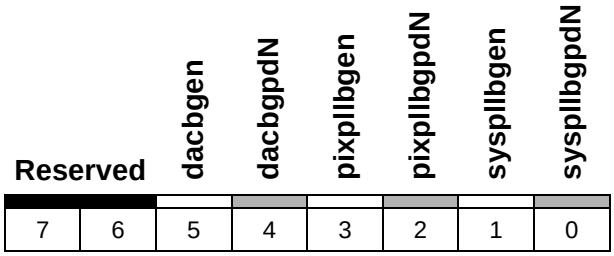
syslock <6> System PLL lock status.

- 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 0000b



syspllbgpdN <0> System PLL voltage reference block power down.

- 0: power down
- 1: power up

➸ Note: The **syspllbgpdN** field must be set to '1'.

syspllbgen <1> System PLL voltage reference enable.

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

➸ Note: The **syspllbgen** field must be set to '1'.

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

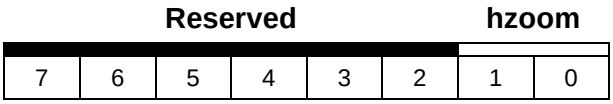
Reserved <7:6> Reserved. When writing to this register, the bits in this field must be set to 0.

➸ Note: 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').

Index 38h

Attributes R/W, BYTE

Reset Value `0000 0000`b



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.

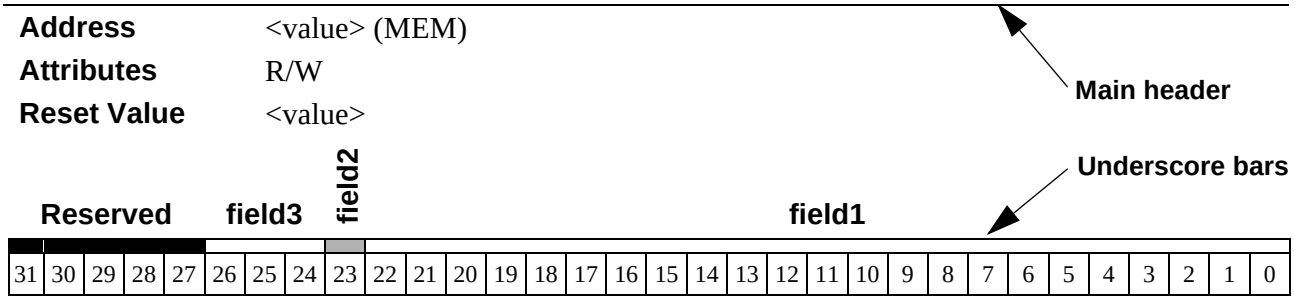
4.4 Video Expansion Registers

4.4.1 Video Expansion Register Descriptions

Video Expansion 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 Video Expansion Register **SAMPLE_VE**

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



field1 <22:0> Field 1. Detailed description of the **field1** field of the **SAMPLE_VE** 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_VE**, which is bit 23.

field3 <26:24> Field 3. Detailed description of the **field3** field of the **SAMPLE_VE** 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 Video Expansion registers are provided in Chapter 3.

➧ Note: VE indicates that the address lies within the video expansion.

Attributes

The Video Expansion register 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.

Reset Value

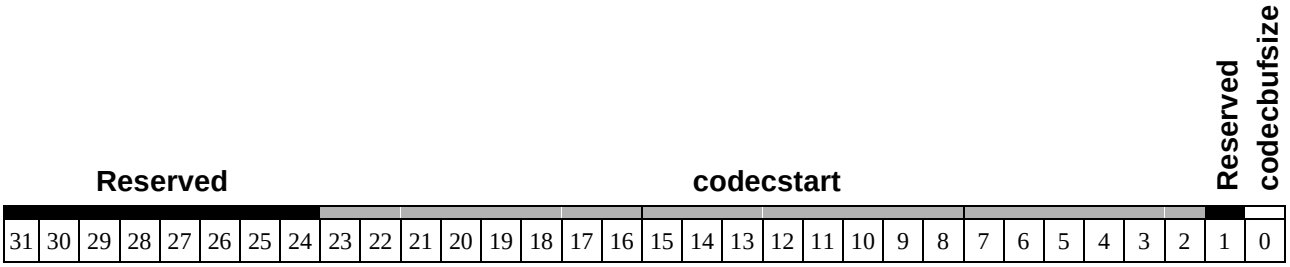
Here are some of the symbols that appear as part of a register's reset value:

- `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)

Address MGABASE1 + 3E44h (MEM)

Attributes WO, BYTE/WORD/DWORD, STATIC

Reset Value Unknown



codecbufsize <0> Codec Buffer Size

- '0' = 128K byte
- '1' = 256K byte

codecstart <23:2> Codec Buffer Start Address. The Codec Interface's buffer start address in the frame buffer is specified on a 1KB boundary (bits <**9:2**> must be loaded with '0').

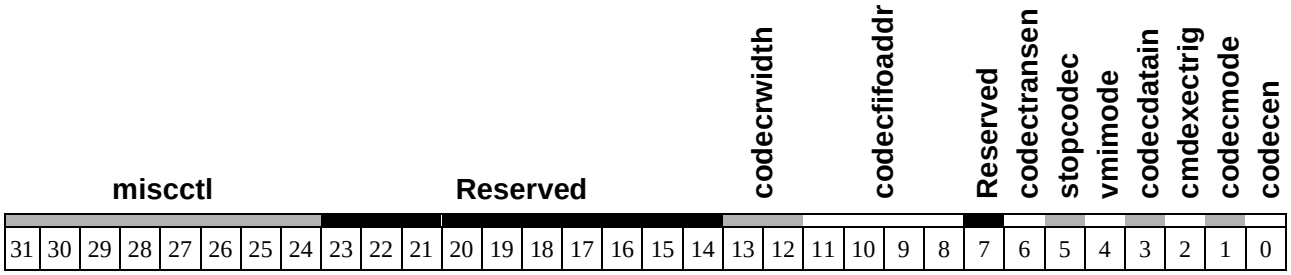
Reserved: <1> <31:24>

Reserved. Must be set to '0'.

Address MGABASE1 + 3E40h (MEM)

Attributes WO, BYTE/WORD/DWORD, STATIC

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



codecen <0> Codec Enable. This bit resets the Codec Interface engine, the **CODECHARDPTR** register, the Codec Interface interrupts, and the Codec Interface interrupt enables.

- '0' = disable (default)
- '1' = enable

codecmode <1> Codec Mode.

- '0' = VMI mode
- '1' = I33 mode

cmdexectrig <2> Command Execution Trigger

- '0' = do not execute commands in memory
- '1' = execute register commands in memory

Note: If this bit is written while data transfers are in progress the codec interface will automatically stop data transfers, trash its fifo contents and execute the commands in the command buffer.

codecdatain <3> Codec Data In.

- '0' = decompression (Twister to CODEC)
- '1' = compression (CODEC to Twister)

vmimode <4> VMI mode valid only when **codecmode** = '0'.

- '0' = Mode A
- '1' = Mode B

stopcodec <5> Stop Codec (either compression or decompression) During compressed transfers, this bit determines whether or not more than 1 field will be transferred.

- '0' = do not stop after current field
- '1' = stop after current field

codectransen <6> Codec Transfer Enable. This field enables compressed transfers to begin. After compressed transfers are underway, this bit suspends the transfers to allow software to either fill the frame buffer with more data (during compression) or empty the frame buffer of data (during compression).

- '0' = disable transfer of compressed data
- '1' = enable transfer of compress data

codecfifoaddr <11:8> Compression/Decompression Fifo Address of the Codec. When in VMI mode, the address output is **codecfifoaddr <11:8>**. When in I33 mode, only 3 bits are output, **codecfifoaddr <10:8>.**

codecrwidth <13:12> Pulse recovery width. This bit determines the number of mclkbuf clock cycles between the rising edge of a strobe and the falling edge of the next strobe of consecutive bytes when performing compression or decompression.

I33 mode and VMI mode B:

"00" = 4 mclkbuf cycles between read/write strobes

"01" = 5 mclkbuf cycles between read/write strobes

"10" = 6 mclkbuf cycles between read/write strobes

VMI mode A:

"00" = 5 mclkbuf cycles between data strobes

"01" = 6 mclkbuf cycles between data strobes

"10" = 7 mclkbuf cycles between data strobes

miscctl <31:24> Miscellaneous control. This byte is used to program on 8 bit flip-flop on the board for controlling the chip selects and other functions (SLEEP, START, etc.) of the CODEC chips. The Codec interface must be enabled for the programming sequence to be executed.

Reserved: <7> <23:14>

Reserved. Must be set to '0'.

Address MGABASE1 + 3E50h (MEM)

Attributes RO, BYTE/WORD/DWORD

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b

Reserved																codeclcode															
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

codeclcode <15:0> Used only in compression. It will point to the DWORD offset in the CODEC circular buffer following the last DWORD of the field. This register will be updated after the CODEC asserts its EOI (LCODE) pin.

Reserved <31:16> Reserved field. Must be set to ‘0’.

Address MGABASE1 + 3E4Ch (MEM)

Attributes RO, BYTE/WORD/DWORD

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b

Reserved																codechardptr															
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

codechardptr <15:0> CODEC hardware pointer. The function of this register changes, depending on the direction of the CODEC interface. The value of this register is incremented by the CODEC interface channel to always point to the next location to be accessed

Additional Reset condition: **codecen**

- When compressing video data, this register will point to the offset of the next dword to be written to the codec interface's circular buffer.
- When decompressing video data, this register will point to the offset of the next dword to be read from the codec interface's circular buffer.

Reserved <31:16> Reserved field. Must be set to '0'.

Address MGABASE1 + 3E48h (MEM)

Attributes WO, BYTE/WORD/DWORD, STATIC

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b

Reserved																codechostptr															
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

codechostptr <15:0> CODEC Host Pointer. An interrupt is generated (if enabled) when the dword offset pointed to by this register is accessed by the Codec Interface in its circular buffer. The value loaded must be on an 8 dword boundary (the 3 LSB's must be zero).

Reserved <31:16> Reserved field. Must be set to '0'.

Address MGABASE1 + 3E08h (MEM)

Attributes WO, BYTE/WORD/DWORD, STATIC

Reset Value Unknown

Reserved								vbiaddr0																							
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

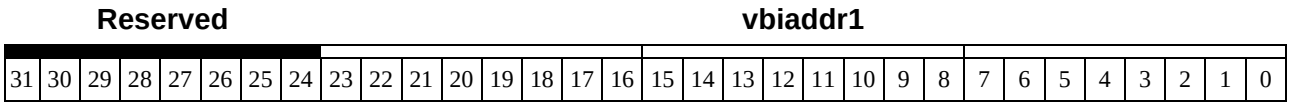
vbiaddr0 <23:0> VBI Data Start Address Window 0. Start address in bytes in the frame buffer of VBI data for Window 0. This field must be loaded with a multiple of 512 (the 9 LSBs = ‘0’).

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

Address MGABASE1 + 3E0Ch (MEM)

Attributes WO, BYTE/WORD/DWORD, STATIC

Reset Value Unknown



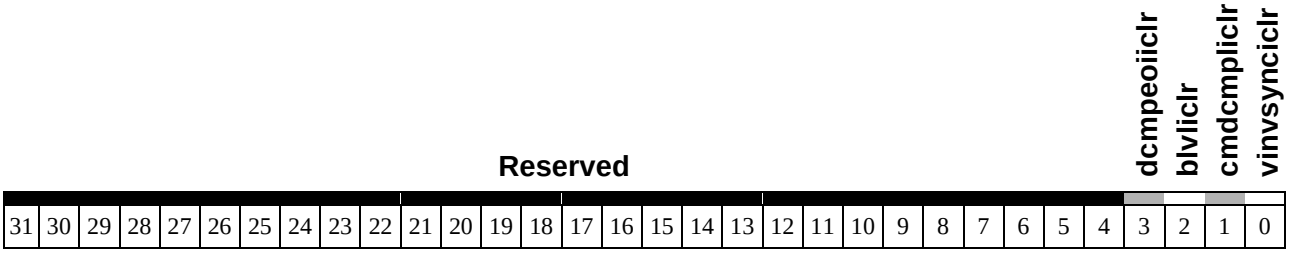
vbiaddr1 <23:0> VBI Data Start Address Window 1. Start address in bytes in the frame buffer of VBI data for Window 1. This field must be loaded with a multiple of 512 (the 9 LSBs = ‘0’).

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

Address MGABASE1 + 3E34h (MEM)

Attributes WO, BYTE/WORD/DWORD, DYNAMIC

Reset Value Unknown



vinvsyncclr <0> Video Input Vsync Interrupt Clear. When writing a ‘1’ to this bit, the input vsync interrupt pending flag is cleared.

cmdcmpliclr <1> Command Complete Interrupt Clear. When writing a ‘1’ to this bit, the command complete interrupt pending flag is cleared.

blvliclr <2> Buffer Level Interrupt Clear. When writing a ‘1’ to this bit, the buffer level interrupt pending flag is cleared.

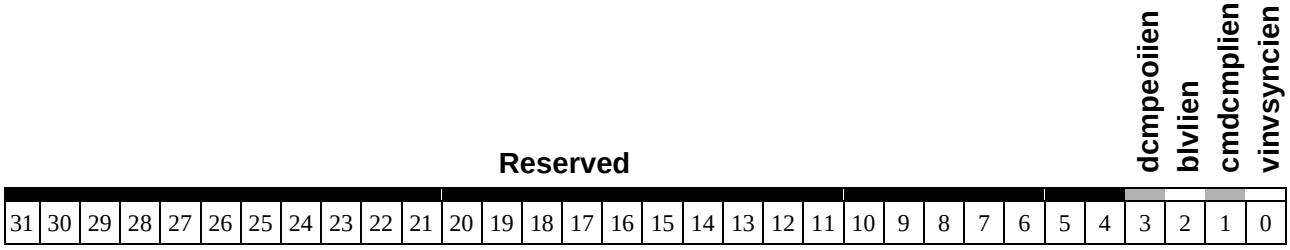
dcmpeoiiclr <3> Codec decompression end of image interrupt clear. When writing a ‘1’ to this bit, the end of image interrupt pending flag is cleared.

Reserved <31:4> Reserved. Writing to this field has not effect.

Address MGABASE1 + 3E38h (MEM)

Attributes WO, BYTE/WORD/DWORD, STATIC, DYNAMIC

Reset Value XXXX XXXX XXXX XXXX XXXX XXXX XXXX 0000b



vinvsyncien <0> Video Input Vsync Interrupt Enable. When set to ‘1’, an interrupt will be generated when the input video interrupt occurs.

cmdcmplien <1> Codec Command Complete Interrupt Enable. When set to ‘1’, an interrupt will be generated when the command execution is complete.

blvlien <2> Buffer Level Interrupt Enable. When set to ‘1’ an interrupt will be generated when the Codec Interface Read pointer for decompression (write pointer for compression) has reached the value set in the **CODECHOSTPTR** register.

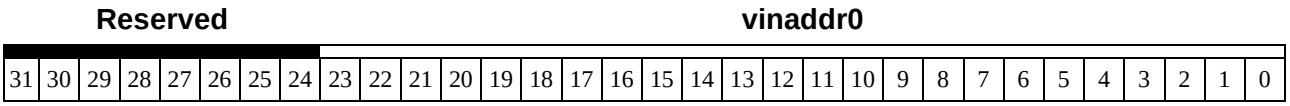
dcmpeoiien <3> Codec decompression end of image interrupt enable. When set to a ‘1’, an interrupt will be generated when the Codec Interface is performing decompression and the end of image marker is detected in the stream.

Reserved <31:4> Reserved. Writing to this field has no effect.

Address MGABASE1 + 3E10h (MEM)

Attributes WO, BYTE/WORD/DWORD, STATIC

Reset Value Unknown



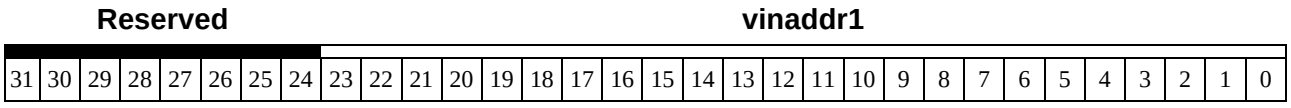
vinaddr0 <23:0> Video Write Start Address for Window 0. Start address in frame buffer of window 0.(byte boundary). This field must be loaded with a multiple of 8 (the 3 LSBs = ‘0’).

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

Address MGABASE1 + 3E14h (MEM)

Attributes WO, BYTE/WORD/DWORD, STATIC

Reset Value Unknown



vinaddr1 <23:0> Video Write Start Address for Window 1. Start address in frame buffer of window 0.(byte boundary). This field must be loaded with a multiple of 8 (the 3 LSBs = ‘0’).

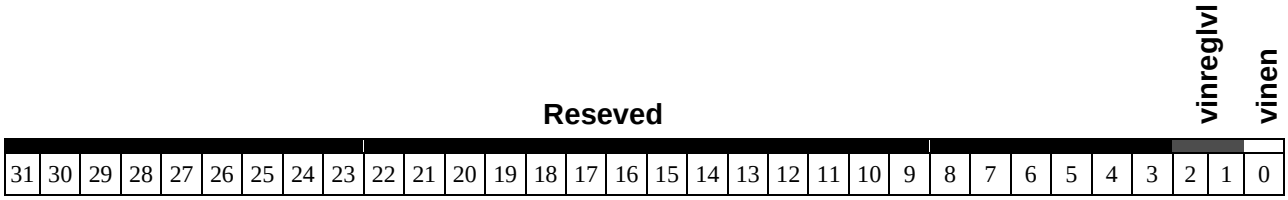
Reserved <31:24> Reserved. Writing to this field has no effect

VINCTL Video Input Control

Address MGABASE1 + 3E1Ch (MEM)

Attributes WO, BYTE/WORD/DWORD, STATIC

Reset Value 00000 0000 0000 0000 0000 0000 0000 0100b



vinen <0> Video in soft enable. When this bit is set to a '0', the video in macro is disabled, the video in control registers are reset, the video in status fields are reset, and the video in interrupt enables are reset.

- 0: disable
- 1: enable

vinreglvl <2:1> Video In request level. This field indicates the number of quadwords in the Video In FIFO when request to the memory controller becomes active.

Once active the request stays active until the FIFO is empty.

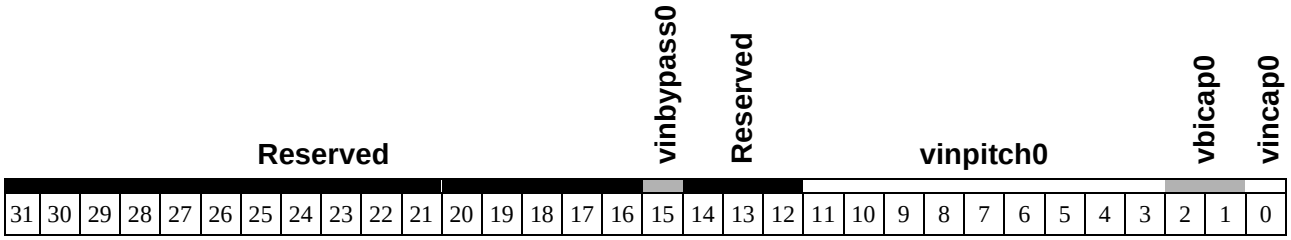
vinreglvl	*request level*
00	1
01	2
10	3
11	Reserved

Reserved <31:3> Reserved. Writing to this field has no effect.

Address MGABASE1 + 3E00h (MEM)

Attributes WO, BYTE/WORD/DWORD, STATIC

Reset Value 00000 0000 0000 0000 0000 0000 0000 0000b



vincap0 <0> Video Input Capture for window 0.

- 0: disable
- 1: enable

vbicap0 <2:1> VBI Capture for window 0

- 00 no vbi capture
- 01 capture raw VBI data as Task B
- 10 Reserved
- 11 capture sliced VBI data as Task B

vinpitch0 <11:3> Video Input Pitch for window 0

vinpitch0 field is mod 4. Mod 4 is used because the incoming video is converted to RGB16, resulting in 4 pixels per qword location. The actual line pitch is **vinpitch0** *4. This allows up to 2048 pixels. ‘0’ indicates a pitch of 2048.

vinbypass0 <15> Video Input Bypass for window 0. When set to ‘1’ the video captured will be written directly into memory, bypassing the filtered upsampling, and color space conversion to RGB16. This bit is only valid when **vincap0** is set to ‘1’.

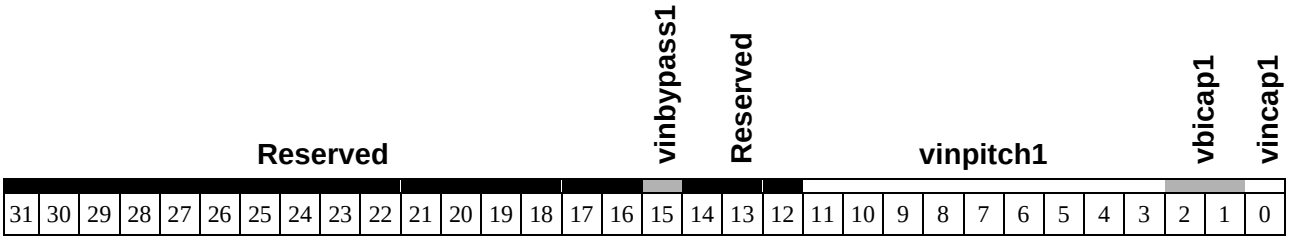
Reserved: <14:12> <31:16>

Reserved. Writing to this field has no effect.

Address MGABASE1 + 3E04h (MEM)

Attributes WO, BYTE/WORD/DWORD, STATIC

Reset Value 00000 0000 0000 0000 0000 0000 0000 0000b



vincap1 <0> Video Input Capture for window 1.

- 0: disable
- 1: enable

vbicap1 <2:1> VBI Capture for window 1

- 00 no vbi capture
- 01 capture raw VBI data as Task B
- 10 Reserved
- 11 capture sliced VBI data as Task B

vinpitch1 <11:3> Video Input Pitch for window 1

vinpitch1 field is mod 4. Mod 4 is used because the incoming video is converted to RGB16, resulting in 4 pixels per qword location. The actual line pitch is **vinpitch1** field *4. This allows up to 2048 pixels. ‘0’ indicates a pitch of 2048.

vinbypass1 <15> Video Input Bypass for window 1. When set to ‘1’ the video captured will be written directly into memory, bypassing the filtered upsampling, and color space conversion to RGB16. This bit is only valid when **vincap1** is set to ‘1’.

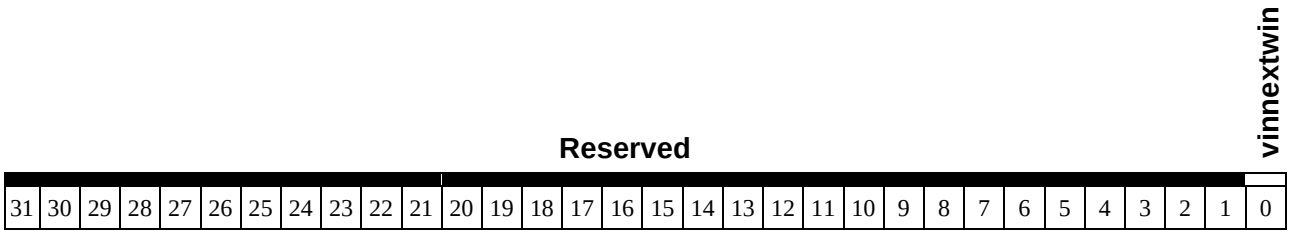
Reserved: <14:12> <31:16>

Reserved. Writing to this field has no effect.

Address MGABASE1 + 3E18h (MEM)

Attributes WO, BYTE/WORD/DWORD, STATIC

Reset Value 00000 0000 0000 0000 0000 0000 0000 0000b



vinnextwin <0> Next Active Window trigger

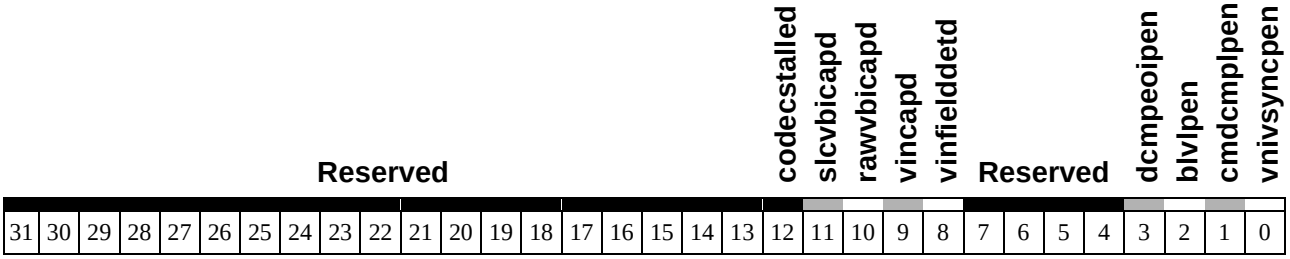
- '0' = grab window 0 next
- '1' = grab window 1 next

Reserved <31:1> Reserved. Writing to this field has no effect.

Address MGABASE1 + 3E30h (MEM)

Attributes RO, BYTE/WORD/DWORD, STATIC

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



vinvsyncpen <0> Video Input Vsync interrupt pending. When set to '1', indicates that a video input vsync interrupt has occurred. When this interrupt occurs the type of field and the data that was captured is set in fields **vinfieldcapd**, **vincapd rawvbicapd** , **slcvbicapd.**

This bit is cleared through the **vinvsynciclr** bit (see **VICLEAR** on page 4-251) or upon a video in soft or a hardreset.

cmdcmplpen <1> Command Complete Interrupt Pending. When set to '1', this bit indicates that the codec interface has completed command execution

blvlpen <2> Buffer Level Interrupt Pending. When set to '1', this bit indicates that Codec Interface read pointer for decompression (write pointer for compression) has reached the value set in the **CODECHOSTPTR** register.

dcmpeoipen <3> Codec decompression end of image interrupt pending. When set to a '1', the Codec interface has detected an end of image marker in the decompression stream.

vinfielddetd <8> Video Input Field Detected. Indicates the previous field type.

- '0' = odd field
- '1' = even field

vincapd <9> Video Input Caputred. When set to '1', indicates that active video was captured. This field is updated at the beginning of each vsync.

This bit is cleared through the **vinvsynciclr** bit (see **VICLEAR** on page 4-251) or upon a video in soft or a hardreset.

rawvbicapd <10> Raw VBI Captured. When set to '1', indicates that raw VBI was captured.This field is updated at the beginning of each vsync.

This bit is cleared through the **vinvsynciclr** bit (see **VICLEAR** on page -251) or upon a video in soft or a hardreset.

slcvbicapd <11> Slice VBI Captured. When set to '1', indicates that sliced VBI was captured.This field is updated at the beginning of each vsync.

This bit is cleared through the **vinvsynciclr** bit (see **VICLEAR** on page -251) or upon a video in soft or a hardreset.

codecstalled <12> Codec Transfers Stalled. When **codectransen** is set to '0', **codecstalled** goes to '1' when the Codec Interface's fifo is empty and it has stalled. This bit is valid in both compression and decompression transfers.

This bit is reset through the **codecen** bit in the **CODECCTL** register or upon resuming compressed data transfers by setting **codectransen** back to '1'.

Reserved: <7:4> <31:13>

Reading this field has no effect.

Chapter 5: Programmer's Specification

This chapter includes:

5.1 Host Interface

5.1.1 Introduction

The MGA-G100 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 64-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 internal video expansion devices.

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 or CACHE are used for different classes of access.

Access	*Type*	*BFIFO*	*MIFIFO*	*CACHE*
Configuration registers	R W			
ROM	R W		W W	R
DMAWIN or **MGABASE3**	R W	 W		
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 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').

➾ Note :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.

5.1.4 PCI Target-Abort Generation

The MGA-G100 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-G100.

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-G100'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-G100 to correct this situation.

The exact action that will be taken depends on the data size (the MGA-G100 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 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-G100 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-G100 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-G100 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-G100 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:

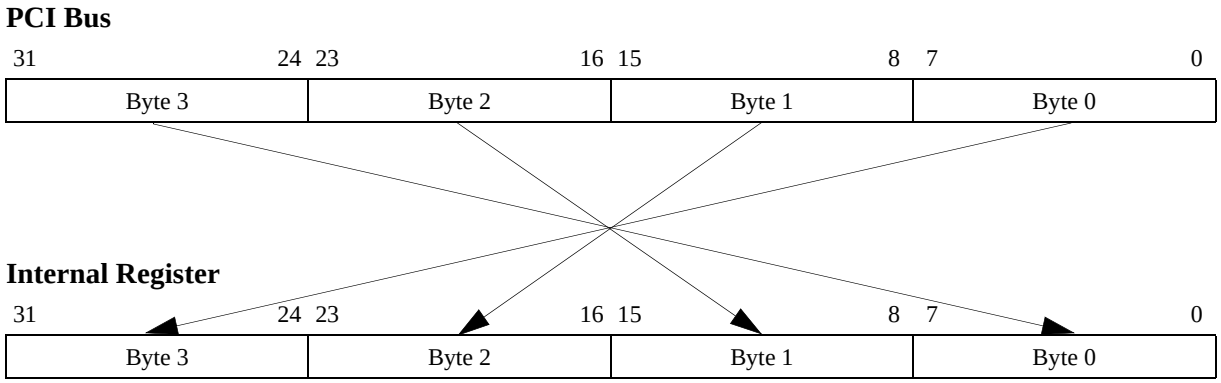
- The VGA compatibility section. Data in this section is 8 bits wide.
- The DAC. Data in this section is 8 bits wide.
- External devices. In this case, the width of the data cannot be known in advance.

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.

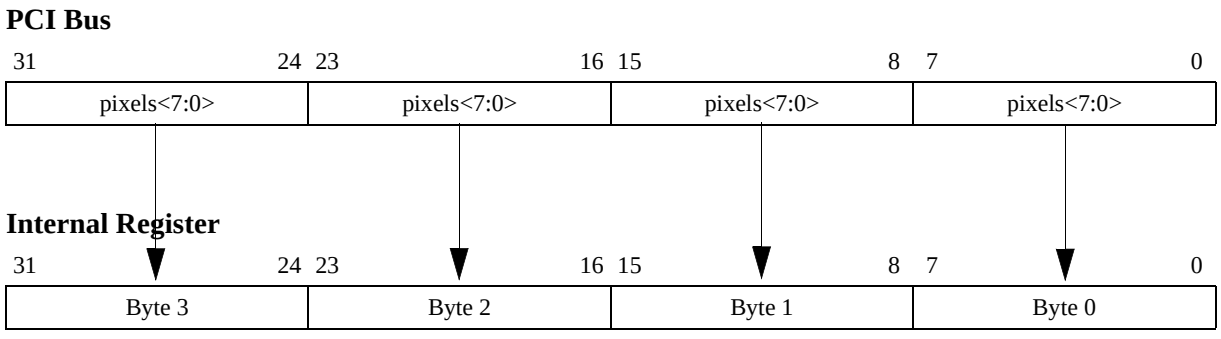
Setting the **powerpc** bit ensures that a 2-bit access by a big endian processor will load the correct data into a 32-bit register. In other words, when data is treated as 32 bit-quantities, it will appear in the identical way to both little and big endian processors. Note however that byte and word 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

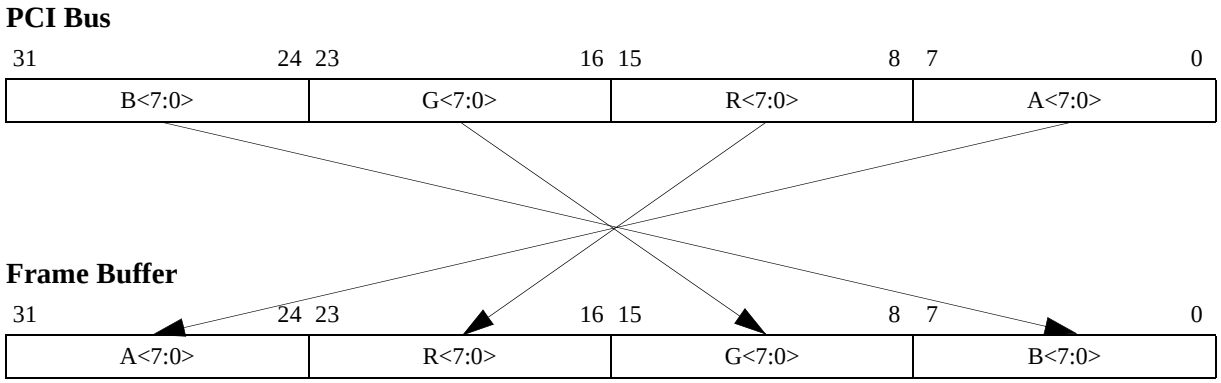


Frame Buffer Space

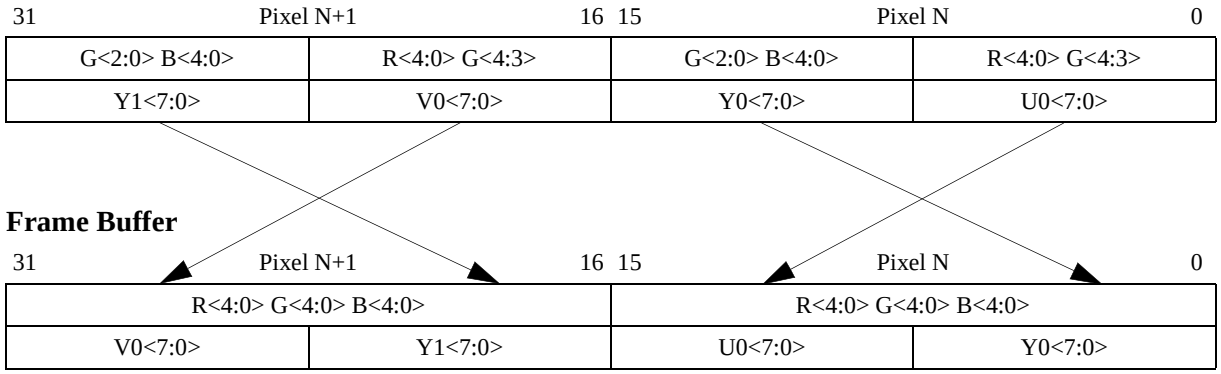
The frame buffer is organized in little endian format, and byte swapping depends on the size of the pixel. As usual, addresses are not modified.

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.

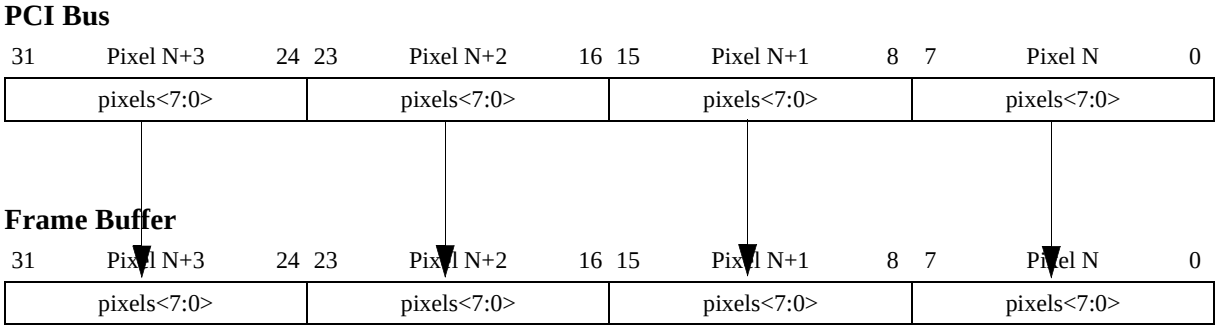
32 bits/pixel, dirDataSiz = 10



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, and ILOAD 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 operations. Refer to Table 5-3 on page 5-54 and Table 5-9 on page 5-89.

	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 operations. Refer to Table 5-3 on page 5-54 and Table 5-9 on page 5-89.

	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 operation. Refer to Table 5-3 on page 5-54 and Table 5-9 on page 5-89.

	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 operation. Refer to Table 5-3 on page 5-54 and Table 5-9 on page 5-89.

	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 operation. Refer to Table 5-3 on page 5-54, Table 5-5 on page 5-56, and Table 5-9 on page 5-89.

	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 operation. Refer to Table 5-3 on page 5-54, Table 5-5 on page 5-56, and Table 5-9 on page 5-89.

	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 5-57.

	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 5-57.

	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 operation. Refer to Table 5-3 on page 5-54, Table 5-5 on page 5-56, and Table 5-9 on page 5-89.

	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 operation. Refer to Table 5-3 on page 5-54, Table 5-5 on page 5-56, and Table 5-9 on page 5-89.

	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 5-54 and Table 5-5 on page 5-56.

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 5-54 and Table 5-5 on page 5-56.

	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 5-54 and Table 5-5 on page 5-56.

	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 5-54 and Table 5-5 on page 5-56.

	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 5-57.

	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 5-57.

	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 5-57.

	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 5-57.

	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 5-55.

	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 5-55.

	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 5-55.

	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	:																															

5.1.9 Programming Bus Mastering for DMA Transfers

The **busmaster** field of the **DEVCTRL** register enables bus mastering. If this bit is disabled, the MGA-G100 will not perform bus mastering, even if the other bus mastering registers are accessed. Disabling **busmaster** does not preclude writing to the host registers related to bus mastering.

The bus mastering feature allows the MGA-G100 to access the system memory through a DMA channel (used in conjunction with Pseudo-DMA mode). In order to send 3D commands to the chip, General Purpose Pseudo-DMA should be used. For texture mapping transfers between system memory and the off-screen area, ILOAD Pseudo-DMA mode should be used (this is the suggested usage - any Pseudo-DMA mode can actually be used for either case).

The DMA channel is built with two sets of registers, as well as interrupt control and status bits. The two sets of registers identify the system memory area to be used for the primary and secondary DMA channels.

- The primary DMA registers are accessible through the host register's base address. They are readable and writable.
- The secondary DMA registers are accessible only by a primary DMA transfer for writes, and through the drawing register base addresses for reads. The secondary DMA registers cannot be written directly through the drawing register base addresses or through the DMAWIN base address, nor can they be written to by a secondary DMA transfer.

5.1.9.1 DMA Registers

Primary Current Address (PRIMADDRESS)

This register must be initialized with the starting address of the primary DMA channel in the system memory area. The two LSBs of this register specify the Pseudo-DMA mode to be used for transfers.

Primary End Address (PRIMEND)

This register must be initialized with the ending address of the primary DMA channel in the system memory area.

Secondary Current Address (SECADDRESS)

This register must be initialized with the starting address of the secondary DMA channel in the system memory area. The two LSBs of this register specify the Pseudo-DMA mode to be used for transfers. This register is accessed using General Purpose Pseudo-DMA mode (primary DMA transfer) in order to be able to start secondary DMA transfers while the primary DMA channel is active.

Secondary End Address (SECEND)

This register must be initialized with the ending address of the secondary DMA channel in the system memory area. It is accessed using General Purpose Pseudo-DMA mode (primary DMA transfer) in order to be able to start secondary DMA transfers while the primary DMA channel is active.This register also contains a bit, **sagpxfer**. This bit is utilized to execute the secondary DMA transfer with PCI protocol or AGP protocol.

Soft Trap Interrupt (SOFTRAP)

This register is useful when the secondary DMA channel cannot be used (due to a system or memory constraint, or other reason). When **SOFTRAP** is written, the MGA-G100 will generate an interrupt (if the **IEN** register is set). In the context of texture mapping, this register must be written with the handle of the texture so that the interrupt handler can know where the texture is located in system memory. Writing this register stops primary DMA transfers. To restart the primary DMA channel, the **PRIMEND** register must be rewritten.

Interrupt Clear (ICLEAR)

Interrupt clearing. This register clears the pending soft trap interrupt.

Interrupt Enable (IEN)

Interrupt enable. This register allows the pending soft trap interrupt to be seen on the PINTA/ line.

Status (STATUS)

End of primary DMA channel status bit and soft trap interrupt pending bit. Use of the primary DMA channel is complete when both the primary and secondary DMA transfers are finished.

5.1.9.2 Using the DMA Channel

To use the DMA channel, follow this sequence:

1. Write a list of commands to a buffer in main memory.
2. Write the starting address of the primary buffer in memory to the **PRIMADDRESS** register. Since this is a 32-bit pointer, the two LSBs are not used as an address but rather as an indication of the type of Pseudo-DMA transfer to be used.
3. Write the address of the first dword after the end of the primary buffer to the **PRIMEND** register.
4. As soon as the **PRIMEND** register is accessed, the primary DMA channel will be activated.
5. A read access will be performed on the host bus at the location pointed to by **PRIMADDRESS**. The data that is read will be fed to the 7K Pseudo-DMA window (which is internal to the chip). **PRIMADDRESS** will be incremented to point to the next dword.
6. If, within this process the host requires the second level of Pseudo-DMA, then the **SECADDRESS** register must be written with the starting address of the secondary buffer in memory and the Pseudo-DMA mode to be used, then the **SECEND** register must be written. In this case, steps 7 to 9 will be taken; if not, operations continue at step 10.

