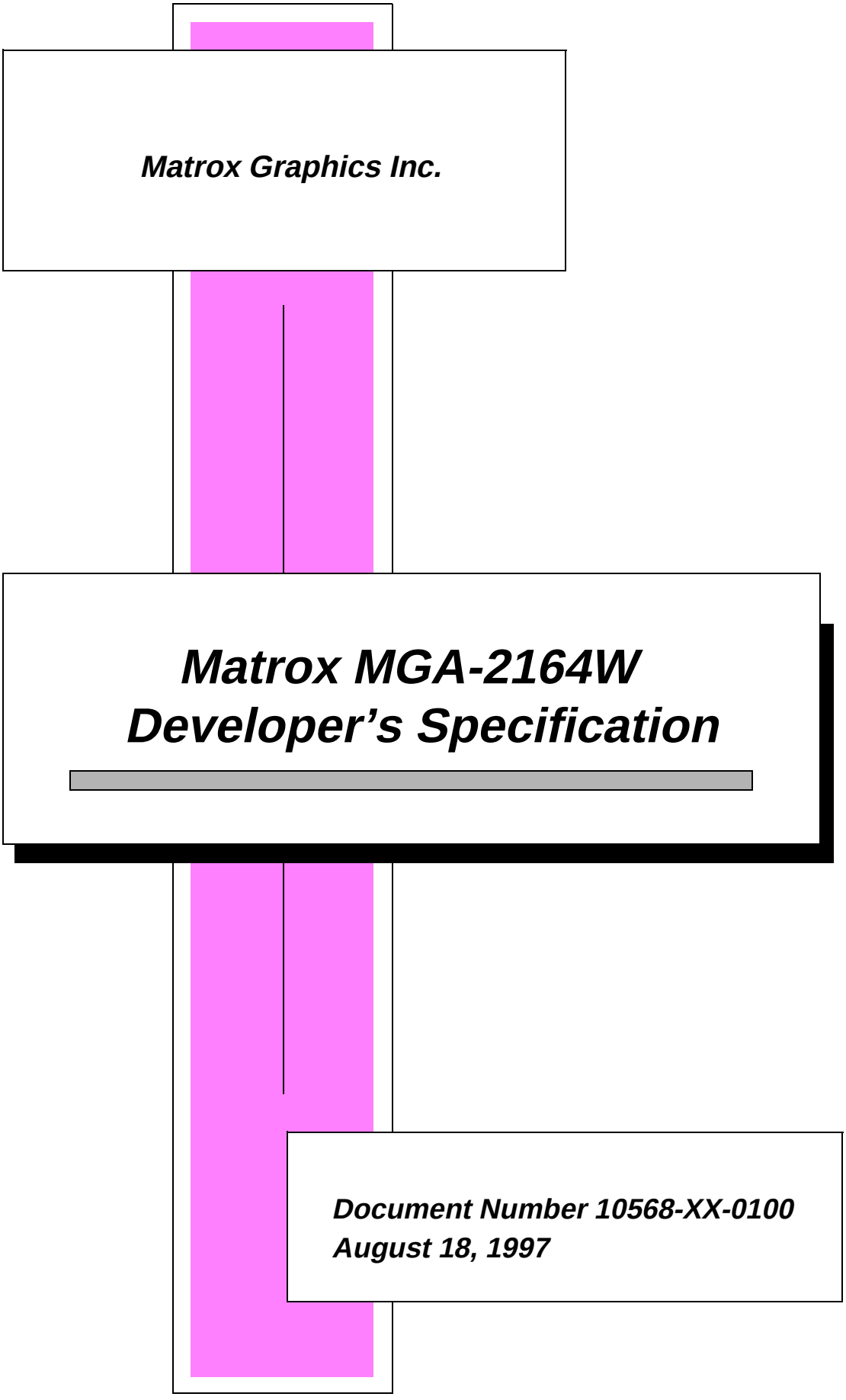

Matrox Graphics Inc.
Matrox MGA-2164W
Developer's Specification
Document Number 10568-XX-0100
August 18, 1997

Trademark Acknowledgements

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

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

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

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

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

AutoCAD® is a registered trademark of Autodesk Inc.

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

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

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

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

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

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 5: Hardware Designer’s Notes

Appendix A: Technical Information

List of Figures

Chapter 1: MGA Overview

Chapter 2: Resource Mapping

Not Applicable

Chapter 3: Register Descriptions

Not Applicable

Chapter 4: Programmer's Specification

Chapter 5: Hardware Designer's Notes

Chapter A: Technical Information

List of Tables

Chapter 1: MGA Overview

Chapter 2: Resource Mapping

Chapter 3: Register Descriptions

Not Applicable

Chapter 4: Programmer's Specification

Chapter 5: Hardware Designer's Notes

Not Applicable

Chapter A: Technical Information

Chapter 1: MGA Overview

1.1 Introduction

The Matrox MGA-2164W is the second member of the MGA-2 family of high performance 3D graphics accelerators. In one low-cost package, the MGA-2164W:

- 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

The Matrox MGA-2164W has special features specifically designed to provide superior 3D performance at a high display resolution. It is designed for GUI environments such as Windows NT, IBM OS/2 PM, Unix X-Windows, AutoCAD, and more.

The MGA-2164W series has the same acceleration core as the Matrox MGA-1064SG, but is intended for applications requiring a high display bandwidth. It controls up to 16 megabytes of dual port WRAM and supports RAMDAC up to 128 bits.

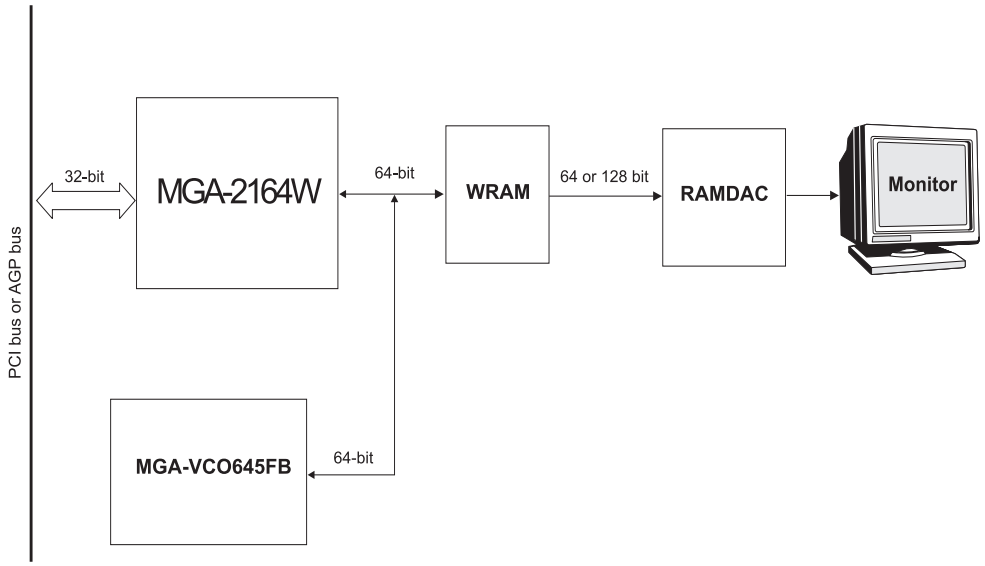
The key feature of the Matrox Fast Texture Architecture is excellent cost/performance. The Matrox texture compression model saves on memory usage, allowing low cost and high performance even within a 2 megabyte frame buffer.

The MGA-2164W 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-2164W can be upgraded with the Matrox MGA-VCO64SFB video engine to achieve a comprehensive set of high quality video capabilities.

This specification covers two chips: the MGA-2164W-PCI that connects to a PCI bus and MGA-2164W-AGP to an AGP bus. The specification applies the term MGA-2164W to both chips. For PCI specific information, the term MGA-2164W-PCI will apply, while the term MGA-2164W-AGP will apply to AGP specific information.

1.2 System Block Diagram

Figure 1-1: System Block Diagram



1.3 Application Areas

The MGA-2164W offers:

- Windows acceleration with high performance levels (ideal for mid-range system requirements). The MGA-2164W complements the MGA-2 family by delivering a strong price/performance for users who require exceptional performance at ultra high resolutions and color depths.
- Full acceleration for 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-2164W 3D engine. In addition, all other types of games will be accelerated by a combination of the MGA-2164W's DirectDraw and Direct Video engine.
- Digital video playback 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-2164W's ultra-fast 32-bit VGA core.

1.4 Typical Implementation

MGA-2164W is ideal for use in an add-in graphics card, or on the motherboard of high performance workstation.

1.4.1 Target Markets

- Professional multimedia PC markets
- Desktop publishing
- CAD
- GUI (such as WINDOWS) accelerator

1.5 Features

1.5.1 Core GUI Accelerator

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

1.5.2 Digital Video Engine

- True linear interpolation scaling filter in both X and Y
- Hardware color space conversion engine
- 8 MByte window for ILOAD and IDUMP operations
- Split frame buffer support for true graphics overlay (graphics and video are in separate sections of the frame buffer):
  - 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

1.5.3 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

1.5.4 Direct 3D Support

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

1.5.5 Memory Interface

- WRAM 256K x 32. Supports block write, aligned bitblit and write per bit for added performance
- Supports from 2 to 16 megabytes of memory in increment of 2 megabytes

1.5.6 Miscellaneous

- Host interface
  - PCI 2.1 compliant (MGA-2164W-PCI) or AGP 1.0 (MGA-2164W-AGP).
- VESA 2.0-compliant DOS applications running at a resolution of 320 x 200 can be scaled up to 640 x 480.
  - Higher resolutions are possible without changing the frame rate.
  - Filtering in the X-direction is supported.

1.6 Function Descriptions

The MGA-2164W block diagram can be broken down into 8 sections:

- Host bus interface
- VGA graphics controller
- VGA attributes controller
- CRTC
- Address Processing Unit (APU)
- Data Processing Unit (DPU)
- Memory Controller

1.6.1 Host Bus interface

This section of the MGA-2164W chip implements the interface with the host processor. It includes:

- Decoding of all resources
- Configuration registers

1.6.2 VGA Graphics Controller

This section of the MGA-2164W implements the VGA-compatible access to the frame buffer. This section includes:

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

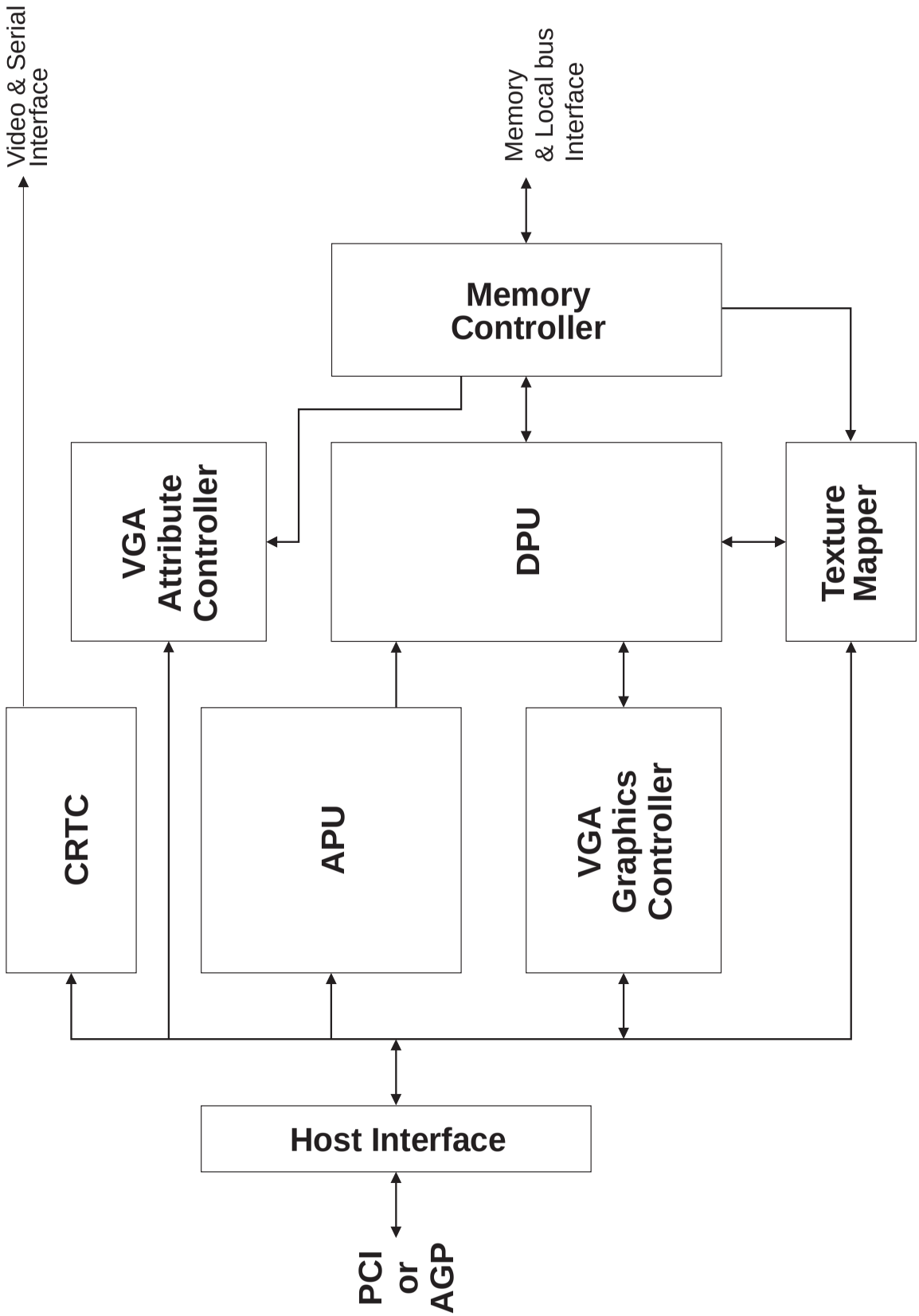
1.6.3 VGA Attributes Controller

This section implements the display refresh for standard VGA modes as well as for all character modes.

1.6.4 CRTC

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

Figure 1-2: MGA-2164W Block Diagram



1.6.5 Address Processing Unit (APU)

This section of the MGA-2164W 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

1.6.6 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

1.6.7 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 including special WRAM features as: split data transfer, single and dual color block mode, and fast blit.
- Interface to the WRAM
- Arbitration of internal requests to the frame buffer
- Depth comparison circuitry
- All control circuitry for external devices

The MGA-2164W chip can interface directly with WRAM chips. A frame buffer of up to 16 megabytes is supported.

1.7 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’. In a bulleted list within 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 3.1.1, 3.1.2, and 3.2.	
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 the **DWGCTL** illustrations (in Chapter 4), the ‘+’ and ‘#’ symbols have a special meaning. This is explained in ‘Overview’ on page 4-26.	**trans** | # | # | # | # |

1.8 Locating Information

The MGA-2164W register descriptions are located in Chapter 3. These descriptions are divided into several sections, and arranged in alphabetical order within each section.

- To find a register by name (when you know which section it’s in): go to the section and search the names at the top of each page for the register you want.
- To find a register by its index or address, refer to the tables in Chapter 2. Indirect access register indexes are duplicated on the description page of the direct access register that they refer to.
- To find a particular field within a register, search in the Alphabetical List of Register Fields at the back of the manual.

Information on how to program the MGA-2164W registers is located in Chapter 4. Hardware design information is located in Chapter 5. Appendix A contains pinout, timing, and other general information.

At the beginning of this manual you will find a complete Table of Contents, a List of (major) Figures, and a List of (major) Tables.

Chapter 2: Resource Mapping

2.1 Memory Mapping

➧ ***Note:*** All addresses and bits within dwords are labelled for a little endian processor (X86 series, for example).

2.1.1 Configuration Space Mapping

Table 2-1: MGA-2164W 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**	Location for reading the Subsystem ID. Writing has no effect.
30h-33h	**ROMBASE**	ROM Base Address
34h-37h	**CAP_PTR** (2)	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**	Location for writing the Subsystem ID. Reading will give 0's.
50h-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-2164W-AGP. For the MGA-2164W-PCI, all these locations are reserved and '0' will be returned when read.

2.1.2 MGA General Map

Table 2-2: 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**) and **romen** = 1 (see **ROMBASE**)	BIOS EPROM (1)
MGABASE1 + 0000h to **MGABASE1** + 3FFFh	MGA control aperture (see Table 2-3)	(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 how much memory has been installed. 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.

2.1.3 MGA Control Aperture

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

MGABASE1 +	*Attr.*	*Mnemonic*	*Device name*
0000h-1BFFh	W	DMAWIN (ILOAD)	7KByte Pseudo-DMA window (1)
	R	DMAWIN (IDUMP)	7KByte Pseudo-DMA window (1)
1C00h-1DFFh	W	DWGREG0	First set of 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 set of drawing registers(2)(3)(4)
2E00h-3BFFh		------	Reserved (6)
3C00h-3C1Fh	R/W	DAC	DAC(3)
3C20h-3DFFh		------	Reserved (6)
3E00h-3FFFh	R/W	EXPDEV	Expansion (7)

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

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

(3) The exact mapping within this range is dependent on the RAMDAC selected and how external connections are made.

(4) Reads of these locations are not decoded.

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

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

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

2.2 Register Mapping

Note: For the values in Table 2-4, reserved locations should not be accessed. Writing to reserved locations may affect other registers. Reading from reserved locations will return unknown data. All footnote references can be found at the end of the table.

Table 2-4: Register Map (Part 1 of 7)

Register Mnemonic Name	*Access*	*Memory Address(1)*	*I/O Address(2)*	*Index*	*Description/Comments*	*Page*
DWGCTL (3)	WO	1C00h(4)	-	-	Drawing Control	3-55
MACCESS (3)	WO	1C04h(4)	-	-	Memory Access	3-70
-	WO	1C08h(4)	-	-	Reserved	-
ZORG	WO	1C0Ch(4)	-	-	Z-Depth Origin	3-91
PAT0	WO	1C10h(4)	-	-	Pattern	3-73
PAT1	WO	1C14h(4)	-	-	Pattern	3-73
-	WO	1C18h(4)	-	-	Reserved	-
PLNWT (3)	WO	1C1Ch(4)	-	-	Plane Write Mask	3-75
BCOL	WO	1C20h(4)	-	-	Background Color / Blit Color Mask	3-30
FCOL	WO	1C24h(4)	-	-	Foreground Color / Blit Color Key	3-62
-	WO	1C28h(4)	-	-	Reserved	-
-	WO	1C2Ch(4)	-	-	Reserved (SRCBLT)	-
SRC0	WO	1C30h(4)	-	-	Source	3-80
SRC1	WO	1C34h(4)	-	-	Source	3-80
SRC2	WO	1C38h(4)	-	-	Source	3-80
SRC3	WO	1C3Ch(4)	-	-	Source	3-80
XYSTRT (5)	WO	1C40h(4)	-	-	XY Start Address	3-85
XYEND (5)	WO	1C44h(4)	-	-	XY End Address	3-84
-	1C48h-1C4Ch(4)		-	-	Reserved	-
SHIFT (5)	WO	1C50h(4)	-	-	Funnel Shifter Control	3-79
DMAPAD (5)	WO	1C54h(4)	-	-	DMA Padding	3-38
SGN (5)	WO	1C58h(4)	-	-	Sign	3-77
LEN (5)	WO	1C5Ch(4)	-	-	Length	3-69
AR0 (5)	WO	1C60h(4)	-	-	Multi-Purpose Address 0	3-23
AR1 (5)	WO	1C64h(4)	-	-	Multi-Purpose Address 1	3-24
AR2 (5)	WO	1C68h(4)	-	-	Multi-Purpose Address 2	3-25
AR3 (5)	WO	1C6Ch(4)	-	-	Multi-Purpose Address 3	3-26
AR4 (5)	WO	1C70h(4)	-	-	Multi-Purpose Address 4	3-27

Table 2-4: Register Map (Part 2 of 7)

Register Mnemonic Name	*Access*	*Memory Address(1)*	*I/O Address(2)*	*Index*	*Description/Comments*	*Page*
AR5 (5)	WO	1C74h(4)	-	-	Multi-Purpose Address 5	3-28
AR6 (5)	WO	1C78h(4)	-	-	Multi-Purpose Address 6	3-29
-	WO	1C7Ch(4)	-	-	Reserved	-
CXBNDRY (5)	WO	1C80h(4)	-	-	Clipper X Boundary	3-31
FXBNDRY (5)	WO	1C84h(4)	-	-	X Address (Boundary)	3-64
YDSTLEN (5)	WO	1C88h(4)	-	-	Y Destination and Length	3-88
PITCH (5)	WO	1C8Ch(4)	-	-	Memory Pitch	3-74
YDST (5)	WO	1C90h(4)	-	-	Y Address	3-87
YDSTORG (5)	WO	1C94h(4)	-	-	Memory Origin	3-89
YTOP (5)	WO	1C98h(4)	-	-	Clipper Y Top Boundary	3-90
YBOT (5)	WO	1C9Ch(4)	-	-	Clipper Y Bottom Boundary	3-86
CXLEFT (5)	WO	1CA0h(4)	-	-	Clipper X Minimum Boundary	3-32
CXRIGHT (5)	WO	1CA4h(4)	-	-	Clipper X Maximum Boundary	3-33
FXLEFT (5)	WO	1CA8h(4)	-	-	X Address (Left)	3-65
FXRIGHT (5)	WO	1CACh(4)	-	-	X Address (Right)	3-66
XDST (5)	WO	1CB0h(4)	-	-	X Destination Address	3-83
-	1CB4h-1CBCh(4)		-	-	Reserved	-
DR0	WO	1CC0h(4)	-	-	Data ALU 0	3-42
-	WO	1CC4h(4)	-	-	Reserved (DR1)	-
DR2	WO	1CC8h(4)	-	-	Data ALU 2	3-43
DR3	WO	1CCCh(4)	-	-	Data ALU 3	3-44
DR4	WO	1CD0h(4)	-	-	Data ALU 4	3-45
-	WO	1CD4h(4)	-	-	Reserved (DR5)	-
DR6	WO	1CD8h(4)	-	-	Data ALU 6	3-46
DR7	WO	1CDCh(4)	-	-	Data ALU 7	3-47
DR8	WO	1CE0h(4)	-	-	Data ALU 8	3-48
	WO	1CE4h(4)	-	-	Reserved (DR9)	-
DR10	WO	1CE8h(4)	-	-	Data ALU 10	3-49
DR11	WO	1CECh(4)	-	-	Data ALU 11	3-50
DR12	WO	1CF0h(4)	-	-	Data ALU 12	3-51
-	WO	1CF4h(4)	-	-	Reserved (DR13)	-

Table 2-4: Register Map (Part 3 of 7)

Register Mnemonic Name	*Access*	*Memory Address(1)*	*I/O Address(2)*	*Index*	*Description/Comments*	*Page*
DR14	WO	1CF8h(4)	-	-	Data ALU 14	3-52
DR15	WO	1CFCh(4)	-	-	Data ALU 15	3-53
-	1D00h-1DFFh(4)		-	-	Same mapping as 1C00h-1CFCh (6)	-
-	1E00h - 1E0Fh		-	-	Reserved	-
FIFOSTATUS	RO	1E10h	-	-	Bus FIFO Status	3-63
STATUS	RO	1E14h	-	-	Status	3-81
ICLEAR	WO	1E18h	-	-	Interrupt Clear	3-67
IEN	R/W	1E1Ch	-	-	Interrupt Enable	3-68
VCOUNT	RO	1E20h	-	-	Vertical Count	3-82
-	1E24h - 1E2Fh		-	-	Reserved	-
DMAMAP30	R/W	1E30h	-	-	DMA Map 3h to 0h	3-34
DMAMAP74	R/W	1E34h	-	-	DMA Map 7h to 4h	3-35
DMAMAPB8	R/W	1E38h	-	-	DMA Map Bh to 8h	3-36
DMAMAPFC	R/W	1E3Ch	-	-	DMA Map Fh to Ch	3-37
RST	R/W	1E40h	-	-	Reset	3-76
-	1E44h - 1E53h		-	-	Reserved	-
OPMODE	R/W	1E54h	-	-	Operating Mode	3-71
-	1E60h - 1E7Fh		-	-	Reserved	-
DWG_INDIR_WT<0>	WO	1E80h(4)	-	-	Drawing Register Indirect Write 0	3-54
DWG_INDIR_WT<1>	WO	1E84h(4)	-	-	Drawing Register Indirect Write 1	3-54
DWG_INDIR_WT<2>	WO	1E88h(4)	-	-	Drawing Register Indirect Write 2	3-54
DWG_INDIR_WT<3>	WO	1E8Ch(4)	-	-	Drawing Register Indirect Write 3	3-54
DWG_INDIR_WT<4>	WO	1E90h(4)	-	-	Drawing Register Indirect Write 4	3-54
DWG_INDIR_WT<5>	WO	1E94h(4)	-	-	Drawing Register Indirect Write 5	3-54
DWG_INDIR_WT<6>	WO	1E98h(4)	-	-	Drawing Register Indirect Write 6	3-54
DWG_INDIR_WT<7>	WO	1E9Ch(4)	-	-	Drawing Register Indirect Write 7	3-54
DWG_INDIR_WT<8>	WO	1EA0h(4)	-	-	Drawing Register Indirect Write 8	3-54
DWG_INDIR_WT<9>	WO	1EA4h(4)	-	-	Drawing Register Indirect Write 9	3-54
DWG_INDIR_WT<10>	WO	1EA8h(4)	-	-	Drawing Register Indirect Write 10	3-54
DWG_INDIR_WT<11>	WO	1EACh(4)	-	-	Drawing Register Indirect Write 11	3-54
DWG_INDIR_WT<12>	WO	1EB0h(4)	-	-	Drawing Register Indirect Write 12	3-54
DWG_INDIR_WT<13>	WO	1EB4h(4)	-	-	Drawing Register Indirect Write 13	3-54
DWG_INDIR_WT<14>	WO	1EB8h(4)	-	-	Drawing Register Indirect Write 14	3-54

Table 2-4: Register Map (Part 4 of 7)

Register Mnemonic Name	*Access*	*Memory Address(1)*	*I/O Address(2)*	*Index*	*Description/Comments*	*Page*
DWG_INDIR_WT<15>	WO	1EBCh(4)	-	-	Drawing Register Indirect Write 15	3-54
-	1EC0h - 1FBFh		-	-	Reserved	-
ATTR (Index)	R/W	1FC0h	3C0h	-	Attribute Controller	3-94
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	3-96
ATTR1	R/W	-	-	01h	Palette entry 1	3-96
ATTR2	R/W	-	-	02h	Palette entry 2	3-96
ATTR3	R/W	-	-	03h	Palette entry 3	3-96
ATTR4	R/W	-	-	04h	Palette entry 4	3-96
ATTR5	R/W	-	-	05h	Palette entry 5	3-96
ATTR6	R/W	-	-	06h	Palette entry 6	3-96
ATTR7	R/W	-	-	07h	Palette entry 7	3-96
ATTR8	R/W	-	-	08h	Palette entry 8	3-96
ATTR9	R/W	-	-	09h	Palette entry 9	3-96
ATTRA	R/W	-	-	0Ah	Palette entry A	3-96
ATTRB	R/W	-	-	0Bh	Palette entry B	3-96
ATTRC	R/W	-	-	0Ch	Palette entry C	3-96
ATTRD	R/W	-	-	0Dh	Palette entry D	3-96
ATTRE	R/W	-	-	0Eh	Palette entry E	3-96
ATTRF	R/W	-	-	0Fh	Palette entry F	3-96
ATTR10	R/W	-	-	10h	Attribute Mode Control	3-97
ATTR11	R/W	-	-	11h	Overscan Color	3-99
ATTR12	R/W	-	-	12h	Color Plane Enable	3-100
ATTR13	R/W	-	-	13h	Horizontal Pel Panning	3-101
ATTR14	R/W	-	-	14h	Color Select	3-102
-	-	-	-	15h - 1Fh: Reserved		-
INSTS0	RO	1FC2h	3C2h	-	Input Status 0	3-158
MISC	WO	1FC2h	3C2h	-	Miscellaneous Output	3-160
-	R/W	1FC3h	3C3h(7)	-	Reserved, not decoded for I/O	
SEQ (Index)	R/W	1FC4h	3C4h	-	Sequencer	3-162
SEQ (Data)	R/W	1FC5h	3C5h	-	Sequencer	-
SEQ0	R/W	-	-	00h	Reset	3-163
SEQ1	R/W	-	-	01h	Clocking Mode	3-164
SEQ2	R/W	-	-	02h	Map Mask	3--165

Table 2-4: Register Map (Part 5 of 7)

Register Mnemonic Name	*Access*	*Memory Address(1)*	*I/O Address(2)*	*Index*	*Description/Comments*	*Page*
SEQ3	R/W	-	-	03h	Character Map Select	3-166
SEQ4	R/W	-	-	04h	Memory Mode	3-167
-	R/W	-	-	05h - 07h: Reserved		-
-	-	1FC6h	-	-	Reserved	-
DACSTAT	RO	1FC7h	3C7h	-	DAC Status(requires a byte access)	3-145
-	WO	1FC7h	-	-	Reserved	-
-	1FC8h-1FC9h		-	-	Reserved	-
FEAT	RO	1FCAh	3CAh	-	Feature Control	3-146
	WO	1FCAh	3CAh	-	Reserved	-
-	-	1FCBh	3CBh(7)	-	Reserved, not decoded for I/O	-
MISC	RO	1FCCh	3CCh	-	Miscellaneous Output	3-160
-	WO	1FCCh	3CCh	-	Reserved	-
-	-	1FCDh	3CDh(7)	-	Reserved, not decoded for I/O	-
GCTL (Index)	R/W	1FCEh	3CEh	-	Graphics Controller	3-147
GCTL (Data)	R/W	1FCFh	3CFh	-	Graphics Controller	-
GCTL0	R/W	-	-	00h	Set/Reset	3-148
GCTL1	R/W	-	-	01h	Enable Set/Reset	3-149
GCTL2	R/W	-	-	02h	Color Compare	3-150
GCTL3	R/W	-	-	03h	Data Rotate	3-151
GCTL4	R/W	-	-	04h	Read Map Select	3-152
GCTL5	R/W	-	-	05h	Graphics Mode	3-153
GCTL6	R/W	-	-	06h	Miscellaneous	3-155
GCTL7	R/W	-	-	07h	Color Don’t Care	3-156
GCTL8	R/W	-	-	08h	Bit Mask	3-157
-	-	-	-	09h - 0Fh: Reserved		-
-	1FD0h-1FD3h		-	-	Reserved	-
CRTC (Index)	R/W	1FD4h	3D4h	-	CRTC Registers (or 3B4h(8))	3-104
CRTC (Data)	R/W	1FD5h	3D5h	-	CRTC Registers (or 3B5h(8))	-
CRTC0	R/W	-	-	00h	Horizontal Total	3-106
CRTC1	R/W	-	-	01h	Horizontal Display Enable End	3-107
CRTC2	R/W	-	-	02h	Start Horizontal Blanking	3-108
CRTC3	R/W	-	-	03h	End Horizontal Blanking	3-109
CRTC4	R/W	-	-	04h	Start Horizontal Retrace Pulse	3-110
CRTC5	R/W	-	-	05h	End Horizontal Retrace	3-111
CRTC6	R/W	-	-	06h	Vertical Total	3-112
CRTC7	R/W	-	-	07h	Overflow	3-113

Table 2-4: Register Map (Part 6 of 7)

Register Mnemonic Name	*Access*	*Memory Address(1)*	*I/O Address(2)*	*Index*	*Description/Comments*	*Page*
CRTC8	R/W	-	-	08h	Preset Row Scan	3-114
CRTC9	R/W	-	-	09h	Maximum Scan Line	3-115
CRTCA	R/W	-	-	0Ah	Cursor Start	3-116
CRTCB	R/W	-	-	0Bh	Cursor End	3-117
CRTCC	R/W	-	-	0Ch	Start Address High	3-118
CRTCD	R/W	-	-	0Dh	Start Address Low	3-119
CRTCE	R/W	-	-	0Eh	Cursor Location High	3-120
CRTCF	R/W	-	-	0Fh	Cursor Location Low	3-121
CRTC10	R/W	-	-	10h	Vertical Retrace Start	3-122
CRTC11	R/W	-	-	11h	Vertical Retrace End	3-123
CRTC12	R/W	-	-	12h	Vertical Display Enable End	3-124
CRTC13	R/W	-	-	13h	Offset	3-125
CRTC14	R/W	-	-	14h	Underline Location	3-126
CRTC15	R/W	-	-	15h	Start Vertical Blank	3-127
CRTC16	R/W	-	-	16h	End Vertical Blank	3-128
CRTC17	R/W	-	-	17h	CRTC Mode Control	3-129
CRTC18	R/W	-	-	18h	Line Compare	3-133
-	-	-	-	19h - 21h: Reserved		-
CRTC22	R/W	-	-	22h	CPU Read Latch	3-134
-	-	-	-	23h	Reserved	-
CRTC24	R/W	-	-	24h	Attributes Address/Data Select	3-135
-	-	-	-	25h	Reserved	-
CRTC26	R/W	-	-	26h	Attributes Address	3-136
-	-	-	-	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))	3-159
FEAT	WO	1FDAh	3DAh	-	Feature Control (or 3BAh(8))	3-146
-	-	1FDBh	3DBh(7)	-	Reserved, not decoded for I/O (or 3BBh(8))	-
-	1FDCh-1FDDh		-	-	Reserved	-
CRTCEXT (Index)	R/W	1FDEh	3DEh	-	CRTC Extension	3-137
CRTCEXT (Data)	R/W	1FDFh	3DFh	-	CRTC Extension	-
CRTCEXT0	R/W	-	-	00h	Address Generator Extensions	3-138

Table 2-4: Register Map (Part 7 of 7)

Register Mnemonic Name	*Access*	*Memory Address*[1]	*I/O Address*[2]	*Index*	*Description/Comments*	*Page*
CRTCEXT1	R/W	-	-	01h	Horizontal Counter Extensions	3-139
CRTCEXT2	R/W	-	-	02h	Vertical Counter Extensions	3-140
CRTCEXT3	R/W	-	-	03h	Miscellaneous	3-141
CRTCEXT4	R/W	-	-	04h	Memory Page	3-143
CRTCEXT5	R/W	-	-	05h	Horizontal Video Half Count	3-144
-	1FE0h - 1FFEh		-	-	Reserved	-
CACHEFLUSH	R/W	1FFFh	-	-	Cache Flush	3-103
-	2C38h-2C4Ch[4]		-	-	Reserved	-
DR0_Z32 LSB	WO	2C50h[4]	-	-	Extended Data ALU 0	3-39
DR0_Z32 MSB	WO	2C54h[4]	-	-	Extended Data ALU 0	3-39
-	-	2C58h[4]	-	-	Reserved	
-	-	2C5Ch[4]	-	-	Reserved	
DR2_Z32 LSB	WO	2C60h[4]	-	-	Extended Data ALU 2	3-40
DR2_Z32 MSB	WO	2C64h[4]	-	-	Extended Data ALU 2	3-40
DR3_Z32 LSB	WO	2C68h[4]	-	-	Extended Data ALU 3	3-41
DR3_Z32 MSB	WO	2C6Ch[4]	-	-	Extended Data ALU 3	3-41

[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] Under some conditions, writing to these registers may create a stall in the graphic pipe because their contents are retained as long as the memory controller needs them for the current operation. 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 finishes its processing before the data processor, we recommend that you initialize these registers first, in order to take advantage of the instruction overlay capability of the address processor.

[6] Accessing a register in this range instructs the drawing engine to start a drawing operation.

[7] Word or dword accesses to these specific reserved locations will be decoded. (The PCI convention states that I/O space should only be accessed in bytes, and that a bridge will not perform byte packing.)

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

Chapter 3: Register Descriptions

Note: All the register descriptions within this chapter are arranged in alphabetical order by mnemonic name. For more information on finding registers see 'Locating Information' on page 1-9.

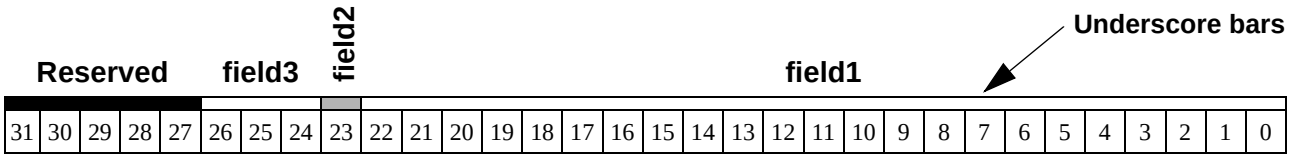
3.1 Power Graphic Mode Register Descriptions

3.1.1 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. Reserved fields are identified by black underscore bars; all other fields display alternating white and gray bars.

Sample Power Graphic Mode Config. Space Register	**SAMPLE_CS**
Address	<value> (CS)
Attributes	R/W
Reset Value	<value>

Main header



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 2.

➾ ***Note:*** CS indicates that the address lies within the configuration space.

Attributes

The Power Graphic Mode configuration space register attributes are:

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

Reset Value

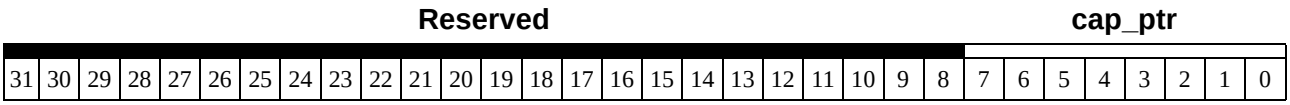
Here are some of the symbols that appear as part of a register's reset value:

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

Address 34h (CS) for MGA-2164W-AGP only

Attributes RO, BYTE/WORD/DWORD, STATIC

Reset Value `0000 0000 0000 0000 0000 0000 1111 0000`b



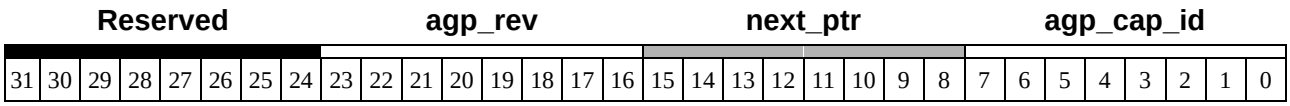
cap_ptr RO<7:0> This field contains the hard-coded offset byte ‘F0h’ within the device configuration space of the AGP Capability Identifier register.

Reserved <31:8> Reserved. Writing has no effect. Reading will give ‘0’s.

Address F0h (CS) for MGA-2164W-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: 02, 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 there is no other capabilities in the list.

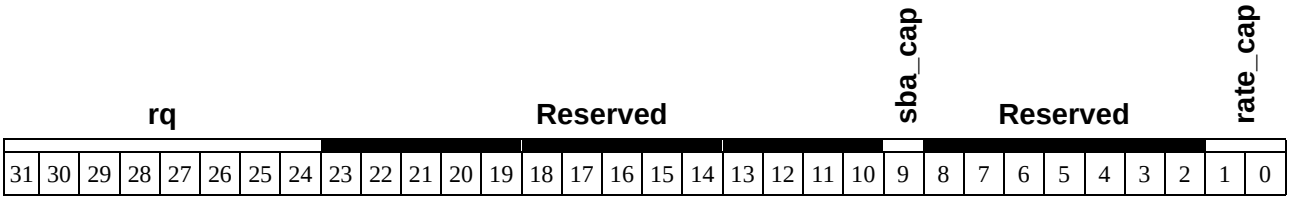
agp_rev <23:16> This field contains the AGP specification revision to which this device complies: 10h (as in 1.0)

Reserved <31:24> Reserved. Writing has no effect. Reading will give ‘0’s.

Address F4h (CS) for MGA-2164W-AGP only

Attributes RO, BYTE/WORD/DWORD, STATIC

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



rate_cap <1:0> The hard-coded ‘00b’ indicates that the device does not support any AGP transfer rate mode.

sba_cap <9> The hard-coded ‘0’ indicates that the device does not support AGP Side band addressing.

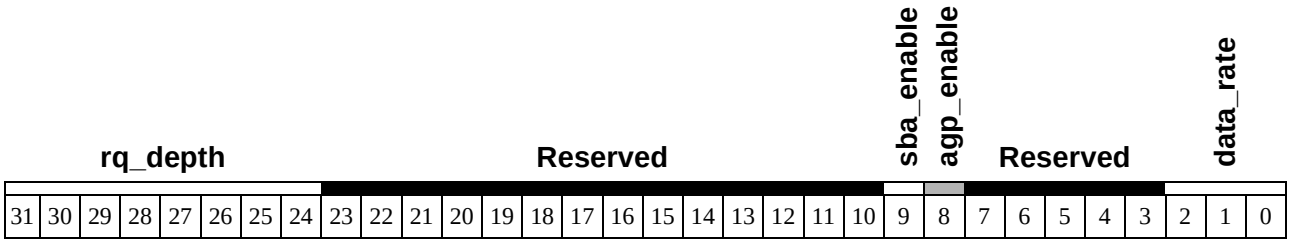
rq <31:24> The hard-coded ‘00h’ indicates that the device does not have an AGP Request Queue.

Reserved <23:10> <8:2> Reserved. Writing has no effect. Reading will give ‘0’s.

Address F8h (CS) (applies to MGA-2164W-AGP only)

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

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
- 010: 2x data rate
- 1xx: reserved
- x11: 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 of the device is enabled.

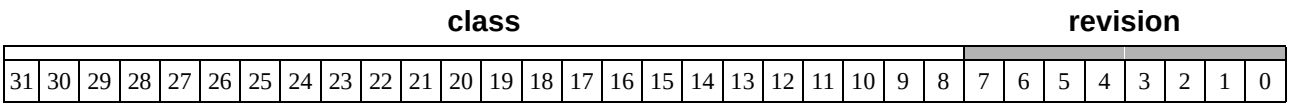
rq_depth <31:24> This should be programmed with the maximum number of pipelined operations that the master (this device) is allowed to queue. This value should be equal, or less, than the value reported in the target rq field of AGP_STS register.

Reserved <23:10> <7:3> Reserved. Writing has no effect. 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.

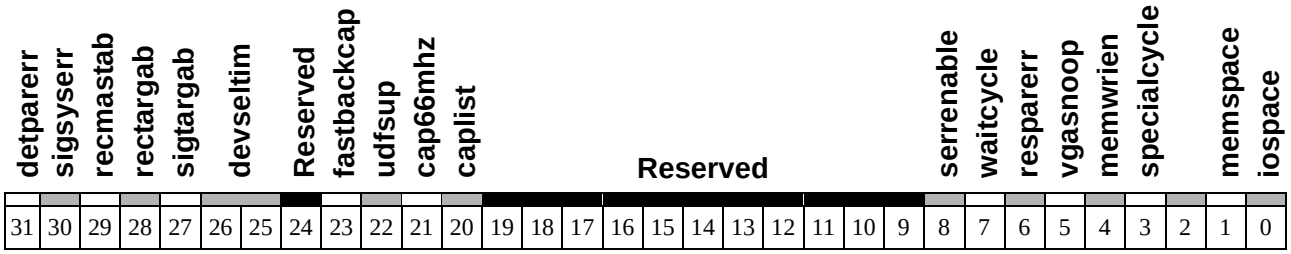
vgaboot strap	*Value*	*Meaning*
'0'	038000h	Non-Super VGA display controller
'1'	030000h	Super VGA compatible controller

The sampled state of the vgaboot strap (pin MDQ<5>, described on page A-4) can be read through this register.

Address 04h (CS)

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

Reset Value 0000 0010 10?? 0000 0000 0000 0000 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

•

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 3-18) 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>

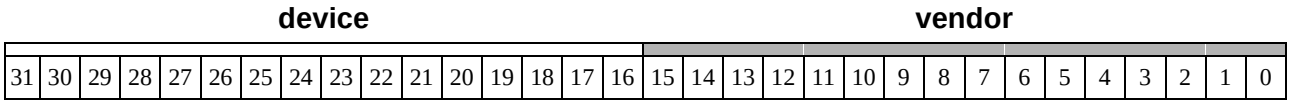
This bit reads as ‘0’, indicating that no address/data stepping is performed for read accesses in the target (data stepping) and the master (address stepping). Writing has no effect.

serrenable RO <8>	This hard-coded ‘0’ indicates that MGA does not generate SERR interrupts. Writing has no effect.
caplist RO <20>	The hard-coded ‘0’ for MGA-2164W-PCI indicates that the configuration space does not contain a capability list. The hard-coded ‘1’ for MGA-2164W-AGP indicates that the device has a capability list in the configuration space. The list is located at the offset in the CAP_PTR register.Writing has no effect.
cap66mhz RO <21>	This bit is forced to ‘0’ in the MGA-2164-W-PCI to indicate that the bus is running at 33 MHz.This bit is forced to ‘1’ in the MGA-2164W-AGP to indicate that the bus is running at 66MHz. Writing has no effect.
udfsup RO <22>	The hard-coded ‘0’ indicates that the MGA does not support user-definable features.
fastbackcap RO <23>	The hard-coded ‘1’ indicates that the MGA supports fast back-to-back transactions when part of the transaction targets a different agent. Writing has no effect.
devseltim RO <26:25>	Device select timing. Specifies the timing of devsel. It is read as ‘01’.
sigtargab R/W <27>	Signaled target abort. Set to ‘1’ when the MGA terminates a transaction in target mode with target-abort. This bit is cleared to ‘0’ when written with ‘1’.
rectargab R/W <28>	Received target abort. Set to ‘1’ when the MGA is a master and a transaction is terminated with target-abort. This bit is cleared to ‘0’ when written with ‘1’.
recmastab R/W <29>	Received master abort. Set to ‘1’ when a transaction is terminated with master-abort by the MGA. This bit is cleared to ‘0’ when written with ‘1’.
sigsyserr RO <30>	MGA does not assert SERR/. Writing has no effect. Reading will give ‘0’s.
detparerr RO <31>	MGA does not detect parity errors. Writing has no effect. Reading will give ‘0’s.
Reserved:	**<20:9> <24>** Reserved. Writing has no effect. Reading will give ‘0’s.

