Matrox Graphics Inc.

Matrox MGA-G200 Specification

Document Number 10582-MS-0301
November 30, 1998

Trademark Acknowledgements

MGA,™ MGA-G200,™ 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

Appendix B: Changes

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

Chapter B: Changes

Not Applicable

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

Chapter A: Technical Information

Chapter B: Changes

Not Applicable

Chapter 1: MGA Overview

1.1 Introduction

The Matrox MGA-G200 which is compatible with the MGA-G100 product but with a significant improvement in performance:

- Supports full AGP features
- Includes high performance triangle setup engine
- Accelerates 3D texture mapped consumer applications such as PC games with the Enhanced Matrox Fast Texture Architecture
- Improves 2-D performance
- Provides superior Windows performance
- Is fully Microsoft DirectDraw, Direct 3D, and Open GL compliant
- Has fast VGA acceleration
- Accelerates digital video features
- Includes integrated frontend and backend scaler
- Includes digital video input port and video output port
- Includes hardware CODEC interface port
- Includes an integrated DAC
- Connects to SSTL or LVTTL SGRAM

The Matrox MGA-G200 has special features specifically designed to provide superior 3D performance in a 4 MByte frame buffer. The Matrox MGA-G200 is intended to provide a complete solution for home PC users who are interested in top performance Windows 95 and DOS 3D game and multimedia applications, but who are also interested in leveraging their home PC as a home office and education centre. It is also suitable for environments such as Windows NT, IBM OS/2 PM, Unix X-Windows, AutoCAD, and more.

The MGA-G200 series has an improved 3D acceleration core over the Matrox MGA-G100, key video capabilities of the MGA-VC064FB video engine and a significantly faster frame buffer interface fo applications requiring a high display bandwidth. It controls up to 16 megabytes of SGRAM.

The integrated DAC in the MGA-G200 eliminates the need for an external DAC. This substantially lowers the cost and space required for the graphics sub-system.

The MGA-G200A is optimized to exploit the AGP bus features and bandwidth. Alternatively, the PCI version or MGA-G200P also provides superior performance in a PCI-based system. In order to optimize performance, both DMA model and execute model are supported. The bus controller uses bus-mastering techniques to fetch command lists from PCI or AGP space, load textures into the texture cache or into the frame buffer, and to blit data between system memory and the frame buffer. The AGP transfers are performed in 2X mode, with sideband signalling and command pipelining to further parallelize and accelerate operations. The graphic engine has been designed to accept the longer latency periods that occur when accessing the system bus. This enables the application to store information, such as texture, in the system memory without any loss in performance.

A fully programmable setup engine increases 3D performance and off-loads the CPU for other tasks. Combined with a new bus controller, it can directly interpret triangle list information. Its instruction cache enables transparent transition between multiple micro-code programs. Multiple pipelined ALUs operate in parallel on floating point or integer data giving the MGA-G200 high performance and versatility.

A full-featured 3D rendering engine, the Enhanced Matrox Fast Texture Architecture, is the centerpiece of the MGA-G200. This 3D engine is an advanced renderer with full perspective correct texture mapping, lighting, Gouraud shading, specular lighting, fogging, stipple and true alpha blending, optional 16-bit or 32 bit Z-buffering, capable of bus mastering and keying on texture color or texel alpha key bit. Combined with the video engine, the 3D engine has the ability to use video as a source for texturing. The Matrox

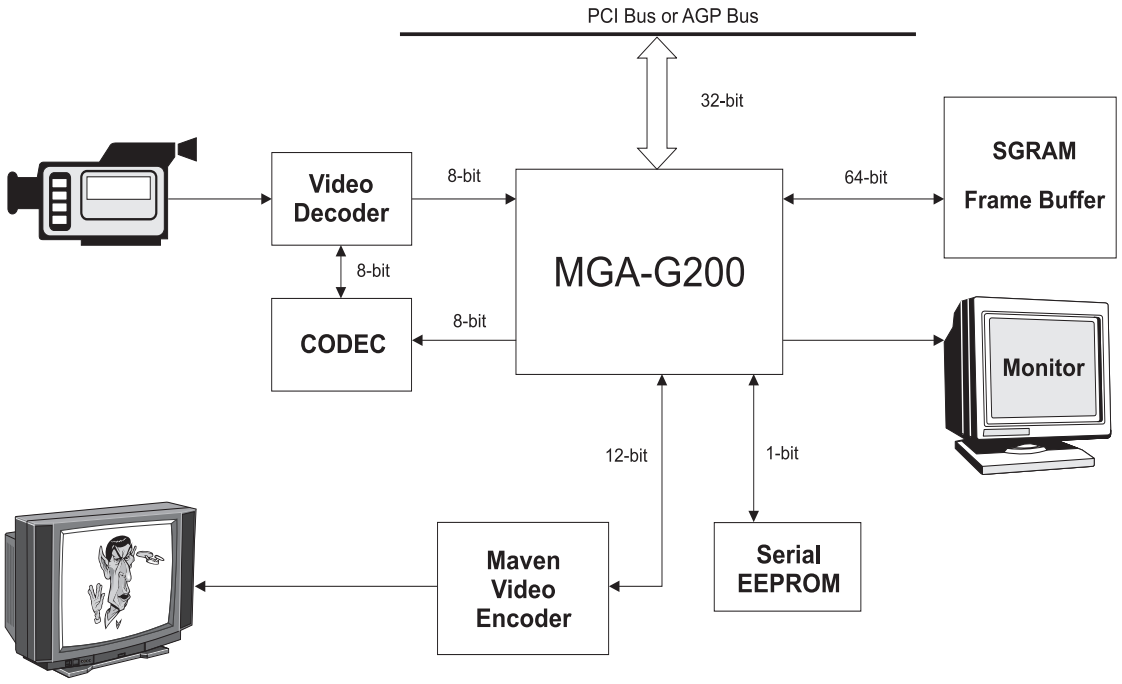
texture compression model saves on memory usage, allowing low cost and high performance even within a frame buffer as small as 2MBytes.

The MGA-G200 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-G200 can be upgraded with the Matrox Video Encoder (Maven) which provides high quality output to a TV or VCR.

This specification covers two chips: the MGA-G200P that connects to a PCI bus and MGA-G200A to an AGP bus. The specification applies the term MGA-G200 to both chips. For PCI specific information, the term MGA-G200P will apply, while the term MGA-G200A will apply to AGP specific information.

1.2 System Block Diagram

Figure 1-1: System Block Diagram



1.3 Application Areas

- A Windows accelerator with high performance levels. The MGA-G200 will complement the MGA family by delivering a strong price/performance point for users who need top performance at high resolution and color depths.
- Full acceleration of Windows multimedia and game applications. Specifically, 3D texture mapped games achieve a significant boost in performance and image quality with the MGA-G200 3D and triangle setup engine. In addition, all other types of games will be accelerated by a combination of the MGA-G200's DirectDraw, Direct 3D, and Direct Video engine.
- Digital video playback is accelerated to full screen, full motion, with high-quality scaling.The architecture supports all of today's popular CODECs.
- Full acceleration of all MS-DOS applications via MGA-G200's ultra-fast 32-bit VGA core.
- Video capture
- DVD and MPEG2 playback
- Video editing
- Video out to a TV with MAVEN

1.4 Target Markets

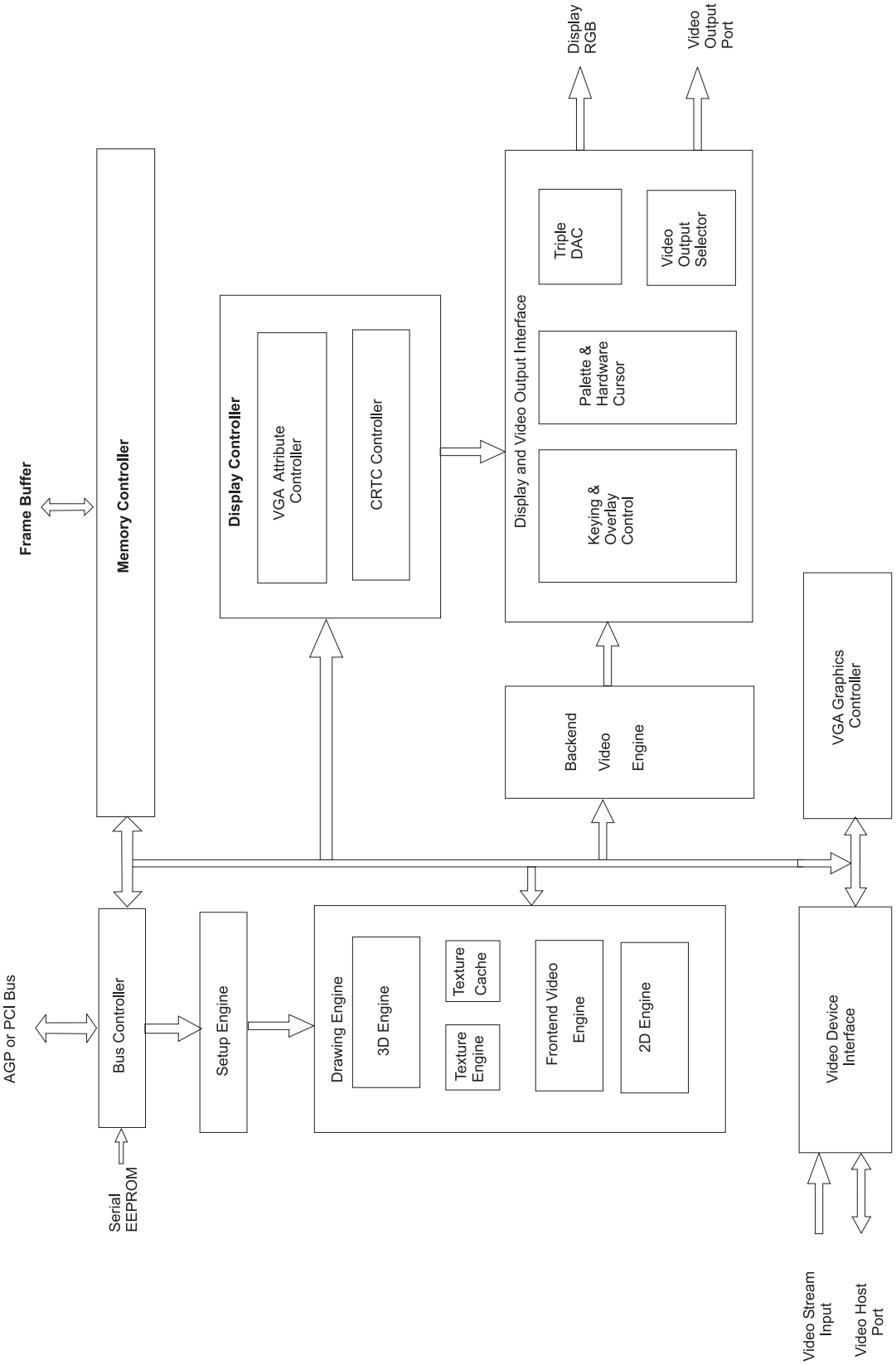
- Home, SOHO, and multimedia PC markets
- Mainstream business markets
- Computer gaming
- Workstation market
- Professional multimedia PC markets
- Desktop publishing
- CAD

1.5 Block Diagram

The MGA-G200 is composed of 9 sections:

- Bus Controller
- Setup Engine
- Drawing Engine
- Video Device Interface
- VGA Graphics Controller
- Backend Video Engine
- Memory Controller
- Display Controller
- Display and Video Output Interface

Figure 1-2: MGA-G200 Block Diagram



1.6 Features

1.6.1 PCI / AGP Bus Controller

- PCI 2.1 compliant
- AGP 1.0 compliant
- Supports AGP 2X mode
- Supports AGP sideband signalling
- PCI and AGP bus mastering support for texture fetching, command list execution and system memory to frame buffer and frame buffer to system memory transfers
- Supports AGP command pipelining
- Big and Little Endian support

1.6.2 Triangle Setup Engine

- Fully programmable setup engine with floating point and integer operations
- Instruction cache
- Off-loads the CPU
- Pipelined for increased performance
- Parallel execution of multiple instructions
- Fast access to the 3D rendering engine

1.6.3 3D and Texture Mapping Engine

- 16-bit or 32-bit Z-buffer (optionally enabled or disabled)
- 3D polygons with Gouraud shading
- Double and triple buffering
- Sub-pixel positioning
- Hardware dithering including dithering of LUT textures
- Perspective correct texture mapping
- Storage of source textures in off-screen frame buffer and system memory
- Texture cache for increased performance
- Selectable high quality texture filtering modes, including on-the-fly minify-magnify filter selection
- True color lighting of textures
- Specular lighting
- Depth cuing and fogging
- Stipple and true Alpha blending
- Transparency
- Keying on textures is supported
- Source textures may be in the following formats:
    - Color Look Up Table (compressed) 4 bpp (bit/pixel) or 8 bpp
    - True Color: 5:6:5, 1:5:5:5, 4:4:4:4, 8:8:8:8
    - Video: YCbCr 4:2:2 using hardware color space conversion
- Direct 3D support and acceleration
- Open GL support and acceleration

1.6.4 2D Engine

- Line draw engine with patterning
- 2D polygons with patterning capabilities
- BITBLT engine
- Stretch BLT
- System memory to frame buffer BLT
- Frame buffer to system memory BLT
- Color expansion
- Clipping
- Transparency and color keying
- Dithering
- Direct Draw support

1.6.5 VGA Engine

- Fully VGA compatible
- Accelerated performance

1.6.6 Video Engine

- Video scaling is supported in both frontend and backend video engines
- Independent X and Y scaling with high quality filtering
- Support YCbCr 4:2:2 and YCbCr 4:2:0 formats
- Support for true graphics overlay in a rectangular window, and optionally with color keying
- Synchronized video/graphics updates (no tearing) are supported
- Supports any number of video windows/sprites simultaneously
- Sync reset input for video genlock and overlay
- Hardware color space conversion
- ITU-R 656 compatible video input port
- Parallel video device host port with DMA capability
- Video pass-through mode to video output port
- Proprietary 12-bit video output port
- Support and acceleration of DVD and MPEG2 playback
- Direct Video support and acceleration

1.6.7 Display Engine

- Integrated DAC
- 250 MHz operation
- Supports shared memory and graphic overlay modes
- 3 x 256 x 8 look-up table
- Hardware color cursor
- VGA compatible
- Hardware pan and zoom
- DDC level 2B compliant

1.6.8 Memory Controller

- Supports from 2 to 16 MBytes of memory
  - up to 4 banks of 2bank x 128Kword x 32bit SGRAM
  - up to 4 banks of 2bank x 256Kword x 32bit SGRAM
  - up to 4 banks of 4bank x 128Kword x 32bit SGRAM
  - up to 2 banks of 2bank x 512Kword x 16bit SDRAM
- Supports block write and write per bit for added performance
- Supports operating frequencies up to 143 MHz
- Configurable SSTL or LVTLL support

1.6.9 Other Features

- PCI bus power management compliant
- PC98 compliant
- Serial EEPROM interface
- VESA 2.0-compliant

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, 3.2.1, and 3.3.1.	
In a table, X = “don’t care” (the value doesn’t matter)	1X = Register Set C
Emphasized text and table column titles are set in bold italics.	This bit ***must be set***.
In the **DWGCTL** illustrations (in Chapter 4), the ‘+’ and ‘#’ symbols have a special meaning. This is explained in ‘Programmer’s Specification’ on page 4-1.	**trans** \# \# \# \#

1.8 Locating Information

The MGA-G200 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 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-G200 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-G200 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**	Capabilities Pointer
38h-3Bh	Reserved (1)	—
3Ch-3Fh	**INTCTRL**	Interrupt Control
40h-43h	**OPTION**	Option register number 1
44h-47h	**MGA_INDEX**(2)	MGA Indirect Access Index
48h-4Bh	**MGA_DATA**(2)	MGA Indirect Access Data
4Ch-4Fh	**SUBSYSID**	Location for writing the Subsystem ID. Reading will give 0's.
50h-53h	**OPTION2**	Option register number 2
54h-DBh	Reserved (1)	—
DCh-DFh	**PM_IDENT**	Power Management Identifier
E0h-E3h	**PM_CSR**	Power Management Control / Status
E4h-EFh	Reserved (1)	—
F0h-F3h	**AGP_IDENT** (3)	AGP Capability Identifier
F4h-F7h	**AGP_STS** (3)	AGP Status
F8h-FBh	**AGP_CMD** (3)	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) Not supported when powerpc is '1'. Reading to these locations will return unkown values; writing to these locations may modify any register described in the MGABASE1 range.

(3) These locations exist only for the MGA-G200-AGP. For the MGA-G200-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) *(Note 2 applies **only** if MGAMODE = '1')*
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)(5)

(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.

(5) This memory space is Write Only. Reads will return ***unknown*** values.

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	7KByte Pseudo-DMA window (1)(4)
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 (3)(5)
2000h-207Fh	R/W	WIMEMDATA	WARP instruction memory (2)(3)
2080h-2BFFh	—	—	Reserved (6)
2C00h-2DFFh	W	DWGREG1	Second set of drawing registers(2)(3)(4)
2E00h-3BFFh	—	—	Reserved (6)
3C00h-3C0Fh	R/W	DAC	RAMDAC registers(3)
3C10h-3CFFh	—	—	Reserved (6)
3D00h-3DFFh	R/W	BESREG	Backend Scaler register (2)(3)
3E00h-3EFFh	R/W	VINCODEC	Video-in and codec interface (2)(3)
3F00h-3FFFh	—	—	Reserved (6)

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

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

(3) See the register map in Table 2-4 for a more detailed view of this memory space.

(4) Reads of these locations return ***unknown*** values (***except*** for range 2C40 to 2C4F and 2CD0 to 2CD7).

(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.

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 13)

Register Mnemonic Name	*Access*	*Memory Address*(1)	*I/O Address*(2)	*Index*	*Description/Comments*	*Page*
DWGCTL	WO	1C00h	—	00h	Drawing Control	3-99
MACCESS	WO	1C04h	—	01h	Memory Access	3-119
MCTLWTST	WO	1C08h	—	02h	Memory Control Wait State	3-121
ZORG	WO	1C0Ch	—	03h	Z-Depth Origin	3-223
PAT0	WO	1C10h	—	04h	Pattern	3-128
PAT1	WO	1C14h	—	05h	Pattern	"
—	WO	1C18h	—	—	Reserved	—
PLNWT	WO	1C1Ch	—	07h	Plane Write Mask	3-130
BCOL	WO	1C20h	—	08h	Background Color / Blit Color Mask	3-43
FCOL	WO	1C24h	—	09h	Foreground Color / Blit Color Key	3-107
—	—	1C28h	—	—	Reserved	—
—	WO	1C2Ch	—	0Bh	Reserved (SRCBLT)	—
SRC0	WO	1C30h	—	0Ch	Source	3-153
SRC1	WO	1C34h	—	0Dh	Source	"
SRC2	WO	1C38h	—	0Eh	Source	"
SRC3	WO	1C3Ch	—	0Fh	Source	"
XYSTRT (3)	WO	1C40h	—	10h	XY Start Address	3-217
XYEND (3)	WO	1C44h	—	11h	XY End Address	3-216
—	1C48h-1C4Fh		—	—	Reserved	—
SHIFT (3)	WO	1C50h	—	14h	Funnel Shifter Control	3-142
DMAPAD (3)	WO	1C54h	—	15h	DMA Padding	3-81
SGN (3)	WO	1C58h	—	16h	Sign	3-139
LEN (3)	WO	1C5Ch	—	17h	Length	3-118
AR0 (3)	WO	1C60h	—	18h	Multi-Purpose Address 0	3-36
AR1 (3)	WO	1C64h	—	19h	Multi-Purpose Address 1	3-37
AR2 (3)	WO	1C68h	—	1Ah	Multi-Purpose Address 2	3-38
AR3 (3)	WO	1C6Ch	—	1Bh	Multi-Purpose Address 3	3-39
AR4 (3)	WO	1C70h	—	1Ch	Multi-Purpose Address 4	3-40
AR5 (3)	WO	1C74h	—	1Dh	Multi-Purpose Address 5	3-41
AR6 (3)	WO	1C78h	—	1Eh	Multi-Purpose Address 6	3-42
—	—	1C7Ch	—	—	Reserved	—
CXBNDRY (3)	WO	1C80h	—	20h	Clipper X Boundary	3-74

Table 2-4: Register Map (Part 2 of 13)

Register Mnemonic Name	*Access*	*Memory Address(1)*	*I/O Address(2)*	*Index*	*Description/Comments*	*Page*
FXBNDRY (3)	WO	1C84h	—	21h	X Address (Boundary)	3-113
YDSTLEN (3)	WO	1C88h	—	22h	Y Destination and Length	3-220
PITCH (3)	WO	1C8Ch	—	23h	Memory Pitch	3-129
YDST (3)	WO	1C90h	—	24h	Y Address	3-219
YDSTORG (3)	WO	1C94h	—	25h	Memory Origin	3-221
YTOP (3)	WO	1C98h	—	26h	Clipper Y Top Boundary	3-222
YBOT (3)	WO	1C9Ch	—	27h	Clipper Y Bottom Boundary	3-218
CXLEFT (3)	WO	1CA0h	—	28h	Clipper X Minimum Boundary	3-75
CXRIGHT (3)	WO	1CA4h	—	29h	Clipper X Maximum Boundary	3-76
FXLEFT (3)	WO	1CA8h	—	2Ah	X Address (Left)	3-114
FXRIGHT (3)	WO	1CACh	—	2Bh	X Address (Right)	3-115
XDST (3)	WO	1CB0h	—	2Ch	X Destination Address	3-215
—	1CB4h-1CBFh		—	—	Reserved	—
DR0	WO	1CC0h	—	30h	Data ALU 0	3-85
FOGSTART	WO	1CC4h	—	31h	Fog Start	3-110
DR2	WO	1CC8h	—	32h	Data ALU 2	3-86
DR3	WO	1CCCh	—	33h	Data ALU 3	3-87
DR4	WO	1CD0h	—	34h	Data ALU 4	3-88
FOGXINC	WO	1CD4h	—	35h	Fog X Inc	3-111
DR6	WO	1CD8h	—	36h	Data ALU 6	3-89
DR7	WO	1CDCh	—	37h	Data ALU 7	3-90
DR8	WO	1CE0h	—	38h	Data ALU 8	3-91
FOGYINC	WO	1CE4h	—	39h	Fog Y Inc	3-112
DR10	WO	1CE8h	—	3Ah	Data ALU 10	3-92
DR11	WO	1CECh	—	3Bh	Data ALU 11	3-93
DR12	WO	1CF0h	—	3Ch	Data ALU 12	3-94
FOGCOL	WO	1CF4h	—	3Dh	Fog Color	3-109
DR14	WO	1CF8h	—	3Eh	Data ALU 14	3-95
DR15	WO	1CFCh	—	3Fh	Data ALU 15	3-96
—	1D00h-1DBFh		—	(6)	Same mapping as 1C00h-1CBFh (4)	—
WIADDR	WO	1DC0h	—	70h	WARP Instruction Address	3-203
WFLAG	WO	1DC4h	—	71h	WARP Flags	3-200
WGETMSB	WO	1DC8h	—	72h	WARP GetMSB Value	3-202
WVRTXSZ	WO	1DCCh	—	73h	WARP Vertex Size	3-210
—	WO	1DD0h	—	74h	Reserved (WDBR)	—
—	1DD4h - 1E0Fh		—	—	Reserved	—