   ➽ ***Note:*** It is not permitted to set SECEND to the same value as SECADDRESS.

7. Read accesses will be performed on the host bus at the location pointed by **SECADDRESS**. The data that is read will be fed to the 7K Pseudo-DMA window (which is internal to the chip). The secondary current address will be incremented to point to the next dword.
8. The **SECADDRESS** and **SECEND** registers cannot be accessed while they are being used by the secondary DMA channel. This will produce unpredictable results.
9. **SECADDRESS** and **SECEND** are compared. If they are different, the secondary DMA continues (refer to step 7). If they are equal, the secondary DMA is finished and the primary DMA goes on. It should be noted that when the primary DMA resumes, the selected Pseudo-DMA mode restarts. For example, if the General Purpose Pseudo-DMA mode is selected, the first dword fetch will be interpreted as a set of four register indexes.
10. **PRIMADDRESS** is compared with **PRIMEND**. If they are different, the DMA transfer continues (refer to step 5). If they are equal, the DMA transfer is complete.
11. If the **SOFTRAP** register is accessed in the primary DMA channel, the primary DMA transfer will stop and an interrupt will be generated (see the **softrapen** field). In the context of texture mapping, the **SOFTRAP** register must be written with the handle of the texture so that the interrupt routine will know what action to take. To restart the primary DMA channel, the **PRIMEND** register must be written. Until **PRIMEND** is written, any operation can be undertaken with the MGA-G100. If the Pseudo-DMA window is accessed before **PRIMEND** is written, it will be controlled by the **dmamod** field of the **OPTION** register (when the **SOFTRAP** register is written, a DMA reset will occur).

AGP Funtionality

MGA-G100 has the ability to perform DMA transfers with AGP or PCI protocol. Primary DMA transfers

can only be done with PCI protocol. To enable AGP transfers on the secondary DMA channel the AGP_CMD register and the **sagpxfer** field in the **SECEND** register must be programmed.

Special Cases

The PCI Specification indicates that when the MGA-G100 acts as a master on the PCI bus, two particular circumstances can arise (aside from the regular transfer of data):

1. The first case is a ***master-abort***, which occurs when an access is done and no device responds to it (often due to a glitch in programming). When this happens, the **recmastab** bit of the **DEVCTRL** register will be set (and will remain set until a '1' is written to it).
2. The second case is a ***target-abort***. This is a target termination that is used when the target detects a problem with an access generated by the MGA-G100. The MGA-G100 always generates its accesses in the correct form, so this situation really depends on the target. Target-aborts should not occur, since the PCI Specification indicates that they occur with ***I/O*** accesses, while the MGA-G100 generates ***memory*** accesses. There is no possible way to change the MGA-G100 or its programming to prevent a target-abort from occurring. If this happens, the **rectargab** bit of the **DEVCTRL** register will be set (and will remain set until a '1' is written to it).

The software must write to the **softreset** bit of the **RST** register when either a master-abort or a target-abort occurs (the **SECADDRESS** register will indicate this) to reset the DMA channel and the BFIFO. This must also be done when a warm boot occurs (on a PC, when Ctrl+Alt+Del is pressed).

Reset of the Pseudo-DMA sequence:

A reset of the Pseudo-DMA sequence will be generated under the following conditions:

1. When the **PRIMADDRESS** register is written.
2. When the **SECEND** register is written, except where **SECEND** is written with the same value as **SECADDRESS**.
3. When the **SOFTRAP** register is written.
4. When secondary DMA transfers end (that is, when **SECADDRESS** becomes equal to **SECEND** at the end of the secondary DMA, and not when **SECADDRESS** is written with the **SECEND** value).
5. When a master-abort or target-abort is detected.

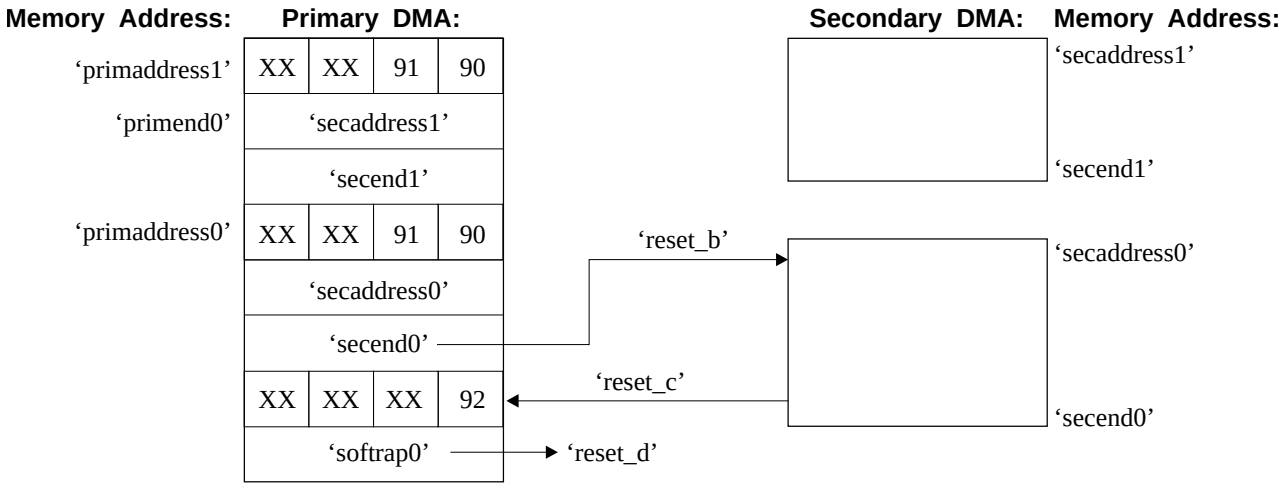
- Note that there is no reset of the Pseudo-DMA sequence when **PRIMEND** is written, since **PRIMEND** starts the primary DMA transfers, and this can happen more than once to extend the list (even while the list is still being transferred).
- Additionally, there is no reset of the Pseudo-DMA sequence when primary DMA transfers end. If commands are added to the primary display list, **PRIMEND** simply has to be written with its new value to restart the primary DMA transfers.

  This means that if you intend to write **PRIMEND** more than once (without re-writing **PRIMADDRESS**) you must fill the last set of Pseudo-DMA transfers with no-ops (reserved registers). Otherwise, the Pseudo-DMA transfers will restart at the last Pseudo-DMA location (either index or data when in General Purpose Pseudo-DMA mode).

Example:

- When **PRIMADDRESS** is written with 'primaddress0', a reset of the Pseudo-DMA sequence is executed.
- When **SOFTRAP** is written (through a primary DMA transfer), another reset is executed ('reset_d').
- When **SECEND** is written, and when the secondary DMA ends, two other resets are executed

(‘reset_b’ and ‘reset_c’).



Additional Information

- When the DMA channel is used (mastering), it is possible to know which parts of the buffer have been executed by the drawing engine, since the DMA current pointer is readable by the CPU through the **PRIMADDRESS** or **SECADDRESS** registers.
- The **endprdmasts** field of the **STATUS** register always indicates whether or not the primary and secondary DMA channels have been read completely (**PRIMADDRESS** = **PRIMEND** and **SECADDRESS** = **SECEND**). It is ***also*** set to ‘1’ when a soft trap interrupt occurs. This bit toggles to ‘0’ as soon as the **PRIMEND** register is written.
- To get an interrupt when the primary DMA channel terminates, just include a write to the **SOFTRAP** register as the last DMA transfer.
- The data read from the DMA channel is swapped according to the setting of the **dmadatasiz** field of the **OPMODE** register.

5.2 Memory Interface

5.2.1 Frame Buffer Organization

The MGA-G100 supports up to four 2 MByte banks of memory (using 8 Mbit devices) or up to four 4MByte of memory (using 16 Mbit devices). Using this configuration, it is possible to design a 2, 4, 6, 8, 12 or 16 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.

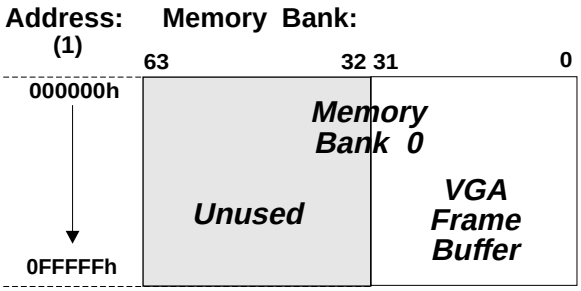
Supported Resolutions

	Single Frame Buffer mode				***Single Z Buffer***							
	No Z				***Z 16 bits***				***Z 32 bits***			
Resolution	***8-bit***	***16-bit***	***24-bit***	***32-bit***	***8-bit***	***16-bit***	***24-bit***	***32-bit***	***8-bit***	***16-bit***	***24-bit***	***32-bit***
640 x 480	2M	2M	2M	2M	2M	2M	-	2M	2M	2M	-	4M
720 x 480	2M	2M	2M	2M	2M	2M	-	2M	2M	2M	-	4M
800 x 600	2M	2M	2M	2M	2M	2M		4M	4M	4M		4M
1024 x 768	2M	2M	4M	4M	4M	4M		8M	4M	8M		8M
1152 x 864	2M	2M	4M	4M	4M	4M	-	8M	8M	8M	-	8M
1280 x 1024	2M	4M	4M	8M	4M	8M	-	8M	8M	8M	-	10M
1600 x 1200	2M	4M	8M	8M	8M	8M		12M	10M	12M		16M

The memory type must be properly initialized at power up. In normal operation, the **memconfig** fields should not change, as it is used to determine the ***type*** of memory. However, **splitmode** 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<13:12>** are set according to the illustration on the left side to produce the configuration depicted on the right side.

■ SGRAM, 8 Mbit devices

OPTION<13:12>

splitmode	memconfig
0	0

Address	Memory Bank
000000h – 1FFFFFh	*SGRAM Bank 0*
200000h – 3FFFFFh	*SGRAM Bank 1 (optional)*
400000h – 5FFFFFh	*SGRAM Bank 2 (optional)*
600000h – 7FFFFFh	*SGRAM Bank 3 (optional)*

■ SGRAM, 16 Mbit devices

OPTION<13:12>

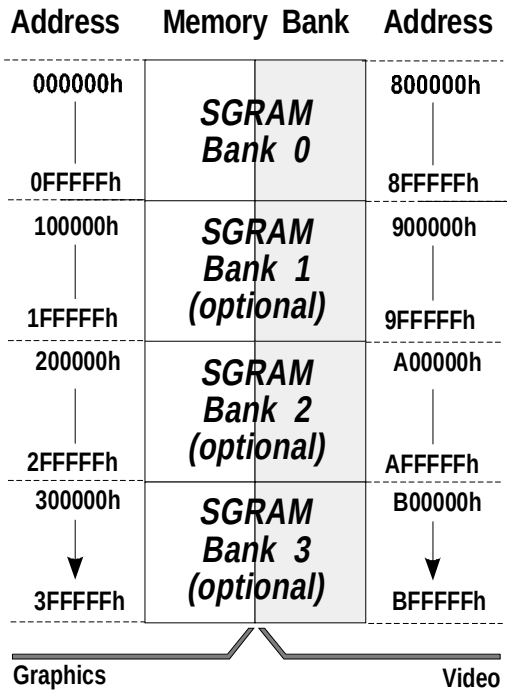
splitmode	memconfig
0	1

Address	Memory Bank
000000h – 3FFFFFh	*SGRAM Bank 0*
400000h – 7FFFFFh	*SGRAM Bank 1 (optional)*
800000h – BFFFFFh	*SGRAM Bank 2 (optional)*
C00000h – FFFFFFh	*SGRAM Bank 3 (optional)*

■ SGRAM, split mode, 8 Mbit devices

OPTION<13:12>

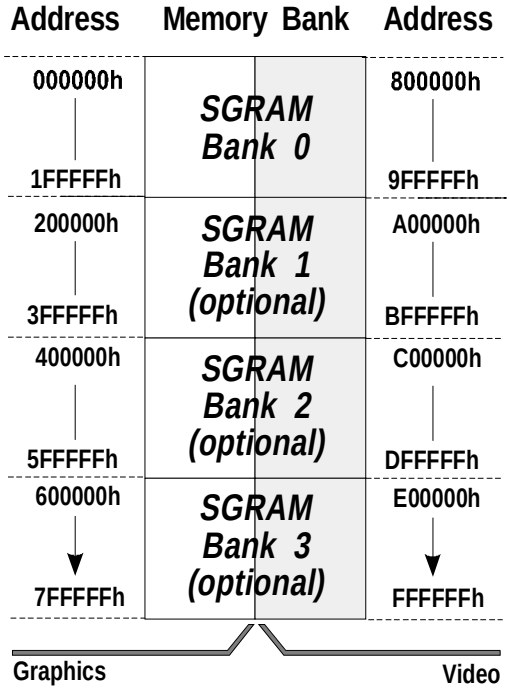
splitmode	memconfig
1	0



■ SGRAM, split mode, 16 Mbit devices

OPTION<13:12>

splitmode	memconfig
1	1

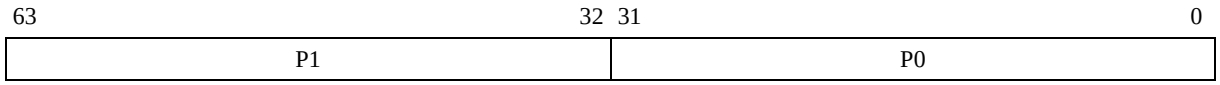


5.2.2 Pixel Format

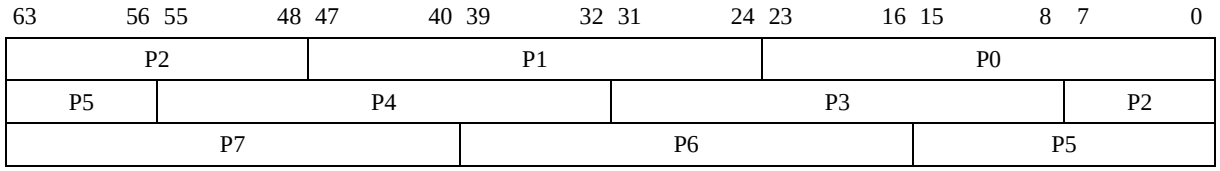
The slice is 64 bits long and is organized as follows. In all cases, the least significant bit is 0. The Alpha part of the color is the section of a pixel that is not used to drive the DAC. Note that the data is always true color, but in 8 bit/pixel formats pseudo color can be used when shading is not 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.

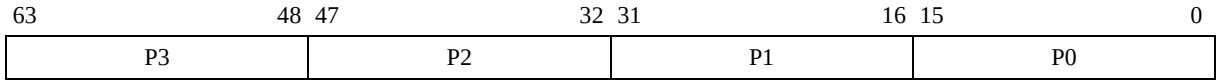
32 bits/pixel



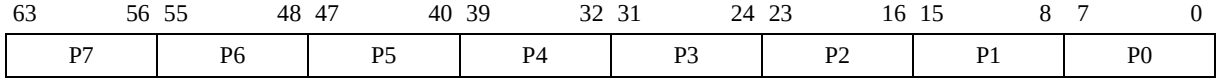
24 bits/pixel



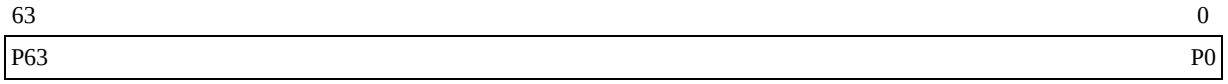
16 bits/pixel



8 bits/pixel

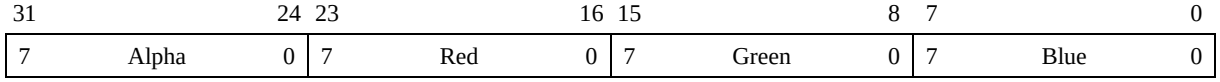


Monochrome

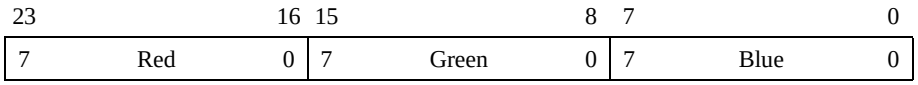


For each of these modes, the pixels are arranged as follows:

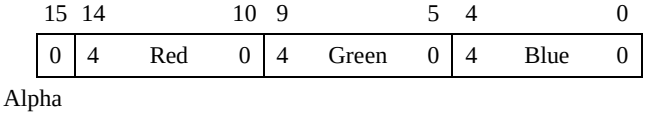
32 bits/pixel



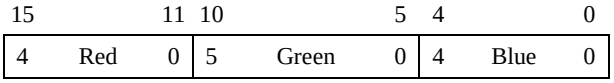
24 bits/pixel



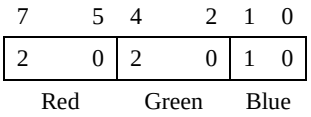
16 bits/pixel (5:5:5)



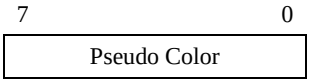
16 bits/pixel (5:6:5)



8 bits/pixel



8 bits/pixel



5.3 Chip Configuration and Initialization

5.3.1 Reset

The MGA-G100 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** or **OPTION2** registers.

Strap Name	*Pins*	*Description*
biosen	HDATA<1>	Indicates whether a BIOS ROM is installed (‘1’) or VPD ROM is installed (‘0’). The biosen strap also controls the **biosen** field of the **OPTION** register.
vgaboot	HDATA<0>	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 4-93 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.

Video In has its own soft reset , see the **VINCTL** register.

Codec Interface has its own soft reset, see the **CODECCTL** register.

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, MCLK, VCLK, and PIXCLK) will be based on the PCI bus clock and enabled. The FMCLK 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 25.159 MHZ when the PLL is powered up.
- The system PLL registers will program the system PLL to oscillate at 142.714 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-G100'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.
- 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	'000000'	The refresh counter defines the rate of MGA memory refresh. For a typical 71.5 MHz MCLK, a value of '11h' 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-G100 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 FMCLK to 142.714 MHz, MCLK to 71.357 MHz, GCLK to 71.357 MHz, and PIXCLK to 25.159 MHz. Refer to Section 5.7.8.3 on page 5-82.

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 Initialization

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

Step 2. Program the fields 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** and **jedecrst** fields of **MACCESS**.

Step 6. Start the refresh by programming the **rfhcnt** field of the **OPTION** register.

5.3.4 Operation Mode Selection

The MGA-G100 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 2)

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
108	VGA	80x60 Text	640x480	16
10A	VGA	132x43 Text	1056x350	16
109	VGA	132x25 Text	1056x400	16
10B	VGA	132x50 Text	1056x400	16
10C	VGA	132x60 Text	1056x480	16
100	SVGA	Packed-pixel 8 bpp	640x400	256
101	SVGA	Packed-pixel 8 bpp	640x480	256
110	SVGA	Packed-pixel 16 bpp	640x480	32K
111	SVGA	Packed-pixel 16 bpp	640x480	64K
112	SVGA	Packed-pixel 32 bpp	640x480	16M
102	SVGA	Multi-plane 4 bpp	800x600	16
103	SVGA	Packed-pixel 8 bpp	800x600	256
113	SVGA	Packed-pixel 16 bpp	800x600	32K
114	SVGA	Packed-pixel 16 bpp	800x600	64K
115	SVGA	Packed-pixel 32 bpp	800x600	16M
105	SVGA	Packed-pixel 8 bpp	1024x768	256

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

Mode	*Type*	*Organization*	*Resolution*	*No. of colors*
116	SVGA	Packed-pixel 16 bpp	1024x768	32K
117	SVGA	Packed-pixel 16 bpp	1024x768	64K
118(1)	SVGA	Packed-pixel 32 bpp	1024x768	16M
107	SVGA	Packed-pixel 8 bpp	1280x1024	256
119(1)	SVGA	Packed-pixel 16 bpp	1280x1024	32K
11A(1)	SVGA	Packed-pixel 16 bpp	1280x1024	64K
11B(1)	SVGA	Packed-pixel 32 bpp	1280x1024	16M
11C	SVGA	Packed-pixel 8 bpp	1600x1200	256
11D(1)	SVGA	Packed-pixel 16 bpp	1600x1200	32K
11E(1)	SVGA	Packed-pixel 16 bpp	1600x1200	64K

(1) Possible only with a frame buffer of 8 megabytes.

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-G100 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-G100'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 operation 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

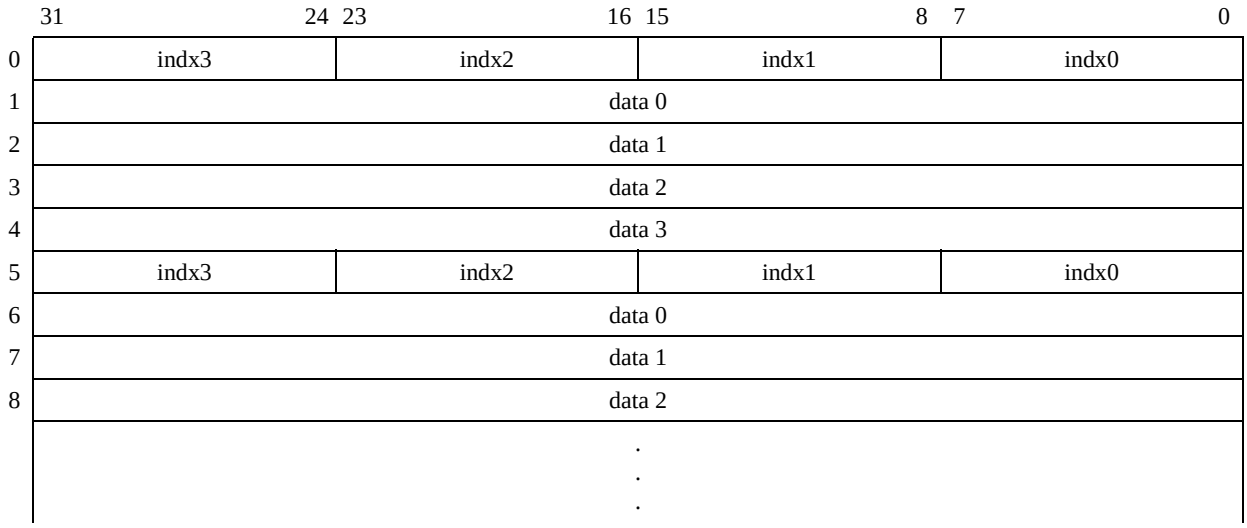
The general purpose Pseudo-DMA operations are performed through the DMAWIN aperture in the MGA control register space, or in the 8 MByte Pseudo-DMA window. It is recommended that host CPU instructions be used in such a way that each transfer increments the address. This way, the PCI bridge can proceed 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



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 4-63).

   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), dithering mode and z presicion(16 or 32 bits)	Used in those primitives listed • Lines with depth • Gouraud shaded traps/ Rectangle fills • Texture mapping • Unformatted ILOAD
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 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:

Reserved	**transc**	**pattern**	**bltmod**				**Reserved**	**trans**				**bop**				**Reserved**	**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:

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

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

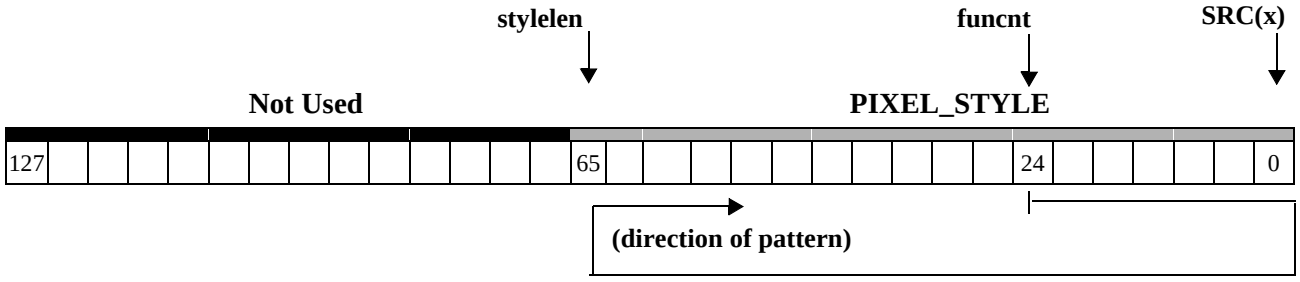
Register	*Function*	*Comment / Alternate Function*
SHIFT	Linestyle length (stylelen), linestyle start point within the pattern (funcnt)	
SRC0	Linestyle pattern storage	
SRC1	Linestyle pattern storage	If **stylelen** is from 32-63
SRC2	Linestyle pattern storage	If **stylelen** is from 64-95
SRC3	Linestyle 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

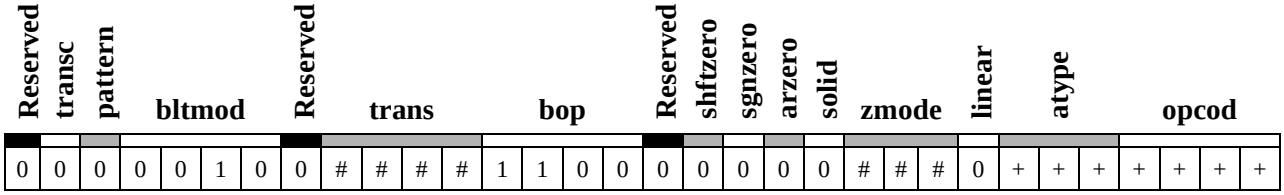
SRC0 : **srcreg0** = PIXEL_STYLE(31:0)
SRC1 : **srcreg1** = PIXEL_STYLE(63:32)
SRC2 : **srcreg2** = PIXEL_STYLE(65:64)



- 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:



- **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 (if zwidth = 0) **DR0_Z32LSB, DR0_Z32MSB** (if zwidth = 1)	The z start position	Only if **zmode** <> NOZCMP or **atype** = ZI
DR2 (if zwidth = 0) **DR2_Z32LSB, DR2_Z32MSB** (if zwidth = 1)	The z major increment	Only if **zmode** <> NOZCMP or **atype** = ZI
DR3 (if zwidth = 0) **DR3_Z32LSB, DR3_Z32MSB** (if zwidth = 1)	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 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

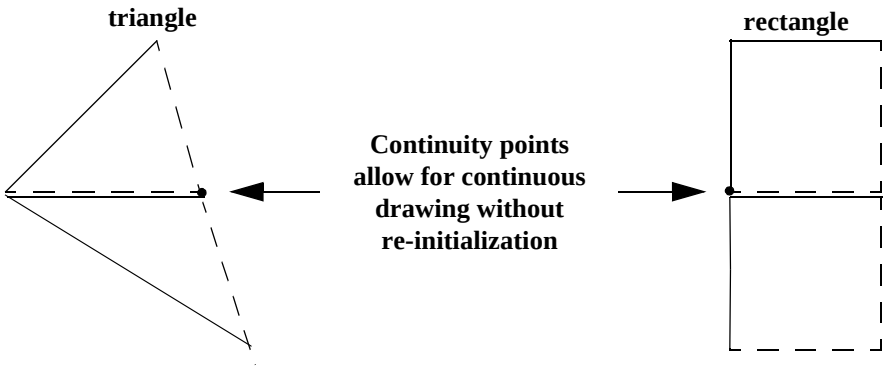
The following subsections list the registers that must be specifically programmed for constant and Gouraud shaded, patterned, and textured trapezoids, including rectangle and span line fills. Remember to program the registers listed in section Section 5.5.3 and in the tables in subsection 5.5.5.1 first. 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:

	Reserved	**transc**	**pattern**	**bltmod**				**Reserved**	**trans**				**bop**				**Reserved**	**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 4-63).

5.5.5.3 Patterned Trapezoids / Rectangle Fills

DWGCTL:

	Reserved	transc	pattern	bltmod				Reserved	trans			bop				Reserved	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

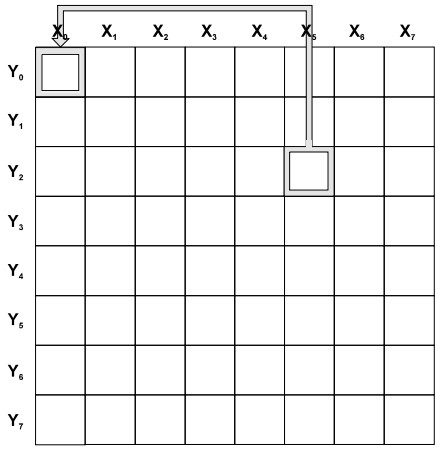
- **transc:** if **atype** is BLK, an opaque background is not supported- the value of **transc** must be '1'
- **trans**: if **atype** is BLK, the transparency pattern is not supported - the value of **trans** must be '0000'
- **bop**: uses any Boolean operation if **atype** is RSTR; if **atype** is 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

Y coordinates \ *X 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

Y coordinates \ *X 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

Y coordinates \ *X 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

Y coordinates \ *X 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:

	Reserved	**transc**	**pattern**	**bltmod**				**Reserved**	**trans**				**bop**				**Reserved**	**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 (if zwidth = 0) **DR0_Z32LSB,** **DR0_Z32MSB** (if zwidth = 1)	The z start position	Only if **zmode** <> NOZCMP or **atype** = ZI
DR2 (if zwidth = 0) **DR2_Z32LSB,** **DR2_Z32MSB** (if zwidth = 1)	The z increment for x	Only if **zmode** <> NOZCMP or **atype** = ZI
DR3 (if zwidth = 0) **DR3_Z32LSB,** **DR3_Z32MSB** (if zwidth = 1)	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:

	Reserved	transc	pattern	bltmod				Reserved	trans				bop				Reserved	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 (if zwidth = 0) **DR0_Z32LSB, DR0_Z32MSB** (if zwidth = 1)	The z start position	Only if zmode <> NOZCMP or atype = ZI
DR2 (if zwidth = 0) **DR2_Z32LSB, DR2_Z32MSB** (if zwidth = 1)	The z increment for x	Only if zmode <> NOZCMP or atype = ZI
DR3 (if zwidth = 0) **DR3_Z32LSB, DR3_Z32MSB** (if zwidth = 1)	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 5-51.

5.5.5.6 Texture Mapping

DWGCTL

	Reserved	transc	pattern	bltmod	Reserved	trans	bop	Reserved	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 1 0
RECT	0	0	0	0 0 0 0	0	# # # #	1 1 0 0	0	1	1	1	0	# # #	0	+ + +	0 1 1 0

- **atype**: must be either ZI or I

Register	*Function*	*Comment / Alternate Function*
TEXWIDTH	Texture width, width mask, and round-up factor	
TEXHEIGHT	Texture height, height mask, and round-up factor	
TEXORG	Texture base address	
TEXCTL	Texel width, texture pitch, texture alpha key, texture alpha mask, alphazeroextend, texture pitch linear, palette select, decal with color key, **clampmode**, **tmodulate**, **strans**, **itrans,**	
TEXTRANS	Transparency color key, texture keying mask	
TMR0	s/wc increment for x	
TMR1	s/wc increment for y	
TMR2	t/wc increment for x	
TMR3	t/wc increment for y	
TMR4	q/wc increment for x	
TMR5	q/wc increment for y	
TMR6	s/wc start value	
TMR7	t/wc start value	
TMR8	q/wc start value	
FCOL	Alpha value	Only if **pwidth** = 32, or **pwidth** = 16 and **dit555** = 1.
DR0 (if zwidth = 0) **DR0_Z32LSB, DR0_Z32MSB** (if zwidth = 1)	Z start position	Only if **zmode** <> NOZCMP or **atype** = ZI
DR2 (if zwidth = 0) **DR2_Z32LSB, DR2_Z32MSB** (if zwidth = 1)	Z increment for x	”
DR3 (if zwidth = 0) **DR3_Z32LSB, DR3_Z32MSB** (if zwidth = 1)	Z increment for y	”
DR4	Red start value	Only if modulate or decal is used
DR6	Red increment on x axis	”

Register	*Function*	*Comment / Alternate Function*
DR7	Red increment on y axis	”
DR8	Green start value	”
DR10	Green increment on x axis	”
DR11	Green increment on y axis	”
DR12	Blue start value	”
DR14	Blue increment on x axis	”
DR15	Blue increment on y axis	”
FOGSTART	The fog factor start value	only if fogen = 1
FOGXINC	The fog factor increment for x	only if fogen = 1
FOGYINC	The fog factor increment for y	only if fogen = 1
FOGCOL	The fog color	only if fogen = 1
ALPHASTART	The alpha start value	see not below
ALPHAXINC	The alpha increment on major axis	see not below
ALPHAYINC	The alpha increment on diagonal axis	see not below
ALPHACTRL	astipple, alphasel	
TEXFILTER	filter	

- Note that the **MACCESS** register’s **pwidth** field must not be set to 24 bits per pixel (PW24) when drawing texture map trapezoids.
- If **twidth** = TW4 or TW8, the color palette must be initialized.
- **ALPHASTART**, **ALPHAXINC**, **ALPHAYINC** are to be programmed when **astipple** field in the register **ALPHACTRL** is set to ‘1’.
- When programming **twmask** (**thmask**) with a value which is not a power of two minus one then:
  - Only clamp mode is supported
  - **tw** (th) must be programmed with the log2 of the texture after it has been rounded up to a power of two.
  - s /wc and t /wc, both xinc and yinc for each, must be scaled by (texture size) / (next power of 2 size)

5.5.5.7 Texture Mapper Used as Video Scaler

Register programming is the same as for Texture Mapping , except:

The perspective effect must be disabled,

TMR4 = 0x00000000

TMR5 = 0x00000000

TMR8 = 0x00010000

The texture engine supports up and down scaling, The video source can be any format the texture engine supports, but is normally TW16.

Note:

tpitchext field is set to the pitch of the image.

npcen field is set to 1.

twmask field of the **TEXWIDTH** register is set to the image width minus one

tw field of the **TEXWIDTH** register is set to the log2 of the video image width after it has been rounded up to a power of two.

thmask field of the **TEXHEIGHT** register is set to the image height minus one.

th field of the **TEXHEIGHT** register is set to the log2 of the video image width after it has been rounded up to a power of two.

The s/wc increment along the x axis (**tmr0** field in the **TMR0** register) should be programmed to

$$\frac{\text{video image width}}{\text{next power of 2 size}} \times \frac{1}{\text{scaling factor in x direction}}$$

The s/wc increment along the y axis (**tmr1** field in the **TMR1** register) should be programmed to ‘0’.

The t/wc increment along the x axis (**tmr2** field in the **TMR2** register) should be programmed to ‘0’

The t/wc increment along the y axis (**tmr3** field in the **TMR3** register) should be programmed to

$$\frac{\text{video image height}}{\text{next power of 2 size}} \times \frac{1}{\text{scaling factor in y direction}}$$

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 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:

	Reserved	transc	pattern	bltmod				Reserved	trans				bop				Reserved	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.

Note: The number of pixels in the source must equal to the number of pixels in the destination.

5.5.6.3 Color Patterning 8 x 8

DWGCTL:

Reserved	transc	pattern	bltmod	Reserved	trans	bop	Reserved	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 5-38, 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:

Reserved	transc	pattern	bltmod	Reserved	trans	bop	Reserved	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:

Reserved	**transc**	**pattern**	**bltmod**				**Reserved**	**trans**				**bop**				**Reserved**	**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 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:

Reserved	transc	pattern	bltmod				Reserved	trans				bop				Reserved	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’

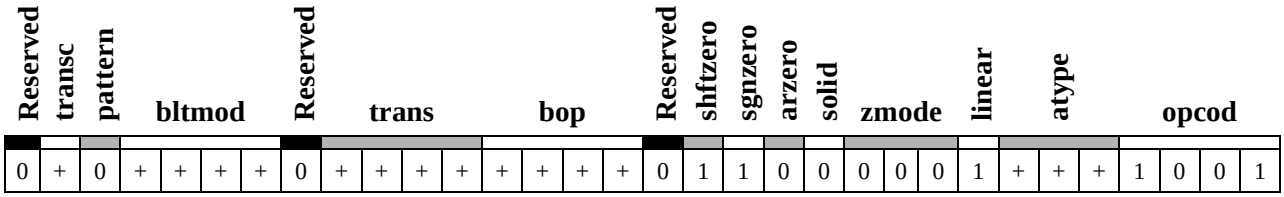
There are some restrictions in the data formats that are supported for this operation.Table 5-3 shows all the valid format combinations. The structure of the buffers to be transferred is defined for each data format (as shown the 'Pixel Formats' illustrations starting on page 5-8).

Table 5-3: ILOAD Supported Formats

Processor Type	bltmod	dmaDataSiz	pwidth	*Data Format*
Little endian	BFCOL	'00'	PW8	8-bit A
			PW16	16-bit A
			PW24	24-bit A
			PW32	32-bit A
	BU24RGB	'00'	PW8	24-bit A
			PW16	24-bit A
			PW32	24-bit A
	BU24BGR	'00'	PW8	24-bit B
			PW16	24-bit B
			PW32	24-bit B
	BU32RGB	'00'	PW8	32-bit A
			PW16	32-bit A
			PW32	32-bit A
	BU32BGR	'00'	PW8	32-bit B
			PW16	32-bit B
			PW32	32-bit B
	BUYUV	'00'	PW8	YUV A
			PW16	YUV A
			PW32	YUV A
		'01'	PW8	YUV B
			PW16	YUV B
			PW32	YUV B
Big endian	BFCOL	'00'	PW8	8-bit B
		'01'	PW16	16-bit B
		'10'	PW32	32-bit A
	BU32RGB	'10'	PW8	32-bit A
			PW16	32-bit A
			PW32	32-bit A
	BU32BGR	'10'	PW8	32-bit B
			PW16	32-bit B
			PW32	32-bit B
	BUYUV	'00'	PW8	YUV C
			PW16	YUV C
			PW32	YUV C
		'01'	PW8	YUV D
			PW16	YUV D
			PW32	YUV D

5.5.7.3 ILOAD with Expansion (Character Drawing)

DWGCTL:



- **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-G100 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

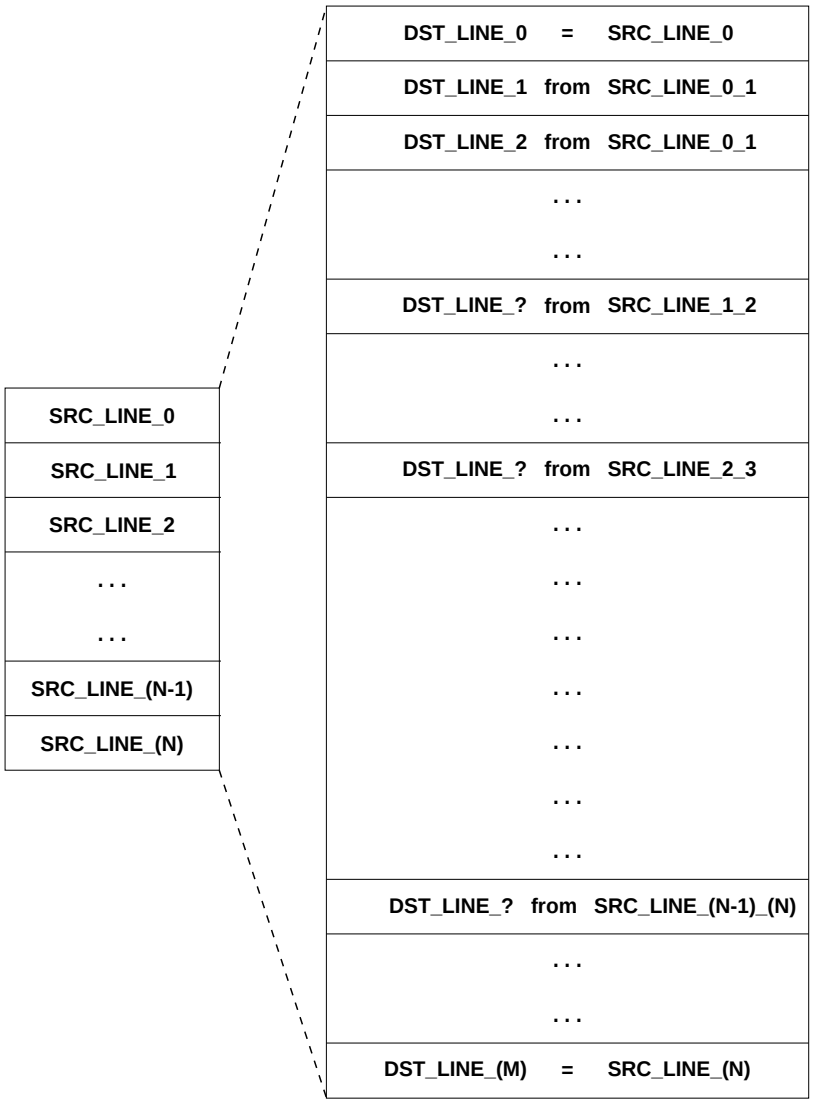
Processor Type	**bltmod**	**dmaDataSiz**	*Data Format*
Little endian	BU24RGB	00	24-bit A
	BU24BGR	00	24-bit B
	BU32RGB	00	32-bit A
	BU32BGR	00	32-bit B
	BUYUV	00	YUV A
	BUYUV	01	YUV B
Big endian	BU32RGB	10	32-bit A
	BU32BGR	10	32-bit B
	BUYUV	00	YUV C
	BUYUV	01	YUV D

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



$$DST\_LINE\_? = \frac{SRC\_BUF\_0\ (16 - beta\ ') + SRC\_BUF\_1\ (beta\ ')}{16}$$

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):

	MSB			LSB	
SRC_BUF_0=	V00	Y01	U00	Y00	Pixel 0_1
	V02	Y03	U02	Y02	Pixel 2_3
	V04	Y05	U04	Y04	Pixel 4_5
	. . .	. . .	. . .	. . .	. . .