Address 00h (CS)

Attributes RO, BYTE/WORD/DWORD, STATIC

Reset Value `0000 0101 0001 1?11 0001 0000 0010 1011`b



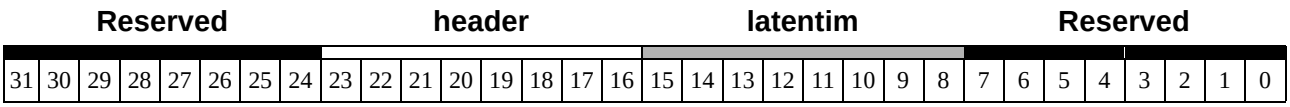
vendor <15:0> This field contains the Matrox manufacturer identifier for PCI: 102Bh.

device <31:16> This field contains the Matrox device identifier, which for the MGA-2164W-PCI is: 051Bh; for the MGA-2164W-AGP it is: 051Fh.

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. Writing has no effect. Reading will give ‘0’s.

Address 3Ch (CS)

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

Reset Value `0000 0000 0000 0000 0000 0001 1111 1111`b

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-2164W 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-2164W 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 4-25).

Address 44h (CS)

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

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b

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

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

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

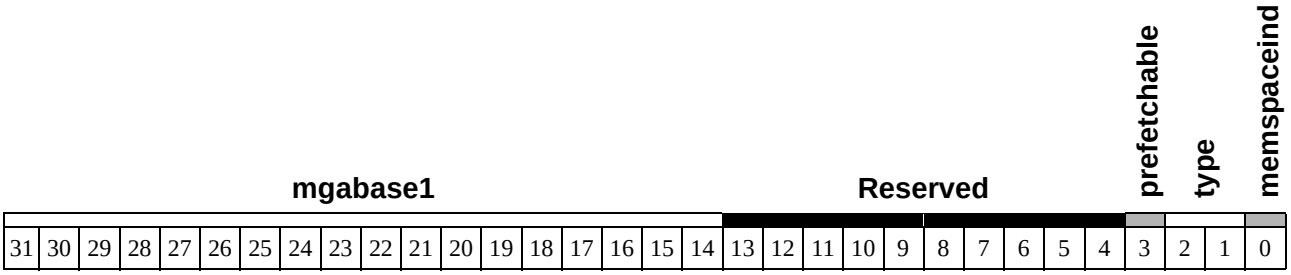
Reserved. When writing to this register, the bits in this field must be set to ‘0’. Reading will give ‘0’s.

The **MGA_INDEX** and **MGA_DATA** registers cannot be used in Pseudo-DMA mode (see page 4-25).

Address 14h (CS)

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

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



memspaceind 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 Kilobyte control aperture).

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

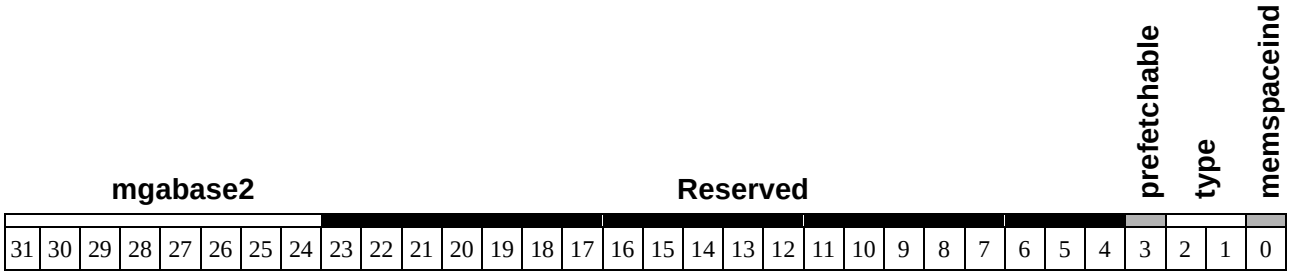
1. BIOS EPROM (highest precedence)
2. MGA control aperture
3. 8 MByte Pseudo-DMA window
4. VGA frame buffer aperture
5. MGA frame buffer aperture (lowest precedence)

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



memspaceind 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 megabytes 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)

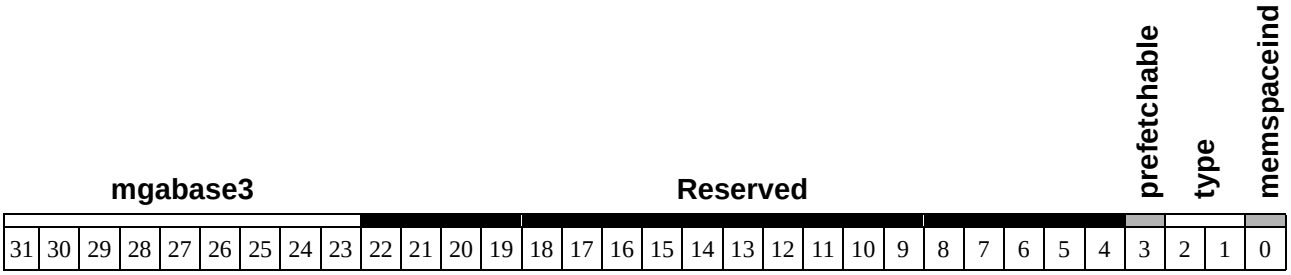
When **mgamode** = 0 (**CRTCEXT3**<7>), the MGA frame buffer Aperture is not usable.

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:

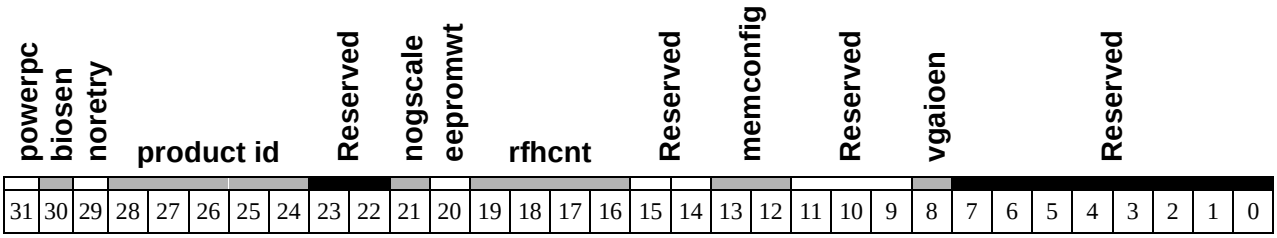
1. BIOS EPROM (highest precedence)
2. MGA control aperture
3. 8 MByte Pseudo-DMA window
4. VGA frame buffer aperture
5. MGA frame buffer aperture (lowest precedence)

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 0S0S SSSS 0000 0000 0000 000S 0000 0000b



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 (MDQ<5>) will replace the **vgaioen** value.

➧ ***Note:*** The MGA control registers and MGA frame buffer map are always enabled for all modes.

memconfig <13:12> Memory configuration. These 2 bits select the proper memory organization to match the type of external RAMDAC used.

memconfig	*Status*
00	32 bits RAMDAC, single frame buffer mode or 64 bits RAMDAC split frame buffer mode.
01	64 bits RAMDAC single frame buffer mode or 128 bits RAMDAC split frame buffer mode.
10	128 bits RAMDAC single frame buffer mode.
11	Reserved

➧ ***Note:*** The 128 bits RAMDAC single frame buffer mode requires a RAMDAC capable of de-interleaving the serial pixels.

➧ ***Note:*** This field must be set to 00b in vga emulation mode (mga mode = 0).

rfhcnt <19:16> Refresh counter. Defines the rate of the MGA-2164W's memory refresh. Note that the page cycles and co-processor acknowledges will not be interrupted by a refresh request unless a second request is queued (in this case, the refresh request becomes the highest priority after the screen refresh).

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

33.2 µ s >= (rfhcnt <3:1> * 512 + rfhcnt <0> *64 + 1) * gclk_ period * gscaling _factor

The gscaling_factor is tied to the value of nogscale, as shown below:

nogscale	gscaling_factor
'0'	4
'1'	1

Note: setting **rfhcnt** to zero halts the memory refresh. Since zero is the hard reset value, no refresh activity will take place after a reset. By waiting 200µs before programming this register, the proper memory initialization requirements will be met.

eepromwt <20> EEPROM write enable. When set to 1, a write access to the BIOS EPROM aperture will program that location. When set to 0, write access to the BIOS EPROM aperture has no effect.

nogscale <21> Graphic clock pre-scaler. When set to 0, the gclk signal is divided by 4 internally, and when set to 1, gclk is not divided. The gclk divider could be used when the external PLL is not able to lower the gclk enough to achieve power-down mode.

productid RO <28:24> Product ID. Sampled state of the MDQ<4:0> pins after a hard reset.

These bits are available to help board designers encode their product options so that the software and diagnostics can know which options are installed. (This field could encode the amount of memory, an indication if a writable ROM is present, and so on). These bits do not control hardware within the chip.

noretry <29> Retry disable. A ‘1’ disables generation of the retry sequence on the PCI bus (except during a VGA snoop cycle). At this setting, violation of the PCI latency rules may occur.

biosen <30> BIOS enable. On hard reset, the sampled biosen strap (MDQ<6>) 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.

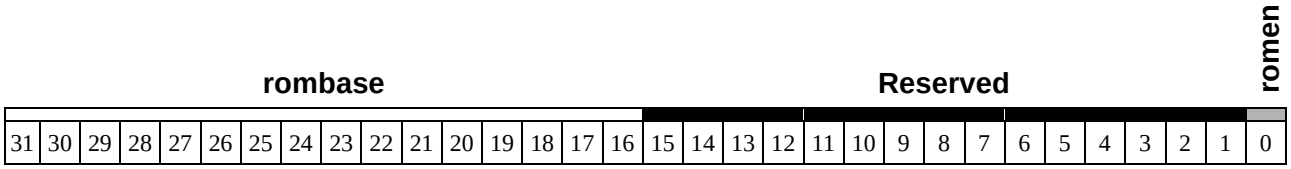
Reserved: <23:22><15:14><11:9><7:0>

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 EPROM used is application-specific (could be 32K or 64K).

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

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

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

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

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

Attributes BYTE/WORD/DWORD, STATIC

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b

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

subsysvid <15:0> Subsystem vendor ID. This field is reset with the value that is found in word location 7FF8h of the BIOS ROM (32K ROM used), or at word location FFF8h of the BIOS ROM (64K ROM used). It indicates a subsystem vendor ID as provided by the PCI Special Interest Group to the manufacturer of the add-in board which contains the MGA-2164W chip.

subsysid <31:16> Subsystem ID. This field is reset with the value that is found in word location 7FFAh of the BIOS ROM (32K ROM used), or at word location FFFAh of the BIOS ROM (64K ROM used). It indicates a subsystem ID as determined by the manufacturer of the add-in board which contains the MGA-2164W chip.

➨ ***Note***: If the biosen strap is '0', the ROM will not be read and the value found in the register will be 00000000h. In this case, the driver ***must write the correct values*** to this register (at location 4Ch) after power-up.

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

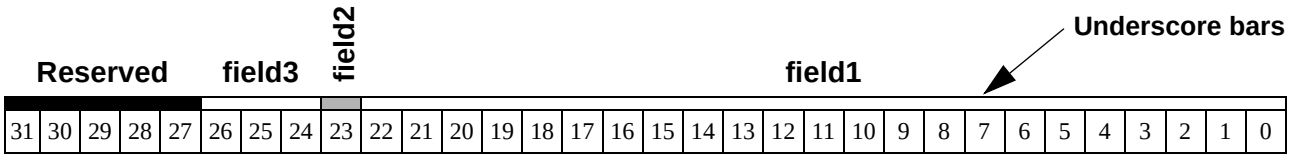
➨ ***Note:*** There may be a delay of up to 500 PCLKs following a hard reset before this register is initialized.

3.1.2 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. Reserved fields are identified by black underscore bars; all other fields display alternating white and gray bars.

Sample Power Graphic Mode Memory Space Register	**SAMPLE_PG**
Address	<value>
Attributes	R/W
Reset Value	<value>

Main header



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 2. Note: MEM indicates that the address lies in the memory space; IO indicates that the address lies in the I/O space.

Attributes

The Power Graphic Mode attributes are:

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

Reset Value

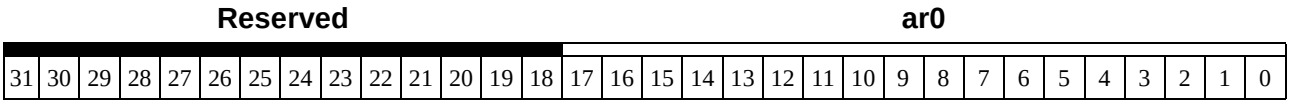
Here are some of the symbols that appear as part of a register's reset value. Most bits are reset on hard reset. Some bits are also reset on soft reset, and they are underlined when they appear in the register description headers.

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

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

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown



> ***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.

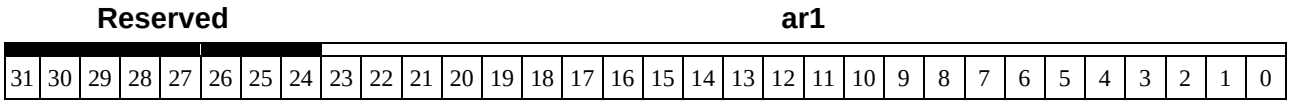
- For AUTOLINE, this register holds the x end address (in pixels). See the **XYEND** register on page 3-84.
- For LINE, it holds 2 x 'b'.
- For a filled trapezoid, it holds 'dYl'.
- For a BLIT, **ar0** holds the line end source address (in pixels).
- For an ILOAD_SCALE or ILOAD_FILTER, **ar0** holds 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'

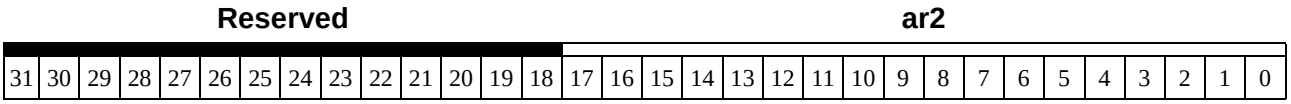
- For a BLIT, **ar1** holds the line start source address (in pixels). Because the start source address is also required by **ar3**, and because **ar1** is loaded when writing **ar3** this register doesn't need to be explicitly initialized.
- In the ILOAD_SCALE and ILOAD_FILTER algorithms, **ar1** contains 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



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 3-84.
- 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:

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

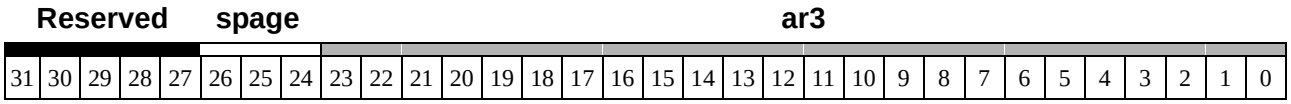
This register is **not** used for BLIT operations without scaling.

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.

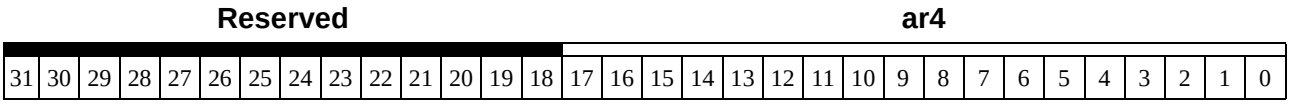
Reserved <31:27>

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



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'

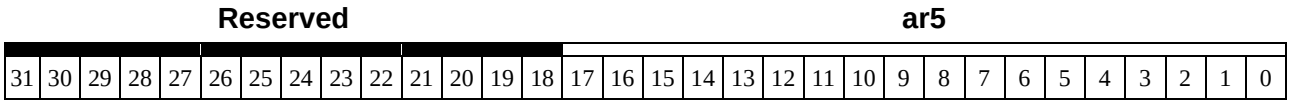
• This register is used during AUTOLINE, but doesn't need to be initialized.
• This register is not used for LINE or BLIT operations without scaling.
• For the ILOAD_SCALE, ILOAD_FILTER, ILOAD_HIQH, and ILOAD_HIQHV, it holds the error term, but it doesn't need to be initialized.

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.

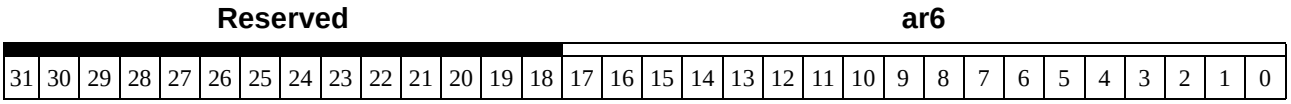
- At the beginning of AUTOLINE, **ar5** holds the x start address (in pixels). See the **XYSTRT** register on page 3-85. At the end of AUTOLINE the register is loaded with the x end, so it is not necessary to reload the register when drawing a polyline.
- This register is not used for LINE without auto 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



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 3-85. 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)

Note: 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) or IDUMP operations.

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 4-19 for the definition of the slice in each mode.

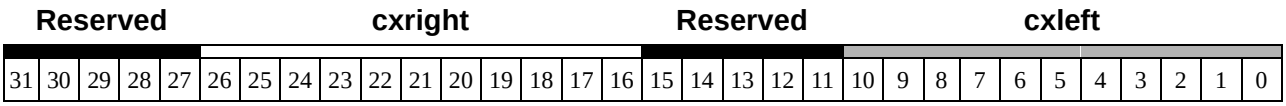
bltcmsk <31:0> Blit color mask. This field enables blit transparency comparison on a planar basis ('0' indicates a masked bit). Refer to the description of the **transc** field of **DWGCTL** for the transparency equation.

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



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 3-32.

cxright <26:16> Clipper x right boundary. See the **CXRIGHT** register on page 3-33.

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

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

cxleft <10:0> Clipper x left boundary. The **cxleft** field contains an unsigned 11-bit value which is interpreted as a positive pixel address and compared with the current **xdst** (see **YDST** on page 3-87). 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.

➧ ***Note:*** 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 3-87). The value of **xdst** must be less than or equal to **cxright** to be inside the drawing window.