Table 2-4: Register Map (Part 3 of 13)

Register Mnemonic Name	*Access*	*Memory Address(1)*	*I/O Address(2)*	*Index*	*Description/Comments*	*Page*
FIFOSTATUS	RO	1E10h	—	—	Bus FIFO Status	3-108
STATUS	R/W	1E14h	—	—	Status	3-155
ICLEAR	WO	1E18h	—	—	Interrupt Clear	3-116
IEN	R/W	1E1Ch	—	—	Interrupt Enable	3-117
VCOUNT	RO	1E20h	—	—	Vertical Count	3-188
—	1E24h - 1E2Fh		—	—	Reserved	—
DMAMAP30	R/W	1E30h	—	—	DMA Map 3h to 0h	3-77
DMAMAP74	R/W	1E34h	—	—	DMA Map 7h to 4h	3-78
DMAMAPB8	R/W	1E38h	—	—	DMA Map Bh to 8h	3-79
DMAMAPFC	R/W	1E3Ch	—	—	DMA Map Fh to Ch	3-80
RST	R/W	1E40h	—	—	Reset	3-134
MEMRDBK	R/W	1E44h	—	—	Memory Read Back	3-124
TEST0	R/W	1E48h	—	—	Test0	3-157
AGP_PLL	R/W	1E4Ch	—	—	AGP 2X PLL Control/Status	3-30
PRIMPTR	R/W	1E50h	—	—	Primary List Status Fetch Pointer	3-133
OPMODE	R/W	1E54h	—	—	Operating Mode	3-126
PRIMADDRESS	R/W	1E58h	—	—	Primary DMA Current Address	3-131
PRIMEND	R/W	1E5Ch	—	—	Primary DMA End Address	3-132
WIADDRNB	R/W	1E60h	—	—	WARP Instruct. Add. (Non-Blocking)	3-205
WFLAGNB	R/W	1E64h	—	—	WARP Flags (Non-Blocking)	3-201
WIMEMADDR	WO	1E68h	—	—	WARP Instruction Memory Address	3-206
WCODEADDR	RO	1E6Ch	—	—	WARP Microcode Address	3-199
WMISC	R/W	1E70h	—	—	WARP Miscellaneous	3-208
—	1E7Ch - 1E7Fh		—	—	Reserved	—
DWG_INDIR_WT<0>	WO	1E80h	—	—	Drawing Register Indirect Write 0	3-98
…	WO	1E84h-1EB8h			…	
DWG_INDIR_WT<15>	WO	1EBCh	—	—	Drawing Register Indirect Write 15	3-98
—	1EC0h - 1FBFh		—	—	Reserved	—
ATTR (Index)	R/W	1FC0h	3C0h	—	Attribute Controller	3-226
ATTR (Data)	WO	1FC0h	3C0h	—	Attribute Controller	"
ATTR (Data)	RO	1FC1h	3C1h	—	Attribute Controller	"
—	—	1FC1h	3C1h	—	Reserved	—
ATTR0	R/W	—	—	00h	Palette entry 0	3-226
ATTR1	R/W	—	—	01h	Palette entry 1	"
ATTR2	R/W	—	—	02h	Palette entry 2	"
ATTR3	R/W	—	—	03h	Palette entry 3	"

Table 2-4: Register Map (Part 4 of 13)

Register Mnemonic Name	*Access*	*Memory Address(1)*	*I/O Address(2)*	*Index*	*Description/Comments*	*Page*
ATTR4	R/W	—	—	04h	Palette entry 4	"
ATTR5	R/W	—	—	05h	Palette entry 5	"
ATTR6	R/W	—	—	06h	Palette entry 6	"
ATTR7	R/W	—	—	07h	Palette entry 7	"
ATTR8	R/W	—	—	08h	Palette entry 8	"
ATTR9	R/W	—	—	09h	Palette entry 9	"
ATTRA	R/W	—	—	0Ah	Palette entry A	"
ATTRB	R/W	—	—	0Bh	Palette entry B	"
ATTRC	R/W	—	—	0Ch	Palette entry C	"
ATTRD	R/W	—	—	0Dh	Palette entry D	"
ATTRE	R/W	—	—	0Eh	Palette entry E	"
ATTRF	R/W	—	—	0Fh	Palette entry F	"
ATTR10	R/W	—	—	10h	Attribute Mode Control	3-229
ATTR11	R/W	—	—	11h	Overscan Color	3-231
ATTR12	R/W	—	—	12h	Color Plane Enable	3-232
ATTR13	R/W	—	—	13h	Horizontal Pel Panning	3-233
ATTR14	R/W	—	—	14h	Color Select	3-234
—	—	—	—	15h - 1Fh: Reserved		—
INSTS0	RO	1FC2h	3C2h	—	Input Status 0	3-292
MISC	WO	1FC2h	3C2h	—	Miscellaneous Output	3-294
—	R/W	1FC3h	3C3h(5)	—	Reserved, not decoded for I/O	—
SEQ (Index)	R/W	1FC4h	3C4h	—	Sequencer	3-296
SEQ (Data)	R/W	1FC5h	3C5h	—	Sequencer	—
SEQ0	R/W	—	—	00h	SEQ0	3-297
SEQ1	R/W	—	—	01h	Clocking Mode	3-298
SEQ2	R/W	—	—	02h	Map Mask	3-299
SEQ3	R/W	—	—	03h	Character Map Select	3-300
SEQ4	R/W	—	—	04h	Memory Mode	3-301
—	R/W	—	—	05h - 07h: Reserved		—
—	—	1FC6h	—	—	Reserved	—
DACSTAT	RO	1FC7h	3C7h	—	DAC Status(requires a byte access)	3-279
—	WO	1FC7h	—	—	Reserved	—
—	1FC8h-1FC9h		—	—	Reserved	—
FEAT	RO	1FCAh	3CAh	—	Feature Control	3-280
—	WO	1FCAh	3CAh	—	Reserved	—
—	—	1FCBh	3CBh(5)	—	Reserved, not decoded for I/O	—

Table 2-4: Register Map (Part 5 of 13)

Register Mnemonic Name	*Access*	*Memory Address(1)*	*I/O Address(2)*	*Index*	*Description/Comments*	*Page*
MISC	RO	1FCCh	3CCh	—	Miscellaneous Output	3-294
—	WO	1FCCh	3CCh	—	Reserved	—
—	—	1FCDh	3CDh(5)	—	Reserved, not decoded for I/O	—
GCTL (Index)	R/W	1FCEh	3CEh	—	Graphics Controller	3-281
GCTL (Data)	R/W	1FCFh	3CFh	—	Graphics Controller	"
GCTL0	R/W	—	—	00h	Set/Reset	3-282
GCTL1	R/W	—	—	01h	Enable Set/Reset	3-283
GCTL2	R/W	—	—	02h	Color Compare	3-284
GCTL3	R/W	—	—	03h	Data Rotate	3-285
GCTL4	R/W	—	—	04h	Read Map Select	3-286
GCTL5	R/W	—	—	05h	Graphics Mode	3-287
GCTL6	R/W	—	—	06h	Miscellaneous	3-289
GCTL7	R/W	—	—	07h	Color Don’t Care	3-290
GCTL8	R/W	—	—	08h	Bit Mask	3-291
—	—	—	—	09h - 0Fh: Reserved		—
—	1FD0h-1FD3h		—	—	Reserved	—
CRTC (Index)	R/W	1FD4h	3D4h	—	CRTC Registers (or 3B4h(6))	3-236
CRTC (Data)	R/W	1FD5h	3D5h	—	CRTC Registers (or 3B5h(6))	"
CRTC0	R/W	—	—	00h	Horizontal Total	3-238
CRTC1	R/W	—	—	01h	Horizontal Display Enable End	3-239
CRTC2	R/W	—	—	02h	Start Horizontal Blanking	3-240
CRTC3	R/W	—	—	03h	End Horizontal Blanking	3-241
CRTC4	R/W	—	—	04h	Start Horizontal Retrace Pulse	3-242
CRTC5	R/W	—	—	05h	End Horizontal Retrace	3-243
CRTC6	R/W	—	—	06h	Vertical Total	3-244
CRTC7	R/W	—	—	07h	Overflow	3-245
CRTC8	R/W	—	—	08h	Preset Row Scan	3-246
CRTC9	R/W	—	—	09h	Maximum Scan Line	3-247
CRTCA	R/W	—	—	0Ah	Cursor Start	3-248
CRTCB	R/W	—	—	0Bh	Cursor End	3-249
CRTCC	R/W	—	—	0Ch	Start Address High	3-250
CRTCD	R/W	—	—	0Dh	Start Address Low	3-251
CRTCE	R/W	—	—	0Eh	Cursor Location High	3-252
CRTCF	R/W	—	—	0Fh	Cursor Location Low	3-253
CRTC10	R/W	—	—	10h	Vertical Retrace Start	3-254
CRTC11	R/W	—	—	11h	Vertical Retrace End	3-255

Table 2-4: Register Map (Part 6 of 13)

Register Mnemonic Name	*Access*	*Memory Address(1)*	*I/O Address(2)*	*Index*	*Description/Comments*	*Page*
CRTC12	R/W	—	—	12h	Vertical Display Enable End	3-256
CRTC13	R/W	—	—	13h	Offset	3-257
CRTC14	R/W	—	—	14h	Underline Location	3-258
CRTC15	R/W	—	—	15h	Start Vertical Blank	3-259
CRTC16	R/W	—	—	16h	End Vertical Blank	3-260
CRTC17	R/W	—	—	17h	CRTC Mode Control	3-261
CRTC18	R/W	—	—	18h	Line Compare	3-265
-	-	—	—	19h - 21h: Reserved		—
CRTC22	R/W	—	—	22h	CPU Read Latch	3-266
-	-	—	—	23h	Reserved	—
CRTC24	R/W	—	—	24h	Attributes Address/Data Select	3-267
-	-	—	—	25h	Reserved	—
CRTC26	R/W	—	—	26h	Attributes Address	3-268
—	—	—	—	27h - 3Fh: Reserved		—
—	—	1FD6h	3D6h(5)	—	Reserved, not decoded for I/O (or 3B6h(6))	—
—	—	1FD7h	3D7h(5)	—	Reserved, not decoded for I/O (or 3B7h(6))	—
—	1FD8h-1FD9h		—	—	Reserved	—
INSTS1	RO	1FDAh	3DAh	—	Input Status 1 (or 3BAh(6))	3-293
FEAT	WO	1FDAh	3DAh	—	Feature Control (or 3BAh(6))	3-280
—	—	1FDBh	3DBh(5)	—	Reserved, not decoded for I/O (or 3BBh(6))	—
—	1FDCh-1FDDh		—	—	Reserved	—
CRTCEXT (Index)	R/W	1FDEh	3DEh	—	CRTC Extension	3-269
CRTCEXT (Data)	R/W	1FDFh	3DFh	—	CRTC Extension	"
CRTCEXT0	R/W	—	—	00h	Address Generator Extensions	3-270
CRTCEXT1	R/W	—	—	01h	Horizontal Counter Extensions	3-271
CRTCEXT2	R/W	—	—	02h	Vertical Counter Extensions	3-272
CRTCEXT3	R/W	—	—	03h	Miscellaneous	3-273
CRTCEXT4	R/W	—	—	04h	Memory Page	3-275
CRTCEXT5	R/W	—	—	05h	Horizontal Video Half Count	3-276
CRTCEXT6	R/W	—	—	06h	Priority Request Control	3-277
CRTCEXT7	R/W	—	—	07h	Requester Control	3-278
—	1FE0h - 1FFEh		—	—	Reserved	—
CACHEFLUSH	R/W	1FFFh	—	—	Cache Flush	3-235
WIMEMDATA	R/W	2000h-207Fh		—	WARP Instruction Memory Data	3-207

Table 2-4: Register Map (Part 7 of 13)

Register Mnemonic Name	Access	Memory Address(1)	I/O Address(2)	Index	Description/Comments	Page
—	2080h-2BFFh		—	—	Reserved	—
TMR0	WO	2C00h	—	80h	Texture Mapping ALU 0	3-177
TMR1	WO	2C04h	—	81h	Texture Mapping ALU 1	3-178
TMR2	WO	2C08h	—	82h	Texture Mapping ALU 2	3-179
TMR3	WO	2C0Ch	—	83h	Texture Mapping ALU 3	3-180
TMR4	WO	2C10h	—	84h	Texture Mapping ALU 4	3-181
TMR5	WO	2C14h	—	85h	Texture Mapping ALU 5	3-182
TMR6	WO	2C18h	—	86h	Texture Mapping ALU 6	3-183
TMR7	WO	2C1Ch	—	87h	Texture Mapping ALU 7	3-184
TMR8	WO	2C20h	—	88h	Texture Mapping ALU 8	3-185
TEXORG	WO	2C24h	—	89h	Texture Origin	3-169
TEXWIDTH	WO	2C28h	—	8Ah	Texture Width	3-176
TEXHEIGHT	WO	2C2Ch	—	8Bh	Texture Height	3-168
TEXCTL	WO	2C30h	—	8Ch	Texture Map Control	3-161
TEXTRANS	WO	2C34h	—	8Dh	Texture Transparency	3-174
TEXTRANSHIGH	WO	2C38h	—	8Eh	Texture Transparency	3-175
TEXCTL2	WO	2C3Ch	—	8Fh	Texture Map Control 2	3-165
SECADDRESS	R/W	2C40h	—	90h	Secondary DMA Current Address(7)	3-135
SECEND	R/W	2C44h	—	91h	Secondary DMA End Address (7)	3-136
SOFTRAP	R/W	2C48h	—	92h	Soft Trap Handle (7)	3-143
DWGSYNC	R/W	2C4Ch	—	93h	Drawing Synchronisation	3-106
DR0_Z32 LSB	WO	2C50h	—	94h	Extended Data ALU 0	3-82
DR0_Z32 MSB	WO	2C54h	—	95h	Extended Data ALU 0	"
TEXFILTER	WO	2C58h	—	96h	Texture Filtering	3-166
TEXBORDERCOL	WO	2C5Ch	—	97h	Texture Border Color	3-160
DR2_Z32 LSB	WO	2C60h	—	98h	Extended Data ALU 2	3-83
DR2_Z32 MSB	WO	2C64h	—	99h	Extended Data ALU 2	"
DR3_Z32 LSB	WO	2C68h	—	9Ah	Extended Data ALU 3	3-84
DR3_Z32 MSB	WO	2C6Ch	—	9Bh	Extended Data ALU 3	"
ALPHASTART	WO	2C70h	—	9Ch	Alpha Start	3-33
ALPHAXINC	WO	2C74h	—	9Dh	Alpha X Inc	3-34
ALPHAYINC	WO	2C78h	—	9Eh	Alpha Y Inc	3-35
ALPHACTRL	WO	2C7Ch	—	9Fh	Alpha CTRL	3-31
SPECRSTART	WO	2C80h	—	A0h	Specular Lighting Red Start	3-150
SPECRXINC	WO	2C84h	—	A1h	Specular Lighting Red X Inc	3-151
SPECRYINC	WO	2C88h	—	A2h	Specular Lighting Red Y Inc	3-152

Table 2-4: Register Map (Part 8 of 13)

Register Mnemonic Name	*Access*	*Memory Address(1)*	*I/O Address(2)*	*Index*	*Description/Comments*	*Page*
SPECGSTART	WO	2C8Ch	—	A3h	Specular Lighting Green Start	3-147
SPECGXINC	WO	2C90h	—	A4h	Specular Lighting Green X Inc	3-148
SPECGYINC	WO	2C94h	—	A5h	Specular Lighting Green Y Inc	3-149
SPECBSTART	WO	2C98h	—	A6h	Specular Lighting Blue Start	3-144
SPECBXINC	WO	2C9Ch	—	A7h	Specular Lighting Blue X Inc	3-145
SPECBYINC	WO	2CA0h	—	A8h	Specular Lighting Blue Y Inc	3-146
TEXORG1	WO	2CA4h	—	A9h	Texture Origin 1	3-170
TEXORG2	WO	2CA8h	—	AAh	Texture Origin 2	3-171
TEXORG3	WO	2CACh	—	ABh	Texture Origin 3	3-172
TEXORG4	WO	2CB0h	—	ACh	Texture Origin 4	3-173
SRCORG	WO	2CB4h	—	ADh	Source Origin	3-154
DSTORG	WO	2CB8h	—	AEh	Destination Origin	3-97
—	2CBCh - 2CCFh		—	—	Reserved	—
SETUPADDRESS	R/W	2CD0h	—	B4h	Setup DMA Current Address (7)	3-137
SETUPEND	R/W	2CD4h	—	B5h	Setup DMA End Address (7)	3-138
—	2CD8h - 2CFFh		—	—	Reserved	—
WR0	WO	2D00h	—	C0h	WARP Register 0	3-209
WR1	WO	2D04h	—	C1h	WARP Register 1	"
WR2	WO	2D08h	—	C2h	WARP Register 2	"
…					…	…
WR63	WO	2DFCh	—	FFh	WARP Register 63	3-209
—	2E00h-3BFFh		—	—	Reserved	—
PALWTADD	R/W	3C00h	3C8h	—	Palette RAM Write Address	3-307
PALDATA	R/W	3C01h	3C9h	—	Palette RAM Data	3-305
PIXRDMSK	R/W	3C02h	3C6h	—	Pixel Read Mask	3-308
PALRDADD	R/W	3C03h	3C7h	—	Palette RAM Read Address. This register is WO for I/O accesses.	3-306
—	3C04h - 3C09h		—	—	Reserved	—
X_DATAREG	R/W	3C0Ah	—	—	Indexed Data	3-309
—	—	—	—	00h - 03h: Reserved		—
XCURADDL	R/W	—	—	04h	Cursor Base Address Low	3-320
XCURADDH	R/W	—	—	05h	Cursor Base Address High	3-319
XCURCTRL	R/W	—	—	06h	Cursor Control	3-322
—	—	—	—	07h	Reserved	—
XCURCOL0RED	R/W	—	—	08h	Cursor Control 0 Red	3-321
XCURCOL0GREEN	R/W	—	—	09h	Cursor Control 0 Green	"
XCURCOL0BLUE	R/W	—	—	0Ah	Cursor Control 0 Blue	"

Table 2-4: Register Map (Part 9 of 13)

Register Mnemonic Name	Access	Memory Address(1)	I/O Address(2)	Index	Description/Comments	Page
—	—	—	—	0Bh	Reserved	—
XCURCOL1RED	R/W	—	—	0Ch	Cursor Control 1 Red	3-321
XCURCOL1GREEN	R/W	—	—	0Dh	Cursor Control 1 Green	3-321
XCURCOL1BLUE	R/W	—	—	0Eh	Cursor Control1 Blue	"
—	—	—	—	0Fh	Reserved	—
XCURCOL2RED	R/W	—	—	10h	Cursor Control 2 Red	3-321
XCURCOL2GREEN	R/W	—	—	11h	Cursor Control 2 Green	"
XCURCOL2BLUE	R/W	—	—	12h	Cursor Control 2 Blue	"
—	—	—	—	13h - 17h: Reserved		—
XVREFCTRL	R/W	—	—	18h	Voltage Reference Control	3-340
XMULCTRL	R/W	—	—	19h	Multiplex Control	3-329
XPIXCLKCTRL	R/W	—	—	1Ah	Pixel Clock Control	3-330
—	—	—	—	1Bh - 1Ch: Reserved		—
XGENCTRL	R/W	—	—	1Dh	General Control	3-324
XMISCCTRL	R/W	—	—	1Eh	Miscellaneous Control	3-328
—	—	—	—	1Fh - 29h: Reserved		—
XGENIOCTRL	R/W	—	—	2Ah	General Purpose I/O Control	3-325
XGENIODATA	R/W	—	—	2Bh	General Purpose I/O Data	3-326
XSYSPLLM	R/W	—	—	2Ch	SYSPLL M Value	3-336
XSYSPLLN	R/W	—	—	2Dh	SYSPLL N Value	3-337
XSYSPLLP	R/W	—	—	2Eh	SYSPLL P Value	3-338
XSYSPLLSTAT	RO	—	—	2Fh	SYSPLL Status	3-339
—	—	—	—	30h - 37h: Reserved		—
XZOOMCTRL	R/W	—	—	38h	Zoom Control	3-341
—	—	—	—	39h	Reserved	—
XSENSETEST	R/W	—	—	3Ah	Sense Test	3-335
—	—	—	—	3Bh	Reserved	—
XCRCREML	RO	—	—	3Ch	CRC Remainder Low	3-318
XCRCREMH	RO	—	—	3Dh	CRC Remainder High	3-317
XCRCBITSEL	R/W	—	—	3Eh	CRC Bit Select	3-316
—	—	—	—	3Fh	Reserved	—
XCOLMSK	R/W	—	—	40h	Color Key Mask	3-314
—	—	—	—	41h	Reserved	—
XCOLKEY	R/W	—	—	42h	Color Key	3-312
—	—	—	—	43h	Reserved	—
XPIXPLLAM	R/W	—	—	44h	PIXPLL M Value Register Set A	3-331

Table 2-4: Register Map (Part 10 of 13)