	MSB			LSB	
SRC_BUF_1=	V10	Y11	U10	Y10	Pixel 0_1
	V12	Y13	U12	Y12	Pixel 2_3
	V14	Y15	U14	Y14	Pixel 4_5
	. . .	. . .	. . .	. . .	. . .

	MSB			LSB	
DW to Send to the Chip	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):

	MSB			LSB	
SRC_BUF_0=	Y01	V00	Y00	U00	Pixel 0_1
	Y03	V02	Y02	U02	Pixel 2_3
	Y05	V04	Y04	U04	Pixel 4_5
	. . .	. . .	. . .	. . .	. . .

	MSB			LSB	
SRC_BUF_1=	Y11	V10	Y10	U10	Pixel 0_1
	Y13	V12	Y12	U12	Pixel 2_3
	Y15	V14	Y14	U14	Pixel 4_5
	. . .	. . .	. . .	. . .	. . .

	MSB			LSB	
DW to Send to the Chip	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.

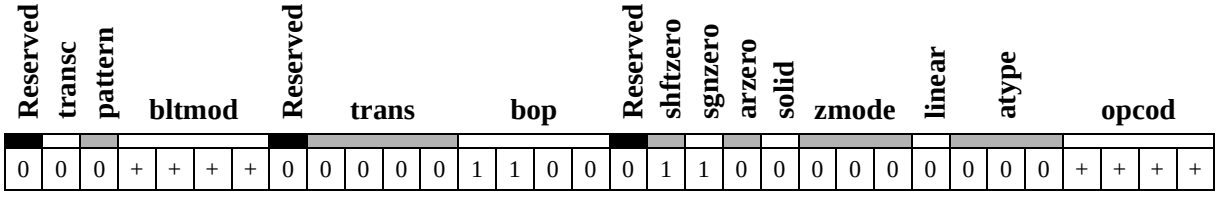
5.5.8.3 Scaling Steps

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 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:



- **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 5-51), 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	1	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	$\frac{(\text{SRC\_X\_DIMEN} - 1) << 16}{(\text{DST\_X\_DIMEN} - 1)} + 1$	
AR3	DST_START_ADDRESS - DST_Y_INC	
AR5	DST_Y_INC	Only required if performing vertical scaling using the BITBLT function.
AR6	$\frac{(\text{SRC\_X\_DIMEN} - \text{DST\_X\_DIMEN}) << 16}{(\text{DST\_X\_DIMEN} - 1)}$	
FXBNDRY	Destination boundary (left and right)	Can use **FXLEFT** and **FXRIGHT**
YDST	Y start position	

5.5.9 Loading the Texture Color Palette

This operation is similar to a normal BITBLT or ILOAD operation. In this case, a source texture color palette is loaded into the destination 256 x 1 x 16 bpp LUT (Look-Up Table) that is used in texture mapping. Any portion of the LUT can be programmed independently.

This color palette is used to perform color expansion during texture mapping when textures are in 4 or 8 bpp format. When 4 bpp textures are used, 16 palettes are available in the LUT. The choice of the palette used to do color expansion is determined by the **palsel** field of the **TEXCTL** register.

	Reserved	**transc**	**pattern**	**bltmod**				**Reserved**	**trans**				**bop**				**Reserved**	**shftzero**	**sgnzero**	**arzero**	**solid**	**zmode**			**linear**	**atype**			**opcod**			
BITBLT	0	0	0	0	1	1	1	0	0	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	1	0	0	0	1	0	0	0
ILOAD	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	1	0	0	0	1	0	0	1

Register	*Function*	*Comment / Alternate Function*
AR0	Source end address	
AR3	Source start address	
PITCH	**iy** = 1024	
YDSTLEN	**yval**: Start position in the LUT (0 to 255) **length**: Number of locations to fill in the LUT (1 to 256)	
YDSTORG	0	
FXBNDRY	0	
MACCESS	**pwidth** = PW16 , **tlutload** = 1	
OPMODE	**dmamod** = 01, **dmadatsiz** = 00 (little endian) **dmadatsiz** = 01 (big endian)	For ILOAD only.

➾ Note: The **PITCH** and **MACCESS** registers are not normally modified during drawing operations, and may have to be re-programmed after this operation in order to return the drawing engine to its original state.

5.6 CRTC Programming

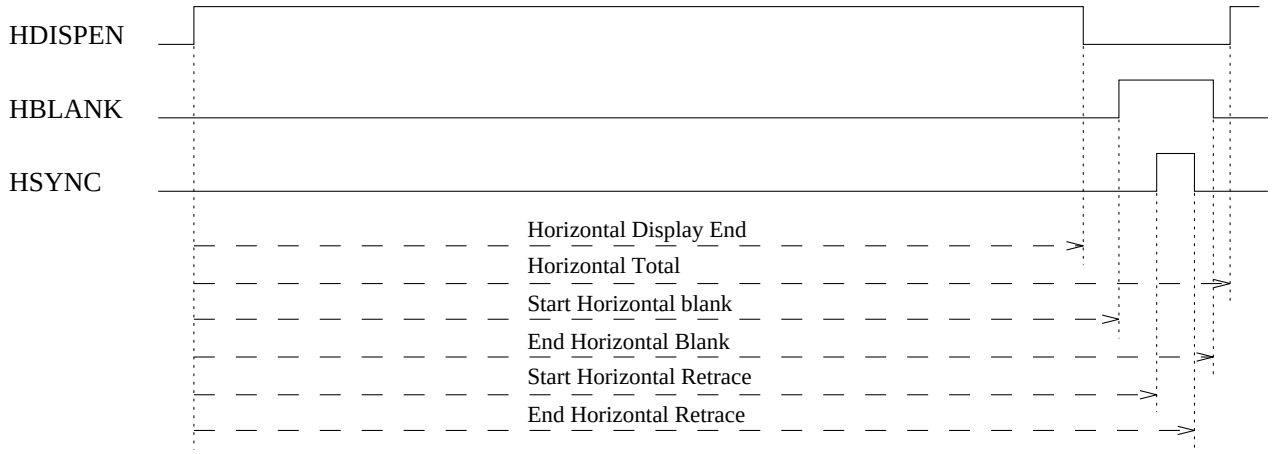
The CRTC can be programmed in one of two modes: VGA mode or Power Graphic mode. The **mgamode** field of the **CRTCEXT3** register is used to select the operating mode.

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'.

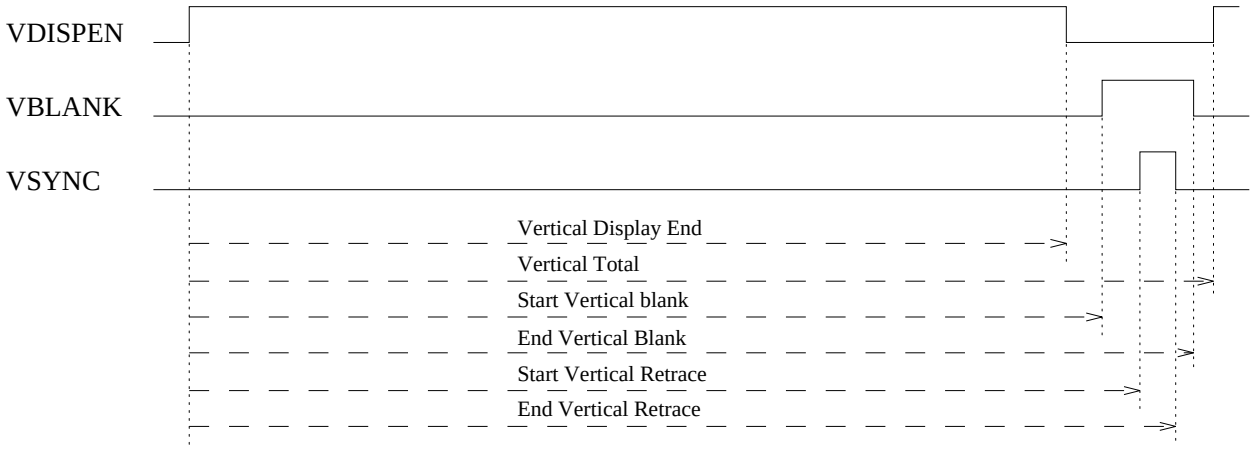
The units of the horizontal counter are ‘character clocks’ for VGA mode, or 8 pixels in Power Graphic mode. The **scale** field of the **CRTCEXT3** register is used to bring the VCLK clock down to an ‘8 pixel’ clock.

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.
curloc<15:0>	Cursor location
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 double slices in one display line.

- The display can be placed in interlace mode if the **interlace** bit of the **CRTCEXT0** register is set to '1'.
- The **curloc**, **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-G100 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 - hdispend** must be greater than 0
- **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-G100'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:

$$\textbf{offset} = \frac{\text{video pitch} * \text{bpp} * \text{fsplit}}{128}$$

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 occurring 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 - hdispend** 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.
- In HZOOM = 00, (Horizontal Total)*(scale+1) MOD 8 MUST NOT be 7.
- In HZOOM = 01, (Horizontal Total)*(scale+1) MOD 16 MUST NOT be 15.
- In HZOOM = 1x, (Horizontal Total)*(scale+1) MOD 32 MUST NOT be 31.

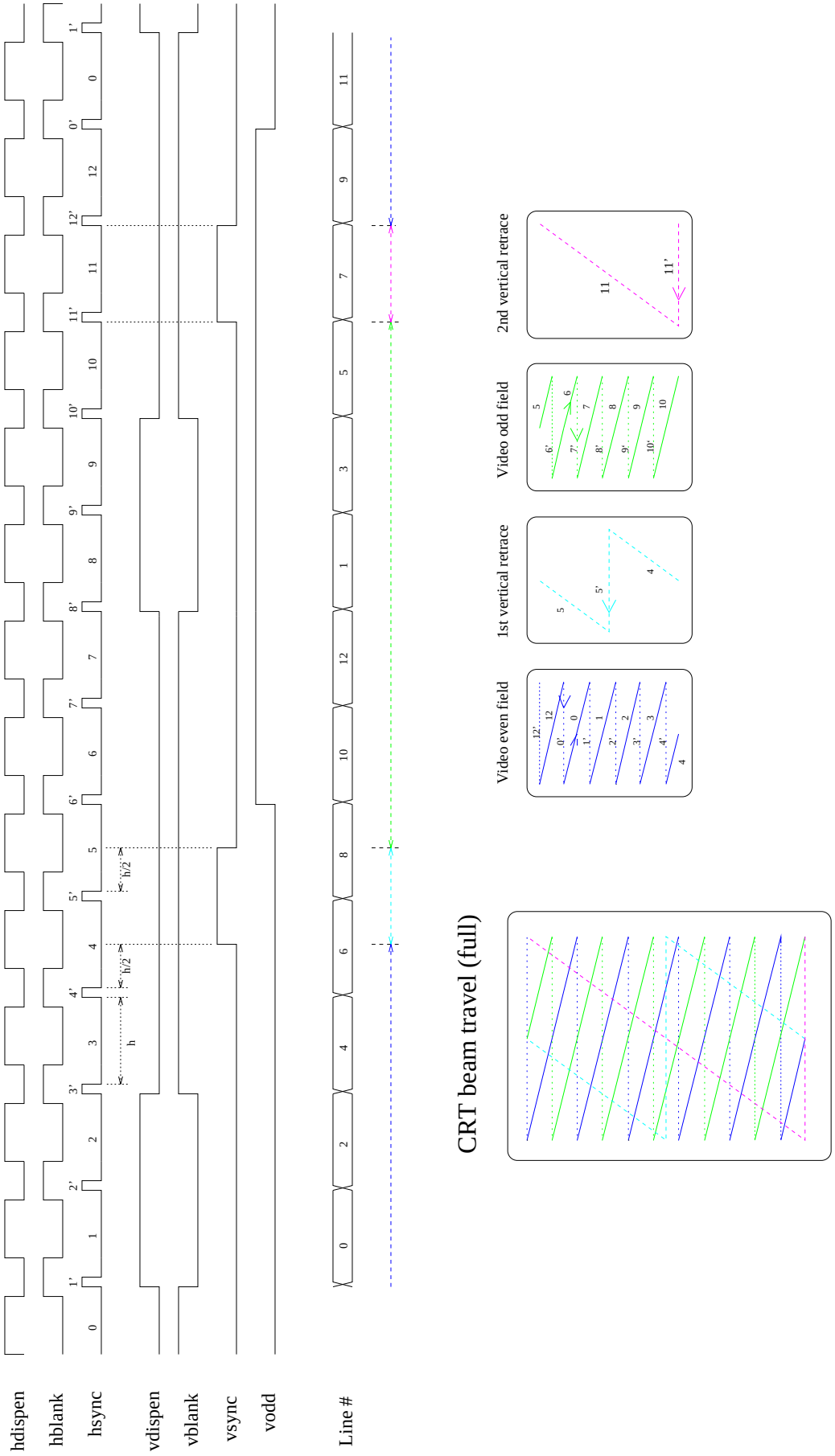
CRTC Latency Formulas

This section presents several rules that must be followed in Power graphic mode in order to adhere to the latency constraints of the MGA-G100'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-G100'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	000	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-G100'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-G100 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-G100'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. In big endian mode, the cursor pattern must be swapped according to Section 5.1.7 before being written to the frame buffer.

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 +2 * Tpixclk, where:
A = 93 (the memory controller's cursor request latency), Tmclk = MCLK cycle time (ns) and Tpixclk = PIXCLK 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-G100.

5.7.6 Video Out Functions

The MGA-G100 supports a special Matrox 12-bit MAFC connector for video-out functionality connected to a video encoder. The modes of operation can be programmed via the **XMISCCTRL** DAC register. See the description of the fields in this register for more information on the behaviour of each mode.

The MAFC connector takes 24-bits of data at the input of the DAC and multiplexes it on both edges of the clock to effectively transfer one 24-bit RGB pixel in one clock cycle. In this mode the interface outputs the same data that is going to the display (console). Alternatively, the video-out interface can pass-through YUV video-in data that is normally fed into the VD<7:0> bus for video-in purposes. In this mode the interface outputs to the video encoder 8 bit 4:2:2 YUV video, independent of the contents of the display (console).

The video-out interface consists of a 12-bit data bus VDOUT <11:0> and a VOBLANK/ signal that qualifies the data. Refer to a Section 6.6.2 for a description of the waveforms and data mapping for this connector.

5.7.7 Test Functions

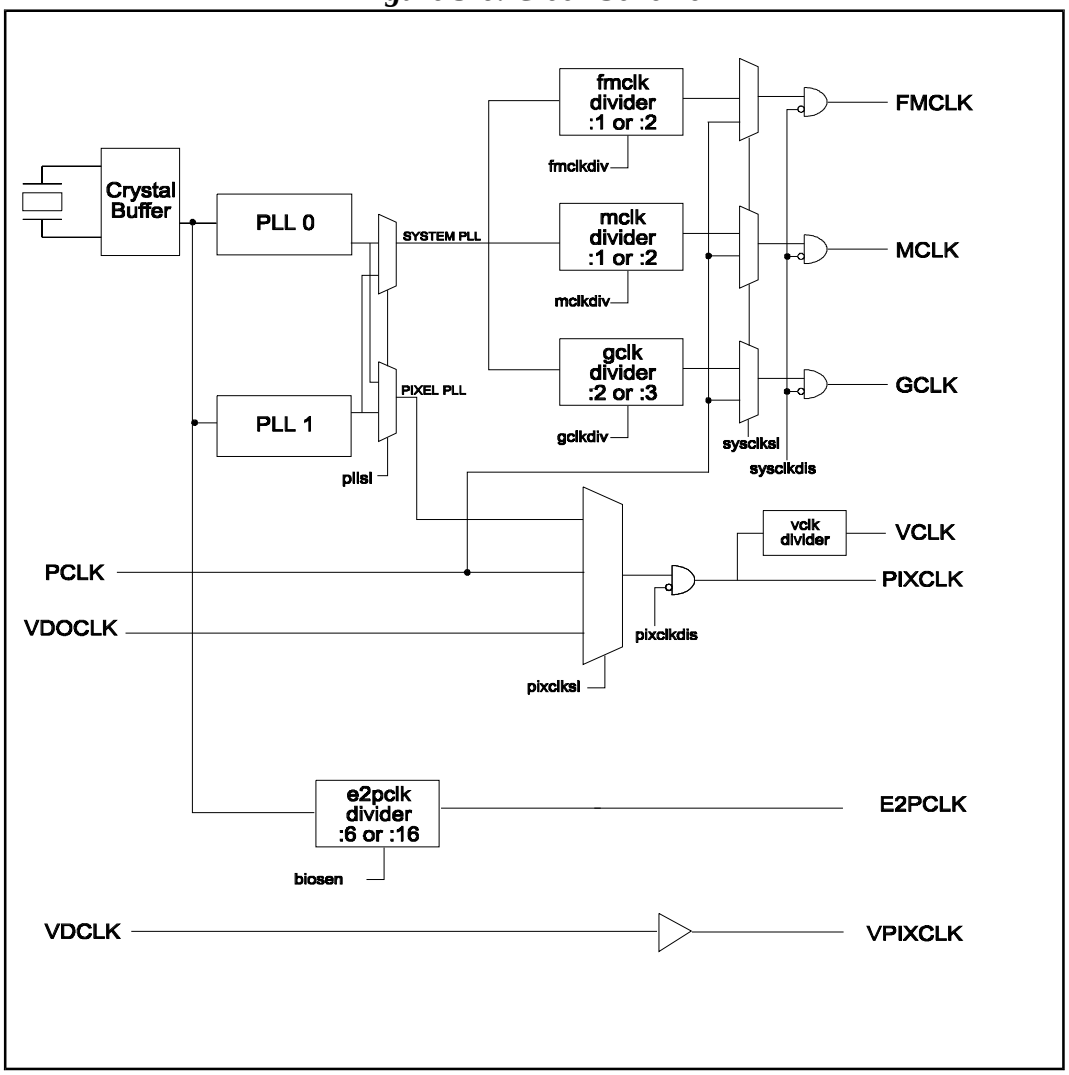
A 16-bit CRC is provided to verify video integrity at the input of the DAC. The CRC can be read via the **XCRCREMH** and **XCRCREML** registers when the vertical sync is active. The CRC is cleared at the end of the vertical retrace period, and calculated only when the video is active. The **crcsel** 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-G100's DAC has two independent programmable Phase Lock Loops (PLLs). One is selected as the system PLL, which is used for the system clocks: the memory clock (MCLK), fast memory clock (FMCLK) and the graphics engine clock (GCLK). The other one is selected as 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 = 27.000 MHz.

7 <= N <= 127 (feedback divider)

1 <= M <= 6 (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-G100 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 143 MHz FMCLK and 71.5 MHz GCLK when the **gclkdiv, fmclkdiv** and **mclkdiv** fields are set to '0'.

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 FMCLK, MCLK or GCLK (see Section 5.7.8.3 on page 5-82). 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 = 27.000 MHz.

7 <= N <= 127 (feedback divider)

1 <= M <= 6 (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 25.159 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 230 MHz (1600 x 1200 resolution at a 85 Hz refresh rate). The minimum period of the VCLK signal is 12.7 ns (1280 x 1024, 24-bit packed pixel at a 85 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 VDOCLK pin (when slaving the MGA-G100 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 VDOCLK 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-G100'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, FMCLK, or GCLK Division Factor

1. Set **sysclkdis** to ‘1’ (disable the system clocks).
2. Select the new clock source or change the **fmclkdiv** 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.

➽ Note: Steps (B) and (C) must be executed in an order which keeps **FMCLK** and **GCLK** within their specified values.

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	3.415	2.439
0.0 IRE	3.231	2.255

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 Video In Interface

5.8.1 Overview of the video grabber

MGA-G100’s field based video grabber captures the incoming video data in 4:2:2 format only, performs the programmed conversion and writes it into the framebuffer. There are two sets of registers that act as a double buffered set, one can be active during a field while the other is programmed. VBI data, either raw or decoded, can also be captured and written to the framebuffer. Active video data in 4:2:2 format, may be written directly into the framebuffer or is upsampled to YUV 4:4:4, color space converted, dithered to RGB 16 and written into the framebuffer. The Video Grabber works in a ‘one-shot’ mode. Software needs to program for every field that needs to be captured.

If both even and odd fields are desired, then the pitch of both windows (**vinpitch0** and **vinpitch1**) should be set to twice the anticipated line width, and the start address (**vinaddrX**) of the second window should be set to a value of 1 line width higher that the first windows’s start address.

5.8.2 MAFC Mode Selection

See the **XMISCCTRL** register to allow video in data to be driven back out the VDOUT(7:0) pins. The video in data is registered once with VDCLK, so there will be a one cycle delay between the input data

and the output data. The Video In interface does not have to be enabled in order for this pass-through mode to work, but it can be enabled if the stream is desired to be captured.

5.8.3 VBI data Capture

The Video In unit can be programmed to capture 2 types of VBI data: raw and sliced. If raw data is chosen, then all of the valid VBI data sent from the decoder will be captured and placed in the corresponding VBIADDRX location in the frame buffer. If necessary, the data will be padded with '0's in order to ensure that each line ends on a qword boundary. If sliced data capture is programmed, then the first 48 Y (luminance) samples will be captured, regardless of their value, and put in the correct part of the frame buffer.

Software must ensure that the decoder is programmed to send any VBI data to be captured before any active video lines are sent. If active video is sent first, followed by VBI data, and then active video again, data corruption will occur.

5.8.4 Programming sequence

Since the video grabber is a field based grabber the sequence is the same for all types of captures: odd only, even only, both odd and even, and VBI captures.The grabber registers are programmed between **vsync** for capture of the field following the next **vsync**.

Note that registers for window0 cannot be reprogrammed while window0 is active, and registers for window1 cannot be reprogrammed while window1 is active. It is legal to reprogram window1 registers when window0 is active, and reprogram window0 registers when window1 is active. If a particular window's registers are reprogrammed while that window is active, then data corruption will occur.

Before any programming of the Video In Interface is made:

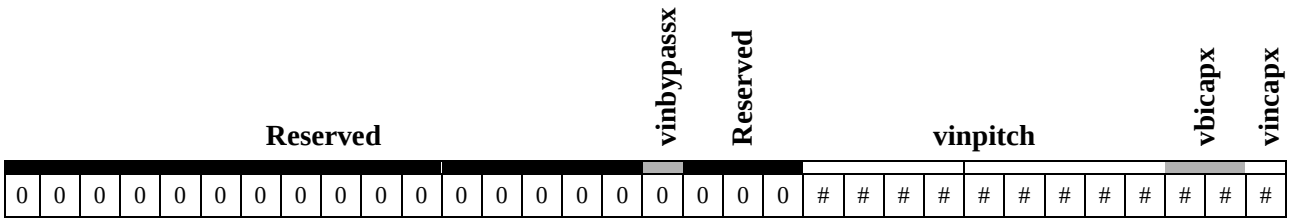
1. Initialize the decoder so it is providing a 27Mhz clock to the MGA-G100 on pin VDCLK, then

2. Enable the Video In interface using the **VINCTL** register. Once the Video In Interface is enabled, the **vinfielddetd** bit in the **VSTATUS** register is valid immediately following the next vsync. This is the case regardless of the state of the **vinvsyncien** field in the **VIEN** register.

Programming steps:

1. Clear the vsync flag in the **VSTATUS** register to clear the previous vsync status.
2. Enable the video input vsync interrupt (**vinvsyncien**).
3. At the next vsync interrupt read the **VSTATUS** register. If the **vinvsyncpen** bit is active clear the flag like in step 1. If the completed **vinfieldetd** bit indicates the field desired to capture go to step 4. Other wise repeat this step.
4. Program all the Video In window registers 0 or 1 related to video capture.

VINCTLX



Register	*Function*	*Comments/Alternate Function*
VBIADDRX	VBI Write Address	if vbicapx is not ‘00b’
VINADDRX	Video Write Address	if vincapx is ‘1’

5. Program the **vinnextwin** bit in the **VINNEXTWIN** register to select the same window that was chosen in Step 4.
6. If another field is desired following the field just programmed go to step 3, otherwise disable interrupts.

5.9 Interface with a CODEC

A CODEC can be used in conjunction with the MGA-G100 chip to compress and decompress a video stream in real time.

➽ Note: When programming **CODECHOSTPTR** for compression/decompression, the Codec Interface will not stop transferring data when the PTR value is reached. Software should suspend the Codec Interface’s memory accesses until more data is put into memory (or more space is available) by setting **codectransen** of **CODECCTL** to a ‘0’

5.9.1 Memory Organization

Three main sections of memory are reserved for Codec Interface usage. The **CODECADDR** register is used to set the location of the buffer in the off-screen memory area. Figure 5-7 shows the organization of the CODEC interface

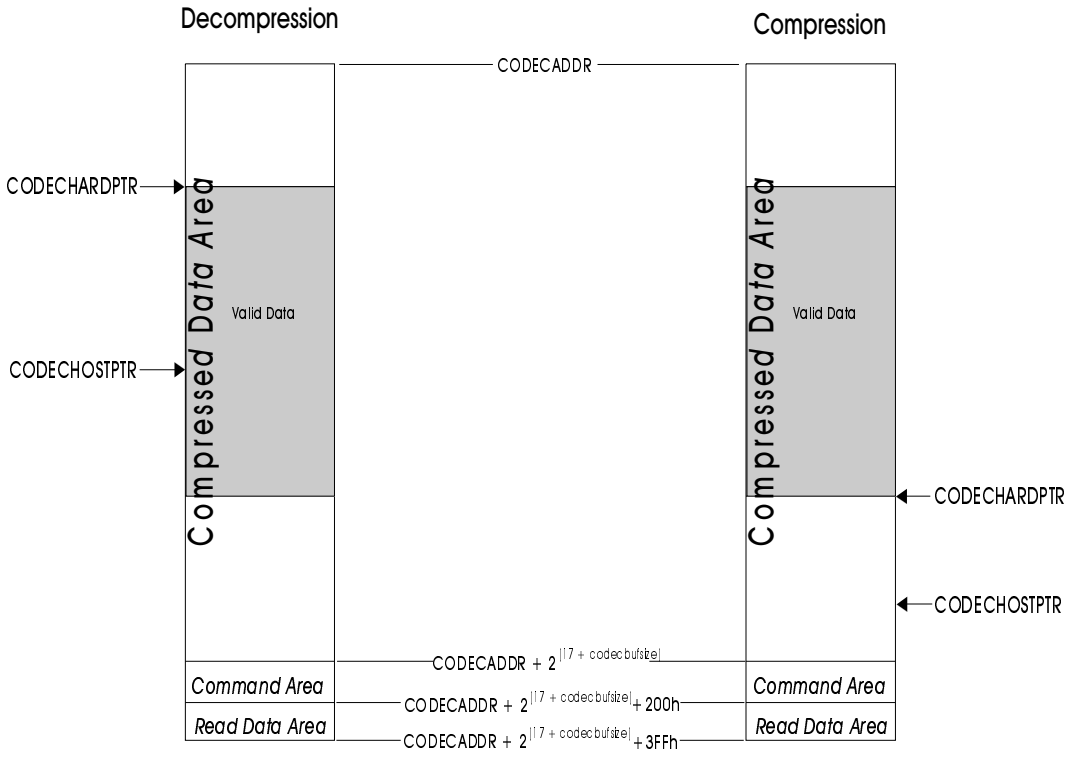
Compressed data area

The compressed data area is used both for compression and decompression operations. This buffer size is selectable between 128Kbytes or 256Kbytes. The functions of **CODECHARDPTR** differ for compression and decompression (refer to the register definitions in Chapter 4 for more details).

The compressed data area acts as a circular queue. Data from the beginning of one field is loaded into the dword which follows the last data from the previous field. It is the sole responsibility of the software to ensure that the compressed data area never overflows. The Codec Interface engine does not verify that there is valid compressed data in the buffer before reading, nor does it check to see that there is enough free space in the buffer before writing. The buffer level interrupt has been provided to help the software to ensure that overflows never occur.

.

Figure 5-7: CODEC Interface Organization



Command area

The command area is used to store the commands to be sent to the CODEC. The organization of these commands and their execution is explained in more detail in the 5.9.2 section. The command area is set to a fixed size of 512 bytes.

Read data area

The read data area is used to store the data which has been read from the CODEC. When more than one location is read with a single command, the data is packed to take full advantage of the 8-byte wide memory locations. However, if only a single CODEC location is read, the remaining 7 bytes of the qword will be unused.The read data is always stored beginning with the LSB. The read data area is set to a fixed size of 512 bytes.

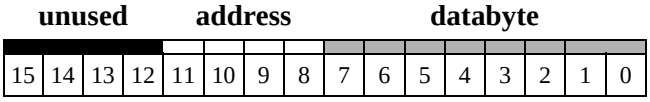
5.9.2 Command Execution

Register read and write commands are stored in an off-screen command buffer. Each qword in the buffer may contain either a command or, in the case of a write command, write data. Each command, with its accompanying data, is stored one after the other in the command buffer. Note that the first qword in the queue must always contain a command. The format of a command, with its accompanying data, is shown in Figure 5-8.

Figure 5-8: CODEC Command Format

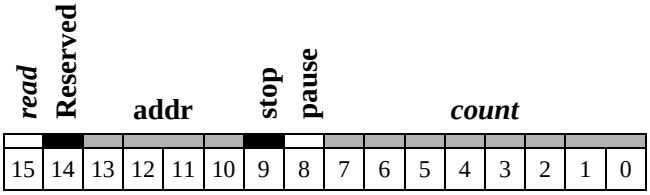
63			16 15	0
			Command	
write data 4	write data3	write data2	Write data1	
:	:	:	:	
:	:	:	:	
write data n	write data n-1	write data n-2	write data n-3	

Write data:



The command word itself is composed of several control bits which affect command execution:

Command Word Definition:



count*<7:0>** The ***count specifies the number of registers to be written to or read from the CODEC. When executing writes, the count will refer to how many words in Write Data qword(s) will be processed. When executing reads, the count will refer to how many times the address **addr** in the Command Word will be read. A value of "00"h is interpreted as 256 (decimal).

pause*<8>** The intended use for this bit is for compression. The ***pause bit is used to suspend register accesses until next end-of-field indication. During the vertical blanking interval, software will load the off-screen command buffer with register writes to the indirect registers of the CODEC to be used for the next field. The last register write will set the ***pause*** bit. Software will then write additional commands to the command buffer following the register writes, which will cause the CODEC Interface engine to read the field statistics and place the information in the read buffer.

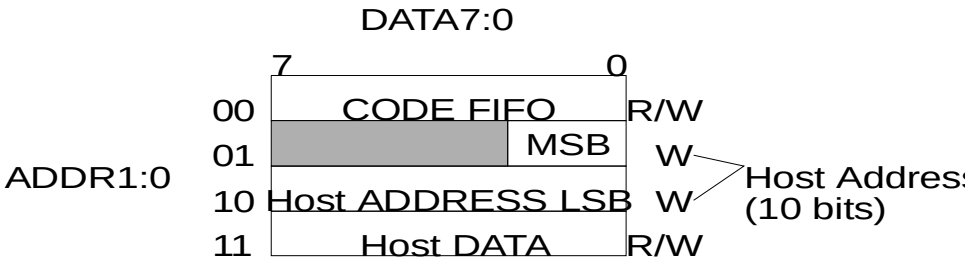
The CODEC Interface engine will execute all of the register write commands until it reaches the ***pause*** bit. At this point, the CODEC begins compression transfers. At the next end-of-field indication, register access execution automatically resumes, and hardware reads the field statistics from the CODEC and writes the information to the off-screen buffer. At the completion of these read operations, the stop bit is encountered, and the software is interrupted. Software then reads the information provided by the DMA engine in off-screen memory, and the process repeats.

stop*<9>** The ***stop bit is used to halt register accesses. When active (set to '1'), the command word in which it is contained is executed. At this point, register accesses are complete, and the **cmdcmplpen** field of the **VSTATUS** register is set. When software triggers command execution once again, execution begins from the first qword in the command buffer (the command buffer is not a circular queue).
Note: Setting both the pause bit and the stop bit is illegal and will cause data corruption.

***addr* <13:10>** Address to access for register reads

When executing read commands, **addr<13:10>** will indicate which address the Codec Interface will read from, in the Codec Interface's address space. This address will be read as many times as was programmed in **count<7:0>**. When executing register writes, these 4 bits are unused. When transferring compressed data, the address asserted is programmed by software in the **CODECCTL** register. When in VMI mode, <13:10> are output. When in I33 mode, <12:10> are output.

Address space of I33 CODEC (In CODE SLAVE MODE)



***read*<15>** This bit indicates the direction of transfer for reads or writes, with respect to the CODEC. Read and write commands may be interleaved in any manner.

- 0 : write data to the CODEC registers
- 1 : read data from the CODEC registers

1. The Codec Interface engine begins executing commands when the **CODECCTL** register is written with an access that sets the **cmdexectrig** field (the command execution trigger field does not need to be cleared by software). When software triggers command execution, the Codec Interface engine resets its Command Area pointer and Read Data Area pointer.
   - The first pointer selects the next qword to be read. This pointer is reset to point to the beginning of the command area when the **cmdexectrig** field is set.
   - The second pointer selects the next qword to be written in the read data area. This pointer is reset to the beginning of the read data area when the **cmdexectrig** field is set.
2. The Codec Interface engine then fetches the first command.
   - If the command is a write, the appropriate number of qwords are read from the command area and written to the CODEC. After one qword of data is read from the command area, the data is written to the CODEC one word at a time, starting with word 0, then word 1, and so on to word 3. If the write command completes before all 4 words are written, the remaining data is dropped and not used. On the next write command, data is again fetched and sent to the CODEC, starting with word 0.
   - If the command is a read, the appropriate number of bytes are read and stored in the read data area. The data is read from the CODEC one byte at a time, and is accumulated until a complete qword has been received. The first byte read is loaded into byte 0, the second into byte 1, and so on to byte 7 (the eighth byte). The resulting qword is then written to the next available location of the read data area. If the read command completes before all 8 bytes are filled, the data is written to the off-screen buffer as is (unfilled). On the next read command, data is again filled, starting with byte 0.
3. Upon completing the read or write command, the Codec Interface engine fetches the next command from the command buffer, and the process repeats until a STOP command is executed.

Examples

Table 5-9 shows the contents of a command area, while Table 5-10 shows the contents of the read data

area after the execution of all commands has taken place.

Table 5-9: Contents of the Command Area

Location	Contents	Meaning
+ 8000h	XXXXXXXXXXXX8401h	Read address “0001”, count=1
+ 8008h	XXXXXXXXXXXX0002h	Write 2 locations
+ 8010h	XXXXXXXX03FF02AAh	Write addresses and data
+ 8018h	XXXXXXXXXXXX0003h	Write 3 locations
+ 8020h	XXXX06BB057504EEh	Write addresses and data
+ 8028h	XXXXXXXXXXXX0104h	Write 4 locations, pause
+ 8030h	07DD06CC05BB04AAh	Write addresses and data
+ 8038h	XXXXXXXXXXXX9E04h	Read address “0111”, count=4, stop

Table 5-10: Contents of the Read Data Area

Location	Contents	Meaning
+ 8200h	XXXXXXXXXXXXXX00h	Read data from address “0001”
+ 8208h	XXXXXXXXDDDDDDDDh	Read data from address “0111”

The first command read 1 byte, 00h, from address 01h. The data was written to memory at address +8200h. The next command was a write of 2 locations. The data is fetched in memory and written as data AAh to address 02h, and data FFh to address 03h. The next command is a write to 3 locations. The data is fetched from memory and written as data EEh to address 04h, data 75h to address 05h, and data BBh to address 06h. The next command is a write to 4 locations with a pause. The data is fetched and written as data AAh to address 04h, data BBh to address 05h, data CCh to address 06h, and data DDh to address 07h.

Since the pause bit was asserted in this command, the Codec Interface engine will not fetch its next command until it receives the end-of-field indication from the CODEC. At this point, compressed data transfers begin. When the end-of-field is detected, the interface engine proceeds with the next command, which is a read from address 07h, count of 4, with stop. The data is read as DDh, DDh, DDh, DDh, and put into location +8208h. Since the stop bit was asserted in this command, the interface engine will not send any further commands to the CODEC and will assert the **cmdcmplpen** field of **VSTATUS**.

5.9.3 Output mode

The Codec Interface may operate in 3 possible output modes: VMI Mode A, VMI Mode B, and Zoran I33 compatible mode. The mode is programmed in the **CODECCTL** register. All examples set forth assume I33 mode. All programming procedures are identical in all modes (except for setting the proper mode in the **CODECCTL** register. When programming the Codec Interface in VMI mode, the DDC(2) pin must be programmed to be an input. This DDC(2) pin may not be used for any other functions when the Codec Interface is enabled and in VMI mode.

5.9.4 Codec Interface IDLE State

In order to have the Codec Interface enabled but in and IDLE state, the following fields need to be set as indicated after the Codec Interface is disabled:

codecen = ‘1’

cmdexectrig = ‘0’

codectransen = ‘0’

all other fields = ‘X’

5.9.5 Recovery Width Programming

The strobe recovery pulse width in the Codec Interface engine is programmable in the **CODECCTL** register. The **codecrwidth** should be programmed before the Codec Interface is enabled (with **codecen**) to begin transfers to the CODEC. If the **codecrwidth** is reprogrammed during data transfers, data corruption may occur. The formula to compute the optimal recovery time, Trec (in ns), is:

Trec = Tcodecack + 2*Tmclkbuf, and Trec > Tcodecrec

where Trec = Twister’s minimum recovery time

Tcodecack = max time it takes for the CODEC to deassert its acknowledge

Tmclkbuf = the period of Twister’s mclk

Tcodecrec = the CODEC’s minimum required recovery width

For example:

Given mclkbuf (internal memory clock) = 100Mhz, thus Tmclkbuf = 10 ns

Tcodecack = 25 ns

Tcodecrec = 55.5 ns

Then: Trec = 45 ns, which is less that Tcodecrec = 55.5 ns.

Thus, Trec should equal 60 ns, corresponding to 6 mclkbuf cycles (the next highest integer number of cycles).

5.9.6 CODECTRANSEN Bit Programming

Startup of Compressed Data Transfers

When the Codec Interface is initially programmed to begin compression or decompression, the **codectransen** is set to ‘1’. If **codectransen** is set to ‘0’ then the compressed data transfer will not begin. They will only start when **codectransen** is set to ‘1’.

During Decompression

When performing decompression transfers, the **codectransen** bit causes the Codec Interface to stall after it has sent all 4 qwords in its fifo to the CODEC. If software triggers command execution before the Codec Interface has a chance to write all 4 qword to the CODEC, then any data present in the Codec Interface’s fifo at command execution will be trashed. Thus, to ensure that no data is lost, software should poll the **codecstalled** bit in the **VSTATUS** register after it sets **codectransen** to a ‘0’. When **codecstalled** equals ‘1’, the Codec Interface has finished transferring all 32 bytes in its fifo to the CODEC and has stalled, indicating that command execution can safely be triggered.

Software can continue decompression transfers after completion of command(s) (signified by a stop in the last command and **cmdcmplpen** interrupt), by setting the **codectransen** bit to a ‘1’ after command execution has been triggered. Using this method, the Codec Interface will wait until it has finished the last command, and then restart the decompression transfers without any corruption of the compressed data.

During Compression

When performing compression transfers, the **codectransen** bit causes the Codec Interface to stall after it has written all 4 qwords in its fifo to memory. If **codectransen** is set to ‘0’, and then software triggers command execution before the Codec Interface has chance to flush its fifo to memory, then any data present in this fifo at command execution will be trashed.

In order to preserve the integrity of data when requiring command execution during compression, software should poll the **codecstalled** bit in the **VSTATUS** register after it sets **codectransen** to '0'. When **codecstalled** equals '1', this indicates that the Codec Interface has finished reading 32 bytes of compressed data from the CODEC and it has written them to the frame buffer. Software can then safely issue command execution. If compression is desired to be continued after command execution, then software should set **codectransen** back to a '1' after it has issued the command execution trigger. This way, once the last command has been completed, the Codec Interface will continue with compression transfers.

5.9.7 STOPCODEC Field programming

The **stopcodec** bit enables the Codec Interface to stop transferring compressed data upon detection of the end of the field. It is used for both compression and decompression transfers in I33 mode only. The **stopcodec** must be set to '0' when in VMI mode.

Use During Compression

When software sets **stopcodec** to a '1' upon starting compression transfers, the Codec Interface polls for an end of image (LOCDE) signal from the CODEC. This LCODE informs the Codec Interface that its current read is the last byte of field. The Codec Interface will write any data its fifo to the compressed data area of the frame buffer and resume command execution.