Note: 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 DWGREG0 range )
     map_reg? = ( drawing_reg byte address >> 2 )
     & 0 x 7F
else if ( address is within DWGREG1 range )
     map_reg? = (drawing byte address >> 2)
     & 0 x 7F | 0 x 80
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 DWGREG0 range )
     map_reg? = ( drawing_reg byte address >> 2 )
     & 0 x 7F
else if ( address is within DWGREG1 range )
     map_reg? = (drawing byte address >> 2)
     & 0 x 7F | 0 x 80
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 DWGREG0 range )
     map_reg? = ( drawing_reg byte address >> 2 )
     & 0 x 7F
else if ( address is within DWGREG1 range )
     map_reg? = (drawing byte address >> 2)
     & 0 x 7F | 0 x 80
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 DWGREG0 range )
     map_reg? = ( drawing_reg byte address >> 2 )
     & 0 x 7F
else if ( address is within DWGREG1 range )
     map_reg? = (drawing byte address >> 2)
     & 0 x 7F | 0 x 80
else
     error, can't use indirect mapping
```

Address MGABASE1 + 1C54h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value Unknown

dmapad

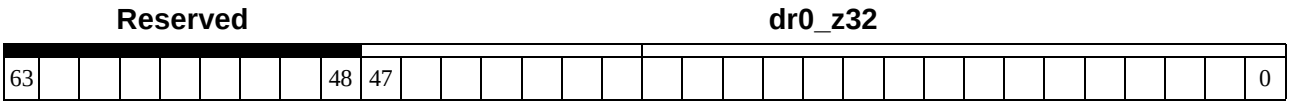
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

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

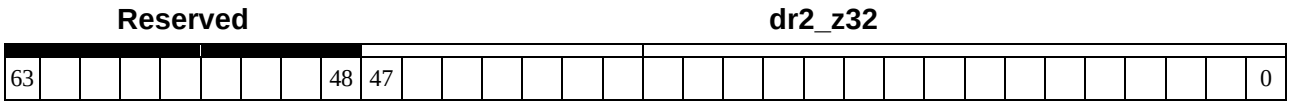
- For TRAP with z, the DR0_Z32 register is used to scan the left edge of the trapezoid and must be initialized with its starting z value. In this case, DR0_Z32 is signed 33.15 in two's complement notation.
- For LINE with z, the DR0_Z32 register holds the z value for the current drawn pixel and must 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. Reading will give '0's.

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

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown



dr2_z32 <47:0> Data ALU 2

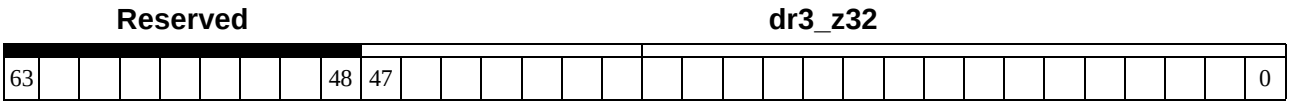
- For TRAP with z, the DR2_Z32 register is used to scan the left edge of the trapezoid and must be initialized with its starting z value. In this case, DR2_Z32 is signed 33.15 in two's complement notation.
- For LINE with z, the DR2_Z32 register holds the z value for the current drawn pixel and must 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. Reading will give '0's.

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

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown



dr3_z32 <47:0> Data ALU 3.
- For 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. Reading will give '0's.

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

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown

dr0

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

dr0 <31:0> Data ALU register 0.

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

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_32MSB. 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 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 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

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

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

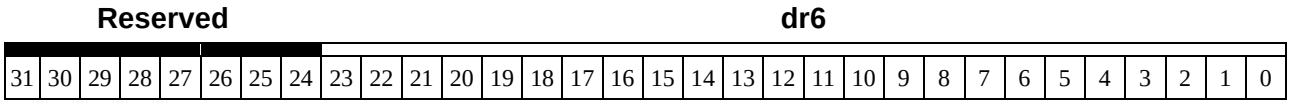
- For TRAP with z, the DR4 register is used to scan the left edge of the trapezoid for the red color (Gouraud shading). This register must be initialized with its starting red color value.
- For TRAP_ILOAD, this register is not used, and will be corrupted.
- For LINE with z, the DR4 register holds the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

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

Address MGABASE1 + 1CD8h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value Unknown



dr6 <23:0> Data ALU register 6. This field holds a signed 9.15 value in two's complement notation.

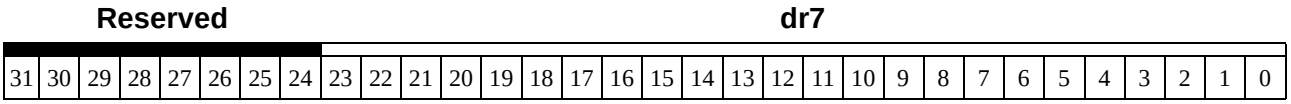
- For TRAP with z, the DR6 register holds the red increment value along the x-axis.
- For TRAP_ILOAD, this register is not used.
- For LINE with z, the DR6 register holds the red increment value along the major axis.

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

Address MGABASE1 + 1CDCh (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value Unknown



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

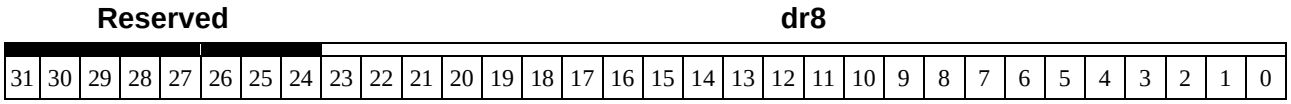
- For TRAP with z, the DR7 register holds the red increment value along the y-axis.
- For TRAP_ILOAD, this register is not used.
- For LINE with z, the DR7 register holds the red increment value along the diagonal axis.

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

Address MGABASE1 + 1CE0h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown



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

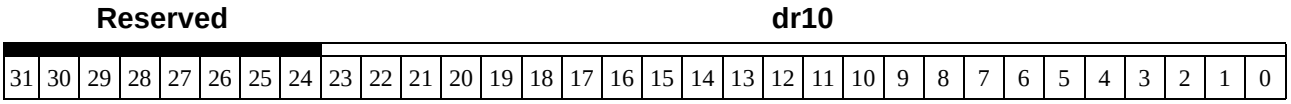
- For TRAP with z, the DR8 register is used to scan the left edge of the trapezoid for the green color (Gouraud shading). This register must be initialized with its starting green color value.
- For TRAP_ILOAD, this register is not used, but will be corrupted.
- For LINE with z, the DR8 register holds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

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

Address MGABASE1 + 1CE8h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value Unknown



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

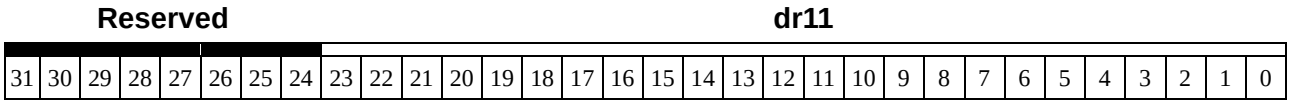
- For TRAP with z, the **DR10** register holds the green increment value along the x-axis.
- For TRAP_ILOAD, this register is not used.
- For LINE with z, the **DR10** register holds the green increment value along the major axis.

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

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.

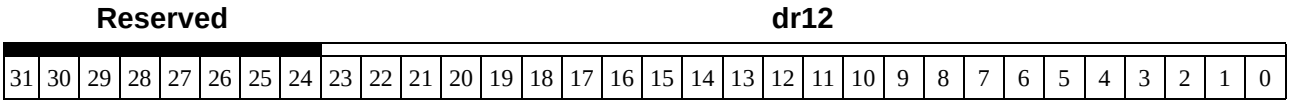
- For TRAP with z, the DR11 register holds the green increment value along the y-axis.
- For TRAP_ILOAD, this register is not used.
- For LINE with z, the DR11 register holds the green increment value along the diagonal axis.

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.

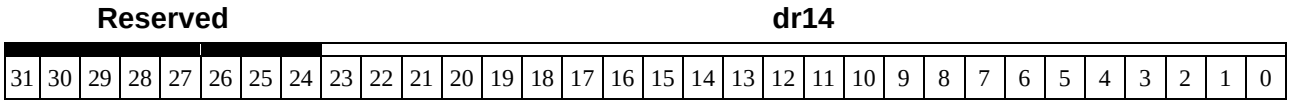
- For TRAP with z, the **DR12** register is used to scan the left edge of the trapezoid for the blue color (Gouraud shading). This register must be initialized with its starting blue color value.
- For TRAP_ILOAD, this register is not used, but will be corrupted.
- For LINE with z, the **DR12** register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

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.

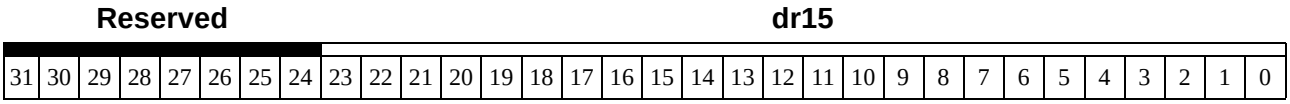
- For TRAP with z, the DR14 register holds the blue increment value along the x-axis.
- For TRAP_ILOAD, this register is not used.
- For LINE with z, the DR14 register holds the blue increment value along the major axis.

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.

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

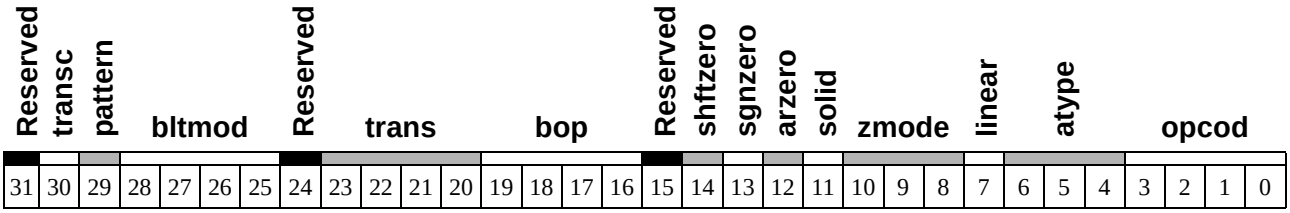
lutentry N <31:0> These 16 registers are a lookup table that can be used in conjunction with the **DMAMAP** registers. Writing to these locations address the register that is programmed in the Nth byte of the **DMAMAP**. This indirect write register provides a means to access non-sequential drawing registers 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
Blit	RAM -> RAM	‘1000’	BITBLT
	RAM -> RAM	‘1100’	FBITBLIT
	HOST -> RAM	‘1001’	ILOAD
	HOST -> RAM scale	‘1101’	ILOAD_SCALE
	HOST -> RAM scale, filter	‘1111’	ILOAD_FILTER
	RAM -> HOST	‘1010’	IDUMP
	HOST -> RAM scale, high-quality filter	‘0111’	ILOAD_HIQH
	HOST -> RAM horizontal and vertical scale, high-quality filter	‘1110’	ILOAD_HIQHV
	Reserved	‘1011’	

atype <6:4>

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

atype		RAM Access
Value	*Mnemonic*	
'000'	RPL	Write (replace)
'001'	RSTR	Read-modify-write (raster)
'010'		Reserved
'011'	ZI	Depth mode with Gouraud
'100'	BLK	Block write mode (1)
'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		Pixel Update
Value	*Mnemonic*	
'000'	NOZCMP	Always
'001'		Reserved
'010'	ZE	When depth is =
'011'	ZNE	When depth is < >
'100'	ZLT	When depth is <
'101'	ZLTE	When depth is <=
'110'	ZGT	When depth is >
'111'	ZGTE	When depth is >=

solid <11>

Solid line or constant trapezoid. The solid register is not a physical register. It provides an alternate way to load the **SRC** registers (see page 3-80).

- 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 3-23).

- 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 3-74) is added to **ydst** (see **YDST** on page 3-87)

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

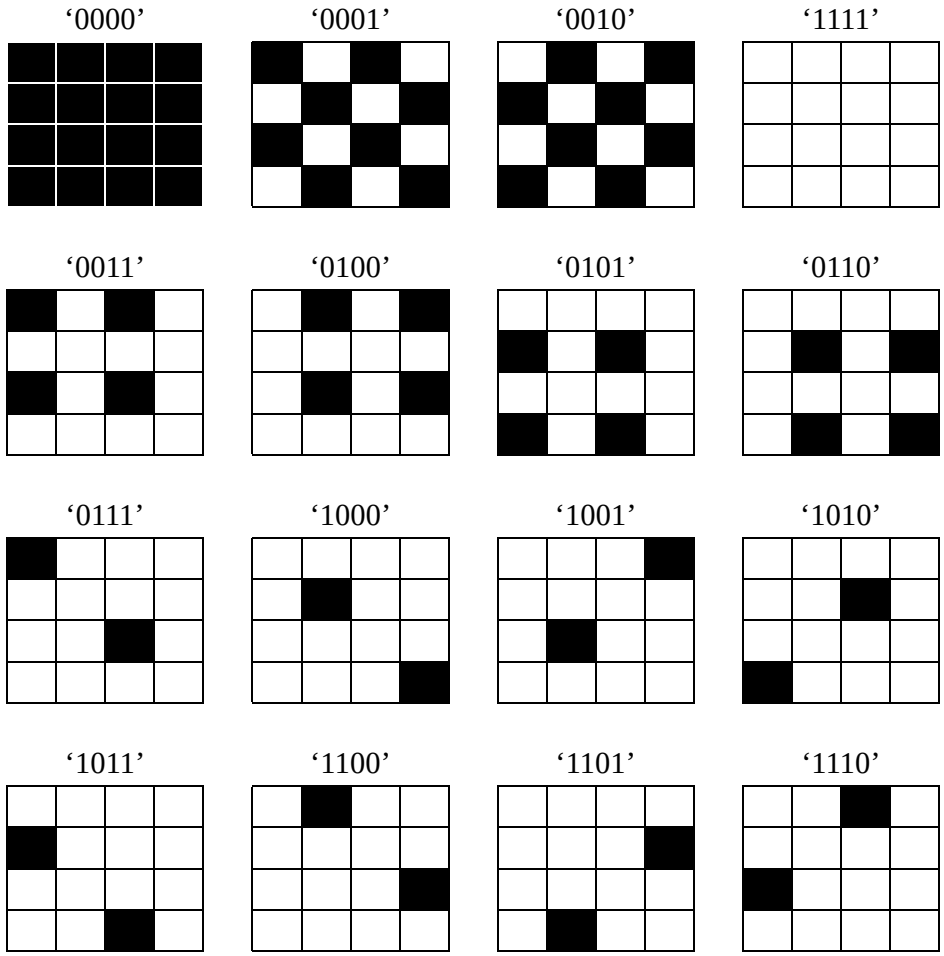
bop <19:16>

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>

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



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

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

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

Refer to the subsections contained in 'Drawing in Power Graphic Mode' on page 4-25 for more information on how to use this field. That section also presents the definition of the various pixel formats.

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

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

transc <30>

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

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

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

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

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

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

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

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 4-19 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/pixel), 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>.

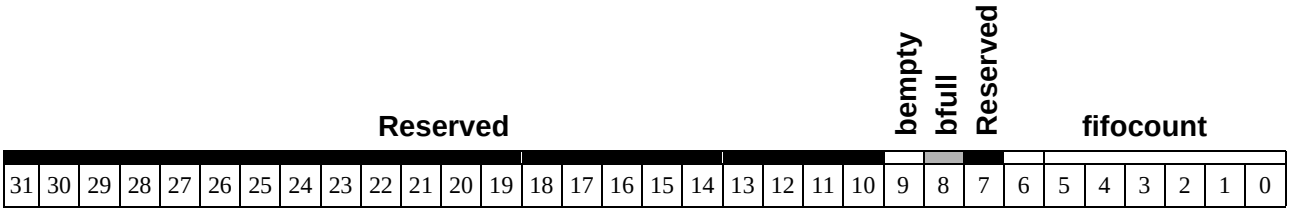
bltckey <31:0> Blit color key. This field specifies the value of the color that is defined as the 'transparent' color. Planes that are not used must be set to '0'. Refer to the description of the **transc** field of **DWGCTL** for the transparency equation

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 <5:0> Indicates the number of free locations in the Bus FIFO. On soft or hard reset, the contents of the Bus FIFO are flushed and the FIFO count is set to 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).

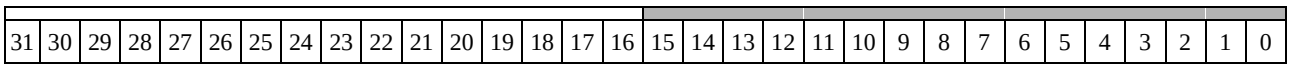
Reserved: **<7> <31:10>**

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

Address MGABASE1 + 1C84h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown



The **FXBNDRY** register is not a physical register; it is a more efficient way to load the **FXRIGHT** and **FXLEFT** registers.

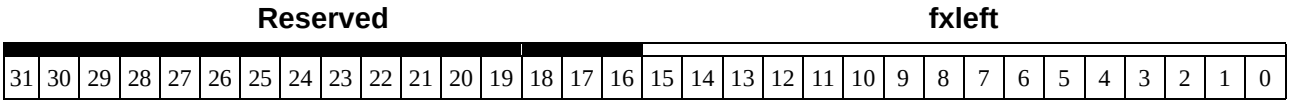
fxleft <15:0> Filled object x left-coordinate. Refer to the **FXLEFT** register for a detailed description.

fxright <31:16> Filled object x right-coordinate. See the **FXRIGHT** register on page 3-66.

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.

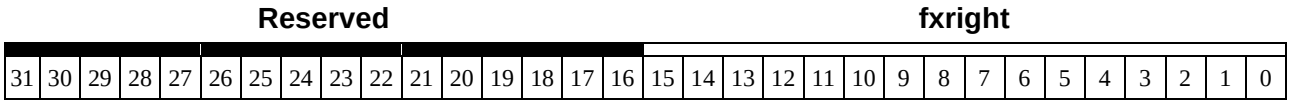
- The **fxleft** field is not used for line drawing.
- During filled trapezoid drawing, **fxleft** is updated during the left edge scan.
- During a BLIT operation, **fxleft** is static, and specifies the left pixel boundary of the area being written to.

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.

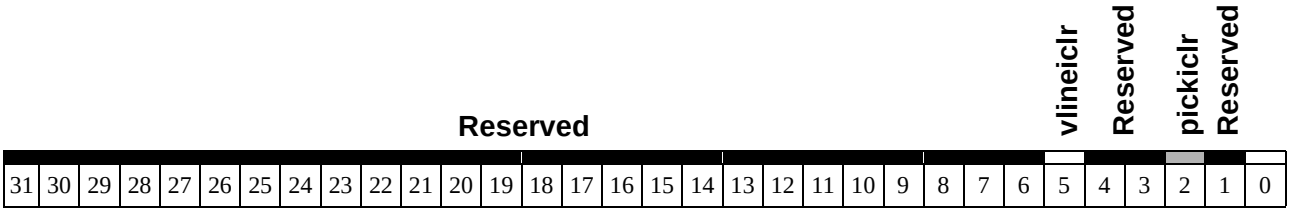
- The **fxright** field is not used for line drawing.
- During filled trapezoid drawing, **fxright** is updated during the right edge scan.
- During a BLIT operation, **fxright** is static, and specifies the right pixel boundary of the area being written to.

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



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.

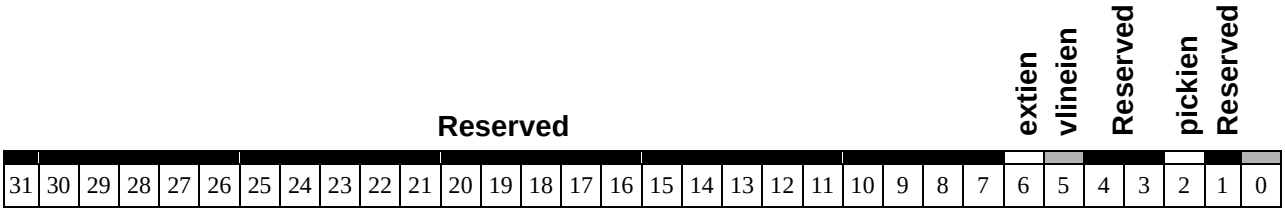
Reserved: <1> <4:3> <31:6>

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



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.

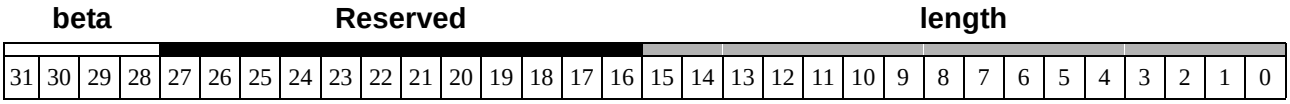
Reserved: <1> <4:3> <31:7>

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.

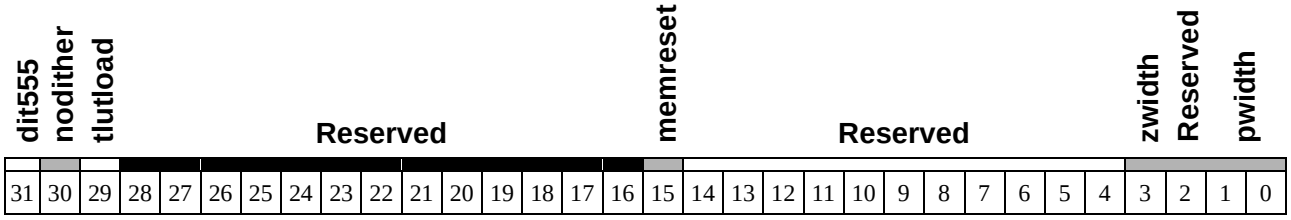
beta <31:28> Beta factor. This field is used to drive the vertical scaling in ILOAD_HIQHV (it is not used for other opcodes). The **beta** field represents the four least significant bits of a value between 1 and 16 (16 is 0000b), which represents a beta factor of 1/16 through 16/16.

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

memreset <15> Resets the RAM. When this bit is set to '1', the memory sequencer will generate a reset cycle to the RAMs.

Caution: Refer to Section 4.3.3 on page 4-22 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.

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

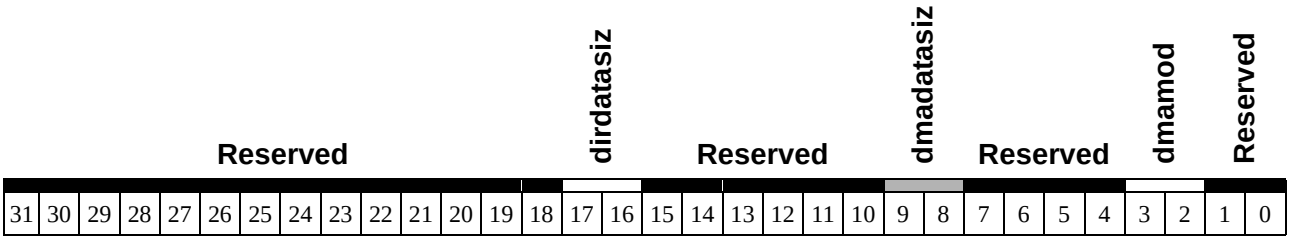
Reserved **<14:4><2><28:16>**

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.

dmadatasiz *<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.

dirdatasiz *<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				

Note: 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.

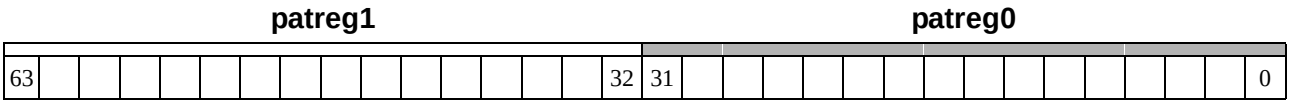
Reserved: **<1:0> <7:4> <15:10> <31:18>**

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)

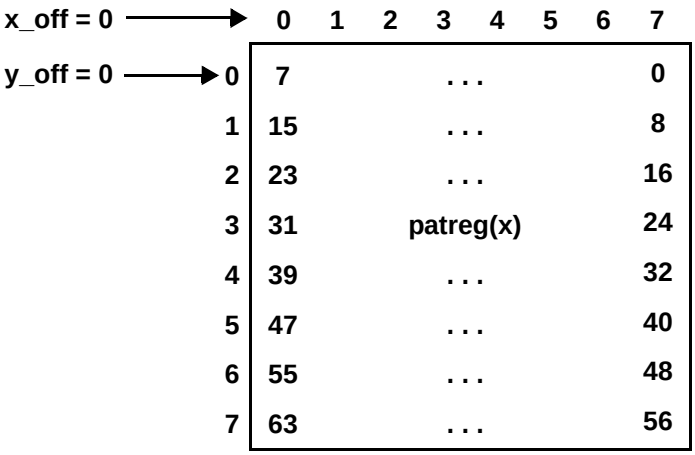
Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown



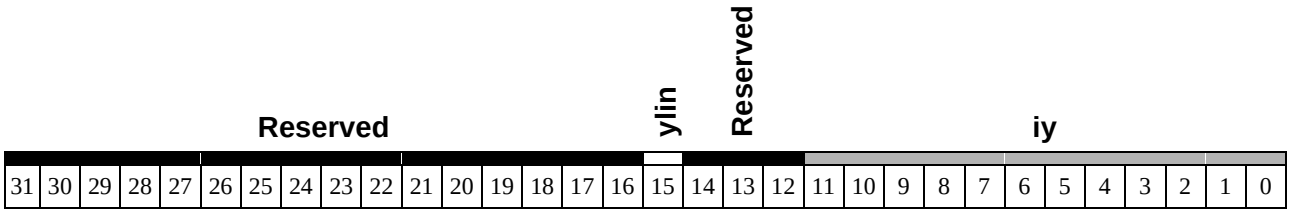
patreg <63:0> Pattern register. The **PAT** registers are not physical registers. They simply provide an alternate way to load the **SRC** registers with a Windows format 8 x 8 pattern.

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 3-80.

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 measured in pixel unit and must be a multiple of 32 (the five LSB = 0). It 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 3-87) between two destination lines. The **iy** field is also used as the multiplication factor for linearizing the **ydst** register.

The hardware linearization unit is capable of only a few values of pitch. If the required pitch is not within the hardware capabilities, the **ylin** bit should be used to disable the linearization operation and the linearization need to be performed in software. The following table provides the supported pitches for linearization:

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

This register must be loaded with a value that is a multiple of 32, 64, 128, or 256 due to a restriction involving block mode, according to the table below. See ‘Constant Shaded Trapezoids / Rectangle Fills’ on page 4-36. See page 3-56 for additional restrictions that apply to block mode (**atype** = BLK).

pwidth	memconfig = 00	memconfig = 01
PW8	64	128
PW16	32	64
PW24	64	128
PW32	32	32

ylin <15>

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

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

Reserved: <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

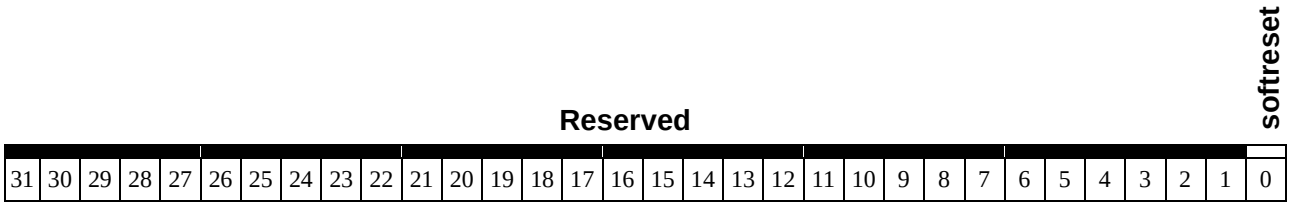
The bits from the **plnwrmsk**<31:0> register are output on the MDQ<31:0> signal and also on MDQ<63:32>. In 8 and 16 bit/pixel configurations, all bits in **plnwrmsk**<31:0> are used, so the mask information must be replicated on all bytes. In 24 bits/pixel, the plane masking feature is limited to the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

Refer to ‘Pixel Format’ on page 4-19 for the definition of the slice in each mode.

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’, this resets all bits that permit software resets. This has the effect of flushing the BFIFO, the MOFIFO (used by idump), 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 µs to ensure correct reset. After that period, a ‘0’ must be programmed to remove the soft reset.

Refer to Section 4.3.3 on page 4-22 for instructions on when to use this field.

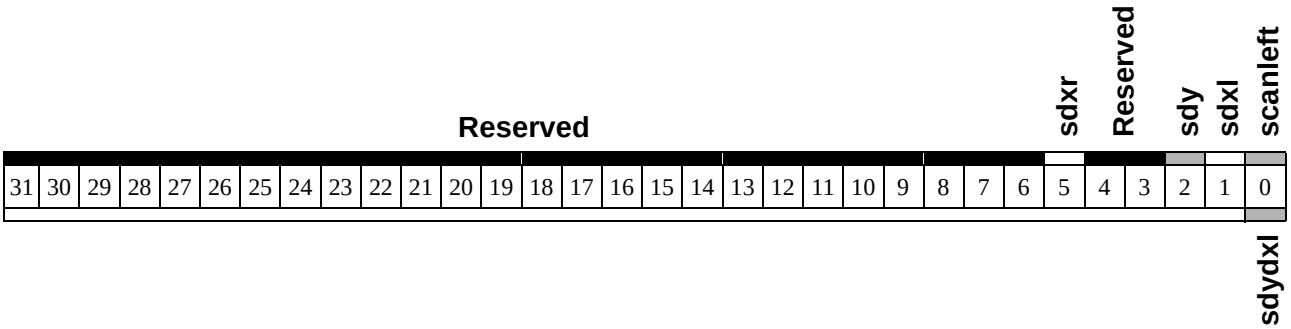
WARNING! A soft reset will not re-read the chip strapping.

Reserved <31:1> Reserved. When writing to this register, the bits in this field must be set to ‘0’. Reading will give ‘0’s.

Address **MGABASE1** + 1C58h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value Unknown



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

sdydxl <0> Sign of delta y minus delta x. This bit is shared with **scanleft**. It is defined for LINE drawing only and specifies the major axis. This bit is automatically initialized during AUTOLINE operations.

- 0: major axis is y
- 1: major axis is x

scanleft <0> Horizontal scan direction left (1) vs. right (0). This bit is shared with **sdydxl** and affects TRAPs and BLITs; **scanleft** is set according to the x scanning direction in a BLIT.

Normally, this bit is always programmed to zero except for BITBLT when **bltmod** = BFCOL (see **DWGCTL** on page 3-55). 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

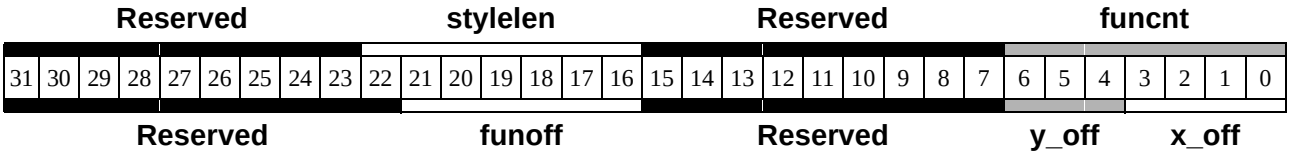
Reserved: <4:3> <31:6>

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 3-80), so its value is the number of bits in the complete pattern minus one. For 3D vectors, this field must be initialized to ‘0’.

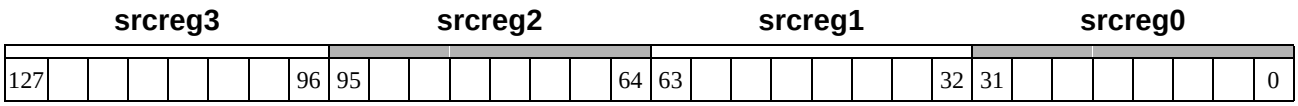
Reserved: <15:7> <31:23/22>

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

Address **MGABASE1** + 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 4.5.4.3 on page 4-30 for more details.

For TRAP with the RPL or RSTR attribute, the source register is used to store an 8×8 pattern (the odd bytes of the SRC registers must be a copy of the even bytes). Refer to Section 4.5.5.3 on page 4-37 for more details.

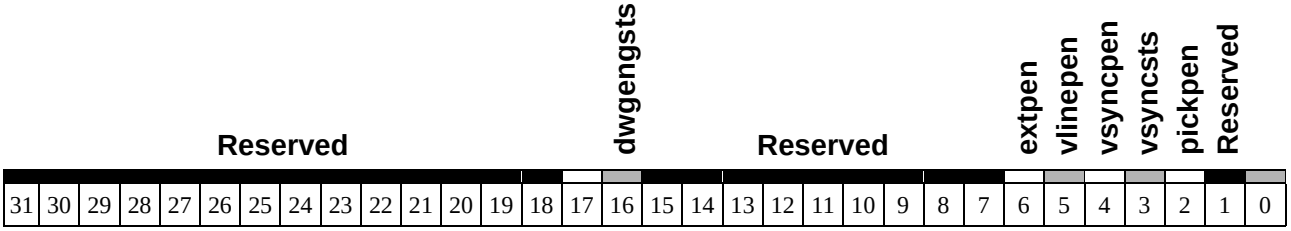
For all BLIT operations, and for TRAP or LINE using depth mode, the source register is used internally for intermediate data.

A write to the **PAT** registers (see page 3-73) will load the **SRC** registers.

Address **MGABASE1** + 1E14h (MEM)

Attributes RO, DYNAMIC, BYTE/WORD/DWORD

Reset Value 0000 0000 0000 0000 0000 0000 0?00 0000b



pickpen <2> Pick interrupt pending. When set to '1', indicates that a pick interrupt has occurred.

This bit is cleared through the **pickiclr** bit (see **ICLEAR** on page 3-67) 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 3-133.** This bit is cleared through the **vlineiclr** bit (see **ICLEAR** on page 3-67) or upon soft or hard reset.

extpen <6> External interrupt pending. When set to '1', indicates that the external interrupt line is driven. This bit is cleared by conforming to the interrupt clear protocol of the external device that drive the EXTINT/ line. After a hard reset, the state of this bit is unknown (as indicated by the question mark in the 'Reset Value' above), as it depends on the state of the EXTINT/ pin during the hard reset.

dwgengsts <16> Drawing engine status. Set to '1' when the drawing engine is busy (a busy condition will be maintained until the BFIFO is empty, the drawing engine is finished with the last drawing command, and the memory controller has completed the last memory access)

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

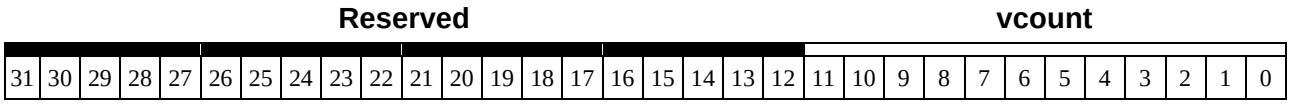
Reserved. When writing to this register, the bits in these fields must be set to '0'. Reading will give '0's.

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 + 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.

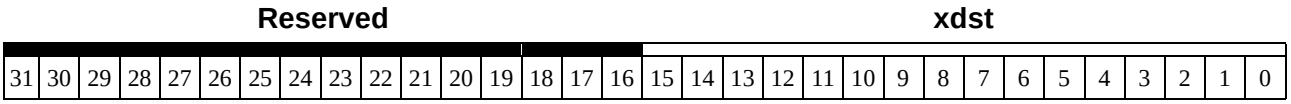
Note: This register must be read using a word or dword access, because the value might change between two byte accesses. A sample and hold circuit will ensure a stable value for the duration of one PCI read access.

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 3-65) and **fxright** (see **FXRIGHT** on page 3-66), 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 3-32.)

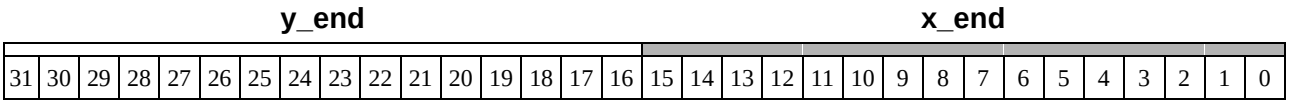
Reserved <31:16>

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.

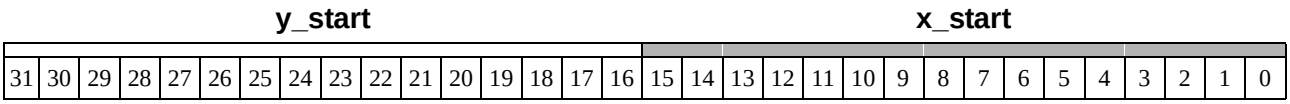
When **XYEND** is written, the following registers are affected:

- **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.

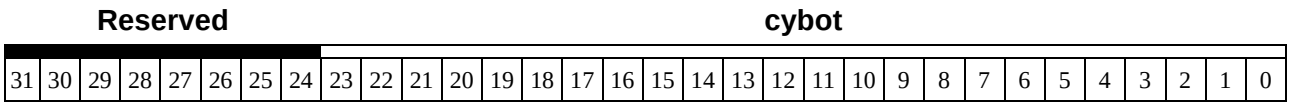
When **XYSTRT** is written, the following registers are affected:

- **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 3-87). 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

sellin	Reserved	ydst

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

ydst <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 3-74).

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.

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.

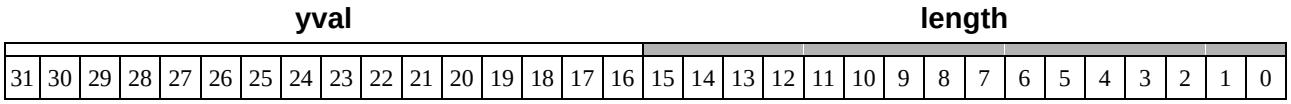
Reserved <28:23>

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



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 3-69.

yval <31:16>

The y destination value. See the **YDST** register on page 3-87. 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.

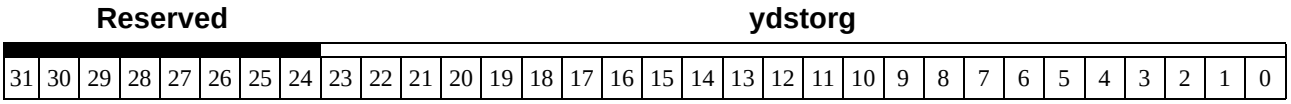
ydst<22:0> <= sign extension (**yval**<31:16>)

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



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 intensity buffer. This register is used to initialize the YDST address.