Register Mnemonic Name	*Access*	*Memory Address(1)*	*I/O Address(2)*	*Index*	*Description/Comments*	*Page*
XPIXPLLAN	R/W	—	—	45h	PIXPLL N Value Register Set A	3-332
XPIXPLLAP	R/W	—	—	46h	PIXPLL P Value Register Set A	3-333
—	—	—	—	47h	Reserved	—
XPIXPLLBM	R/W	—	—	48h	PIXPLL M Value Register Set B	3-331
XPIXPLLBN	R/W	—	—	49h	PIXPLL N Value Register Set B	3-332
XPIXPLLBP	R/W	—	—	4Ah	PIXPLL P Value Register Set B	3-333
—	—	—	—	4Bh	Reserved	—
XPIXPLLCM	R/W	—	—	4Ch	PIXPLL M Value Register Set C	3-331
XPIXPLLCN	R/W	—	—	4Dh	PIXPLL N Value Register Set C	3-332
XPIXPLLCP	R/W	—	—	4Eh	PIXPLL P Value Register Set C	3-333
XPIXPLLSTAT	RO	—	—	4Fh	PIXPLL Status	3-334
—	—	—	—	50h	Reserved	—
XKEYOPMODE	R/W	—	—	51h	KEYING Operating Mode	3-327
XCOLMSK0RED	R/W	—	—	52h	Color Mask 0 Red	3-315
XCOLMSK0GREEN	R/W	—	—	53h	Color Mask 0 Green	"
XCOLMSK0BLUE	R/W	—	—	54h	Color Mask 0 Blue	"
XCOLKEY0RED	R/W	—	—	55h	Color Key 0 Red	3-313
XCOLKEY0GREEN	R/W	—	—	56h	Color Key 0 Blue	"
XCOLKEY0BLUE	R/W	—	—	57h	Color Key 0 Green	"
—	—	—	—	58h - 5Fh: Reserved		—
XCURCOL3RED	R/W	—	—	60h	Cursor Color 3 Red	3-321
XCURCOL3GREEN	R/W	—	—	61h	Cursor Color 3 Green	"
XCURCOL3BLUE	R/W	—	—	62h	Cursor Color 3 Blue	"
XCURCOL4RED	R/W	—	—	63h	Cursor Color 4 Red	"
XCURCOL4GREEN	R/W	—	—	64h	Cursor Color 4 Green	"
XCURCOL4BLUE	R/W	—	—	65h	Cursor Color 4 Blue	"
XCURCOL5RED	R/W	—	—	66h	Cursor Color 5 Red	"
XCURCOL5GREEN	R/W	—	—	67h	Cursor Color 5 Green	"
XCURCOL5BLUE	R/W	—	—	68h	Cursor Color 5 Blue	"
XCURCOL6RED	R/W	—	—	69h	Cursor Color 6 Red	"
XCURCOL6GREEN	R/W	—	—	6Ah	Cursor Color 6 Green	"
XCURCOL6BLUE	R/W	—	—	6Bh	Cursor Color 6 Blue	"
XCURCOL7RED	R/W	—	—	6Ch	Cursor Color 7 Red	"
XCURCOL7GREEN	R/W	—	—	6Dh	Cursor Color 7 Green	"
XCURCOL7BLUE	R/W	—	—	6Eh	Cursor Color 7 Blue	"
XCURCOL8RED	R/W	—	—	6Fh	Cursor Color 8 Red	"

Table 2-4: Register Map (Part 11 of 13)

Register Mnemonic Name	Access	Memory Address(1)	I/O Address(2)	Index	Description/Comments	Page
XCURCOL8GREEN	R/W	—	—	70h	Cursor Color 8 Green	"
XCURCOL8BLUE	R/W	—	—	71h	Cursor Color 8 Blue	"
XCURCOL9RED	R/W	—	—	72h	Cursor Color 9 Red	"
XCURCOL9GREEN	R/W	—	—	73h	Cursor Color 9 Green	"
XCURCOL9BLUE	R/W	—	—	74h	Cursor Color 9 Blue	"
XCURCOL10RED	R/W	—	—	75h	Cursor Color 10 Red	"
XCURCOL10GREEN	R/W	—	—	76h	Cursor Color 10 Green	"
XCURCOL10BLUE	R/W	—	—	77h	Cursor Color 10 Blue	"
XCURCOL11RED	R/W	—	—	78h	Cursor Color 11 Red	"
XCURCOL11GREEN	R/W	—	—	79h	Cursor Color 11 Green	"
XCURCOL11BLUE	R/W	—	—	7Ah	Cursor Color 11 Blue	"
XCURCOL12RED	R/W	—	—	7Bh	Cursor Color 12 Red	"
XCURCOL12GREEN	R/W	—	—	7Ch	Cursor Color 12 Green	"
XCURCOL12BLUE	R/W	—	—	7Dh	Cursor Color 12 Blue	"
XCURCOL13RED	R/W	—	—	7Eh	Cursor Color 13 Red	"
XCURCOL13GREEN	R/W	—	—	7Fh	Cursor Color 13 Green	"
XCURCOL13BLUE	R/W	—	—	80h	Cursor Color 13 Blue	"
XCURCOL14RED	R/W	—	—	81h	Cursor Color 14 Red	"
XCURCOL14GREEN	R/W	—	—	82h	Cursor Color 14 Green	"
XCURCOL14BLUE	R/W	—	—	83h	Cursor Color 14 Blue	"
XCURCOL15RED	R/W	—	—	84h	Cursor Color 15 Red	"
XCURCOL15GREEN	R/W	—	—	85h	Cursor Color 15 Green	"
XCURCOL15BLUE	R/W	—	—	86h	Cursor Color 15 Blue	"
—	—	3C0Bh	—	—	Reserved	—
CURPOSXL	R/W	3C0Ch	—	—	Cursor Position X LSB	3-304
CURPOSXH	R/W	3C0Dh	—	—	Cursor Position X MSB	"
CURPOSYL	R/W	3C0Eh	—	—	Cursor Position Y LSB	"
CURPOSYH	R/W	3C0Fh	—	—	Cursor Position Y MSB	"
—	3C10-3CFFh		—	—	Reserved	—
BESA1ORG	WO	3D00h	—	—	BES Buffer A-1 Org.	3-45
BESA2ORG	WO	3D04h	—	—	BES Buffer A-2 Org.	3-47
BESB1ORG	WO	3D08h	—	—	BES Buffer B-1 Org.	3-49
BESB2ORG	WO	3D0Ch	—	—	BES Buffer B-2 Org.	3-51
BESA1CORG	WO	3D10h	—	—	BES Buffer A-1 Chroma Org.	3-44
BESA2CORG	WO	3D14h	—	—	BES Buffer A-2 Chroma Org.	3-46
BESB1CORG	WO	3D18h	—	—	BES Buffer B-1 Chroma Org.	3-48

Table 2-4: Register Map (Part 12 of 13)

Register Mnemonic Name	Access	Memory Address(1)	I/O Address(2)	Index	Description/Comments	Page
BESB2CORG	WO	3D1Ch	—	—	BES Buffer B-2 Chroma Org.	3-50
BESCTL	R/W	3D20h	—	—	BES Control	3-52
BESPITCH	WO	3D24h	—	—	BES Pitch	3-60
BESHCOORD	WO	3D28h	—	—	BES Horiz. Coordinates	3-55
BESVCOORD	WO	3D2Ch	—	—	BES Vert. Coordinates	3-66
BESHISCAL	WO	3D30h	—	—	BES Horiz. Inv. Scaling Factor	3-56
BESVISCAL	WO	3D34h	—	—	BES Vert. Inv. Scaling Factor	3-67
BESHSRCST	WO	3D38h	—	—	BES Horiz. Source Start	3-59
BESHSRCEND	WO	3D3Ch	—	—	BES Horiz. Source Ending	3-57
—	—	3D40h - 3D44h		—	Reserved	—
BESV1WGHT	WO	3D48h	—	—	BES Field 1 Vert. Weight Start	3-64
BESV2WGHT	WO	3D4Ch	—	—	BES Field 2 Vert. Weight Start	3-65
BESHSRCLST	WO	3D50h	—	—	BES Horiz. Source Last	3-58
BESV1SRCLST	WO	3D54h	—	—	BES Field 1 Vert. Source Last Pos.	3-62
BESV2SRCLST	WO	3D58h	—	—	BES Field 2 Vert. Source Last Pos.	3-63
—	—	3D5Ch - 3DBCh		—	Reserved	—
BESGLOBCTL	R/W	3DC0h	—	—	BES Global Control	3-54
BESSTATUS	RO	3DC4h	—	—	BES Status	3-61
—	3DC8h - 3DFFh		—	—	Reserved	—
VINCTL0	WO	3E00h	—	—	Video Input Control Window 0	3-194
VINCTL1	WO	3E04h	—	—	Video Input Control Window 1	3-195
VBIADDR0	WO	3E08h	—	—	VBI Address Window 0	3-186
VBIADDR1	WO	3E0Ch	—	—	VBI Address Window 1	3-187
VINADDR0	WO	3E10h	—	—	Video Input Address Window 0	3-191
VINADDR1	WO	3E14h	—	—	Video Input Address Window 1	3-192
VINNEXTWIN	WO	3E18h	—	—	Video Input Next Window	3-196
VINCTL	WO	3E1Ch	—	—	Video Input Control	3-193
—	3E20h - 3E2Fh		—	—	Reserved	—
VSTATUS	RO	3E30h	—	—	Video Status	3-197
VICLEAR	WO	3E34h	—	—	Video Interrupt Clear	3-189
VIEN	R/W	3E38h	—	—	Video Interrupt Enable	3-190
—	—	3E3Ch	—	—	Reserved	—
CODECCTL	WO	3E40h	—	—	CODEC Control	3-69
CODECADDR	WO	3E44h	—	—	CODEC Buffer Start Address	3-68
CODECHOSTPTR	WO	3E48h	—	—	CODEC Host Pointer	3-72
CODECHARDPTR	RO	3E4Ch	—	—	CODEC Hard Pointer	3-71

Table 2-4: Register Map (Part 13 of 13)

Register Mnemonic Name	*Access*	*Memory Address*(1)	*I/O Address*(2)	*Index*	*Description/Comments*	*Page*
CODECLCODE	RO	3E50h	—	—	CODEC LCODE Pointer	3-73
—	3E54 - 3FFF		—	—	Reserved	—

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

(2) I/O space accesses are decoded only if VGA emulation is active (see the **OPTION** configuration register) and **iospace** = 1 (see the **DEVCTRL** configuration register).

(3) 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.

(4) Accessing a register in this range instructs the drawing engine to start a drawing operation. For the General Purpose index value, use the index value found in the corresponding register and/or the index with 40h.

(5) 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.)

(6) 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.

(7) These registers are not writable through **MGABASE1** + 2C40h to 2C48h and **MGABASE1** + 2CDxh. They can only be written via bus mastering operations from the MGA-G200. They ***can*** be read through **MGABASE1** + 2C4xh and **MGABASE1** + 2CDxh.

Legend:

A shaded cell indicates an index used by the General Purpose DMA.

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 1-10.

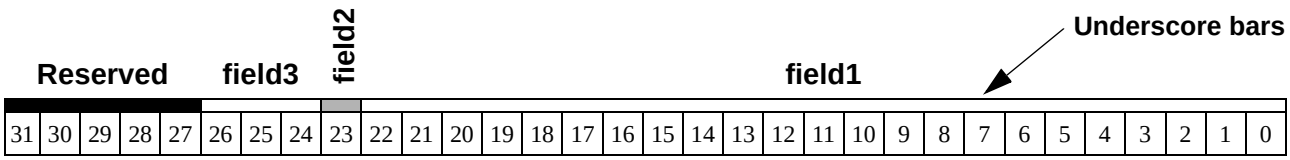
3.1 Power Graphic Mode Register Descriptions

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>



field1 <22:0> Field 1. Detailed description of the **field1** field of the **SAMPLE_CS** register, which comprises bits 22 to 0. ***Font and case changes within the text indicate a register or field.***

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

Reset Value

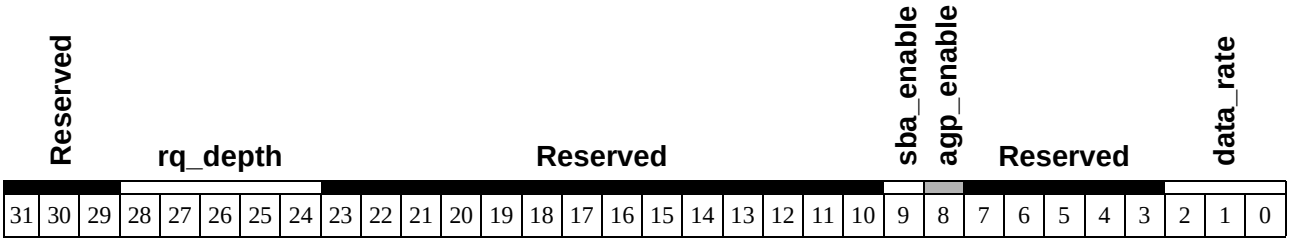
Here are some of the symbols that appear as part of a register's reset value:

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

Address F8h (CS) (applies to MGA-G200-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:

data_rate	*description*
'000'	reset value
'001'	1 x data rate
'010'	2 x data rate
'1XX'	Reserved
'X11'	Reserved

agp_enable <8> When set, this bit enables the MGA-G200 to initiate AGP operation.

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

rq_depth <28:24> This should be programmed with the maximum number of pipelined operations that the MGA-G200 is allowed to queue. This value should be equal, or less, than the value reported in the **rq** field of **AGP_STS** register.

Reserved **<31:29> <23:10> <7:3>**

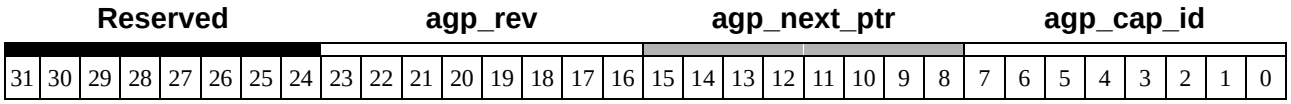
Reserved. Writing has no effect. Reading will give '0's.

Note: To initiate the AGP cycle, both **agp_enable** and **sba_enable** must be set together with the proper **data_rate** value (either 1X or 2X). Furthermore, for the AGP-2X cycle, the **agp2xpllen** bit of the **AGP_PLL** register, ***must*** be set.

Address F0h (CS) for MGA-G200-AGP only

Attributes RO, BYTE/WORD/DWORD, STATIC

Reset Value `0000 0000 0001 0000 0000 0000 0000 0010`b



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

agp_next_ptr <15:8> This field contains the hard-coded value of 00h, which indicates that there is no other capabilities in the list.

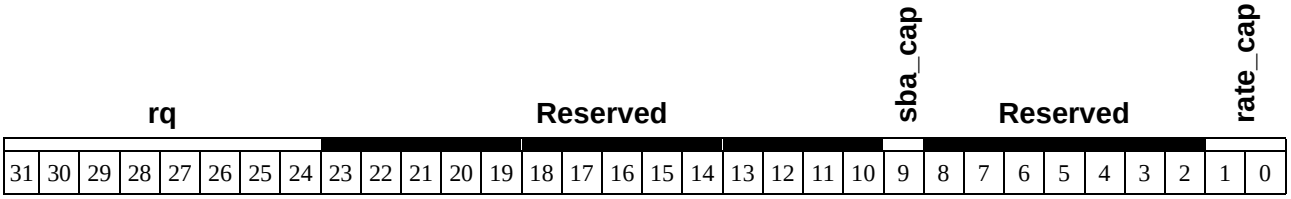
agp_rev <23:16> This field contains the AGP specification revision to which this device complies: 10h (as in 1.0)

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

Address F4h (CS) for MGA-G200-AGP only

Attributes RO, BYTE/WORD/DWORD, STATIC

Reset Value 0001 1111 0000 0000 0000 0010 0000 0011b



rate_cap <1:0> The hard-coded ‘11b’ indicates that the device supports both AGP transfer rate modes (1X, 2X).

sba_cap <9> The hard-coded ‘1’ indicates that the device supports AGP Side band addressing.

rq <31:24> The hard-coded ‘1Fh’ indicates that the device can manage 32 outstanding AGP Requests.

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

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

Address 34h (CS)

Attributes RO, BYTE/WORD/DWORD, STATIC

Reset Value 0000 0000 0000 0000 0000 0000 1101 1100b

Reserved																						cap_ptr							
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

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

Reserved <31:8> 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 0001`b

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

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

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

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

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

Address 04h (CS)

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

Reset Value 0000 0010 1001 0000 0000 0000 0000 0000b

detparerr	sigsyserr	recmastab	rectargab	sigtargab	devseltim	Reserved	fastbackcap	udfsup	cap66mhz	caplist	Reserved	serrenable	waitcycle	resparerr	vgasnoop	memwrien	specialcycle	busmaster	memspace	iospace
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

iospace
R/W <0>

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

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

memspace
R/W <1>

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

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

busmaster
R/W <2>

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

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

specialcycle
RO <3>

The hard-coded '0' indicates that the MGA will ***not*** respond to a special cycle.

memwrien
RO <4>

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

vgasnoop
R/W <5>

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

The hard-coded ‘0’ indicates that the device does *not* comply with the PCI 66 MHz timing specification. Writing has no effect.

Note: PCI transactions at 66 Mhz are supported as per the AGP timing specification.

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 return ‘0’s.

detparerr
RO <31>

MGA does *not* detect parity errors. Writing has no effect. Reading will return ‘0’s.

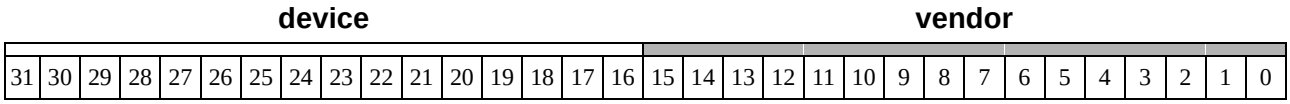
Reserved **<19:9> <24>**

Reserved. Writing has no effect. Reading will return ‘0’s.

Address 00h (CS)

Attributes RO, BYTE/WORD/DWORD, STATIC

Reset Value `0000 0101 0010 000? 0001 0000 0010 1011`b



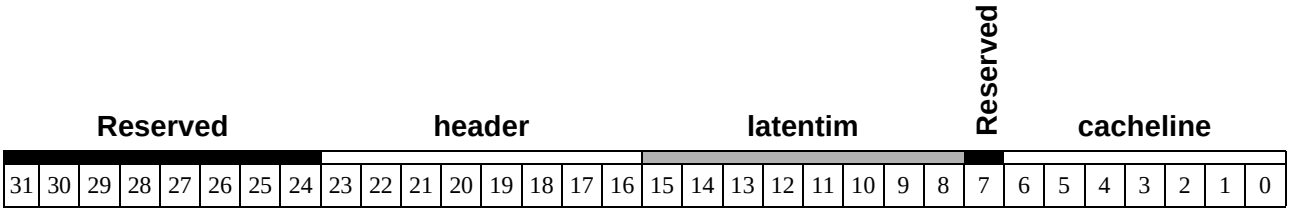
vendor <15:0> This field contains the Matrox manufacturer identifier for PCI: 102Bh.

device <31:16> This field contains the Matrox device identifier, which for the MGA-G200-PCI is: 0520h; for the MGA-G200-AGP it is: 0521h.

Address 0Ch (CS)

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

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



cacheline <6:0>

This read/write field specifies the system cacheline size in units of 32-bit words. This field, together with **enhmemacc** (**OPTION**), controls the type of PCI command used by the bus master (could issue either memory read, memory read multiply, or memory read line). Any value can be programmed, but the value used by the controller will be the power of 2 smaller or equal to the value programmed. Values smaller than 4 will be considered as 0.

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

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

header RO <23:16>

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

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

Reserved. Writing has no effect. Reading will return ‘0’s.

Address 3Ch (CS)

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

Reset Value `0010 0000 0001 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 in 1/4 µs, assuming a clock rate of 33 MHz.

maxlat
RO <31:24>

This field specifies how often the PCI device must gain access to the PCI bus in 1/4 µs, assuming a clock rate of 33 MHz.

Address 48h (CS)
Attributes R/W, BYTE/WORD/DWORD, STATIC
Reset Value unknown

mga_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

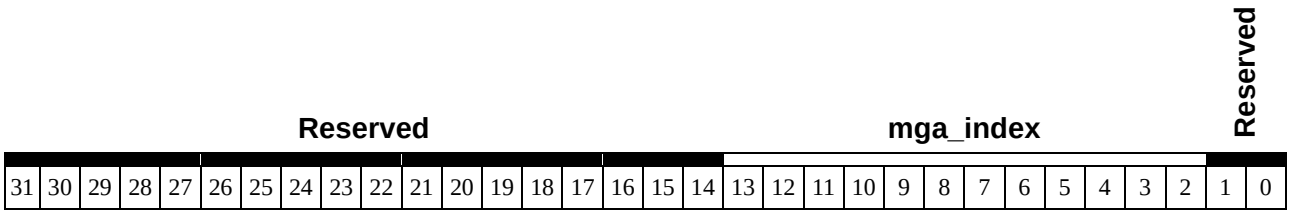
mga_data <31:0> Data. Will read or write data at the control register address provided by **MGA_INDEX**. If **MGA_INDEX** does *not* point to a valid range, unknown data will be returned.

➧ ***Note:*** The **MGA_INDEX** and **MGA_DATA** registers cannot be used to access Pseudo-DMA windows (DMAWIN). (see page 4-29)

Address 44h (CS)

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

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



mga_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 return ‘0’s.

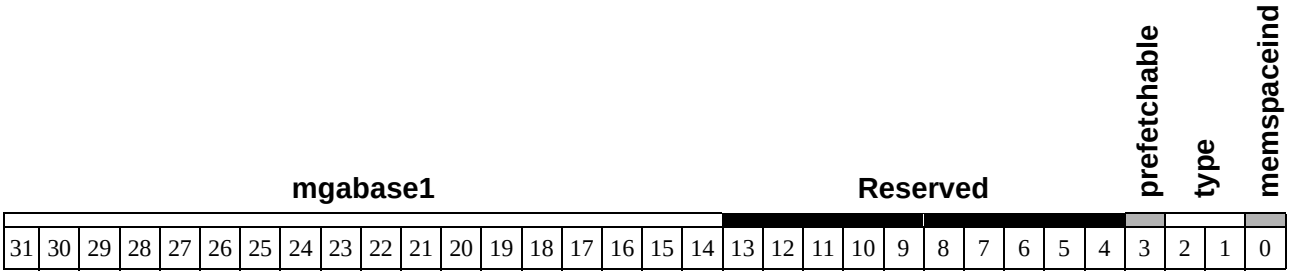
➧ ***Note:*** The **MGA_INDEX** and **MGA_DATA** registers cannot be used to access Pseudo-DMA windows (DMAWIN)(see page 4-29).