If software wishes to continue compressing the next field, it should:

1. wait for **cmdcmplpen** interrupt to signal the completion of the commands for the previous field.
2. write new commands for the next field in the command area of the frame buffer.
3. trigger command execution and set **stopcodec** to '0' to continue compression transfers (compression will resume when the Codec Interface encounters a stop or pause in the current command).
4. poll the **CODECHARDPTR** register to see when the Codec Interface has begun writing more compressed data into the frame buffer.
5. set **stopcodec** back to '1' if software desires to stop compression at the end of the current field.

Use During Decompression

When **stopcodec** is set to a '1' in the **CODECCTL** register and decompression is enabled, the Codec Interface will look for FFD9 end of image marker in the compressed data stream. When the end of image is detected, the Codec Interface will stall and post the **dcmpeoipen** interrupt (if enabled).

At this point, software has 3 options:

1. Reset the Codec Interface engine
2. Set the **stopcodec** bit to '0'. If just **stopcodec** is changed in the **CODECCTL** register, then decompression transfers will resume with the next byte in the Codec Interface's fifo.
3. Execute commands. In order to do this without data corruption, software should first set **stopcodec** to '0' in order to resume decompression transfers, and then set **codectransen** to '0' to stall data transfers when the internal 32 byte fifo is empty. When software detects that the **codecstalled** field is asserted in the **VSTATUS** register, it can then safely trigger command execution.

If **stopcodec** is set to '0' for decompression transfers, then the Codec Interface will not detect the FFD9h

end of image marker in the stream. It is up to software to stop the Codec Interface engine when it has determined that the desired field is complete (by polling the buffer level interrupt).

5.9.8 Miscellaneous Control Programming

The **miscctl** byte located in the **CODECCTL** register is used to program an 8 bit flip-flop on the graphics card. The values of the **miscctl** field are used to set various inputs to the CODEC and MPEG2 chips used with Twister (SLEEP, START, etc.). By writing to the **miscctl** field, software triggers a sequence to program the on-board flip-flop with the corresponding data.

Note that in order for this automated sequence to be executed, the Codec Interface engine must be enabled and in one of the following modes/states: IDLE, COMPRESSION, or DECOMPRESSION. For example, if the chip select for a CODEC is connected to bit(0) of the on-board flip-flop, and is active low, then software could enable the CODEC as follows:

1. write “00000000”b to the lower byte of the **CODECCTL** register (to reset it, optional)
2. write “00000001”b to the lower byte of the **CODECCTL** register (to enable it)
3. write “11111110”b to the **miscctl** field of the **CODECCTL** register (to set the chip select to the CODEC).

If the Codec Interface is in the process of compression or decompression when the **miscctl** field is written to, then the Interface will wait until the current byte transfer is complete. At that point the on-board flip-flop will be programmed, and then compressed data transfers will resume with the next byte. No compressed data will be lost or corrupted during this process.

5.9.9 DVD Decoding with a CODEC with HRDY signal

When using a CODEC with programmable HRDY signal, that signal is connected to the MGA-G100 DDC(2) input pin. It is recommended that the CODEC be programmed to deassert HRDY when only 32 byte locations are available in its fifo. (note that this value is only a starting point and may not be the optimum value.) When software programs the Codec Interface in VMI mode and sends a stream to be decoded, the Codec Interface will poll the state of HRDY every 32 bytes. Therefore, the Codec Interface will only request a 32 byte packet from the frame buffer when there is space for it in the CODEC. If software needs to access the CODEC’s registers, it should stall compressed data transfers with the **codectransen** bit (see the section entitled CODECTRANSEN Bit Programming) and wait for **codecstalled** field in **VSTATUS** to be ‘1’.

5.9.10 Initialization Sequence

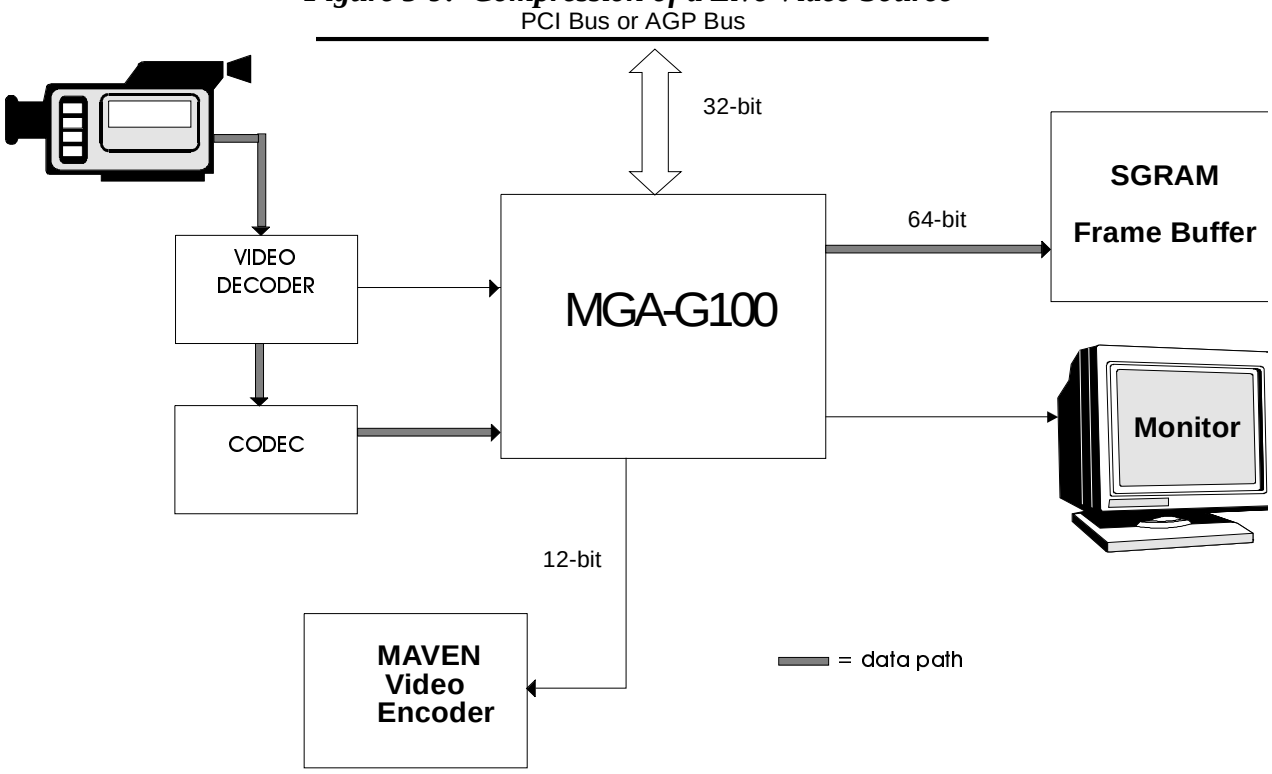
Before the Codec Interface is enabled to begin any types of transfers, the following steps must be taken:

1. Ensure that standard VGA connector mode is not selected. This is done by having **mfcsel** in the XMISCCTRL register not equal to 00b.
2. Ensure that the Serial Eprom controller is done with all transfers.

5.9.11 Compressing data

Data being compressed comes from the video decoder. The compressed video frame is returned to the frame buffer through the Codec Interface channel.

Figure 5-9: Compression of a Live Video Source



Two interrupts are used while the CODEC is compressing data. The buffer level interrupt is used to tell software the current fill state of the frame buffer. The command execution completed interrupt is used to request host services in order to adjust the compression factors.

The following steps must be performed in order to compress video:

Step 1. Program the video decoder according to its specification.

Step 2. Software must reset the Codec Interface engine:

Register	*Function*	*Comment / Alternate Function*
CODECCTL	Reset the Codec Interface engine	Write 00h

Step 3. Software must then initialize the following registers in the Codec Interface engine:

Register	*Function*	*Comment / Alternate Function*
CODECADDR	Address of off-screen buffer	

Step 4. The CODEC must be initialized with the mode and other parameters shown below. To do this, software must transfer CODEC register write commands into off-screen memory. Typically, the registers to be written are:

- *Mode Control*
- *FIFO Control*
- *HSTART*
- *HEND*
- *VSTART*
- *VEND*
- *Compression Ratio* (set the ***stop*** bit; see 'Command Word Definition:' on page 5-87)

➾Note: The CODEC specification will have detailed information regarding which registers

need to be programmed.

Step 5. Trigger the writing of commands to the CODEC:

Register	*Function*	*Comment / Alternate Function*
CODECCTL	Enable the Codec Interface engine	00001111b

Step 6. At the completion of this command, the Codec Interface engine will set the **cmpcmplpen** field of
VSTATUS. The host will read the status in order to know when the command has been executed. The host must also clear the **cmpcmplpen** flag.

Register	*Function*	*Comment / Alternate Function*
VICLEAR	Clear interrupts	02h

Step 7. The host must then transfer the following commands to off-screen memory:

- program registers in CODEC (set the ***pause*** bit)
- read field information from CODEC
- read the FIFO status for the error conditions (if necessary, see CODEC specification) (set the ***stop*** bit)

Step 8. The host must then enable all interrupt bits.

Register	*Function*	*Comment / Alternate Function*
CODECHOSTPTR	Next level interrupt	value desired by the software

Step 9. Trigger the write of the commands to the CODEC and enable the transfer of compressed data:

Register	*Function*	*Comment / Alternate Function*
CODECCTL	Reset the Codec Interface engine	01001111b

Step 10. The Codec Interface engine will then begin to transfer data from the CODEC to the compressed data area. Note that since the first field will probably be corrupted, software should discard it.

Step 11. The Codec Interface engine will interrupt the host every time **CODECHARDPTR** is equal to **CODECHOSTPTR**. The host can detect this situation by reading the **blvlpen** field of the **VSTATUS** register. As part of the interrupt routine, the host must perform transfers from the compressed data area to the system memory or hard disk. The host must also clear the **blvlpen** flag and update its pointer.

Register	*Function*	*Comment / Alternate Function*
CODECHOSTPTR	Next level interrupt	
VICLEAR	Clear Codec Status Register	04h

Step 12. When the CODEC asserts LCODE, this informs the Codec Interface engine that its current read is the last byte of the field. The Codec Interface engine will write all remaining data in its memory interface buffer to the compressed data area and resume command execution.

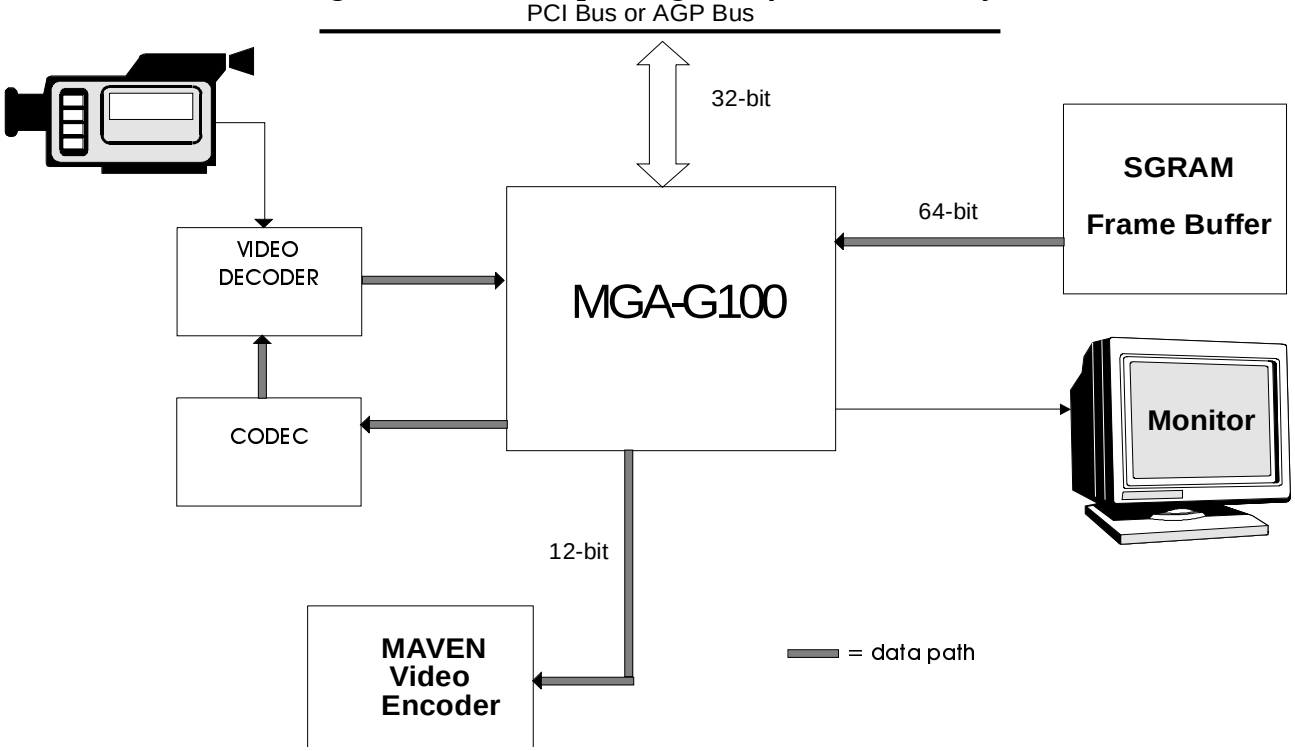
Step 13. The Codec Interface engine will then interrupt software when the command is completed. The host can detect this situation by reading the **cmpcmplpen** field of the **VSTATUS** register. Based on the statistics, software must calculate new specs for compression field, and update the write commands in the off-screen buffer (see Step 6). The host must also clear the **cmpcmplpen** flag and restart command execution.

Register	*Function*	*Comment / Alternate Function*
VSTATUS	Status of memory commands	00000010b
VICLEAR	Clear all pending CODEC interrupts	06h
CODECCTL	Reset the Codec Interface engine	00000000b

5.9.12 Decompressing data

When data is being decompressed, the compressed information is sent to the CODEC through the Codec Interface port. From there, the data is sent to the decoder and back to the MGA-G100 to be displayed on the monitor or TV (through MAVEN):

Figure 5-10: Decompressing Data from the Memory



Two interrupts are used when the Codec Interface is decompressing data. One is the buffer level interrupt. It is used to request more data from the host. The other is the decompression end of image interrupt. It is used to notify software when the end of the field has been detected.

The following steps must be performed in order to decompress video:

Step 1. Program the video decoder according to its specification and the ‘Video In Interface’ on page 5-83.

Step 2. Software must reset the Codec Interface engine:

Register	*Function*	*Comment / Alternate Function*
CODECCTL	Reset the Codec Interface engine	Write 00h

Step 3. Software must then initialize the following registers in the Codec Interface engine:

Register	*Function*	*Comment / Alternate Function*
CODECADDR	Start address of buffer in off-screen memory	

Step 4. The CODEC must be initialized with the mode and other parameters shown below. To do this, software must transfer CODEC register write commands into off-screen memory. Typically, the registers to be written are:

- *Mode Control*
- *FIFO Control*
- *HSTART*
- *HEND*
- *VSTART*
- *VEND* (set the ***stop*** bit; see 'Command Word Definition:' on page 5-87)

Note: The CODEC specification will have detailed information regarding which registers need to be programmed.

Step 5. Trigger the writing of commands to the CODEC:

Register	*Function*	*Comment / Alternate Function*
CODECCTL	Enable the Codec Interface engine	00000111b

Step 6. The host must then fill at least half of the compressed data area and write its pointer.

Register	*Function*	*Comment / Alternate Function*
CODECHOSTPTR	Next level interrupt	

Step 7. At the completion of the command, the Codec Interface engine will set the **cmpcmplpen** field. The host will read the status in order to know when the command has been executed. The host must also clear the **cmpcmplpen** flag.

Register	*Function*	*Comment / Alternate Function*
VICLEAR	Clear all interrupts	02h

Step 8. The host must then enable the buffer level interrupt and enable the transfer of compressed data.

Register	*Function*	*Comment / Alternate Function*
CODECCTL	Enable the Codec Interface engine	01000011b

Step 9. The Codec Interface engine will then begin to transfer data from the compressed data area to the CODEC.

Step 10. The Codec Interface engine will interrupt the host every time **CODECHARDPTR** is equal to **CODECHOSTPTR**. At that time, the host should add more data to the compressed data area. The host must also clear the **blvlpen** flag and update its pointer.

Register	*Function*	*Comment / Alternate Function*
CODECHOSTPTR	Next level interrupt	
VICLEAR	Status of memory commands	04h

5.9.13 Error Recovery

The Codec Interface gives highest priority to command execution. Thus, when transferring compressed data, if software determines there is an error condition with the CODEC, software can trigger command execution and pre-empt compressed data transfers. If the Codec Interface is triggered to execute

commands while it is the process of transferring compressed data, the Codec Interface engine will disregard any data presently in its 4 qword fifo and load and execute the first command in the command data buffer.

When the interface engine has completed all the commands (signalled by a STOP in the last command), it will resume compressed data transfers wherever it left off (the **CODECHARDPTR** does not reset under these conditions). Note that any data that was loaded into the 4 qword fifo (either from the CODEC or from the frame buffer) when command execution was triggered was lost, unless the Codec Interface was first stalled with the **codectransen** bit and polling of the **codecstalled** bit was done.

5.10 EEPROM Programming

5.10.1 Requirements

Writing the EEPROM has programming requirements that must be followed and are listed below:

1. All page writes must be page boundary aligned.
2. During a page write, the ROMFIFO must not be allowed to become empty during the page.
3. Software must control the size of the page writes.

Reading the EEPROM has no requirements.

5.10.2 Writing to the EEPROM

Writing to the EEPROM is performed with the following steps.

1. Read the page_size from the EEPROM
2. Set the **eepromwt** field in the **OPTION2** register.
3. Write **page_size** number of bytes to the EEPROM.
4. Read any valid address from the EEPROM.
5. Goto step 2 until all the desired pages are written.

Reading the EEPROM has no special sequence.

5.11 Interrupt Programming

The MGA-G100 has 8 interrupt sources: 5 graphics interrupts and 4 video interrupts.

The graphics engine interrupts are:

1. Soft trap interrupt

   This interrupt is generated when a write to the **SOFTRAP** register is executed (refer to ‘Programming Bus Mastering for DMA Transfers’ on page 5-14 and to the **SOFTRAP** register description).

2. 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.

3. 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.

4. 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).

5. 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 functions.

The video interrupts are:

1. Video In Vertical sync interrupt

   This interrupt is generated when a video input vsync is detected. This interrupt is located in the **VSTATUS** register.

2. Codec command complete interrupt

   This interrupt is generated when the codec interface has completed executing the commands in the command buffer. The interrupt is located in the **VSTATUS** register

3. Codec buffer level interrupt

   This interrupt is generated when the Codec interface's hardware pointer (**CODECHARDPTR**) is equal to the value set in the **CODECHOSTPTR**. This interrupt is located in the **VSTATUS** register.

4. Decompression end of image interrupt

   This interrupt is posted when **stopcodec** is a '1'. The Codec Interface is decompressing data, and the FFD9h end of image marker was detected in the data stream. This interrupt is located in the **VSTATUS** register.

The following table summarizes the supported functionality that is associated with each interrupt source.

Interrupt	***STATUS***	***EVENT***	***ENABLE***	***CLEAR***
Soft trap	-	**softrapen**	**softrapen**	**softrapiclr**
	-	**STATUS**<0>	**IEN**<0>	**ICLEAR**<0>
Pick	-	**pickpen**	**pickien**	**pickiclr**
	-	**STATUS**<2>	**IEN**<2>	**ICLEAR**<2>
Vertical sync	**vsyncsts**	**vsyncpen**	**vinten**	**vintclr**
	STATUS<3>	**STATUS**<4>	**CRTC11**<5>	**CRTC11**<4>
Vertical line	-	**vlinepen**	**vlineien**	**vlineiclr**
	-	**STATUS**<5>	**IEN**<5>	**ICLEAR**<5>

Interrupt	*STATUS*	*EVENT*	*ENABLE*	*CLEAR*
External	**extpen** **STATUS<6>**	-	**extien** **IEN**<6>	-
Video In vsync	- -	**vinvsyncpen** **VSTATUS**<0>	**vinvsyncien** **VIEN**<0>	**vinvsynciclr** **VICLEAR**<0>
Codec command done	- -	**cmdcmplpen** **VSTATUS**<1>	**cmdcmplien** **VIEN**<1>	**cmdcmpliclr** **VICLEAR**<1>
Codec buffer level	- -	**blvlpen** **VSTATUS**<2>	**blvlien** **VIEN**<2>	**blvliclr** **VICLEAR**<2>
Codec decompression end of image marker	- -	**dcmpeoipen** **VSTATUS**<3>	**dcmpeoiien** **VIEN**<3>	**dcmpeoiiclr** **VICLEAR**<3>

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 except for **vsyncpen** EVENT.

VSTATUS Indicates which bit reports the current state of the video interrupt source.

VICLEAR A pending bit is kept set until is cleared by the associated clear bit.

VIEN Each interrupt source may or may not take part in activating the PINTA/hardware interrupt line. The **VSTATUS** flags are only set when the enable bit in **VIEN** for each respective source is on.

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
- **vinvsyncpen** is set after a vsync is detected and the video in unit has completed writing the data to memory.

5.12 Power Saving Features

The MGA-G100 supports the following 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 video section can be powered down using the PCI Bus Power Management state transitions D0 to D3 or using the following steps:
  1. Set bits **scroff**, **hsyncoff** and **vsyncoff** to '1'.
  2. Disable the cursor (set the **curmode** field to '00').
  3. Set the **pixclkdis** field of **XPIXCLKCTRL** to '1'.
  4. Power down the DAC.
  5. Power down the LUT.
  6. Power down the Pixel PLL.
- The power consumption of the chip can be further reduced by shutting-down the drawing engine and slowing-down the system clocks.The procedure below must be followed:
  1. Power down the video section following the procedure above.
  2. Wait for **dwgengsts** to become '0'.
  3. If the contents of the frame buffer must be preserved, FMCLK must be running and the **rfhcnt** field of the OPTION register must be re-programmed according to the new FMCLK frequency (normally, set rfhcnt to '000001').
  4. Program the MCLK to the desired value following the procedure in section 5.7.8.3 (see (B) Changing the System PLL Frequency on page 5-82). The recommended PLL oscillation frequency is 6.65MHz (N=62, M=31, P=7,S=0).
  5. Set fmclkdiv to '1' (gclkdiv should already be '0' and must be set to '0' if that is not already the case) following the procedure in section 5.7.8.3 (see (C) Changing the System Clock Source, FMCLK, or GCLK Division Factor on page 5-85)

Note: In Power Saving mode, it is not permitted to use, or even initialize, the drawing engine.

Note: MGA-G100 supports PCI Bus Power Management Interface spec 1.0.

5.13 Coming Out of Power Saving Mode

- Set fmclkdiv to '0' following the procedure in section **5.7.8.3 (C) "Changing the System Clock Source, FMCLK, or GCLK Division Factor"**.
- Program the System PLL to normal frequency following the procedure in section **5.7.8.3 (B) "Changing the System PLL Frequency"**.
- Program rfhcnt to its normal value.
- Power up the Pixel PLL.
- Power up the LUT.
- Power up the DAC.
- Set the pixclkdis field of XPIXCLKCTRL to '0', or reprogram the Pixel PLL to a new operating frequency if desired by following the procedure in section **5.7.8.3 (A) "Changing the Pixel Clock Frequency or Source".**

- Reset bits SCROFF, VSYNCOFF and HSYNCOFF to ‘0’.

Chapter 6: Hardware Designer's Notes

This chapter includes:

6.1 Introduction

The MGA-G100 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/AGP bus
- All necessary support for external devices such as ROM
- Direct connection to the RAM
- Direct interface to the video and feature connectors

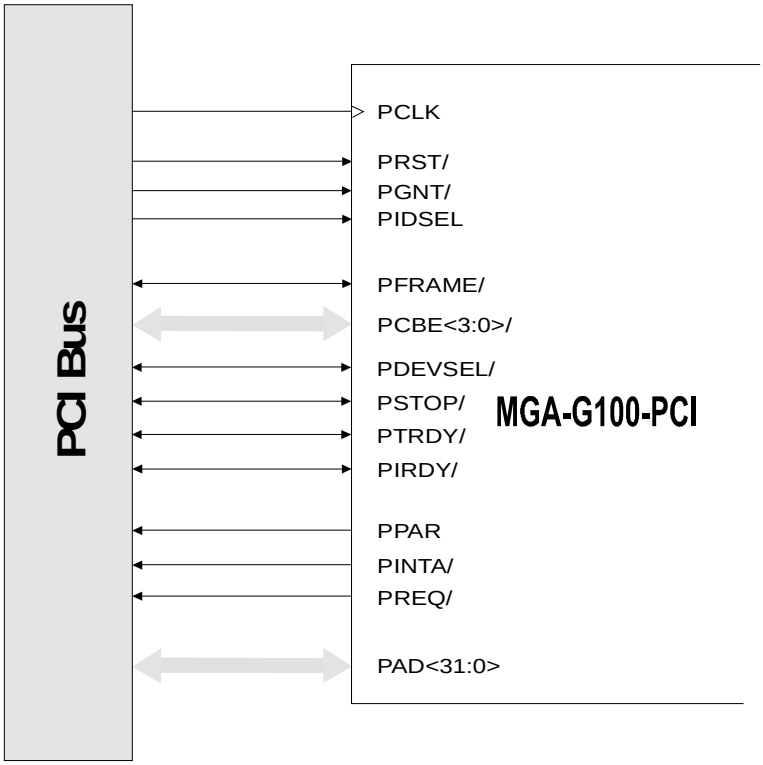
6.2 HOST Interface

6.2.1 PCI Interface

The MGA-G100-PCI interfaces directly with PCI as shown in Figure 6-1. The MGA-G100-PCI 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. The MGA-G100-PCI can also act as a master on the PCI bus - refer to Section 5.1.9 for more information.

Figure 6-1: PCI Interface

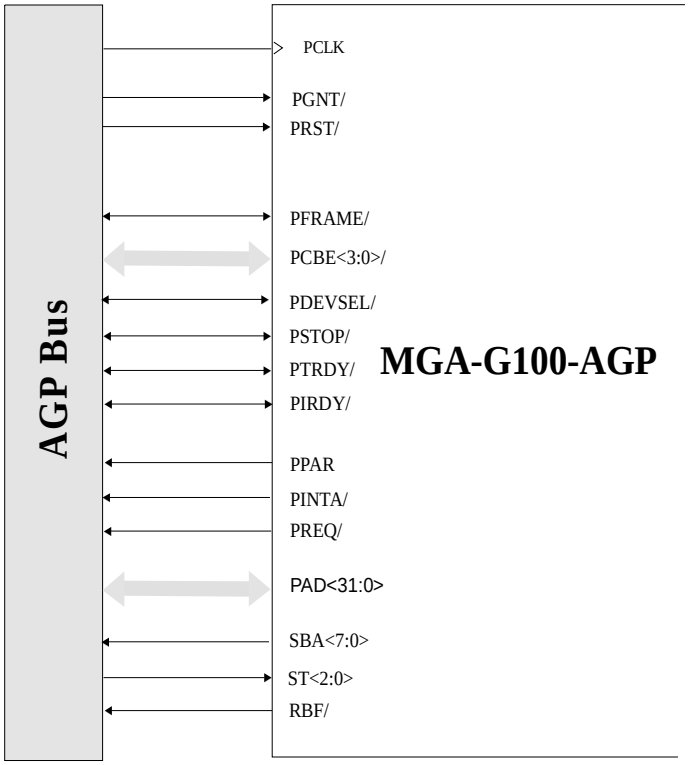


6.2.2 AGP Interface

The MGA-G100 -AGP interfaces with the AGP bus as shown in Figure 6-2. The MGA-G100-AGP supports the AGP target interface as medium device (i.e.. it responds with PDEVSEL/ during the second

clock after PFRAME/ is asserted). It uses the AGP sideband signal addressing mechanism.

Figure 6-2: AGP Interface



(1) PIDSEL must be connected internally to PAD16.

6.3 Snooping

The MGA-G100 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-G100 is unable to process the access immediately, it takes control of the bus, and a retry cycle is performed.
2. If the MGA-G100 is able to process the access, the access is snooped, and the MGA-G100 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.

6.4 EPROM Devices

The MGA-G100 supports a few external devices (the SPI-EEPROM is a standard expansion device that is supported by the MGA-G100).

Figure 6-3 shows how to connect the EEPROM to the MGA-G100.

BIOS EEPROM

The MGA-G100 supports 32K x 8 EEPROMs, 64K x 8 EEPROMs, and 128 byte EEPROMs. The

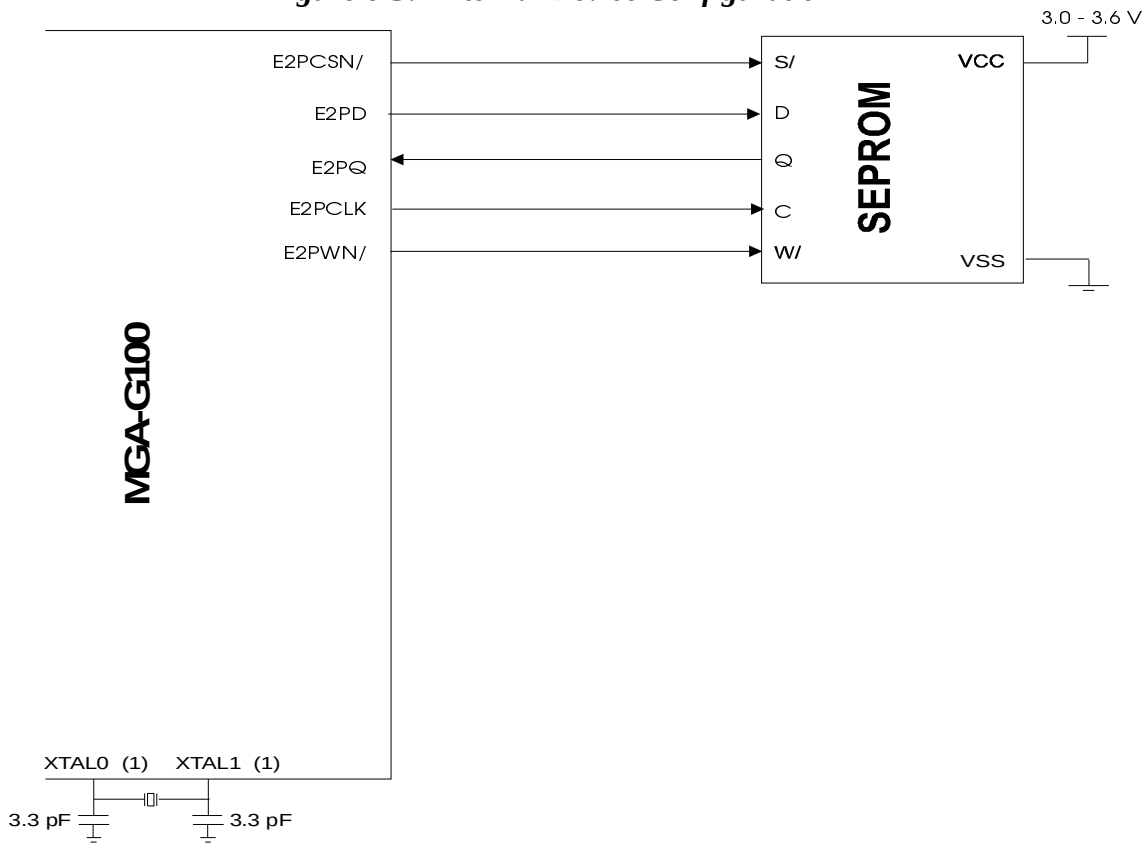
following table lists specific EPROM devices that have been verified to work with the MGA-G100:

A write cycle to the EEPROM has been defined in order to support flash memory. Another bit which locks write accesses to the EEPROM 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 requirements must be guaranteed by software (refer to the device specification and section 5.10 EEPROM Programming).

	EEPROM		
Manufacturer	***32K x 8***	***64K x 8***	***128 x 8***
AMD			
Atmel			AT 25010
SGS	M35560		ST 24x01

Figure 6-3: External Device Configuration



(1) If a local oscillator is used instead of a crystal, it is connected to XTAL0, and XTAL1 is left unconnected.

6.5 Memory Interface

6.5.1 SGRAM Configurations

The principal characteristics of the MGA-G100's 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-G100

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]
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 [4]	
Bank Activate / Row Address (Mask Enabled)	ACTM	L	L	H	H	H	X	V	Row Address [4]	
Read / Column Address (Auto-Precharge Disabled)	READ	L	H	L	H	L	X	V	L	Column Address [5]
Write / Column Address (Auto-Precharge Disabled)	WRITE	L	H	L	L	L	X	V	L	Column Address [5]
Block Write / Column Address (Auto-precharge Disabled)	BWRIT	L	H	L	L	H	X	V	L	Column Address [5] [6]
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 [7]
Write Data / Enable Read Output	X	X	X	X	X	X	L	X	X	X [7]
2 Legend: H = Logical High, L = Logical Low, V = Valid, X = “Don’t Care”, ‘/’ indicates an active low signal.										

[1] MCS = MCS<3:0>.

[2] The MGA-G100 supports CAS latency (CL=2, CL=3, CL =4); burst type = sequential; burst length = 1.
opcode1: ‘0’: CLbits:’0000’.

CLbits	Caslatency
010	2
011	3
100	4

[3] A5 = 1 for a mask register access, and A6 = 1 for a color register access. Both registers ***cannot*** be accessed simultaneously.
opcode2 = ‘00000100000’ <- load mask register
opcode2 = ‘00001000000’ <- load color register

[4] For 16 MBit SGRAM device: Row Address = MA<9:0>.
For 8 MBit SGRAM device: Row Address = MA<8:0>.

[5] The MGA-G100 does not support the auto-precharge function, so AP will always be forced low for READ/WRITE/BWRIT commands.
Column Address = MA<7:0>.

[6] MA<2:0> are ‘don’t care’ for block write commands.

[7] Not a command - ‘MDQ mask enable’.

2 Note that the MGA-G100 does ***not*** drive CKE: it should be driven high externally.

2 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: 11-bit Address Configuration (*memconfig *= 1) (16 Mbit device)

splitmode		MA										
		10(BS)	*9(AP)*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
0	Row	A11	A21	A20	A19	A18	A17	A16	A15	A14	A13	A12
	Column	A11	‘0’	‘0’	A10	A9	A8	A7	A6	A5	A4	A3
1	Row	A10	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11
	Column	A10	‘0’	‘0’	A9	A8	A7	A6	A23	A5	A4	A3

Table 6-3: 10-bit Address Configuration (*memconfig *= 0) (8 Mbit device)

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	A23	A5	A4	A3

Figure 6-4: SGRAM Connection, 8M bytes

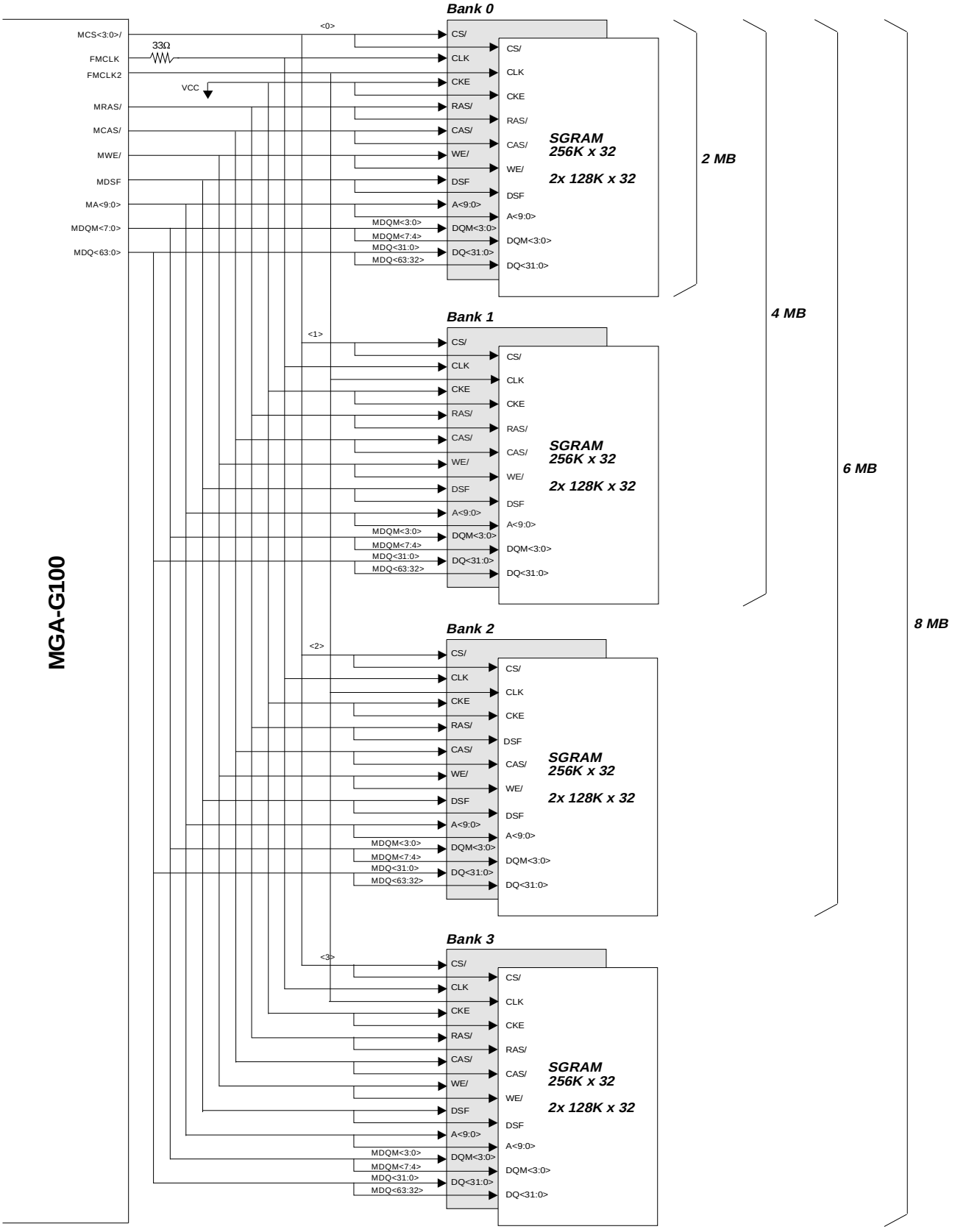
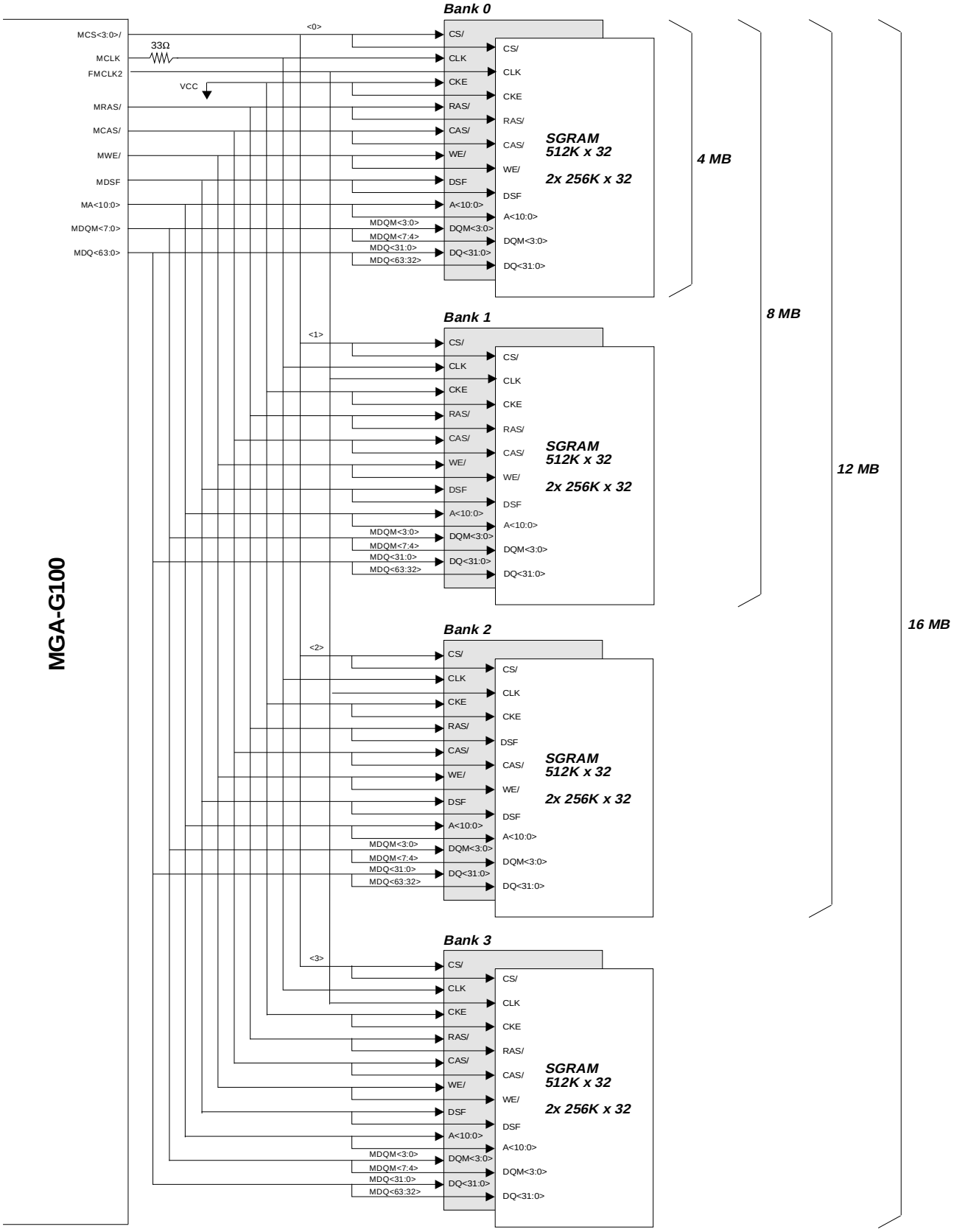
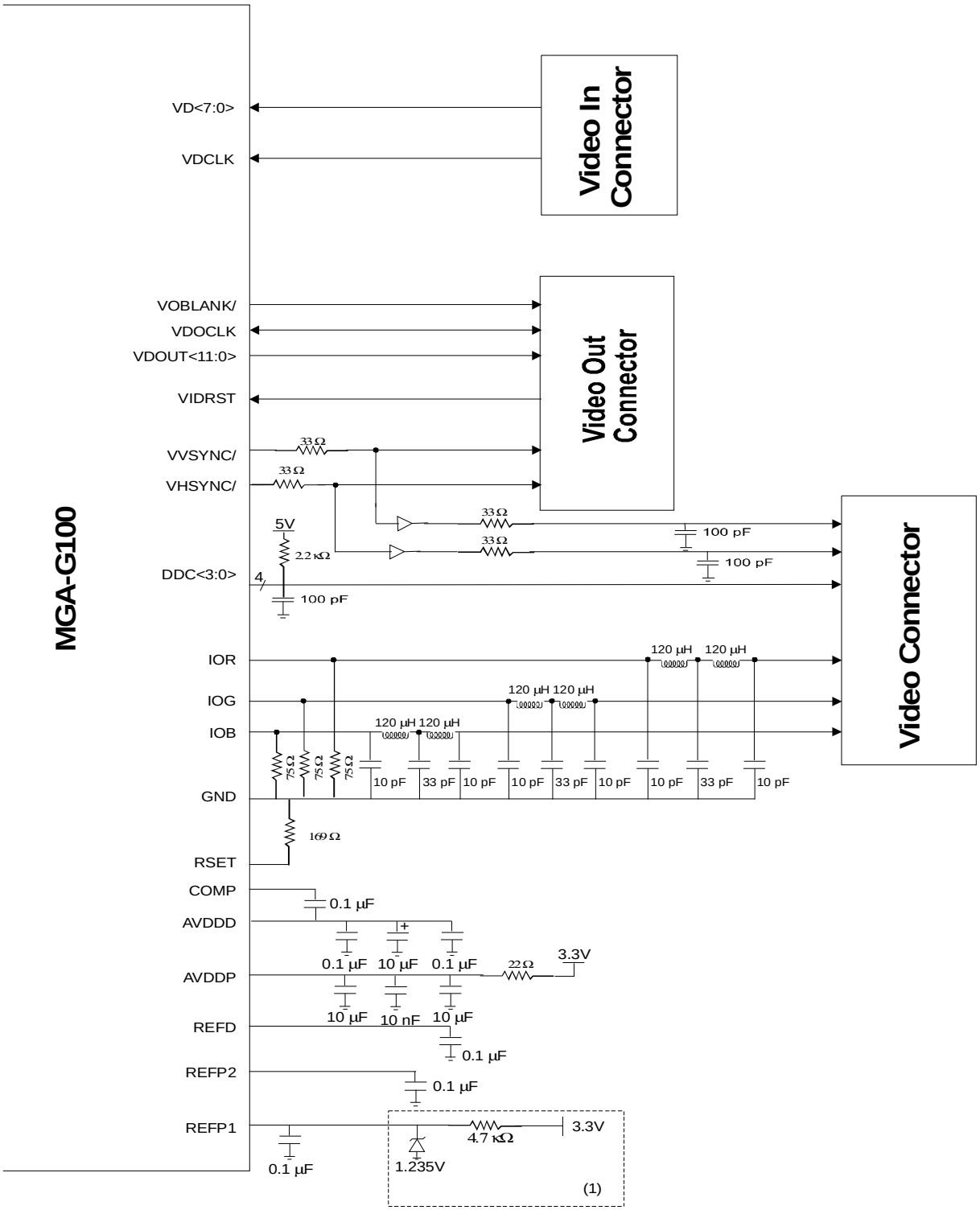


Figure 6-5: SGRAM Connection, 16M bytes



6.6 Video interface

Figure 6-6: Video Connector, Video In Connector, Video Out Connector



(1) Remove this circuitry when the internal reference is used.