This register must be loaded with a value that is a multiple of 32, 64, 128, or 256 according to the table below, due to a restriction involving block mode. See 'Constant Shaded Trapezoids / Rectangle Fills' on page 4-36. See page 3-56 for additional restrictions that apply to block mode (**atype** = BLK).

pwidth	memconfig = 00	memconfig = 01	memconfig = 10
PW8	64	128	256
PW16	32	64	128
PW24	64	128	256
PW32	32	32	64

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

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

cytop <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 3-87). 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: The **cytop** value is interpreted as positive, any negative **ydst** value is automatically outside the clipping window.

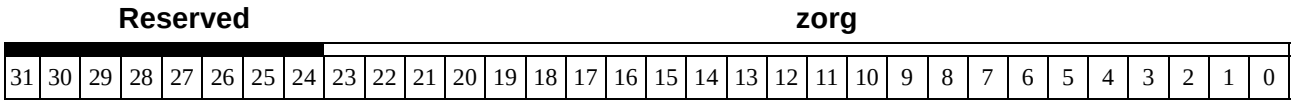
There is no way to disable clipping.

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).

Equation	zwidth
zorg = Z depth origin - ydstorg * 2	0
zorg = Z depth origin - ydstorg * 4	1

Reserved <31:24>

Reserved. When writing to this register, the bits in this field are ignored.

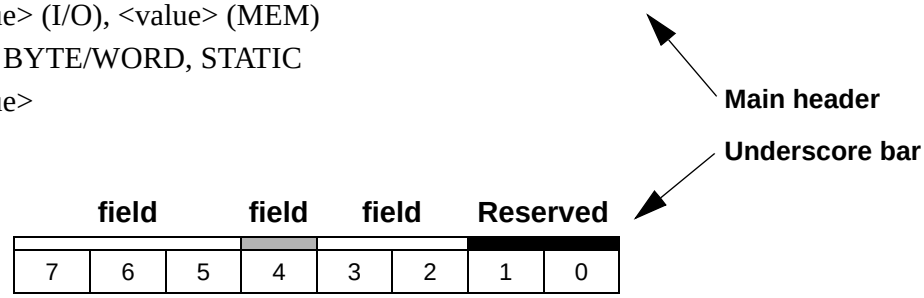
Setting ZORG to 200000h or 400000h will yield the fastest performance for primitives using the Z buffer.

3.2 VGA Mode Register Descriptions

The MGA-2164W 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. Reserved bit fields are identified by black underscore bars; all other fields display alternating white and gray bars.

Sample VGA Mode Register Description — SAMPLE_VGA

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



Address

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

Index

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

Attributes

The VGA mode attributes are:

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

Reset Value

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

Address R/W at port 03C0h (I/O), **MGABASE1** + 1FC0h (MEM) VGA
R at port 03C1h (I/O), **MGABASE1** + 1FC1h (MEM) VGA

Attributes BYTE, STATIC

Reset Value nnnn nnnn 0000 0000b

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

attrx <4:0> Attribute controller index register. VGA.

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

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

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

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

Example of a palette write:

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

Example of a palette read:

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

pas <5>

Palette address source. VGA.

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

attrd <15:8>

ATTR data register.

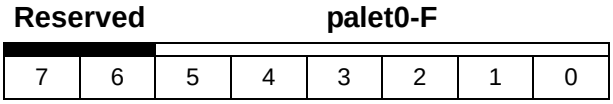
Retrieve or write the contents of the register pointed to by the **attrx** field.

Reserved <7:6>

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

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

Reset Value 0000 0000b



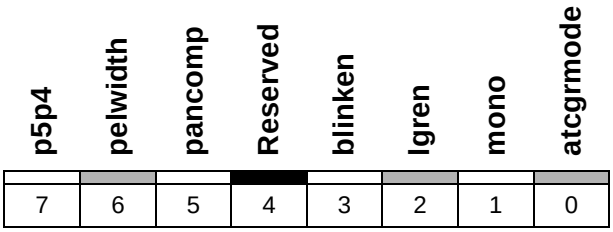
palet0-F <5:0> Internal palette data. VGA.

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 3-102.

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

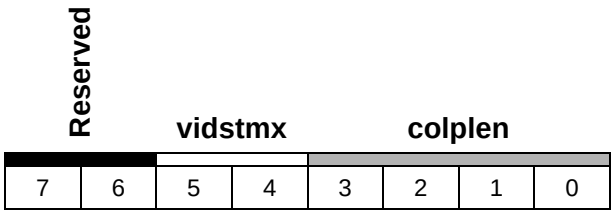
ovscol <7:0>

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 3-159.

Reserved <7:6> Reserved. When writing to this register, the bits in this field must be set to 0.13.

Index **attrx** = 13h

Reset Value 0000 0000b

Reserved				hpelcnt			
7	6	5	4	3	2	1	0

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

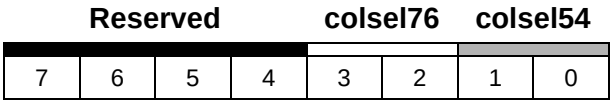
This 4-bit value specifies the number of picture elements to shift the video data horizontally to the left, according to the following table (values 9 to 15 are reserved):

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

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

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

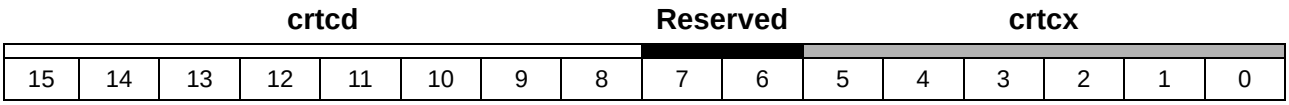
cacheflush <7:0> Flush the cache. Writes to this register will flush the cache. For additional details, refer to 'Direct Access Read Cache' on page 4-4.

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>

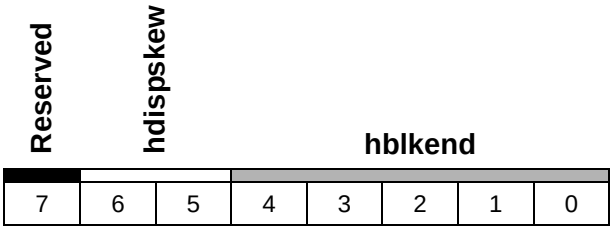
Start horizontal blanking. VGA/MGA.

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>

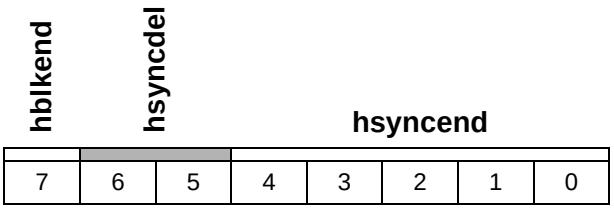
Start horizontal retrace pulse. VGA/MGA.

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 3-109.

Index **crtcx** = 06h

Reset Value 0000 0000b

vtotal

7	6	5	4	3	2	1	0

vtotal <7:0>

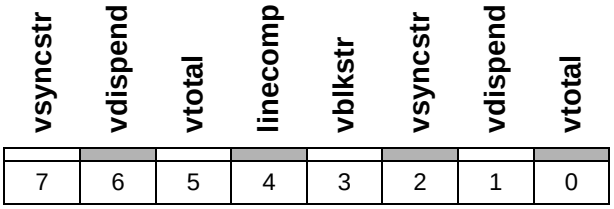
Vertical total. VGA/MGA.

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 3-112.

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 3-124.

vsyncstr <2> Vertical retrace start bit 8. VGA/MGA.
Contains bit 8 of the Vertical Retrace Start. See the **CRTC10** register on page 3-122.

vblkstr <3> Start vertical blank bit 8. VGA/MGA.
Contains bit 8 of the Start Vertical Blank. See the **CRTC15** register on page 3-127.

linecomp <4> Line compare bit 8. VGA/MGA.
Line compare bit 8. See the **CRTC18** register on page 3-133. This bit is not write-protected by **crtcprotect** (**CRTC11**<7>).

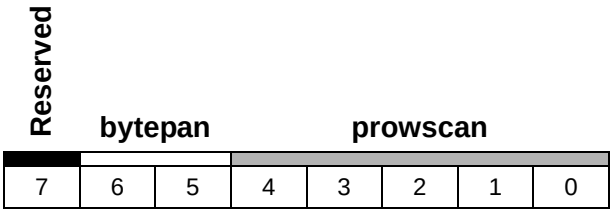
vtotal <5> Vertical total bit 9. VGA/MGA.
Contains bit 9 of the Vertical Total. See the **CRTC6** register on page 3-112.

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 3-124.

vsyncstr <7> Vertical retrace start bit 9. VGA/MGA.
Contains bit 9 of the Vertical Retrace Start. See the **CRTC10** register on page 3-122.

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

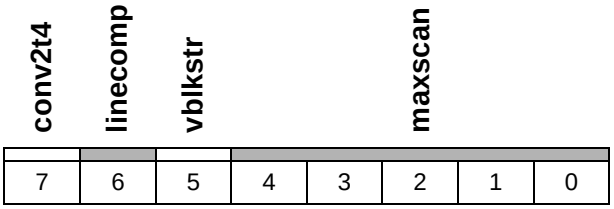
bytepan <6:5> Byte panning control. VGA/MGA.

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 3-127.

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

Bit 9 of the Line Compare register. See the **CRTC18** register on page 3-133.

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

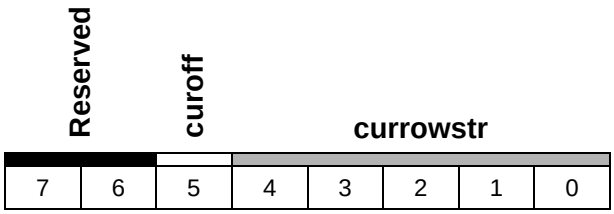
Controls the row scan counter clock and the time when the start address latch loads a new memory address:

- **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.

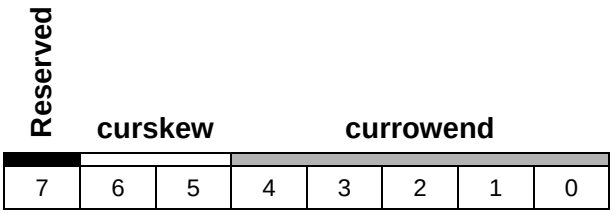
When the cursor start register is programmed with a value greater than the cursor end register, no cursor is generated.

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 20-bit value from the **startadd** (**CRTCEXT0**<3:0>) high-order and (**CRTCD**<7:0>) low-order start address registers is the first address after the vertical retrace on each screen refresh.

See 'Programming in Power Graphic Mode' on page 4-69 for more information on **startadd** programming.

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 3-118.

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 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 3-120.

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.

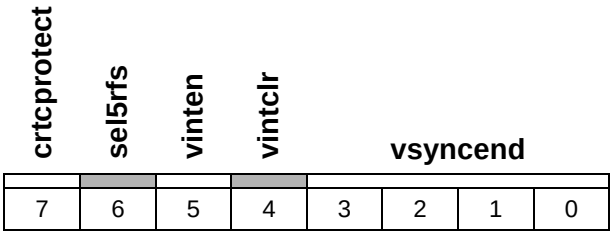
The vertical sync signal becomes active when the vertical line counter reaches the vertical retrace start value (a 12-bit value). The lower eight bits are located here. Bit 8 is in **CRTC7**<2>, bit 9 is in **CRTC7**<7>, and bits 10 and 11 are in **CRTCEXT2**<6:5>.

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

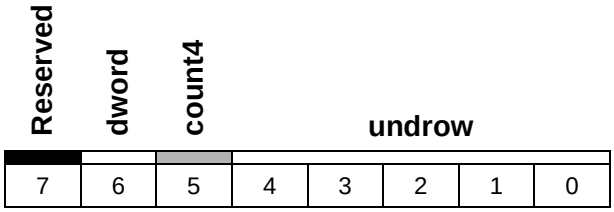
offset <7:0> Logical line width of the screen. VGA/MGA.

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>. The value is the number of double words (**dword** (**CRTC14**<6>) = 1) or single words (**dword** = 0) in one line.

See 'Programming in Power Graphic Mode' on page 4-69 for more information about **offset** programming.

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 supersede **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'.

2 ***Note:*** In MGA mode, dword 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

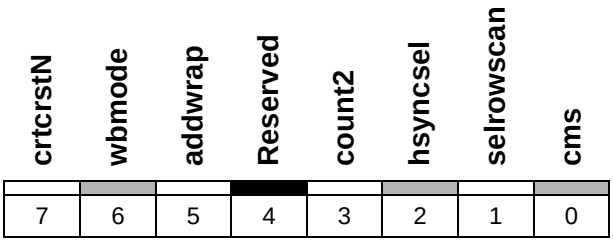
vblkend <7:0>

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.

- 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.

2 ***Note:*** In MGA mode, wbmode must be set to 1, selrowscan set to 1, and cms to 1.

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

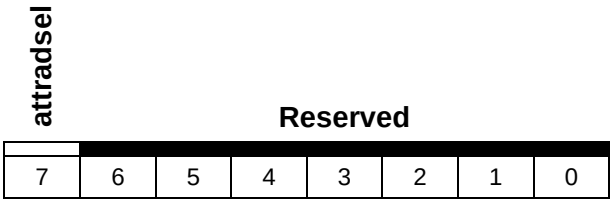
7	6	5	4	3	2	1	0

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



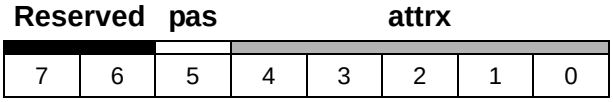
attradsel <7> Attributes address/data select. VGA.

- 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 3-94.

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

Attributes R/W, BYTE/WORD, STATIC

Reset Value nnnn nnnn 0000 0000b

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

crtcextx <2:0> CRTC extension index register.

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

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

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

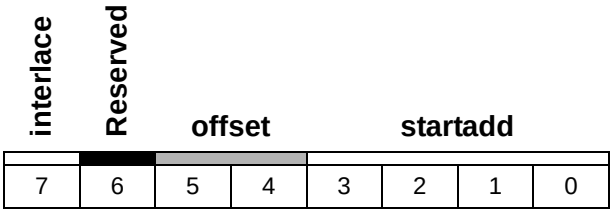
crtcextd <15:8> CRTC extension data register.

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

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

Index **crtcextx** = 00h

Reset Value 0000 0000b



startadd <3:0>

Start address bits 19, 18, 17, and 16.

These are the four most significant bits of the start address. See the **CRTCC** register on page 3-118.

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 3-125.

interlace <7>

Interlace enable.

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

Reserved <6>

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

Index **crtcextx** = 01h

Reset Value 0000 0000b

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

htotal <0>
Horizontal total bit 8.

This is the most significant bit of the **htotal** (horizontal total) register. See the **CRTC0** register on page 3-106.

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 3-108.

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 3-110.

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**).

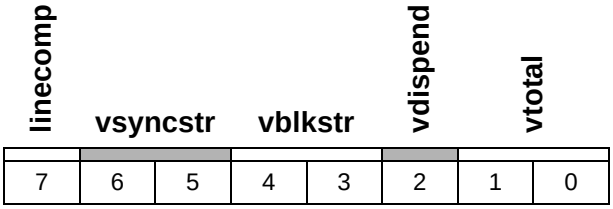
Bit 6 of the End Horizontal Blanking value. See the **CRTC3** register on page 3-109.

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 3-112.

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 3-124.

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 3-127.

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 3-122.

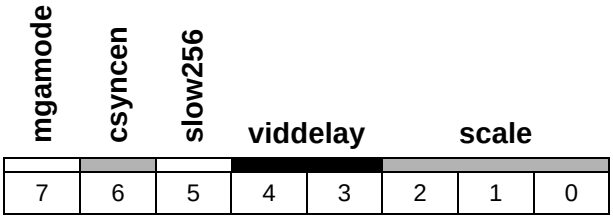
linecomp <7>

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 3-133.

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

viddelay <4:3> Specifies the delay between the CRTC signals and the delayed external signals. The number of delays to be added to the signal depends on the product's configuration:

viddelay	*Configuration*
00	4 MB board
01	2 MB board
1x	8 to 16 MB board

slow256 <5> 256 color mode acceleration disable.

- 0: Direct frame buffer accesses are accelerated in VGA mode 13.
- 1: VGA Mode 13 direct frame buffer access acceleration is disabled. Unless otherwise specified, this bit should always be '0'.

syncen <6> Composite sync enable.

Generates a composite sync signal on the VHSYNC/ pin.

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

mgamode <7> MGA mode enable.

- 0: Select VGA compatibility mode. In this mode, VGA data is output on the MDQ<63:56> bus. The memory address counter clock will be selected by the **count2** (**CRTC17**<3> and **count4** (**CRTC14**<5>) bits. This mode should be used for all VGA modes up to mode 13, and for all Super VGA alpha modes. The VGA port of the RAMDAC should be selected. The load clock that is sent to the RAMDAC is VCLK (or VCLK/2 if mode 13 is selected). When **mgamode** = '0', the MGA frame buffer aperture mapped to **MGABASE2** is unusable.
- 1: Select MGA mode. In this mode, it is the memory serial output stream that is sent directly to the RAMDAC. The main port of the RAMDAC should be selected. The memory address counter is clocked with the VCLK pin divided by 2. The load clock sent to the RAMDAC is VCLK. This mode should be used for all Super VGA graphics modes and all accelerated graphics modes.

Index **crtcextx** = 04h

Reset Value 0000 0000b

page

7	6	5	4	3	2	1	0

page <7:0> 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 megabytes 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<7:1>, CPUA<16:0>	128K window
'01'	CRTCEXT4<7:0>, CPUA<15:0>	64K window
'1X'	Undefined	Window is too small

Index **crtcextx** = 05h

Reset Value 0000 0000b

hvidmid

7	6	5	4	3	2	1	0

hvidmid <7:0> Horizontal video half count.

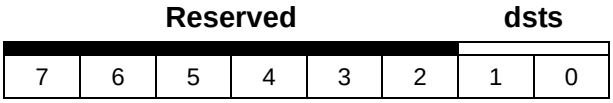
This register specifies the horizontal count at which the vertical counter should be clocked when in interlaced display in field 1. This register is only used in interlaced mode. The value to program is:

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

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

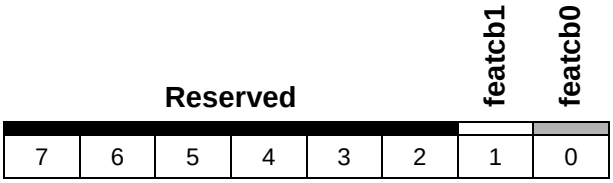
Reserved <7:2> This field returns zeroes when read.

Data read 03C7h will not be transmitted to the RAMDAC, and the contents of the DACSTAT register will be presented on the PCI bus. Writes to 03C7h will be transmitted to the RAMDAC.

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.

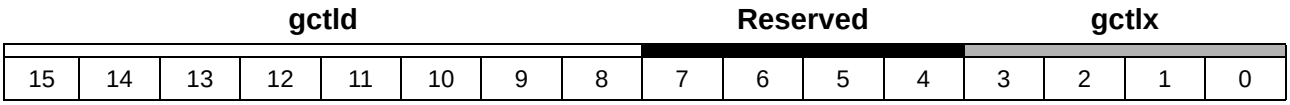
featcb1<1> Feature control bit 1. VGA. General read/write bit.

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



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	**GCTL7**	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.

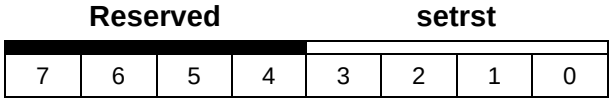
gctld <15:8> Graphics controller data register.

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.

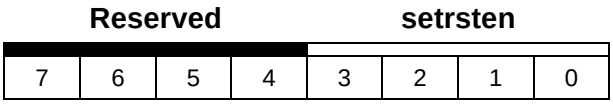
These bits allow setting or resetting byte values in the four video maps:
- 1: Set the byte, assuming the corresponding set/reset enable bit is ‘1’.
- 0: Reset the byte, assuming the corresponding set/reset enable bit is ‘0’.

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.

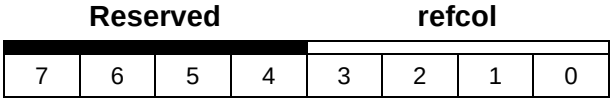
When a set/reset plane is enabled (the corresponding bit is ‘1’) and the write mode is 0 (**wrmode** (GCTL5<1:0>) = 00), the value written to all eight bits of that plane represents the contents of the set/reset register. Otherwise, the rotated CPU data is used.

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



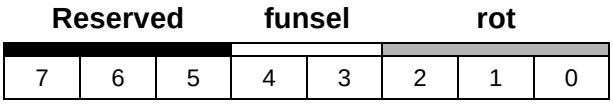
refcol <3:0> Reference color. VGA.

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 3-156) 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 3-157) register to select which pixel is written.

funsel <4:3> Function select. VGA.

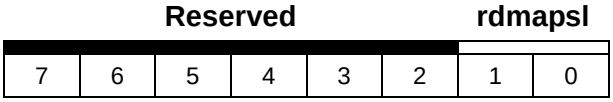
Specifies one of four logical operations between the video memory data latches and any data (the source depends on the write mode).

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



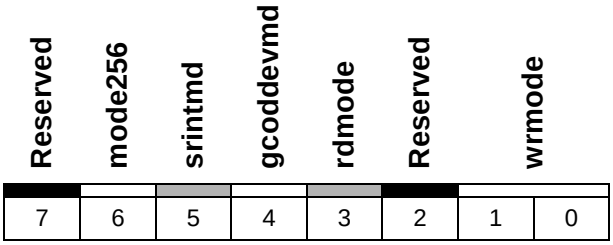
rdmapsl <1:0> Read map select. VGA.

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 3-157) 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-numberedmap
  - Serial data with the even-numbered bits from both maps in the even-numbered maps.

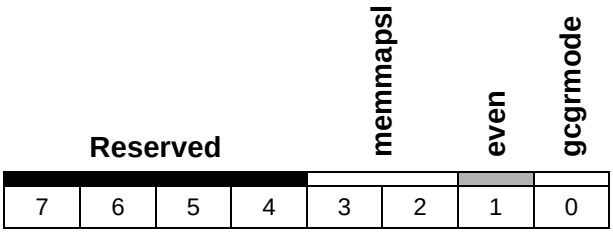
mode256<6> 256-color mode. VGA.
- 0: The loading of the shift registers is controlled by the **srintmd** field.
- 1: The shift registers are loaded in a manner which supports 256-color mode.

Reserved **<2> <7>**

Reserved. When writing to this register, the bits in these fields must be set to '0'. These fields return ‘0’s 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 3-160).

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

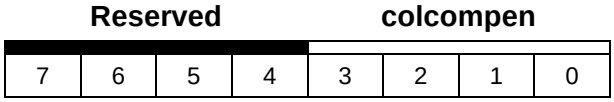
These bits select where the video memory is mapped, as shown below:

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>

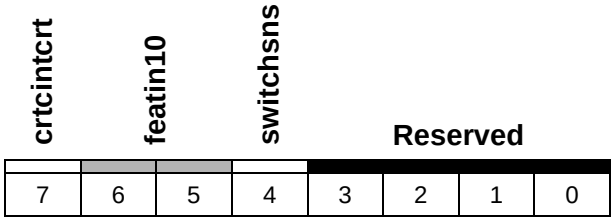
Data write mask for pixels 7 to 0. VGA.

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.

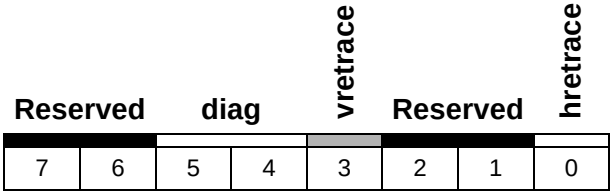
- 0: Vertical retrace interrupt is cleared.
- 1: Vertical retrace interrupt is pending.

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

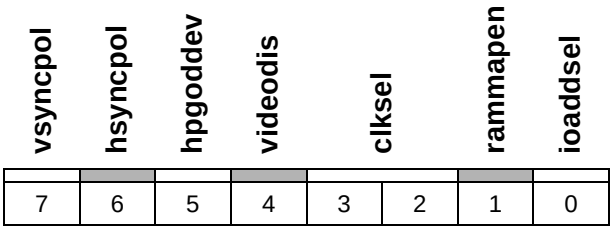
Reserved <2:1> <7:6>

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

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

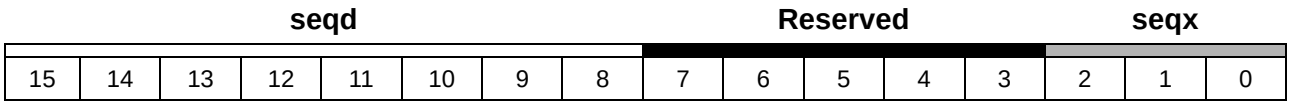
vsyncpol <7> Vertical sync polarity. VGA/MGA.

- 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.

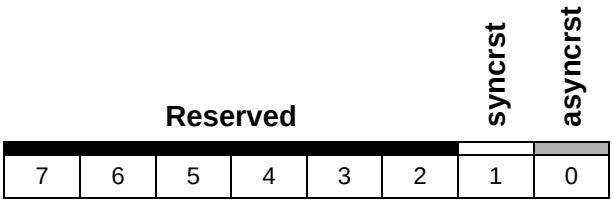
seqd <15:8> Sequencer data register.

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.

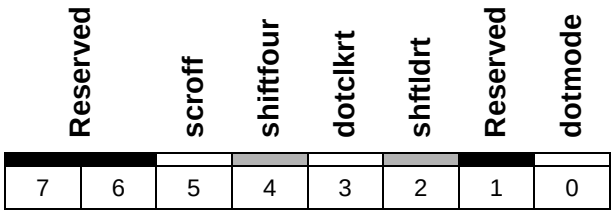
syncrst <1> Synchronous reset. VGA.

- 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-2164W 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 VCLK pin.
- 1: The dot clock rate is slowed to one-half the clock at the VCLK pin. The character clock and shift/load signals are also slowed to half their normal speed.

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.

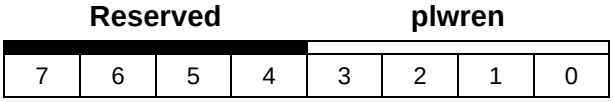
- 0: Normal video operation.
- 1: Turns off the video, and maximum memory bandwidth is assigned to the system. The display is blanked, however, all sync pulses are generated normally.

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

Reserved. When writing to this register, the bits in these fields must be set to '0'. These fields return ‘0’s 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 kilobytes of Map 2
'001'	1	3rd 8 kilobytes of Map 2
'010'	2	5th 8 kilobytes of Map 2
'011'	3	7th 8 kilobytes of Map 2
'100'	4	2nd 8 kilobytes of Map 2
'101'	5	4th 8 kilobytes of Map 2
'110'	6	6th 8 kilobytes of Map 2
'111'	7	8th 8 kilobytes 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:

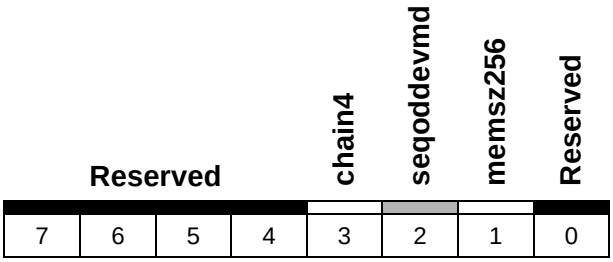
mapasel	*Map#*	*Map location*
'000'	0	1st 8 kilobytes of Map 2
'001'	1	3rd 8 kilobytes of Map 2
'010'	2	5th 8 kilobytes of Map 2
'011'	3	7th 8 kilobytes of Map 2
'100'	4	2nd 8 kilobytes of Map 2
'101'	5	4th 8 kilobytes of Map 2
'110'	6	6th 8 kilobytes of Map 2
'111'	7	8th 8 kilobytes of Map 2

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 ‘0’s when read.

Chapter 4: Programmer's Specification

4.1 HOST Interface

4.1.1 Introduction

The MGA-2164W-PCI chip interacts directly with the PCI interface, while the MGA-2164W-AGP chip deals directly with the AGP interface. Each interface has been optimized to improve the performance of the graphics subsystem. As a result, 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 3 with the 'FIFO' attribute. The BFIFO is also used for the data by ILOAD operations.

MIFIFO This is an 8-entry FIFO which is used for direct frame buffer VGA/MGA accesses, for accesses to the DAC, and for accesses to external devices.

MOFIFO This is a 4-entry FIFO which is used for IDUMP operations.

CACHE This is a 4-location cache, which is used for direct frame buffer VGA/MGA read accesses, for accesses to the DAC, or for accesses to external devices.

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

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

(1) DRWI: Drawing Register Window Indirect access

4.1.2 PCI Retry Handling

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

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

In certain situations, retries can increase efficiency and simplify the software. 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 only a few retries. In this case, you must check the BFIFO flag (thereby limiting the number of retries) to prevent a system error.

4.1.3 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 2.1.3 on page 2-4 for the exact addresses.

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

4.1.4 PCI Target-Abort Generation

The MGA-2164W 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-2164W.

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 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’.

4.1.5 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-2164W’s VGA or DAC.

4.1.6 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). As a result, it is the software's responsibility to invalidate the cache, using one of the methods listed above, whenever any bit is written that affects the frame buffer configuration or contents. The **CACHEFLUSH** register can be used, since it occupies a reserved address in the memory mapped VGA register space (**MGABASE1** + 1FFFh).

4.1.7 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-2164W to correct this situation.

The exact action that will be taken depends on the data size (the MGA-2164W must be aware of the data size when processing Big-Endian data). The data size depends on the location of the data (the specific memory space), and the pixel size (when the data is a pixel).

There are six distinct memory spaces:

1. Configuration space.
2. Boot space (EPROM).
3. I/O space.
4. Register space.
5. Frame buffer space.
6. ILOAD and IDUMP space.

Configuration space