➧ ***Note:*** The valid range for **MGA_INDEX** is 1C00h to 3FFCh (see Table 2-3 for device addresses).

Address 14h (CS)

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

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



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

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

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

mgabase1 <31:14> Specifies the base address of the MGA memory mapped control registers (16 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 priority):

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

An aperture will be decoded *only* if the preceding ones are ***not*** decoded. If no aperture is decoded, the MGA-G200 will ***not*** respond to memory accesses.

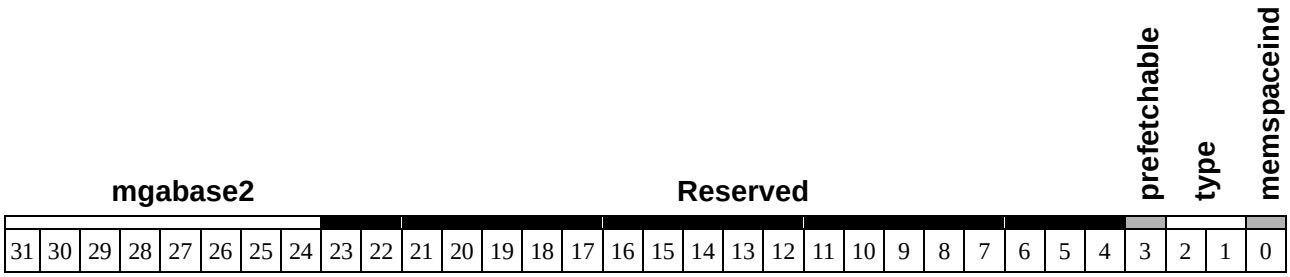
Decoding of an aperture is related to the address (if it corresponds to one of the base addresses), the command, and some of the control bits, such as **memspace**, **biosen**, **romen**, and **rammapen**.

Reserved <13:4> Reserved. When writing to this register, the bits in this field ***must*** be set to ‘0’. Reading will return ‘0’s.

Address 10h (CS)

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

Reset Value 0000 0000 0000 0000 0000 0000 0000 1000b



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

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

prefetchable RO <3> A ‘1’ indicates that this space can be prefetchable (better system performance can be achieved when the bridge enables prefetching into that range).

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

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

An aperture will be decoded *only* if the preceding ones are ***not*** decoded. If no aperture is decoded, the MGA-G200 will ***not*** respond to memory accesses.

Decoding of an aperture is related to the address (if it corresponds to one of the base addresses), the command, and some of the control bits, such as **memspace**, **biosen**, **romen**, and **rammapen**.

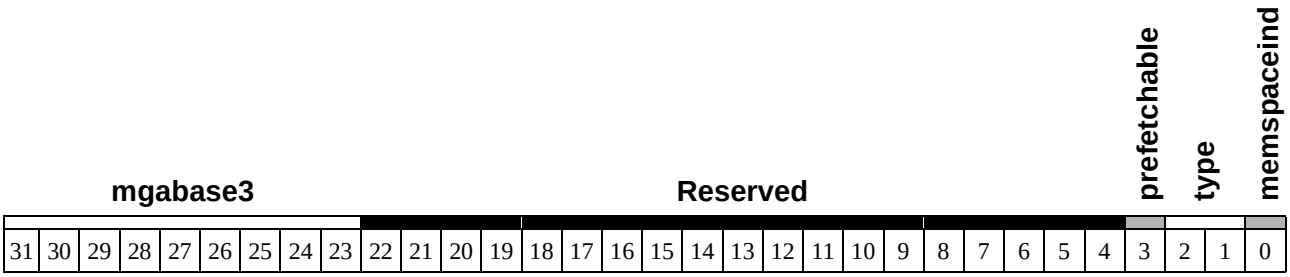
When **mgamode** = 0 (**CRTCEXT3**<7>), the MGA frame buffer Aperture is ***not*** usable.

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

Address 18h (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.

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 priority:

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

An aperture will be decoded *only* if the preceding ones are ***not*** decoded. If no aperture is decoded, the MGA-G200 will ***not*** respond to memory accesses.

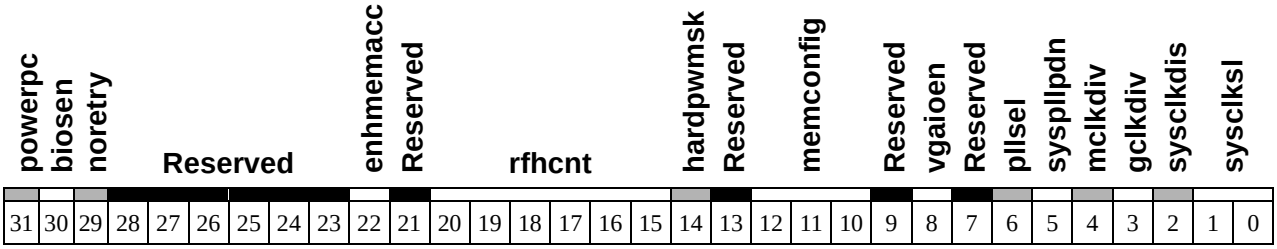
Decoding of an aperture is related to the address (if it corresponds to one of the base addresses), the command, and some of the control bits, such as **memspace**, **biosen**, **romen**, and **rammapen**.

Reserved <22:4> Reserved. When writing to this register, the bits in this field ***must*** be set to ‘0’. Reading will return ‘0’s.

Address 40h (CS)

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

Reset Value `0S00 0000 0000 0000 0000 000S 0000 0000`b



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

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

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

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

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

- 0: divide by 2
- 1: divide by 3/2

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

- 0: divide by 2
- 1: divide by 3/2

syspllpdN <5> System PLL power down.

- 0: power down
- 1: power up

pllsel <6> PLL Select. When set to '1', the pixel clock comes from **syspll**.

- 0: PLL P1 drives the pixel clock
  PLL P2 drives the system clock
- 1: PLL P1 drives the system clock
  PLL P2 drives the pixel clock

Note: This bit ***must*** be set to '0' for normal operation. (A '1' will swap **pixpll** with **syspll**).

vgaioen <8>

VGA I/O map Enable.

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

On hard reset, the sampled vgaboot strap (HDATA[0]) will replace the **vgaioen** value.

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

memconfig <12:10>

Memory Configuration. This field ***must*** be loaded before initiating a memory reset or attempting to read or write to the frame buffer.

Address boundaries for the bank select, chip select, row address, and column address are determined using **memconfig**. This signal should ***not*** change during normal operation.

Definition of **memconfig** (2:0)

memconfig (0)	*Number of banks on Base-board*
'0'	one bank on base-board (as in: mscN[2:1] re-mapped to mcsN[3:2])
'1'	two banks on the base-board

memconfig (2:1)	*Memory Organization of parts used in Frame Buffer*
'00'	(8Mb) 2 x 128K x32; 512 rows, 256 columns
'01'	(16Mb) 2 x 256K x 32; 1024 rows, 256 columns
'10'	(16Mb) 2 x 512K x 16; 2048 rows, 256 columns
'11'	(16Mb) 4 x 128K x 32; 512 rows, 256 columns

hardpwmsk <14>

Hardware Plane Write Mask. This field is used to enable SGRAM special functions. This field must ***always*** be set to '0' when SDRAM is used. (when SGRAM is used, software ***must*** set **hardpwmsk** to '1' in order to take advantage of special SGRAM functions).

This field must ***always*** be loaded ***before*** attempting to write to the frame buffer and should ***not*** be changed during normal operation.

- 0: Special SGRAM functions are ***not*** available; however, a plane write mask cycle will be emulated in the MGA-G200 at a reduced performance level.
- 1: Special SGRAM functions are enabled, so plane write mask operations will be performed by the memory (with optimal performance) and block mode operations are available.

➧ ***Note:*** **hardpwmsk** must ***never be set to '1'*** when the memory does ***not*** consist of ***SGRAM***.

rfhcnt <20:15>

Refresh counter. Defines the rate of the MGA-G200's memory refresh. Page cycles will ***not*** be interrupted by a refresh request unless a second request is queued (in this case, the refresh request becomes the highest priority after the screen refresh). Since all banks have to be pre-charged, both queued refreshes will keep this new highest priority.

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

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

➧ ***Note:*** Setting **rfhcnt** to zero ***halts*** the memory refresh.

enhmemacc <22>

Enable the use of advance Read commands by the PCI Master (MRL & MRM).

noretry <29>

Retry disable. A '1' disables generation of the retry sequence and the delayed read on the PCI bus (except during a VGA snoop cycle). At this setting, PCI latency rules may be violated.

biosen <30>

BIOS Enable. On hard reset, the sampled bios boot strap (HDATA[1]) is loaded into this field.

- 0: The **ROMBASE** space is automatically disabled. A small serial eeprom can be present on board.
- 1: The **ROMBASE** space is enabled - **rombase** must be correctly initialized since it contains ***unpredictable*** data. A big serial eeprom is present on board.

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** + 2000h to **MGABASE1** + 3BFFh and **MGABASE1** + 3D00h to **MGABASE1** + 3FFFh. This swapping allows a Big-Endian processor to access the information in the same manner as a Little-Endian processor.

➧ ***Note:*** There is ***no*** swapping in the configuration space.

Reserved

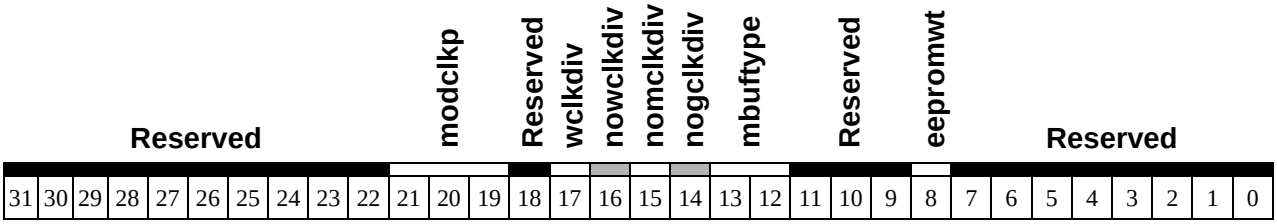
<7> <9> <13> <21> <28:23>

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

Address 50h (CS)

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

Reset Value 0000 0000 00SS S000 1011 0000 0000 0000b



eepromwt <8> EEPROM write enable. When set to '1', a write access to the BIOS EPROM aperture will program that location. When set to '0', write access to the BIOS EPROM aperture has ***no*** effect.

mbuftype <13:12> Memory Buffer Type. This field ***must*** be loaded before initiating a memory reset or attempting to read or write to the frame buffer.

- mbuftype(0). Memory control signal Buffer Type. This bit controls the buffer type for the mrasN, mcasN, mweN, mdsf, mcsN[3:0], mdqm[7:0] and ma[11:0] pins.
  0: LVTTL buffer configuration
  1: SSTL-3 buffer configuration requiring vref pin
- mbuftype(1). Memory data and byte enable signal buffer type. This bit controls the buffer type for the mdq[63:0] pins.
  0: LVTTL buffer configuration
  1: SSTL-3 buffer configuration requiring vref pin

nogclkdiv <14> No Graphics Clock Divider. Selects whether or not to bypass the clock divider.

- 0: use the **GCLKDIV** field (**GCLK** = **SYSCLK**/**GCLKDIV**)
- 1: do *not* divide **SYSCLK** for **GCLK** (**GCLK** = **SYSCLK**).

nomclkdiv <15> No Memory Clock Divider select. Selects whether or not to bypass the clock divider.

- 0: use the **MCLKDIV** field (**MCLK** = **SYSCLK**/**MCLKDIV**)
- 1: do ***not*** divide **SYSCLK** for **MCLK** (**MCLK** = **SYSCLK**).

nowclkdiv <16> No WARP Clock Divider select. Selects whether or not to bypass the clock divider.

- 0: use the **WCLKDIV** field (**WCLK** = **SYSCLK**/**WCLKDIV**)
- 1: do ***not*** divide **SYSCLK** for **WCLK** (**WCLK** = **SYSCLK**).

wclkdiv <17> WARP Clock Divider select. Selects the ratio by which the system clock is divided in order to generate the WARP clock:

- 0: divide by 2
- 1: divide by 3/2

➨ ***Note:*** The WARP clock must be programmed so that it is always faster than the PCI clock.

modclkp <21:19> Module Clock Period. On hard reset, the sampled module clock period strap (MDQ<31:29>) value will replace the value of **modclkp**.

This field is used to determine the frequency at which an LVTTL memory expansion

module is designed to operate.

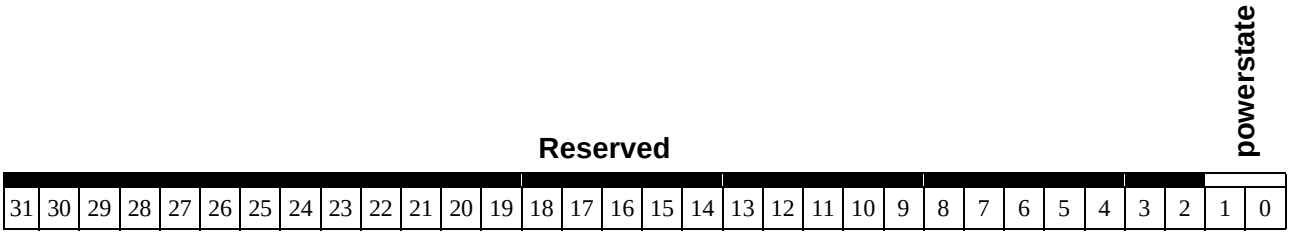
Reserved **<7:0> <11:9> <18> <31:22>**

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

Address E0 h (CS)

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

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



powerstate <1:0> This two bit field is used to determine the current power state of the device, and to set the device into a new power state. Writing to this register will place the device in the appropriate power state.

powerstate	*Video Controller Power State definition*[1][2] [3]
'00'	D0 • Back-end: On • Video Controller Context: Preserved • Video Memory Contents: Preserved • Interrupts: Possible
'11'	D3 (Power may be removed) • Back-end: Off • Video Controller Context: Lost • Video memory Contents: Lost • Interrupts: Disabled Mask bits of **DEVCTRL** register: • **iospace** = '0' • **memspace** = '0' • **busmaster** = '0' Power down of SSTL buffers (memory bus) • data/address/command • by selecting LVTTL buffers Power down of RAMDAC section: • **pixclkdis** = '1' • **dacpdN** = '0' • **ramcs** ='0' • **pixpllpdN** = '0' Power down of MEMORY/GRAPHIC/WARP section: • **sysclkdis** = '1' • **syspllpdN** = '0' Power down of TCACHE: Power down of ZORAN_I_33 and ZORAN_I_34:

(1) Since D1 and D2 mode are not supported, only writes with valid values will modify the powerstate field.

(2) PCI BPMI Spec. states that hardware has at least 16 PCI clocks after the powerstate bits were written to D3 before **PCLK** may be stopped.

(3) An internal reset (used like the hard reset) is generated for 16 PCI clocks when returning from D3 (with power) to D0.

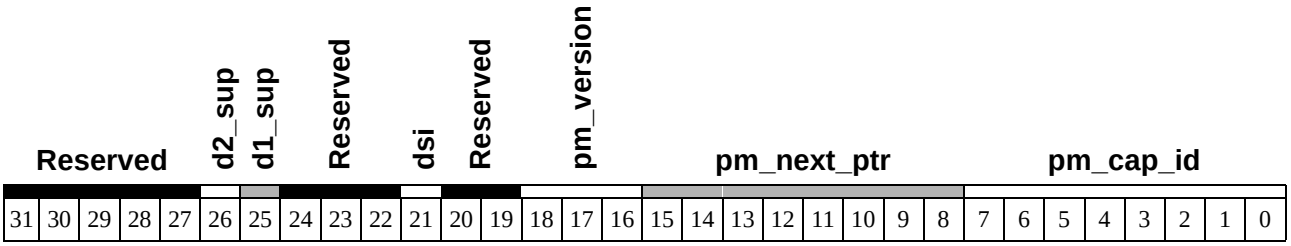
➨ ***Note:*** The above table complies with the Display Device Class Power Management Reference Specification.

Reserved: <31:2> Reserved. When writing to this register, the bits in this field ***must*** be set to '0'.

Address DC h (CS)

Attributes RO, BYTE/WORD/DWORD

Reset Value 0000 0000 0010 0001 ???? 0000 0000 0001b



pm_cap_id <7:0> Power Management Capabilities Identifier. This field contains the PCI Bus Power Management Interface Specification Capabilities Identifier: 01h, which describes the information contained in the capability entry (DCh-E0h)

pm_next_ptr <15:8> Next pointer. This field contains the pointer to the next capability in the link list. For MGA-G200-PCI, a hard-coded “00h” indicates that there are no other capabilities in the list. For MGA-G200-AGP, a hard-coded “F0h” points to the AGP capabilities.

pm_version <18:16> Version. Version “001b” indicates that the MGA-G200 complies with revision 1.0 of the PCI Power Management Interface Specification.

dsi <21> Device Specific Initialization. A ‘1’ indicates that the MGA-G200 requires a device specific initialization sequence when returning from **d3**.

d1_sup <25> The D1 Power Management state is ***not*** supported.

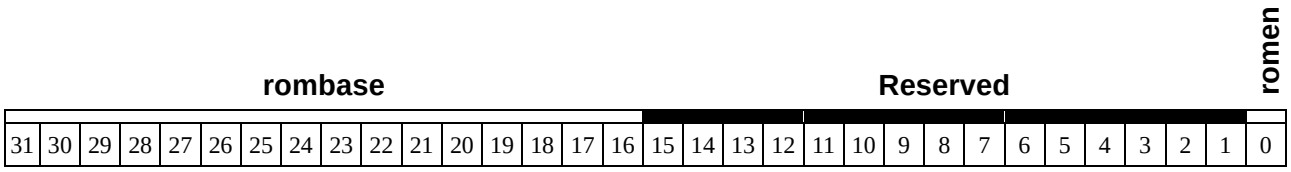
d2_sup <26> The D2 Power Management state is ***not*** supported.

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

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 128 bytes, 32 KBytes, or 64 KBytes).

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)

An aperture will be decoded *only* if the preceding ones are ***not*** decoded. If no aperture is decoded, the MGA-G200 will ***not*** respond to memory accesses.

Decoding of an aperture is related to the address (if it corresponds to one of the base addresses), the command, and some of the control bits, such as **memspace**, **biosen**, **romen**, and **rammapen**.

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

Reserved <15:1> Reserved. When writing to this register, the bits in this field ***must*** be set to '0'. Reading will return '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 0078h (128 bytes ROM used), or 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-G200 chip.

subsysid <31:16> Subsystem ID. This field is reset with the value that is found in word location 007Ah (128 bytes ROM used), at 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-G200 chip.

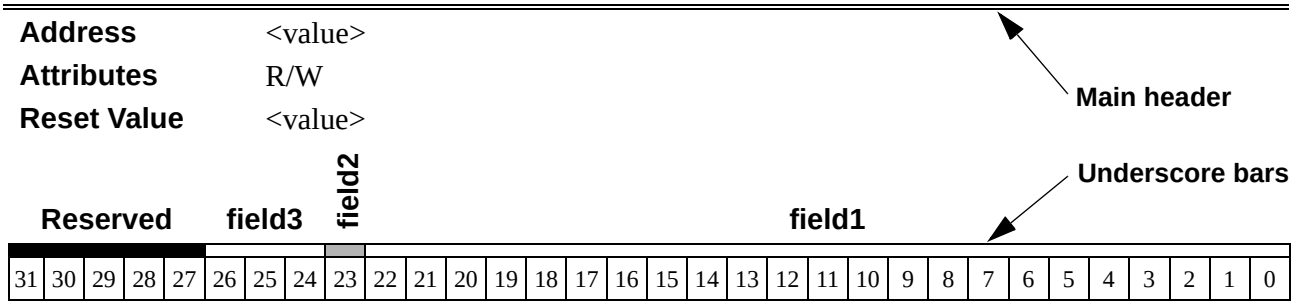
- ***Note***: If the bios boot strap is '0', this may mean that no ROM, or a small ROM, is present (usually on a motherboard implementation). If no ROM is present, the value found on the register will be incorrect. In this case, the system bios ***must write the correct values*** to this register (at location 4Ch) after the delay following a hard reset.
- ***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 will be a delay following a hard reset before this register is initialized. For a small serial eeprom, this delay is 50µs. For a big serial eeprom, this delay is 15µs.
- ***Note:*** Writing to addresses FFF8h to FFFBh is supported for 128 bytes, 32k ROM and 64k ROM. (If the address is bigger than the size of the ROM, it is wrapped around).

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>



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

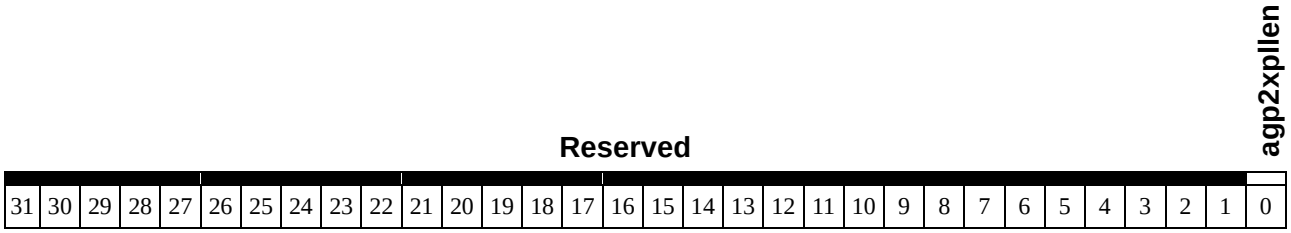
Here are some of the symbols that appear as part of a register's reset value. Most bits are reset on hard reset. Some bits are also reset on soft reset, and they are underlined when they appear in the register description headers.

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

Address MGABASE1 + 1E4Ch (MEM)

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

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



agp2xpllen <0> AGP 2x mode PLL Enable.