6.6.1 Slaving the MGA-G100

This section describes the operations of the VIDRST (video reset input) signal. A VIDRST is detected on the first rising edge of **VDOCLK** where VIDRST is high. The video reset can affect both the horizontal and/or vertical circuitry.

The first time that the MGA-G100'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.

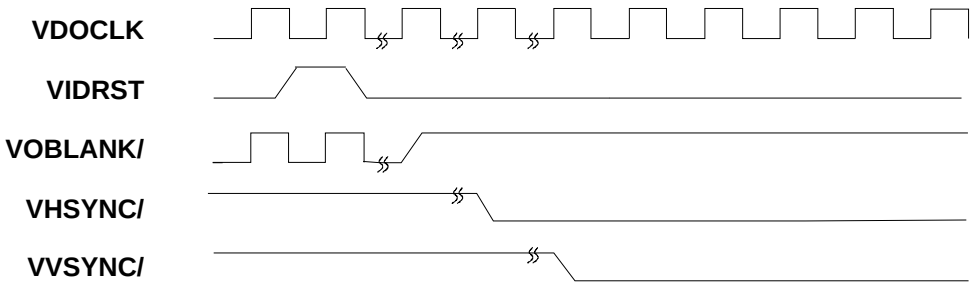
2Note: In order for the MGA-G100 to be synchronous with any other source, the MGA-G100 CRTC must be programmed with the same video parameters as that other source. **VDOCLK** can also be modulated in order to align both CRTCs.

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 SYNC.

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 SYNC in the even field.

Horizontal active and horizontal retrace are not affected by VIDRST when only the vertical reset is active. Figure 6-7 shows the relationship between VIDRST, the horizontal retrace, and the internal horizontal and vertical active signals, when both the horizontal and vertical counters are reset.

Figure 6-7: VIDRST, External HSYNC/VSYNC,and VOBLANK/



6.6.2 Genlock Mode

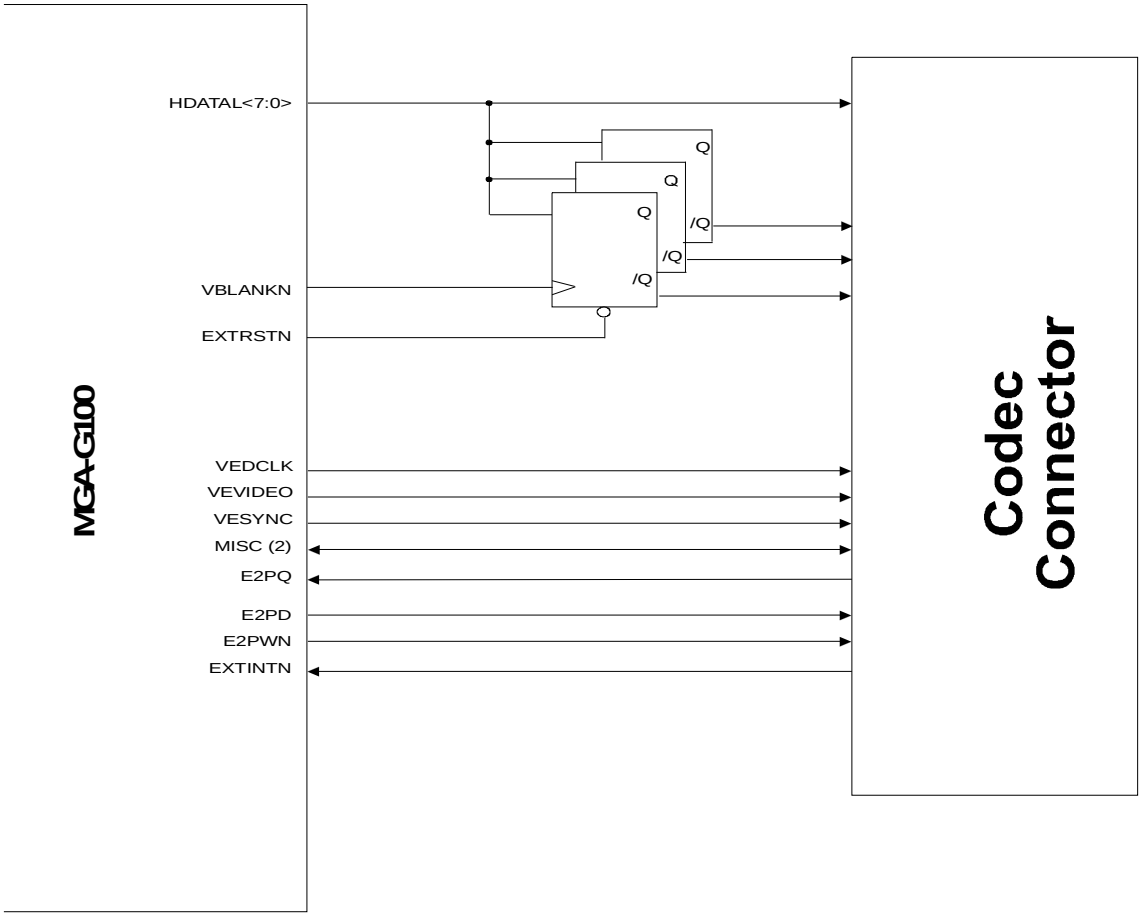
The VIDRST pin can be used to reset the CRTC horizontal counter to the **hsyncstr** and the vertical counter to the **vsyncstr** value. VIDRST must be maintained high for at least 1 **VDOCLK** cycle for the reset to take effect in the MGA-G100. When it is not used, the VIDRST pin ***must*** be maintained 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 (ie. a horizontal and vertical sync will be generated), according to the number of **VDOCLK**s shown in the following table:

Pixel Width	*Delay to VOBLANK/ active (VDOCLKs)*
BPP8	30
BPP15	20
BPP16	20
BPP24	16
BPP32	15

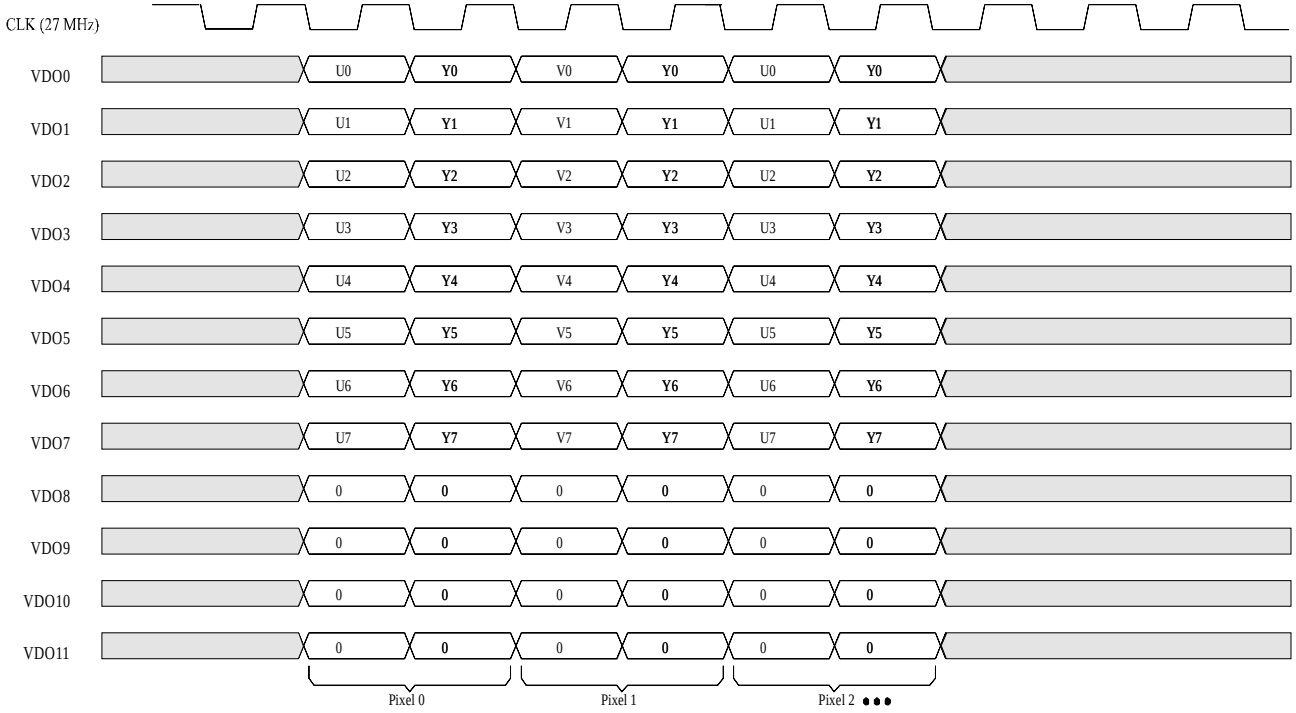
Genlocking is not supported in VGA mode.

Figure 6-8: Codec Connector



6.6.3 VIDEO OUT Data Sequence

Figure 6-9: MAVEN YUV 8 bit By-Pass Mode



note : CLK (27 MHz) from Decoder synchronizes the data

Figure 6-10: MAVEN RGB 12 bit Multiplexed Mode (MAFC Mode A)

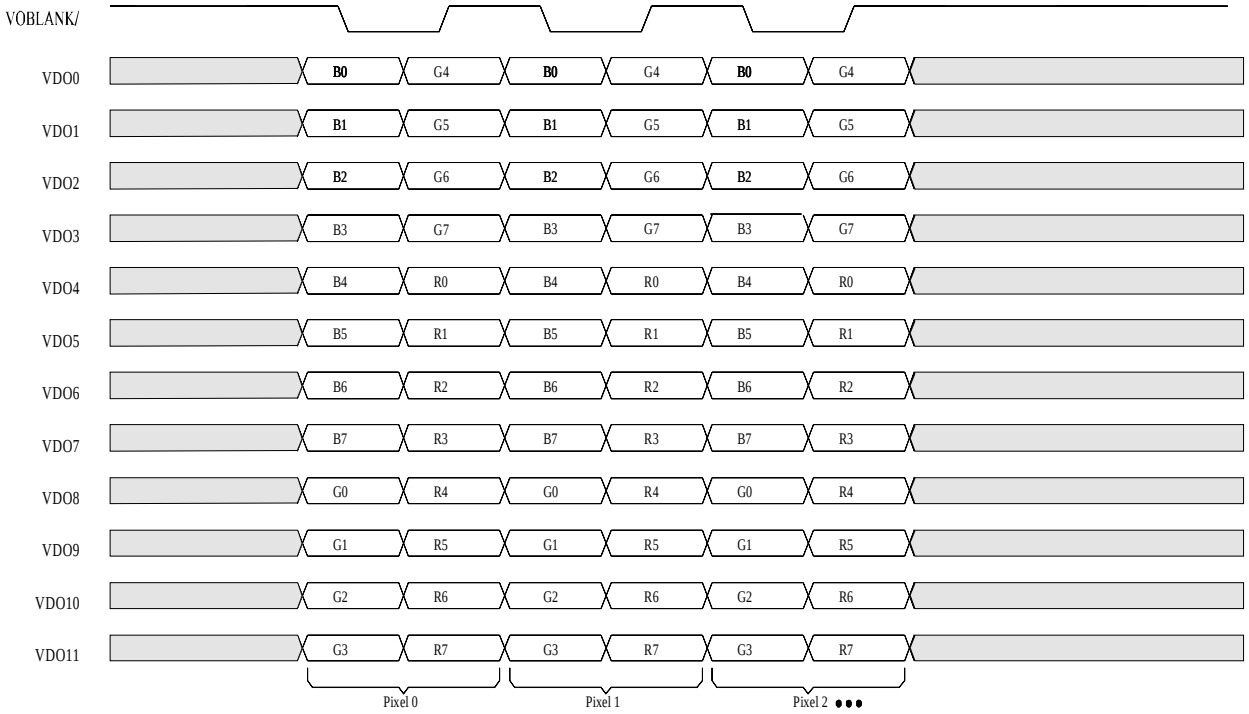
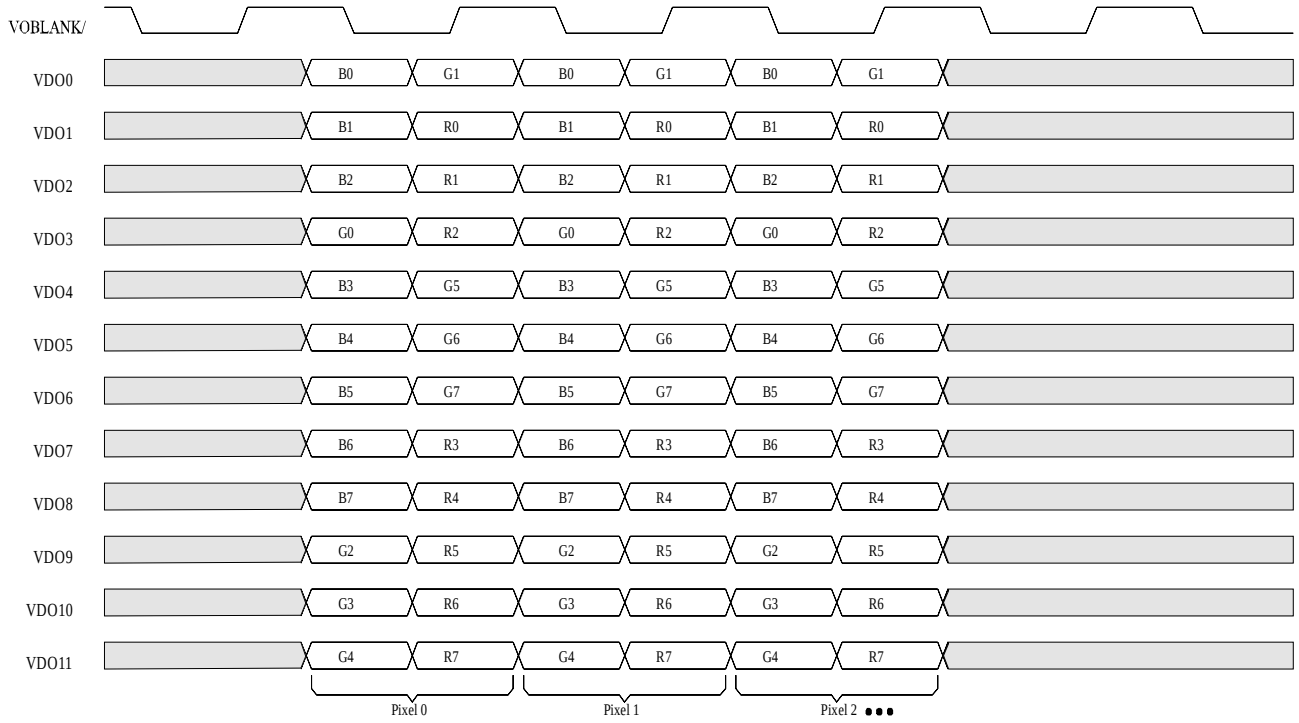
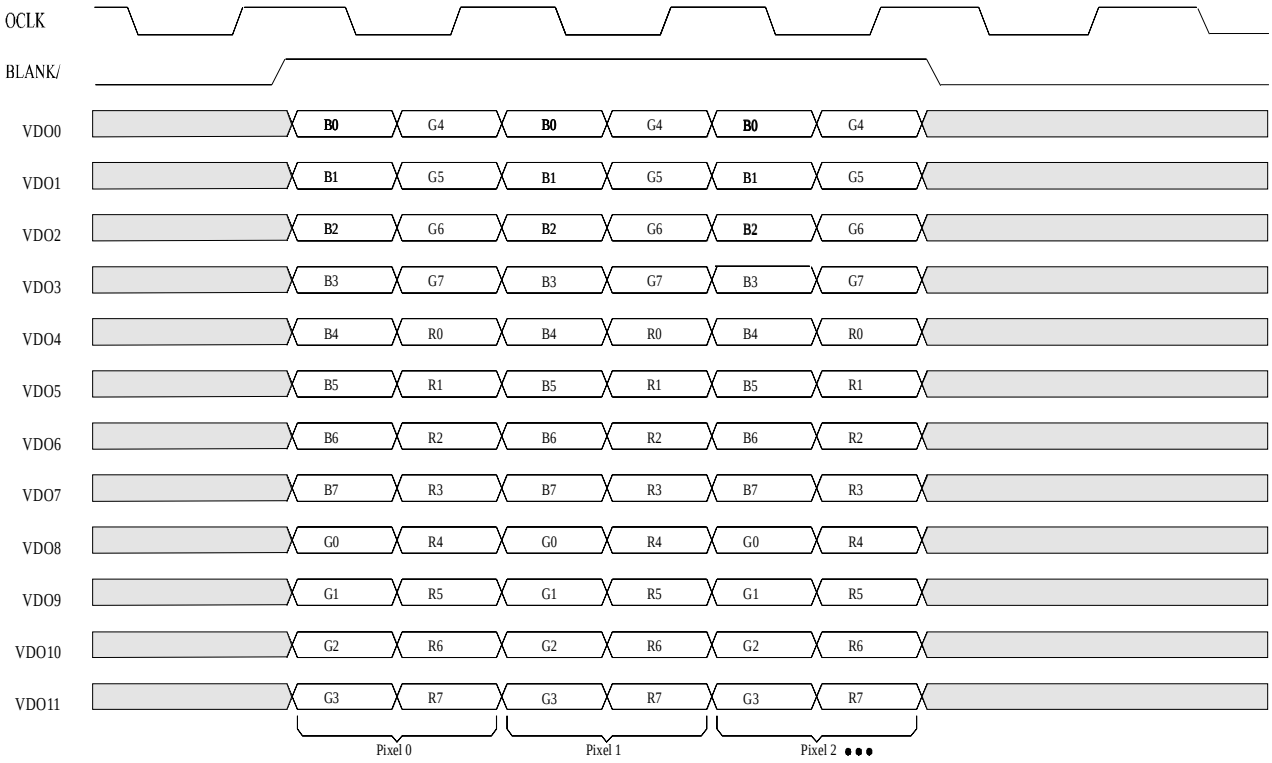


Figure 6-11: CHRONTEL RGB 12 bit Multiplexed Mode (MAFC Mode B)



note : VVSYNC and VHSYNC from TWISTER qualify the data

Figure 6-12: Panel Link Mode



note: Twister drives VDOCLK

Figure 6-13: Horizontal Input Timing (Panel Link)

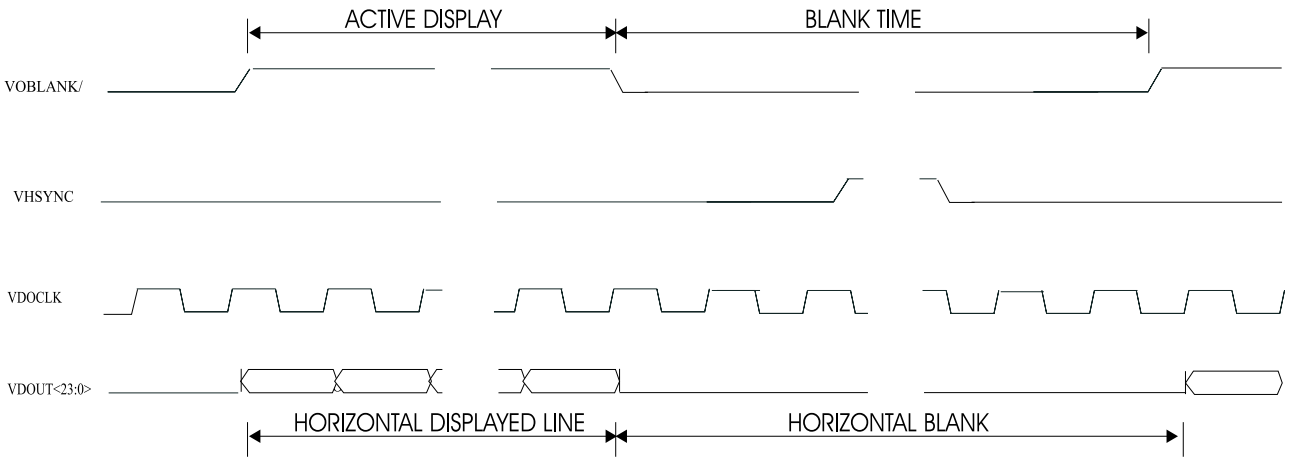
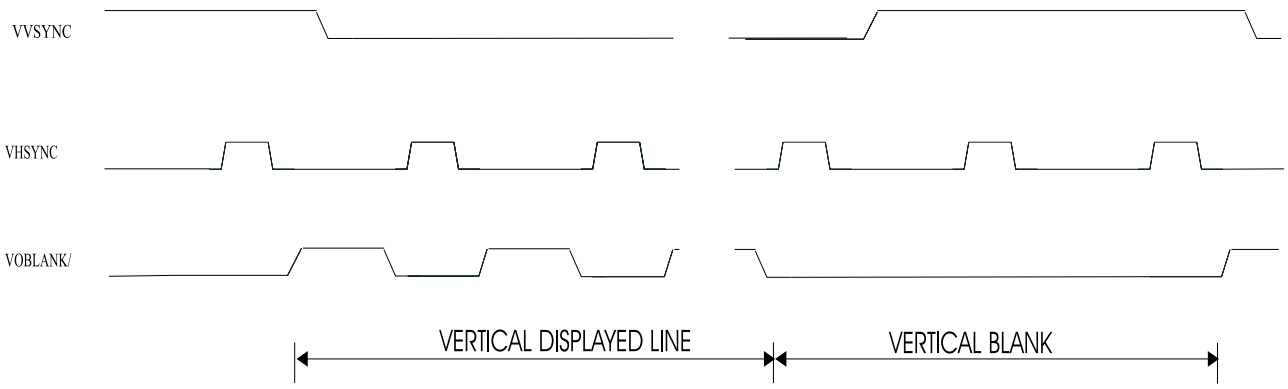


Figure 6-14: Vertical Input Timing (Panel LInk)



6.7 Crystal Resonator Specification

Frequency	27.000 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

Appendix A: Technical Information

A.1 Pin List

Table A-1: Pin Count Summary MGA-G100-PCI

Group	*Total*	*I*	*O*	*I/O*
Host PCI	48	4	3	41
Host Local	2	1	1	
Memory	93		29	64
Video Display	6		2	4
Video In	9	9		
MAFC/Video Out	15	1	13	1
Codec	12	3	1	8
SEPROM	5	1	4	
Analog Signals	11	6	5	
Miscellaneous Functions	3			3
Test	3	3		
VCC/GND	54			
Reserved	11			
PBGA Total	272			

Table A-2: Pin Count Summary MGA-G100-AGP

Group	*Total*	*I*	*O*	*I/O*
Host PCI	58	6	11	41
Host Local	2	1	1	
Memory	93		29	64
Video Display	6		2	4
Video In	9	9		
MAFC/Video Out	15	1	13	1
Codec	12	3	1	8
SEPROM	5	1	4	
Analog Signals	11	6	5	
Miscellaneous Functions	3			3
Test	3	3		
VCC/GND	48			
Reserved	7			
PBGA Total	272			

A.1.1 Host (PCI/AGP)

Name	*# Pins*	*Type*	*Description*
PAD<31:0>	32	I/O	PCI address and data bus. During the address phase of a PCI transaction, PAD contains a physical address. During the data phase, it contains the data that is read or written.
PCBE<3:0>/	4	I/O	PCI bus command, and byte enable. During the address phase, PCBE<3:0>/ provides the bus command. During the data phase, PCBE<3:0>/ is used as the byte enable.
PCLK	1	I	PCI bus clock. All PCI bus activities are referenced to this clock.
PDEVSEL/	1	I/O	Device select. Will be asserted when a transaction is within the MGA address range and space.
PFRAME/	1	I/O	Cycle frame. Indicates the beginning and duration of an access.
PGNT/	1	I	Grant. Indicates to the MGA-G100 that access to the PCI bus has been granted.
PIDSEL	1	I	Initialization device select. Used as a chip select during configuration read and write transactions. MGA-G100-PCI only.
PINTA/	1	O	Interrupt request signal.
PIRDY/	1	I/O	Initiator ready. Indicates the initiating agent's ability to complete the current data phase of the transaction (used in conjunction with PTRDY/). Wait cycles are inserted until both PIRDY/ and PTRDY/ are asserted together.
PPAR	1	O	PCI even parity bit for the PAD<31:0> and PCBE<3:0>/ lines. Parity is generated during read data phases and during the address phase throughout the PCI mastering cycle.
PREQ/	1	O	Request. Indicates to the arbiter that the MGA-G100 wishes to use the bus.
PRST/	1	I	PCI reset. This signal is used as the chip's hard reset.
PSTOP/	1	I/O	Stop. Forces the current transaction to terminate.
PTRDY/	1	I/O	Target ready. When asserted, indicates that the current data phase of the transaction can be completed (used in conjunction with PIRDY/). Wait cycles are inserted until both PIRDY/ and PTRDY/ are asserted together. In target mode, PTRDY/ is used as an input for snooping operations.
SBA<7:0>	8	O	Sideband Address port. Provides an additional bus for the MGA-G100-AGP to pass address and command.
ST<2:0>	3	I	Status. Provides additional information from the arbitor on what the MGA-G100-AGP may do when it receives a GNT/.

A.1.2 Host (Local Mode)

Name	*# Pins*	*Type*	*Description*
EXTINT/	1	I	External interrupt pin. Can be used by a companion chip to generate interrupts on the PCI bus. The interrupt is an active low level interrupt.
EXTRST/	1	O	External reset signal. Used to reset the expansion devices.

A.1.3 Memory Interface

Name	*# Pins*	*Type*	*Description*
MA<10:0>	11	O	Memory addresses (row, column, bank: multiplexed). MA<10:0> are used as either bank select or memory addresses, depending on the type of memory.
MCAS/	1	O	Column address strobe.
FMCLK	1	O	Memory clock to SGRAM.
FMCLK2	1	O	Memory clock to SGRAM.
MCS<3:0>/	4	O	Chip select.
MDQ<63:0>	64	I/O	Memory data bus. Used during read and write transactions.
MDQM<7:0>	8	O	Memory data input/output mask.
MDSF	1	O	Designated special function pin.
MRAS/	1	O	Row address strobe.
MWE/	1	O	Memory write enable

A.1.4 Video Display Interface

Name	*# Pins*	*Type*	*Description*
DDC<3:0>	4	I/O	Display Data Channel. Used to communicate with the monitor.
VHSYNC/	1	O	Horizontal sync.
VVSYNC/	1	O	Vertical sync.

A.1.5 Video In Interface

Name	*# Pins*	*Type*	*Description*
VD<7:0>	8	I	Video In Data. Referenced to VDCLK.
VDCLK	1	I	Video In Clock. Maximum: 30 MHz.

A.1.6 Video Out Interface

Name	*# Pins*	*Type*	*Description*
VDOCLK	1	I/O	Video Out Clock. Input for mafc modes A and B. Output in Panel link mode.
VDOUT<11:0>	12	O	Video Out Data. In Video In Bypass mode VDOUT<11:8>='0000' and VDOUT<7:0> are the VD<7:0> delayed by 1 clock cycle.
VOBLANK/	1	O	Video Out Blank. Gated feedback clock in mafc mode A, none-gated clock in mafc mode B, vblank/ in Panel link mode, and Video clock in Video In Bypass mode.
VIDRST	1	I	Video reset input. Used to synchronize the CRTC on an external source. The VIDRST pin must be forced inactive when not in use.

A.1.7 CODEC Interface

Name	*# Pins*	*Type*	*Description*
VBLANK/	1	O	Data strobe for exernal register
VEDCLK	1	I	CODEC address <2>
VESYNC	1	I	CODEC address <0>

VEVIDEO	1	I	CODEC address <1>
HDATA <7:0>	8	I/O	CODEC data port. Also used for \chip strapping HDATA <0> VGA boot strap or switch HDATA <1> BIOS EPROM/$\overline{\text{VPD EPROM}}$ installed strap or switch

A.1.8 SEPROM

Name	*# Pins*	*Type*	*Description*
E2PCLK	1	O	Clock for the Serial EPROM.
E2CS/	1	O	Serial EPROM chip select.
E2PD	1	O	Serial EPROM write data. Also, the Codec Interface read signal when Codec interface is enabled.
E2PQ	1	I	Serial EPROM read data. Also, the Codec Interface read signal when Codec intafece is enabled.
E2PW/	1	O	Serial EPROM write data. Also, the Codec interface write signal when Codec interface in enabled.

A.1.9 Analog Signals

Name	*# Pins*	*Type*	*Description*
COMP	1	O	Compensation. COMP provides compensation for the internal reference amplifier. A 0.1 uF ceramic capacitor is required between COMP and analog ground. The capacitor must be as close to the device as possible to avoid noise pick-up.
IOB	1	O	Analog current output blue. This output can drive a 37.5Ω load directly (doubly-terminated 75Ω line).
IOG	1	O	Analog current output green. This output can drive a 37.5Ω load directly (doubly-terminated 75Ω line).
IOR	1	O	Analog current output red. This output can drive a 37.5Ω load directly (doubly-terminated 75Ω line).
REFD REFP1 REFP2	3	I	Voltage reference for DACs and PLLs. An internal voltage reference of nominally 1.235 V may or may not be provided, which would require an external 0.1 uF ceramic capacitor between REF and analog GND. However, the internal reference voltage can be overdriven by an externally supplied reference voltage.
REFSSTL	1	I	Reference voltage for the SSTL/LVTTL I/O buffers.
RSET	1	I	Full-scale adjustment pin. A resistor connected between this pin and ground controls the full-scale range of the DACs.
XTAL1 XTAL0	1 1	O I	Connection for a series resonant crystal as a reference for the frequency synthesizer PLLs. XTAL0 may be used as a TTL reference clock (local oscillator) input, in which case XTAL1 is left unconnected.

A.1.10 Miscellaneous Functions

Name	*# Pins*	*Type*	*Description*
MISC<2:0>	3	I/O	Miscellaneous Control. General purpose I/O pins. MISC <0> can be used for I2CDAT MISC <1> can be used for I2CCLK MISC <2> can be used for CODEC Address <3> or EOI/

A.1.11 Test

Name	*# Pins*	*Type*	*Description*
TST<1:0>	2	I	These pins place the chip in test mode. They should be tied to a pull-up during normal operation.
RBFN_LFT	1	I	This pin is for chip test only, but should be connected to RBFN of the AGP bus for MGA-G100-AGP and should be tied to a pull-up for MGA-G100-PCI.

A.1.12 PCI VCC/GND

Name	*# Pins*	*Type*	*Description*
VDD	12	-	Attaches to +3.3 volts.
AVDDD	3	-	Analog, Attaches to +3.3 volts.
AVDDP	2	-	Analog VDD to PLLs.
DVDDD	1	-	Digital VDD of DAC, attaches to +3.3 volts.
DVDDP	3	-	Digital VDD of PLLs, attaches to +3.3 volts
VDD5	6	-	Attaches to +5 volts.
GND	21	-	Attaches to ground.
AGNDP	2	-	Analog ground of PLLs.
DGNDD	1	-	Digital ground of DAC, attaches to ground.
DGNDP	3	-	Digital ground of PLLs, attaches to ground.

A.1.13 AGP VCC/GND

Name	*# Pins*	*Type*	*Description*
VDD	12	-	Attaches to +3.3 volts.
AVDDD	3	-	Analog, attaches to +3.3 volts.
AVDDP	2	-	Analog VDD of PLLs.
DVDDD	1	-	Digital VDD of DAC, attaches to +3.3 volts.
DVDDP	3	-	Digital VDD of PLLs, attaches to +3.3 volts.
GND	21	-	Attaches to ground.
AGNDP	2	-	Analog ground to PLLs.
DGNDD	1	-	Digital ground of DAC, attaches to ground.
DGNDP	3	-	Digital ground of PLLs, attaches to ground.

A.2 Pinout Illustration and Table

The illustration below shows the locations of the MGA-G100's 272 pins on the chip. The table on the next page lists the signal names with their respective pin numbers, in numeric order.