Each register in the configuration space is 32 bits, and should be addressed using dword accesses. For these registers, no byte swapping is done, and bytes will appear in different positions, depending on the endian mode of the host processor. Keep in mind that the MGA-2164W 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-2164W 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-2164W 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-2164W I/O mapping on non x86 platforms.

Register Space

The majority of the data in the register space is 32 bits wide, with a few exceptions:

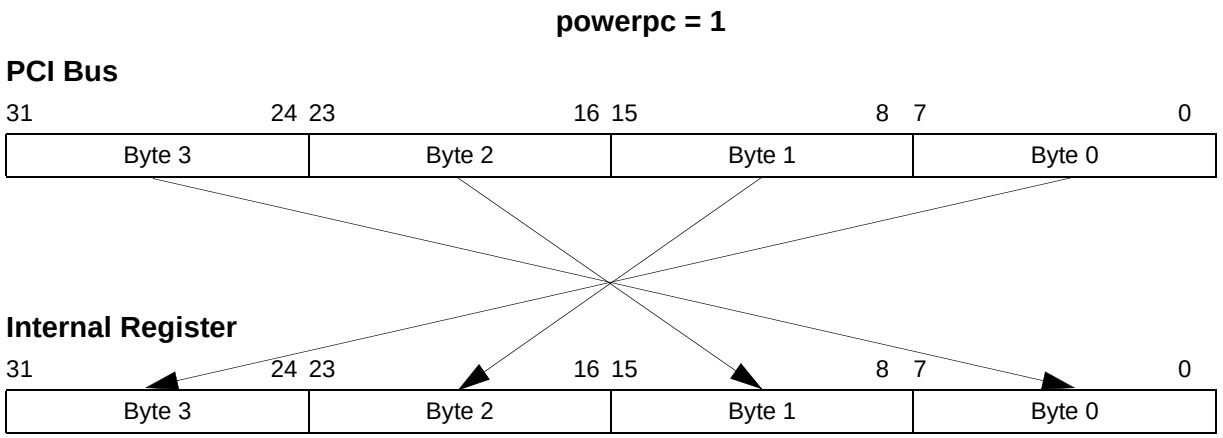
- The VGA compatibility section. Data in this section is 8 bits wide.
- The DAC. Data in this section is 8 bits wide.
- External devices. In this case, the width of the data cannot be known in advance.

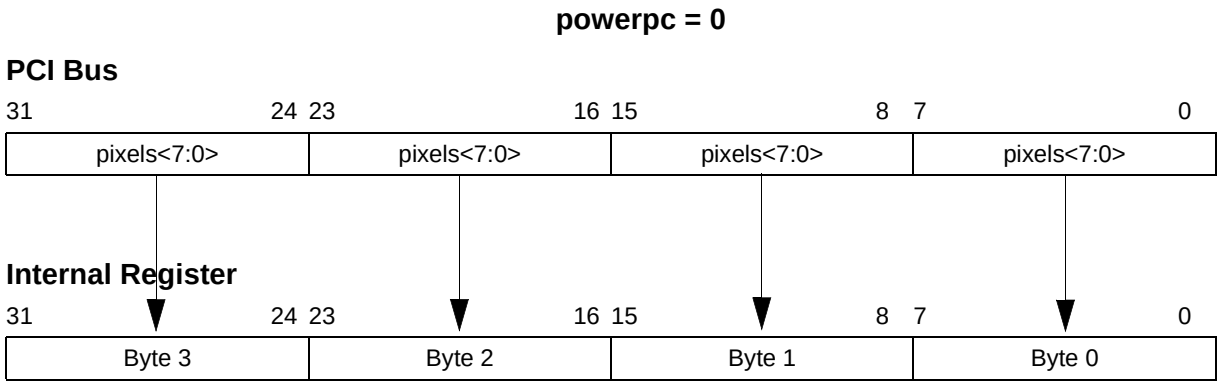
Byte swapping for Big Endian processors can be enabled in the register space by setting the **OPTION** configuration space register's **powerpc** bit to 1.

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

Note: 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.



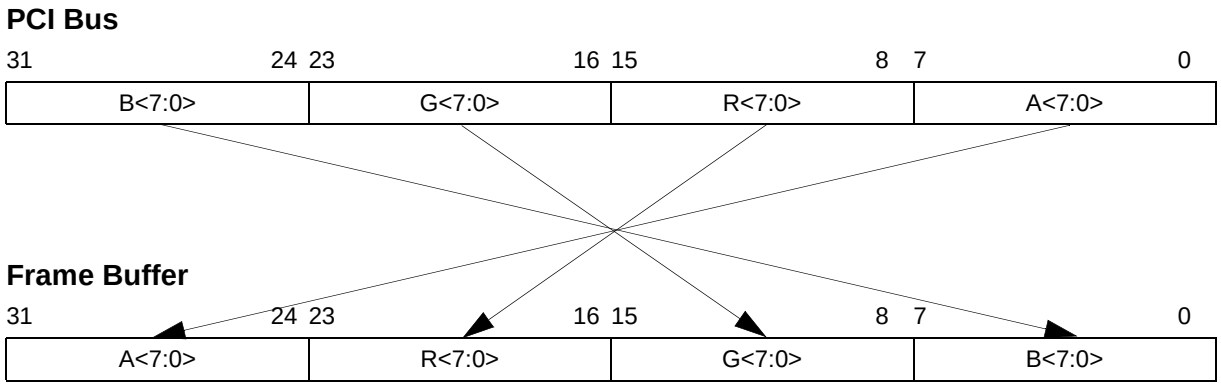


Frame Buffer Space

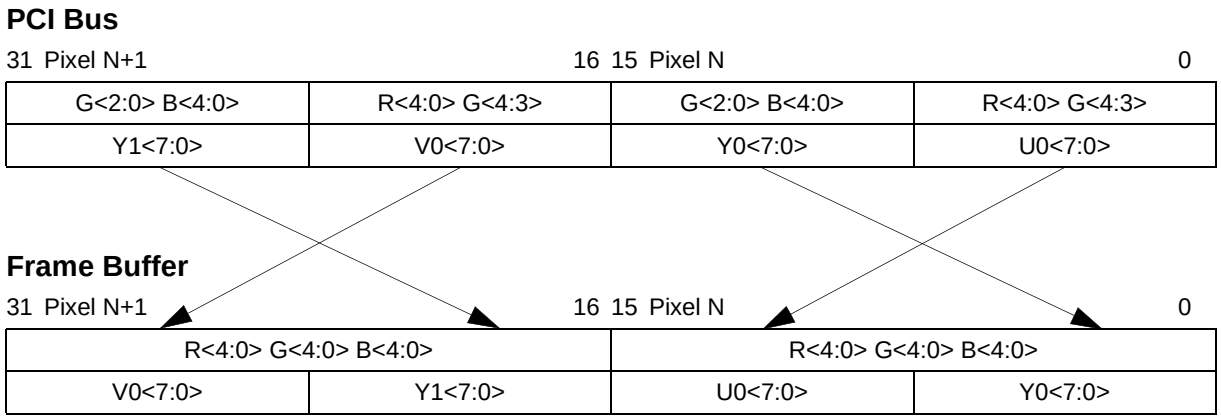
The frame buffer is organized in Little Endian format, and byte swapping depends on the size of the pixel. As usual, addresses are not modified.

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.

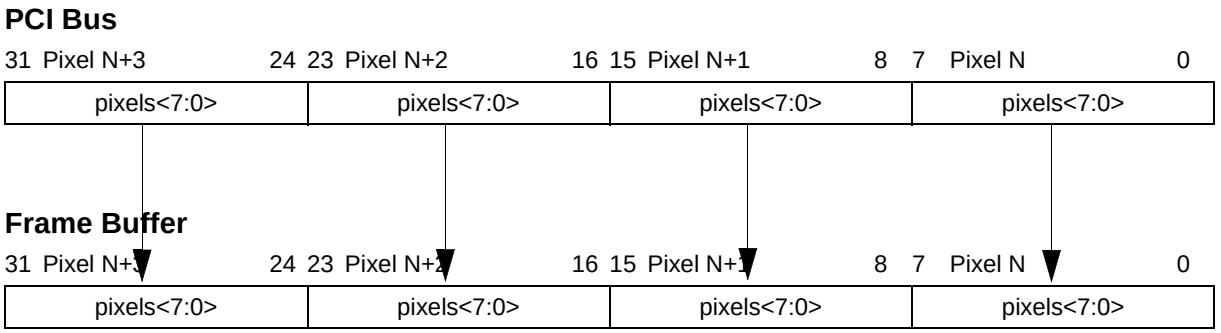
32 bits/pixel, dirDataSiz = 10



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'.

4.1.8 Host Pixel Format

There are several ways to access the frame buffer. The pixel format used by the host depends on the following:

- The current frame buffer's data format
- The access method
- The processor type (Big Endian or Little Endian)
- The control bits which select the type of byte swapping

The supported data formats are listed below, and are shown from the processor's perspective. The supported formats for direct frame buffer access, ILOAD, and IDUMP are explained in their respective sections of this chapter.

Note: For Big Endian processors, these tables assume that the CPU-to-PCI bridge respects the *PCI Specification*, which states that byte address coherency must be preserved. This is the case for PREP systems and for Macintosh computers.

Pixel Format (From the Processor's Perspective)

8-bit A Little endian 8-bit (see the **powerpc** field of **OPTION**) is used in ILOAD and IDUMP operations. Refer to Table 4-4 on page 4-233 and Table 4-12 on page 4-245.

	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**) is used in ILOAD and IDUMP operations. Refer to Table 4-4 on page 4-233 and Table 4-12 on page 4-245.

	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**) is used in ILOAD and IDUMP operations. Refer to Table 4-4 on page 4-233 and Table 4-12 on page 4-245.

	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**) is used in ILOAD and IDUMP operations. Refer to Table 4-4 on page 4-233 and Table 4-12 on page 4-245.

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

32-bit A 32-bit RGB, used in ILOAD and IDUMP operations. Refer to Table 4-4 on page 4-233, Table 4-6 on page 4-235, and Table 4-12 on page 4-245.

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

32-bit B 32-bit BGR used in ILOAD and IDUMP operations. Refer to Table 4-4 on page 4-233, Table 4-6 on page 4-235, and Table 4-12 on page 4-245.

	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 4-7 on page 4-235.

	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 4-7 on page 4-235.

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

24-bit A 24-bit RGB packed pixel, used in ILOAD and IDUMP operations. Refer to Table 4-4 on page 4-233, Table 4-6 on page 4-235, and Table 4-12 on page 4-245.

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

24-bit B 24-bit BGR packed pixel, used in ILOAD and IDUMP operations. Refer to Table 4-4 on page 4-233, Table 4-6 on page 4-235, and Table 4-12 on page 4-245.

	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 4-4 on page 4-233 and Table 4-6 on page 4-235.

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 4-4 on page 4-233 and Table 4-6 on page 4-235.

	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 4-4 on page 4-233 and Table 4-6 on page 4-235.

	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 4-4 on page 4-233 and Table 4-6 on page 4-235.

	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 4-7 on page 4-235.

	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 4-7 on page 4-235.

	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 4-7 on page 4-235.

	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 4-7 on page 4-235.

	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 4-5 on page 4-234.

	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 4-5 on page 4-234.

	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 4-5 on page 4-234.

	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	:

4.2 Memory Interface

4.2.1 Frame Buffer Organization

The MGA-2164W supports a total of 16 megabytes of WRAM memory divided in eight 2 MByte banks. It is possible to connect a 64 or 128 bits RAMDAC. The actual quantity of visible memory is a function of the RAMDAC used. The first 2 Mbyte bank is always visible and the last four 2 Mbyte banks are never visible.

There are three different frame buffer organizations, described below:

- VGA Mode
- Power Graphic Mode (single frame buffer modes)
- Power Graphic Mode (split frame buffer modes)

Split frame buffer modes are implemented by allocated a complete bank of memory per buffer. One bank of memory will drive the lower part of the serial bus and the other bank will drive the upper part of the serial bus. Obviously, split frame buffer modes demand at least 2 memory banks and a RAMDAC capable of understanding that the serial bus is now made of 2 video streams. The following table summarizes the chip capabilities:

Table 4-1: Chip Capabilities

RAMDAC serial port	*Total memory possible*	*Visible memory*	*Split mode capability*
64	2M	2M	NONE
	4M	4M	2*2M
	8M	8M	2*2M
	10,12,14,16M	8M	2*2M
128	2M	2M	NONE
	4M	4M	2*2M
	8M	8M	2*4M
	10,12,14,16M	8M	2*4M

4.2.1.1 Supported Resolutions

In Power Graphic Mode, the resolution depends on the amount of available memory. The following table shows the memory requirements for each standard VESA resolution and pixel depth.

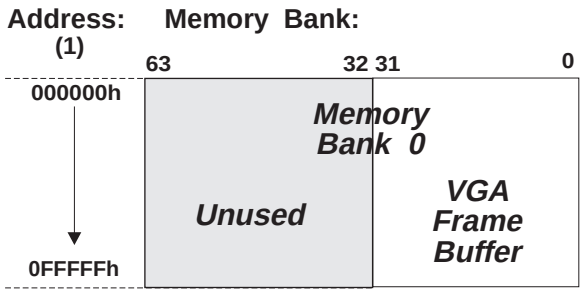
	Single Frame Buffer Mode				*Single Z Buffer*							
	No Z				*Z 16 bits*				*Z 32 bits*			
Resolution	*8-bit*	*16-bit*	*24-bit*	*32-bit*	*8-bit*	*16-bit*	*24-bit*	*32-bit*	*8-bit*	*16-bit*	*24-bit*	*32-bit*
640 x 480	2M	2M	2M	2M	2M	2M	-	2M	2M	2M	-	4M
720 x 480	2M	2M	2M	2M	2M	2M	-	2M	2M	2M	-	4M
800 x 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 resolution is also a function of the DAC bandwidth. The following table demonstrates the minimum requirement for a given resolution:

Resolution @ 85Hz	*Pixel Clock MHz*	*Pixel Width*			
		8	*16*	*24*	*32*
640x480	36	DAC64	DAC64	DAC64	DAC64
720x400	35.5	DAC64	DAC64	DAC64	DAC64
800x600	56.25	DAC64	DAC64	DAC64	DAC64
1024x768	94.50	DAC64	DAC64	DAC64	DAC64
1152x864	121.5	DAC64	DAC64	DAC64	DAC64
1280x1024	157.5	DAC64	DAC64	DAC64	DAC128
1600x1200	229.5	DAC64	DAC64	DAC128	DAC128

4.2.1.2 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.

4.2.1.3 Power Graphic Mode

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

Note: All addresses are hexadecimal and are byte addresses.

Figure 4-1: memconfig [1:0] = 00

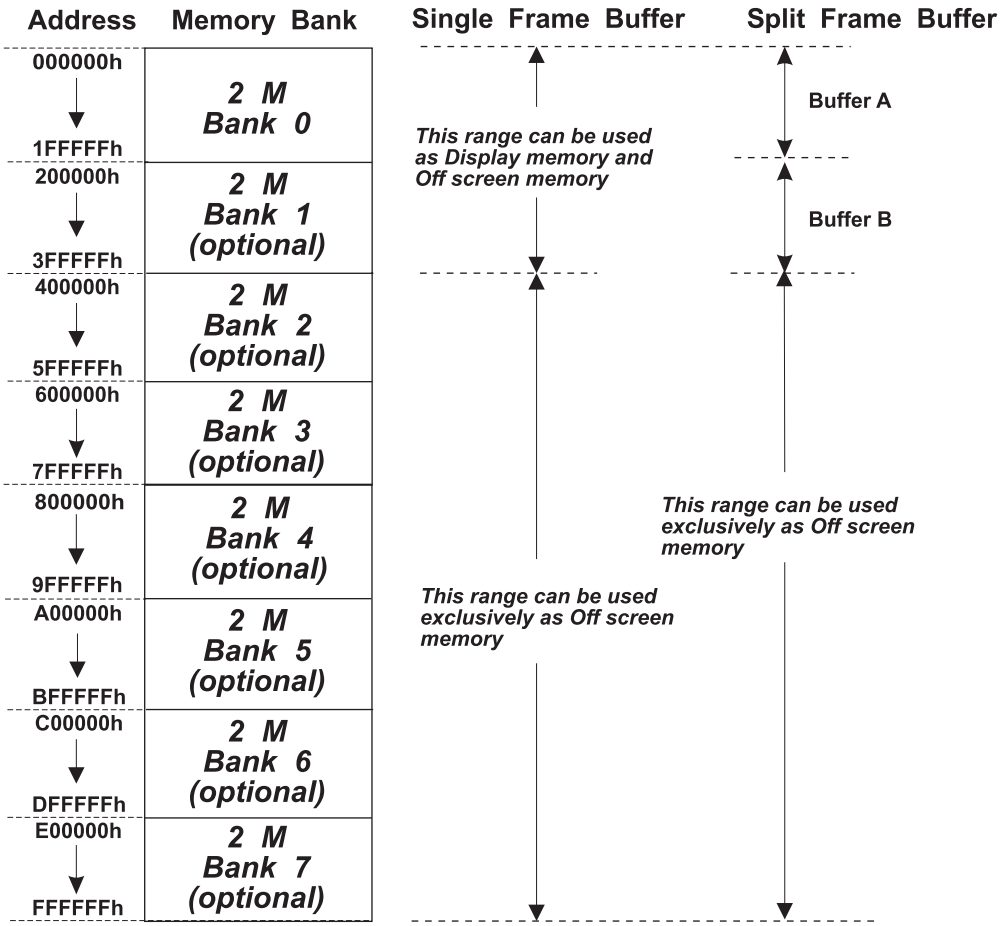


Figure 4-2: memconfig [1:0] = 01

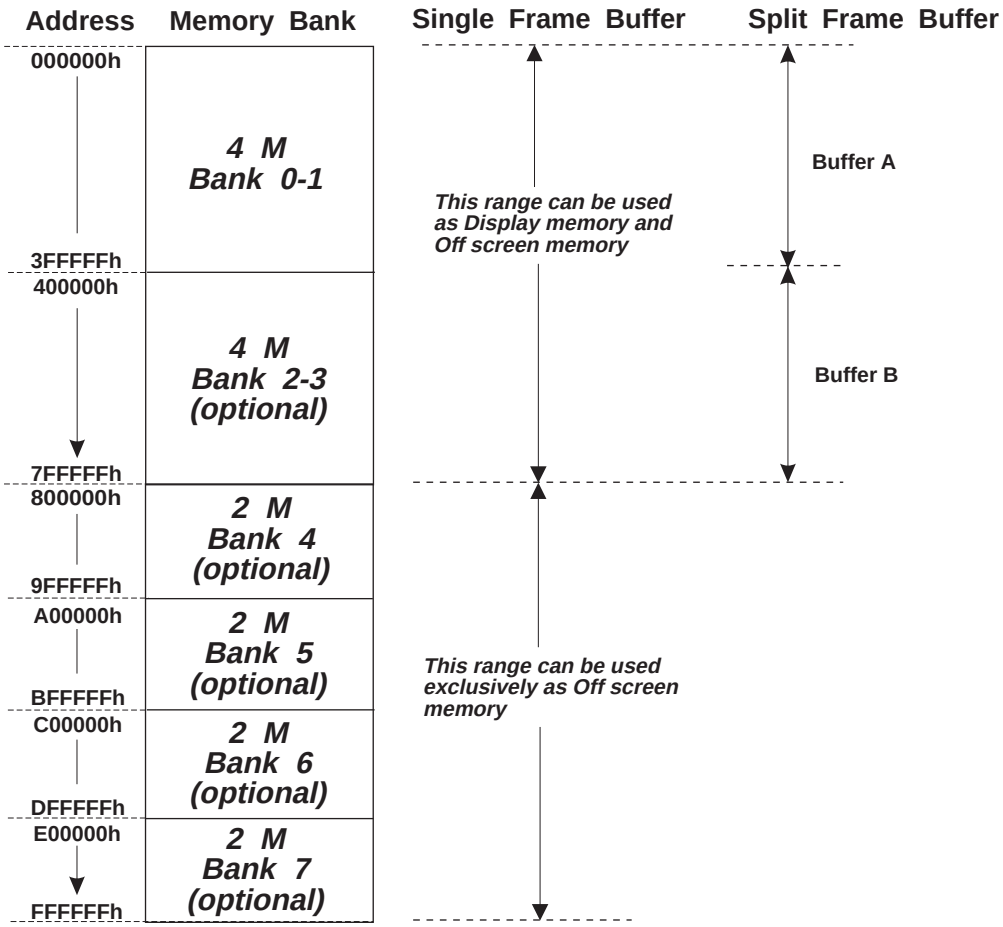
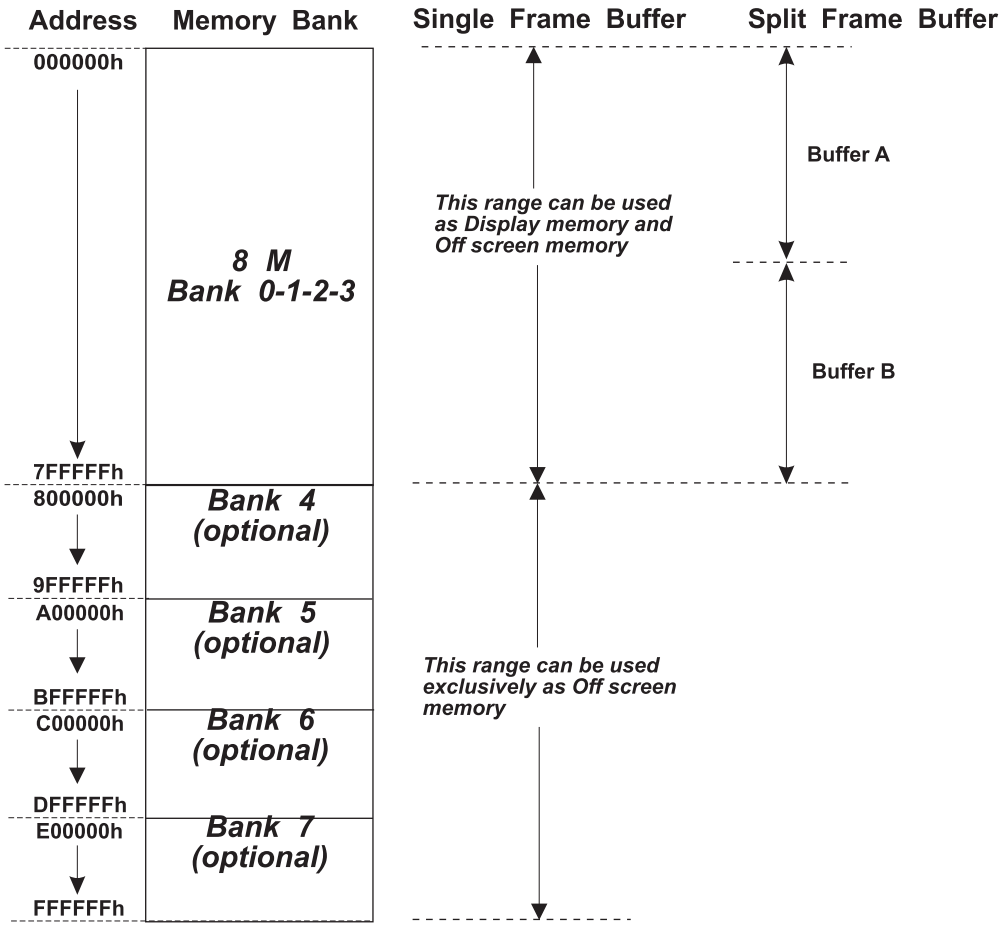


Figure 4-3: memconfig [1:0] = 10

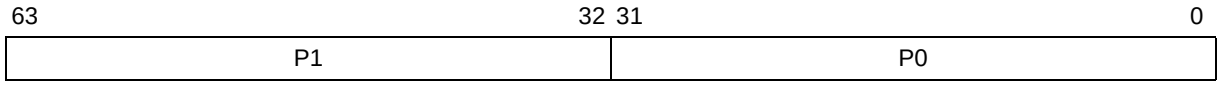


4.2.2 Pixel Format

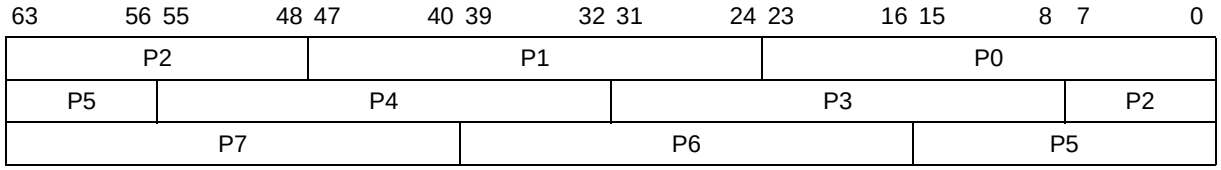
The slice is 64 bits long and is organized as follows. In all cases, the least significant bit is 0. The Alpha part of the color is the section of a pixel that is not used to drive the DAC. Note that the data is always true color, but in 8 bit/pixel formats pseudo color can be used when shading is not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

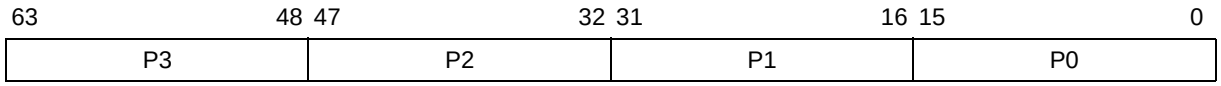
32 bits/pixel



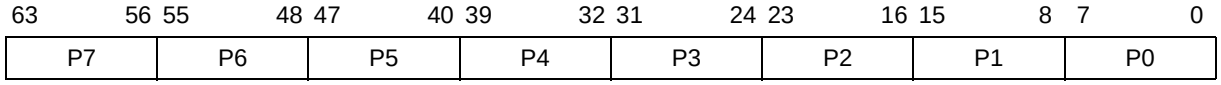
24 bits/pixel



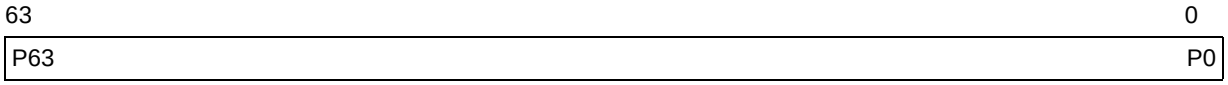
16 bits/pixel



8 bits/pixel

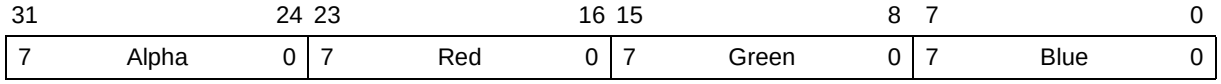


Monochrome

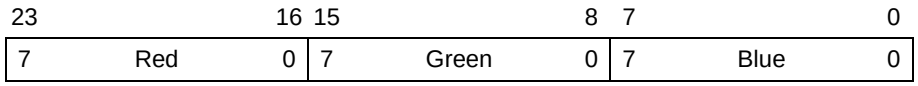


For each of these modes, the pixels are arranged as follows:

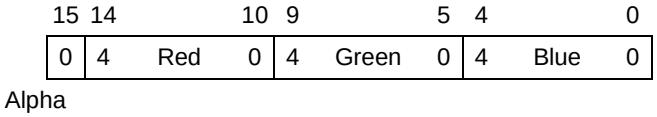
32 bits/pixel



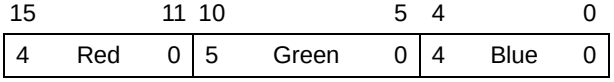
24 bits/pixel



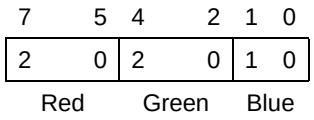
16 bits/pixel (5:5:5)



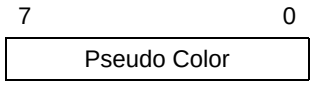
16 bits/pixel (5:6:5)



8 bits/pixel



8 bits/pixel



4.3 Chip Configuration and Initialization

4.3.1 Reset

The MGA-2164W 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 3 to determine which bits are hard reset.
- All state machines are reset (possibly with termination of the current operation).
- FIFOs will be emptied, and the cache will be invalidated.
- A hard reset will activate the local bus reset (EXTRST/) in order to reset expansion devices when required. The EXTRST/ signal is synchronous on PCLK.

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

Strap Name	*Pins*	*Description*
biosen	MDQ<6>	Indicates whether a ROM is installed (‘1’) or not (‘0’). The biosen strap also controls the **biosen** field of the **OPTION** register.
pid<4:0>	MDQ<4:0>	User-defined. Undefined bits should be strapped ***high*** by default. These bits are loaded into the **productid** field of the **OPTION** register.
vgaboot	MDQ<5>	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’).
Reserved	MDQ <7>	Must be pulled-down by a 10k resistor.

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 3-76 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.

4.3.2 Operations After Hard Reset

- After a hard reset, the chip will be in a VGA-compatible state.
- Register bits that do not have a reset value will wake up with unknown values.
- Frame buffer memory refreshing is not running.

4.3.3 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 essential to the correct behavior of the chip:

Name	*Reset Value*	*Description*
eepromwt	'0'	To be set to '1' if a FLASH ROM is used, and writes are to be done to the ROM.
powerpc	'0'	To be set to '1' to support Big Endian processor accesses.
rfhcnt	'0000'	The refresh counter defines the rate of MGA memory refresh. For a typical 62.5 MHz GCLK, a value of 8 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.

4.3.3.1 WRAM Reset Sequence

In order to properly initialize the WRAM, the following sequence must be followed at power-up after a hard reset.

Step 1. Initialize the clock generator to the proper gclk value.

Step 2. Program the graphic pre-scaler (the **OPTION** register's **nogscale** field).

Step 3. Set the **scroff** blanking field (**SEQ1**<5>) to prevent any transfer.

Step 4. Initialize the **CRTC** (See 'CRTC Programming' on page 4-247).

Step 5. Set the **softreset** bit of the **RST** register.

Step 6. Wait a minimum of 200 µs.

Step 7. Clear the **softreset** bit.

Step 8. Program the refresh register (the **OPTION** register's rfhcnt field).

Step 9. Wait a minimum of 200 µs. (allowing the performance of more than eight refresh cycles).

Step 10. Wait for the vertical retrace.

Step 11. Enable the video.

Step 12. Wait for the next vertical retrace.

Step 13. Set the **memreset** bit of the **MACCESS** register.

Step 14. Wait 1 µs

Step 15. Wait until the drawing engine is idle.

Step 16. From this point on, the WRAM is initialized and the drawing engine can be accessed. The drawing engine is in VGA Mode (**mgamode** = 0).

>>> *Soft Reset Sequence*

Use this sequence whenever the **RST** register softreset bit is used.

Step 1. Set the **softreset** bit of the **RST** register

Step 2. Wait 1µs

Step 3. Clear the **softreset** bit

Step 4. Set the memreset bit of the **MACCESS** register

Step 5. Wait 1µs

Step 6. Wait until the drawing engine is idle.

4.3.4 Operation Mode Selection

The MGA-2164W provides three different display modes: text (VGA or SVGA), VGA graphics, and SVGA graphics. Table 4-2 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 4-2: 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 4-2: 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(2)	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) Only possible with a frame buffer of 8 megabytes or more.

(2) Only possible with a frame buffer of 4 megabytes or more

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 4.6).
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.

4.4 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-2164W 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.

4.5 Drawing in Power Graphic Mode

This section explains how to program the MGA-2164W'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 3.
- Pseudo-DMA. In this case, the addresses of the individual registers to be accessed are embedded in the data stream. Pseudo-DMA can be used in four different ways:
  - The General Purpose Pseudo-DMA mode can used with any command.
  - The DMA Vector Write mode is specifically dedicated to polyline operations.
  - ILOAD and IDUMP operations always use Pseudo-DMA transfers for exchanging data with the frame buffer.

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

4.5.1 Drawing Register Initialization Using General Purpose Pseudo-DMA

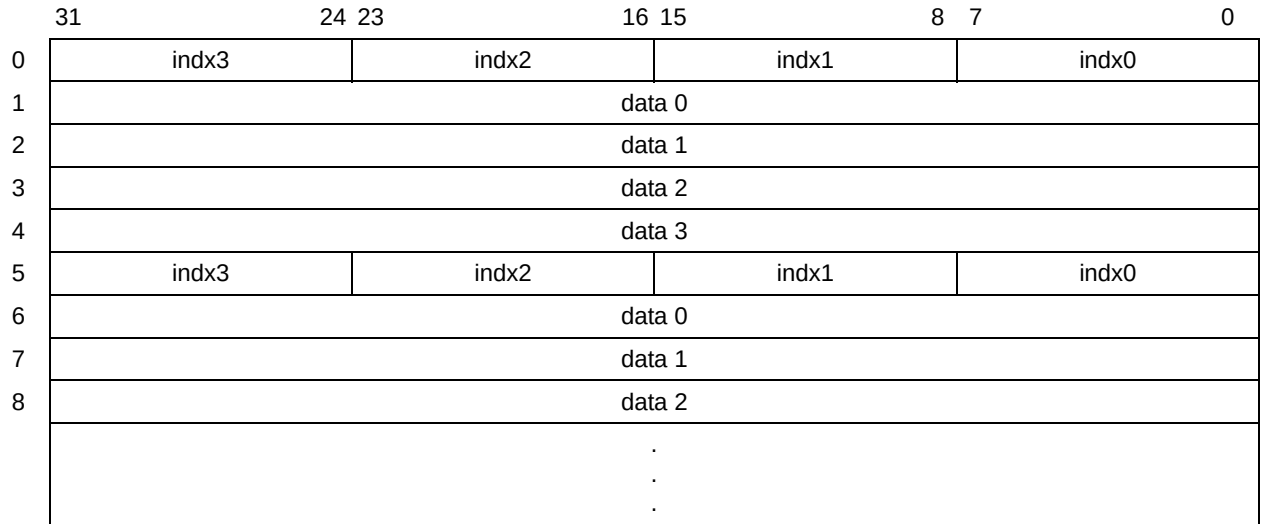
The general purpose Pseudo-DMA operations are performed through the DMAWIN aperture in the MGA control register space, or in the 8 MByte Pseudo-DMA window. It is recommended that host CPU instructions be used in such a way that each transfer increments the address. This way, the PCI bridge can proceed 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 2-3 on page 2-4). 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 3 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



4.5.2 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 4.5.4.2.
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 3-55).

   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.

4.5.3 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) and Z precision (16 or 32 bits)	Some limitations apply
CXBNDRY	Left/right clipping limits	Can use **CXLEFT** and **CXRIGHT** instead
YTOP	Top clipping limit	
YBOT	Bottom clipping limit	
PLNWT	Plane write mask	
ZORG	Z origin position	Only required for depth operations

Regarding future product compatibility: *It is recommended that the last register programmed be in the range 1d00h to 1DBFh.*

4.5.4 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 4.5.3 and subsection 4.5.4.1 first. Also, ***the last register you program must be accessed in the 1D00h-1DFFh range in order to start the drawing engine***.

4.5.4.1 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).

> ***Note:*** 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

4.5.4.2 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	

4.5.4.3 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	+ + +	+ + + +

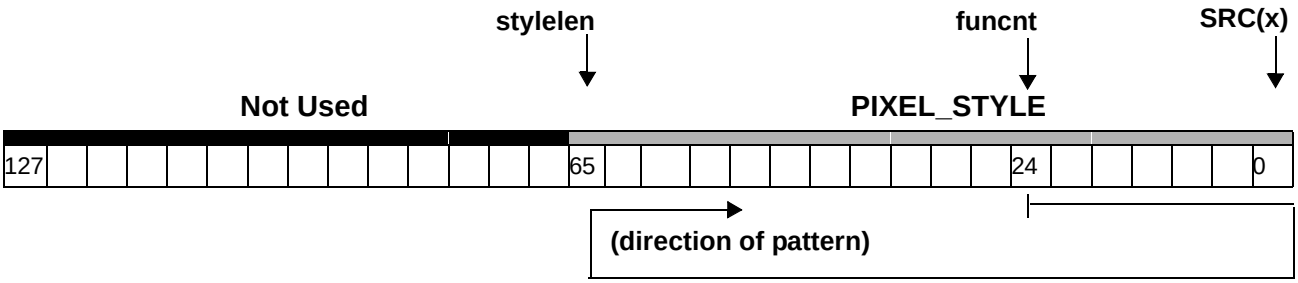
- **bop**: uses any Boolean operation if **atype** is RSTR; if **atype** is RPL, **bop** must be loaded with '0000', '0011', '1100', or '1111'
- **atype**: can only be RPL or RSTR
- **opcod**: must be LINE_OPEN, LINE_CLOSE, AUTOLINE_OPEN, or AUTOLINE_CLOSE

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	

Note: 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’

4.5.4.4 Lines with Depth

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	# # # #	1 1 0 0	0	0	0	0	0	# # #	0	+ + +	+ + + +

- **atype**: must be either ZI or I
- **opcod**: must be set to LINE_OPEN, LINE_CLOSE, AUTOLINE_OPEN, or AUTOLINE_CLOSE

Register	*Function*	*Comment / Alternate Function*
DR0 (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.

4.5.4.5 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 4.5.3) must be set.
- Solid lines can be selected by initializing the registers as explained in subsection 4.5.4.2. Lines with linestyle can be selected by initializing the registers as explained in subsection 4.5.4.3. 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.

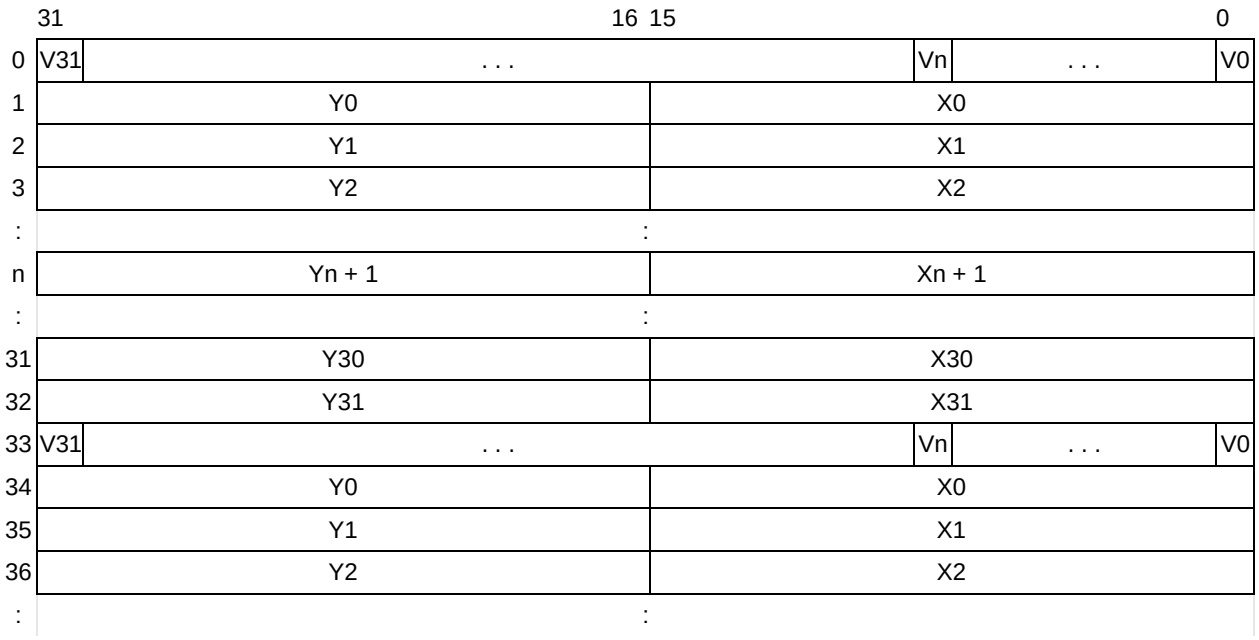
The first double word that is transferred is loaded into the Address Generator. This double word contains one bit of ‘address select’ for each of the next 32 vector vertices to be sent to the drawing registers. These 32 bits are called the vector tags. The next 32 double word transfers contain the xy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



4.5.5 Trapezoid / Rectangle Fill Programming

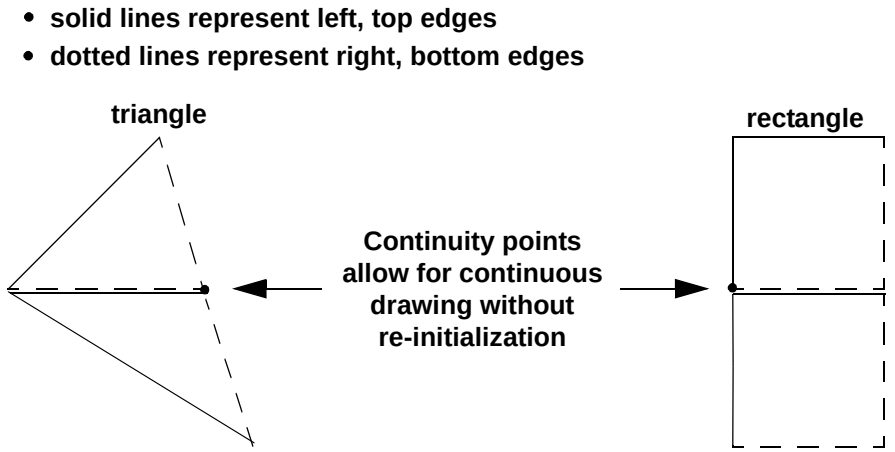
The following subsections list the registers that must be specifically programmed for constant and Gouraud shaded, patterned, and textured trapezoids, including rectangle and span line fills. Remember to program the registers listed in section 4.5.3 and in the tables in subsection 4.5.5.1 first. Also, ***the last register you program must be accessed in the 1D00h-1DFFh range in order to start the drawing engine***.

4.5.5.1 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 4-4: 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**)

4.5.5.2 Constant Shaded Trapezoids / Rectangle Fills

DWGCTL:

	Reserved	transc	pattern	bltmod				Reserved	trans				bop				Reserved	shftzero	sgnzero	arzero	solid	zmode				linear	atype			opcod			
TRAP	0	1	0	0	0	0	0	0	+	+	+	+	+	+	+	+	0	1	0	0	1	0	0	0	0	+	+	+	0	1	0	0
RECT	0	1	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 WRAM. It should be used whenever possible for the fastest performance, although certain restrictions apply (see the **atype** field of the **DWGCTL** register on page 3-55).

4.5.5.3 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

- **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: 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

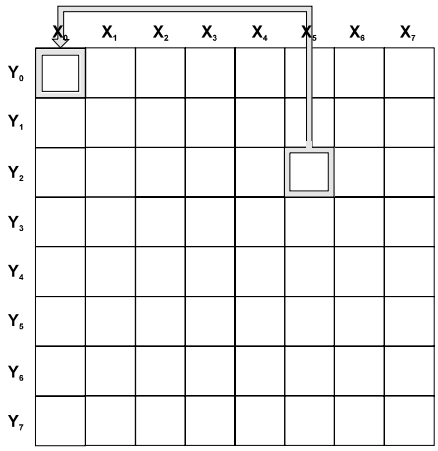
or

- **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.

Note: If **atype** is BLK the WRAM feature of blockmode is used (single color block mode when **transc** = 1, or dual color block mode when **transc** = 0). When **memconfig** = 10, the case **atype** = BLK cannot be used.

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 \ X	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 \ X	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 \ X	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 \ X	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’

4.5.5.4 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: The **MACCESS** register's **pwidth** field must not be set to 24 bits per pixel (PW24) when drawing Gouraud shaded trapezoids.

4.5.5.5 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: The **MACCESS** register's **pwidth** field must not be set to 24 bits per pixel (PW24) when drawing this type of trapezoid.

Note: 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.

Note: The procedure for ILOAD (image load: Host -> RAM) operations is described in 'ILOAD Programming' on page 4-230.

4.5.6 Bitblt Programming

The following subsections list the registers that must be specifically programmed for Bitblt operations. Remember to program the registers listed in section 4.5.3 and subsection 4.5.6.1 first. Also, ***the last register you program must be accessed in the 1D00h-1DFFh range in order to start the drawing engine***.

4.5.6.1 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**)

4.5.6.2 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 the number of pixels in the destination.

4.5.6.3 Two-operand Fast Bitblts

Register	*Function*	*Comment / Alternate Function*
DWGCTL	040A600C	
AR0	Source end address	The last pixel of the first line
AR3	Source start address	
AR5	Source y increment	

➧ ***Note:*** When programming the **AR0**, **AR3**, and **FXBNDRY** registers, the destination and source must be aligned according to the following table:

memconfig	**pwidth**	*Alignment Constraint*
00	PW8	(SRC_START_ADDRESS mod 64) == ((DST_LEFT_BND + Y_LINEAR + yorg) mod 64)
	PW16	(SRC_START_ADDRESS mod 32) == (DST_LEFT_BND mod 32)
	PW24	(SRC_START_ADDRESS mod 64) == ((DST_LEFT_BND + Y_LINEAR + yorg) mod 64)
	PW32	(SRC_START_ADDRESS mod 16) == (DST_LEFT_BND mod 16)
01	PW8	(SRC_START_ADDRESS mod 128) == ((DST_LEFT_BND + Y_LINEAR + yorg) mod 128)
	PW16	(SRC_START_ADDRESS mod 64) == ((DST_LEFT_BND + Y_LINEAR + yorg) mod 64)
	PW24	(SRC_START_ADDRESS mod 128) == ((DST_LEFT_BND + Y_LINEAR + yorg) mod 128)
	PW32	(SRC_START_ADDRESS mod 32) == (DST_LEFT_BND mod 32)

➧ ***Note:*** Because fast bitblt is a WRAM internal feature, the source and destination must reside in the same memory chip.

➧ ***Note:*** The Fast bitblt function cannot be used when **memconfig** = 10

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