- 0: disable the AGP 2x mode PLL oscillations
- 1: enable the AGP 2x mode PLL oscillations

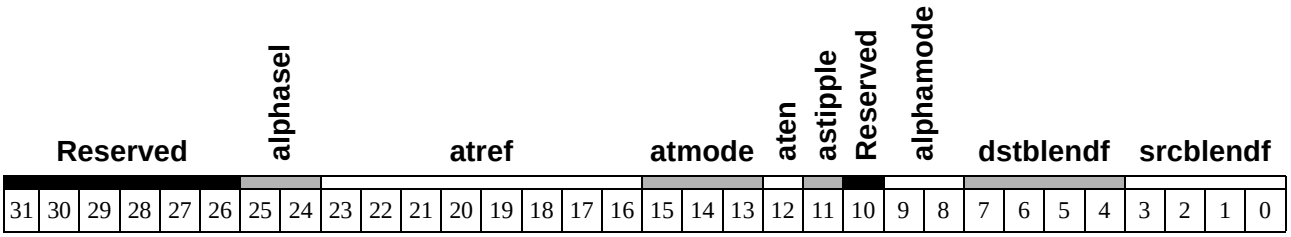
Reserved <31:1> Reserved. When writing to this register, the bits in this field ***must*** be set to '0'.

Note: For MGA-G200-PCI, this register has no effect.

Address MGABASE1 + 2C7Ch (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value 0000 0000 0000 0000 0000 0000 0000 0001b



srcblendf <3:0> Source Blend Function.

srcblendf	*source blending function*
'0000'	ZERO
'0001'	ONE
'0010'	DST_COLOR
'0011'	ONE_MINUS_DST_COLOR
'0100'	SRC_ALPHA
'0101'	ONE_MINUS_SRC_ALPHA
'0110'	DST_ALPHA
'0111'	ONE_MINUS_DST_ALPHA
'1000'	SRC_ALPHA_SATURATE

Note: To disable alpha blending, **srcblendf** must be programmed with 1 and **dstblendf** with 0.

dstblendf <7:4> Destination Blend Function.

dstblendf	*destination blending function*
'0000'	ZERO
'0001'	ONE
'0010'	SRC_COLOR
'0011'	ONE_MINUS_SRC_COLOR
'0100'	SRC_ALPHA
'0101'	ONE_MINUS_SRC_ALPHA
'0110'	DST_ALPHA
'0111'	ONE_MINUS_DST_ALPHA

alphamode <9:8> Select the alpha mode that will be written in the frame buffer.

alphamode	*description*
'00'	FCOL
'01'	alpha channel
'10'	video alpha (1-((1-SrcAlpha) * (1-DstAlpha)))
'11'	RSVD

astipple <11>

Alpha Stipple Mode. Approximation of alpha blending using a dithering matrix. When the alpha stipple mode is selected, only the following Src and Dst function combinations are supported:

Src function	*Dst function*	*screen-door mask*
ZERO	ONE	100% opaque mask
ONE	ZERO	0% mask
SRC_ALPHA	ONE_MINUS_SRC_ALPHA	src_alpha% opaque mask
ONE_MINUS_SRC_ALPHA	SRC_ALPHA	(1-src_alpha)% opaque mask

aten <12>

Alpha Test Enable. Enables the first alpha test.

atmode <15:13>

Alpha Test Mode. Update the pixel when the alpha test comparison with **atref** is true.

atmode		
Value	*Mnemonic*	*Pixel Update*
‘000’	NOACMP	Always
‘001’		Reserved
‘010’	AE	When alpha is =
‘011’	ANE	When alpha is <>
‘100’	ALT	When alpha is <
‘101’	ALTE	When alpha is <=
‘110’	AGT	When alpha is >
‘111’	AGTE	When alpha is >=

atref <23:16>

Alpha Test Reference. Reference value for the alpha test.

alphasel <25:24>

Alpha Select. Determine the alpha for the pixel.

alphasel	*description*
‘00’	alpha from texture
‘01’	diffused alpha
‘10’	modulated alpha
‘11’	transparency? 0: diffused alpha

Note: Alpha testing is done in *two* separate places. Both use the same reference value (atRef) and alpha test mode (atMode). The first test is done right after texture filtering and can be disabled by setting the ‘atEn’ bit to ‘0’, or by setting ‘atmode’ to “ALWAYS”. The second test is done before alpha blending and can be disabled by programming ‘atmode’ to “ALWAYS”.

Reserved **<10> <31:26>**

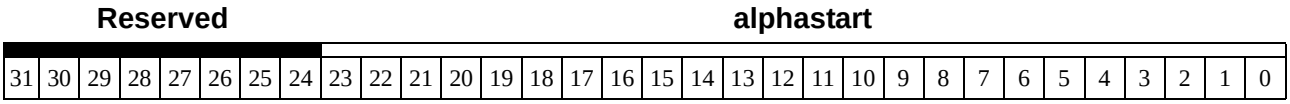
Reserved. When writing to this register, the bits in this field ***must*** be set to ‘0’.

.

Address MGABASE1 + 2C70h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value unknown



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

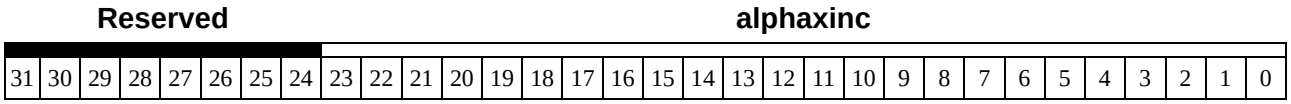
For 3D primitives, the **ALPHASTART** register is used to scan the left edge of the trapezoid for the alpha component of the source. This register must be initialized with its starting alpha value.

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

Address MGABASE1 + 2C74h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value unknown



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

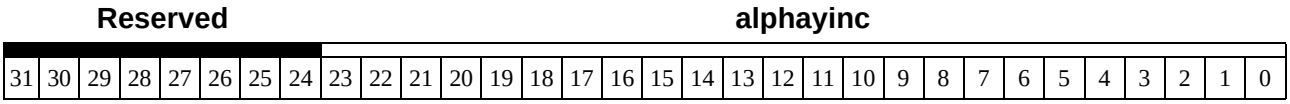
For 3D primitives, the **ALPHAXINC** register holds the alpha increment along the x axis.

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

Address MGABASE1 + 2C78h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value unknown



alphayinc <23:0> Alpha ALU register 2. This field holds a signed 9.15 value in two's complement notation.

For 3D primitives, the **ALPHAYINC** register holds the alpha increment along the y-axis.

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

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

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value unknown

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

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 3-212.
- 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).

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

Reserved	ar1

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

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.

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

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

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value unknown

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

ar2 <17:0> Address register 2. The **ar2** field is an 18-bit signed value in two's complement notation.

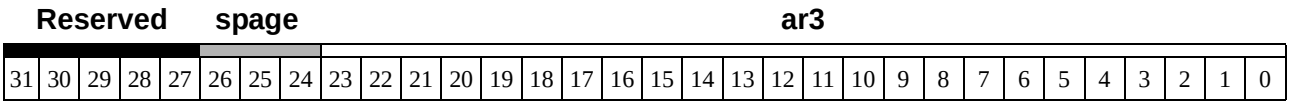
- For AUTOLINE, this register holds the y end address (in pixels). See the **XYEND** register on page 3-212.
- For LINE, it holds the minor axis error increment (initially 2 x 'b' - 2 x 'a').
- For a filled trapezoid, it holds the minor axis increment (-|dXl|).

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



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.

spage <26:24>

These three bits are used as an extension to **ar3** in order to generate a 27-bit source or pattern address (in pixels). They are ***not*** modified by ALU operations.

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

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

Reserved <31:27>

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

Address **MGABASE1** + 1C70h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value unknown

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

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.

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

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

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 3-213. At the end of AUTOLINE the register is loaded with the x end, so it is not necessary to reload the register when drawing a polyline.
- This register is *not* used for LINE without auto initialization.
- For TRAP, it holds the minor axis increment (-|dXr|).
- In BLIT algorithms, **ar5** holds the pitch (in pixels) of the source operand. A negative pitch value specifies that the source is scanned from bottom to top while a positive pitch value specifies a top to bottom scan.

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

Address **MGABASE1** + 1C78h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value unknown

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

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 3-213. 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').

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 bits/pixel only **backcol**<7:0> is used.
- In 16 bits/pixel, only **backcol**<15:0> is used.
- 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 4-21 for the definition of the slice in each mode.

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

- In 8 bits/pixel only **bltcmsk**<7:0> is used.
- In 16 bits/pixel, only **bltcmsk**<15:0> is used.

Address MGABASE1 + 3D10h (MEM)

Attributes WO, STATIC, BYTE, WORD, DWORD

Reset Value unknown

Reserved	besa1corg

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

besa1corg <23:0> Backend Scaler buffer A field 1 Chroma plane Origin. Top left corner of the buffer A field 1 chroma plane data used by the backend scaler. The field **besa1corg** corresponds to a byte address in memory and holds a 24-bit unsigned value which provides an offset value (the base address) in order to position the first pixel chroma plane data to be read from the buffer A field 1 of the window to scale. This register must be initialized when in planar 4:2:0 format (2 planes).

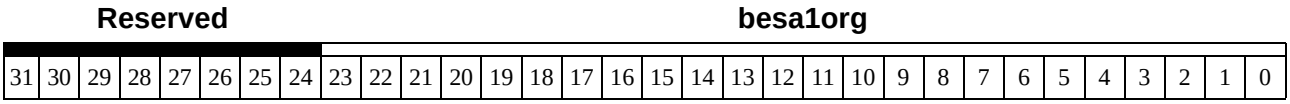
➧ ***Note:*** This address must be aligned on a qword boundary when no horizontal mirror is used (the 3 LSBs must be set to ’000’) and on the last byte of a qword when horizontal mirror is used (the 3 LSBs must be set to ’111’).

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

Address MGABASE1 + 3D00h (MEM)

Attributes WO, STATIC, BYTE, WORD, DWORD

Reset Value unknown



besa1org <23:0> Backend Scaler buffer A field 1 Origin. Top left corner of the buffer A field 1 data used by the backend scaler. The field **besa1org** corresponds to a byte address in memory and it holds a 24-bit unsigned value which provides an offset value (the base address), in order to position the first pixel to read of the buffer A field 1 of the window to scale. In 4:2:0 mode, this register is the luma plane origin.

Note: This address must be aligned on a qword boundary when no horizontal mirror is used (the 3 LSBs must be set to ’000’), and on the last byte of a qword when horizontal mirror is used (the 3 LSBs must be set to ’111’).

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

Address **MGABASE1** + 3D14h (MEM)

Attributes WO, STATIC, BYTE, WORD, DWORD

Reset Value unknown

Reserved								besa2corg																							
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

besa2corg <23:0> Backend Scaler buffer A field 2 Chroma plane Origin. Top left corner of the buffer A field 2 chroma plane data used by the backend scaler. The field **besa2corg** corresponds to a byte address in memory and holds a 24-bit unsigned value which provides an offset value (the base address) in order to position the first pixel chroma plane data to be read from the buffer A field 2 of the window to scale. This register must be initialized when in planar 4:2:0 format (2 planes).

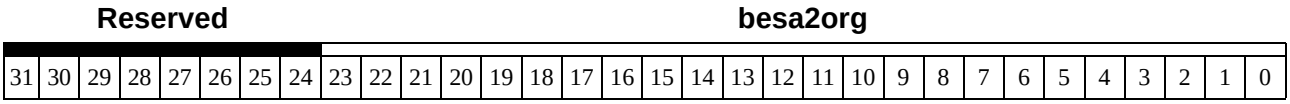
➽ ***Note:*** This address must be aligned on a qword boundary when no horizontal mirror is used (the 3 LSBs must be set to ’000’) and on the last byte of a qword when horizontal mirror is used (the 3 LSBs must be set to ’111’).

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

Address **MGABASE1** + 3D04h (MEM)

Attributes WO, STATIC, BYTE, WORD, DWORD

Reset Value unknown



besa2org <23:0> Backend Scaler buffer A field 2 Origin. Top left corner of the buffer A field 2 data used by the backend scaler. The field **besa2org** corresponds to a byte address in memory and holds a 24-bit unsigned value which provides an offset value (the base address) in order to position the first pixel to be read from the buffer A field 2 of the window to scale. In 4:2:0 mode, this register is the luma plane origin.

Note: This address must be aligned on a qword boundary when no horizontal mirror is used (the 3 LSBs must be set to ’000’) and on the last byte of a qword when horizontal mirror is used (the 3 LSBs must be set to ’111’).

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

Address MGABASE1 + 3D18h (MEM)

Attributes WO, STATIC, BYTE, WORD, DWORD

Reset Value unknown

Reserved	besb1corg

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

besb1corg <23:0> Backend Scaler buffer B field 1 Chroma plane Origin. Top left corner of the buffer B field 1 chroma plane data used by the backend scaler. The field **besb1corg** corresponds to a byte address in memory and holds a 24-bit unsigned value which provides an offset value (the base address) in order to position the first pixel chroma plane data to be read from the buffer B field 1 of the window to scale. This register must be initialized when in planar 4:2:0 format (2 planes).

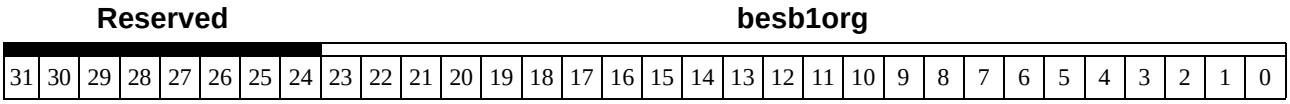
➽ ***Note:*** This address must be aligned on a qword boundary when no horizontal mirror is used (the 3 LSBs must be set to ’000’) and on the last byte of a qword when horizontal mirror is used (the 3 LSBs must be set to ’111’).

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

Address MGABASE1 + 3D08h (MEM)

Attributes WO, STATIC, BYTE, WORD, DWORD

Reset Value unknown



besb1org <23:0> Backend Scaler buffer B field 1 Origin. Top left corner of the buffer B field 1 data used by the backend scaler. The field **besb1org** corresponds to a byte address in memory and holds a 24-bit unsigned value which provides an offset value (the base address) in order to position the first pixel to be read from the buffer B field 1 of the window to scale. In 4:2:0 mode, this register is the luma plane origin.

Note: This address must be aligned on a qword boundary when no horizontal mirror is used (the 3 LSBs must be set to ’000’) and on the last byte of a qword when horizontal mirror is used (the 3 LSBs must be set to ’111’).

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

Address **MGABASE1** + 3D1Ch (MEM)

Attributes WO, STATIC, BYTE, WORD, DWORD

Reset Value unknown

Reserved								besb2corg																							
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

besb2corg <23:0> Backend Scaler buffer B field 2 Chroma plane Origin. Top left corner of the buffer B field 2 chroma plane data used by the backend scaler. The field **besb2corg** corresponds to a byte address in memory and holds a 24-bit unsigned value which provides an offset value (the base address) in order to position the first pixel chroma plane data to be read from the buffer B field 2 of the window to scale. This register must be initialized when in planar 4:2:0 format (2 planes).

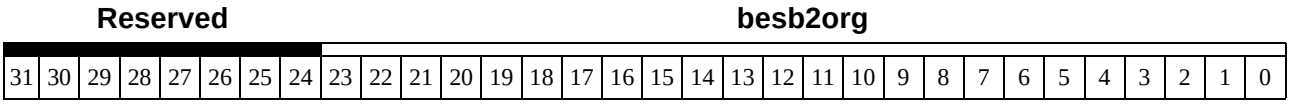
➽ ***Note:*** This address must be aligned on a qword boundary when no horizontal mirror is used (the 3 LSBs must be set to ’000’) and on the last byte of a qword when horizontal mirror is used (the 3 LSBs must be set to ’111’).

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

Address MGABASE1 + 3D0Ch (MEM)

Attributes WO, STATIC, BYTE, WORD, DWORD

Reset Value unknown



besb2org <23:0> Backend Scaler buffer B field 2 Origin. Top left corner of the buffer B field 2 data used by the backend scaler. The field **besb2org** corresponds to a byte address in memory and holds a 24-bit unsigned value which provides an offset value (the base address) in order to position the first pixel to be read from the buffer B field 2 of the window to scale. In 4:2:0 mode, this register is the luma plane origin.

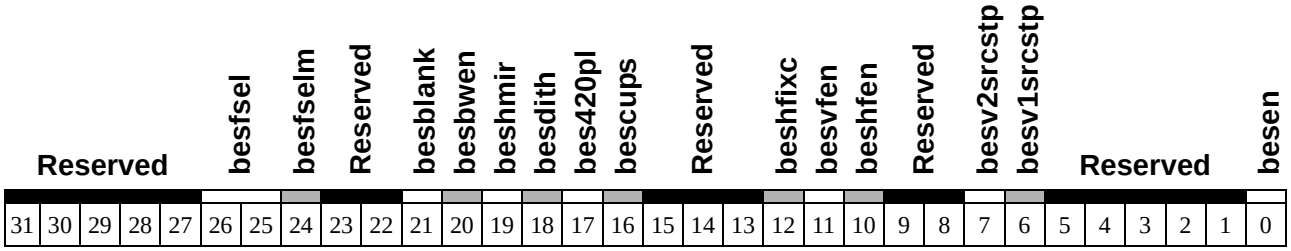
Note: This address must be aligned on a qword boundary when no horizontal mirror is used (the 3 LSBs must be set to ’000’) and on the last byte of a qword when horizontal mirror is used (the 3 LSBs must be set to ’111’).

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

Address **MGABASE1** + 3D20h (MEM)

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

Reset Value ???? ???0 ???? ???? ???? ???? ???? ???0



besen <0>

Backend Scaler Enable.

- 0: The backend scaler is disabled
- 1: The backend scaler is enabled

besv1srcstp <6>

Backend Scaler field 1 vertical source start polarity.

- 0: source start line is even
- 1: source start line is odd

besv2srcstp <7>

Backend Scaler field 2 vertical source start polarity.

- 0: source start line is even
- 1: source start line is odd

beshfen <10>

Backend Scaler Horizontal Filtering Enable.

- 0: The horizontal filtering is disabled
  - •Drop algorithm is used in downscaling and replicated in upscaling.
- 1: The horizontal filtering is enabled (***only*** when horizontal scaling is performed).
  - • Interpolation algorithm is used

besvfen <11>

Backend Scaler Vertical Filtering Enable.

- 0: The vertical filtering is disabled
  - • Drop algorithm is used in downscaling and replicated in upscaling.
- 1: The vertical filtering is enabled (***only*** when vertical scaling is performed)
  - • Interpolation algorithm is used.

beshfixc <12>

Backend Scaler Horizontal Fixed Coefficient. Forces the horizontal weight to 0.5 for the interpolation regardless of the actual calculated weight.

- 0: The horizontal actual calculated weight is used for interpolation.
- 1: The horizontal fixed coefficient is used for interpolation

bescups <16>

Backend Scaler Chroma Upsampling enable. Horizontally interpolates a new chroma value with a fixed coefficient of 0.5 in between each sample pair.

- 0: Chroma upsampling is disabled
- 1: Chroma upsampling is enabled

bes420pl <17>

Backend Scaler 4:2:0 Planar data format.

- 0: The source data is in 4:2:2 format
- 1: The source data is in 4:2:0, 2 plane format

besdith <18>

Backend Scaler Dither enable. Dithering is applied to smooth out some non-linearities.

- 0: Dithering is disabled
- 1: Dithering is enabled

beshmir <19>

Backend Scaler Horizontal Mirror enable. The source data is read linearly with descendant addressing instead of ascendant. Origin registers must point to the latest pixel of the line instead of the first.

Address origins must be aligned on a qword boundary when no horizontal mirror is used (the 3 LSBs must be set to '000') and on the last byte of the qword when horizontal mirror is used (the LSBs must be set to '111').

- 0: The horizontal mirror is disabled
- 1: The horizontal mirror is enabled

besbwen <20>

Backend Scaler Black and White enable. This bit forces both chromas to 128.

- 0: window is in color
- 1: window is in black and white

besblank <21>

Backend Scaler Blanking enable. When blanking is applied, the image in the window becomes black.

- 0: Blanking is disabled
- 1: Blanking is enabled

besfselm <24>

Backend Scaler Field Select Mode.

- 0: Software field select
- 1: Hardware automatic field select triggered by the video in port.

besfsel <26:25>

Backend Scaler Field Select.

- ‘00’: Buffer A field 1 is displayed
- ‘01’: Buffer A field 2 is displayed
- ‘10’: Buffer B field 1 is displayed
- ‘11’: Buffer B field 2 is displayed

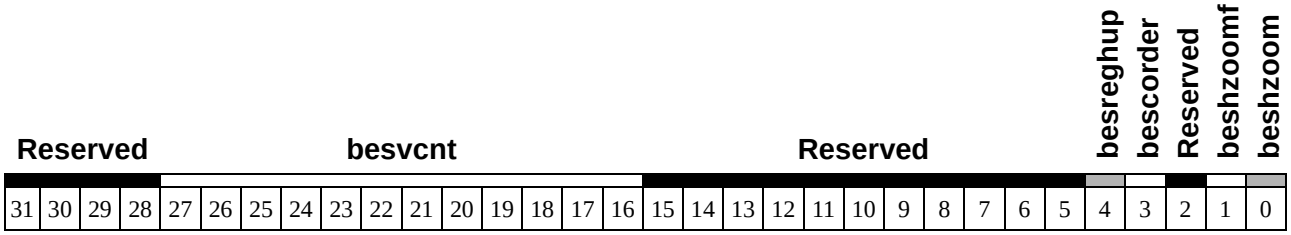
Reserved <31:27> <23:22> <15:13> <9:8><5:1>

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

Address **MGABASE1** + 3DC0h (MEM)

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

Reset Value ???? ???? ???? ???? ???? ???? ???0 0?00



beshzoom <0> Backend Scaler accelerated 2x Horizontal Zoom enable. Must be used when the pixel clock is faster than 135 MHz.

- 0: Accelerated 2x horizontal zoom is disabled
- 1: Accelerated 2x horizontal zoom is enabled

beshzoomf <1> Backend Scaler accelerated 2x Horizontal Zoom filtering enable. Enables the interpolation of the new pixels for the accelerated 2x mode.

- 0: Accelerated 2x horizontal zoom filtering is disabled
- 1: Accelerated 2x horizontal zoom filtering is enabled

bescorder <3> Backend Scaler Chroma samples Order (4:2:0 mode only).

- 0: Cb samples are in bytes 0, 2, 4, 6 of the slice and Cr in 1, 3, 5, 7
- 1: Cb samples are in bytes 1, 3, 5, 7 of the slice and Cr in 0, 2, 4, 6

besreghup <4> Backend Scaler Register Update on Horizontal sync for test. When this bit is set, the backend scaler double buffer registers will take their new values at each horizontal sync.

Note: Used for testing *only*.