Figure A-1: PCI Pinout Illustration (Bottom View)

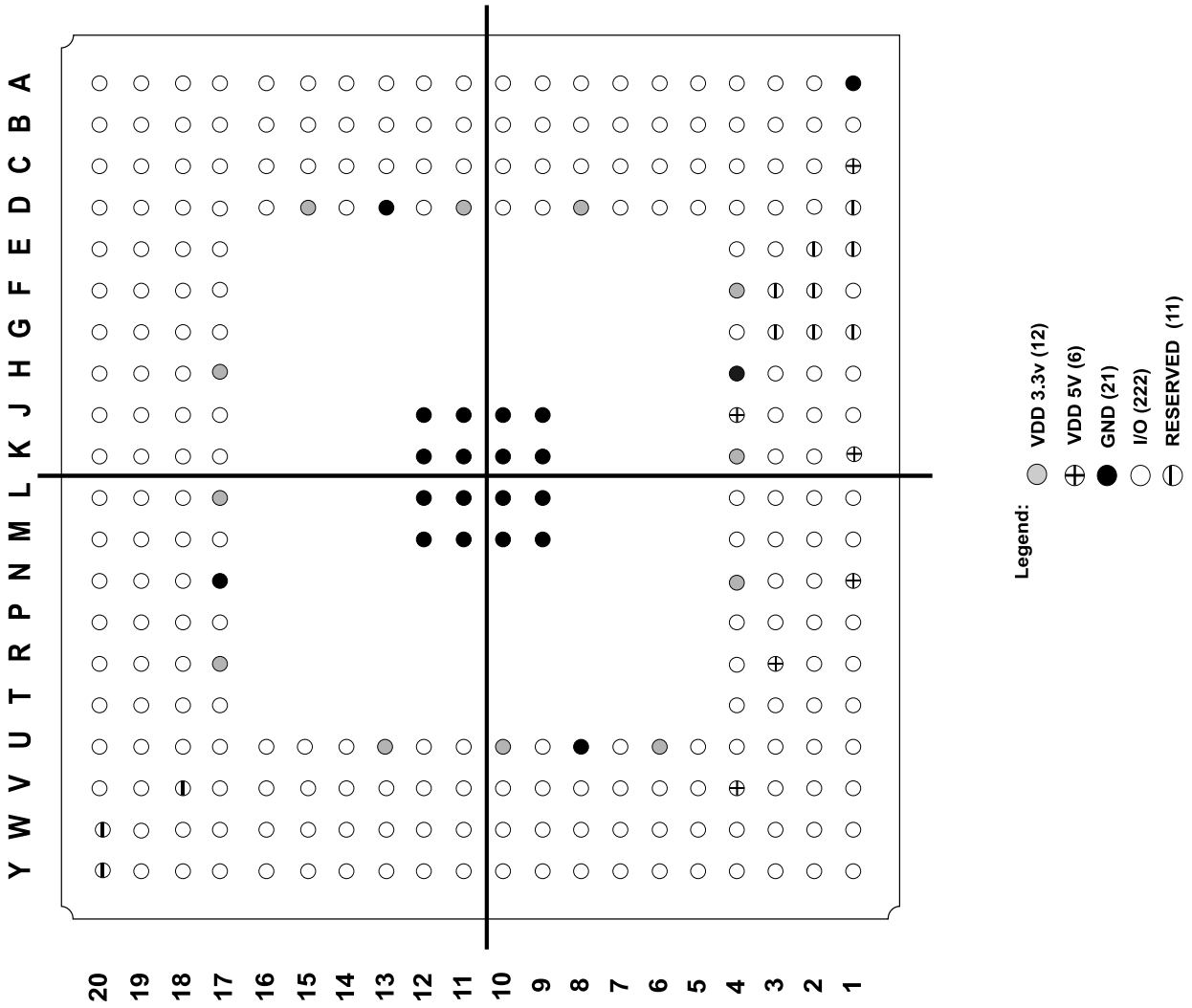
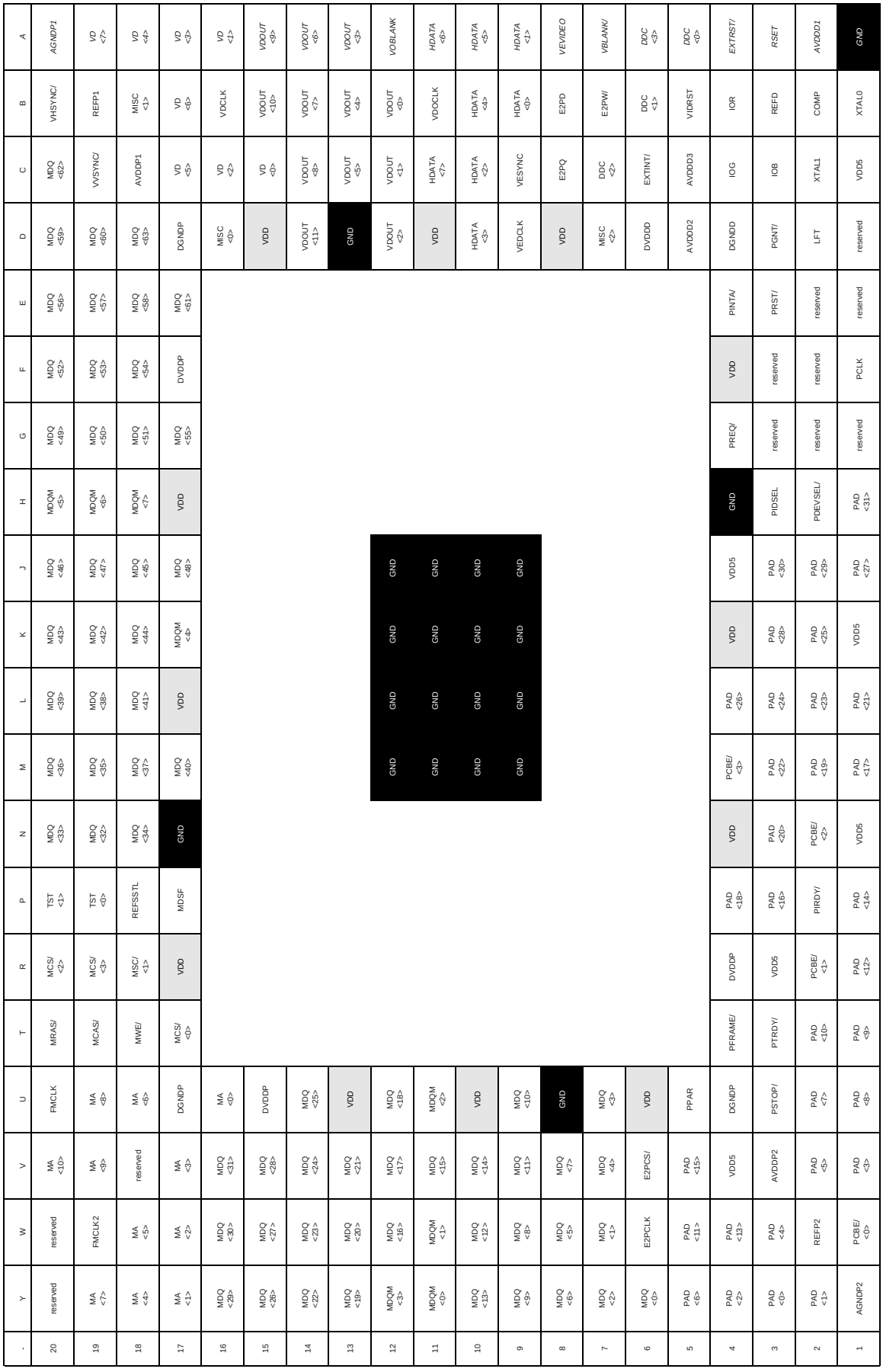


Table 1: PCI Pinout Legend (Bottom View)



-	Y	W	V	U	T	R	P	N	M	L	K	J	H	G	F	E	D	C	B	A
20	reserved	reserved	MA <10>	FMCLK	MRAS/	MCS/ <2>	TST <1>	MDQ <33>	MDQ <36>	MDQ <39>	MDQ <43>	MDQ <46>	MDQM <5>	MDQ <49>	MDQ <52>	MDQ <56>	MDQ <59>	MDQ <62>	VHSYNC/	AGNDP1
19	MA <7>	FMCLK2	MA <9>	MA <8>	MCAS/	MCS/ <3>	TST <0>	MDQ <32>	MDQ <35>	MDQ <38>	MDQ <42>	MDQ <47>	MDQM <6>	MDQ <50>	MDQ <53>	MDQ <57>	MDQ <60>	VVSYNC/	REFP1	VD <7>
18	MA <4>	MA <5>	reserved	MA <6>	MWE/	MSC/ <1>	REFSSTL	MDQ <34>	MDQ <37>	MDQ <41>	MDQ <44>	MDQ <45>	MDQM <7>	MDQ <51>	MDQ <54>	MDQ <58>	MDQ <63>	AVDDP1	MISC <1>	VD <4>
17	MA <1>	MA <2>	MA <3>	DGNDP	MCS/ <0>	VDD	MDSF	GND	MDQ <40>	VDD	MDQM <4>	MDQ <48>	VDD	MDQ <55>	DVDDP	MDQ <61>	DGNDP	VD <5>	VD <6>	VD <3>
16	MDQ <29>	MDQ <30>	MDQ <31>	MA <0>													MISC <0>	VD <2>	VDCLK	VD <1>
15	MDQ <28>	MDQ <27>	MDQ <28>	DVDDP													VDD	VD <0>	VDOUT <10>	VDOUT <9>
14	MDQ <22>	MDQ <23>	MDQ <24>	MDQ <25>													VDOUT <11>	VDOUT <8>	VDOUT <7>	VDOUT <6>
13	MDQ <19>	MDQ <20>	MDQ <21>	VDD													GND	VDOUT <5>	VDOUT <4>	VDOUT <3>
12	MDQM <3>	MDQ <16>	MDQ <17>	MDQ <18>					GND	GND	GND	GND					VDOUT <2>	VDOUT <1>	VDOUT <0>	VOBLANK
11	MDQM <0>	MDQM <1>	MDQ <15>	MDQM <2>					GND	GND	GND	GND					VDD	HDATA <7>	VDOCLK	HDATA <6>
10	MDQ <13>	MDQ <12>	MDQ <14>	VDD					GND	GND	GND	GND					HDATA <3>	HDATA <2>	HDATA <4>	HDATA <5>
9	MDQ <9>	MDQ <8>	MDQ <11>	MDQ <10>					GND	GND	GND	GND					VEDCLK	VESYNC	HDATA <0>	HDATA <1>
8	MDQ <6>	MDQ <5>	MDQ <7>	GND													VDD	E2PQ	E2PD	VEVIDEO
7	MDQ <2>	MDQ <1>	MDQ <4>	MDQ <3>													MISC <2>	DDC <2>	E2PW/	VBLANK/
6	MDQ <0>	E2PCLK	E2PCS/	VDD													DVDDD	EXTINT/	DDC <1>	DDC <3>
5	PAD <6>	PAD <11>	PAD <15>	PPAR													AVDDD2	AVDDD3	VIDRST	DDC <0>
4	PAD <2>	PAD <13>	VDD5	DGNDP	PFRAME/	DVDDP	PAD <18>	VDD	PCBE/ <3>	PAD <26>	PAD <28>	PAD <30>	GND	PREQ/	VDD	PINTA/	DGNDD	IOG	IOR	EXTRST/
3	PAD <0>	PAD <4>	AVDDP2	PSTOP/	PTRDY/	VDD5	PAD <16>	PAD <20>	PAD <22>	PAD <24>	PAD <25>	VDD5	PIDSEL	reserved	reserved	PRST/	PGNT/	IOB	REFD	RSET
2	PAD <1>	REFP2	PAD <5>	PAD <7>	PAD <10>	PCBE/ <1>	PIRDY/	PCBE/ <2>	PAD <19>	PAD <23>	PAD <25>	PAD <29>	PDEVSEL/	reserved	reserved	reserved	LFT	XTAL1	COMP	AVDDD1
1	AGNDP2	PCBE/ <0>	PAD <3>	PAD <8>	PAD <9>	PAD <12>	PAD <14>	VDD5	PAD <17>	PAD <21>	VDD5	PAD <27>	PAD <31>	reserved	PCLK	reserved	reserved	VDD5	XTAL0	GND

A.3 AGP Pinout Illustration and Table

The illustration below shows the locations of the MGA-G100's 272 pins on the chip. The table on the next page lists the signal names with their respective pin numbers, in numeric order.

Figure A-2: AGP Pinout Illustration (Bottom View)

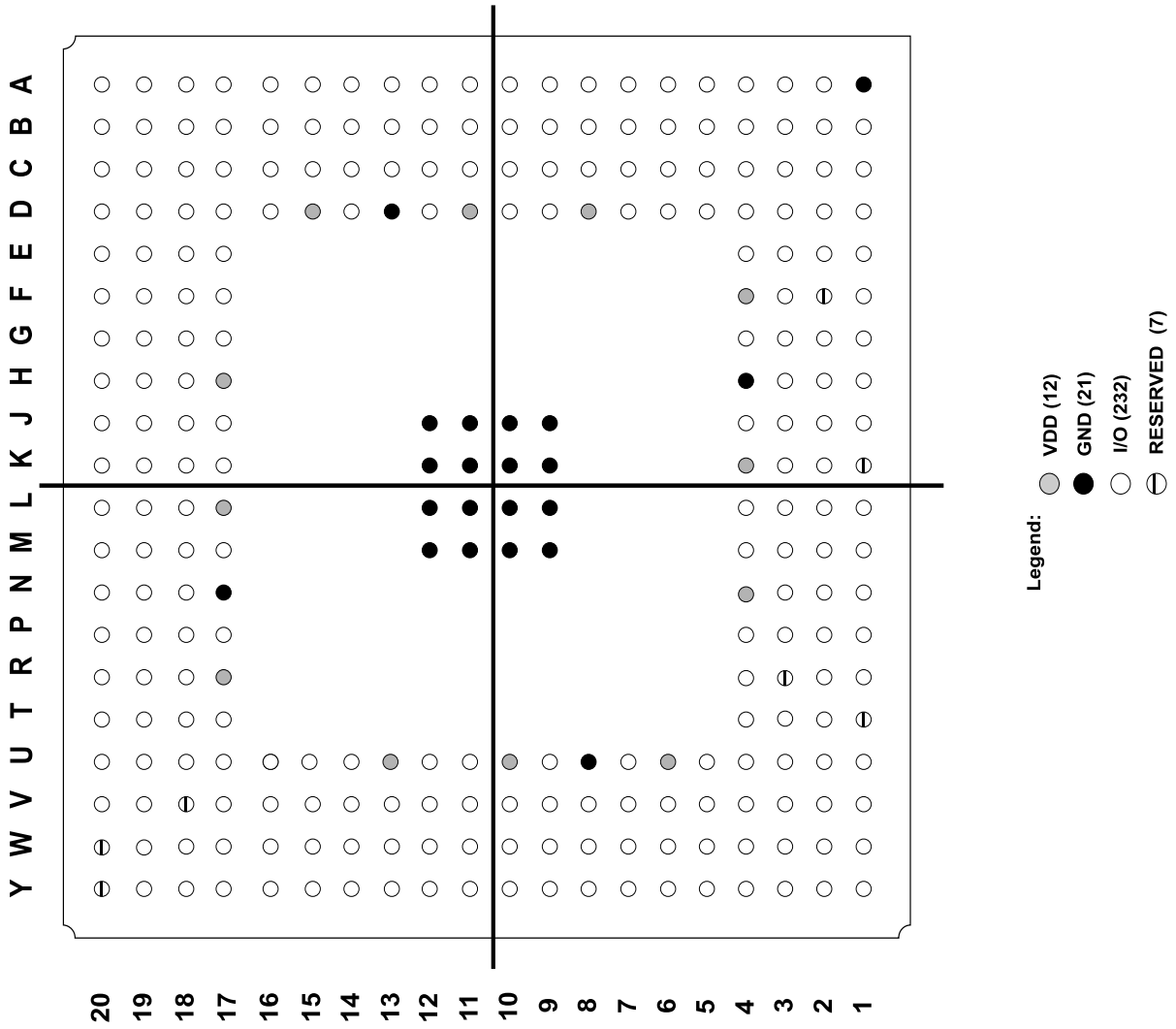


Table A-3: AGP Pinout Legend (Bottom View)

-	Y	W	V	U	T	R	P	N	M	L	K	J	H	G	F	E	D	C	B	A
20	reserved	reserved	MA <10>	FMCLK	MRAS/	MCS/ <2>	TST <1>	MDQ <33>	MDQ <36>	MDQ <39>	MDQ <43>	MDQ <46>	MDQM <5>	MDQ <49>	MDQ <52>	MDQ <56>	MDQ <59>	MDQ <62>	VHSYNC/	AGNDP1
19	MA <7>	FMCLK2	MA <9>	MA<8>	MCAS/	MCS/ <3>	TST <0>	MDQ <32>	MDQ <35>	MDQ <38>	MDQ <42>	MDQ <47>	MDQM <6>	MDQ <50>	MDQ <53>	MDQ <57>	MDQ <60>	VVSYNC/	REFP1	VD <7>
18	MA <4>	MA <5>	reserved	MA <6>	MWE/	MSC/ <1>	REFSSTL	MDQ <34>	MDQ <37>	MDQ <41>	MDQ <44>	MDQ <45>	MDQM <7>	MDQ <51>	MDQ <54>	MDQ <58>	MDQ <63>	AVDDP1	MISC <1>	VD <4>
17	MA <1>	MA <2>	MA <3>	DGNDP	MCS/ <0>	VDD	MDSF	GND	MDQ <40>	VDD	MDQM <4>	MDQ <48>	VDD	MDQ <55>	DVDDP	MDQ <61>	DGNDP	VD <5>	VD <6>	VD <3>
16	MDQ <29>	MDQ <30>	MDQ <31>	MA <0>													MISC <0>	VD <2>	VDCLK	VD <1>
15	MDQ <26>	MDQ <27>	MDQ <28>	DVDDP													VDD	VD <0>	VDOUT <10>	VDOUT <9>
14	MDQ <22>	MDQ <23>	MDQ <24>	MDQ <25>													VDOUT <11>	VDOUT <8>	VDOUT <7>	VDOUT <6>
13	MDQ <19>	MDQ <20>	MDQ <21>	VDD													GND	VDOUT <5>	VDOUT <4>	VDOUT <3>
12	MDQM <3>	MDQ <16>	MDQ <17>	MDQ <18>					GND	GND	GND	GND					VDOUT <2>	VDOUT <1>	VDOUT <0>	VOBLANK
11	MDQM <0>	MDQM <1>	MDQ <15>	MDQM <2>					GND	GND	GND	GND					VDD	HDATA <7>	VDOCLK	HDATA <6>
10	MDQ <13>	MDQ <12>	MDQ <14>	VDD					GND	GND	GND	GND					HDATA <3>	HDATA <2>	HDATA <4>	HDATA <5>
9	MDQ <9>	MDQ <8>	MDQ <11>	MDQ <10>					GND	GND	GND	GND					VEDCLK	VESYNC	HDATA <0>	HDATA <1>
8	MDQ <6>	MDQ <5>	MDQ <7>	GND													VDD	E2PQ	E2PD	VEVIDEO
7	MDQ <2>	MDQ <1>	MDQ <4>	MDQ <3>													MISC <2>	DDC <2>	E2PW/	VBLANK/
6	MDQ <0>	E2PCLK	E2PCS/	VDD													DVDDD	EXTINT/	DDC <1>	DDC <3>
5	PAD <6>	PAD <11>	PAD <15>	PPAR													AVDDD2	AVDDD3	VIDRST	DDC <0>
4	PAD <2>	PAD <13>	PAD <9>	DGNDP	PFRAME/	DVDDP	PAD <18>	VDD	PCBE/ <3>	PAD <26>	VDD	SBA_7	GND	PREQ/	VDD	PINTA/	DGNDD	IOG	IOR	EXTRST/
3	PAD <0>	PAD <4>	AVDDP2	PSTOP/	PTRDY/	reserved	PAD <16>	PAD <20>	PAD <22>	PAD <24>	PAD <28>	PAD <30>	SBA_5	SBA_1	SBA_2	PRST/	PGNT/	IOB	REFD	RSET
2	PAD <1>	REFP2	PAD <5>	PAD <7>	PAD <10>	PCBE/ <1>	PIRDY/	PCBE/ <2>	PAD <19>	PAD <23>	PAD <25>	PAD <29>	SBA_6	SBA_3	reserved	ST_1	RBFN	XTAL1	COMP	AVDDD1
1	AGNDP2	PCBE/ <0>	PAD <3>	PAD <8>	reserved	PAD <12>	PAD <14>	PDEVSEL/	PAD <17>	PAD <21>	reserved	PAD <27>	PAD <31>	SBA_4	PCLK	SBA_0	ST_2	ST_0	XTAL0	GND

A.4 Electrical Specification

A.4.1 DC Specifications

Table A-4: Absolute Maximum Rating

Symbol	*Parameter*	*Conditions*	*Min.*	*Max.*	*Units*	*Notes*
VDD3V	Power Supply Voltage		-0.5	4.6	V	
VDD5V	Power Supply Voltage		-0.5	6.6	V	
V_I	Input Voltage					
	IO-3, IO-6, IO-9, IO-12, I-0, I-S	V_I < VDD3 + 0.5V	-0.5	4.6	V	
	IO-6-5V, IO-9-5V	V_I < VDD3 + 3.0V	-0.5	6.6	V	
	I-PCI 33, IO-PCI 33	V_I < VDD5 + 0.5V	-0.5	6.6	V	(1)
	IO_SSTL1, IO_SSTL2	V_I < VDD3 + 0.3V	-0.3	4.6	V	
V_O	Output Voltage					(2)
	IO-3, IO-6, IO-9, IO-12	V_O < VDD3 + 0.5V	-0.5	4.6	V	
	IO-6-5V, IO-9-5V	V_O < VDD3 + 3.0V	-0.5	6.6	V	
	IO-PCI 33	V_O < VDD5 + 0.5V	-0.5	6.6	V	(1)
	IO_SSTL1, IO_SSTL2	V_O < VDD3 + 0.3V	-0.3	4.6	V	
I_O	Output Current					(2)
	IO-3			10	mA	
	IO-6			20	mA	
	IO-9			30	mA	
	IO-12			40	mA	
	IO-PCI33			?	mA	
	IO_SSTL1				mA	
	IO_SSTL2				mA	
T_A	Operating Temperature		-40	85	^oC	
T_{STG}	Storage Temperature		-65	150	^oC	

(1) MGA-G100-PCI only

(2) V_O: the range of voltage which will not cause damage when applied to the output pin.
I_O: the maximum current which will not cause damage when flowing to or from the output pin.

Caution: Exposure to the absolute maximum rating for extended periods may affect device reliability; exceeding the rating could cause permanent damage. The device should not be operated outside the recommended operating conditions.

Table A-5: Recommended Operating Conditions

Symbol	*Parameter*	*Min.*	*Max.*	*Units*
VDD5	Power Supply	4.75	5.25	V
VDD3		3.0	3.6	V
Vref	Input reference	1.3	1.7	V
VTT	Termination Voltage	Vref-0.05	Vref+0.05	
V_{IH}	High-Level Input Voltage			
	IO-12, I-0, I-S	2.0	VDD3	V
	IO-9-5V, I-0-5V	2.0	5.5V	V
	I-PCI33, IO-PCI33	2.0	5.5V	V
	IO-SSTL1, IO-SSTL2	Vref+0.4	VDD3	
V_{IL}	Low-Level Input Voltage			
	IO-12, I-0, I-S	0	0.8	V
	IO-9-5V, I-0-5V	0	0.8	V
	I-PCI33, IO-PCI33	0	0.8	V
	IO-SSTL1, IO-SSTL2	0	Vref-0.4	
t_r	Input Rise Time	0	200	ns
t_f	Input Fall Time	0	200	ns

Table A-6: DC Characteristics (VDD3=3.3 ±0.3V, VDD5 = 5.0 ±0.25V, TA = 0 to 55°) (Part 1 of 2)

Symbol	*Parameter*	*Conditions*	*Min.*	*Typ.*	*Max.*	*Units*	*Notes*
I_{OS}	Ouput Short-Circuit Current	V_O = 0V			-250	mA	(1)
I_i	Input Leakage Current	V_i = VDD3 or 0V			±10	µA	
I_{OL}	Low-Level Output Current	V_{OL} = 0.4V					
	O-3		3			mA	
	O-6		6			mA	
	O-9		9			mA	
	O-12, IO-12		12			mA	
	O-24		24			mA	
	IO-PCI33, O-PCI33					mA	(2)
	IO-SSTL1		8			mA	
	IO-SSTL2		16			mA	
I_{OH}	High-Level Output Current	V_{OH} = 2.4V					
	O-3		-3			mA	
	O-6		-6			mA	
	O-12, IO-12		-12			mA	
	O-24		-24			mA	
	IO-PCI33, O-PCI33					mA	(2)
	IO-SSTL1		-8			mA	
	IO-SSTL2		-16			mA	

Table A-6: DC Characteristics (VDD3=3.3 ±0.3V, VDD5 = 5.0 ±0.25V, TA = 0 to 55°) (Part 2 of 2)

Symbol	*Parameter*	*Conditions*	*Min.*	*Typ.*	*Max.*	*Units*	*Notes*
V_{OL}	Low-Level Output Voltage	I_{OL} = 0 mA			0.1	V	
	IO-SSTL				VTT-0.6	V	
	IO-SSTL2				VTT-0.8	V	
V_{OH}	High-Level Output Voltage	I_{OH} = 0 mA	VDD3 - 0.1			V	
	IO-SSTL1		VTT+0.6			V	
	IO-SSTL2		VTT+0.8			V	
θ_{JA}	Junction-to-Air Thermal Coefficient	No Air Flow			TBD	°c/w	(3)
C_{PIN}	Pin Capacitance	F = 1 MHz			7	pF	
ICC3	VDD3 Supply Current		TBD(4)	TBD	TBD	mA	
ICC5	VDD5 Supply Current		TBD(5)	TBD	TBD	mA	

(1) The Output Short-Circuit time is less than one second for one pin only.

(2) PCI buffers are characterized by their V/I curves (see Figure A-5)(MGA-G100-PCI only).

(3) All GND ball connected to PCB ground plane and all VDD3 balls connected to PCB VDD plane

(4) VDD3 Supply Current in PCI Bus Power Management state D3 hot.

(5) VDD5 Supply Current in PCI Bus Power Management state D3 hot (MGA -1264SSG-PCI only).

Table A-7: DAC DC Parameter List

Parameter	*Min.*	*Typ.*	*Max.*	*Units*
Resolution (each DAC)	8	8	8	Bits
Integral linearity error (each DAC)	-	-	+/-1	LSB
Differential linearity error (each DAC)	-	-	+/-1	LSB
Output current, White vs. Blank	-	19.05	-	mA
Output current, White vs. Black (7.5 IRE only)	-	17.62	-	mA
Output current, Black vs. Blank (7.5 IRE only)	-	1.44	-	mA
Blank level on IOR, IOB	-	0	-	uA
Blank level on IOG (SYNC enabled)	-	7.62	-	mA
Sync level on IOG (SYNC enabled)	-	0	-	uA
LSB Size	-	69.10	-	uA
DAC to DAC matching	-	-	-	%
Analog output compliance, Vcc	-	-	-	V
Analog output capacitance (f=1 Mhz, IOUT=0 mA)	10	-	20	pF
Voltage reference output, Vref	1.15	1.235	1.26	V
SENSE - voltage reference Vsref	-	335.0	-	mV

Figure A-3: SSTL Class 1 Buffer V/I Curves

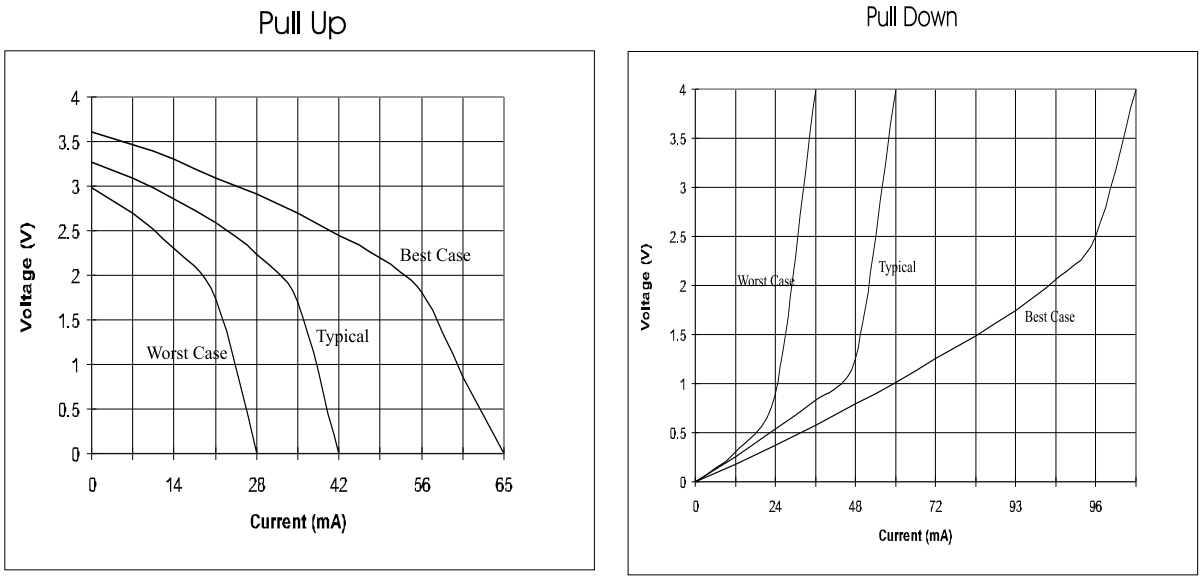


Figure A-4: SSTL Class 2 Buffer V/I Curves

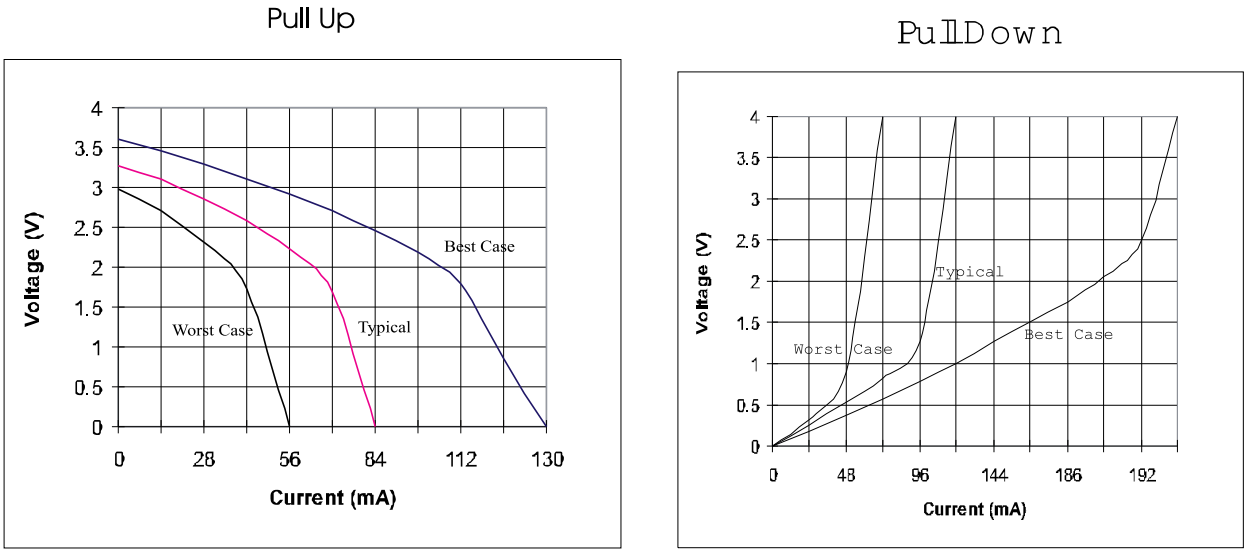
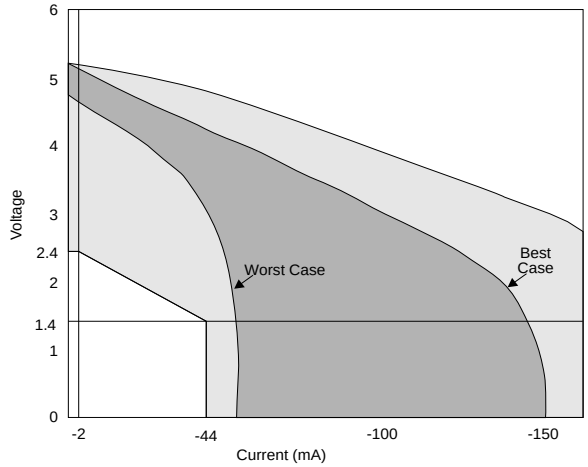


Figure A-5: V/I Curves for IO_PCI33 Buffers

Pull Up



Pull Down

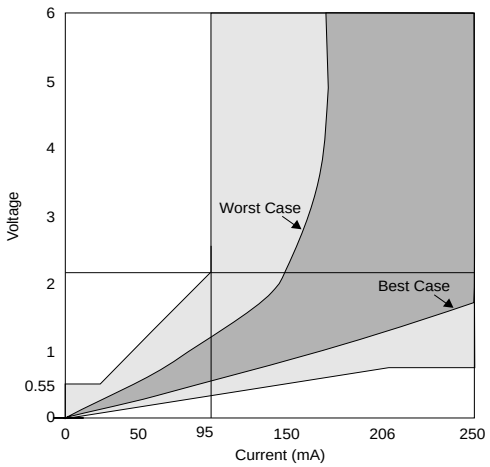
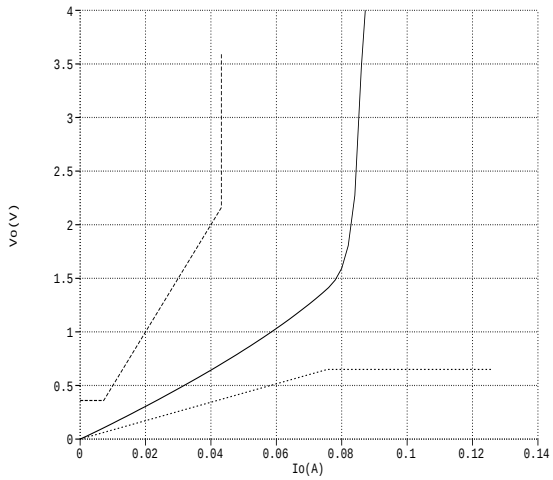


Figure A-6: AGP Buffer V/I Curve Pull-down (Best Case and Worst Case)



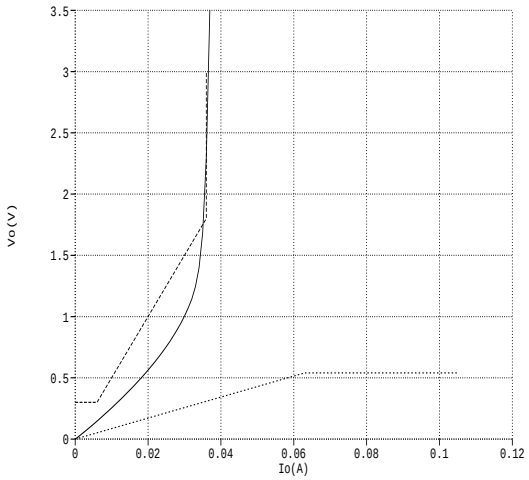
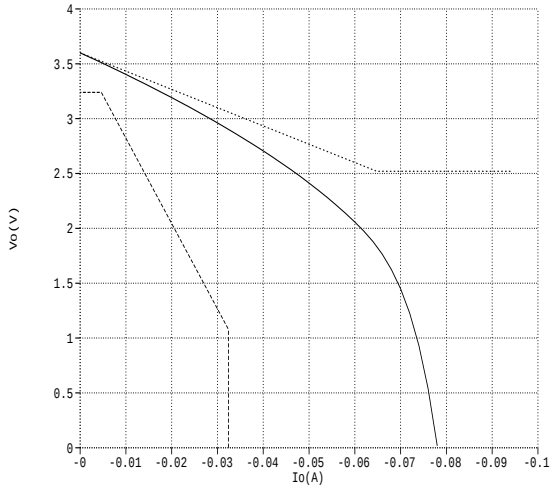


Figure A-7: AGP Buffer V/I Curve Pull-up (Best Case and Worst Case)



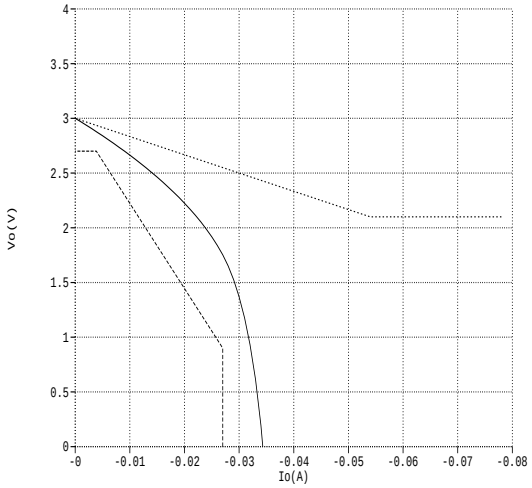


Table A-8: PCI Buffer Type and Pin Load (Part 1 of 2)

Signal Name	*Buffer Type*	*Load (pF)*		*Damping (ohms)*	*Notes*
		Min.	*Max.*		
PAD<31:0>	IO_PCI33	50	50	-	
PCBE<3:0>/	IO_PCI33	50	50	-	
PCLK	I_PCI33	-	-	-	
PDEVSEL/	IO_PCI33	50	50	-	
PFRAME/	IO_PCI33	50	50	-	
PGNT/	I_PCI33	-	-	-	
PIDSEL	I_PCI33	-	-	-	
PINTA/	IO_PCI33	50	50	-	
PIRDY/	IO_PCI33	50	50	-	
PPAR	IO_PCI33	50	50	-	(1)
PREQ/	IO_PCI33	50	50	-	(1)
PRST/	I_PCI33	-	-	-	
PSTOP/	IO_PCI33	50	50	-	
PTRDY/	IO_PCI33	50	50	-	
EXTINT/	I_0_5V	-	-		
EXTRST/	IO_6	50	50	-	(1)
E2PCLK	IO_6	50	50	-	(1)
E2PD	IO_6	50	50		(1)
E2PQ	I_0_5V	-	-	-	
E2PW/	IO_6	25	50		(1)
E2PCS/	IO_3	25	50		(1)
MA<10:0>	IO_SSTL2	15	65	-	(1)
MCAS/	IO_SSTL2	15	65	-	(1)
FMCLK	IO_SSTL2	15	65	-	
MCS<3:0>/	IO_SSTL1	12	25	-	(1)
MDQ<63:0>	IO_SSTL1	10	40	-	
MDQM<7:0>	IO_SSTL1	10	40	-	(1)
MDSF	IO_SSTL2	15	65	-	(1)
MRAS/	IO_SSTL2	15	65	-	(1)
MWE/	IO_SSTL2	15	65	-	(1)
VBLANK/	IO_6	15	35	33	(1)
VD<7:0>	I_0_5V	-	-	-	
VDCLK	I_0_5V_S	-	-	-	
VDOCLK	IO_9	10	20	-	
VDOUT<11:0>	IO_6	10	20	-	(1)
VEDCLK	IO_6	20	40	-	(1)
VESYNC	IO_6	20	40	-	(1)
VEVIDEO	IO_6	20	40	-	(1)
HDATA<7:0>	IO_6_5V	15	30		
VHSYNC/	IO_6	10	20	33	(1)

Table A-8: PCI Buffer Type and Pin Load (Part 2 of 2)

Signal Name	Buffer Type	Load (pF)		Damping (ohms)	Notes
		Min.	Max.		
VIDRST	I_0	-	-	-	
VOBLANK/	IO_6	10	20	-	(1)
VVSYNC/	IO_6	10	20	33	(1)
DDC<0>	IO_9_5V	50	50	-	
DDC<3:1>	IO_6_5V	50	50	-	
MISC<2:0>	IO_6_5V	50	50	-	
TST<1:0>	I_0	-	-	-	
RBFN_LFT	I_0	-	-	-	

(1) input mode only when TST<1:0> = '00' (HiZ mode for NAND Tree test)

Table A-9: AGP Buffer Type and Pin Load (Part 1 of 2)

Signal Name	Buffer Type	Load (pF)		Damping (ohms)	Notes
		Min.	Max.		
PAD<31:0>	IO_12	10	10	-	
PCBE<3:0>/	IO_12	10	10	-	
PCLK	I_0	-	-	-	
PDEVSEL/	IO_12	10	10	-	
PFRAME/	IO_12	10	10	-	
PGNT/	I_0	-	-	-	
PINTA/	IO_12	10	10	-	
PIRDY/	IO_12	10	10	-	
PPAR	IO_12	10	10	-	(1)
PREQ/	IO_12	10	10	-	(1)
PRST/	I_0	-	-	-	
PSTOP/	IO_12	10	10	-	
PTRDY/	IO_12	10	10	-	
SBA<7:0>	IO_12	10	10		(1)
ST<2:0>	I_0	-	-		
EXTINT/	I_0_5V	-	-		
EXTRST/	IO_6	50	50	-	(1)
E2PCLK	IO_6	50	50	-	(1)
E2PD	IO_6	50	50		(1)
E2PQ	I_0_5V	-	-	-	
E2PW/	IO_6	25	50		(1)
E2PCS/	IO_3	25	50		(1)
MA<10:0>	IO_SSTL2	15	65	-	(1)
MCAS/	IO_SSTL2	15	65	-	(1)
FMCLK	IO_SSTL2	15	65	-	
MCS<3:0>/	IO_SSTL1	12	25	-	(1)
MDQ<63:0>	IO_SSTL1	10	40	-	
MDQM<7:0>	IO_SSTL1	10	40	-	(1)

Table A-9: AGP Buffer Type and Pin Load (Part 2 of 2)

Signal Name	*Buffer Type*	*Load (pF)*		*Damping (ohms)*	*Notes*
		Min.	*Max.*		
MDSF	IO_SSTL2	15	65	-	(1)
MRAS/	IO_SSTL2	15	65	-	(1)
MWE/	IO_SSTL2	15	65	-	(1)
VBLANK/	IO_6	15	35	33	(1)
VD<7:0>	I_0_5V	-	-	-	
VDCLK	I_0_5V_S	-	-	-	
VDOCLK	IO_9	10	20	-	
VDOUT<11:0>	IO_6	10	20	-	(1)
VEDCLK	IO_6	20	40	-	(1)
VESYNC	IO_6	20	40	-	(1)
VEVIDEO	IO_6	20	40	-	(1)
HDATA<7:0>	IO_6_5V	15	30		
VHSYNC/	IO_6	10	20	33	(1)
VIDRST	I_0	-	-	-	
VOBLANK/	IO_6	10	20	-	(1)
VVSYNC/	IO_6	10	20	33	(1)
DDC<0>	IO_9_5V	50	50	-	
DDC<3:1>	IO_6_5V	50	50	-	
MISC<2:0>	IO_6_5V	50	50	-	
TST<1:0>	I_0	-	-	-	
RBFN_LFT	I_0	-	-	-	
(1)					

(1) input mode only when TST<1:0> = ‘00’ (HiZ mode for NAND Tree test)

A.4.2 AC Specification

A.4.2.1 DAC AC parameters

Table A-10: DAC Parameter List

Parameter	*Min.*	*Typ.*	*Max.*	*Units*
Analog output delay (Td) (1)	1.5	-	4.4	ns
Analog output settling time (Ts) (2)	4.2	-	5.5	ns
Analog output rise/fall time (Tr/f) (3)	1.8	-	2.3	ns
Glitch impulse	-	100	-	pV/s
DAC to DAC crosstalk	-	-	-	-
Analog output skew	-	-	-	ns

(1) Measured from the 90% point of the rising clock edge to 50% of full-scale transition.

(2) Measured from 50% of full-scale transition to settled output, within +/- 1 LSB.

(3) Measured between 10% and 90% of full-scale transition.