4.5.6.4 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'

Note: 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 4-218; color patterning is performed in a similar manner to monochrome patterning (except that the **SHIFT** register is not used for pinning).

Note: 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.

Note: 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

4.5.6.5 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

- **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: 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.

Note: If **atype** is BLK the WRAM feature of block mode is used (single color block mode when **transc** = 1, and dual color block mode when **transc** = 0). When **memconfig** = 10, the case where **atype** = BLK cannot be used.

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

4.5.6.6 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 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	

Note: For **MACCESS** the planar bitblts are **not** supported with 24 bits/pixel (PW24).

4.5.7 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 4.5.3 and subsection 4.5.7.1. Depending on the type of operation you wish to perform, you must also program the registers in subsection 4.5.7.2 or subsection 4.5.7.3.

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). However, if the host transfers ***more pixels*** than expected, the extra pixels will be interpreted by the drawing engine as register accesses.

Note: The ILOAD command must not be used when no data is transferred.

The total number of dwords to be transferred will differ, depending on whether or not the source is linear:

- When the source is ***linear***: the data is padded at the end of the source.

  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 4-3

Table 4-3: 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

4.5.7.1 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**)

4.5.7.2 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 4-4 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 4-184).

Table 4-4: 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

4.5.7.3 ILOAD with Expansion (Character Drawing)

DWGCTL:

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

- **bltmod**: must be set to either BMONOLEF or BMONOWF
- **trans**: if atype is 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: 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 4-5 shows all the valid format combinations. The structure of the buffers to be transferred is defined for each data format (as shown the ‘Pixel Formats’ illustrations starting on page 4-184).

Table 4-5: 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

4.5.8 Scaling Operations

The MGA-2164W 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

4.5.8.1 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 reformatting. There are some restrictions in the data formats that are supported for this operation. Table 4-6 shows all the valid format combinations for ILOAD_SCALE, ILOAD_FILTER, and ILOAD_HIQH. Table 4-7 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 4-184).

Table 4-6: Scaling Supported Formats: ILOAD_SCALE, ILOAD_FILTER, and ILOAD_HIQH

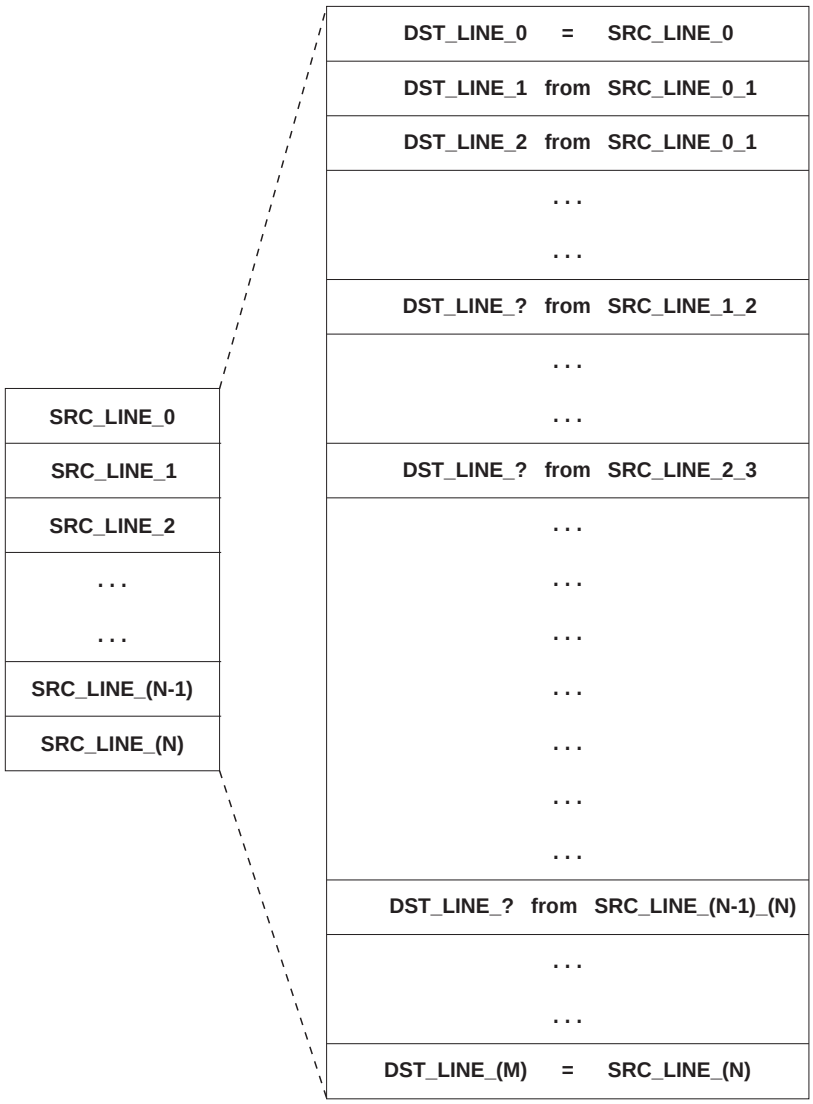
Processor Type	bltmod	dmaDataSiz	*Data Format*
Little endian	BU24RGB	00	24-bit A
	BU24BGR	00	24-bit B
	BU32RGB	00	32-bit A
	BU32BGR	00	32-bit B
	BUYUV	00	YUV A
	BUYUV	01	YUV B
Big endian	BU32RGB	10	32-bit A
	BU32BGR	10	32-bit B
	BUYUV	00	YUV C
	BUYUV	01	YUV D

Table 4-7: 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 4-5: 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):

SRC_BUF_0=

MSB			LSB	
A00	R00	G00	B00	Pixel 0
A01	R01	G01	B01	Pixel 1
A02	R02	G02	B02	Pixel 2
. . .	. . .	. . .	. . .	. . .

SRC_BUF_1=

MSB			LSB	
A10	R10	G10	B10	Pixel 0
A11	R11	G11	B11	Pixel 1
A12	R12	G12	B12	Pixel 2
. . .	. . .	. . .	. . .	. . .

DW to Send to the Chip

MSB			LSB	
G00	R10	G10	B10	DW0
G01	R11	G11	B11	DW1
G02	R12	G12	B12	DW2
. . .	. . .	. . .	. . .	. . .

BU32BGR (32-bit D):

SRC_BUF_0=

MSB			LSB	
A00	B00	G00	R00	Pixel 0
A01	B01	G01	R01	Pixel 1
A02	B02	G02	R02	Pixel 2
. . .	. . .	. . .	. . .	. . .

SRC_BUF_1=

MSB			LSB	
A10	B10	G10	R10	Pixel 0
A11	B11	G11	R11	Pixel 1
A12	B12	G12	R12	Pixel 2
. . .	. . .	. . .	. . .	. . .

DW to Send to the Chip

MSB			LSB	
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
	. . .	. . .	. . .	. . .	. . .

4.5.8.2 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; one 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.

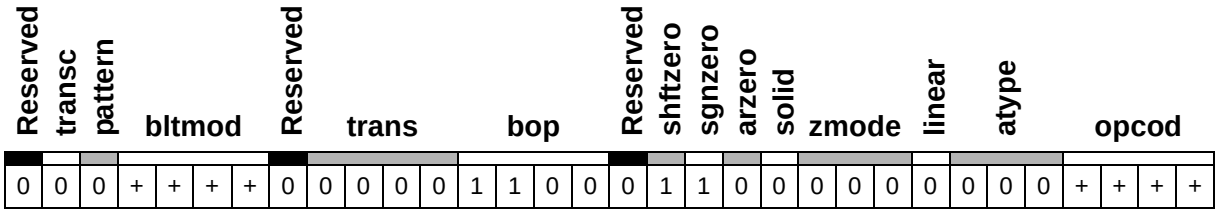
4.5.8.3 Scaling Steps

The following steps must be executed for scaling:

Step 1. Initialize the scaling engine as specified in subsection 4.5.8.4. Also, remember to program the registers listed in section 4.5.3. ***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 4-6 shows the various supported data formats. As with normal ILOAD operations (see the Note on page 4-230), 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 4-8
Nlines:	The number of lines to write
psiz:	The source size, according to Table 4-9

Table 4-8: Source Factor

opcod	bltmod	*Factor*
ILOAD_SCALE ILOAD_FILTER ILOAD_HIQH		1
ILOAD_HIQHV	BUYUV	2
	BU32RGB BU32BGR	1

Table 4-9: Source Size

bltmod	psiz
BUYUV	16
BU24RGB	24
BU24BGR	24
BU32RGB	32
BU32BGR	32

Step 4. For ILOAD_HIQHV, skip this step. Initialize the drawing engine for vertical scaling. The last register you program must be accessed in the 1D00h-1DFFh range in order to start the drawing engine.

Register / Space	*Function*	*Comment / Alternate Function*
LEN	Number of lines	Replicated lines
DWGCTL	040C6008h (BITBLT)	

Step 5. Repeat Steps 2 to 4 until the end of the scaling sequence.

4.5.8.4 Scaling Initialization

ILOAD_SCALE

Register	*Function*	*Comment / Alternate Function*
OPMODE	Data format	A 16-bit access is required to prevent modification of the **dirDataSiz** field (bits 17:16).
AR0	DST_END_ADDRESS - DST_Y_INCREMENT	
AR2	SRC_X_DIMENSION	
AR3	DST_START_ADDRESS - DST_Y_INC	
AR5	DST_Y_INC	Only required if vertical scaling is used
AR6	SRC_X_DIMEN - DST_X_DIMEN	
FXBNDRY	Destination boundary (left and right)	Can use **FXLEFT** and **FXRIGHT**
YDST	Y start position	

ILOAD_FILTER

Register	*Function*	*Comment / Alternate Function*
OPMODE	Data format	A 16-bit access is required to prevent modification of the **dirDataSiz** field (bits 17:16).
AR0	DST_END_ADDRESS - DST_Y_INC	
AR2	(2 * SOURCE_X_DIMEN - 1)	
AR3	DST_START_ADDRESS - DST_Y_INC	
AR5	DST_Y_INC	Only required if vertical scaling is used
AR6	(2 * SRC_X_DIMEN - 1) - DST_X_DIMEN	
FXBNDRY	Destination boundary (left and right)	Can use **FXLEFT** and **FXRIGHT**
YDST	Y start position	

ILOAD_HIQH and ILOAD_HIQHV

Register	*Function*	*Comment / Alternate Function*
OPMODE	Data format	A 16-bit access is required to prevent modification of the **dirDataSiz** field (bits 17:16).
AR0	DST_END_ADDRESS - DST_Y_INC	
AR2	$\frac{(SRC\_X\_DIMEN - 1) << 16}{(DST\_X\_DIMEN - 1)} + 1$	
AR3	DST_START_ADDRESS - DST_Y_INC	
AR5	DST_Y_INC	Only required if performing vertical scaling using the BITBLT function.
AR6	$\frac{(SRC\_X\_DIMEN - DST\_X\_DIMEN) << 16}{(DST\_X\_DIMEN - 1)}$	
FXBNDRY	Destination boundary (left and right)	Can use **FXLEFT** and **FXRIGHT**
YDST	Y start position	

4.5.9 IDUMP Programming

The following subsections list the registers that must be specifically programmed for IDUMP (image dump: SD/SGRAM -> Host) operations. You must take the following steps:

Step 1. Initialize the registers. Remember to program the registers listed in section 4.5.3.

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	1	1	0	0	0 0 0	#	0 0 0	1 0 1 0

■ **bltmod**: can be BU32BGR, BU32RGB, BU24BGR, or BU24RGB. See Table 4-12.

Register	*Function*	*Comment / Alternate Function*
OPMODE	Data format	A 16-bit access is required to prevent modification of the **dirDataSiz** field (bits 17:16). There is no need to program the **dmamod** field of the **OPMODE** register - reading the DMAWIN or the 8 MByte Pseudo-DMA window is sufficient to trigger the IDUMP.
AR0	Source end address	
AR3	Source start address	
AR5	Source y increment	Not required for a linear source
FXBNDRY	Destination boundary. Left = 0; Right = number of pixels per line minus 1	Can use **FXLEFT** and **FXRIGHT**
YDSTLEN	The y start position and number of lines	

Note: For **PITCH** the **ylin** field, of this global initialization register, must be set to '0'. The pitch value itself is not used.

Step 2. Program the last register to access the 1D00h-1DFFh range in order to start the drawing engine.

Step 3. Read the data in the appropriate format from either the DMAWIN or 8 MByte Pseudo-DMA memory ranges.

Since the IDUMP operation generates the addresses for the destination, the addresses of the data are not used while accessing either the DMAWIN or 8 MByte Pseudo-DMA window. Subsequently, move string instructions can be used through the 7 KByte space of either the DMAWIN or 8 MByte Pseudo-DMA window to read the data from the MGA-2164W. It is recommended that host CPU instructions be used in such a way that each transfer increments the address. This way, the PCI bridge can proceed using burst transfers (assuming they are supported and enabled).

Dwords are always transferred in whole numbers: depending on the source's width and alignment, part of the last dword of every line transferred may contain irrelevant data. The total number of dwords can be calculated by the following formula:

Total = INT ((psiz * width + 31) / 32) * Nlines

Legend:

Total: The number of dwords to transfer.

width: The number of pixels to be read in the x direction, according to Table 4-11.

Nlines: The number of lines to read, according to Table 4-11.

psiz: The destination size, according to Table 4-10.

Table 4-10: IDUMP Source Size

bltmod	pwidth	psiz
BU32RGB	PW8	8
	PW16	16
	PW24	24
	PW32	32
BU32BGR	-	32
BU24RGB	-	24
BU24BGR	-	24

Table 4-11: IDUMP Width and Nlines parameters

linear	bltmode	*width*	Nlines
0	-	Width of destination boundary.	Number of lines associated with the destination.
1	BU32RGB BU32BGR	Width of destination boundary.	Number of lines associated with the destination.
	BU24RGB BU24BGR	Width of source boundary	1

There are some restrictions in the data formats that are supported for this operation. Table 4-12 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 4-184).

Table 4-12: IDUMP Supported Formats

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

.

4.6 CRTC Programming

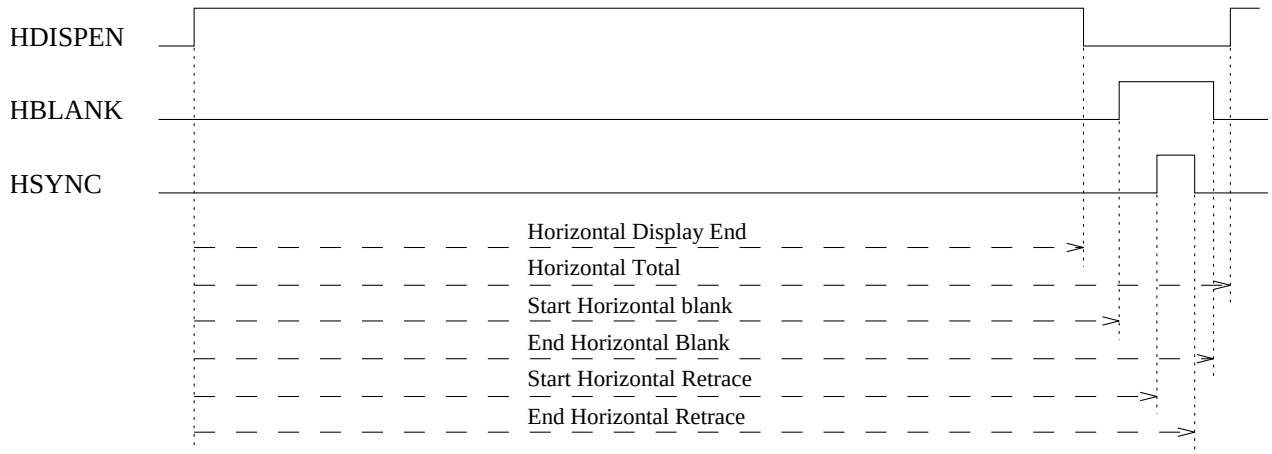
The CRTC can be programmed in one of two modes: VGA Mode or Power Graphic Mode. The **mgamode** field of the **CRTCEXT3** register is used to select the operating mode.

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.

4.6.1 Horizontal Timing

Figure 4-6: 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 minus 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 to **hsyncstr** (**CRTC4**) 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.

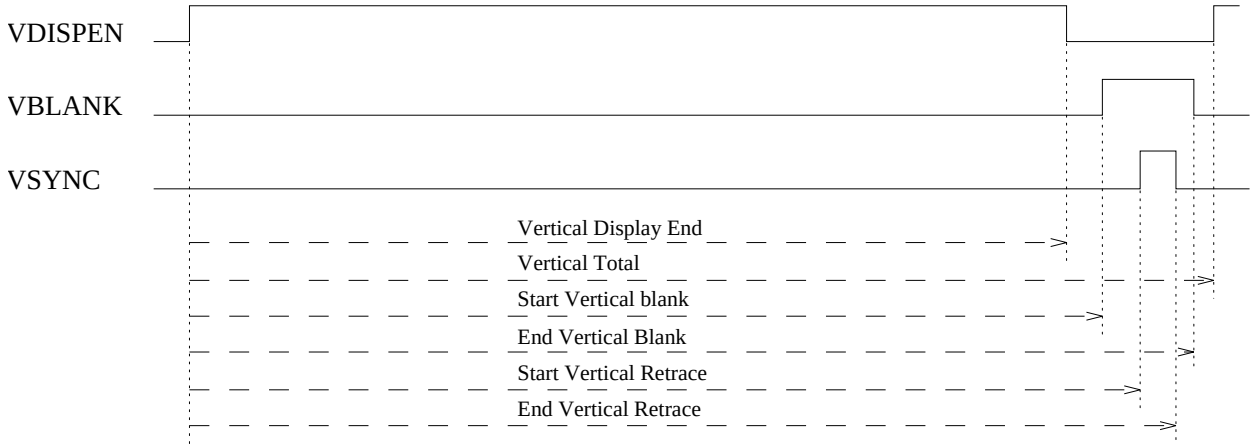
The suggested scale factor settings are shown in the following table:

Bits/Pixel	memconfig = 00	memconfig = 01	memconfig = 10
8	‘001’	‘000’	‘000’*
16	‘011’	‘001’	‘000’
24	‘101’	‘010’	‘010’*
32	‘111’	‘011’	‘001’

(*) Requires special horizontal CRTC programming (use the same parameters as the horizontal zoom by 2 case).

4.6.2 Vertical Timing

Figure 4-7: 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 to **vsyncstr** (CRTC10) 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.

4.6.3 Memory Address Counter

In VGA Mode, the following registers are used to program the memory address counter and the cursor/underline circuitry:

startadd<15:0>	Start address
offset<7:0>	Logical line width of the screen. This is programmed with the number of double or single words in one character line.
curpos<15:0>	Cursor position
prowscan<4:0>	Preset row scan
maxscan<4:0>	Maximum scan line
currowstr<4:0>	Row scan cursor begins
currowend<4:0>	Row scan cursor ends
curoff<4:0>	Cursor off
undrow<4:0>	Horizontal row scan where underline will occur
curskew<1:0>	Cursor skew control

- The row scan counter can be clocked by the horizontal sync signal or by the horizontal sync signal divided by 2, depending on the value of the **conv2t4** (200 to 400 line conversion) field of the **CRTC9** register.
- The memory address counter clock is controlled by **count4 (CRTC14)** and **count2 (CRTC17)**. These fields have no effect in Power Graphic Mode.
- The memory address can be modified by the **dword** (**CRTC14**), **wbmode**, **addwrap**, **selrowscan,** and **cms** (**CRTC17**) fields.

In Power Graphic Mode, the following fields are extended in order to support both a larger display, and up to 8 megabytes of memory.

startadd<19:0> Start address.

offset<9:0> Logical line width of the screen. This is programmed with the number of slices in one character line.

- The display can be placed in interlace mode if the **interlace** bit of the **CRTCEXT0** register is set to '1'.
- The **curpos**, **prowscan**, **currowstr**, **currowend**, **curoff**, **undrow** and **curskew** registers are not used in Power Graphic Mode.
- The **maxscan** field of the **CRTC9** register is used to zoom vertically in Power Graphic Mode.
- Horizontal zooming can be achieved by dividing the pixel clock period and re-programming the horizontal registers.

4.6.4 Programming in VGA Mode

The VGA CRTC of the MGA-2164W 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-2164W'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 GCLKs) must be applied to the formulas which follow, according to whether text or graphics are being displayed:

Variable	*VGA Text*	*VGA Graphics*
A	64	28
B	1	1
C	6	6
D	73	37

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 * Tgclk
2. (cc * 4 - 1) * Tvclk >= A * Tgclk
3. cc * Tvclk >= B * Tgclk
4. (cc * ((H_total - Byte_pan) - H_dispend + 2) - 1) * Tvclk >= (A + C) * Tgclk
5. (cc * ((H_total - Byte_pan) - (H_dispend + MAX(H_dispskew + 2, H_syncstr - H_dispend)) + 2) - 3) * Tvclk >= (A + C) * Tgclk
6. (cc * ((H_total - Byte_pan) - H_dispend + 3) - 1) * Tvclk >= (D + C) * Tgclk

4.6.5 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:

	memconfig = 00	**memconfig = 01**	**memconfig = 10**
offset =	video pitch * bpp 64	video pitch * bpp 128	video pitch * bpp 256

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).

For example, for a 16 bit/pixel frame buffer at a resolution of 1280 x 1024 and a memconfig value of 01:

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	**memconfig = 00**	**memconfig = 01**	**memconfig = 10**
8	64	128	256
16	32	64	128
24	64	128	256
32	16	32	64

The **startadd** field represents the number of pixels to offset the start of the display by:

$$\textbf{startadd} = \frac{\text{address of the first pixel to display}}{\textit{factor}}$$

Depending on the pixel depth, the following *factor*s must be used:

bpp	memconfig = 00	memconfig = 01	memconfig = 10
8	4	8	16
16	2	4	8
24	$1^{1}/_{3}$	$2^{2}/_{3}$	$5^{1}/_{3}$
32	1	2	4

For example, to program **startadd** to use an offset of 64 with a 16 bit/pixel frame buffer while memconfig = 01, **startadd** = 64/4 = 16. With a 24 bit/pixel frame buffer, **startadd** = $64/2^{2}/_{3} = 24$.

Note: When accessing the three-part **startadd** field, the portion which is located in **CRTCEXT0** must ***always*** be written; it must always be 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. Tearing will occur if the four LSBs of **startadd** change value during the active vertical time.

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: The Attributes Controller (ATC) is not available in Power Graphic Mode.

Note: When memconfig = 10, the startadd must be an even number.

Note: When memconfig = 00, the start address must be selected so that no line crosses the 2 Mbyte boundary. When memconfig = 01, the start address must be selected so that no line crosses the 4 Mbyte boundary. There is no boundary restriction for memconfig = 10.

Within Power Graphic Mode there is no overscan, therefore, use the following:

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.

Note: The composite sync is always active low. 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 by slowing down the pixel clock and re-programming the horizontal registers of the CRTC.

- For example, to obtain a horizontal zoom rate of x2, slow the pixel clock by two and re-program all th horizontal CRTC registers so that the period, active time, front and back porch, blank and sync width (in ns) remain the same. The horizontal counter will have a precision of 16 or 32 pixels on the screen for zoom rates of x2 or x4 respectively.

Vertical zooming is achieved by re-scanning a line 'n' times. Program the **CRTC9** register's **maxscan** field with the appropriate value, n-1, to obtain a vertical zoom.

- For example, set **maxscan** = 3 to obtain a vertical zoom rate of x4.

***Limitations*:**

- **htotal** must be greater than 0 (because of the delay registers on the **htotal** comparator)
- **htotal** - **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.
- **(htotal** modulo 16) must *not* equal 15
- **(htotal)***(scale + 1) MOD 16 must *not be 15*.

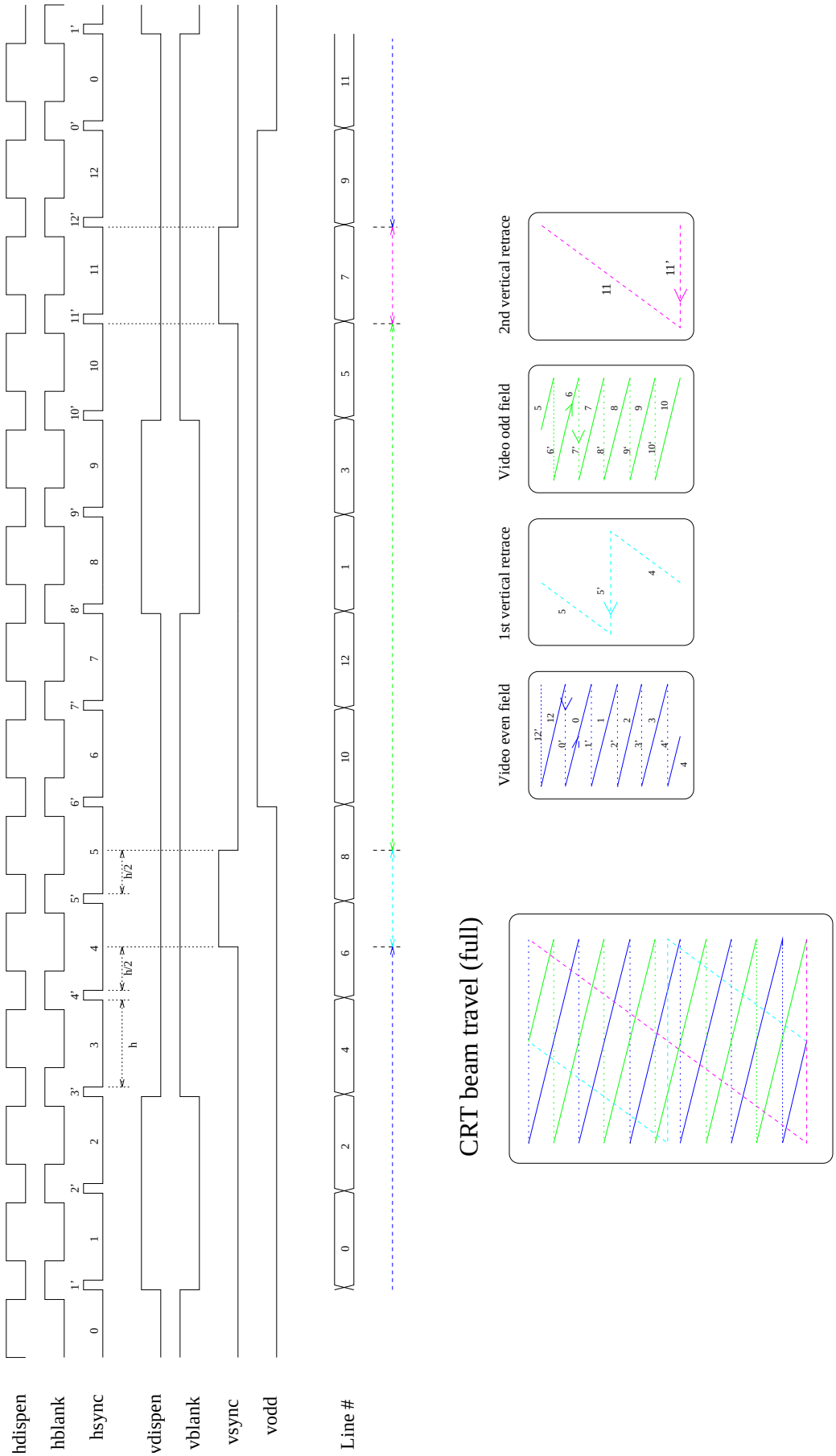
CRTC Latency Formulas

This section presents several rules that must be followed in Power Graphic Mode in order to adhere to the latency constraints of the CRTC.

In the formulas below, ‘cc’ represents the number of VCLKs per character (8 pixels). Using these values, we can determine the following rules:

1. (cc * (H_total - (H_dispend + MAX(startadd<3:0> + 1/cc, H_syncstr - H_dispend))) - 1.5) * Tvclk >=68 * Tgclk
2. 58.5 * Tvclk >=68 * Tgclk
3. 16 * Tvclk >= Tgclk
4. (cc * (H_total - H_dispend) + MOD(pitch*cc/8 - 1, 16) + 1.5) * Tvclk >= 77 * Tgclk
5. (cc * (H_total - (H_dispend + MAX(startadd<3:0> + 1/cc, H_syncstr - H_dispend)) +1) - 1.5) * Tvclk >=24 * Tgclk
6. (cc * (H_total - H_dispend + 1) + MOD(pitch*cc/8 - 1, 16) + 1.5) * Tvclk >= 83 * Tgclk

Figure 4-8: Video Timing in Interlace Mode



4.7 Interrupt Programming

The MGA-2164W has five interrupt sources:

1. **Pick interrupt**

   This interrupt is used to help with item selection in a drawing. A rectangular pick region is programmed using the clipper registers (**YTOP**, **YBOT**, **CXLEFT**, **CXRIGHT**). All planes must be masked by writing FFFFFFFFh to the **PLNWT** register. The drawing engine then redraws every primitive in the drawing. When pixels are output in the clipped region, the pick pending status is set. After a primitive has been initialized, the **STATUS** register's dwgengsts bit can be polled to determine if some portion of the primitive lies within the clipping region.

   Picking interrupts are generated when primitives are drawn using either RPL, RSTR, ZI, or I. These access types are explained in the atype field description for the **DWGCTL** register in Chapter 3.

2. **Vertical sync interrupt**

   This interrupt is generated every time the vsync signal goes active. It can be used to synchronize a process with the video raster such as frame by frame animation, etc. The vsync interrupt enable and clear are both located in the **CRTC11** VGA register.

3. **Vertical line interrupt**

   This interrupt is generated when the value of the linecomp field of **CRTC18** equals the current vertical count value. This interrupt is more flexible than the vertical sync interrupt because it allows interruption on any horizontal line (including blank and sync lines).

4. **External interrupt**

   This interrupt is generated when the external interrupt line is driven active. It is the responsibility of the external device to provide the clear and enable functions.

The following table summarizes the supported functionality that is associated with each interrupt source.

Interrupt	*STATUS*	*EVENT*	*ENABLE*	*CLEAR*
Pick	- -	**pickpen** **STATUS**<2>	**pickien** **IEN**<2>	**pickiclr** **ICLEAR**<2>
Vertical sync	**vsyncsts** **STATUS**<3>	**vsyncpen** **STATUS**<4>	**vinten** **CRTC11**<5>	**vintclr** **CRTC11**<4>
Vertical line	- -	**vlinepen** **STATUS**<5>	**vlineien** **IEN**<5>	**vlineiclr** **ICLEAR**<5>
External	**extpen** **STATUS**<6>	-	**extien** **IEN**<6>	-

STATUS Indicates which bit reports the current state of the interrupt source.

EVENT Indicates which bit reports that the interrupt event has occurred.

ICLEAR A pending bit is kept set until it is cleared by the associated clear bit.

IEN Each interrupt source may or may not take part in activating the PINTA/ hardware interrupt line. The EVENT and STATUS flags are not affected by interrupt enabling or disabling, **vsyncpen** is the only exception. When **vinten** = 0, **vsyncpen** will not be generated; **vinten** must be set to '0' for **vsyncpen** to be generated.

Note:

- You should 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.