- 0: Registers update at the vertical count programmed in **besvcnt**
- 1: Registers update at each horizontal sync

besvcnt <27:16> Backend Scaler Vertical Counter registers update. All backend scaler registers will take effect with their new programmed values when the CRTC vertical counter equals this register.

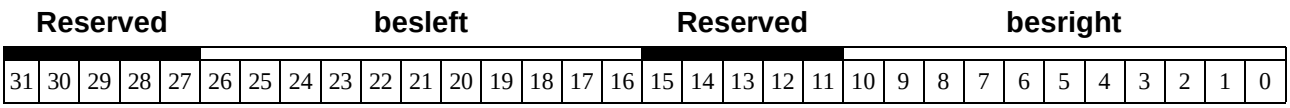
Reserved **<31:28> <15:5> <2>**

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

Address **MGABASE1** + 3D28h (MEM)

Attributes WO, STATIC, BYTE, WORD, DWORD

Reset Value `???? ?000 0000 0000 ???? ?000 0000 0000`



besright <10:0> Backend Scaler Right edge coordinate. This is a pixel coordinate of the destination window in the desktop. Must be higher than **besleft** and not higher than the max. horizontal desktop.

besleft <26:16> Backend Scaler Left edge coordinate. This is a pixel coordinate of the destination window in the desktop. Must be lower than **besright**.

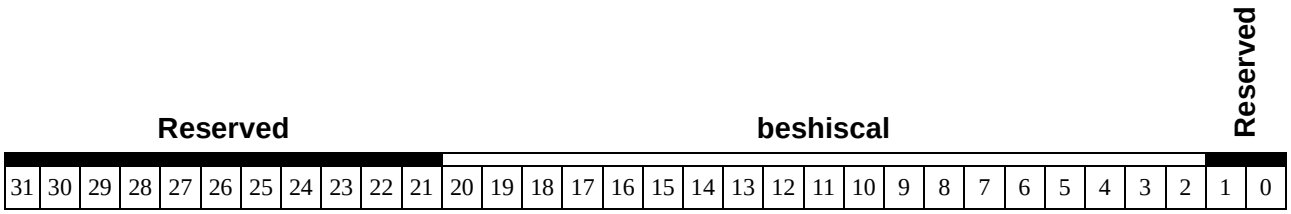
Reserved **<31:27> <15:11>**

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

Address **MGABASE1** + 3D30h (MEM)

Attributes WO, STATIC, BYTE, WORD, DWORD

Reset Value unknown



beshiscal <20:2> Backend Scaler Horizontal Inverse Scaling factor. This is a 5.14 value. (For information on calculating this field, see Chapter 4: Programmer's Specification).

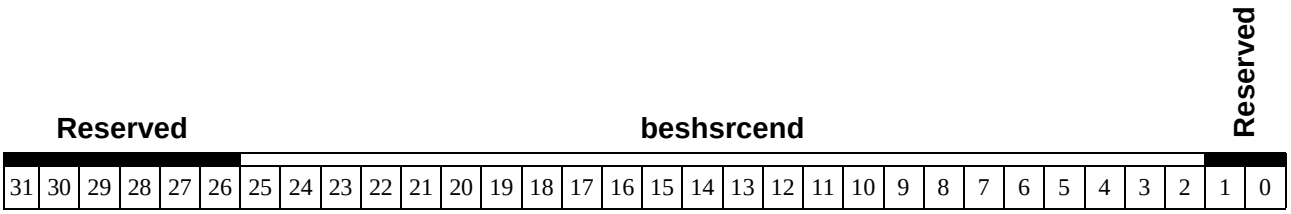
Reserved **<31:21> <1:0>**

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

Address MGABASE1 + 3D3Ch (MEM)

Attributes WO, STATIC, BYTE, WORD, DWORD

Reset Value unknown



beshsrcend <25:2> Backend Scaler Horizontal Ending source position. Ending position in the source of the last pixel that will contribute to the last right destination pixel. This is a 10.14 value. (For information on calculating this field, see Chapter 4: Programmer's Specification).

Reserved **<31:26> <1:0>**

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

Address **MGABASE1** + 3D50h (MEM)

Attributes WO, STATIC, BYTE, WORD, DWORD

Reset Value unknown

Reserved	beshsrclst	Reserved

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

beshsrclst <25:16> Backend Scaler Horizontal Source Last position. Position in the source of the last pixel of the complete image. This field must be programmed with horizontal source width - 1.

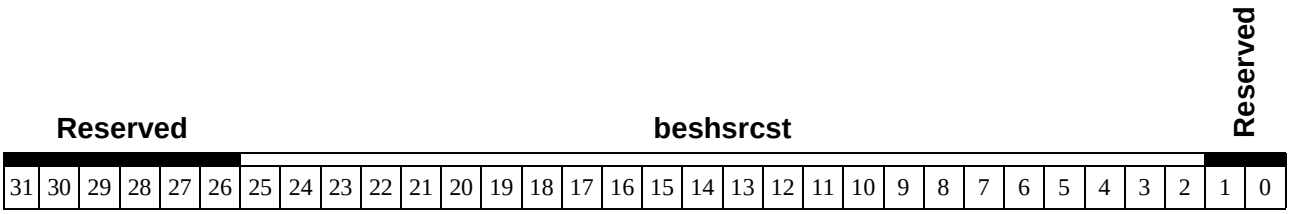
Reserved **<31:26> <15:0>**

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

Address MGABASE1 + 3D38h (MEM)

Attributes WO, STATIC, BYTE, WORD, DWORD

Reset Value unknown



beshsrcst <25:2> Backend Scaler horizontal starting source position. Starting position in the source of the first pixel that will contribute to the left first destination pixel. This is a 10.14 value. Must be ‘0’ when no left cropping is necessary. The cropping can be on the source, because a part of the destination is not visible or is on both.

Reserved **<31:26> <1:0>**

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

Address MGABASE1 + 3D24h (MEM)

Attributes WO, STATIC, BYTE, WORD, DWORD

Reset Value unknown

Reserved																			bespitch												
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

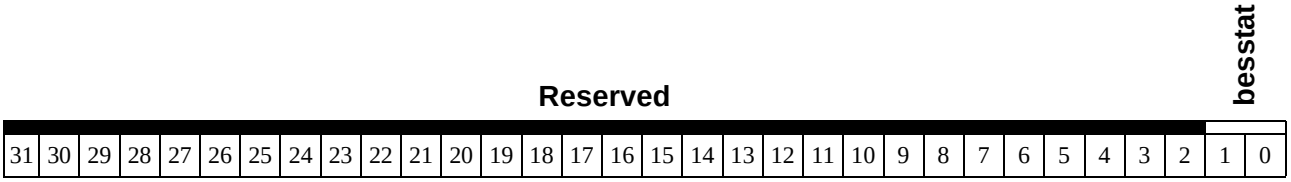
bespitch <11:0> Backend Scaler Pitch. Offset from the beginning of one line to the next from the field currently read for the source data window (in number of pixels). Must be a multiple of 4 in 4:2:2 format and multiple of 8 in 4:2:0 planar format.

Reserved <31:12> Reserved. When writing to this register, the bits in this field ***must*** be set to ‘0’.

Address MGABASE1 + 3DC4h (MEM)

Attributes RO, DYNAMIC, BYTE, WORD, DWORD

Reset Value unknown



besstat <1:0> Backend Scaler Status.

- '00': window is currently displaying buffer A field 1
- '01': window is currently displaying buffer A field 2
- '10': window is currently displaying buffer B field 1
- '11': window is currently displaying buffer B field 2

Reserved <31:2> Reserved. When reading this register, the bits in this field will return *unknown*, '0', or '1'.

Address MGABASE1 + 3D54h (MEM)

Attributes WO, STATIC, BYTE, WORD, DWORD

Reset Value unknown

Reserved																				besv1srclast									
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

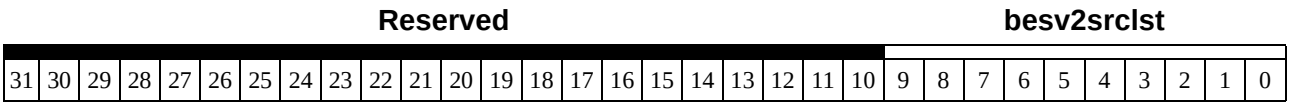
besv1 srclast <9:0> Backend Scaler field 1 vertical source last position. The position of the last line (of the complete image) in the source, in reference to the line pointed to by the window field 1 origin.

Reserved <31:10> Reserved. When writing to this register, the bits in this field ***must*** be set to 0.

Address MGABASE1 + 3D58h (MEM)

Attributes WO, STATIC, BYTE, WORD, DWORD

Reset Value unknown



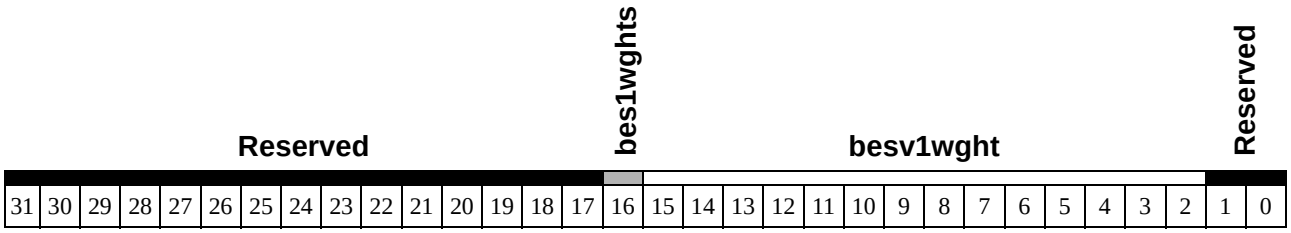
besv2srclst <9:0> Backend Scaler field 2 Vertical Source Last position. The position of the last line (of the complete image) in the source, in reference to the line pointed to by the window field 2 origin.

Reserved <31:10> Reserved. When writing to this register, the bits in this field ***must*** be set to 0.

Address **MGABASE1** + 3D48h (MEM)

Attributes WO, STATIC, BYTE, WORD, DWORD

Reset Value unknown



besv1wght <15:2> Backend Scaler field 1 Vertical Weight starting value. This field contains only the 14 bits of the fractional part. Used for a window starting with vertical subpixel positioning and to adjust the positioning of the frame based on the field 1 in de-interlacing mode.

bes1wghts <16> Backend Scaler field 1 vertical Weight starting sign. Used for a window starting with a vertical subpixel position and to adjust the positioning of the frame on the field 1 in de-interlacing mode.

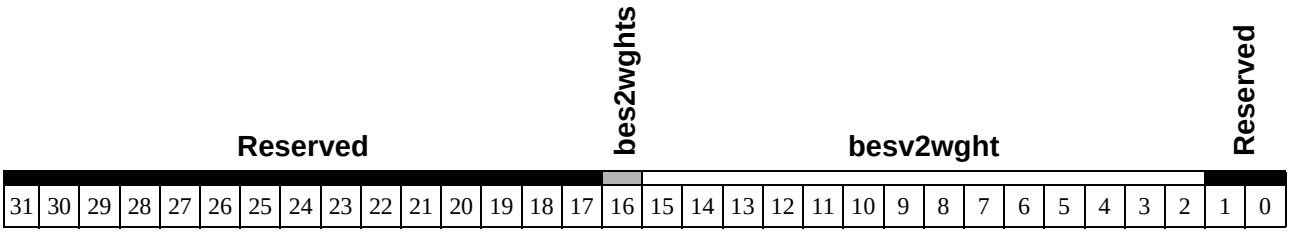
Reserved **<31:17> <1:0>**

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

Address MGABASE1 + 3D4Ch (MEM)

Attributes WO, STATIC, BYTE, WORD, DWORD

Reset Value unknown



besv2wght <15:2> Backend Scaler field 2 Vertical Weight starting value. This field contains only the 14 bits of the fractional part. Used for a window starting with vertical subpixel positioning and to adjust the positioning of the frame based on the field 2 in de-interlacing mode.

bes2wghts <16> Backend Scaler field 2 vertical Weight starting sign. Used for a window starting with vertical subpixel positioning and to adjust the positioning of the frame based on the field 2 in de-interlacing mode.

Reserved **<31:17> <1:0>**

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

Address **MGABASE1** + 3D2Ch (MEM)

Attributes WO, STATIC, BYTE, WORD, DWORD

Reset Value unknown

Reserved	bestop	Reserved	besbot

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

besbot <10:0> Backend Scaler Bottom edge coordinate. This is a pixel coordinate of the destination window in the desktop. Must be higher than **bestop** and not higher than the max. vertical desktop.

bestop <26:16> Backend Scaler Top edge coordinate. This is a pixel coordinate of the destination window in the desktop. Must be lower than **besbot**.

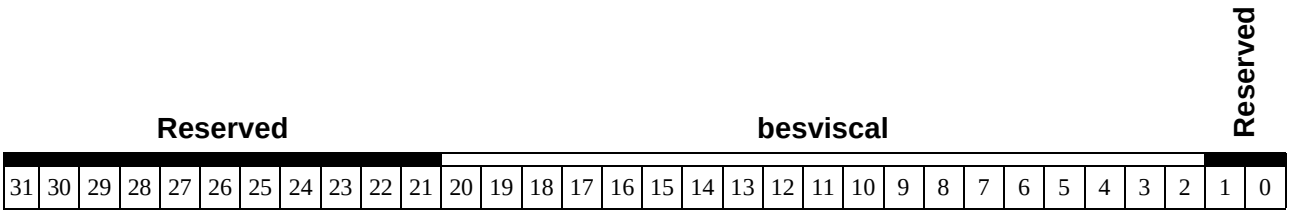
Reserved **<31:27> <15:11>**

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

Address MGABASE1 + 3D34h (MEM)

Attributes WO, STATIC, BYTE, WORD, DWORD

Reset Value unknown



besviscal <20:2> Backend Scaler Vertical Inverse Scaling. This is a 5.14 value. (For information on calculating this field, see Chapter 4: Programmer's Specification).

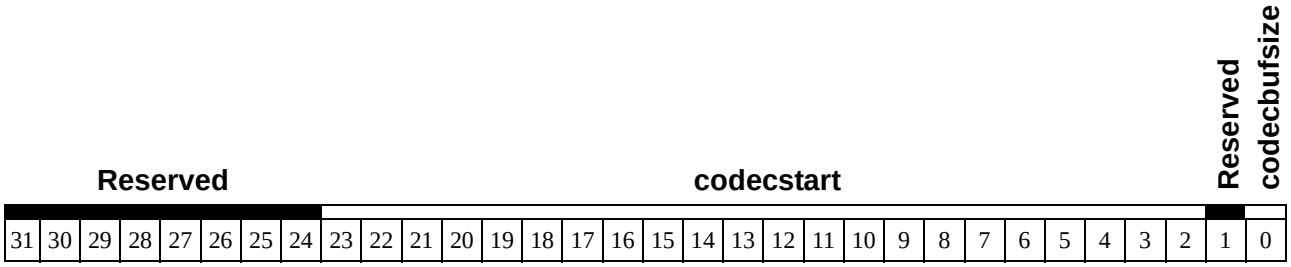
Reserved <31:21> <1:0>

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

Address MGABASE1 + 3E44h (MEM)

Attributes WO, BYTE/WORD/DWORD, STATIC

Reset Value unknown



codecbufsize <0> Codec Buffer Size

- 0: 128K byte
- 1: 256K byte

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

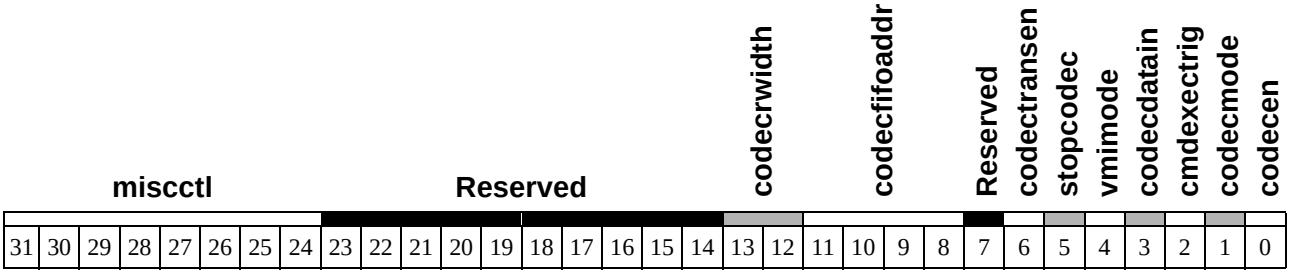
Reserved <1> <31:24>

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

Address MGABASE1 + 3E40h (MEM)

Attributes WO, BYTE/WORD/DWORD, STATIC

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



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

- 0: disable (default)
- 1: enable

codecmode <1> Codec Mode.

- 0: VMI mode
- 1: I33 mode

cmdexectrig <2> Command Execution Trigger

- 0: do ***not*** execute commands in memory
- 1: execute register commands in memory

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

codecdatain <3> Codec Data In.

- 0: decompression (off Screen Frame buffer to CODEC)
- 1: compression (CODEC to off screen Frame buffer)

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

- 0: Mode A
- 1: Mode B

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

- 0: do ***not*** stop after current field
- 1: stop after current field

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

- 0: disable transfer
- 1: enable transfer

codecfifo addr <11:8>

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

codecrwidth <13:12>

Pulse recovery width. This bit determines the number of **gclkbuf** clock cycles between the rising edge of a strobe and the falling edge of the next strobe of consecutive bytes when performing compression or decompression.

I33 mode and VMI mode B:

- '00': 4 gclkbuf cycles between read/write strobes
- '01': 5 gclkbuf cycles between read/write strobes
- '10': 6 gclkbuf cycles between read/write strobes

VMI mode A:

- '00': 5 gclkbuf cycles between data strobes
- '01': 6 gclkbuf cycles between data strobes
- '10': 7 gclkbuf cycles between data strobes

miscctl <31:24>

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

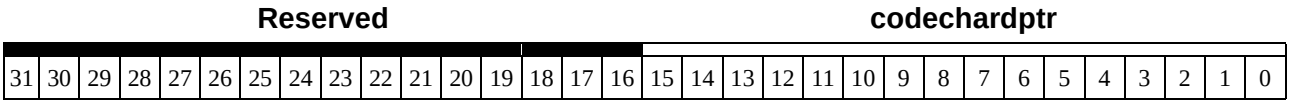
Reserved **<7> <23:14>**

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

Address MGABASE1 + 3E4Ch (MEM)

Attributes RO, BYTE/WORD/DWORD, DYNAMIC

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



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

Additional Reset condition: **codecen**

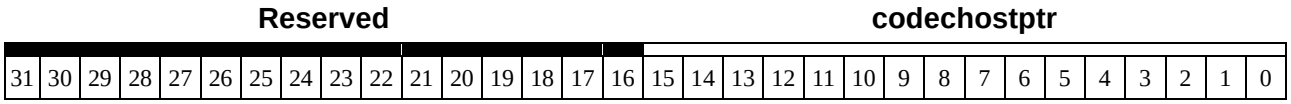
- When ***compressing*** video data, this register will point to the offset of the next dword to be written to the codec interface's circular buffer.
- When ***decompressing*** video data, this register will point to the offset of the next dword to be read from the codec interface's circular buffer.

Reserved <31:16> Reserved. When writing to this register, the bits in this field ***must*** be set to 0.

Address MGABASE1 + 3E48h (MEM)

Attributes WO, BYTE/WORD/DWORD, STATIC

Reset Value unknown



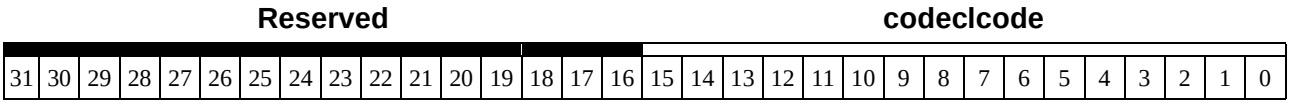
codechostptr <15:0> CODEC Host Pointer. An interrupt is generated (if enabled) when the dword offset pointed to by this register is accessed by the Codec Interface in its circular buffer.

Reserved <31:16> Reserved. When writing to this register, the bits in this field ***must*** be set to 0.

Address MGABASE1 + 3E50h (MEM)

Attributes RO, BYTE/WORD/DWORD, DYNAMIC

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



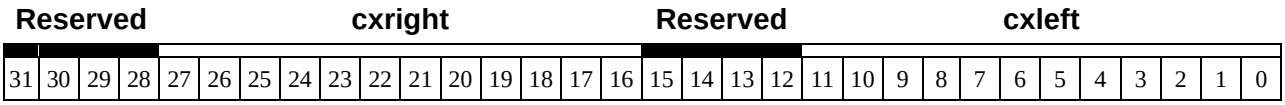
codeclcode <15:0> Used only in compression. It will point to the DWORD offset in the CODEC circular buffer following the last DWORD of the field. This register will be updated after the CODEC asserts its EOI (LCODE) pin.

Reserved <31:16> Reserved. When writing to this register, the bits in this field ***must*** be set to '0'.

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 <11:0> Clipper x left boundary. See the **CXLEFT** register on page 3-75.

cxright <27:16> Clipper x right boundary. See the **CXRIGHT** register on page 3-76.

Reserved **<31:28> <15:12>**

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 <11:0> Clipper x left boundary. The **cxleft** field contains an unsigned 12-bit value which is interpreted as a positive pixel address and compared with the current **xdst** (see YDST on page 3-215). The value of **xdst** must be greater than or equal to **cxleft** to be inside the drawing window.

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

Note: Clipping can be disabled by the **clipdis** bit in DWGCTL without changing **cxleft**.

Reserved <31:12> 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 <11:0> Clipper x right boundary. The **cxright** field contains an unsigned 12-bit value which is interpreted as a positive pixel address and compared with the current **xdst** (see **YDST** on page 3-215). The value of **xdst** must be less than or equal to **cxright** to be inside the drawing window.

Note: Clipping can be disabled by the **clipdis** bit in **DWGCTL** without changing **cxright**.