Table A-11: PLL Parameter List

Parameter	*Min.*	*Typ.*	*Max.*	*Units*
Pixel clock PLL frequency	6.25	-	-	MHz
PLL duty cycle variation	45	-	55	%
System PLL frequency	6.25	-	-	MHz
Frequency select to PLL output stable	-	1.00	-	ms
PLL phase jitter	-	-	-	-
PLL duty cycle jitter	-	-	-	-

A.4.2.2 Host Interface Timing

Figure A-8: PCI 33 MHz Waveform (MGA-G100-PCI only)

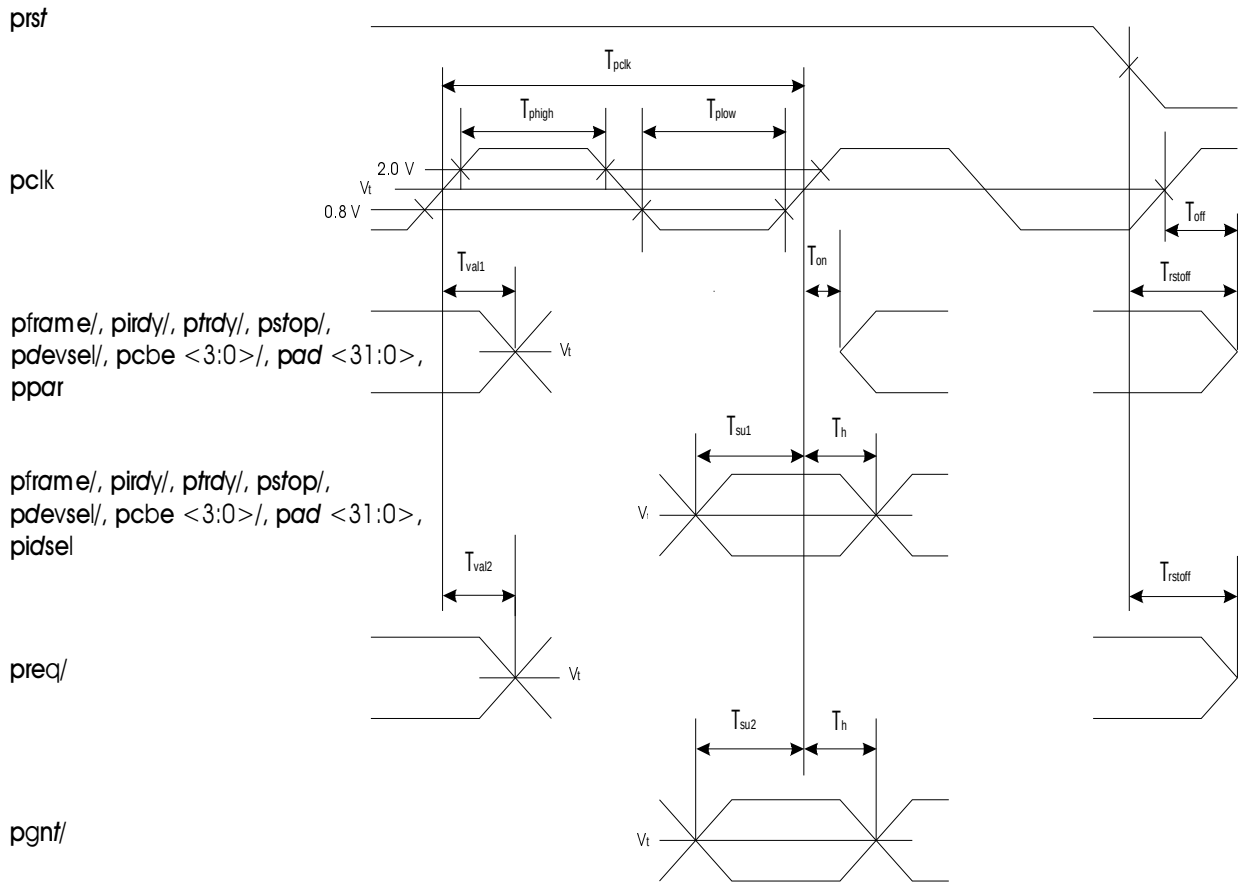


Table A-12: PCI 33 MHz 5V Signaling Environment Timing (MGA-G100-PCI only)[(1)]

Symbol	*Parameter*	*Min*	*Max*	*Unit*	*Notes*
T_{pclk}	PCLK cycle time	30		ns	
T_{plow}	PCLK low time	11		ns	
T_{phigh}	PCLK high time	11		ns	
T_{on}	Float to active delay	2		ns	
T_{val1}	PCLK to signal valid delay	2	11	ns	(2),(3)
T_{val2}	PCLK to signal valid delay	2	12	ns	(3),(4)
T_{off}	Active to float delay		28	ns	(5)
T_{rstoff}	Reset active to output float delay		40	ns	(5)
T_{su1}	Input setup time to PCLK	7		ns	(6)
T_{su2}	Input setup time to PCLK	10		ns	(7)
T_h	Input hold time from PCLK	0		ns	

(1) V_t = 1.5V

(2) Applies only to pframe/, pridy/, ptrdy/, pctop/, pdevsel/, pcbe <3:4>/,pad <31:0>, ppar

(3) Minimum times are evaluated with 0 pF lumped loud.
Maximum times are evaluated with 50 pF lumped loud.

(4) Applies only to preq/

(5) Hi-Z or off-state is achieved when the total current delivered through the component pin is less than or equal to the leakage current specification.

(6) Applies only to pfame/, pridy/, ptrsy/, pstop/, pdevsel/, pcbe <3:0>,pad <31:0> pidsel

(7) Applies only to pgnt/

Figure A-9: AGP 1X Waveform (MGA-G100-AGP only)

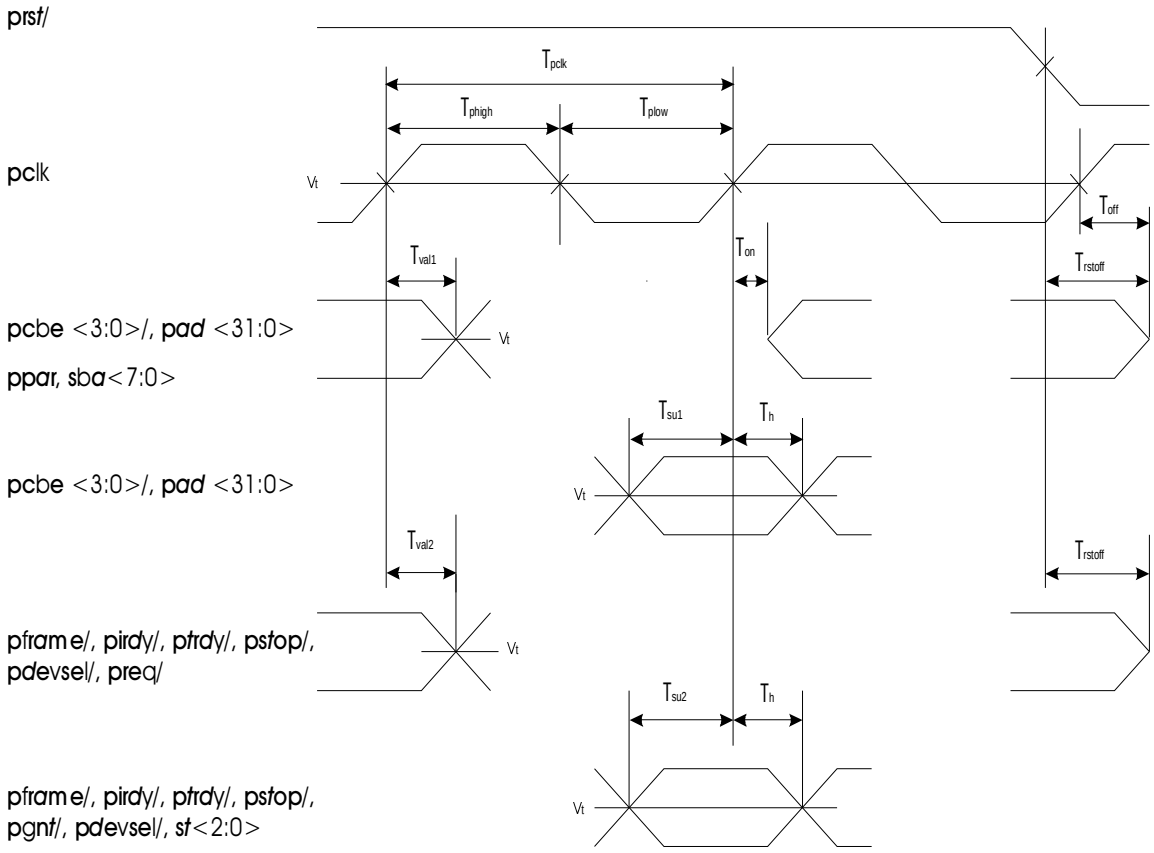


Table A-13: AGP1X Timing (MGA-G100-AGP only)[1]

Symbol	*Parameter*	*Min*	*Max*	*Unit*	*Notes*
T_{pclk}	PCLK cycle time	15.0		ns	
T_{plow}	PCLK low time	6.0		ns	
T_{phigh}	PCLK high time	6.0		ns	
T_{on}	Float to active delay	1.0	6.0	ns	
T_{val1}	PCLK to signal valid delay	1.0	6.0	ns	(2)
T_{val2}	PCLK to signal valid delay	1.0	5.5	ns	(3)
T_{off}	Active to float delay	1.0	14.0	ns	(4)
T_{rstoff}	Reset active to output float delay		40.0	ns	
T_{su1}	Input setup time to PCLK	5.5		ns	(5)
T_{su2}	Input setup time to PCLK	6.0		ns	(6)
T_h	Input setup time from PCLK	0		ns	

(1) Timings are evaluated with a 10pF lumped load.
Vt = 0.4 V_{DD}

(2) Applies only to data signals: pcbe<3:0., pad<31:0>, ppar, and sba<7:0>.

(3) Applies only to control signals: pdevsel/, prtdy/, pstop/, pframe/, pirdy/, and preq/.

(4) Hi-Z or off state is achieved when the total current delivered through the component pin is less than or equal to the leakage current specification.

(5) Applies only to data signals: pad<31:0> and pcbe<3:0>.

(6) Applies only to control signals: pdevsel/, ptndy/, pstop/, pframe/, pirdy/, st<2:0>, and psnt/.

A.4.2.3 Internal Clock Timing

Table A-14: Internal Clocks

Internal Clock	*Maximum Frequency (MHz)*
MCLK	76
GCLK	55.5
PIXCLK	186
VCLK	78.7

A.4.2.4 Serial EPROM Device Timing

Figure A-10: Serial EPROM Read Waveform

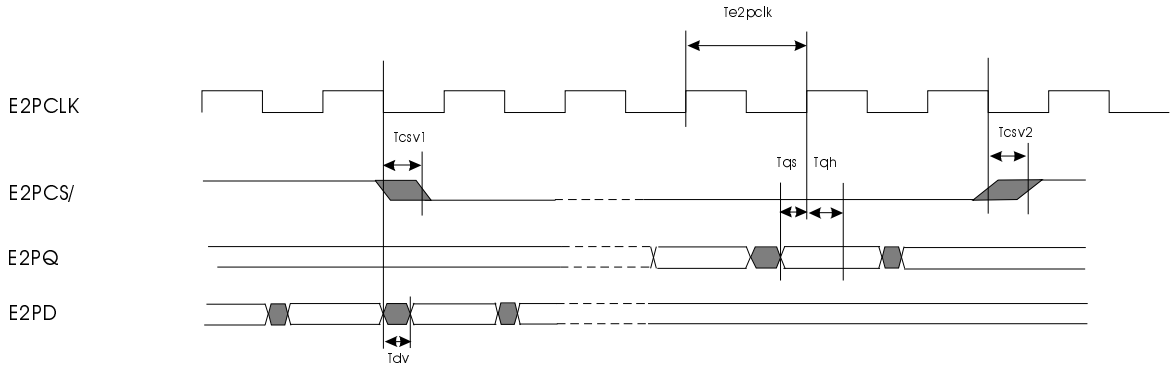


Table A-15: Serial EPROM Read/Write Parameter List

Name	*Min. (ns)*	*Max. (ns)*	*Comment*
Te2pclk	200	-	Clock Period
Tcsv1	0	10	Clock low to CS/ low
Tcsv2	0	10	Clock low to CS/ high
Tqs	10	-	Q setup required
Tqh	5	-	Q hold required
Tdv	0	10	D valid

A.4.2.5 MAFC Feature Connector

Figure A-11: MAFC Waveform (MAFC Mode A)

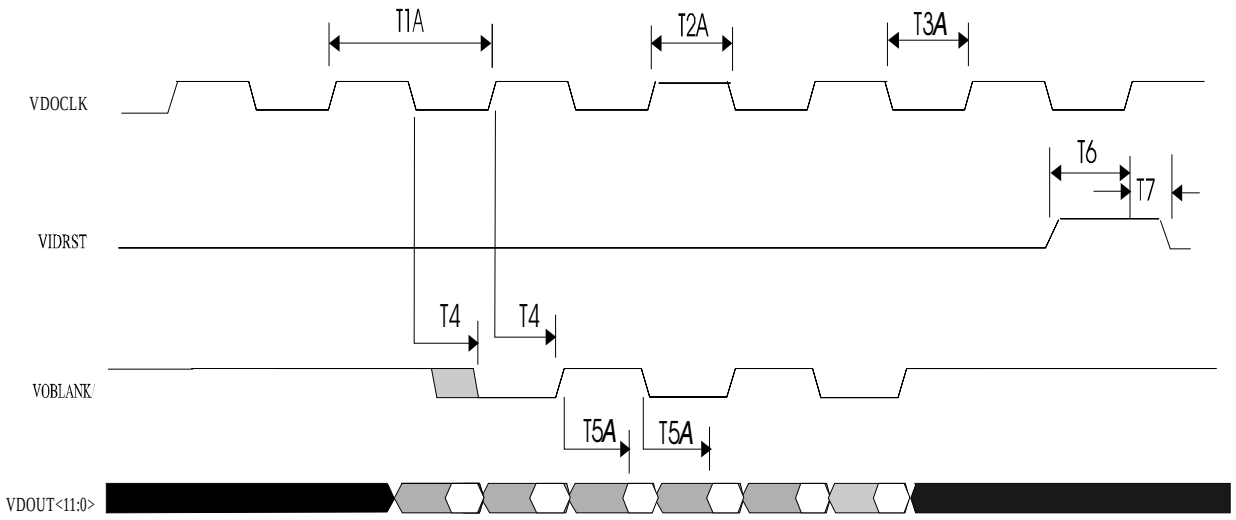
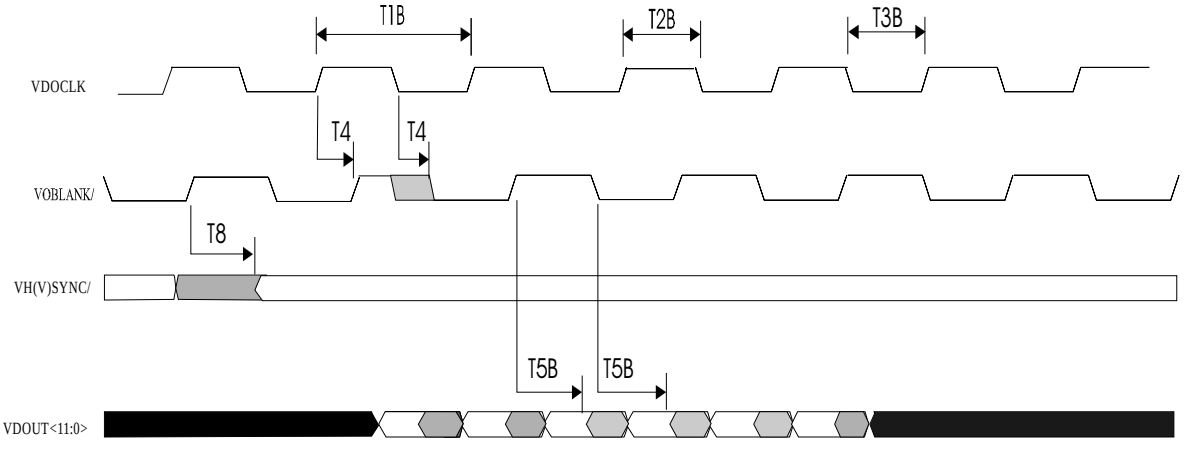


Figure A-12: MAFC Waveform (MAFC Mode B)



2

Figure A-13: Panel Link Mode

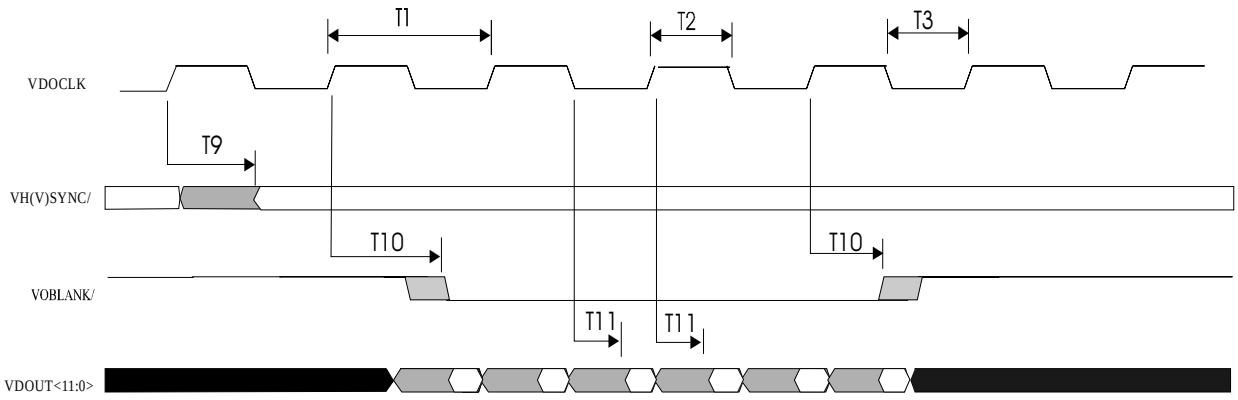


Figure A-14: Bypass Mode

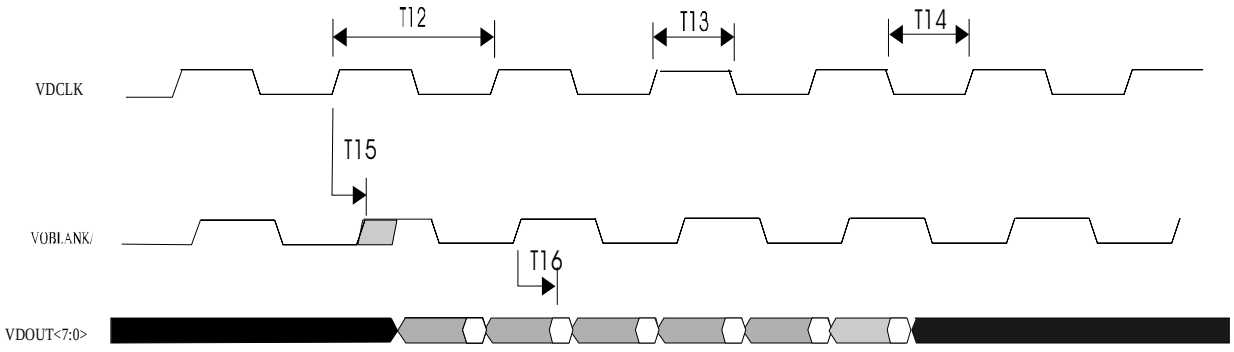


Table A-16: MAFC Waveforms data information

Name	*Min. (ns)*	*Max. (ns)*	*Comment*
T1A	15	-	VDOCLK period Mode A
T2A	6.75	-	VDOCLK high Mode A
T3A	6.75	-	VDOCLK low Mode A
T1B	20	-	VDOCLK period Mode B
T2B	8	-	VDOCLK high Mode B
T3B	8	-	VDOCLK low Mode B
T4	2	7	VDOCLK => VOBLANK/
T5A	1	5	Valid data time (VOBLANK/ => VDOUT)
T5B	2	7	Valid data time (VOBLANK/ => VDOUT)
T6	2	-	Setup time (VIDRST => VDOCLK)

Table A-16: MAFC Waveforms data information

Name	*Min. (ns)*	*Max. (ns)*	*Comment*
T7	0	-	Hold time (VIDRST => VDOCLK)
T8	2	5	VOBLANK/ => VH(V)SYNC
T9	1	2	VDOCLK => VH(V)SYNC
T10	1	4	VDOCLK => VOBLANK/
T11	2	6	Valid data time (VDOCLK => VDOUT)
T12	37	-	VDCLK period
T13	14	-	VDCLK high
T14	14	-	VDCLK low
T15	3	9	VDCLK -> VOBLANK/
T16	2	8	VDCLK -> VDOUT<7:0>

A.4.2.6 Memory Interface Timing

Figure A-15: Memory Interface Waveform

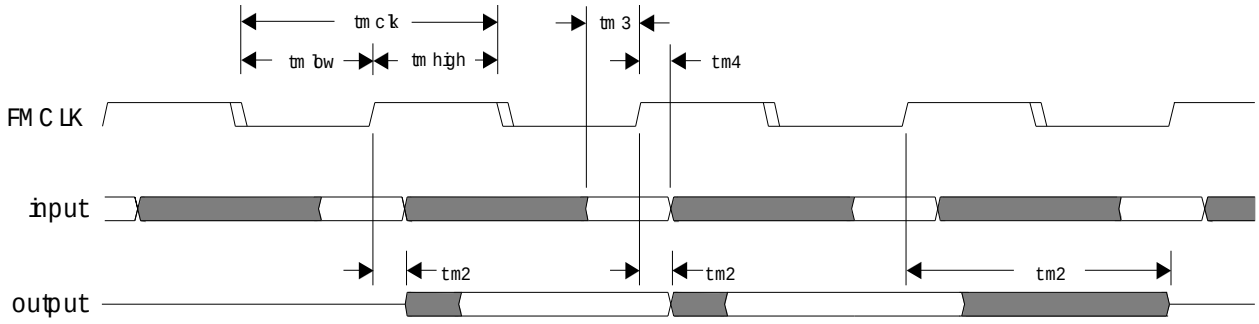


Table A-17: Memory Interface Parameter List

Timing	*Description*	*Min. (ns)*	*Max. (ns)*
Tmclk	FMCLK period	8.4	-
Tmlow	FMCLK low	3.1	-
Tmhigh	FMCLK high	3.5	-
Tm2	MDQ, MDQM, MA, MCAS/, MCS/, MRAS/, MWE/, MDSF clock --> output	1	4
Tm3	MDQ setup time	0.5	-
Tm4	MDQ hold time	2	-
Tm5	clock -> MDQ out low-z	0.85	-
Tm6	clock -> MDQ out high-z	-	1.88

Figure A-16: Read Followed by Precharge (Tcl=3)

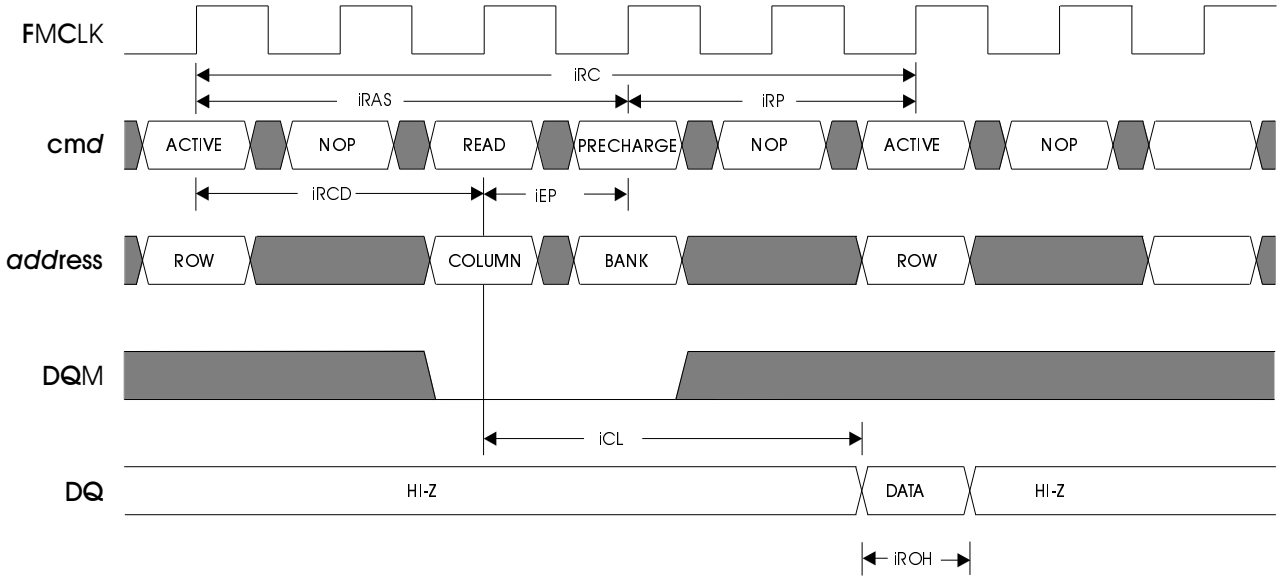


Figure A-17: Read Followed by a Precharge (Tcl = 2)

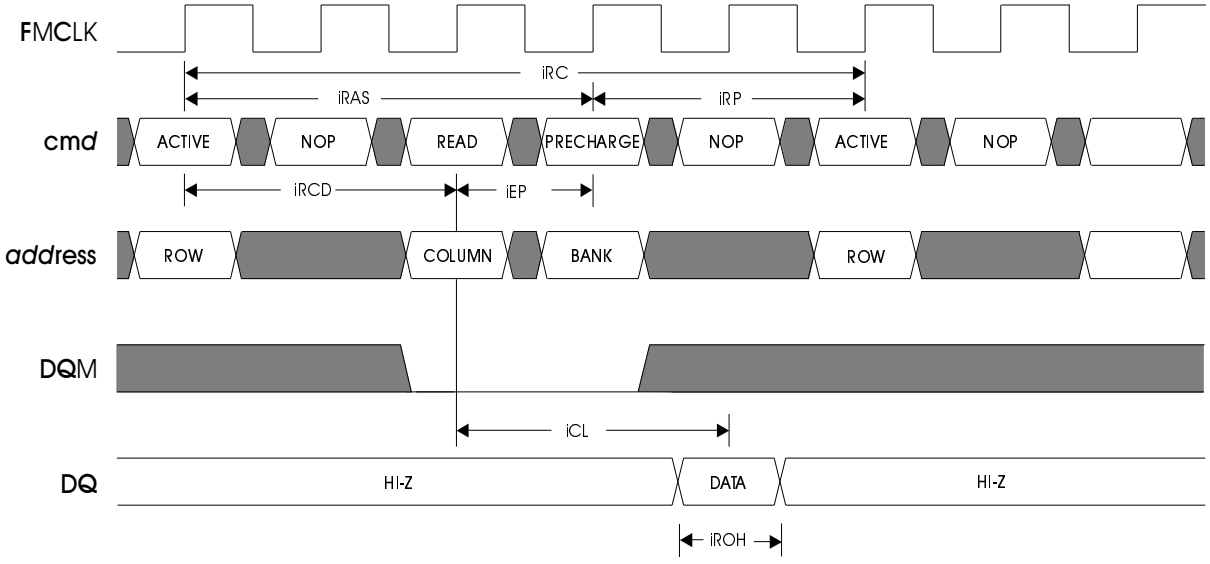


Figure A-18: Power-On Sequence

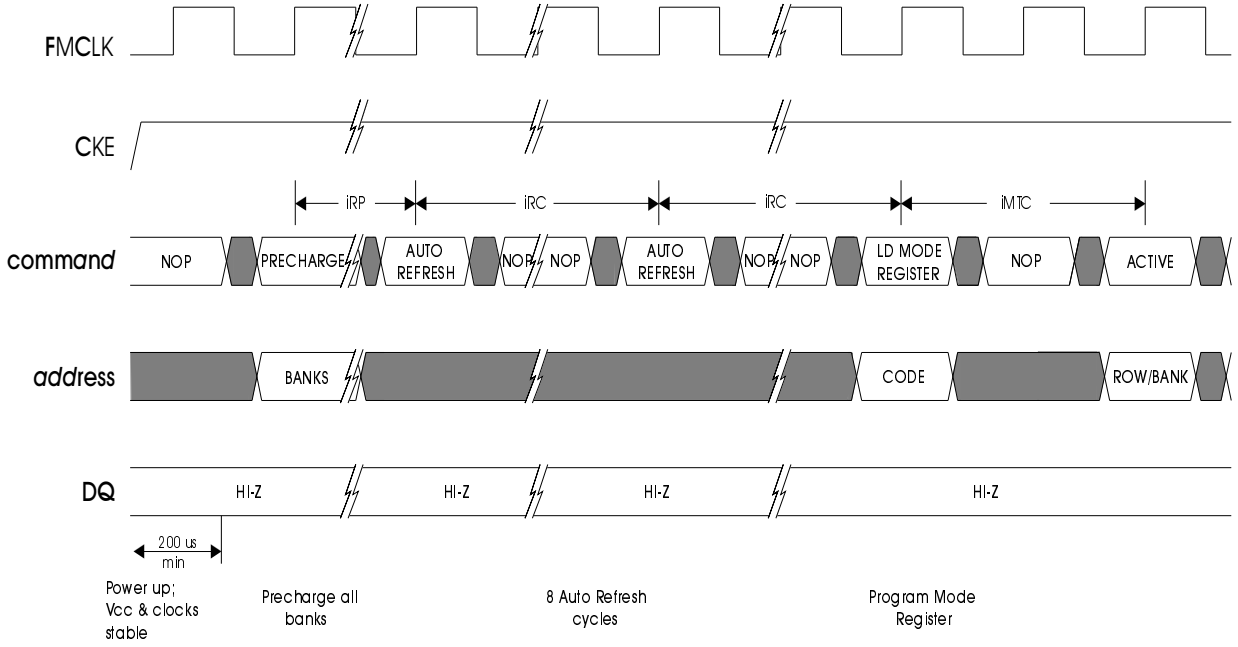


Figure A-19: Refresh Operation

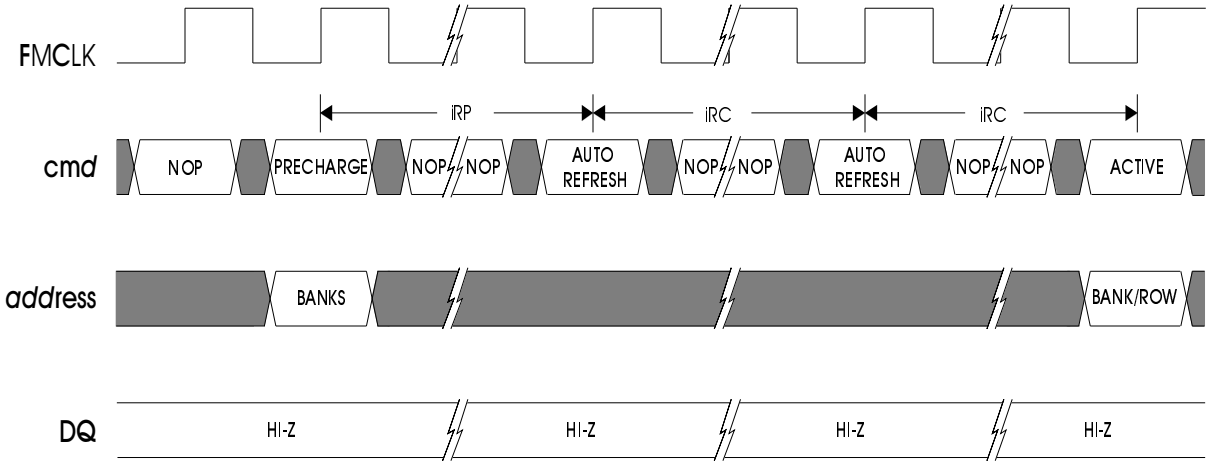


Figure A-20: Write Followed by Precharge

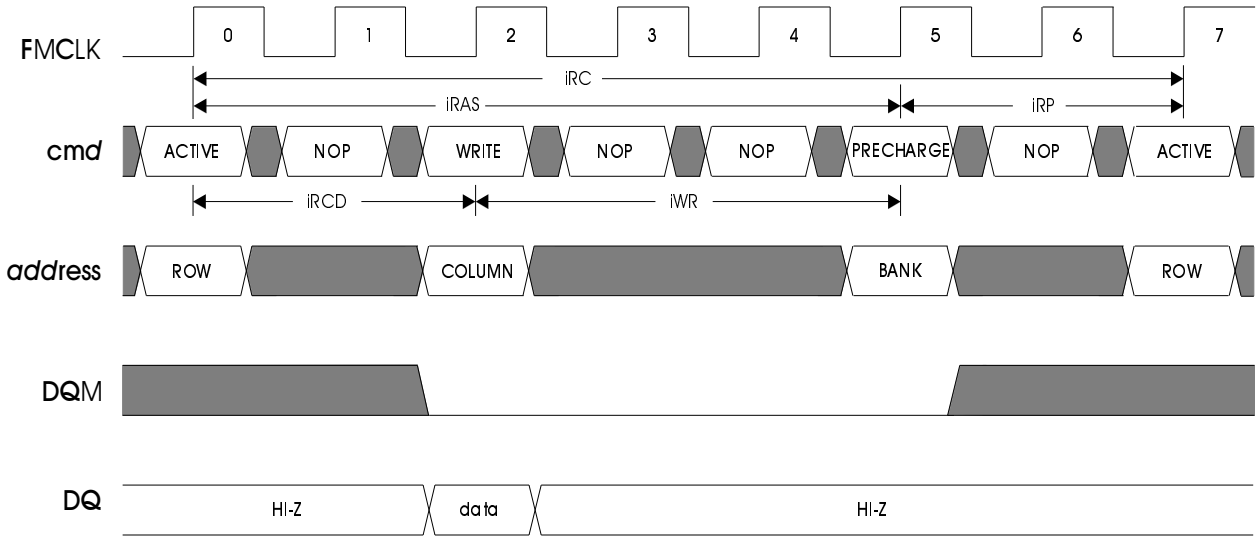


Figure A-21: Read Followed by Write (Tcl =2)

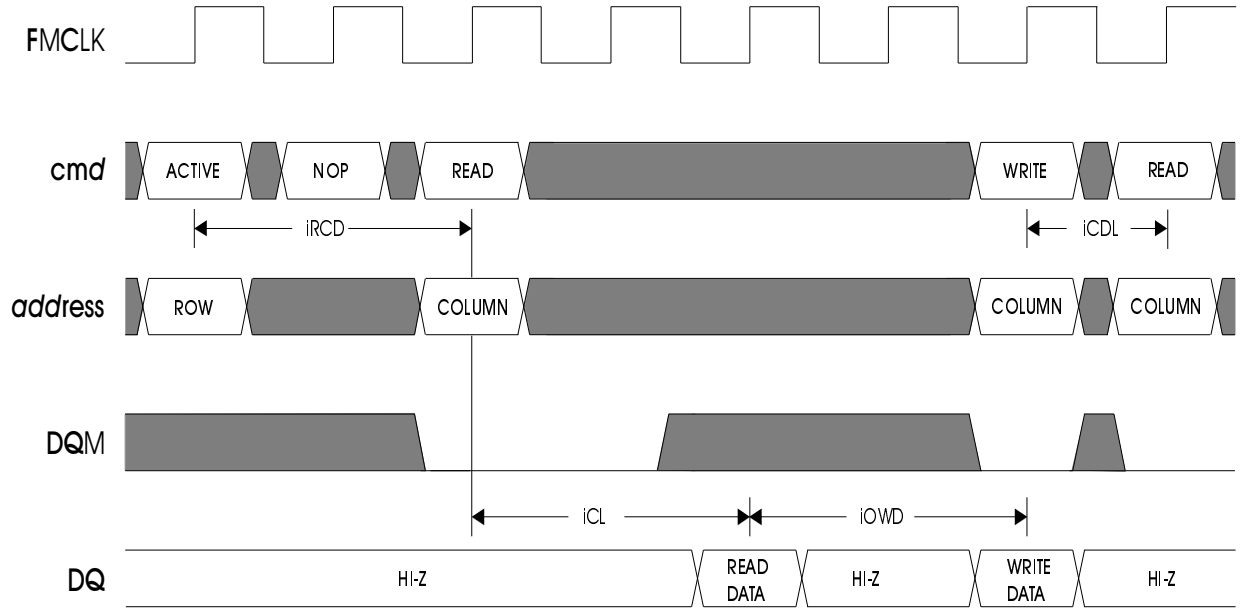


Figure A-22: Read8 Operation Without Bank Crossing (Tcl = 2)

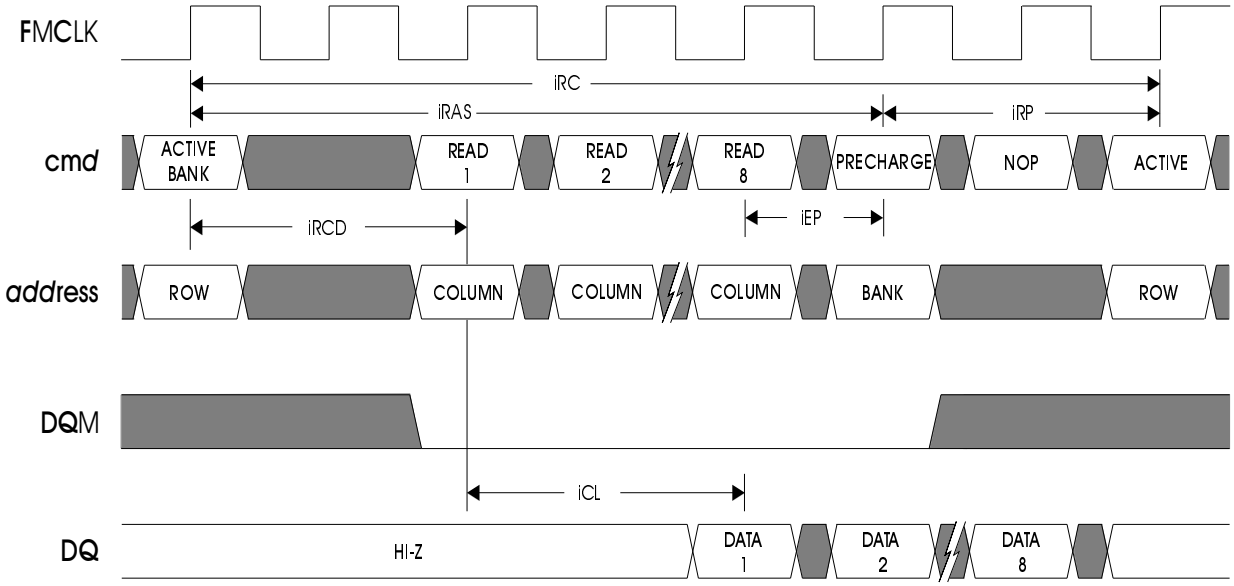


Figure A-23: Read8 Operation with Bank Crossing (Tcl = 2)

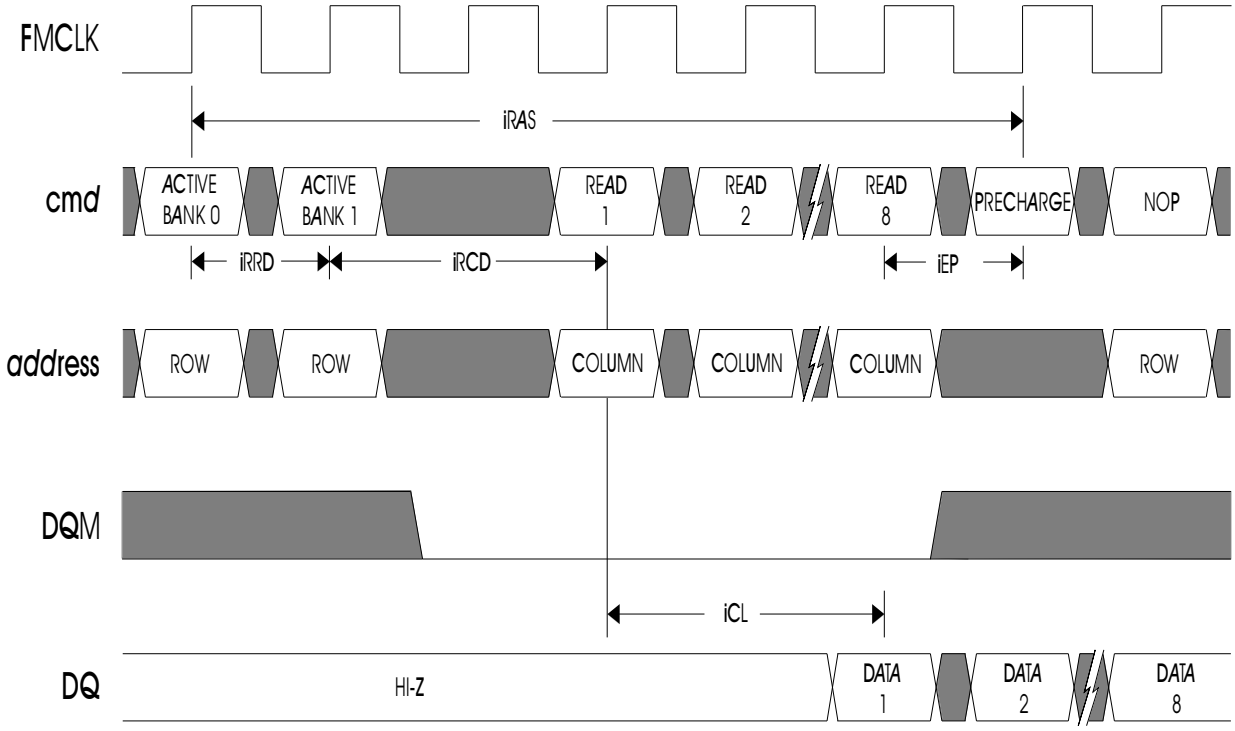


Figure A-24: Block Write and Special Mode Register Command

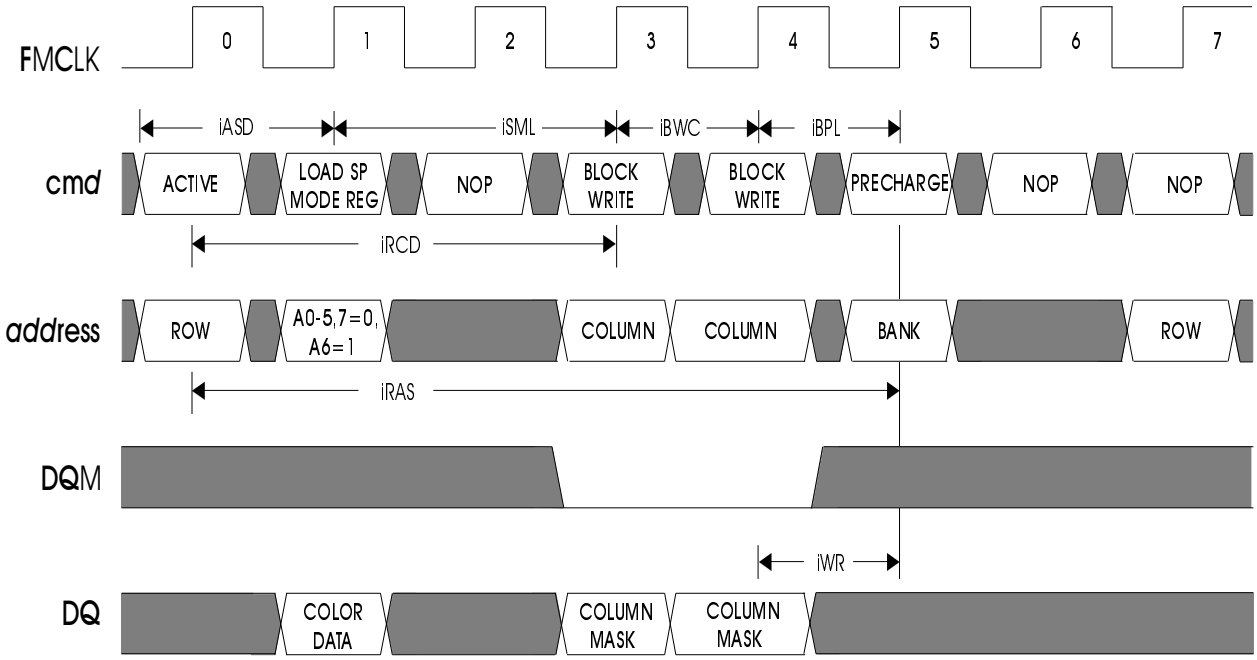


Table A-18: MGA-G100 Sync. RAM Clock-Based Parameter Table

Name	*Number of Clocks*	*Comments*	*Notes*
iCL	(2, 3, 4)	CAS latency	(1)
iRCD	(2, 3, 4)	Active command to column address command (min.)	(1)
iRAS	(3, 4, 5, 6, 7, 8)	Row active to precharge time (min.)	(1)
iRC	tras(min)+trp	Row cycle time (min.)	(1)
iRRD	(1, 2, 3, 4)	Active command to active command (other bank)	(1)
iRP	(2, 3, 4, 5)	Row precharge to row active time (min.)	(1)
iCDL	1	Write to read command time	
iWR	2	Last data in to precharge command (write recovery time)	(1)
iEP	1 or iCL-1	Read to early precharge command	(2)
iOWD	2	Last data out to write command	
ROH	1	Read data out (high-Z)	
iMTC	2	MRS to row active command	(1)
iASD	1	Active command to SMRS command	
iSML	2	SMRS to command	(1)
iBWC	(1, 2, 3)	Block write cycle time (min.)	(1)
iBPL	(1, 2, 3, 4, 5)	Block write command to precharge command	(1)

(1) Programmable parameters are based on the device rating and tCK. For a given AC parameter tXXX: iXXX = tXXX/tCK, rounded to the next largest integer.
For example: tCK=13.3ns, tRAS=84ns => iRAS = 84/13.33 = 6.32. Therefore iRAS = 7.