4.8 Power Saving Features

The MGA-2164W supports two power conservation features:

- DPMS is supported directly, through the following control bits:
  - Video can be disabled using **scroff** blanking bit (**SEQ1**<5>)
  - Vertical sync can be forced inactive using **vsyncoff** (**CRTCEXT1**)
  - Horizontal sync can be forced inactive using **hsyncoff** (**CRTCEXT1**)
- The power consumption of the chip can be reduced by slowing down the system clocks and stopping the video clocks. An internal divide by 4 (see the GCALE bit in the OPTION register) is available to further reduce the gclk period.
- If you want to preserve the frame buffer contents in power-down, the **rfhcnt** field must be appropriately programmed.

Chapter 5: Hardware Designer's Notes

5.1 Introduction

The MGA-2164W chip has been designed in such a way as to minimize the amount of external logic required to implement a board. Included among its features are:

- Direct interface to the PCI bus (MGA-2164W-PCI) or AGP bus (MGA-2164W-AGP).
- All necessary support for external devices such as ROM, RAMDAC, video co-processor (including MGA-VCO64SFB), and others
- Direct connection to the RAM

5.2 PCI Interface

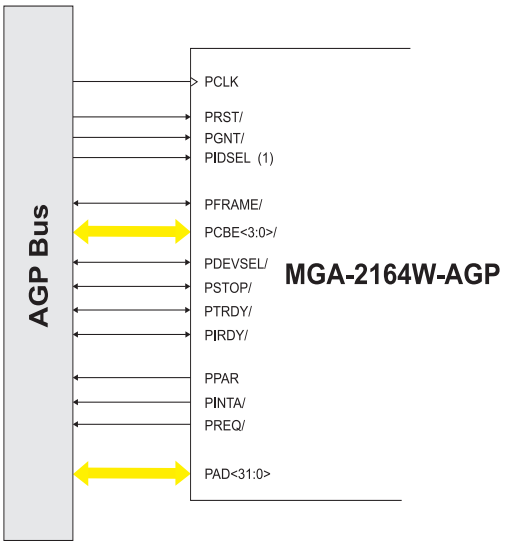
The MGA-2164W-PCI interfaces directly with PCI as shown in Figure 5-2. The MGA-2164W-PCI is a medium-speed (target) device which will respond with PDEVSEL/ during the second clock after PFRAME/ is asserted.

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 4.1.2 and 4.1.3 for more information. The MGA-2164W-PCI can also act as a master on the PCI bus - refer to Section 4.1.9 for more information.

5.3 AGP Interface

The MGA-2164W-AGP interfaces with the AGP bus as shown in Figure 5-1. The MGA-2164W-AGP supports the PCI 66MHz interface as medium device (i.e. it responds with PDEVSEL/ during the second clock after PFRAME/ is asserted). It does not use the AGP sideband signals nor the PIPE/ mechanism.

Figure 5-1: AGP Interface



(1) The add-in card manufacturer should connect PAD16 to the PIDSEL pin of the MGA-2164W-AGP because the PIDSEL is not a pin on the AGP connector.

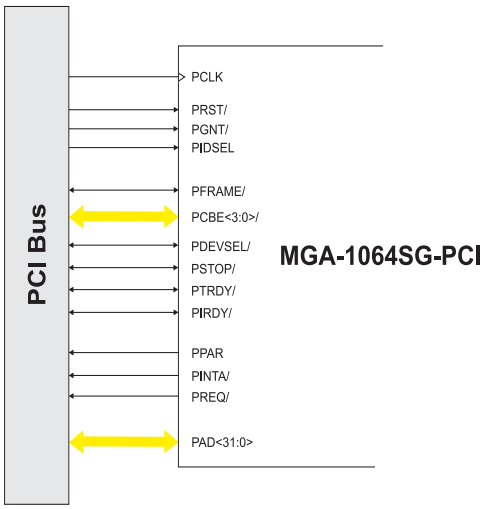
5.4 Snooping

The MGA-2164W 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-2164W is unable to process the access immediately, it takes control of the bus, and a retry cycle is performed.
2. If the MGA-2164W is able to process the access, the access is snooped, and the MGA-2164W 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, there will be a conflict on the PCI bus.

Figure 5-2: PCI Interface



5.5 External Devices

The MGA-2164W supports a few external devices (the EPROM is a standard expansion device that is supported by the MGA-2164W). Other devices can also be added by using the MGA-2164W's EXTCS/ strobe.

Figure 5-3 shows how to connect the standard expansion devices to the MGA-2164W. It should be noted that the local bus interface shares pins with the RAM. This limits the load on the MDQ bus to 10 pF (1 load) per bit, which is automatically the case when there are no extra external devices.

EPROM

The MGA-2164W supports both 256K x 8 and 512K x 8 EPROMs, as well as flash memory. Flash memory provides the capability to modify the BIOS 'on the fly'. The following table lists specific EPROM and flash memory devices that have been verified to work with the MGA-2164W:

	Flash Memory		*EPROM*	
Manufacturer	***256K x 8***	***512K x 8***	***256K x 8***	***512K x 8***
AMD	AM28F256-150	AM28F512-150	AM27256-200	AM27C512-200
Atmel	AT29C257-12 AT29C257-15 AT29C257-90	AT29C512-90 AT29C512-120 AT29C512-150		
SGS	M28F256-15			
Intel	N28F256A-150	N28F512-150		
Toshiba			TC57256AD-20	TMM27512AD-20
Texas Instruments		TMS28F512A-10 TMS28F512A-12 TMS28F512A-15	TMS27C256-2	
National			NM27C256Q200	NMC27C512AQ200
Microchip			27C256-20	27C512-20

A write cycle to the EPROM has been defined in order to support flash memory. Another bit which locks write accesses to the EPROM has also been added in order to prevent unexpected writes.

2 ***Note:*** The sequencing of operations to erase and write the memory must be performed by software. Some timing parameters (tWR, tWH1, tWH2) must be guaranteed by software using programming loops (refer to the device specification).

2 ***Note:*** If a 12V power supply is required for flash memory, it will have to be provided on the board (the MGA-2164W will have no ability to control it).

RAMDAC

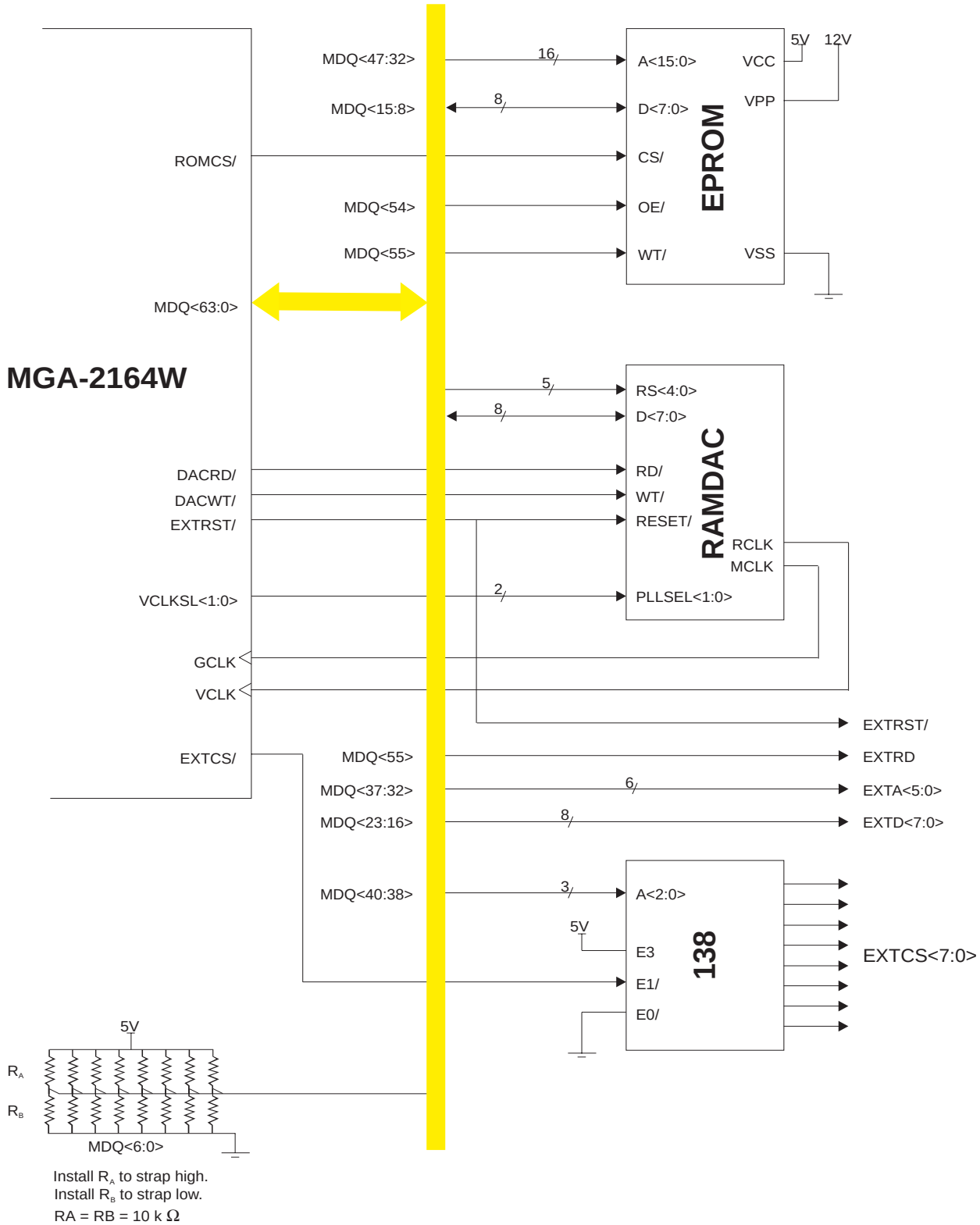
The processor interface of the RAMDAC must be connected as shown in Figure 5-3. For more information on supported RAMDACs and how to connect the video port, refer to section 5.7

Other Devices

Extra devices can be added to the MGA-2164W (in addition to the standard expansion devices mentioned above). If a video co-processor or any other extra device is required, a decoder (as shown in Figure 5-3) can be used to generate multiple CS/ signals. However, in order to respect load constraints on the MDQ bus, the following rules must be respected:

- Read strobes and addresses that are used for both the EPROM and the external devices (including the decoder) must be buffered.
- If multiple devices are added, the data bus to those external devices must be buffered.

Figure 5-3: Expansion Device Connection



5.6 Memory Interface

MGA-2164W connects directly to the WRAM chips (from 2 to 16 MB of WRAM can be connected). The amount of memory will determine the supported resolutions, as described in Section 4.2.1.1.

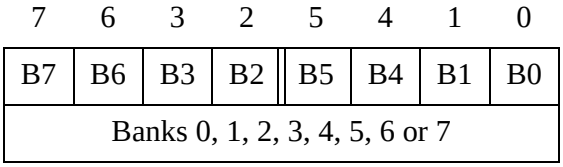
5.6.1 WRAM Connection

Figure 5-4 shows how the WRAM banks connect to the MGA-2164W. Some pins require damping, as shown in Table A-5. This table also lists the maximum load allowed for each pin.

5.6.2 WRAM Byte Organization

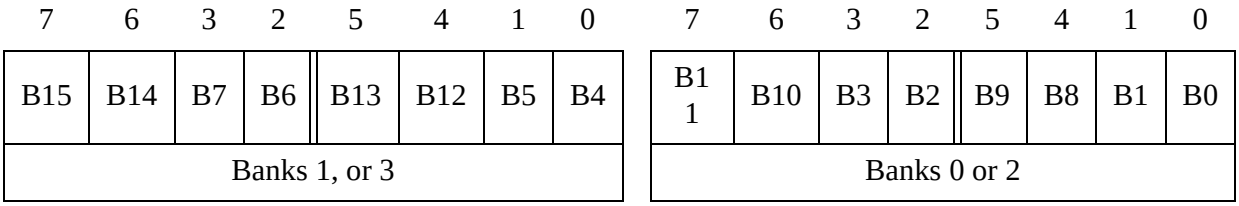
The serials port of the WRAM is half the size of its parallel port. Since MGA-2164W's interface to the frame buffer is 64-bits wide, this provides a 32-bit serial bus. However, for the RAMDAC to perceive the bytes in the correct order, MGA-2164W must take care to format the pixels correctly within a slice. The bytes are organized as follows:

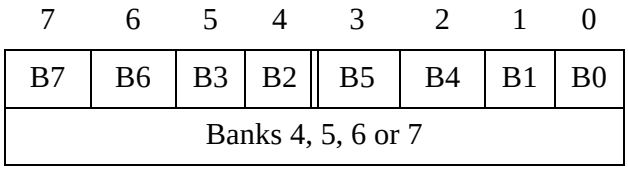
MGA-2164W Byte Line (Non-Interleaved)



When memconfig = 01, an interleave mode is enabled by MGA-2164W for the reasons explained in the previous paragraph, and in order to allow a 64-bit serial bus. Since the pixels are partially multiplexed in the WRAM, however, MGA-2164W must take care to format the pixels correctly. The memory interface can be seen as a 128-bit interface where access to the complete slice requires two separate memory accesses. The bytes are organized in each bank as follows:

MGA-2164W Byte Line (Interleaved)

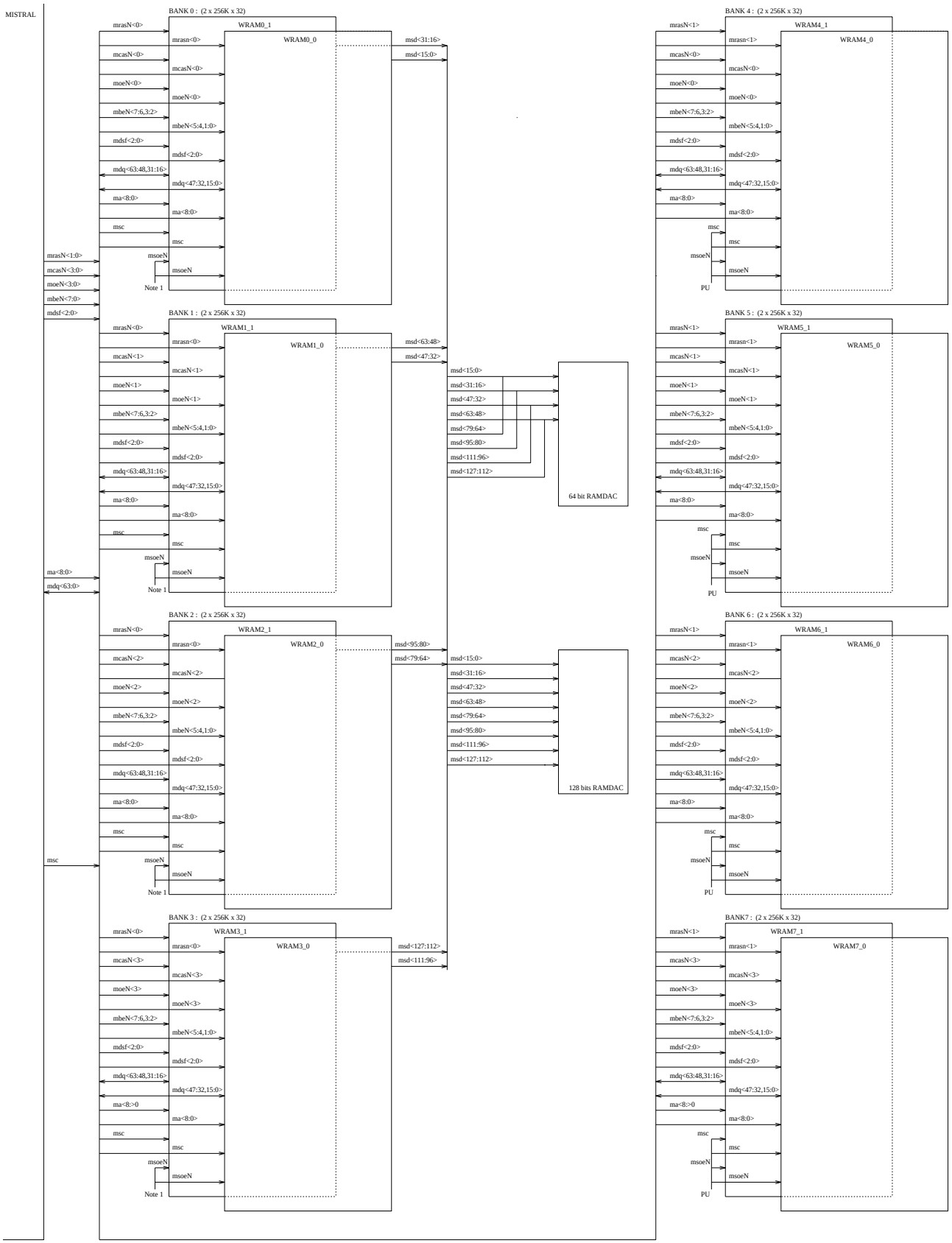




In VGA Mode, only 32 of the 64 bits are used (the unused 32 bits of data are preserved):

- Byte line 0 is used as VGA plane 0
- Byte line 1 is used as VGA plane 1
- Byte line 2 is used as VGA plane 2
- Byte line 3 is used as VGA plane 3

Figure 5-4: Memory Interface Connection



Note : for 64 bits DAC, msoeN<0> controls bank0 and bank1 and msoeN<1> controls bank2 and bank3.
For 128 bits DAC, all WRAM msoeN inputs are tied to GND.

5.7 Video interface

In order to support both high resolutions and high refresh rates, MGA-2164W has been optimized with the Texas Instruments TVP3026,TVP3027, and TVP3033 RAMDAC. If any other RAMDAC is selected, care must be taken when evaluating the timing of the video interface. There are three basic operation modes for the video interface:

1. **VGA Mode:** This requires an 8-bit bus between the MGA-2164W and the RAMDAC.
2. **Power Graphic memconfig = 01 mode:** This requires a 64-bit WRAM-RAMDAC bus.
3. **Power Graphic memconfig = 10 mode:** This requires a 128 bit WRAM-RAMDAC bus (the RAMDAC must be capable of interleave).

5.7.1 VGA Mode

In VGA Mode, data destined for the RAMDAC is always 8 bits wide, and comes from the MGA-2164W chip. It always represents one pixel on the screen on each LDCCLK cycle (in VGA Mode, LDCLK = pixel clock). If a VGA feature connector is not required, MGA-2164W can be interconnected to the RAMDAC without any glue logic. When a feature connector is required, the VGA interface must be modified as shown in Figure 5-8.

5.7.2 Power Graphic Mode

In Power Graphic Mode, there are two ways to connect the pixel port of the RAMDAC: 64 bit RAMDAC or 128 bit RAMDAC. Within a configuration, it is possible to vary the memconfig to get single buffer mode or split frame buffer mode. Figures 5-5, 5-6, and 5-7 show the serial stream formatting under different memconfig settings.

Figure 5-5: memconfig = 00 Video Data

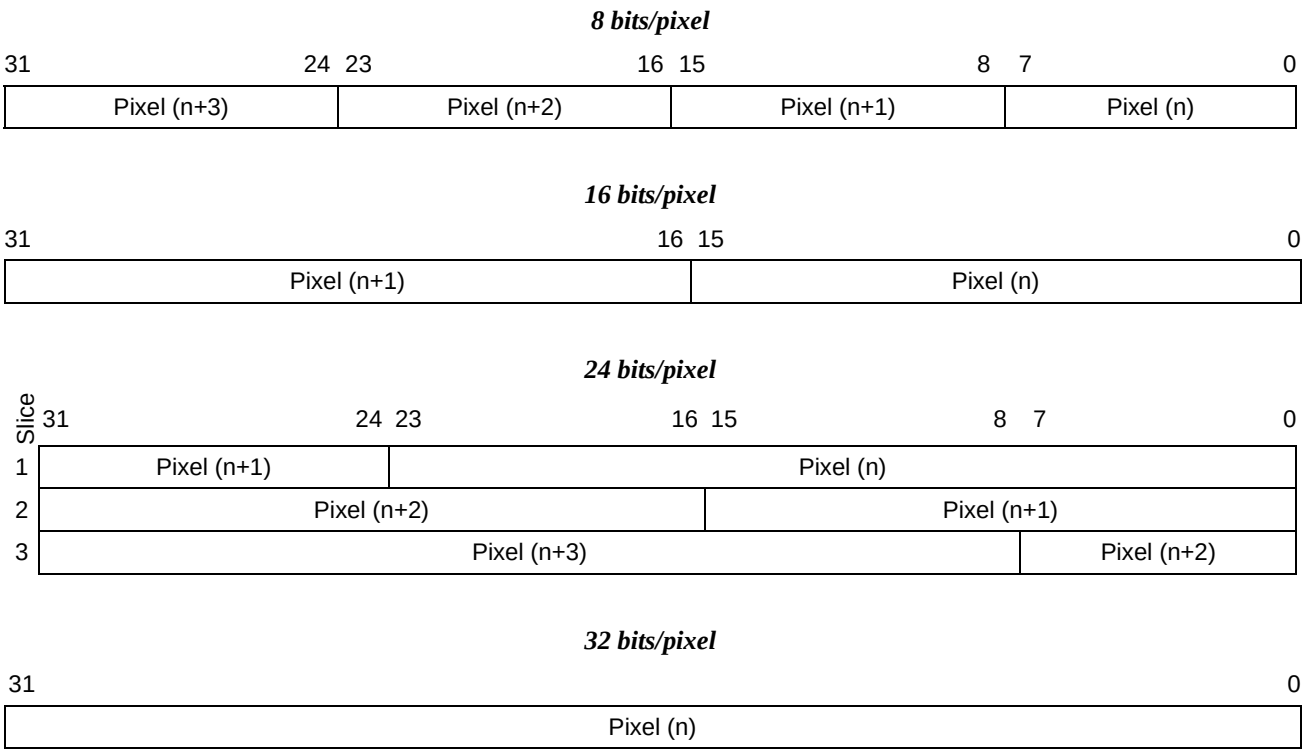


Figure 5-6: memconfig = 01 Video Data

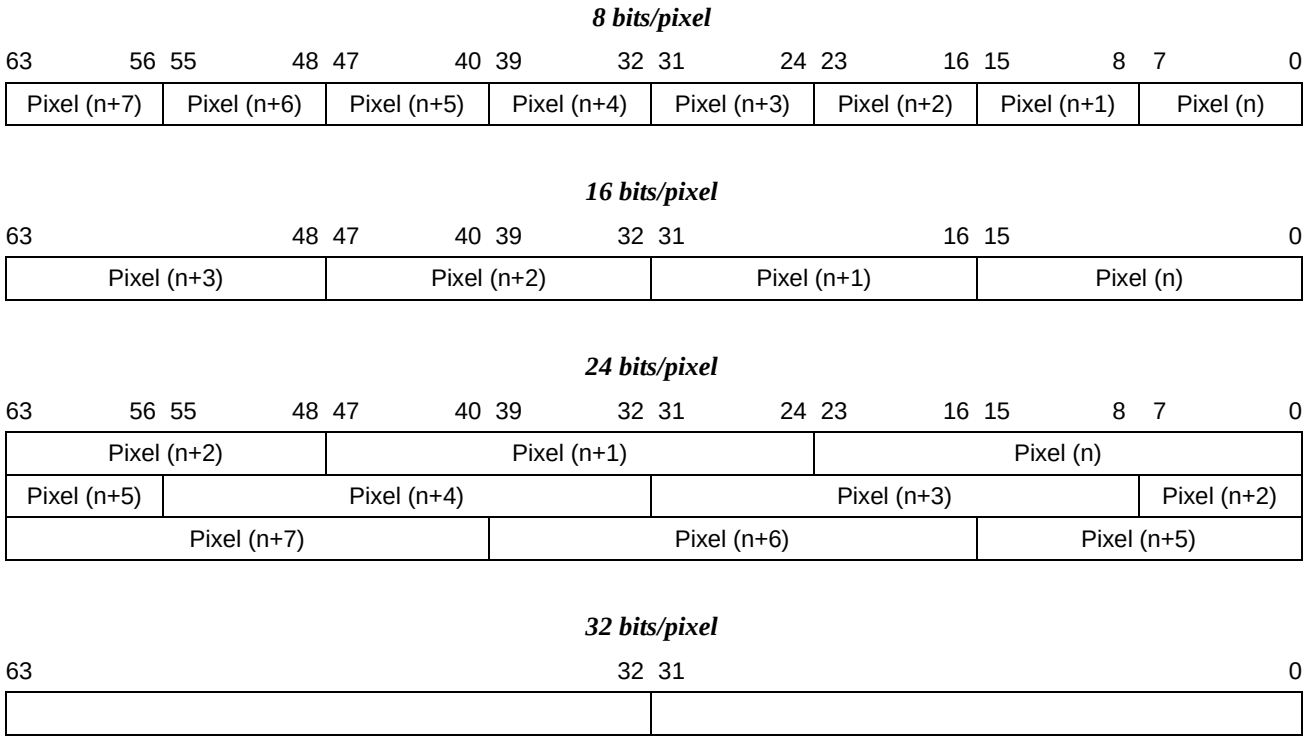
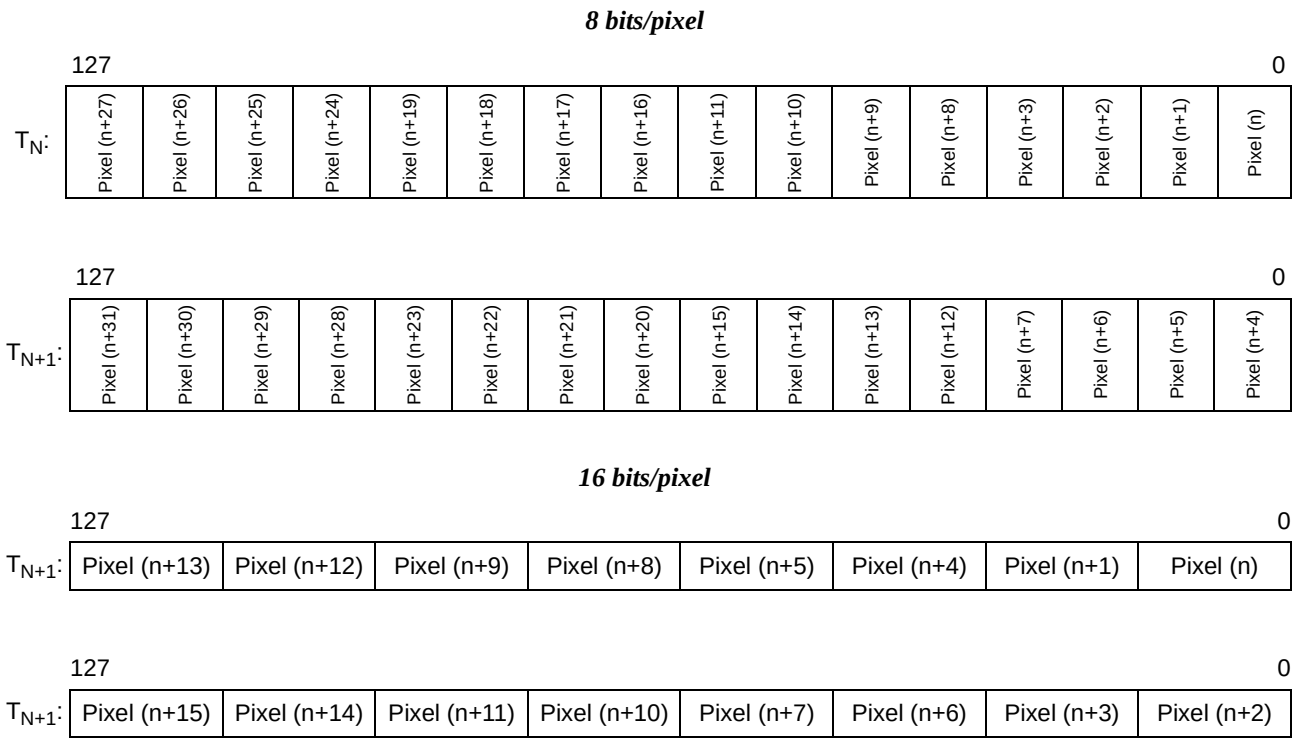


Figure 5-7: memconfig = 10 Video Data



24 bits/pixel

	127															0
T_N:	Blue Pixel (n+9)	Red Pixel (n+8)	Green Pixel (n+8)	Blue Pixel (n+8)	Green Pixel (n+6)	Blue Pixel (n+6)	Red Pixel (n+5)	Green Pixel (n+5)	Red Pixel (n+3)	Green Pixel (n+3)	Blue Pixel (n+3)	Red Pixel (n+2)	Blue Pixel (n+1)	Red Pixel (n)	Green Pixel (n)	Blue Pixel (n)
T_{N+2}:	Red Pixel (n+19)	Green Pixel (n+19)	Blue Pixel (n+19)	Red Pixel (n+18)	Blue Pixel (n+17)	Red Pixel (n+16)	Green Pixel (n+16)	Blue Pixel (n+16)	Green Pixel (n+14)	Blue Pixel (n+14)	Red Pixel (n+13)	Green Pixel (n+13)	Red Pixel (n+11)	Green Pixel (n+11)	Blue Pixel (n+11)	Red Pixel (n+10)
T_{N+4}:	Green Pixel (n+30)	Blue Pixel (n+30)	Red Pixel (n+29)	Green Pixel (n+29)	Red Pixel (n+27)	Green Pixel (n+27)	Blue Pixel (n+27)	Red Pixel (n+26)	Blue Pixel (n+25)	Red Pixel (n+24)	Green Pixel (n+24)	Blue Pixel (n+24)	Green Pixel (n+22)	Blue Pixel (n+22)	Red Pixel (n+21)	Green Pixel (n+21)

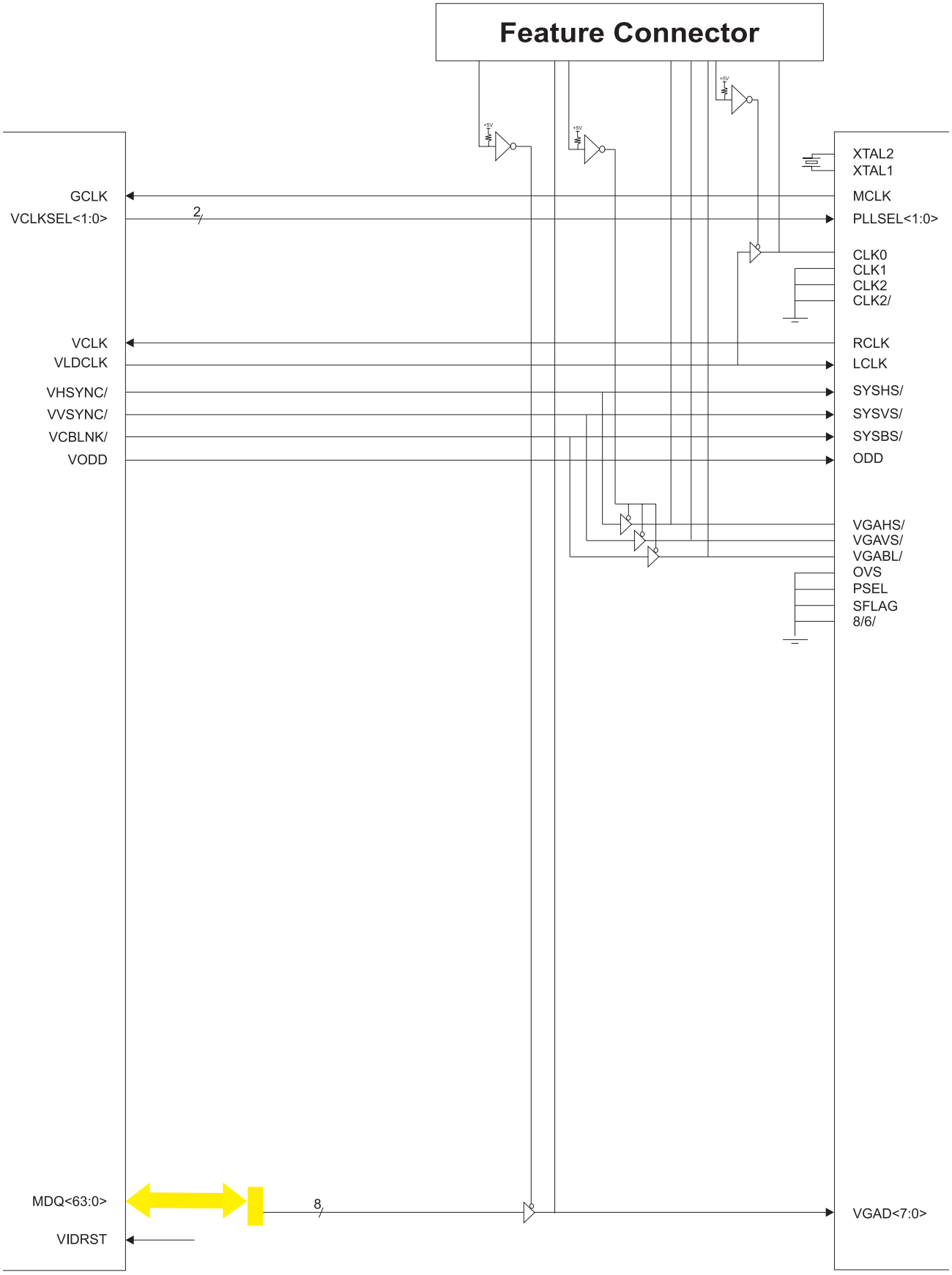
	127															0
T_{N+1}:	Green Pixel (n+10)	Blue Pixel (n+10)	Red Pixel (n+9)	Green Pixel (n+9)	Red Pixel (n+7)	Green Pixel (n+7)	Blue Pixel (n+7)	Red Pixel (n+6)	Blue Pixel (n+5)	Red Pixel (n+4)	Green Pixel (n+4)	Blue Pixel (n+4)	Green Pixel (n+2)	Blue Pixel (n+2)	Red Pixel (n+1)	Green Pixel (n+1)
T_{N+3}:	Blue Pixel (n+21)	Red Pixel (n+20)	Green Pixel (n+20)	Blue Pixel (n+20)	Green Pixel (n+18)	Blue Pixel (n+18)	Red Pixel (n+17)	Green Pixel (n+17)	Red Pixel (n+15)	Green Pixel (n+15)	Blue Pixel (n+15)	Red Pixel (n+14)	Blue Pixel (n+13)	Red Pixel (n+12)	Green Pixel (n+12)	Blue Pixel (n+12)
T_{N+5}:	Red Pixel (n+31)	Green Pixel (n+31)	Blue Pixel (n+31)	Red Pixel (n+30)	Blue Pixel (n+29)	Red Pixel (n+28)	Green Pixel (n+28)	Blue Pixel (n+28)	Green Pixel (n+26)	Blue Pixel (n+26)	Red Pixel (n+25)	Green Pixel (n+25)	Red Pixel (n+23)	Green Pixel (n+23)	Blue Pixel (n+23)	Red Pixel (n+22)

32 bits/pixel

	127			0
T_N:	Pixel (n+6)	Pixel (n+4)	Pixel (n+2)	Pixel (n)

	127			0
T_{N+1}:	Pixel (n+7)	Pixel (n+5)	Pixel (n+3)	Pixel (n+1)

Figure 5-8: Feature Connector



5.7.3 Slaving the MGA-2164W

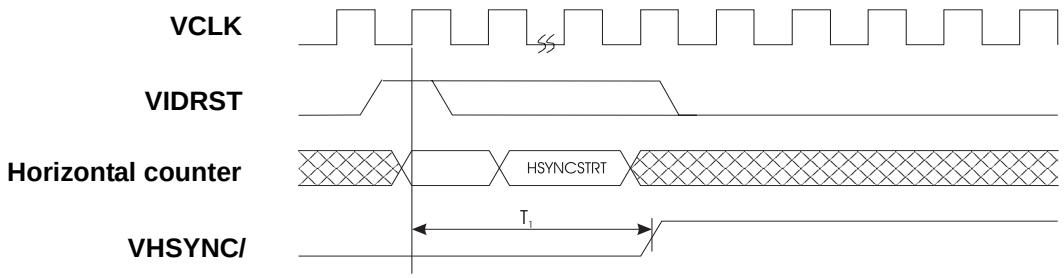
This section describes the operations of the VIDRST (video reset input) signal. A VIDRST is detected on the first rising edge of VCLK where VIDRST is high. The video reset can affect both the horizontal and/or vertical circuitry.

The first time that the MGA-2164W'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.

2 ***Note:*** In order for the MGA-2164W to be synchronous with any other source, the MGA-2164W CRTC must be programmed with the same video parameters as that other source. VCLK 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 retrace. Figure 5-9 shows the relationship between VIDRST, the internal horizontal counter, and VHSYNC/ when the MGA-2164W is already synchronized.

Figure 5-9: VIDRST, Internal Horizontal Active

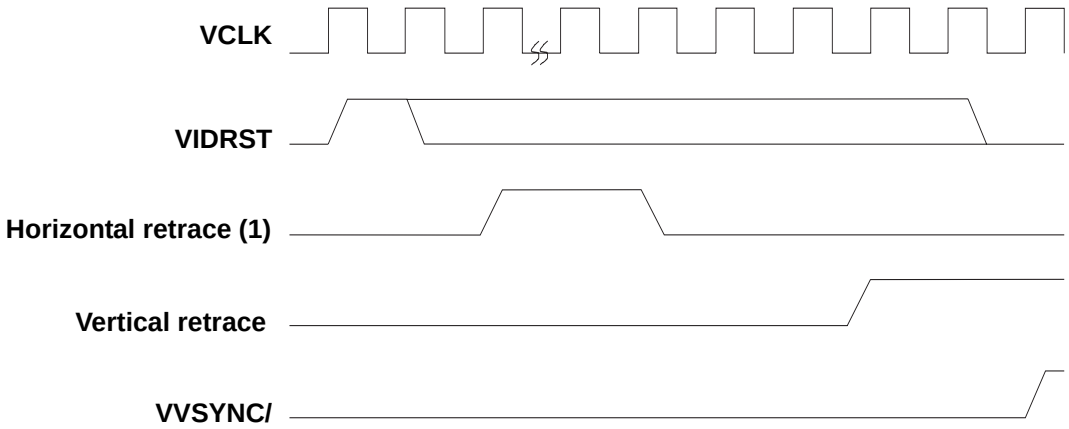


2 Note: The VHSYNC/ pin of the MGA-2164W will become active following the formula shown below (T_1 is a number of VCLK):

$$T_1 = 4 + SCALE + STARTADD<3:0>$$

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 retrace in the even field. Figure 5-10 shows the relationship between VIDRST, the internal horizontal retrace, the internal vertical retrace signal, and VVSYNC/ when only the vertical counter is reset.

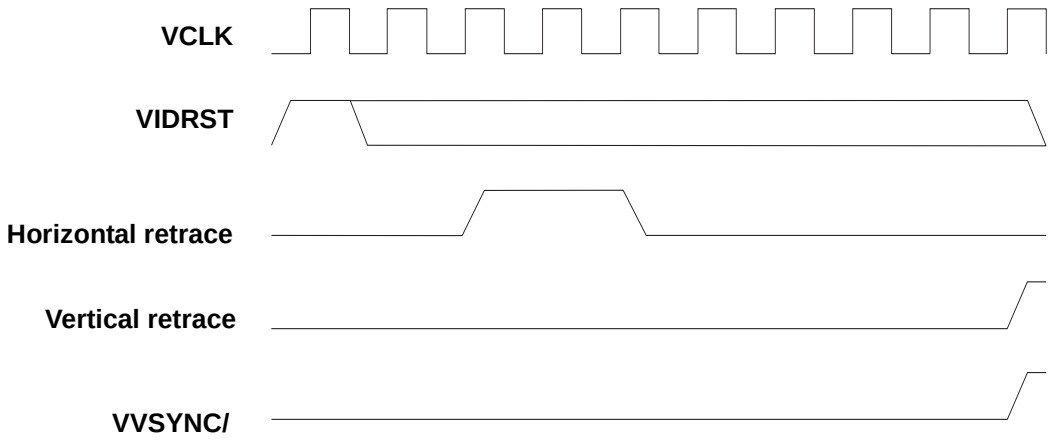
Figure 5-10: VIDRST, Internal Horizontal Retrace/Vertical Retrace



(1) Horizontal counter and horizontal retrace are not affected by VIDRSTS when only the vertical reset is active. They are shown in the waveform as a reference to the location where VIDRSTS can be active in steady state.

Figure 5-11 shows the relationship between VIDRST, the internal horizontal retrace, and the internal horizontal and vertical active signals, when both the horizontal and vertical counters are reset.

Figure 5-11: VIDRST, Internal Horizontal/Vertical Active, and VVSYNC/



5.8 Co-processor Interface

Two pins permit sharing of the WRAM bus:

- MVGNT/ (generated by the MGA-2164W)
- MVREQ/ (generated by the co-processor)

When it releases the bus to the co-processor, the MGA-2164W chip brings all WRAM control signals high before placing them in tristate. The co-processor should do the same when releasing the bus. This procedure will guarantee that no false access will be performed on the memory.

Figure 5-12 shows the normal sequence when the co-processor requests and releases the bus.

The priority of operations in the MGA-2164W is organized in such a way that the MGA-2164W will notify the system when it requires the bus in order to perform data transfer or refresh cycles. When this is the case, the co-processor must return the bus to the MGA-2164W as illustrated in Figure 5-12.

The MGA-2164W's priorities are as follows:

1. Data transfer
2. Second refresh request
3. Co-processor requests
4. Direct frame buffer or external device access
5. Drawing engine
6. First refresh request

When mgamode = 0, co-processor requests will not be granted.

Pull-up resistors (10k Ω) are required on MRAS1/ and MRAS0/ when a co-processor is installed. (These maintain level 1 logic when the bus is tristated during the interval when the bus is transferred between MGA-2164W and the co-processor).

Figure 5-12: Co-processor Requests

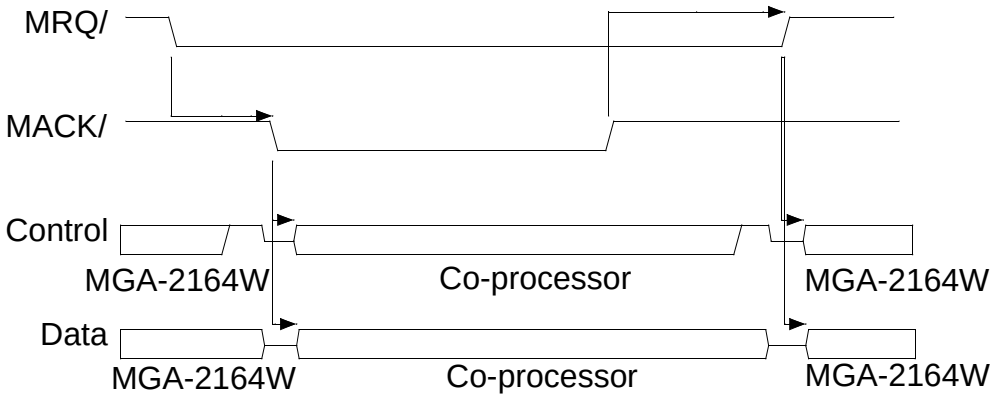
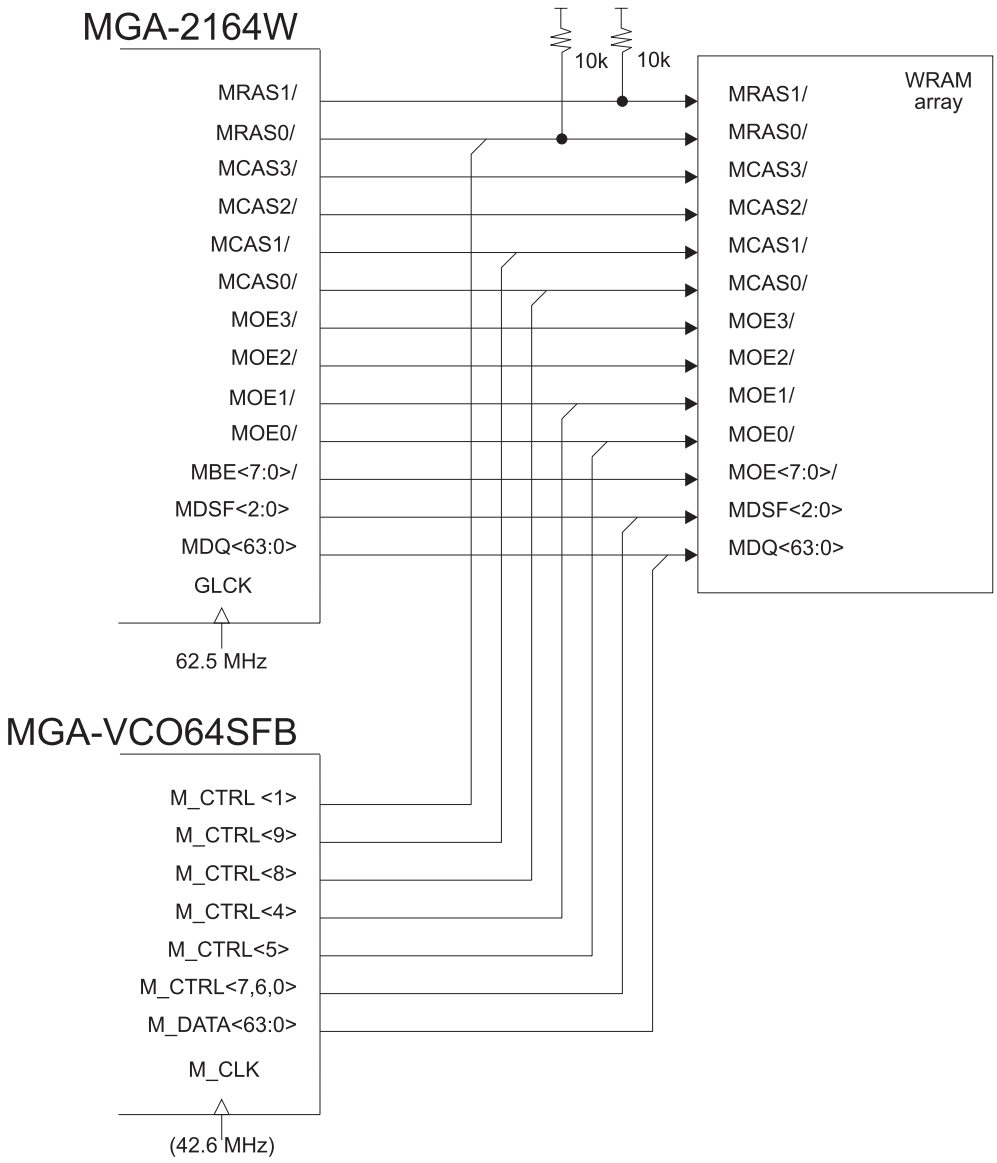


Figure 5-13 details the connection between the MGA-2164W and the video co-processor (MGA-VCO64SFB).

2 ***Note:*** The MGA-VCO64SFB can read and write only WRAM banks 0 and 1 (4 megabytes maximum).

Figure 5-13: Connection with the Co-processor.



Appendix A: Technical Information

A.1 Pin List

Table A-1: Pin Count Summary

Group	*Total*	*I*	*O*	*I/O*
Host	48	4	3	41
Local	6	1	5	
Memory	97	2	31	64
Video	12	2	10	
Test	2	2		
VDD/ GND/ Reserved	60			

A.1.1 Host

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-2164W 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.
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-2164W 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.

A.1.2 Local Interface

Name	*# Pins*	*Type*	*Description*
ROMCS/	1	O	Bios ROM chip select. When ROMCS/ is active: MDQ<47:32> is redefined as the ROM address; MDQ<54> as the output enable, MDQ<55> as the ROMREAD signal; MDQ<15:8> as the data.
DACRD/	1	O	RAMDAC read control signal. When DACRD/ is active: MDQ<52:48> is defined as the dac address, MDQ<31:24> as the data.
DACWT/	1	O	RAMDAC write control signal. When DACWT/ is active: MDQ<52:48> is redefined as the dac address; MDQ<31:24> as the data.
EXTRST/	1	O	External reset signal. Used to reset the RAMDAC and expansion devices.
EXTCS/	1	O	Expansion device select. Use to select companion chips. When EXTCS/ is asserted: pin MDQ<55> is redefined as the read/write signal; MDQ<40:32> as the address; MDQ<23:16> as the data.
EXTINT/	1	I	External interrupt pin. Can be used by a companion chip to generate interrupts on the PCI bus. Interrupt is an active low level interrupt.

A.1.3 Memory Interface

Name	*# Pins*	*Type*	*Description*
MDQ<63:0>	64	I/O	Memory data bus. Used during read and write transactions. Also used for BIOS EPROM and RAMDAC accesses, and chip strapping. MDQ<4:0> = Product ID straps or switches. MDQ<5> = VGA boot strap or switch. MDQ<6> = BIOS EPROM installed strap or switch. MDQ<7> = Strap reserved for future use.Must be pulled down by a 10k resistor. MDQ<15:8> = ROMDQ: ROM data bus. Used (if Flash ROM is present) to read from or write to the BIOS EPROM. MDQ<23:16> = EXTDQ: Expansion device data bus. Used to read from or write to companion chips. MDQ<31:24> = RDACDATA: RAMDAC host data bus. Used to read from or write to the host palette registers. MDQ<47:32> = EXTA<15:0>: BIOS EPROM addresses. Bits EXTA<8:0> are also used for addressing expansion devices. MDQ<52:48> = RS<4:0>: RAMDAC host address bus. MDQ<54> = BIOS EPROM output enable (active low signal). MDQ<55> = ROMREAD and EXTREAD: Used to read from or write to a Flash ROM. (Do not connect when using an EPROM.) Also used to indicate a read or write transaction to an expansion device. MDQ<63:56> = VGADAT<7:0>: VGA data output. Provides the VGA pixel value required for VGA emulation modes. Connects to the VGA pixel port of the RAMDAC.
MA<8:>	9	O	Memory addresses (row, column multiplexed).
MRAS<1:0>/	2	O	Memory row address strobe.
MCAS<3:0>/	4	O	Memory column address strobe.
MOE<3:0>/	4	O	Memory output enable.
MBE<7:0>/	8	O	Memory byte enable. Used to determine which byte field should be written in the 64-bit slice.
MDSF<2:0>	3	O	Controls special functions of the WRAM.
GCLK	1	I	Graphic and memory interface clock.
MVREQ/	1	I	Memory control request. Used by a co-processor to get control of the frame buffer.
MVGNT/	1	O	Memory grant. Informs a co-processor that it has control of the frame buffer.

A.1.4 Video Interface

Name	*# Pins*	*Type*	*Description*
VCLKSL <1:0>	2	O	Clock generator control bits. Comes from the MISC**<3:2>** register.
VCLK	1	I	Video clock for the CRTC and screen refresh operations.
VIDRST	1	I	Video reset input. Used to synchronize the CRTC on an external source.
VHSYNC/	1	O	Horizontal sync.
VVSYNC/	1	O	Vertical sync.
VCBLNK/	1	O	Video composite blank signal.
VLDCLK	1	O	Video output load clock.
VODD	1	O	Video odd frame. Indicates that the odd or even frame is currently serialized out.
MSC	1	O	Memory serial clock.
MSOE<1:0>/	2	O	Memory serial output enable.

A.1.5 Test

Name	*# Pins*	*Type*	*Description*
HIZ/	1	I	This pin puts all output buffers in tristate for test purposes. This pin should be tied to a pull-up during normal operation.
LFT	1	I	This pin is used for chip testing *only*. Connect to GND for normal operation.

A.1.6 VDD/GND

Name	*# Pins*	*Type*	*Description*
VDD5	5		Attaches to +5 volts. This applies **only** to MGA-2164W-PCI. (These are noconnect pins on MGA2164W-AGP)
VDD3	12		Attaches to +3.3 volts.
GND	21		Attaches to ground
Reserved	22		Do not connect.

A.2 PCI Pinout Illustration and Table

The illustration below shows the locations of the MGA-2164W's 225 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

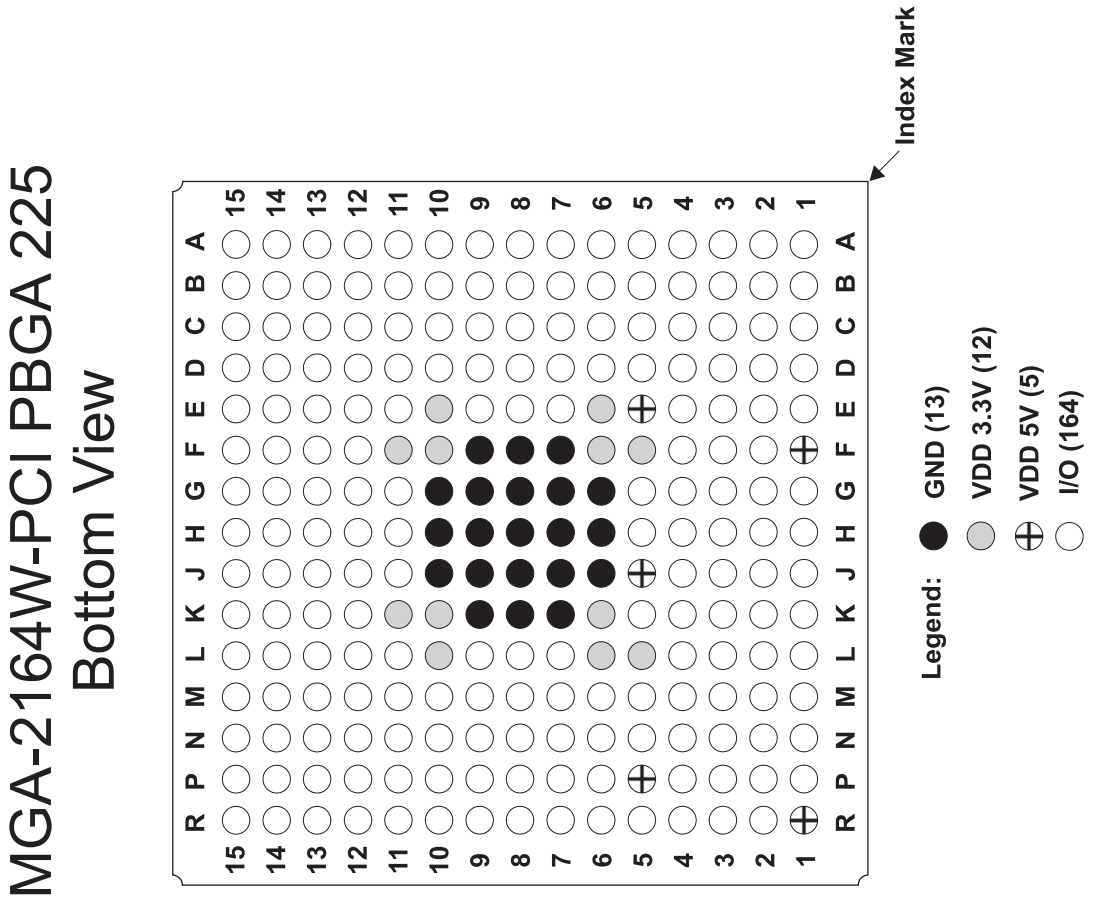


Table A-2: PCI Pinout Legend (Bottom View)

-	R	P	N	M	L	K	J	H	G	F	E	D	C	B	A
15	MDSF <2>	MDSF <0>	MOE <3>/	MSOE <1>/	MDQ <23>	Reserved	MDQ <0>	HIZ/	MDQ <50>	Reserved	MDQ <49>	MDQ <25>	MVREQN	VIDRST	Reserved
14	MDQ <19>	MDQ <18>	MDSF <1>	MDQ <20>	MDQ <22>	MDQ <5>	MDQ <2>	Reserved	MDQ <62>	MDQ <58>	MDQ <57>	MDQ <45>	VCLKSL <0>	VCBLNK/	VODD
13	MDQ <16>	MA <0>	MA <1>	MOE <0>/	MDQ <21>	MDQ <6>	MDQ <3>	MDQ <63>	MDQ <59>	MDQ <56>	MDQ <46>	VCLKSL <1>	VHSYNC/	VVSYNC/	Reserved
12	MA <3>	MDQ <17>	MA <2>	MOE <1>/	MSOE <0>/	MDQ <7>	MDQ <4>	MDQ <53>	MDQ <60>	MDQ <24>	MVGNT/	GCLK	DACRD/	ROMCS/	MDQ <40>
11	MA <6>	Reserved	MA <5>	MA <4>	MOE <2>/	VDD	MDQ <1>	MDQ <61>	MDQ <48>	VDD	DACWT/	EXTRST/	MDQ <41>	MDQ <43>	MDQ <54>
10	MRAS <1>/	MSC	MA <8>	MA <7>	VDD	VDD	GND	GND	GND	VDD	VDD	MDQ <42>	MDQ <52>	Reserved	MDQ <15>
9	MCAS <1>/	MCAS <2>/	VCLK	MCAS <0>/	MRAS <0>/	GND	GND	GND	GND	GND	MDQ <13>	MDQ <14>	MDQ <27>	Reserved	MDQ <26>
8	MCAS <3>/	MBE <3>/	Reserved	EXTINT/	VLDCLK	GND	GND	GND	GND	GND	MDQ <10>	MDQ <11>	MDQ <12>	MDQ <28>	Reserved
7	MBE <1>/	MBE <0>/	MBE <2>/	MBE <7>/	EXTCS/	GND	GND	GND	GND	GND	MDQ <9>	MDQ <8>	Reserved	MDQ <29>	MDQ <30>
6	MBE <5>/	MBE <4>/	MBE <6>/	PCBE <0>	VDD	VDD	GND	GND	GND	VDD	VDD	MDQ <33>	Reserved	MDQ <51>	MDQ <31>
5	PAD <0>/	VDD5V	PAD <2>	Reserved	VDD	PPAR	VDD5V	PAD <19>	PAD <23>	VDD	VDD5V	MDQ <38>	MDQ <35>	MDQ <34>	MDQ <32>
4	PAD <4>	PAD <6>	PAD <1>	PAD <11>	PAD <14>	Reserved	PFRAME/	PAD <17>	PAD <25>	PAD <29>	PAD <30>	PINTA/	Reserved	MDQ <36>	MDQ <37>
3	PAD <9>	PAD <5>	PAD <7>	PAD <10>	PSTOP/	PTRDY/	PAD <16>	PAD <21>	Reserved	PAD <22>	PAD <28>	PCLK	MDQ <55>	MDQ <39>	MDQ <44>
2	PAD <3>	PAD <8>	PAD <13>	PAD <15>	PCBE <1>/	Reserved	Reserved	PAD <20>	PIDSEL	PAD <27>	PAD <24>	Reserved	PGNT/	MDQ <47>	Reserved
1	VDD5V	PAD <12>	Reserved	PDEVSEL/	PIRDY/	PCBE <2>/	Reserved	PAD <18>	PCBE <3>/	VDD5V	PAD <31>	PAD <26>	PREQ/	PRST/	LFT

A.3 AGP Pinout Illustration and Table