Reserved <31:12> 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 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 NMap 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** 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 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 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

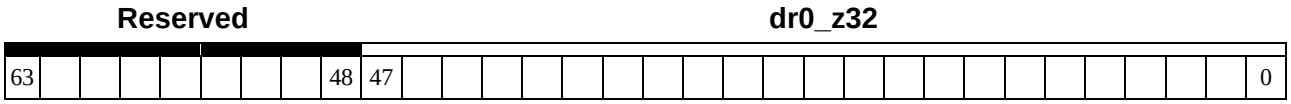
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

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

Note: The field **dr0_z32** is part of two registers:

- dr0_z32[47:32] = DR0_Z32MSB[15:0]
- dr0_z32[31:0] = DR0_Z32LSB[31:0]

Reserved <63:48>

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

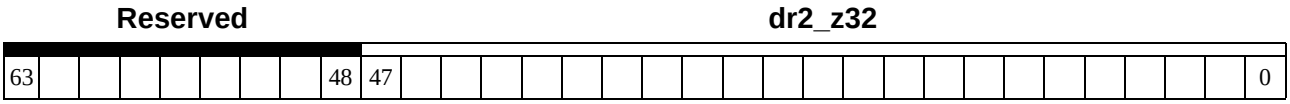
Note: The Reserved field is part of one register:

- dr0_z32[63:48] = DR0_Z32MSB[31:16]

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

Attributes WO, FIFO, STATIC, DWORD

Reset Value unknown



dr2_z32 <47:0> Data ALU 2

- For TRAP or TEXTURE_TRAP with z, the **DR2** register holds the z increment value along the x-axis. In this case, **DR2** is a signed 33.15 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 33.15 value in two's complement notation.

Note: The field **dr2_z32** is part of two registers:
- dr2_z32[63:32] = DR2_Z32MSB[31:0]
- dr2_z32[31:0] = DR2_Z32LSB[31:0]

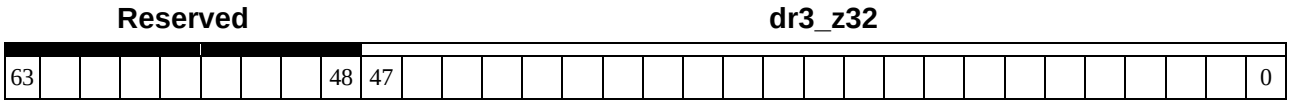
Reserved <63:48> Reserved. When writing to this register, bits 63 to 48 are completely ignored. Reading will return '0's.

Note: The Reserved field is part of one register:
- dr2_z32[63:48] = DR2_Z32MSB[31:16]

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

Attributes WO, FIFO, STATIC, DWORD

Reset Value unknown



dr3_z32 <47:0> Data ALU 3.

- For TRAP or TEXTURE_TRAP with z, **DR3** register holds the z increment value along the y-axis. In this case, **DR3** is a signed 33.15 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 33.15 value in two's complement notation.

Note: The field **dr3_z32** is part of two registers:
- dr3_z32[63:32] = DR3_Z32MSB[31:0]
- dr3_z32[31:0] = DR3_Z32LSB[31:0]

Reserved <63:48> Reserved. When writing to this register, bits 63 to 48 are completely ignored. Reading will return '0's.

Note: The Reserved field is part of one register:
- dr3_z32[63:48] = DR3_Z32MSB[31:16]

Address MGABASE1 + 1CC0h (MEM)
Attributes WO, FIFO, DYNAMIC, DWORD
Reset Value unknown

dr0

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

dr0 <31:0> Data ALU register 0.

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

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

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

dr2

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

dr2 <31:0>

Data ALU register 2.

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

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

Address MGABASE1 + 1CCCh (MEM)
Attributes WO, FIFO, STATIC, DWORD
Reset Value unknown

dr3

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

dr3 <31:0> Data ALU register 3.

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

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

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

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value unknown

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.

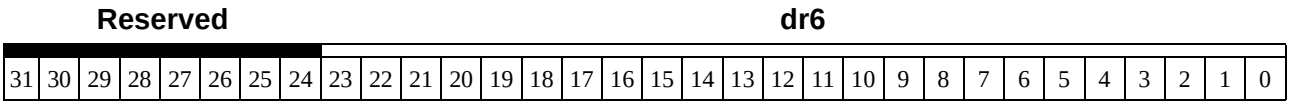
- For TRAP with z, the **DR4** register is used to scan the left edge of the trapezoid for the red color (Gouraud shading). This register must be initialized with its starting red color value.
- For LINE with z, the **DR4** register holds the current red color value for the currently drawn pixel. This register must be initialized with the starting red color.
- For TEXTURE_TRAP with texture modulation (**tmodulate** = '1', see **TEXCTL**), the **DR4** register is used to scan the left edge of the trapezoid for the red modulation factor. This register must be initialized with its starting red modulation factor value.
- For TEXTURE_TRAP using the decal feature (**tmodulate** = '0'), the **DR4** register is used to scan the left edge of the trapezoid for the red (Gouraud shaded surface) color. This register must be initialized with its starting red color value.

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

Address MGABASE1 + 1CD8h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value unknown



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

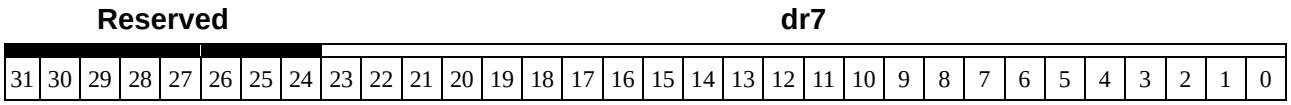
- For TRAP with z, the DR6 register holds the red increment value along the x-axis.
- For LINE with z, the DR6 register holds the red increment value along the major axis.
- For TEXTURE_TRAP with modulation or decal, the DR6 register holds the red increment value along the x-axis.

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

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

Attributes WO, FIFO, STATIC, DWORD

Reset Value unknown



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 LINE with z, the **DR7** register holds the red increment value along the diagonal axis.
- For TEXTURE_TRAP with modulation or decal, the **DR7** register holds the red increment value along the y-axis.

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

Address MGABASE1 + 1CE0h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value unknown

Reserved	dr8

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

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

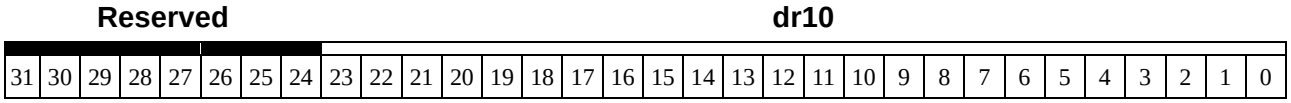
- For TRAP with z, the DR8 register is used to scan the left edge of the trapezoid for the green color (Gouraud shading). This register must be initialized with its starting green color value.
- For LINE with z, the DR8 register holds the current green color value for the currently drawn pixel. This register must be initialized with the starting green color.
- For TEXTURE_TRAP with texture modulation (**tmodulate** = '1', see TEXCTL), the DR8 register is used to scan the left edge of the trapezoid for the green modulation factor. This register must be initialized with its starting green modulation factor value.
- For TEXTURE_TRAP using the decal feature (**tmodulate** = '0'), the DR8 register is used to scan the left edge of the trapezoid for the green (Gouraud shaded surface) color. This register must be initialized with its starting green color value.

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

Address **MGABASE1** + 1CE8h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value unknown



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

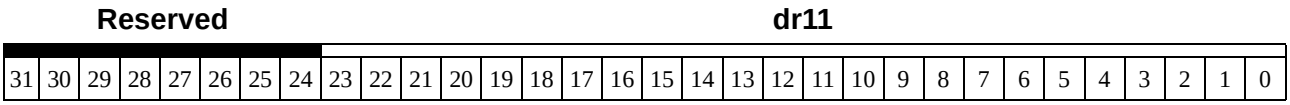
- For TRAP with z, the **DR10** register holds the green increment value along the x-axis.
- For LINE with z, the **DR10** register holds the green increment value along the major axis.
- For TEXTURE_TRAP with modulation or decal, the **DR10** register holds the green increment value along the x-axis.

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

Address MGABASE1 + 1CECh (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value unknown



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

- For TRAP with z, the DR11 register holds the green increment value along the y-axis.
- For LINE with z, the DR11 register holds the green increment value along the diagonal axis.
- For TEXTURE_TRAP with modulation or decal, the DR11 register holds the green increment value along the y-axis.

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

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

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value unknown

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

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

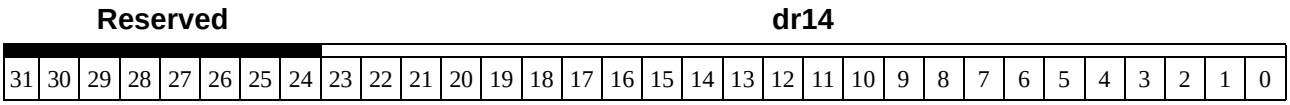
- For TRAP with z, the **DR12** register is used to scan the left edge of the trapezoid for the blue color (Gouraud shading). This register must be initialized with its starting blue color value.
- For LINE with z, the **DR12** register holds the blue color value for the currently drawn pixel. This register must be initialized with the starting blue color.
- For TEXTURE_TRAP with texture modulation (**tmodulate** = '1', see **TEXCTL**), the **DR12** register is used to scan the left edge of the trapezoid for the blue modulation factor. This register must be initialized with its starting blue modulation factor value.
- For TEXTURE_TRAP using the decal feature (**tmodulate** = '0'), the **DR12** register is used to scan the left edge of the trapezoid for the blue (Gouraud shaded surface) color. This register must be initialized with its starting blue color value.

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

Address MGABASE1 + 1CF8h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value unknown



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

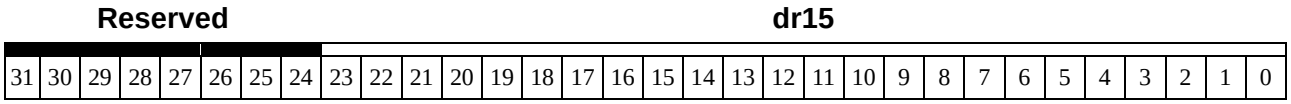
- For TRAP with z, the DR14 register holds the blue increment value along the x-axis.
- For LINE with z, the DR14 register holds the blue increment value along the major axis.
- For TEXTURE_TRAP with modulation or decal, the DR14 register holds the blue increment value along the x-axis.

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

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

Attributes WO, FIFO, STATIC, DWORD

Reset Value unknown



dr15 <23:0> Data ALU register 15. This field holds a signed 9.15 value in two's complement notation.

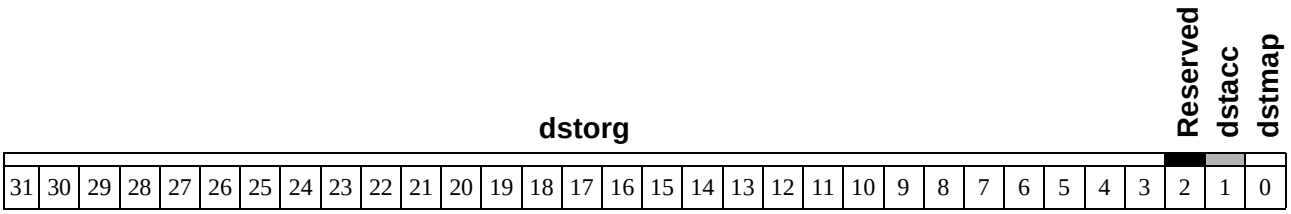
- For TRAP with z, the **DR15** register holds the blue increment value along the y-axis.
- For LINE with z, the **DR15** register holds the blue increment value along the diagonal axis.
- For TEXTURE_TRAP with modulation or decal, the **DR15** register holds the blue increment value along the y-axis.

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

Address **MGABASE1** + 2CB8h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



dstmap <0> Destination Map. A memory space indicator, this field indicates the map location.

- 0: the destination is in the frame buffer memory.
- 1: the destination is in the system memory.

dstacc <1> Destination Access type. This field specifies the mode used to access the map.

- 0: PCI access.
- 1: AGP access.

Note: This field is ***not*** considered if the destination resides in the frame buffer source.

dstorg <31:3> Destination Origin. This field provides an offset value for the position of the first pixel of a destination surface. The **dstorg** field corresponds to a qword address in memory.

The **DSTORG** register must be loaded with a multiple of 3, 4, 8 or 24 qwords according to the following table:

pwidth	atype	DSTORG *(multiples of qwords)*
PW8	!BLK	4(1)
PW16	!BLK	4(1)
PW24	!BLK	3(2)
PW32	!BLK	4(1)
PW8	BLK	8
PW16	BLK	8
PW24	BLK	24(2)
PW32	BLK	8

(1) The **DSTORG** register must be loaded with a multiple of 4 qwords due to a limitation when alpha belnding is enabled.

(2) This restriction is due to the limitation of the CRTC in PW24. If the destination is offscreen then the real limitations are 1 in a !BLK and 8 in BLK mode.

Reserved <2> 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>

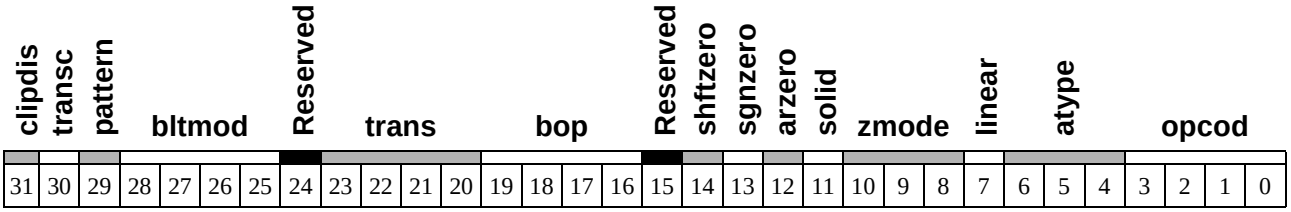
These 16 registers are a lookup table that can be used in conjunction with the **DMAMAP** registers. Writing to these locations address the register that is programmed in the Nth byte of the **DMAMAP**. This indirect write register provides a means to access non-sequential drawing registers sequentially.

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
	Texture mapping	‘0110’	TEXTURE_TRAP
Blit	RAM → RAM	‘1000’	BITBLT
	HOST → RAM	‘1001’	ILOAD
	Reserved	‘0101’ ‘0111’ ‘1010’ ‘1011’ ‘1100’ ‘1101’ ‘1110’ ‘1111’	

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)(1)
'001'	RSTR	Read-modify-write (raster)(1)
'010'		Reserved
'011'	ZI	Depth mode with Gouraud(2)
'100'	BLK	Block write mode (3)
'101'		Reserved
'110'		Reserved
'111'	I	Gouraud (with depth compare)(2)(4)

(1) The Read-Modify-Write sequence depends on the value of the bop field even if **atype** equals RPL, RSTR, ZI, or I. The Read will be performed according to the table in the **bop** field (see page 3-102).

(2) The raster operation also supports ZI and I operations (see page 3-102).

(3) 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.

(4) 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 3-153).

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

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

arzero <12>

AR register at zero. The **arzero** field provides an alternate way to set all **AR** registers (see descriptions starting on page 3-36).

- 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**)

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:
brkleft (see **SGN** on page 3-139)
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 3-129) is added to **ydst** (see **YDST** on page 3-215)

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**

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

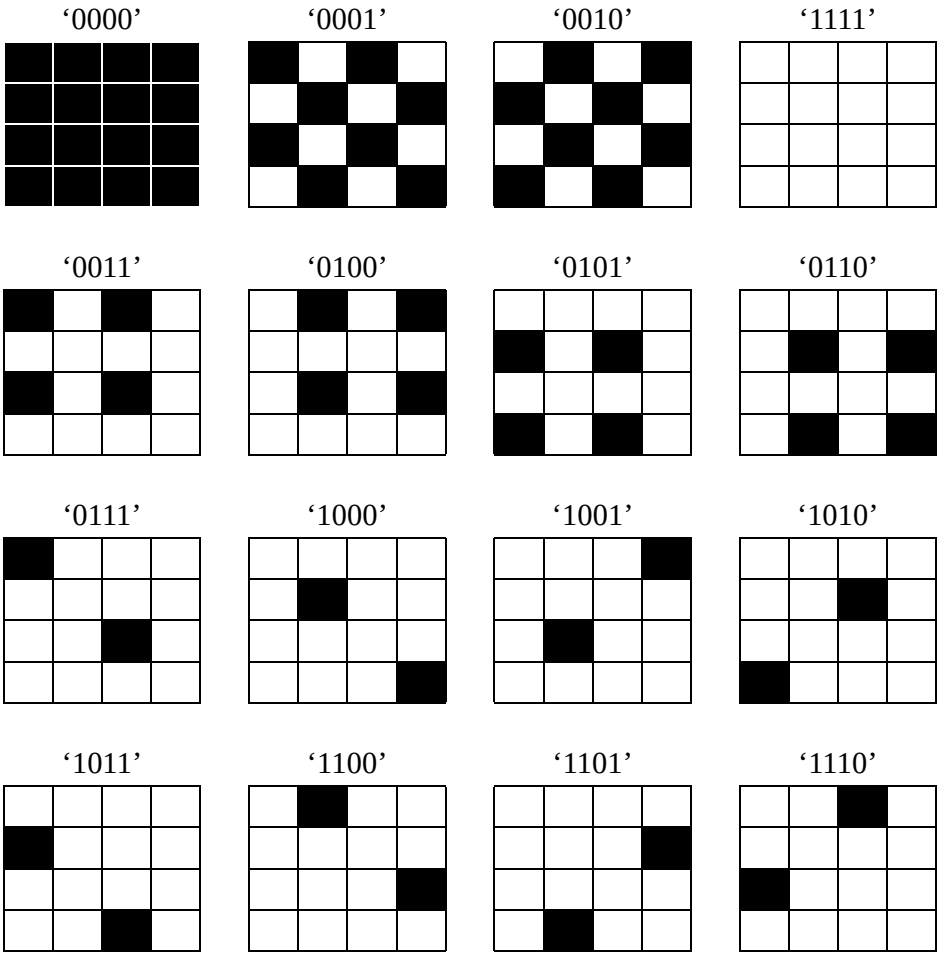
bop <19:16>

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. A raster operation is performed when atype = ZI, I, RPL or RSTR.

bop	*Function*	*Destination read*
‘0000’	0	No
‘0001’	~(D | S)	Yes
‘0010’	D & ~S	Yes
‘0011’	~S	No
‘0100’	(~D) & S	Yes
‘0101’	~D	Yes
‘0110’	D ^ S	Yes
‘0111’	~(D & S)	Yes
‘1000’	D & S	Yes
‘1001’	~(D ^ S)	Yes
‘1010’	D	Yes
‘1011’	D | ~S	Yes
‘1100’	S	No
‘1101’	(~D)| S	Yes
‘1110’	D | S	Yes
‘1111’	1	No

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.
‘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’		”
‘1110’		Reserved

- For line drawing with line style, this field must have the value BFCOL in order to handle the line style properly.

Refer to the subsections contained in ‘Drawing in Power Graphic Mode’ on page 4-29 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.

clipdis <31>

Clipping Disable. This bit has the following effect on the value of CXLEFT, CXRIGHT, CYBOT, and CYTOP:

	‘0’	**‘1’**
CXLEFT	last value programmed in CXLEFT	0h (minimum value)
CXRIGHT	last value programmed in CXRIGHT	FFFh (maximum value)
CYBOT	last value programmed in CYBOT	FFFFFFh (maximum value)
CYTOP	last value programmed in CYTOP	0h (minimum value)

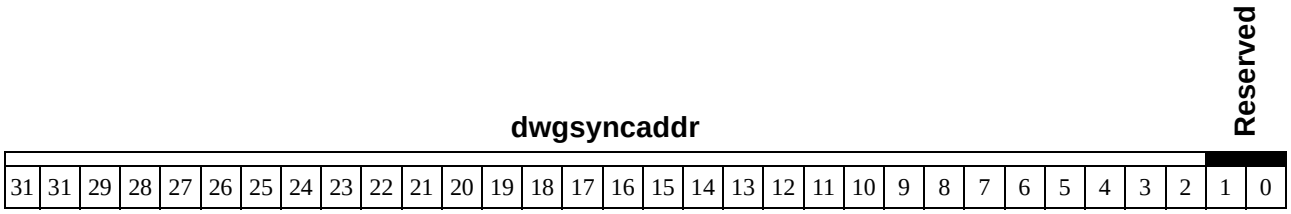
Reserved <15> <24>

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

Address **MGABASE1** + 2C4C

Attributes R/W, FIFO, DYNAMIC, DWORD

Reset Value unknown



dwgsyncaddr <31:2> This register serves as a synchronisation pointer. The value of **dwgsyncaddr** is updated with the value programmed in **dwgsyncaddr** ***only*** when the drawing engine has completed the primitive sent before **DWGSYNC** was programmed.

When the **primptren1** bit is set, this register is written by a PCI access in the location pointed to by **PRIMPTR**. The write is triggered when the drawing engine updates **dwgsyncaddr**.

Reserved <1:0> Reserved. When writing to this register, the bits in this field ***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 bits/pixel, only **forcol**<7:0> is used.
- In 16 bits/pixel, only **forcol**<15:0> is used.
- 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 4-21 for the definition of the slice in each mode.

Part of the **forcol** register is also used for Gouraud shading to generate the alpha bits. In 32 bpp (bits/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>.

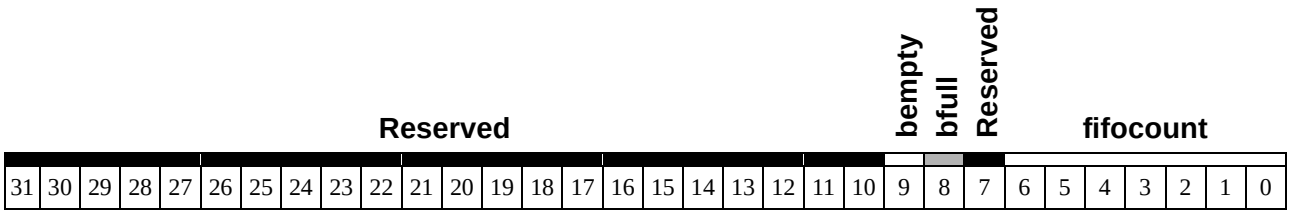
bltckey <31:0> Blit color key. This field specifies the value of the color that is defined as the 'transparent' color. Planes that are ***not*** used must be set to '0'. Refer to the description of the **transc** field of **DWGCTL** for the transparency equation

- In 8 bits/pixel, only **bltckey**<7:0> is used.
- In 16 bits/pixel, only **bltckey**<15:0> is used.