(2) iEP = 1 cycle aft read command if iRDD = 0, or iEP =(iCL - 1) if iRDD = 1

A.4.2.7 CODEC

Figure A-25: I33 Mode, Writes

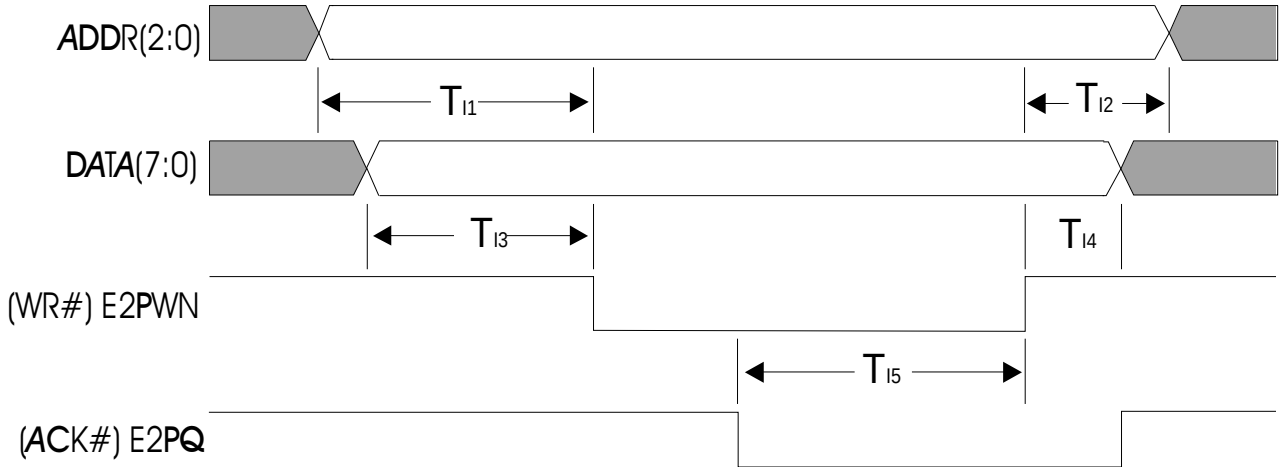


Figure A-26: I33 Mode, Reads

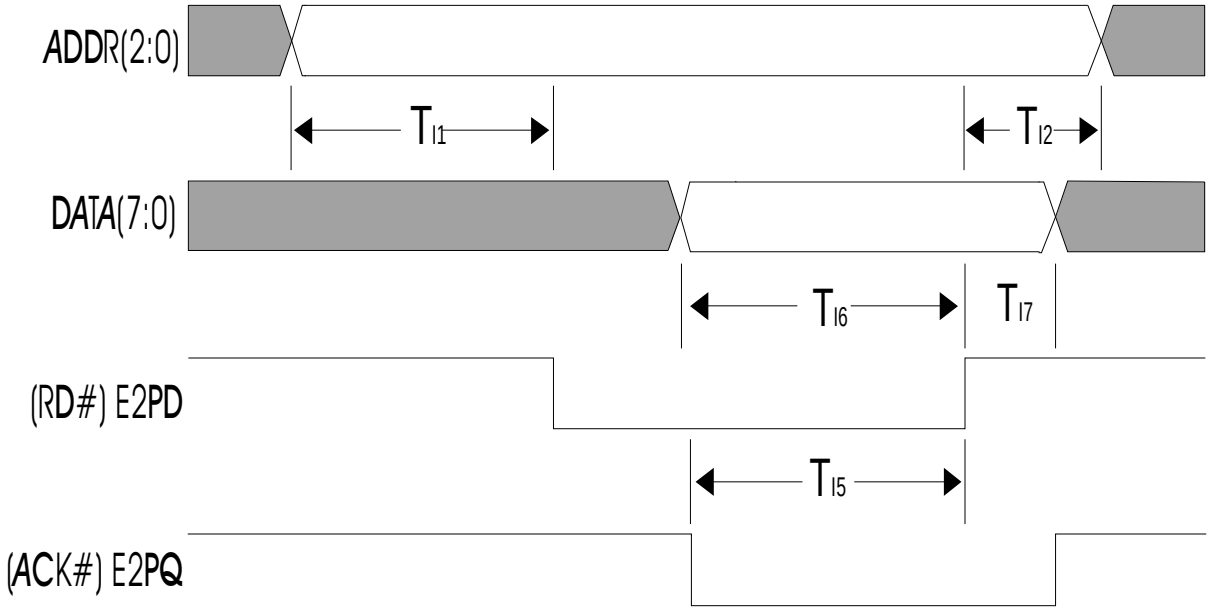


Figure A-27: VMI Mode A, Writes

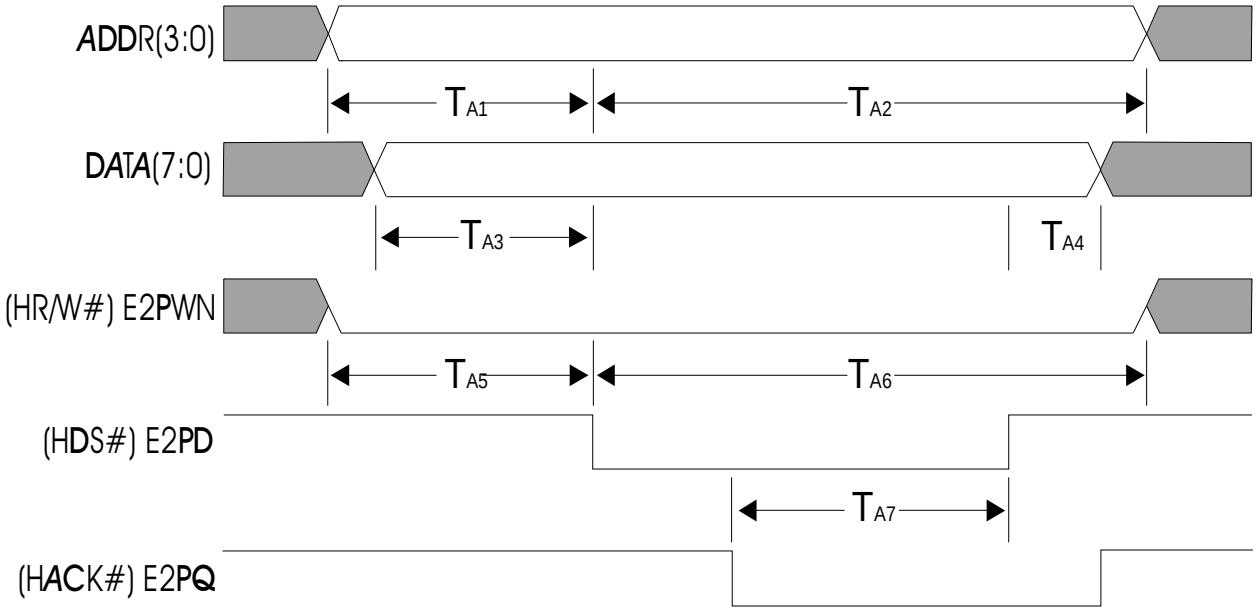


Figure A-28: VMI Mode A, Reads

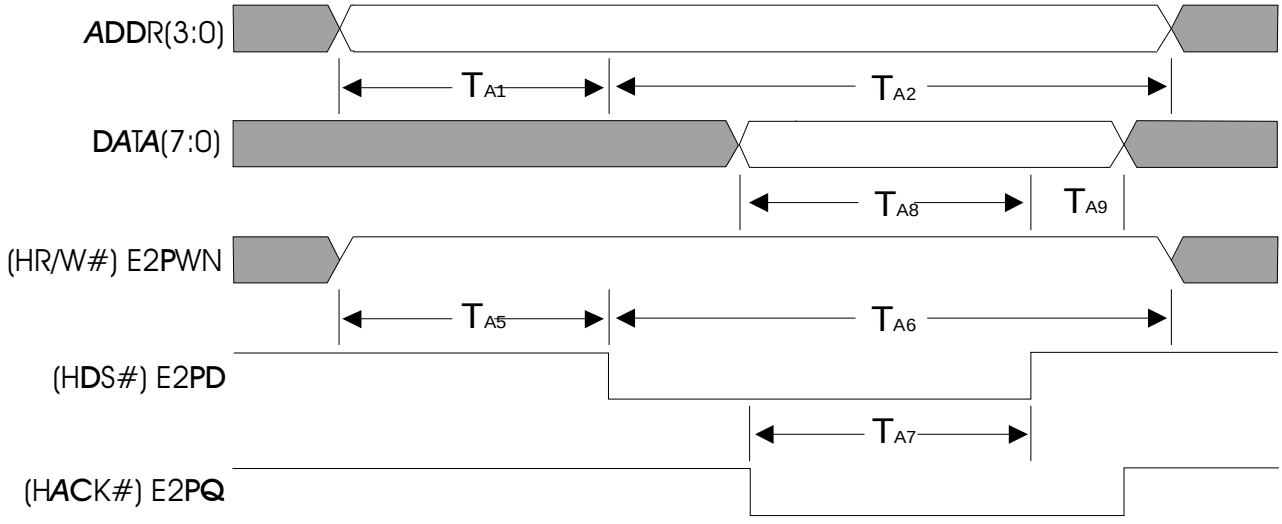


Figure A-29: VMI Mode B, Writes

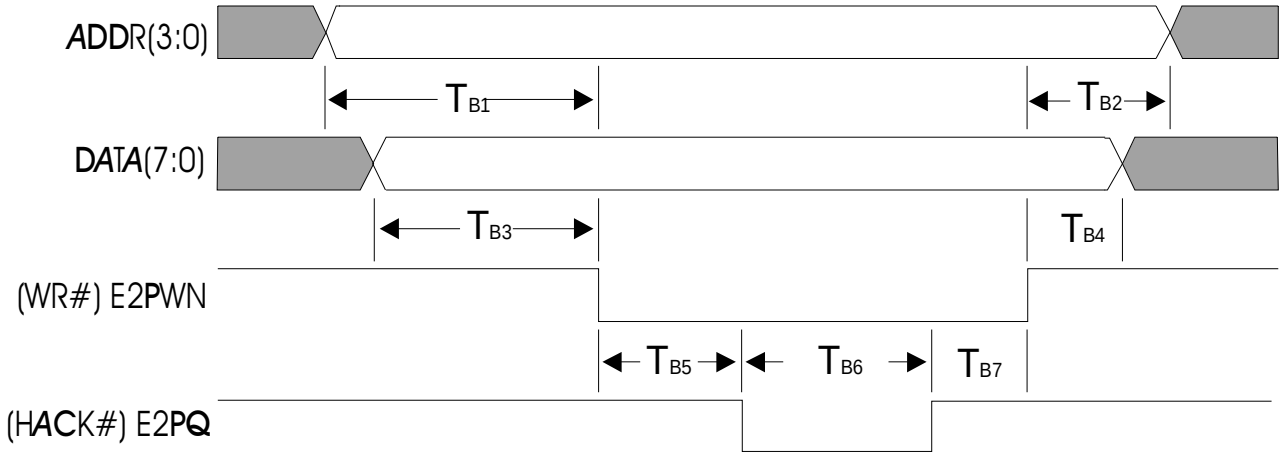


Figure A-30: VMI Mode B, Reads

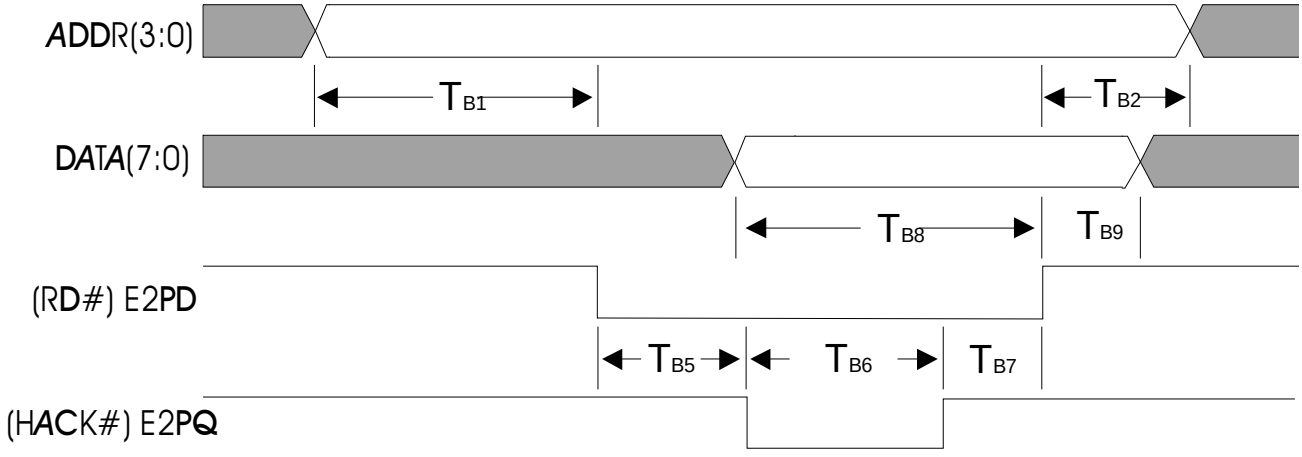
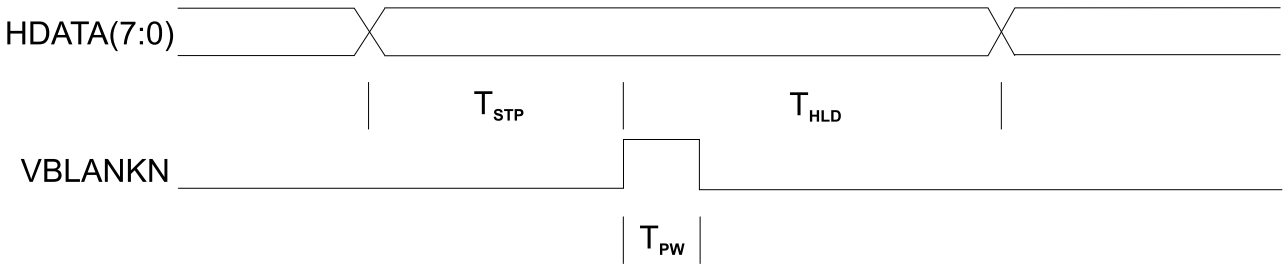


Table A-19: Codec Parameters

name	*Min (ns)*	*Max (ns)*	*Comment*
TI1	11	-	Required data setup time to rising edge of HDS# strobe
TI2	6	-	Address hold time from rising edge of RD#/WR# strobe
TI3	7	-	Data setup time to falling edge of WR# strobe
TI4	5	-	Data hold time form rising edge of WR# strobe
TI5	3+2*mclk	14+3*mclk	RD#/WR# strobe rising edge from ACK# falling edge
TI6	14	-	Required data setup time to rising RD# strobe
TI7	0	-	Required data hold time from rising RD# strobe
TA1	10	-	Address setup time to falling edge of HDS# strobe
TA2	30	-	Address hold time from rising edge of HDS# strobe
TA3	11	-	Data setup time to falling edge of HDS# strobe
TA4	6	-	Data hold time form rising edge of HDS# strobe
TA5	8	-	HR/W# setup time to falling edge of HDS# strobe
TA6	30	-	HR/W# hold time from falling edge of HDS# strobe
TA7	3+2*mclk	13+3*mclk	HDS# strobe rising edge from HACK# falling edge
TA8	15	-	Required data setup time to rising edge of HDS# strobe
TA9	0	-	Required data hold time from rising edge of HDS# strobe
TB1	11	-	Required data setup time to rising edge of HDS# strobe
TB2	6	-	Address hold time from rising edge of RD#/WR# strobe
TB3	13	-	Data setup time to falling edge of WR# strobe
TB4	15	-	Data hold time form rising edge of WR# strobe
TB5	-	20	HACK# falling edge from WR#/RD# falling edge
TB6	0	-	HACK# rising edge from HACK# falling edge
TB7	3+2*mclk	14+3*mclk	RD#/WR# strobe rising edge from ACK# rising edge
TB8	14	-	Required data setup time to rising RD# strobe
TB9	0	-	Required data hold time from rising RD# strobe

Figure A-31: Miscellaneous Register Programing Timing



	MIn	*Max*
Tstp	3*MCLK-8	-
Thld	3*MCLK-7	
Tpw	MCLK-6	MCLK+6

A.4.2.8 Video In

Figure A-32: Video In Timings

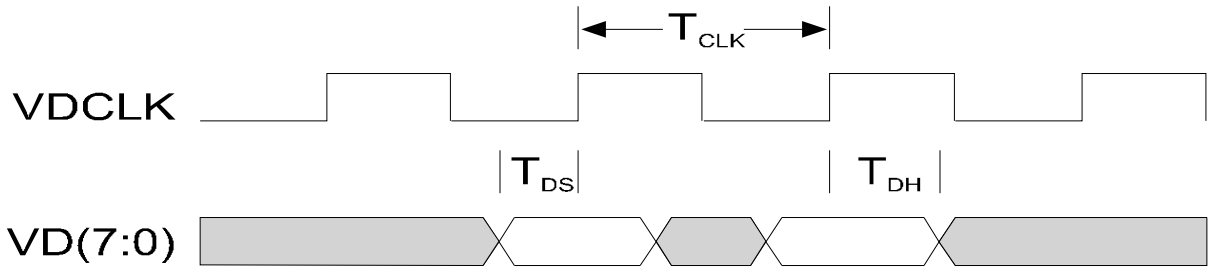


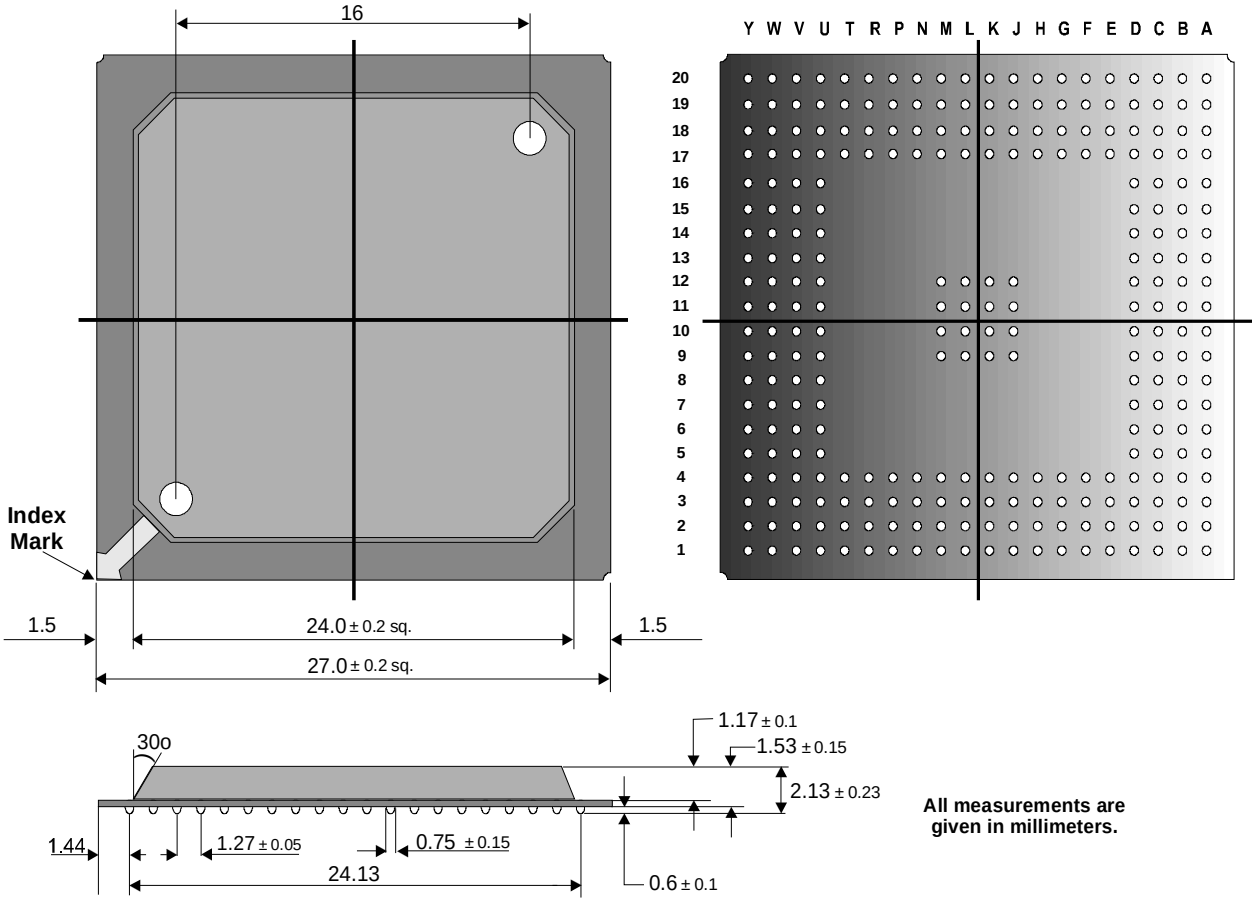
Table A-20: Video In Parametrs

Name	*min (ns)*	*Max (ns)*	*Comment*
Tclk	30	-	Input clock cycle time
Tds	4	-	Required data setup time to rising edge of VDCLK
Tdh	3	-	Required data hold time to rising edge of VDCLK

A.5 Mechanical Specification

Figure A-33: MGA-G100 Mechanical Drawing

MGA-G100 PBGA 272
Plastic Ball Grid Array



A.6 Test Features

A.6.1 NAND Tree

The MGA-G100 is equipped with a ***nand tree*** to allow the lead connections to be verified by production test equipment. The test procedure is as follows:

1. Force the TST pins to '00' to enter test mode and maintain that value during the entire test (for normal operations, the TST pins are tied to pull-ups). This will disable all output drivers except the PINTA/ pin. All pins (except PINTA/, the analog pins, RBFN_LFT, and TST<1:0>) are used as input for the nand tree operation. In test mode, PINTA/ acts as a normal driver and is used for the nand tree output (for normal operations, PINTA/ is an open drain).
2. Force all signal pins to logical '1'. PINTA/ should read '1'.
3. Next, apply a '0' to the first pin in the nand tree. The PINTA/ output should toggle to '0'.
4. Maintain the first pin at '0' and toggle the next pin to '0'. The output should toggle again.
5. Continue the shift-in of '0', following the nand tree order and monitoring the toggling of the PINTA/ pin for each new test vector.

A.6.2 AGP Nand Tree Order

Table A-21: AGP Nand Tree Order (Part 1 of 2)

Tree Order	*Ball No.*	*Pin Name*	*Tree Order*	*Ball No.*	*Pin Name*	*Tree Order*	*Ball No.*	*Pin Name*
1 (1st pin)	A19	VD<7>	31	D10	HDATA<3>	61	H3	SBA<5>
2	B18	MISC<1>	32	A9	HDATA<1>	62	H2	SBA<6>
3	B17	VD<6>	33	B9	HDATA<0>	63	H1	PAD<31>
4	C17	VD<5>	34	C9	VESYNC	64	J4	SBA<7>
5	D16	MISC<0>	35	D9	VEDCLK	65	J3	PAD<30>
6	A18	VD<4>	36	A8	VEVIDEO	66	J2	PAD<29>
7	A17	VD<3>	37	B8	E2PD	67	J1	PAD<27>
8	C16	VD<2>	38	C8	E2PQ	68	K2	PAD<25>
9	B16	VDCLK	39	A7	VBLANK/	69	K3	PAD<28>
10	A16	VD<1>	40	B7	E2PW/	70	L1	PAD<21>
11	C15	VD<0>	41	A6	DDC<3>	71	L2	PAD<23>
12	D14	VDOUT<11>	42	C7	DDC<2>	72	L3	PAD<24>
13	B15	VDOUT<10>	43	B6	DDC<1>	73	L4	PAD<26>
14	A15	VDOUT<9>	44	A5	DDC<0>	74	M1	PAD<17>
15	C14	VDOUT<8>	45	D7	MISC<2>	75	M2	PAD<19>
16	B14	VDOUT<7>	46	C6	EXTINT/	76	M3	PAD<22>
17	A14	VDOUT<6>	47	B5	VIDRST	77	M4	PCBE/<3>
18	C13	VDOUT<5>	48	A4	EXTRST/	78	N1	PDEVSEL/
19	B13	VDOUT<4>	49	D3	PGNT/	79	N2	PCBE/<2>
20	A13	VDOUT<3>	50	C1	ST<0>	80	N3	PAD<20>
21	D12	VDOUT<2>	51	D1	ST<2>	81	P1	PAD<14>
22	C12	VDOUT<1>	52	E3	PRST/	82	P2	PIRDY/
23	B12	VDOUT<0>	53	E2	ST<1>	83	R1	PAD<12>
24	A12	VOBLANK/	54	E1	SBA<0>	84	P3	PAD<16>
25	B11	VDOCLK	55	F3	SBA<2>	85	R2	PCBE/<1>
26	C11	HDATA<7>	56	G4	PREQ/	86	P4	PAD<18>
27	A11	HDATA<6>	57	F1	PCLK	87	T2	PAD<10>
28	A10	HDATA<5>	58	G3	SBA<1>	88	U1	PAD<8>
29	B10	HDATA<4>	59	G2	SBA<3>	89	T3	PTRDY/
30	C10	HDATA<2>	60	G1	SBA<4>	90	U2	PAD<7>

Table A-21: AGP Nand Tree Order (Part 2 of 2)

Tree Order	Ball No.	Pin Name	Tree Order	Ball No.	Pin Name	Tree Order	Ball No.	Pin Name
91	V1	PAD<3>	129	V12	MDQ<17>	167	N20	MDQ<33>
92	T4	PFRAME/	130	U12	MDQ<18>	168	M17	MDQ<40>
93	U3	PSTOP/	131	Y13	MDQ<19>	169	M18	MDQ<37>
94	V2	PAD<5>	132	W13	MDQ<20>	170	M19	MDQ<35>
95	W1	PCBE/<0>	133	V13	MDQ<21>	171	M20	MDQ<36>
96	W3	PAD<4>	134	Y14	MDQ<22>	172	L19	MDQ<38>
97	Y2	PAD<1>	135	W14	MDQ<23>	173	L18	MDQ<41>
98	W4	PAD<13>	136	Y15	MDQ<26>	174	L20	MDQ<39>
99	V4	PAD<9>	137	V14	MDQ<24>	175	K20	MDQ<43>
100	U5	PPAR	138	W15	MDQ<27>	176	K19	MDQ<42>
101	Y3	PAD<0>	139	Y16	MDQ<29>	177	K18	MDQ<44>
102	Y4	PAD<2>	140	U14	MDQ<25>	178	K17	MDQM<4>
103	V5	PAD<15>	141	V15	MDQ<28>	179	J20	MDQ<46>
104	W5	PAD<11>	142	W16	MDQ<30>	180	J19	MDQ<47>
105	Y5	PAD<6>	143	Y17	MA<1>	181	J18	MDQ<45>
106	V6	E2PCS/	144	V16	MDQ<31>	182	J17	MDQ<48>
107	U7	MDQ<3>	145	W17	MA<2>	183	H20	MDQM<5>
108	W6	E2PCLK	146	Y18	MA<4>	184	H19	MDQM<6>
109	Y6	MDQ<0>	147	U16	MA<0>	185	H18	MDQM<7>
110	V7	MDQ<4>	148	V17	MA<3>	186	G20	MDQ<49>
111	W7	MDQ<1>	149	W18	MA<5>	187	G19	MDQ<50>
112	Y7	MDQ<2>	150	Y19	MA<7>	188	F20	MDQ<52>
113	V8	MDQ<7>	151	W19	FMCLK2	189	G18	MDQ<51>
114	W8	MDQ<5>	152	V19	MA<9>	190	F19	MDQ<53>
115	Y8	MDQ<6>	153	U19	MA<8>	191	E20	MDQ<56>
116	U9	MDQ<10>	154	U18	MA<6>	192	G17	MDQ<55>
117	V9	MDQ<11>	155	T17	MCS/<0>	193	F18	MDQ<54>
118	W9	MDQ<8>	156	V20	MA<10>	194	E19	MDQ<57>
119	Y9	MDQ<9>	157	U20	FMCLK	195	D20	MDQ<59>
120	W10	MDQ<12>	158	T18	MWE/	196	E18	MDQ<58>
121	V10	MDQ<14>	159	T19	MCAS/	197	D19	MDQ<60>
122	Y10	MDQ<13>	160	T20	MRAS/	198	C20	MDQ<62>
123	Y11	MDQM<0>	161	R18	MCS/<1>	199	E17	MDQ<61>
124	W11	MDQM<1>	162	P17	MDSF	200	D18	MDQ<63>
125	V11	MDQ<15>	163	R19	MCS/<3>	201	C19	VVSYNC/
126	U11	MDQM<2>	164	R20	MCS/<2>	202 (last pin)	B20	VHSYNC/
127	Y12	MDQM<3>	165	N18	MDQ<34>			
128	W12	MDQ<16>	166	N19	MDQ<32>			

.

A.6.3 PCI Nand Tree Order

Table A-22: PCI Nand Tree Order (Part 1 of 2)

Tree Order	Ball No.	Pin Name	Tree Order	Ball No.	Pin Name	Tree Order	Ball No.	Pin Name
1 (1st pin)	A19	VD<7>	47	B5	VIDRST	93	V5	PAD<15>
2	B18	MISC<1>	48	A4	EXTRST/	94	W5	PAD<11>
3	B17	VD<6>	49	D3	PGNT/	95	Y5	PAD<6>
4	C17	VD<5>	50	E3	PRST/	96	V6	E2PCS/
5	D16	MISC<0>	51	G4	PREQ/	97	U7	MDQ<3>
6	A18	VD<4>	52	F1	PCLK	98	W6	E2PCLK
7	A17	VD<3>	53	H3	PIDSEL	99	Y6	MDQ<0>
8	C16	VD<2>	54	H1	PAD<31>	100	V7	MDQ<4>
9	B16	VDCLK	55	J3	PAD<30>	101	W7	MDQ<1>
10	A16	VD<1>	56	J2	PAD<29>	102	Y7	MDQ<2>
11	C15	VD<0>	57	J1	PAD<27>	103	V8	MDQ<7>
12	D14	VDOUT<11>	58	K2	PAD<25>	104	W8	MDQ<5>
13	B15	VDOUT<10>	59	K3	PAD<28>	105	Y8	MDQ<6>
14	A15	VDOUT<9>	60	L1	PAD<21>	106	U9	MDQ<10>
15	C14	VDOUT<8>	61	L2	PAD<23>	107	V9	MDQ<11>
16	B14	VDOUT<7>	62	L3	PAD<24>	108	W9	MDQ<8>
17	A14	VDOUT<6>	63	L4	PAD<26>	109	Y9	MDQ<9>
18	C13	VDOUT<5>	64	M1	PAD<17>	110	W10	MDQ<12>
19	B13	VDOUT<4>	65	M2	PAD<19>	111	V10	MDQ<14>
20	A13	VDOUT<3>	66	M3	PAD<22>	112	Y10	MDQ<13>
21	D12	VDOUT<2>	67	M4	PCBE/<3>	113	Y11	MDQM<0>
22	C12	VDOUT<1>	68	H2	PDEVSEL/	114	W11	MDQM<1>
23	B12	VDOUT<0>	69	N2	PCBE/<2>	115	V11	MDQ<15>
24	A12	VOBLANK/	70	N3	PAD<20>	116	U11	MDQM<2>
25	B11	VDOCLK	71	P1	PAD<14>	117	Y12	MDQM<3>
26	C11	HDATA<7>	72	P2	PIRDY/	118	W12	MDQ<16>
27	A11	HDATA<6>	73	R1	PAD<12>	119	V12	MDQ<17>
28	A10	HDATA<5>	74	P3	PAD<16>	120	U12	MDQ<18>
29	B10	HDATA<4>	75	R2	PCBE/<1>	121	Y13	MDQ<19>
30	C10	HDATA<2>	76	P4	PAD<18>	122	W13	MDQ<20>
31	D10	HDATA<3>	77	T2	PAD<10>	123	V13	MDQ<21>
32	A9	HDATA<1>	78	U1	PAD<8>	124	Y14	MDQ<22>
33	B9	HDATA<0>	79	T3	PTRDY/	125	W14	MDQ<23>
34	C9	VESYNC	80	U2	PAD<7>	126	Y15	MDQ<26>
35	D9	VEDCLK	81	V1	PAD<3>	127	V14	MDQ<24>
36	A8	VEVIDEO	82	T4	PFRAME/	128	W15	MDQ<27>
37	B8	E2PD	83	U3	PSTOP/	129	Y16	MDQ<29>
38	C8	E2PQ	84	V2	PAD<5>	130	U14	MDQ<25>
39	A7	VBLANK/	85	W1	PCBE/<0>	131	V15	MDQ<28>
40	B7	E2PW/	86	W3	PAD<4>	132	W16	MDQ<30>
41	A6	DDC<3>	87	Y2	PAD<1>	133	Y17	MA<1>
42	C7	DDC<2>	88	W4	PAD<13>	134	V16	MDQ<31>
43	B6	DDC<1>	89	T1	PAD<9>	135	W17	MA<2>
44	A5	DDC<0>	90	U5	PPAR	136	Y18	MA<4>
45	D7	MISC<2>	91	Y3	PAD<0>	137	U16	MA<0>
46	C6	EXTINT/	92	Y4	PAD<2>	138	V17	MA<3>

Table A-22: PCI Nand Tree Order (Part 2 of 2)

Tree Order	Ball No.	Pin Name	Tree Order	Ball No.	Pin Name	Tree Order	Ball No.	Pin Name
139	W18	MA<5>	158	M17	MDQ<40>	177	G19	MDQ<50>
140	Y19	MA<7>	159	M18	MDQ<37>	178	F20	MDQ<52>
141	W19	FMCLK2	160	M19	MDQ<35>	179	G18	MDQ<51>
142	V19	MA<9>	161	M20	MDQ<36>	180	F19	MDQ<53>
143	U19	MA<8>	162	L19	MDQ<38>	181	E20	MDQ<56>
144	U18	MA<6>	163	L18	MDQ<41>	182	G17	MDQ<55>
145	T17	MCS/<0>	164	L20	MDQ<39>	183	F18	MDQ<54>
146	V20	MA<10>	165	K20	MDQ<43>	184	E19	MDQ<57>
147	U20	FMCLK	166	K19	MDQ<42>	185	D20	MDQ<59>
148	T18	MWE/	167	K18	MDQ<44>	186	E18	MDQ<58>
149	T19	MCAS/	168	K17	MDQM<4>	187	D19	MDQ<60>
150	T20	MRAS/	169	J20	MDQ<46>	188	C20	MDQ<62>
151	R18	MCS/<1>	170	J19	MDQ<47>	189	E17	MDQ<61>
152	P17	MDSF	171	J18	MDQ<45>	190	D18	MDQ<63>
153	R19	MCS/<3>	172	J17	MDQ<48>	191	C19	VVSYNC/
154	R20	MCS/<2>	173	H20	MDQM<5>	192 (last pin)	B20	VHSYNC/
155	N18	MDQ<34>	174	H19	MDQM<6>			
156	N19	MDQ<32>	175	H18	MDQM<7>			
157	N20	MDQ<33>	176	G20	MDQ<49>			

A.6.4 Chip Test Modes

Table A-23: TST <1:0> definitions

TST <1:0>	*TEST MODE*
00	HI Z
01	Analog Test
10	IDDQ
11	Normal

Table A-24: RBFN_LFT definition

RBFN_LFT	*TEST MODE*
0	No LFT
1	LFT

NOTE: LFT means Low_Speed Functional Test. This mode minimizes output current during IOLH of buffers to prevent the simultaneous power-up that occurs during product shipment testing. The RBFN_LFT pin is ignored by the chip unless the TST<1:0> = ‘10’b when PRST/ =

A.7 Ordering Information

To receive an AGP version of the MGA-G100, order : MGA-G100A

To receive a PCI version of the MGA-G100, order : MGA-G100P.

Appendix B: Changes

This chapter includes:

B.1 Significant Changes Since Revision 0100

This section contains the revision history of the MGA-G100 Specification, from the first official release (numbered 10534-MS-0100 and dated January 20, 1998). Although every effort has been made to identify all important changes, no guarantee is offered regarding omissions or oversights.

- Global/general
  - IDUMP was removed.
  - All refrences to MGA-1264SSG were changed to MGA-G100.
- Chapter 4
  - **OPTION** register: **hardpwmsk** field was removed.
- Appendix A
  - Table A-4: Parameter list has changed.
  - Table A-8: Parameter list has changed.
  - Table A-9: Parameter list has changed.
  - Table A-19: Parameter list has changed.
  - Ordering information added at the end of Appendix A.
- Appendix B
  - Significant Changes Since revision 0100 was added.

Power Graphic Mode Register Fields (includes configuration space and memory space register fields)

Power Graphic Mode Register Fields (includes configuration space and memory space register fields)

Power Graphic Mode Register Fields (includes configuration space and memory space register fields)

Power Graphic Mode Register Fields (includes configuration space and memory space register fields)

VGA Mode Register Fields

VGA Mode Register Fields

RAMDAC and VIDEO EXPANSION Register Fields

RAMDAC and VIDEO EXPANSION Register Fields

Notes

Notes