The illustration below shows the locations of the MGA-2164W-AGP's 225 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

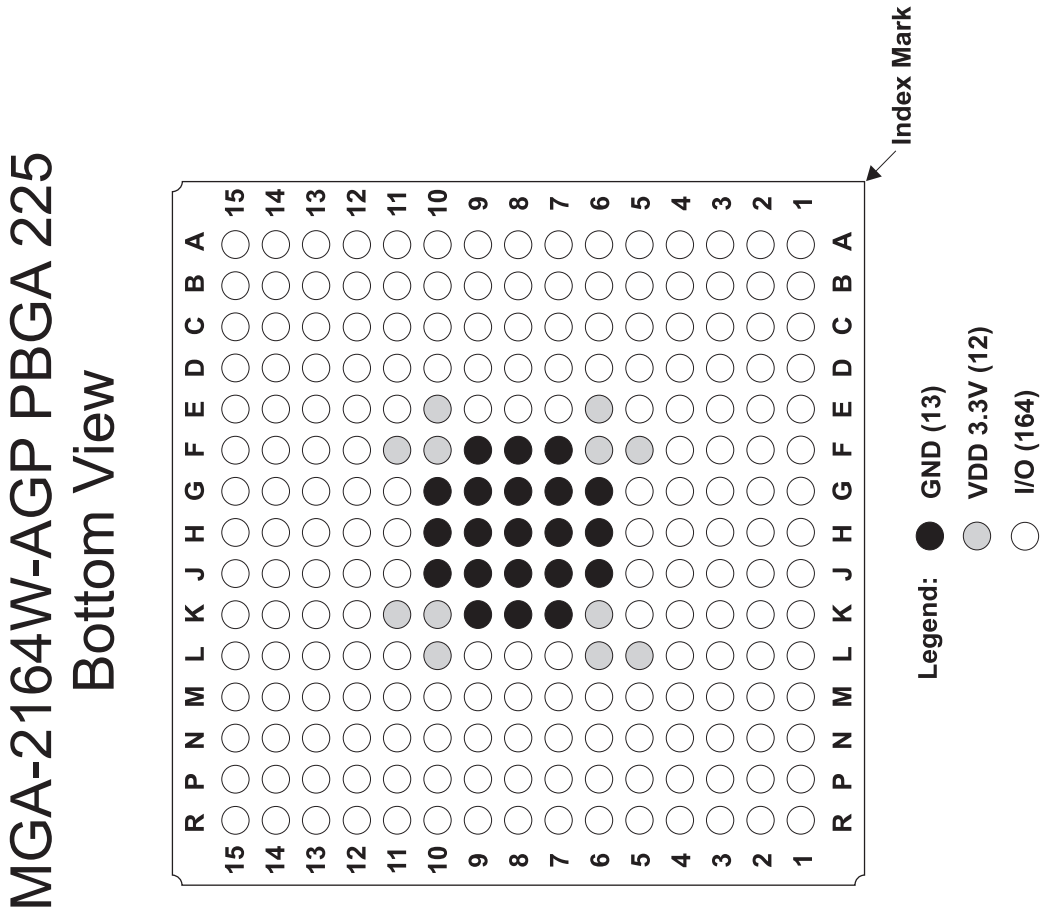


Table A-3: AGP Pinout Legend (Bottom View)

-	R	P	N	M	L	K	J	H	G	F	E	D	C	B	A
15	MDSF <2>	MDSF <0>	MOE <3>/	MSOE <1>/	MDQ <23>	Reserved	MDQ <0>	HIZ/	MDQ <50>	Reserved	MDQ <49>	MDQ <25>	MVREQN/	VIDRST	Reserved
14	MDQ <19>	MDQ <18>	MDSF <1>	MDQ <20>	MDQ <22>	MDQ <5>	MDQ <2>	Reserved	MDQ <62>	MDQ <58>	MDQ <57>	MDQ <45>	VCLKSL <0>	VCBLNK/	VODD
13	MDQ <16>	MA <0>	MA <1>	MOE <0>/	MDQ <21>	MDQ <6>	MDQ <3>	MDQ <63>	MDQ <59>	MDQ <56>	MDQ <46>	VCLKSL <1>	VHSYNC/	VVSYNC/	Reserved
12	MA <3>	MDQ <17>	MA <2>	MOE <1>/	MSOE <0>/	MDQ <7>	MDQ <4>	MDQ <53>	MDQ <60>	MDQ <24>	MVGNT/	GCLK	DACRD/	ROMCS/	MDQ <40>
11	MA <6>	Reserved	MA <5>	MA <4>	MOE <2>/	VDD	MDQ <1>	MDQ <61>	MDQ <48>	VDD	DACWT/	EXTRST/	MDQ <41>	MDQ <43>	MDQ <54>
10	MRAS <1>/	MSC	MA <8>	MA <7>	VDD	VDD	GND	GND	GND	VDD	VDD	MDQ <42>	MDQ <52>	Reserved	MDQ <15>
9	MCAS <1>/	MCAS <2>/	VCLK	MCAS <0>/	MRAS <0>/	GND	GND	GND	GND	GND	MDQ <13>	MDQ <14>	MDQ <27>	Reserved	MDQ <26>
8	MCAS <3>/	MBE <3>/	Reserved	EXTINT/	VLDCLK	GND	GND	GND	GND	GND	MDQ <10>	MDQ <11>	MDQ <12>	MDQ <28>	Reserved
7	MBE <1>/	MBE <0>/	MBE <2>/	MBE <7>/	EXTCS/	GND	GND	GND	GND	GND	MDQ <9>	MDQ <8>	Reserved	MDQ <29>	MDQ <30>
6	MBE <5>/	MBE <4>/	MBE <6>/	PCBE <0>/	VDD	VDD	GND	GND	GND	VDD	VDD	MDQ <33>	Reserved	MDQ <51>	MDQ <31>
5	PAD <0>	Reserved	PAD <2>	Reserved	VDD	PPAR	Reserved	PAD <19>	PAD <23>	VDD	Reserved	MDQ <38>	MDQ <35>	MDQ <34>	MDQ <32>
4	PAD <4>	PAD <6>	PAD <1>	PAD <11>	PAD <14>	Reserved	PFRAME/	PAD <17>	PAD <25>	PAD <29>	PAD <30>	PINTA/	Reserved	MDQ <36>	MDQ <37>
3	PAD <9>	PAD <5>	PAD <7>	PAD <10>	PSTOP/	PTRDY/	PAD <16>	PAD <21>	Reserved	PAD <22>	PAD <28>	PCLK	MDQ <55>	MDQ <39>	MDQ <44>
2	PAD <3>	PAD <8>	PAD <13>	PAD <15>	PCBE <1>/	Reserved	Reserved	PAD <20>	PIDSEL	PAD <27>	PAD <24>	Reserved	PGNT/	MDQ <47>	Reserved
1	Reserved	PAD <12>	Reserved	PDEVSEL/	PIRDY/	PCBE <2>/	Reserved	PAD <18>	PCBE <3>/	Reserved	PAD <31>	PAD <26>	PREQ/	PRST/	LFT

Table A-4: Buffer Assignment

Pin	*MGA-2164W-PCI*	*MGA-2164W-AGP*	*Notes*
PAD <31:0>	IO-PCI33	IO-12	
PCBE <3:0>/	IO-PCI33	IO-12	
PCLK	I-PCI33	I-0	
DEVSEL/	IO-PCI33	IO-12	
PFRAME/	IO-PCI33	IO-12	
PGNT/	I-PCI33	I-0	
PIDSEL	I-PCI33	I-0	
PINTA/	O-PCI33	O-12	(1)
PIRDY	IO-PCI33	IO-12	
PPAR	O-PCI33	O-12	(1)
PREQ/	O-PCI33	O-12	(1)
PRST/	I-PCI33	I-0	
PSTOP/	IO-PCI33	IO-12	
PTRDY/	IO-PCI33	IO-12	
ROMCS/	O-6	O-6	(1)
DACRD/	O-6	O-6	(1)
DACWT/	O-6	O-6	(1)
EXTRST/	O-6	O-6	(1)
EXTCS/	O-6	O-6	(1)
EXTINT/	I-0	I-0	
MDQ<63:0>	IO-9-5V	IO-9-5V	
MA<8:0>	O-12	O-12	(1)
MRAS<1:0>/	O-12	O-12	(1)
MCAS<3:0>/	O-12	O-12	(1)
MOE<3:0>/	O-12	O-12	(1)
MBE<7:0>/	O-12	O-12	(1)
MDSF<2:0>	O-12	O-12	(1)
GCLK	I-0-5V	I-0-5V	
MVREQ/	I-S	I-S	
MVGNT/	O-9	O-9	(1)
VCLKSL<1:0>	O-3	O-3	(1)
VCLK	I-S	I-S	
VIDRST	I-0-5V	I-0-5V	
VHSYNC/	O-6	O-6	(1)
VVSYNC/	O-6	O-6	(1)
VCBLNK/	O-6	O-6	(1)
VLDCLK	O-6	O-6	(1)
VODD	O-6	O-6	(1)
MSC	O-24	O-24	(1)
MSOE<1:0>/	O-6	O-6	(1)
HIZ/	I-0	I-O	

(1) Reconfigured in Input when hiz/ = low (Nand Tree test).

A.4 Electrical Specification

A.4.1 DC Specifications

Table A-5: 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-12, I-0, I-S	Vi < VDD3 + 0.5V	-0.5	4.6	V	
	IO-9-5V	Vi < VDD3 + 3.0V	-0.5	6.6	V	
	I-PCI 33, IO-PCI 33	Vi < VDD5 + 0.5V	-0.5	6.6	V	(1)
V_O	Output Voltage					(2)
	O-3, O-6,O-9, O-12, O-24	Vo < VDD3 + 0.5V	-0.5	4.6	V	
	IO-9-5V	Vo < VDD3 + 3.0V	-0.5	6.6	V	
	IO-PCI 33, O-PCI 33	Vo < VDD5 + 0.5V	-0.5	6.6	V	(1)
V_N	Negative Trigger Voltage					
	I-S		1.3	1.5	V	
V_P	Positive Trigger Voltage					
	I-S		1.5	1.8	V	
V_H	Hysteresis Voltage					
	I-S		0.22	0.33	V	
I_O	Output Current					(2)
	O-3			10	mA	
	O-6			20	mA	
	O-9			30	mA	
	O-12, IO-12			40	mA	
	O-24			75	mA	
	IO-PCI33, O-PCI33			?	mA	
T_A	Operating Temperature		0	55	^oC	
T_{STG}	Storage Temperature		-65	150	^oC	

(1) MGA-2164W-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-6: Recommended Operating Conditions

Symbol	*Parameter*	*Min.*	*Max.*	*Units*
VDD5	Power Supply	4.75	5.25	V
VDD3		3.0	3.6	V
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
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
t_r	Input Rise Time	0	200	ns
t_f	Input Fall Time	0	200	ns

Table A-7: DC Characteristics
(VDD3 = 3.3 ±0.3V, VDD5 = 5.0 ±0.25V, TA = 0 to 55°)

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)
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)
V_{OL}	Low-Level Output Voltage	I_{OL} = 0 mA			0.1	V	

Table A-7: DC Characteristics
(VDD3 = 3.3 ±0.3V, VDD5 = 5.0 ±0.25V, TA = 0 to 55°)

Symbol	*Parameter*	*Conditions*	*Min.*	*Typ.*	*Max.*	*Units*	*Notes*
V_{OH}	High-Level Output Voltage	I_{OH} = 0 mA	VDD3 - 0.1			V	
θ_{JA}	Junction-to-Air Thermal Coefficient	No Air Flow			30	°c/w	(3)
C_{PIN}	Pin Capacitance	F = 1 MHz			7	pF	
ICC3	VDD3 Supply Current			450		mA	
ICC5	VDD5 Supply Current			0		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-7)(MGA-2164W-PCI only).

(3) All GND ball connected to PCB ground plane and all VDD3 balls connected to PCB VDD plane.

Figure A-3: AGP BufferV/I Curve Pull-down (Best Case)

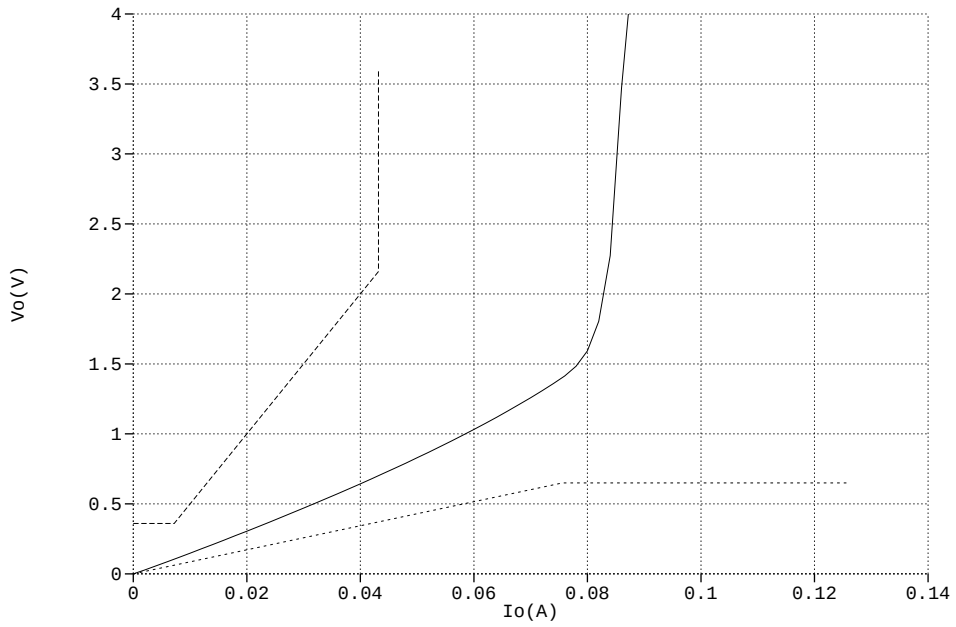


Figure A-4: AGP Buffer V/I Curve Pull-Down (Worst Case)

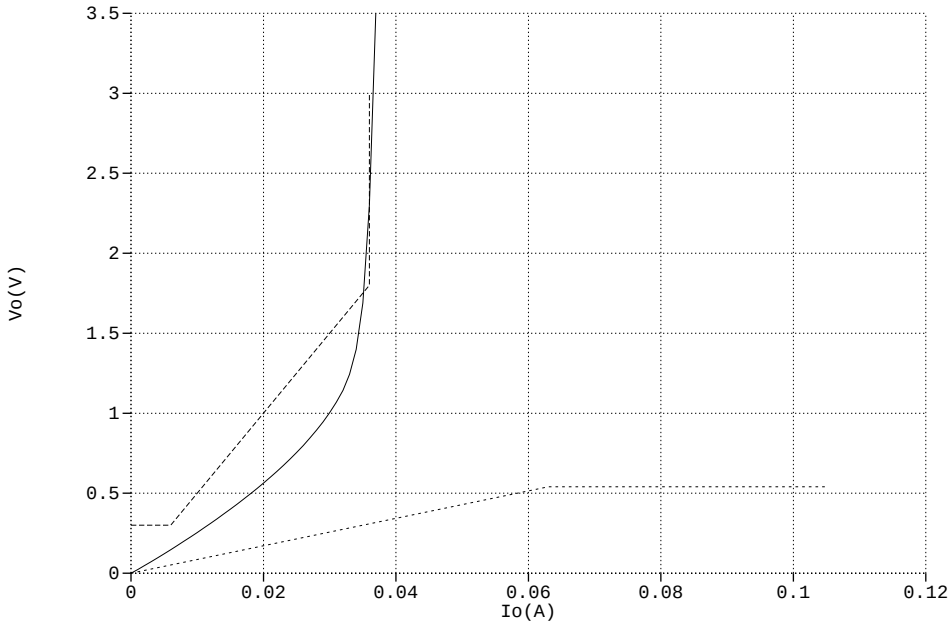


Figure A-5: AGP Buffer V/I Curve Pull-Up (Best Case)

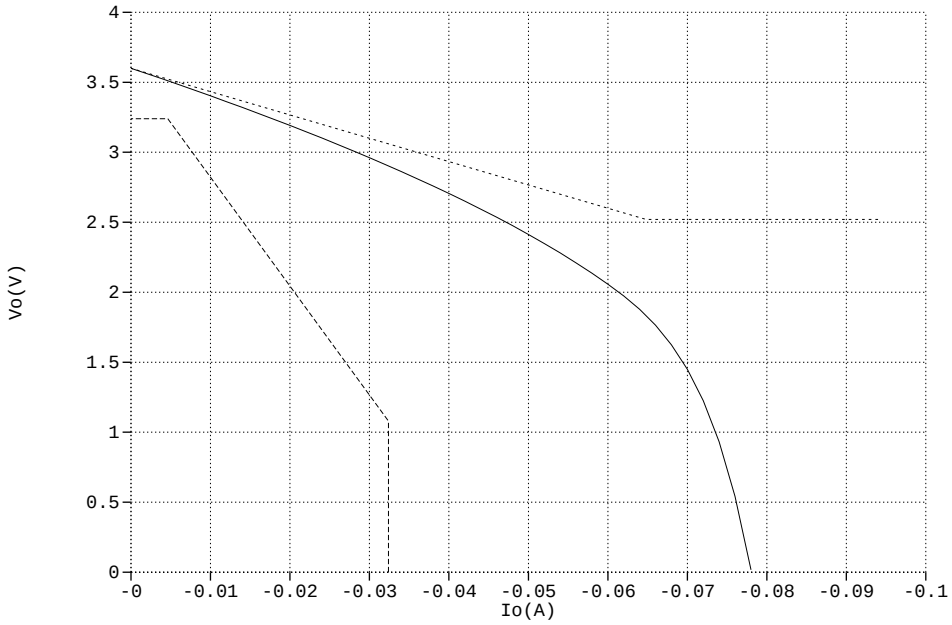


Figure A-6: AGP Buffer V/I Curve Pull-Up (Worst Case)

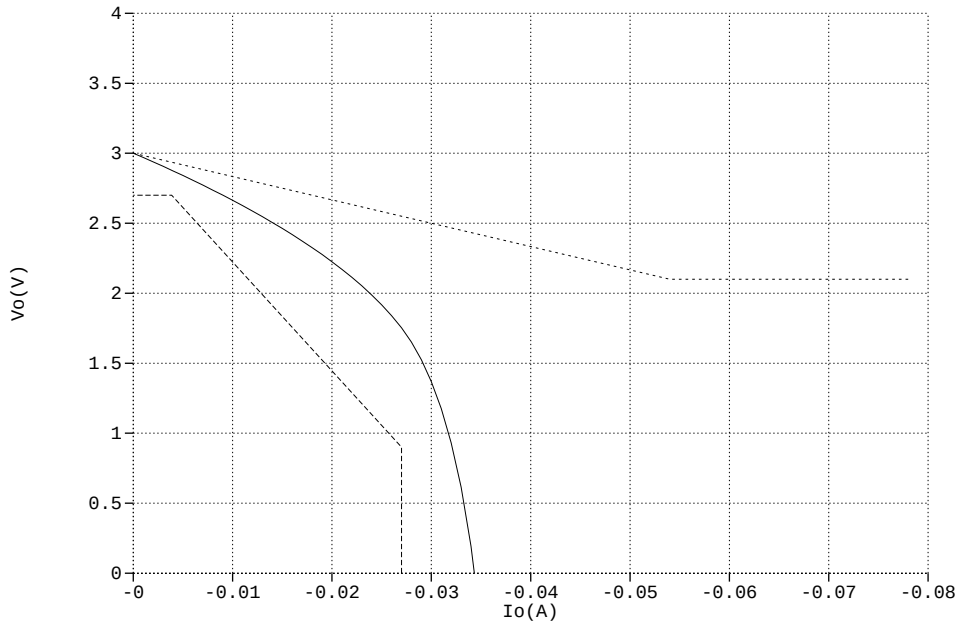
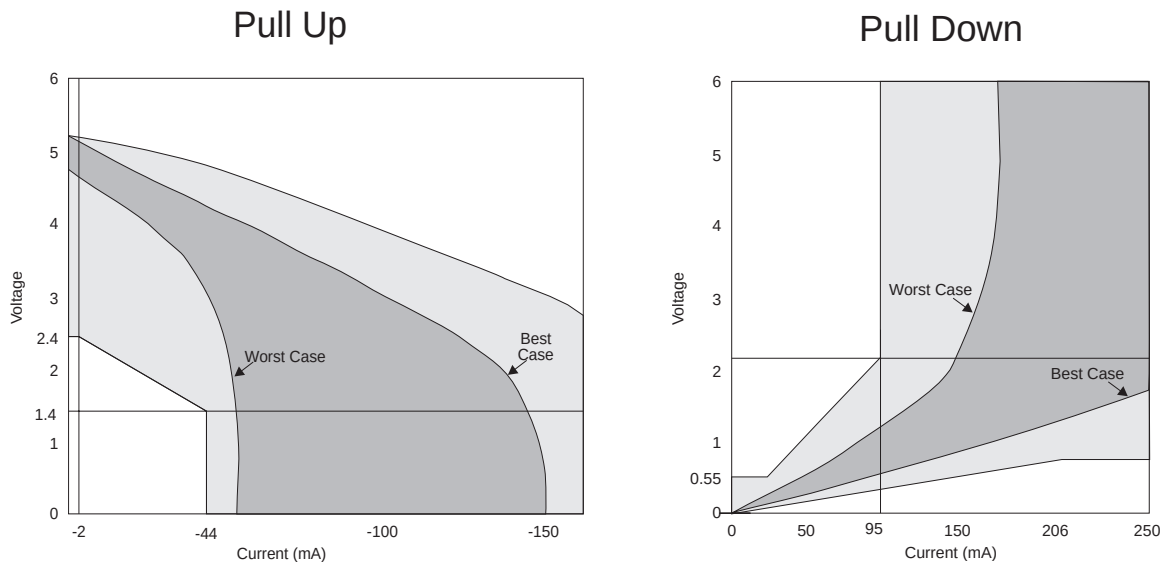


Figure A-7: V/I Curves for O-PCI33 and IO-PCI33 Buffers



A.4.2 AC Specifications

The following timing tables are presented from the user's point: the tables indicate the timing parameters a device (WRAM, BIOS ROM, RAMDAC, or other external device) must meet to work properly with the MGA-2164W chip.

The ROM Read and Write cycle, the RAMDAC Read and Write cycle, the External Read and Write cycle, and all the WRAM cycles assume a minimum **gclk** of 16.0 nS (a minimum of 17.3 nS, if there is more than 8 megabytes of WRAM installed).

The video interface timing gives the chip actual timing. The designer must verify that the serial port of the WRAM and the RAMDAC used respect each other's timing.

> ***Note:*** It is important that the **msoe**<1:0>/ lines always deactivate for 1 **vclk** cycle before reactivating: this ensures that there is no conflict on the serial bus.The **msoe**<1:0>/ lines switch only during the horizontal blank (assuming the **crtc** is properly programmed).

A.4.2.1 Host Interface Timing

Figure A-8: PCI 33 MHz Waveform (MGA-2164W-PCI only)

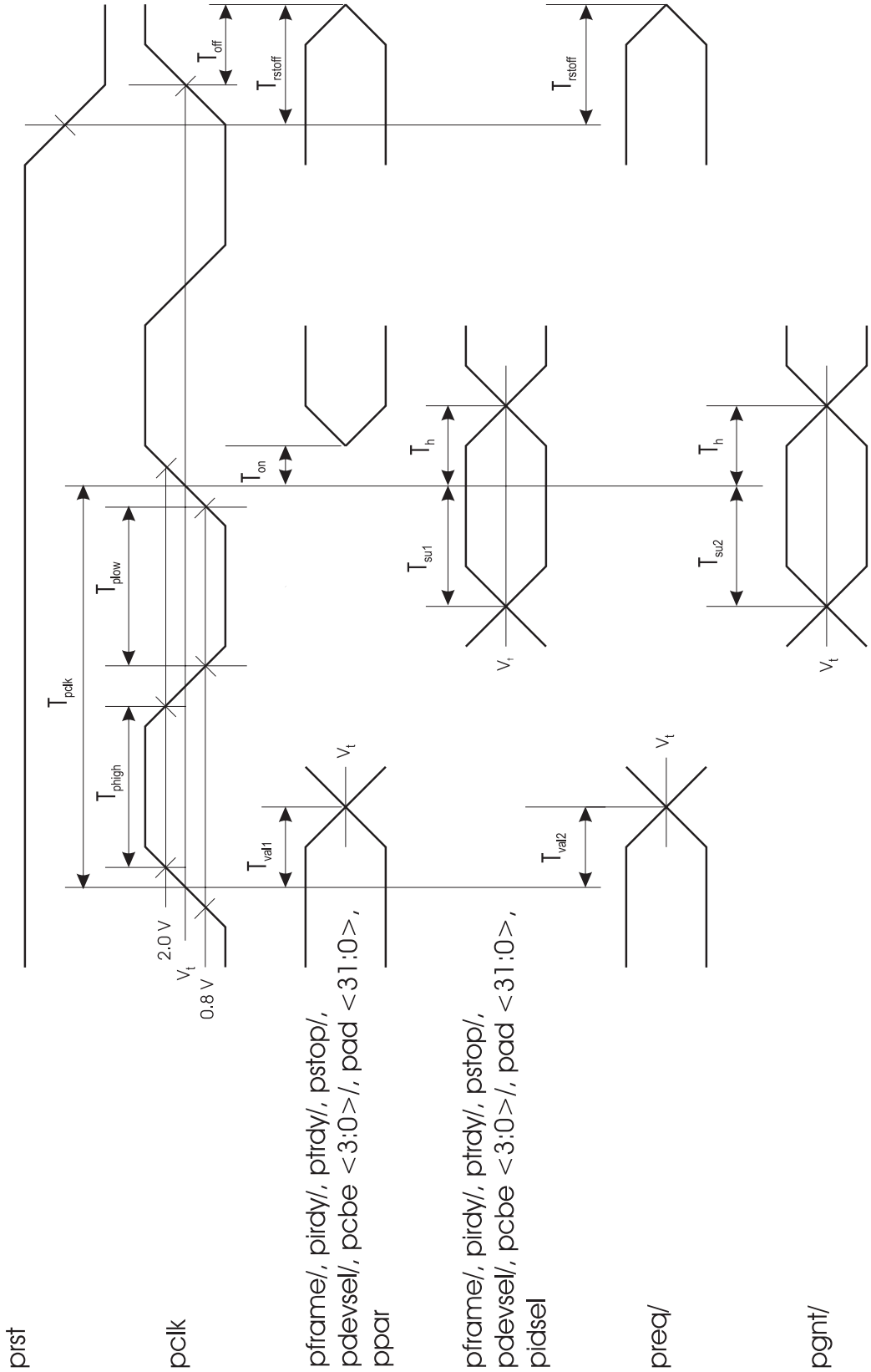


Table A-8: PCI 33 MHz 5V Signaling Environment Timing (MGA-2164W-AGP only)[1]

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 Timing (MGA-2164W-AGP only)

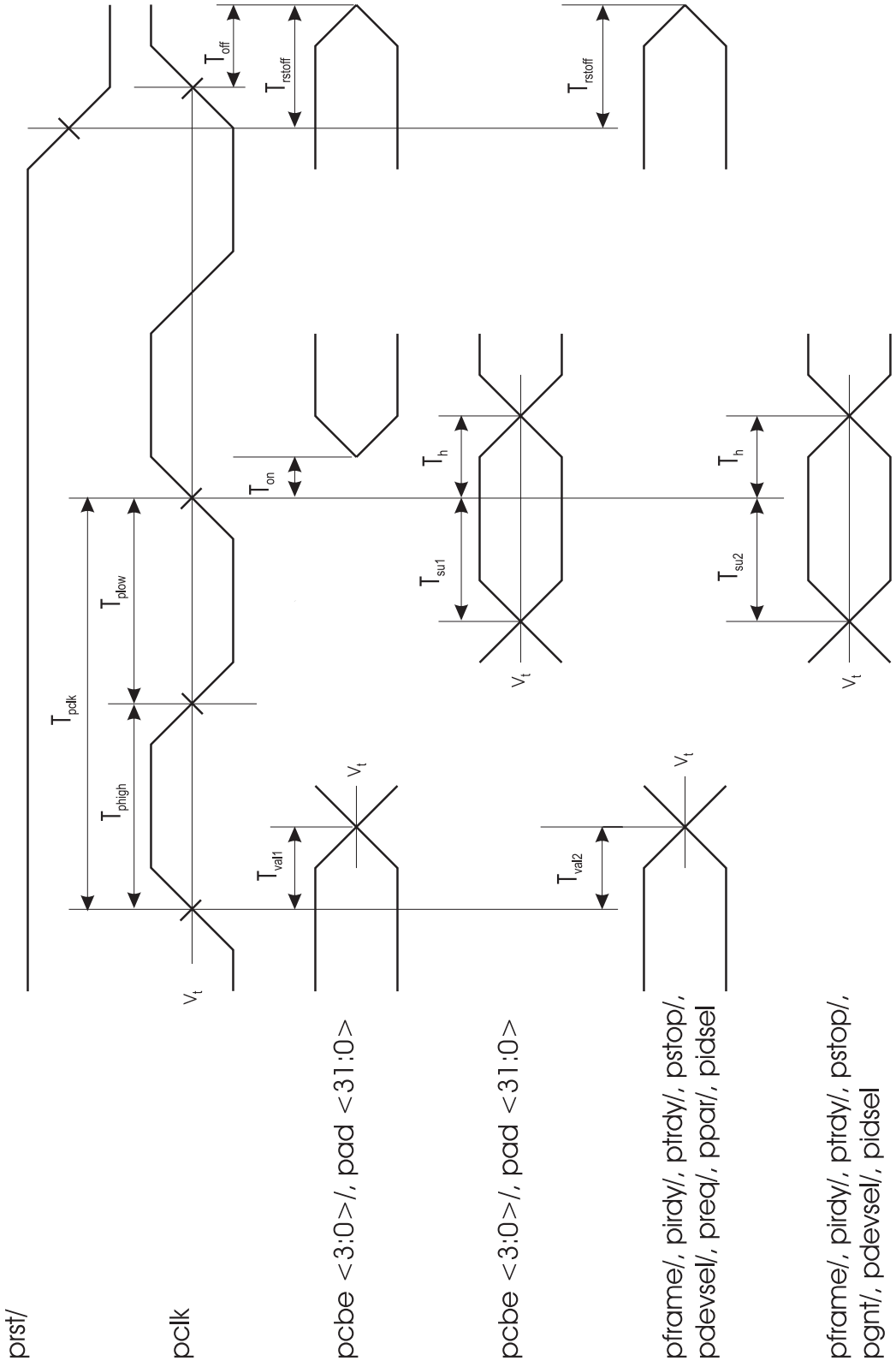


Table A-9: AGP1X Timing (MGA-2164W-AGP only)(1)

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	1.5	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	
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 pframe/, pirdy/, ptrdy/, pstop/, pdevsel/, preq/, and ppar.

(3) Applies only to pad<31:0> and pcbe<3:0>/.

(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 pframe/, pirdy/, ptrdy/, pstop/, pdevsel/, pgnt/, ppar, and pidsel.

A.4.2.2 GCLK Timing

Figure A-10: GCLK Waveform

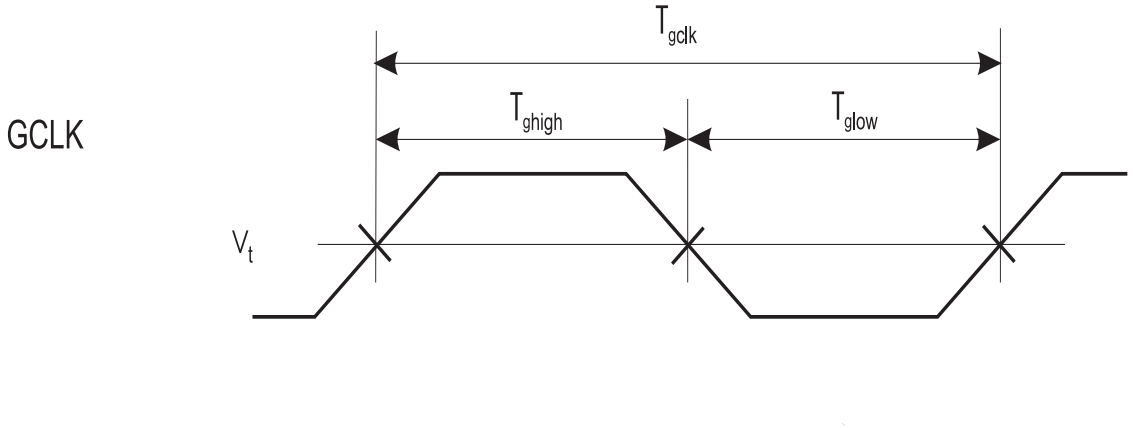


Table A-10: GCLK Timing Requirements(1)

Symbol	*Parameter*	*2, 4 Mbytes*		*8 Mbytes*		*10, 12 Mbytes*		*14, 16 Mbytes*		*Unit*
		Min	*Max*	*Min*	*Max*	*Min*	*Max*	*Min*	*Max*	
T_{gclk}	Cycle Time	17.9	-	20.0	-	22.2	-	23.3	-	nS
T_{ghigh}	High Time	8.0	-	9.0	-	10.0	-	10.5	-	nS
T_{glow}	Low Time	8.0	-	9.0	-	10.0	-	10.5	-	nS

(1) $V_t = 0.5\ V_{DD}$

A.4.2.3 External Device Timing

Figure A-11: ROM Write Waveform

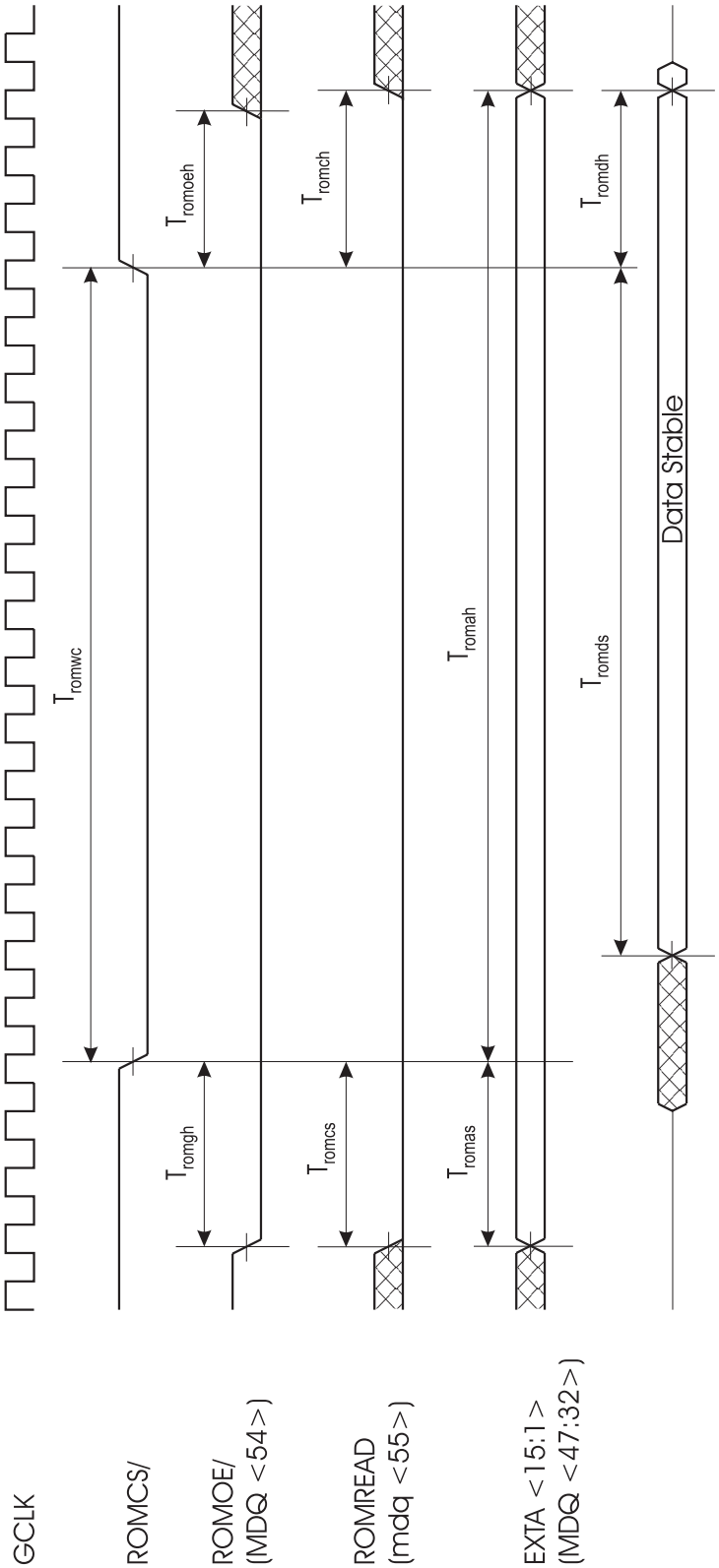


Figure A-12: ROM Read Waveform

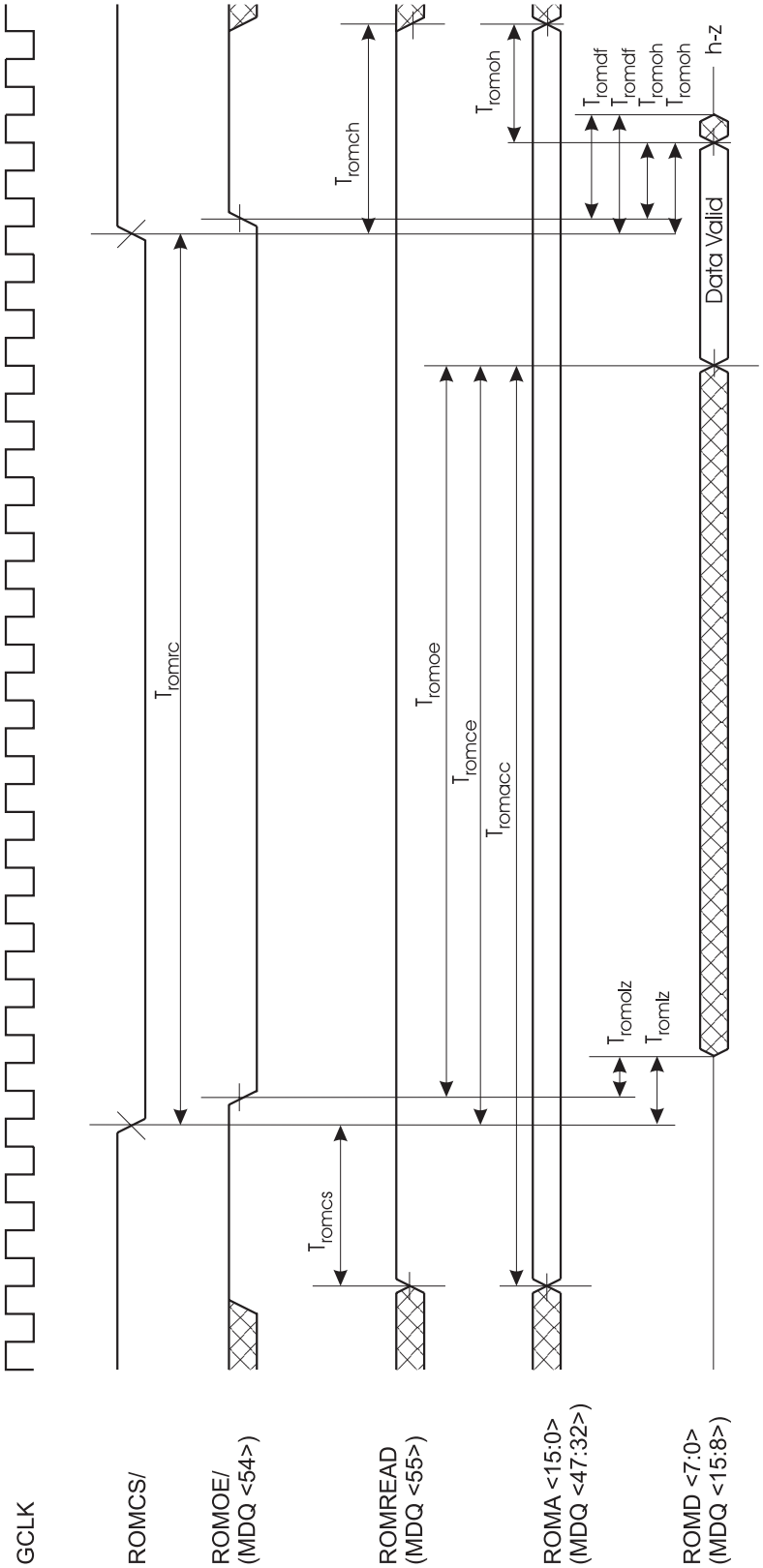


Table A-11: ROM Read and Write Timing(1)

Symbol	*Parameter*	*Min*	*Max*	*Unit*	*Notes*
T_{romrc}	ROMCS/ low time (Read Cycle)	215		nS	
T_{romcs}	ROMREAD setup time to ROMCS/ falling	35		nS	
T_{romch}	ROMREAD hold time from ROMCS/ rising	35		nS	
T_{romlz}	Delay from ROMCS/ to ROMDQ <7:0> low-z	0		nS	
T_{romce}	Delay from ROMCS/ to ROMDQ <7:0> valid		215	nS	
T_{romoh}	Delay from ROMCS/ or ROMOE/ or EXTA <15:0> to ROMDQ <7:0> invalid. (The earliest signal)	0		nS	
T_{romdf}	Delay from ROMCS/ or ROMOE/ to ROMDQ <7:0> hi-z. (The earliest signal)		75	nS	
T_{romolz}	Delay from ROMOE/ to ROMDQ <7:0> low z	0		nS	
T_{romoe}	Delay from ROMOE/ to ROMDQ <7:0> valid		115	nS	
T_{romacc}	Access time from exta <15:0>		215	nS	
T_{romwc}	ROMCS/ low time (Write Cycle)	215		nS	
T_{romgh}	ROMOE/ setup time ROMCS/ falling	15		nS	
T_{romoeh}	ROMOE/ hold time from ROMCS/ rising	25		nS	
T_{romds}	ROMDQ <7:0> setup time to ROMCS/ rising	65		nS	
T_{romdh}	ROMDQ <7:0> hold time from ROMCS/ rising	25		nS	
T_{romas}	EXTA <15:0> setup time to ROMCS/ falling	15		nS	
T_{romah}	EXTA <15:0> hold time from ROMCS/ falling	90		nS	

(1) All timings refer to 0.5 V_{DD}.
The hi-z condition occurs when the total current delivered through the component pin is less than or equal to the leakage current specification.
Load condition: ROMCS/ = 30 pF, ROMOE/ =95 pF, ROMREAD = 95 pF, EXTA <15:0> = 95 pF, ROMDQ <7:0> = 95 pF.

Figure A-13: RAMDAC Write Cycle

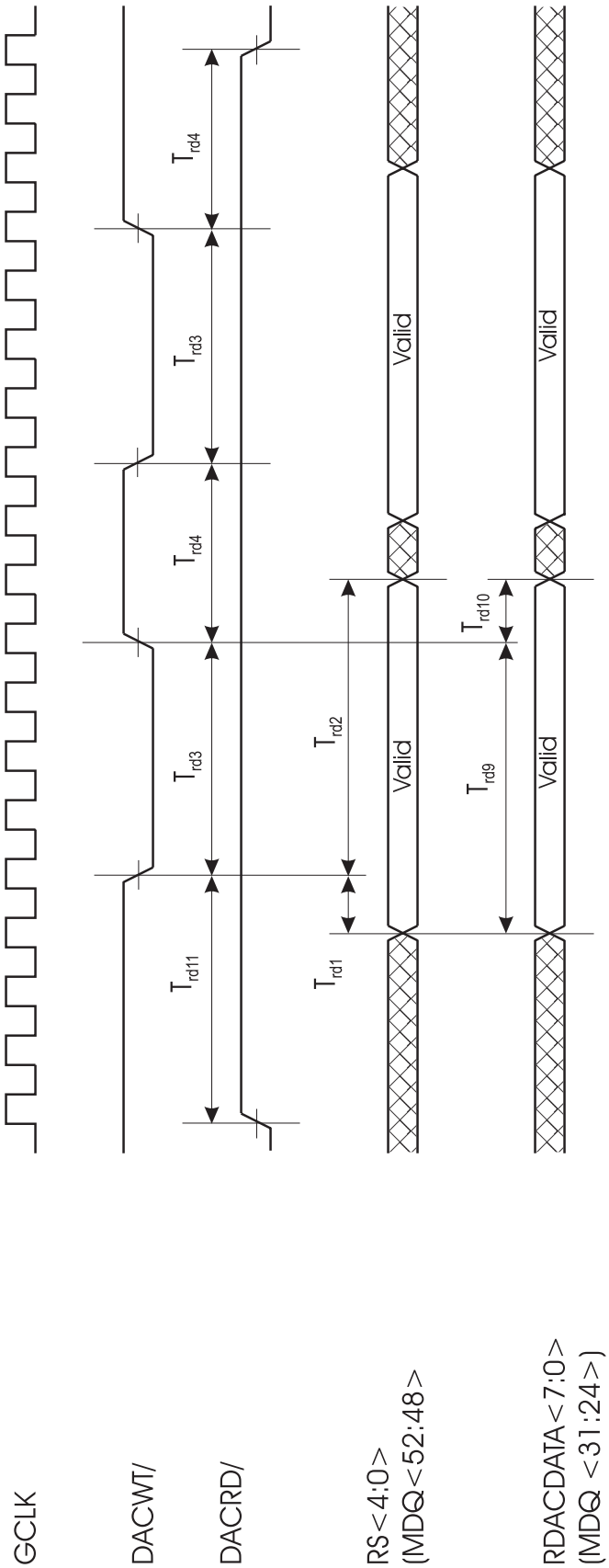


Figure A-14: RAMDAC Read Cycle

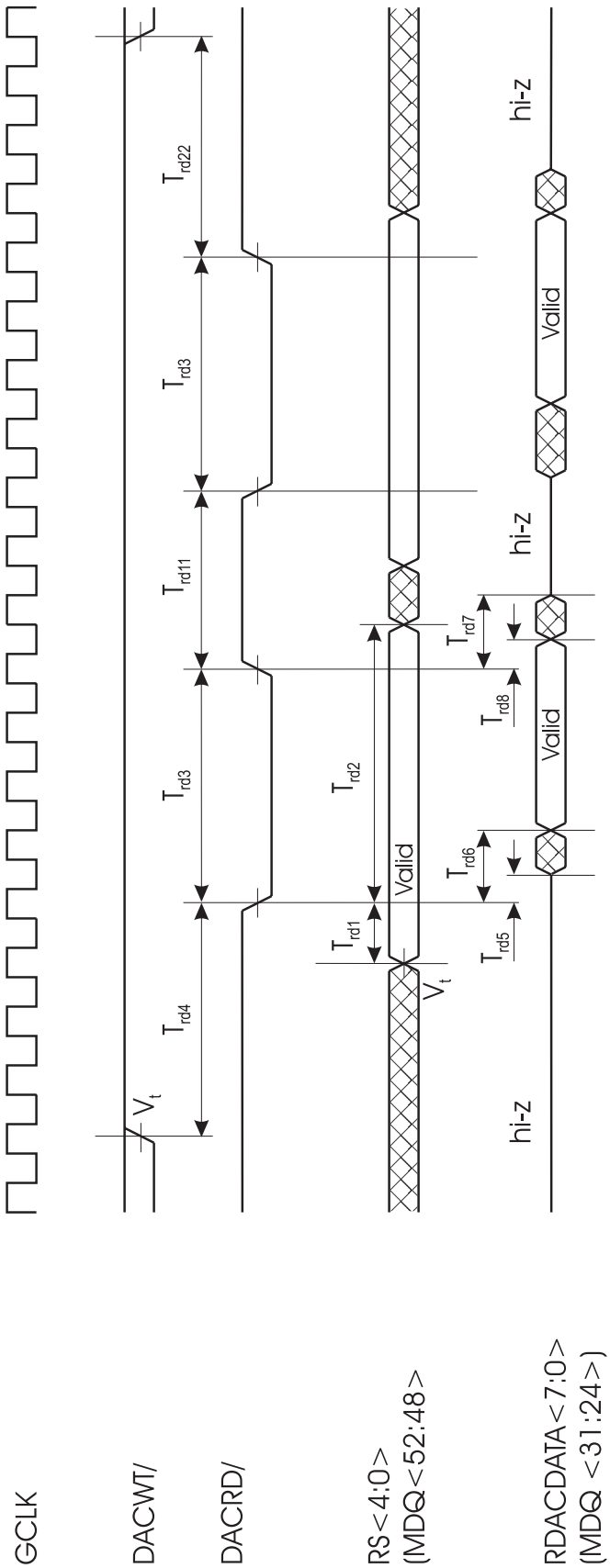


Table A-12: RAMDAC Read and Write Timing(1),(2),(3)

Symbol	*Parameter*	*Min*	*Max*	*Unit*	*Notes*
T_{rd1}	RAMDAC address setup to DACRD/ or DACWT/ falling.	10		nS	
T_{rd2}	RAMDAC address hol from DACRD/ or DACWT/ falling.	10		nS	
T_{rd3}	DACRD/ or DACWT/ low time.	50		nS	
T_{rd4}	DACWT/ to DACWT/, DACWT/ to DACRD/ high time.	30		nS	
T_{rd5}	DACRD/ low to RDACDATA low-z.	0		nS	
T_{rd6}	DACRD/ low to RDACDATA valid.	-	40	nS	
T_{rd7}	DACRD/ high to RDACDATA high-z.	-	17	nS	
T_{rd8}	DACRD/ high to RDACDATA invalid.	0		nS	
T_{rd9}	RDACDATA setup to DACWT/ rising.	35		nS	
T_{rd10}	RDACDATA hold from DAWT/ rising.	0		nS	
T_{rd11}	DACRD/ to DACWT/ high time.	35	-	nS	

(1) All timings refer to 0.5 $V_{OD.}$

(2) The hi-z condition occurs when the total current delivered through the component pin is less than or equal to the leakage current specification.

(3) Load conditions: DACRD/ = 30 pF, DACWT/ = 30 pF, RS<1:0> = 95 pF, RDACDATA<7:0> = 95 pF.

Figure A-15: External Device Write Cycle

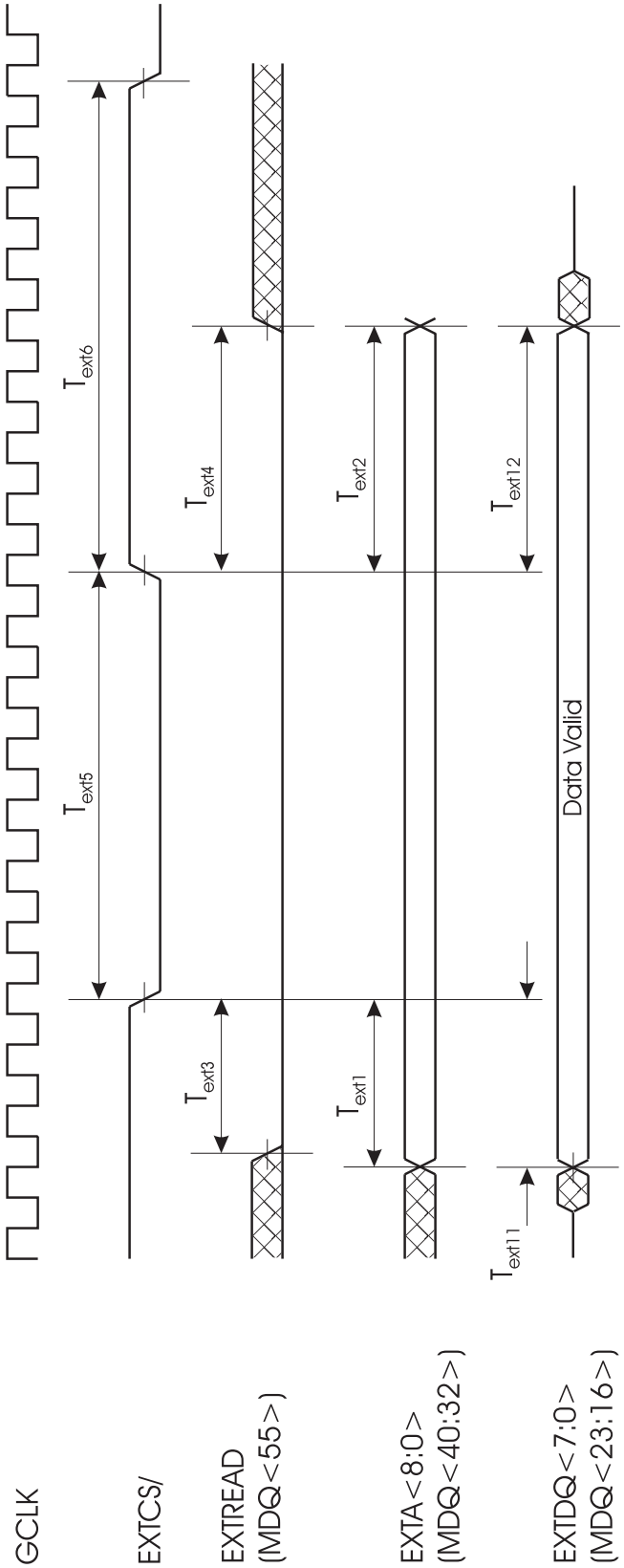


Figure A-16: External Device Read Cycle

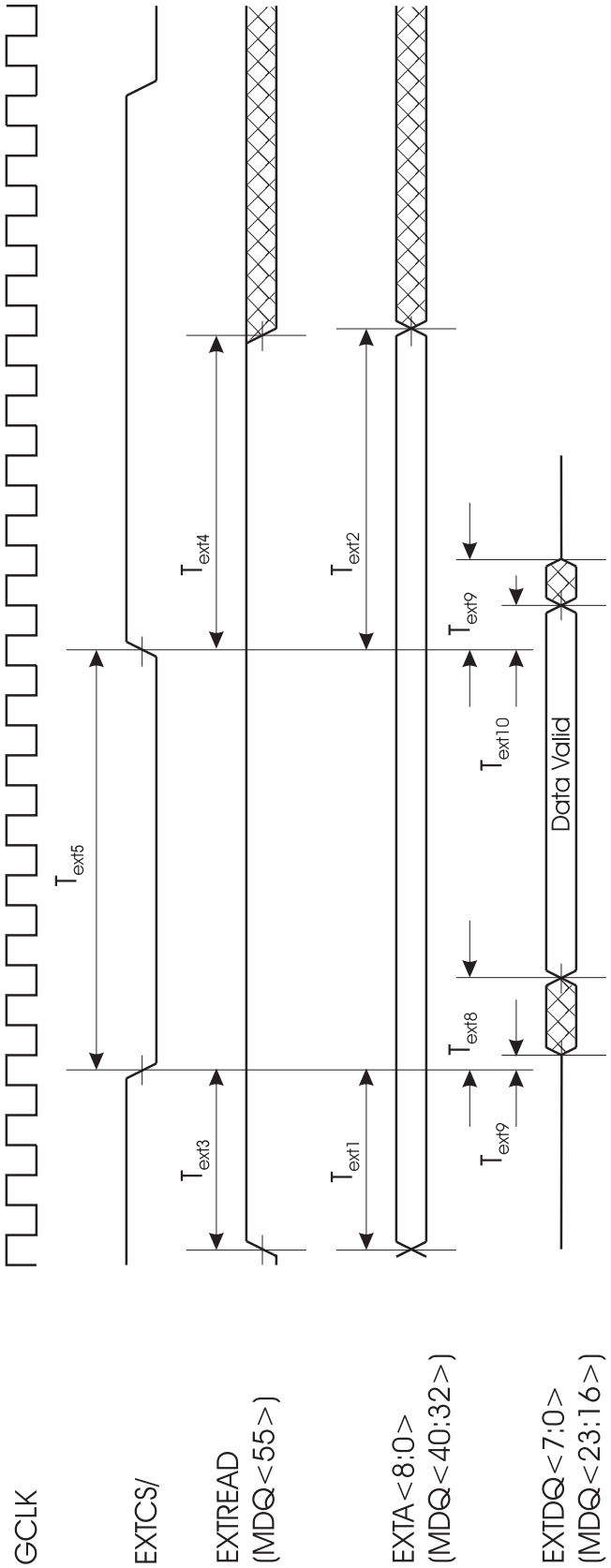


Table A-13: External Device Read and Write Timing[(1),(2),(3)]

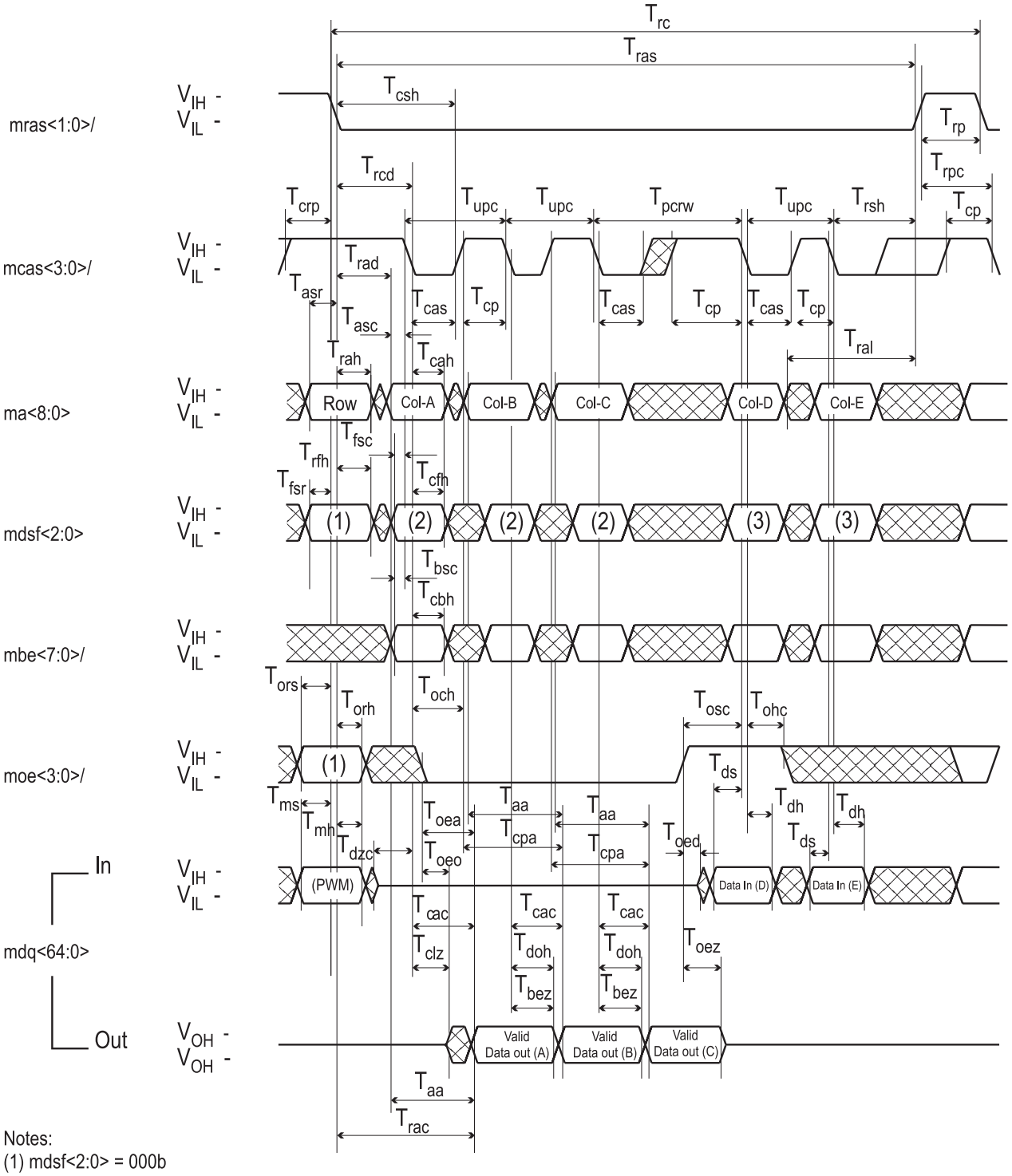
Symbol	*Parameter*	*Min*	*Max*	*Unit*	*Notes*
T_{ext1}	EXTA setup time to EXTCS/ falling	40		nS	
T_{ext2}	EXTA hold-time from EXTCS/ rising	40		nS	
T_{ext3}	EXTREAD setup time to EXTCS/ falling	40		nS	
T_{ext4}	EXTREAD hold-time from EXTCS/ rising	40		nS	
T_{ext5}	EXTCS/ low time	100		nS	
T_{ext6}	EXTCS/ high time	80		nS	
T_{ext7}	Delay from EXTCS/ low to EXTDQ low-z (read cycle only)	2		nS	
T_{ext8}	Delay from EXTCS/ low to EXTDQ low-z (read cycle only)		90	nS	
T_{ext9}	Delay from EXTCS/ high to EXTDQ hi-z		55	nS	
T_{ext10}	Delay from EXTCS/ high to EXTDQ invalid	2		nS	
T_{ext11}	EXTDQ setup time to EXTCS/ falling	-20		nS	
T_{ext12}	EXTDQ hold-time from EXTCS/ rising	45			

(1) All timings refer to 0.5 $V_{DD.}$

(2) The hi-z condition occurs when the total current delivered through the component pin is less than or equal to the leakage current specification.

(3) Load conditions: EXTCS/ = 30 pF, EXTA<8:0> = 95 pF, EXTDQ <7:0> = 95 pF.

Figure A-17: Page Read-Write/Load Cycle



Notes:
(1) mdsf<2:0> = 000b
moe<3:0> = 110b
(2) mdsf<2:0> = 110b
(3) mdsf<2:0> = 000b fblit read
001b color block mode
010b split read transfer
011b fblit write
101b background, foregraound or plane write mask (PWM) load
111b write cycle

Figure A-18: Page Read Cycle

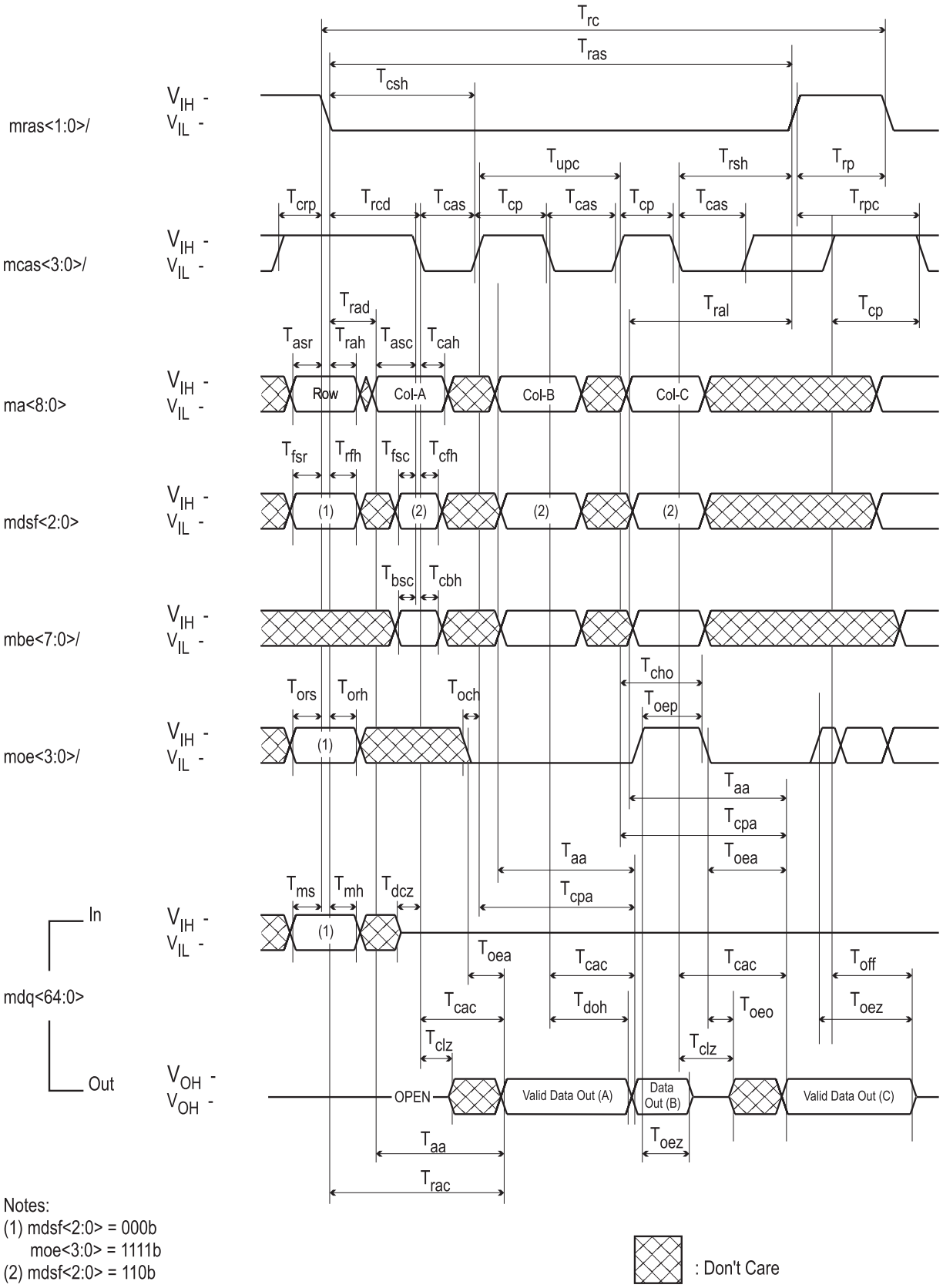
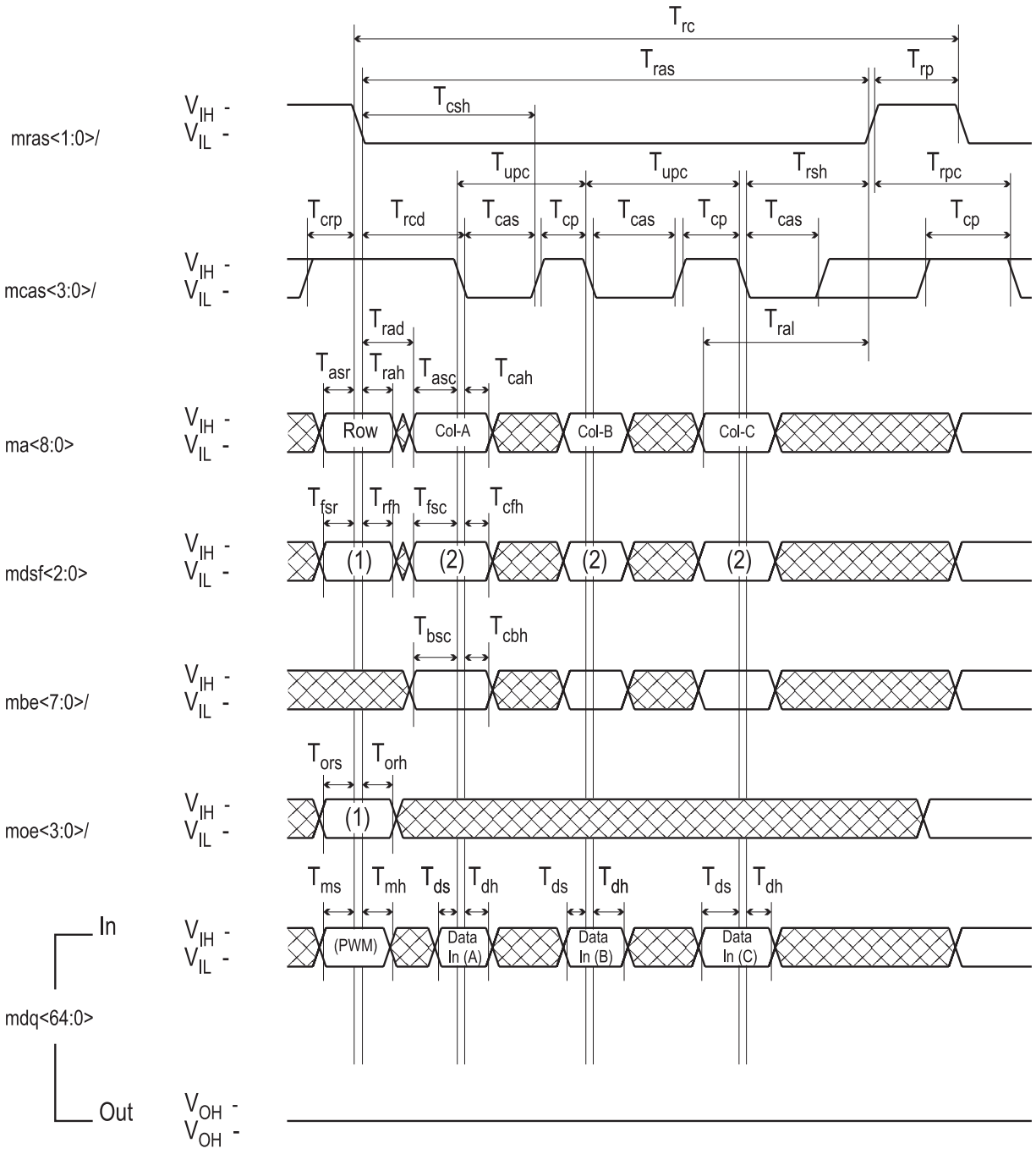


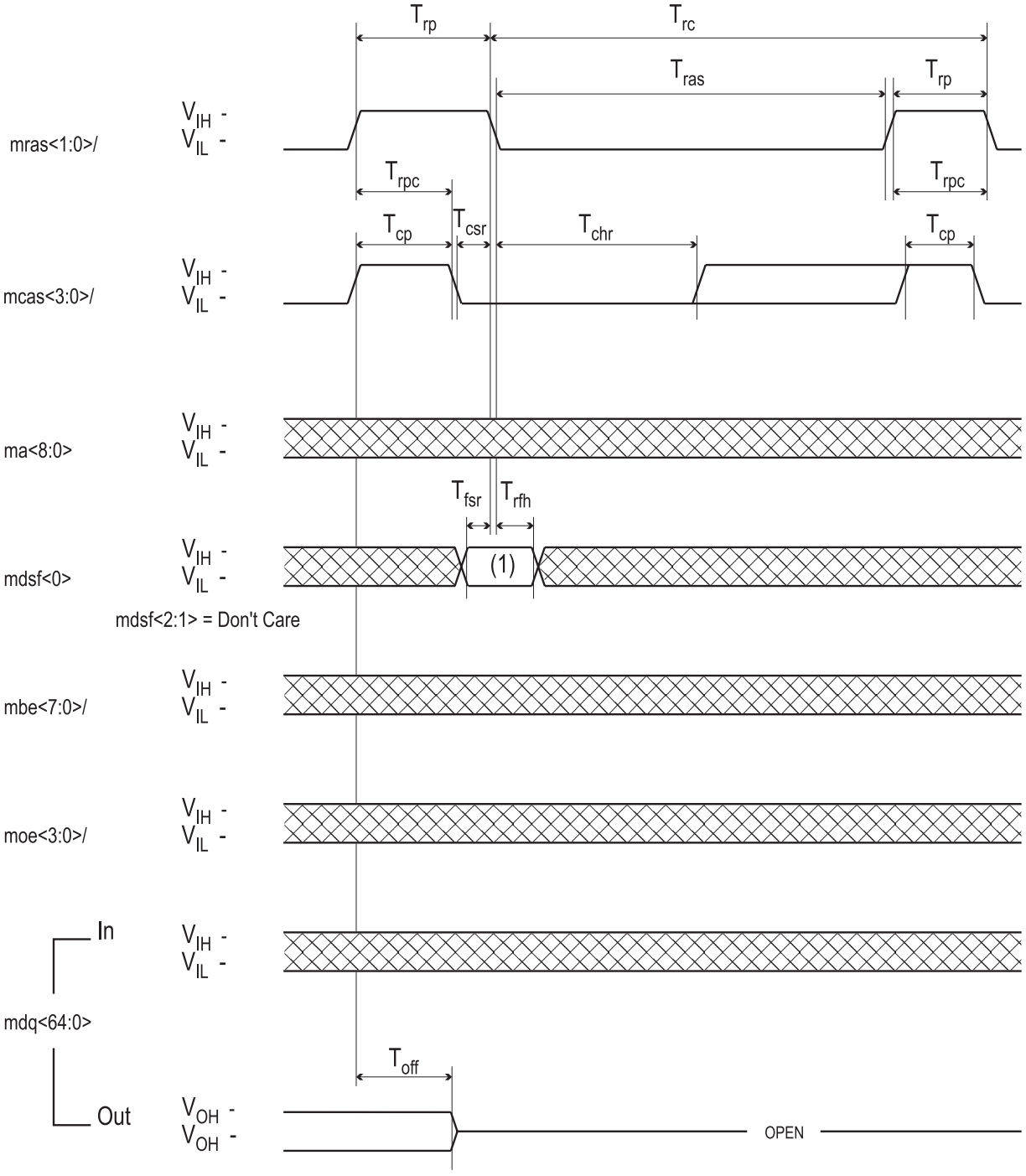
Figure A-19: Page Write/Load Cycle



Notes:
(1) mdsf<2:0> = 000b
moe<3:0> = 1111b
(2) mdsf<2:0> = 000b fblit read
001b color block mode
010b split read transfer
011b fblit write
101b background, foreground or plane write mask (PWM) load
111b write cycle

: Don't Care

Figure A-20: CAS Before RAS Refresh Cycle/Reset Cycle



Notes:
(1) mdsf<0> = 0b for Reset Cycle
mdsf<0> = 1b for CBR Refresh

: Don't Care

Table A-14: CMOS Window RAM Access Parameter List (Part 1 of 2)(1)

Name	*Min. (ns)*	*Max. (ns)*	*Comment*
Tcac		12	Access time from CAS/
Tcpa		20	Access time from CAS/ pre-charge
Taa		22	Access time from column address
Toea		12	Access time from output enable
Trac		50	Access time from RAS/
Tcbh	0		BE/ hold referenced to CAS/
Tbsc	7		BE/ setup referenced to CAS/
Tcsh	45		CAS/ hold time
Tchr	10		CAS/ hold time (CBR refresh)
Tcp	5		CAS/ pre-charge time (Ultra-fast page mode)
Tcas	6		CAS/ pulse width
Tcsr	5		CAS/ setup time (CBR refresh)
Tclz	0		CAS/ to output in low-Z
Tcrp	5		CAS/ to RAS/ pre-charge time
Tcah	0		Column address hold time
Tasc	7		Column address setup time
Tral	22		Column address to RAS/ lead time
Tdh	0		Data hold time
Tds	7		Data setup time
Tdzc	0		Data to CAS/ delay
Tohc	0		OE/ high hold time from CAS/ low
Tcfh	0		DSF hold time referenced to CAS/
Trfh	8		DSF hold time referenced to RAS/
Tfsc	7		DSF setup referenced to CAS/
Tfsr	0		DSF setup referenced to RAS/
Toed	7		Output enable to data input delay
Toff	3	7	Output buffer turn-off delay from CAS/ (RAS/ = high)
Toez	3	7	Output buffer turn-off delay from OE/
Tosc	20		OE/ high to CAS/ low setup time
Torh	8		OE/ hold referenced to RAS/
Tors	0		OE/ setup referenced to RAS/
Tdoh	3		RAM output hold time from CAS/
Trc	90		Random read or write cycle time
Trsh	15		RAS/ hold time
Trp	35		RAS/ pre-charge time
Trpc	10		RAS/ pre-charge to CAS/ hold time
Tras	60	10K	RAS/ pulse width
Trcd	20		RAS/ to CAS/ delay time
Toch	5		OE/ to CAS/ hold time to see valid output
Tcho	5		CAS/ high to OE/ hold time to hide the output
Tbez	3	7	Output buffer turn-off delay from CAS/ (BE/ high at falling edge of CAS/)

Table A-14: CMOS Window RAM Access Parameter List (Part 2 of 2)[1]

Name	Min. (ns)	Max. (ns)	Comment
Toep	5		OE/ pre-charge time
Tpcrw	35		Read-write cycle time
Trah	6		Row address hold time
Tasr	0		Row address setup time
Tupc	16		Ultra-fast page mode cycle time
Tmh	8		Write/bit mask data hold
Tms	0		Write/bit mask data setup
Toeo	0		Output buffer turn-on delay from OE/
Trad	12		RAS to column address delay type

(1) Timings are defined by V_{IL} = 0.8V and V_{IH} = 2.4V

A.4.2.4 Co-processor Interface Timing

Bus Request/Grant

Figure A-21: Bus Request/Grant Waveform

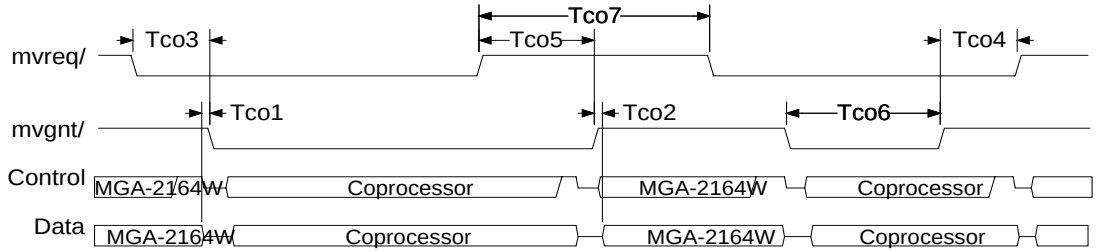


Table A-15: Bus Request/Grant Parameter List

Name	Min. (ns)	Max. (ns)	Comment
Tco1	0		Control & Data High-Z --> MACK/ low
Tco2	0		MACK/ high -> Control & Data Low-Z
Tco3		30*Tgclk	MRQ/ low -> MACK/ low
Tco4		30*Tgclk	MACK/ high -> MRQ/ high
Tco5	2.5*Tgclk	3.5*Tgclk	MRQ/ high -> MACK/ high
Tco6	2*Tgclk		MACK/ low
Tco7	2*Tgclk		MRQ/ high

A.4.2.5 Video Interface Timing

Figure A-22: Clock Waveform

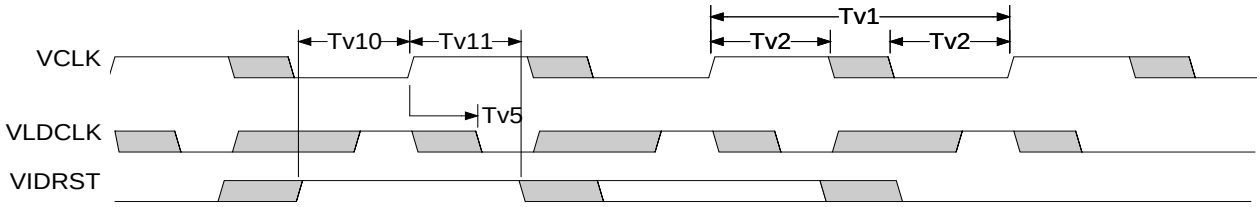


Figure A-23: Power Graphic Mode Waveform

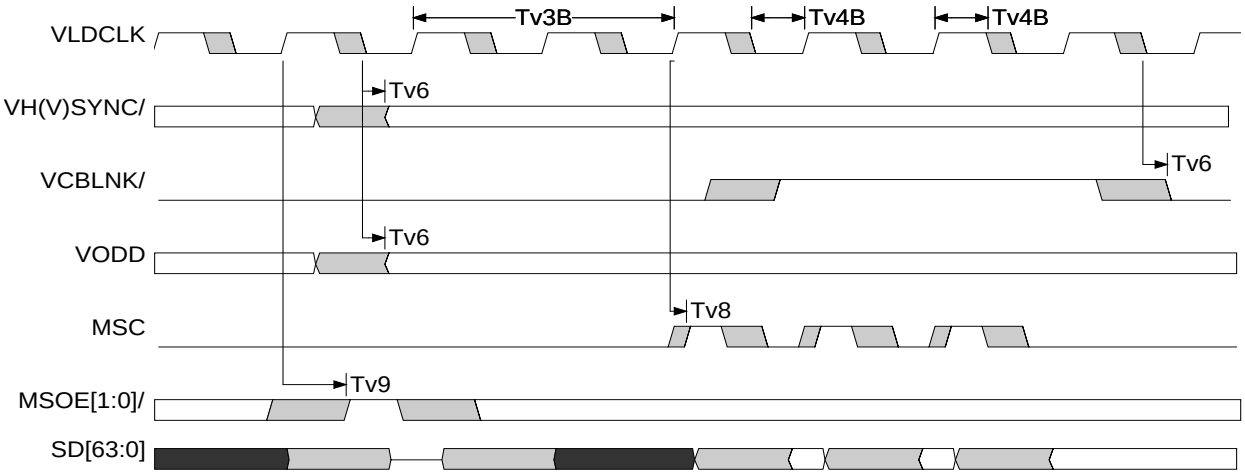


Figure A-24: VGA Mode Waveform

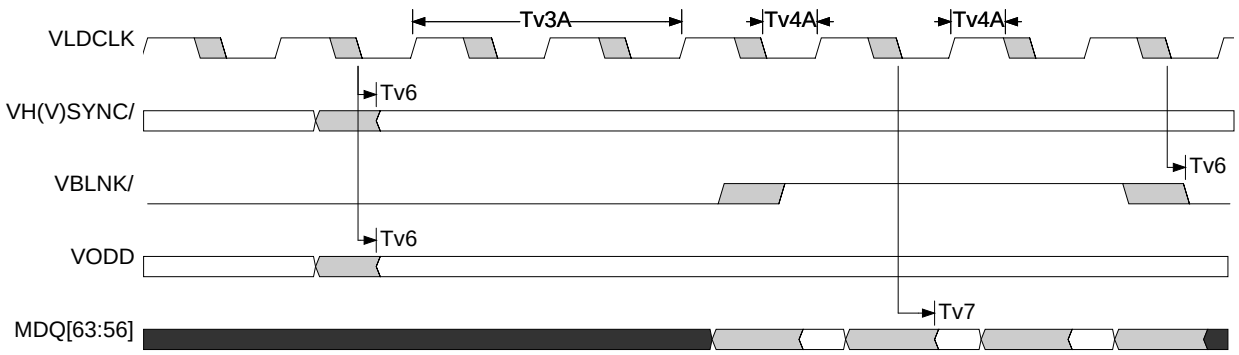
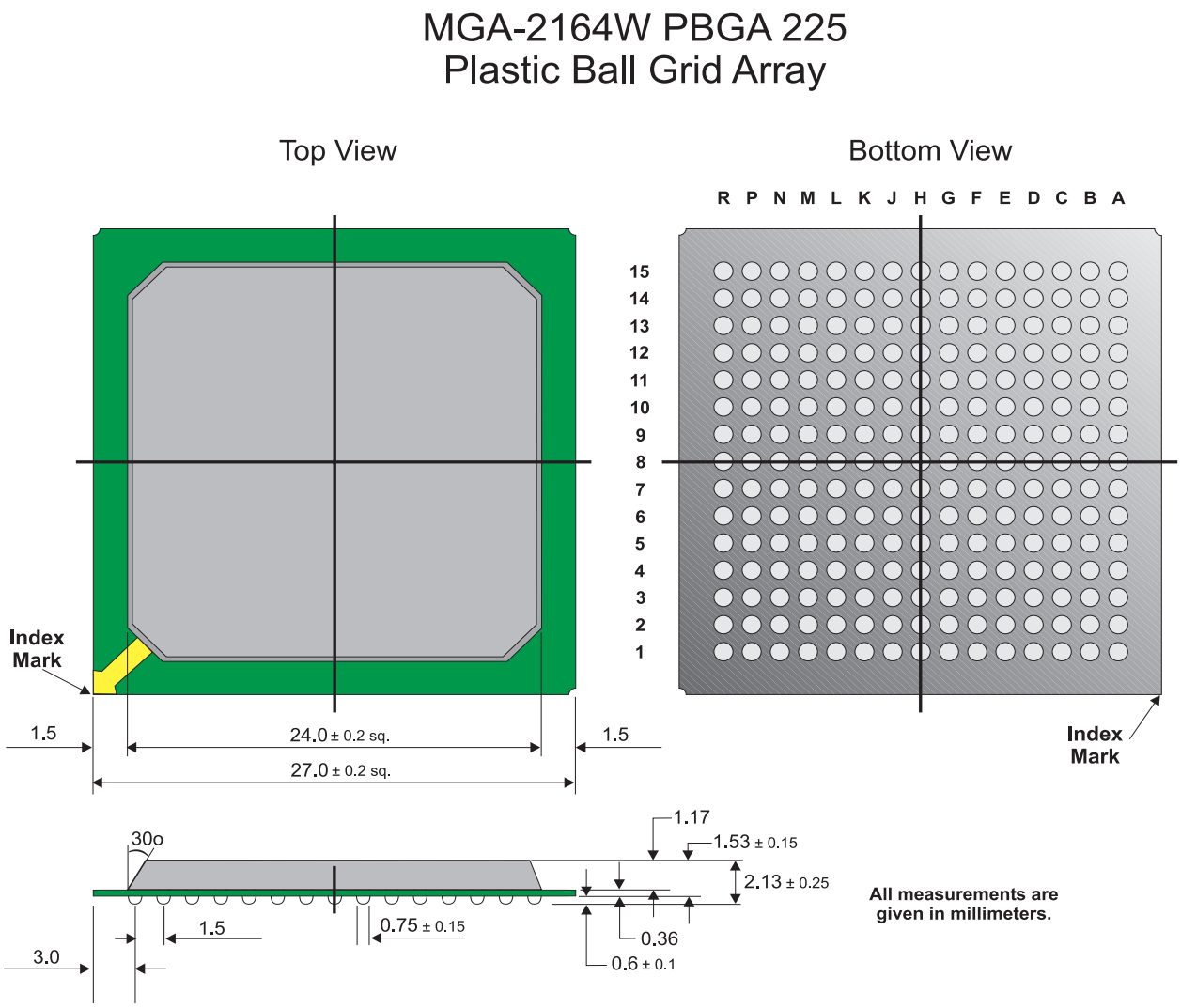


Table A-16: Video Interface Parameter List

Symbol	*Parameter*	*Min. (ns)*	*Max. (ns)*
Tv1A	VCLK period (VGA mode)	20	
Tv1B	VCLK period (Power Graphic mode)	14.81	
Tv2A	VCLK high or low time (VGA mode)	9	
Tv2B	VCLK high or low time (Power Graphic mode)	6.67	
Tv3A	VLDCLK period (VGA mode)	20	
Tv3B	VLDCLK period (Power Graphic mode)	14.81	
Tv4A	VLDCLK high or low time (VGA mode)	8	
Tv4B	VLDCLK high or low time (Power Graphic mode)	3	
Tv5	VCLK -> VLDCLK	1	15
Tv6	VLDCLK falling -> VHSYNC/, VVSYNC/, VCBLNK/, VODD	-3.00	2
Tv7	VLDCLK falling -> MDQ<56:63> (VGADAT)	-4	4
Tv8	VLDCLK rising -> MSC rising	-1.0	1.5
Tv9	VLDCLK -> VSOE<1:0>/	-2	8
Tv10	VIDRST set up -> VCLK	5	
Tv11	VIDRST hold -> VCLK	0.25	

A.5 Mechanical Specification

Figure A-25: MGA-2164W Mechanical Drawing



A.6 Test Feature

The MGA-2164W 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 HIZ/ pins to '0' to enter test mode and maintain that value during the entire test (for normal operations, the HIZ/ pin is tied to the pull-up). This will disable all output drivers except the PINTA/ pin. All pins (except PINTA/ and HIZ/) 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.6.1 Nand Tree Order

Table A-17: 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)	B1	PRST/	32	L4	PAD<14>	63	M9	MCAS<0>/
2	C2	PGNT/	33	M3	PAD<10>	64	L9	MRAS<0>/
3	D3	PCLK	34	N2	PAD<13>	65	R10	MRAS<1>/
4	C1	PREQ/	35	P1	PAD<12>	66	P10	MSC
5	E4	PAD<30>	36	N3	PAD<7>	67	N10	MA<8>
6	D1	PAD<26>	37	P2	PAD<8>	68	R11	MA<6>
7	E3	PAD<28>	38	M4	PAD<11>	69	M10	MA<7>
8	E2	PAD<24>	39	R2	PAD<3>	70	N11	MA<5>
9	E1	PAD<31>	40	P3	PAD<5>	71	R12	MA<3>
10	F4	PAD<29>	41	N4	PAD<1>	72	P12	MDQ<17>
11	F3	PAD<22>	42	R3	PAD<9>	73	M11	MA<4>
12	F2	PAD<27>	43	P4	PAD<6>	74	R13	MDQ<16>
13	G4	PAD<25>	44	R4	PAD<4>	75	N12	MA<2>
14	G2	PIDSEL	45	N5	PAD<2>	76	P13	MA<0>
15	G1	PCBE<3>/	46	R5	PAD<0>	77	R14	MDQ<19>
16	G5	PAD<23>	47	M6	PCBE<0>/	78	N13	MA<1>
17	H3	PAD<21>	48	N6	MBE<6>/	79	P14	MDQ<18>
18	H2	PAD<20>	49	P6	MBE<4>/	80	R15	MDSF<2>
19	H1	PAD<18>	50	R6	MBE<5>/	81	M12	MOE<1>/
20	H4	PAD<17>	51	M7	MBE<7>/	82	P15	MDSF<0>
21	H5	PAD<19>	52	N7	MBE<2>/	83	N14	MDSF<1>
22	J3	PAD<16>	53	P7	MBE<0>/	84	L11	MOE<2>/
23	J4	PFRAME/	54	R7	MBE<1>/	85	M13	MOE<0>/
24	K1	PCBE<2>/	55	L7	EXTCS/	86	N15	MOE<3>/
25	K3	PTRDY/	56	P8	MBE<3>/	87	M14	MDQ<20>
26	L1	PIRDY/	57	R8	MCAS<3>/	88	L12	MSOE<0>/
27	L2	PCBE<1>/	58	M8	EXTINT/	89	M15	MSOE<1>/
28	L3	PSTOP/	59	L8	VLDCLK/	90	L13	MDQ<21>
29	M1	PDEVSEL/	60	P9	MCAS<2>/	91	L14	MDQ<22>
30	K5	PPAR	61	R9	MCAS<1>/	92	L15	MDQ<23>
31	M2	PAD<15>	62	N9	VCLK/	93	K12	MDQ<7>

Table A-17: 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*
94	K13	MDQ<6>	117	E12	MVGNT/	140	A9	MDQ<26>
95	K14	MDQ<5>	118	C15	MVREQ/	141	E9	MDQ<13>
96	J12	MDQ<4>	119	D13	VCLKSL<1>	142	C8	MDQ<12>
97	J13	MDQ<3>	120	C14	VCLKSL<0>	143	B8	MDQ<28>
98	J14	MDQ<2>	121	B15	VIDRST	144	D8	MDQ<11>
99	J15	MDQ<0>	122	C13	VHSYNC/	145	E8	MDQ<10>
100	J11	MDQ<1>	123	B14	VCBLNK/	146	B7	MDQ<29>
101	H13	MDQ<63>	124	D12	GCLK	147	A7	MDQ<30>
102	H12	MDQ<53>	125	B13	VVSYNC/	148	D7	MDQ<8>
103	H11	MDQ<61>	126	A14	VODD	149	E7	MDQ<9>
104	G14	MDQ<62>	127	E11	DACWT/	150	A6	MDQ<31>
105	G15	MDQ<50>	128	C12	DACRD/	151	B6	MDQ<51>
106	G13	MDQ<59>	129	B12	ROMCS/	152	A5	MDQ<32>
107	G12	MDQ<60>	130	D11	EXTRST/	153	B5	MDQ<34>
108	G11	MDQ<48>	131	A12	MDQ<40>	154	D6	MDQ<33>
109	F14	MDQ<58>	132	C11	MDQ<41>	155	C5	MDQ<35>
110	F13	MDQ<56>	133	B11	MDQ<43>	156	A4	MDQ<37>
111	E15	MDQ<49>	134	A11	MDQ<54>	157	B4	MDQ<36>
112	E14	MDQ<57>	135	D10	MDQ<42>	158	D5	MDQ<38>
113	F12	MDQ<24>	136	C10	MDQ<52>	159	A3	MDQ<44>
114	E13	MDQ<46>	137	A10	MDQ<15>	160	B3	MDQ<39>
115	D15	MDQ<25>	138	D9	MDQ<14>	161	C3	MDQ<55>
116	D14	MDQ<45>	139	C9	MDQ<27>	162	B2	MDQ<47>

A.7 Ordering Information

■ To receive a MGA-2164W-PCI, order:	**IS-MGA-2164WP-C**
■ To receive a MGA-2164W-AGP, order:	**IS-MGA-2164WA-B**

Notes

Notes

Power Graphic Mode Register Fields (includes configuration space and memory space register fields)

Alphabetical List of Register Fields

VGA Mode Register Fields

Alphabetical List of Register Fields

<3:2> 3-71
dmapad
<31:0> 3-38
dotclkrt
<3> 3-164
dotmode
<0> 3-164
dr0
<31:0> 3-42
DR0_Z32 LSB 2-11
DR0_Z32 MSB 2-11
dr0_z32
<47:0> 3-39
dr10
<23:0> 3-49
dr11
<23:0> 3-50
dr12
<23:0> 3-51
dr14
<23:0> 3-52
dr15
<23:0> 3-53
dr2
<31:0> 3-43
DR2_Z32 LSB 2-11
DR2_Z32 MSB 2-11
dr2_z32
<47:0> 3-40
dr3
<31:0> 3-44
DR3_Z32 LSB 2-11
DR3_Z32 MSB 2-11
dr3_z32
<47:0> 3-41
dr4
<23:0> 3-45
dr6
<23:0> 3-46
dr7
<23:0> 3-47
dr8
<23:0> 3-48
dsts
<1:0> 3-145
dwgengsts
<16> 3-81
dword<6> 3-126
eepromwt
<20> 3-19

<2> 3-68
pickpen
<2> 3-81
plnwrmsk
<31:0> 3-75
plwren
<3:0> 3-165
powerpc
<31> 3-19
prefetchable
RO <3> 3-15
prefetchable
RO <3> 3-16
prefetchable
RO <3> 3-17
productid
RO <28:24> 3-19
prowscan
<4:0> 3-114
pwidth 212
pwidth
<1:0> 3-70
rammapen
<1> 3-160
rate_cap
<1:0> 3-5
rdmapsl
<1:0> 3-152
rdmode
<3> 3-153
recmastab
R/W <29> 3-9
rectargab
R/W <28> 3-9
refcol
<3:0> 3-150
Reserved
<23:10>
<7:3> 3-6
Reserved
<23:10>
<8:2> 3-5
Reserved
<31:24> 3-4
Reserved
<31:8> 3-3
Reserved
<63:48> 3-39
resparerr
RO <6> 3-8

<7> 3-139
vsyncend
<3:0> 3-123
vsyncoff
<5> 3-139
vsyncpen
<4> 3-81
vsyncpol
<7> 3-161
vsyncstr
<2> 3-113
vsyncstr
<6:5> 3-140
vsyncstr
<7:0> 3-122
vsyncstr
<7> 3-113
vsyncsts
<3> 3-81
vtotal
<0> 3-113
vtotal
<1:0> 3-140
vtotal
<5> 3-113
vtotal
<7:0> 3-112
waitcycle
RO <7> 3-8
wbmode
<6> 3-132
wrmask
<7:0> 3-157
wrmode
<1:0> 3-153
x_end
<15:0> 3-84
x_off
<3:0> 3-79
x_start
<15:0> 3-85
xdst
<15:0> 3-83
y_end
<31:16> 3-84
y_off
<6:4> 3-79
y_start
<31:16> 3-85
ydst

<22:0> 3-87
ydstorg
<23:0> 3-89
ylin
<15> 3-74
yval
<31:16> 3-88
zmode
<10:8> 3-56
zorg
<23:0> 3-91
zwidth
<3> 3-70