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

Attributes RO, DYNAMIC, BYTE/WORD/DWORD

Reset Value 0000 0000 0000 0000 0000 0010 0100 0000b



fifocount <6:0> Indicates the number of free locations in the Bus FIFO. On soft or hard reset, the contents of the Bus FIFO are flushed and the FIFO count is set to 64.

bfull <8> Bus FIFO full flag. When set to '1', indicates that the Bus FIFO is full.

bempty <9> Bus FIFO empty flag. When set to '1', indicates that the Bus FIFO is empty. This bit is identical to **fifocount**<6>.

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

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

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

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

Address MGABASE1 + 1CF4h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value unknown

fogcol

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

fogcol <23:0> Fog Color. When fogging is enabled, the fog field represents the color that is blended, using the fog blending factor, with the current rasterized fragment's color.

Note: FOGCOL is in true color (RGB: 888) format.

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

Address MGABASE1 + 1CC4h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value unknown

Reserved	fogstart

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

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

For 3D primitives, the FOGSTART register is used to scan the left edge of the trapezoid for the fog blending factor (when fogging is enabled).

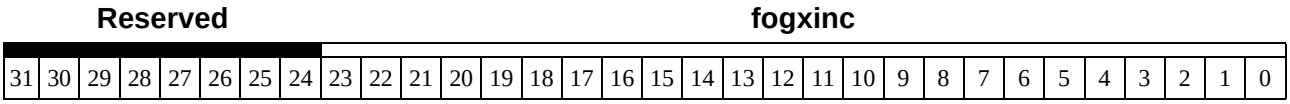
Note: This register must be initialized with its starting fog factor value.

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

Address MGABASE1 + 1CD4h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value unknown



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

For 3D primitives, the FOGXINC register holds the fog blending factor increment along the x (or major) axis.

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

Address MGABASE1 + 1CE4h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value unknown

Reserved	fogyinc

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

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

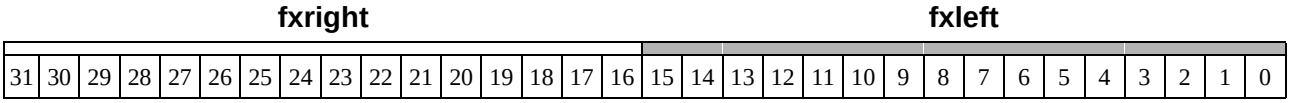
For 3D primitives, the FOGYINC register holds the fog blending factor increment along the y (or diagonal) axis.

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

Address MGABASE1 + 1C84h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value unknown



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 3-115.

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

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value unknown

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

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

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

Reserved <31:16> Reserved. When writing to this register, the bits in this field ***must*** be set to '0'.

Address MGABASE1 + 1CACh (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value unknown

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

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

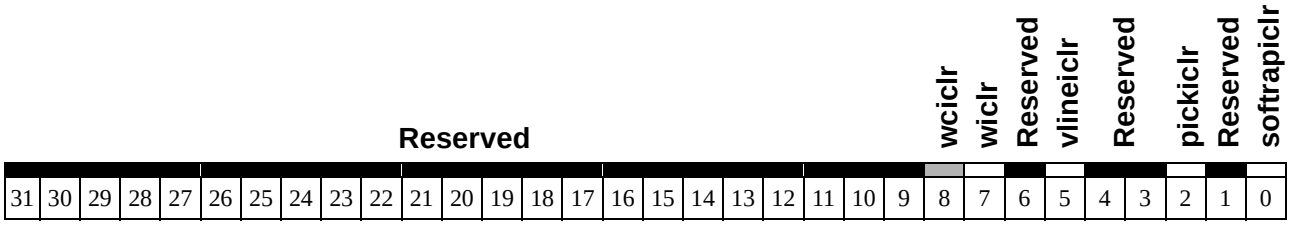
- The **fxright** field is ***not*** used for line drawing.
- During filled trapezoid drawing, **fxright** is updated during the right edge scan.
- During a BLIT operation, **fxright** is static, and specifies the right pixel boundary of the area being written to.

Reserved <31:16> Reserved. When writing to this register, the bits in this field ***must*** be set to '0'.

Address **MGABASE1** + 1E18h (MEM)

Attributes WO, STATIC, BYTE/WORD/DWORD

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



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

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

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

wiclr <7> WARP Interrupt Clear. When a ‘1’ is written to this bit, the WARP interrupt pending flag is cleared.

wciclr <8> WARP Cache interrupt Clear. When a ‘1’ is written to this bit, the WARP cache interrupt pending flag is cleared.

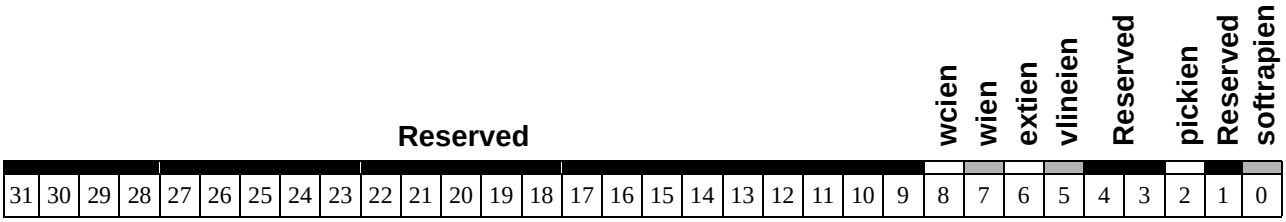
Reserved **<1> <4:3> <6> <31:9>**

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

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

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

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



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

pickien <2> Picking Interrupt Enable. When set to '1', enables interrupts if a picking interrupt occurs.

vlineien <5> Vertical Line Interrupt Enable. When set to '1', an interrupt will be generated when the vertical line counter equals the vertical line interrupt count.

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

wien <7> WARP Interrupt Enable. When set to '1', a WARP interrupt will contribute to the generation of a PCI interrupt on the PINTA/ line.

wcien <8> WARP Cache interrupt enable. When set to '1' a WARP CACHE miss will contribute to the generation of a PCI interrupt on the PINTA/ line.

Reserved **<1> <4:3> <31:9>**

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

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

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b

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

length <15:0> Length. The length field is a 16-bit unsigned value.

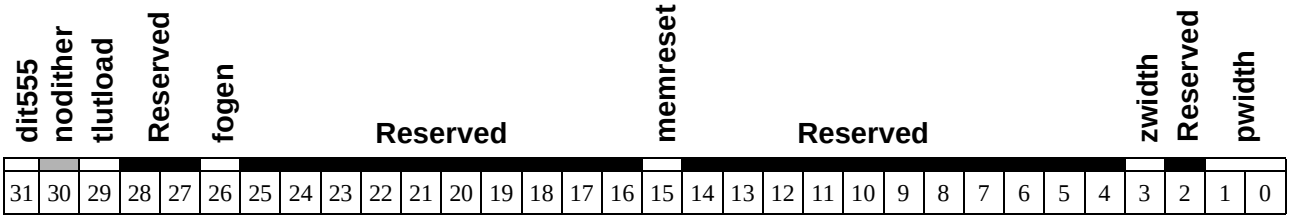
- The **length** field does ***not*** require initialization for auto-init vectors.
- For a vector draw, **length** is programmed with the number of pixels to be drawn.
- For blits and trapezoid fills, **length** is programmed with the number of lines to be filled or blitted.
- To load the texture color palette, **length** is programmed with the number of locations in the LUT to be filled.

Reserved <31:16> 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 4-24 for instructions on when to use this field. The **memreset** field must always be set to '0' except under specific conditions which occur during the reset sequence.

fogen <26> Fogging Enable. Fogging can be performed on any 3D operation.

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

nodither <30> Enable/disable dithering.

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

dit555 <31>

Dither 5:5:5 mode. This field should normally be set to ‘0’, except for 16 bit/pixel configurations, when it affects dithering and shading.

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

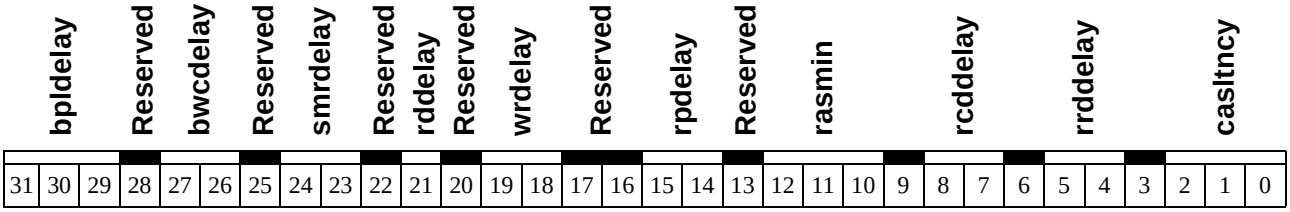
Reserved **<2> <14:4> <25:16> <28:27>**

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

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

Attributes WO, FIFO, STATIC, DWORD

Reset Value 1000 0100 1010 0100 1001 1001 0010 0001b



casltncy <2:0> CAS Latency. This field *must* be programmed prior to executing the memory power-up sequence. This field *must* be loaded before initiating a memory test.

casltncy	*CAS Latency (mclk)*
'000'	2 cycles
'001'	3 cycles
'010'	4 cycles
'011'	5 cycles
'1XX'	Reserved

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

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

rcddelay <8:7> Minimum RAS to CAS Delay. This field *must* be loaded before attempting to read or write to the frame buffer.

rcddelay	*RAS to CAS Delay (mclk)*
'00'	2 cycles
'01'	3 cycles
'10'	4 cycles
'11'	Reserved

rasmin <12:10>

RAS Minimum active time. This field *must* be loaded before initiating a memory reset.

rasmin	*RAS Minimum (mclk)*
'000'	3 cycles
'001'	4 cycles
'010'	5 cycles
'011'	6 cycles
'100'	7 cycles
'101'	8 cycles
'110'	9 cycles
'111'	10 cycles

rpdelay <15:14>

Minimum RAS precharge Delay. This field *must* be loaded before initiating a memory reset.

rpdelay	*Precharge to Activate Delay (mclk)*
'00'	2 cycles
'01'	3 cycles
'10'	4 cycles
'11'	5 cycles

wrdelay <19:18>

Minimum Write Recovery Delay. This field *must* be loaded before attempting to read or write to the frame buffer.

wrdelay	*Write to Precharge Delay (mclk)*
'00'	1 cycle
'01'	2 cycles
'1X'	Reserved

rddelay <21>

Minimum Read to Precharge Delay. This field *must* be loaded before attempting to read or write to the frame buffer.

rddelay	*Read to Precharge Delay*
'0'	*n* cycles
'1'	*n* + (CL - 2) cycles(1)

(1) Where *n* = the amount of data to be read and CL = CAS latency (2, 3, 4, 5).

smrdelay <24:23>

Minimum Special Mode Register Delay. This field *must* be loaded before attempting to read or write to the frame buffer.

smrdelay	*SMR to Command Delay (mclk)*
'00'	1 cycle
'01'	2 cycles
'1X'	Reserved

bwcdelay <27:26>

Minimum Block Write Cycle Delay. This field *must* be loaded before attempting to read or write to the frame buffer.

bwcdelay	*Block Write Cycle Delay (mclk)*
'00'	1 cycle
'01'	2 cycles
'1X'	Reserved

bpldelay <31:29>

Minimum Block write to Precharge Delay. This field *must* be loaded before attempting to read or write to the frame buffer.

bpldelay	*Block Write to Precharge Delay (mclk)*
'000'	1 cycle
'001'	2 cycles
'010'	3 cycles
'011'	4 cycles
'100'	5 cycles
'101'	6 cycles
'11X'	Reserved

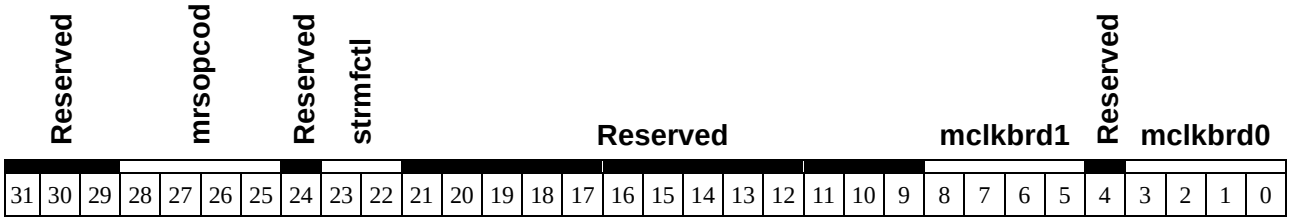
Reserved **<3> <6> <9> <13> <17:16> <20> <22> <25> <28>**

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

Address **MGABASE1** + 1E44h (MEM)

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

Reset Value 0000 0000 0000 0000 0000 0001 0000 1000b



mclkbrd0 <3:0> Memory Clock Base Read Delay 0. This field *must* be loaded after initiating a memory reset and before attempting any other access to the frame buffer.

This field is used to adjust the delay on the clock/strobe used to register/latch the read-back data, MDQ(31:0), from the two banks on the base board. The pointer, **mclkbrd0**, determines where the delay-line for the read-back clock will be tapped. Each increment of **mclkbrd0** adds approximately 0.2ns to the delay on the read-back clock.

- '0000': minimum delay added
- '1111': maximum delay added

Note: This field should be *INVISIBLE* to the user. Software will set the field using read/write trails (which vary the field).

mclkbrd1 <8:5> Memory Clock Base Read Delay 1. This field must be loaded after initiating a memory reset and before attempting another access to the frame buffer.

This field is used to adjust the delay on the clock/strobe used to register/latch the read-back data, MDQ(63:31), from the two banks on the base board. The pointer, **mclkbrd1**, determines where the delay-line for the read-back clock will be tapped. Each increment of **mclkbrd1** adds approximately 0.2ns to the delay on the read-back clock.

- '0000': minimum delay added
- '1111': maximum delay added

Note: This field should be *INVISIBLE* to the user. Software will set the field using read/write trails (which vary the field)

strmfctl <23:22> Streamer Flow Control. This field is used to ensure that CRTC latencies are respected.

strmfctl	*description*
'00'	Do not block access to streamer pipe.
'01'	A maximum of TWO non-CRTC commands can be in the streamer pipe at any time.
'10'	A maximum of ONE non-CRTC commands can be in the streamer pipe at any time.
'11'	Reserved

mrsopcod <28:25>

Mode Register Set command OPCODe. This field *must* be loaded before initializing a memory reset or attempting to read or write to the frame buffer.

This field is used to fill in the 4 MSB (MA(10:7)) of the Mode register set command's opcode.

This field ***must*** be programmed with '0000' during normal operation.

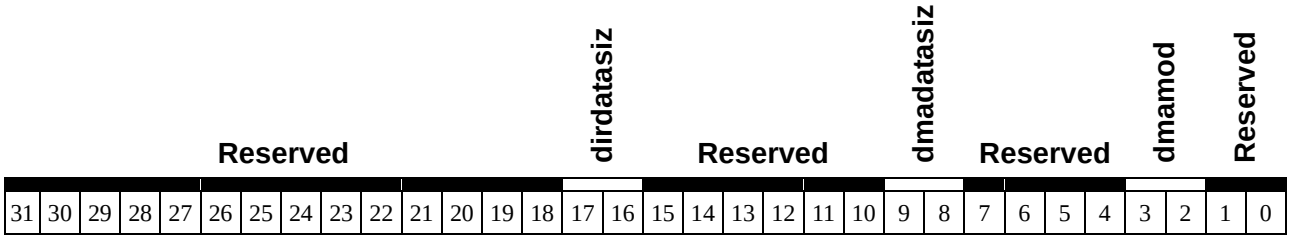
Reserved **<4> <21:9> <24> <31:29>**

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

Address **MGABASE1** + 1E54h (MEM)

Attributes R/W, STATIC, WORD/DWORD

Reset Value 0000 0000 0000 0100 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'	DMA Vertex Write

➧ ***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.

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:*** **dirdatasiz** must be modified ***only*** when there are ***no*** frame buffer reads pending.

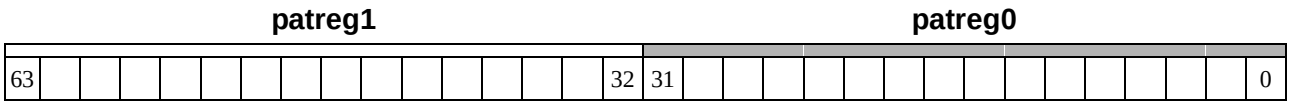
Reserved **<1:0> <7:4> <15:10> <31:18>**

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

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

Attributes WO, FIFO, DYNAMIC, DWORD

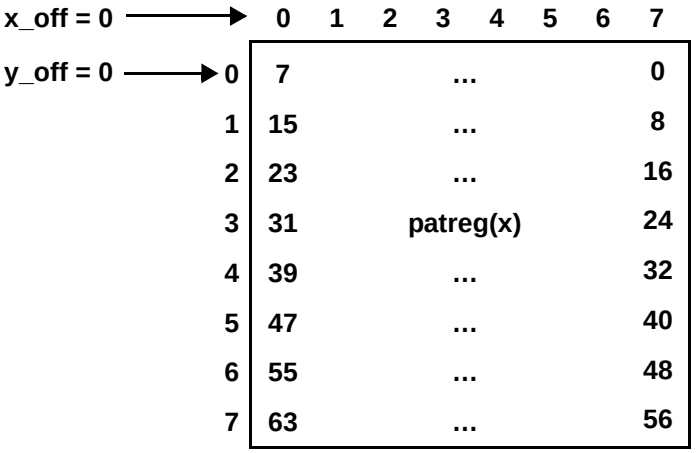
Reset Value unknown



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:

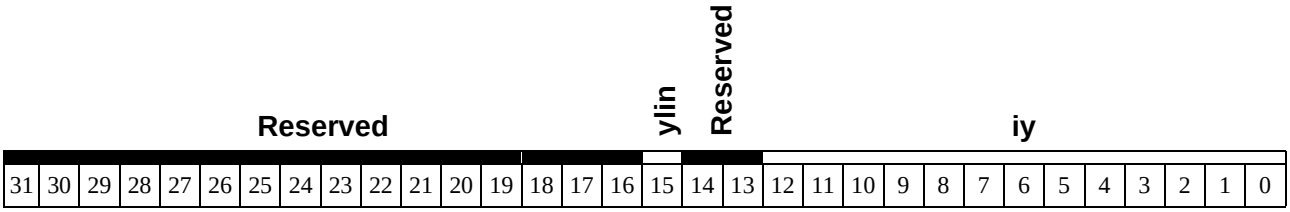


The pattern-pixel pinning can be changed using the **x_off** and **y_off** fields of the **SHIFT** register. See the **SRC0, SRC1, SRC2, SRC3** register on page 3-153.

Address **MGABASE1** + 1C8Ch (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value unknown



iy <12:0>

The y-increment. This field is a 13-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 4096. The **iy** field specifies the increment to be added to or subtracted from **ydst** (see **YDST** on page 3-215) between two destination lines. The **iy** field is also used as the multiplication factor for linearizing the **ydst** register.

Note: Every pitch that is a multiple of 32 (within the range of 32 to 4096 inclusive) is supported for linearization by the hardware.

This register must be loaded with a value that is a multiple of 32 or 64 due to a restriction involving block mode, according to the table below. See 'Constant Shaded Trapezoids / Rectangle Fills' on page 4-40. See page 3-100 for additional restrictions that apply to block mode (**atype** = BLK).

pwidth	value
PW8	64
PW16	32
PW24	64
PW32	32

ylin <15>

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

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

Reserved **<14:13> <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

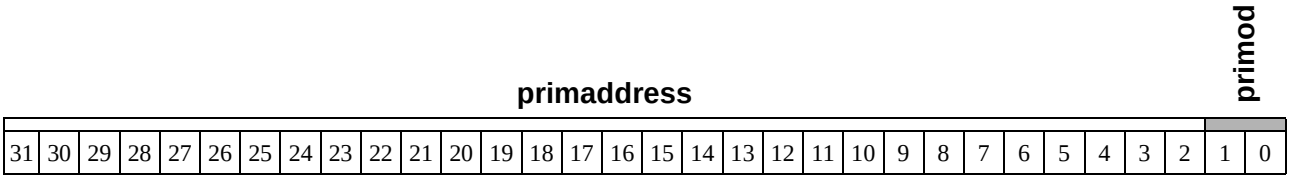
The bits from the **plnwrmsk**<31:0> register are output on the MDQ<31:0> signal and also on MDQ<63:32>. In 8 and 16 bit/pixel configurations, all bits in **plnwrmsk**<31:0> are used, so the mask information must be replicated on all bytes. In 24 bits/pixel, the plane masking feature is limited to the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

Refer to 'Pixel Format' on page 4-21 for the definition of the slice in each mode.

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

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

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



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

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

primaddress <31:2> Primary current address. This field indicates the address to be used to access the primary DMA channel in the system memory when the MGA-G200 is performing bus mastering.

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

The primaddress field is increased by one every time the MGA-G200 terminates a read access at primaddress in the system memory.

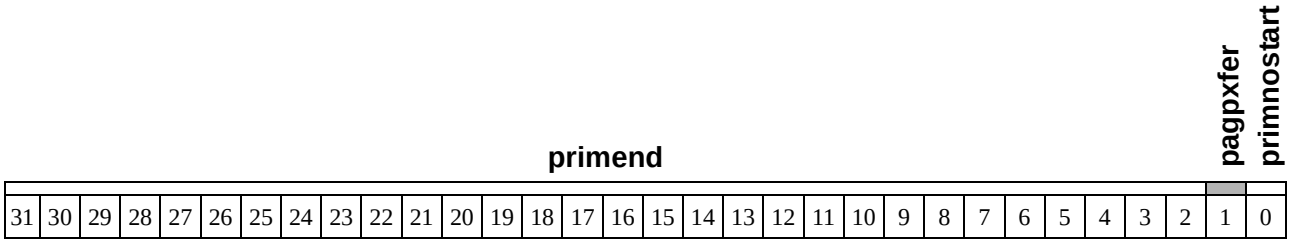
If, when increased, **primaddress** becomes equal to **primend**, the primary channel is empty. Bus mastering stops, and **endprdmasts** is set (refer to the **STATUS** register description).

Refer to 'Programming Bus Mastering for DMA Transfers' on page 4-11 for more details.

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

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

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



primnostart <0> Primary No Start. Writing the **PRIMEND** register with this bit set to '1' will ***not*** restart the primary DMA Channel while a Softrap Interrupt is pending.

pagpxfer <1> Primary AGP Transfer. When '1', the AGP bus cycle will be used for primary DMA data transfer, otherwise, the PCI cycle will be used.

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

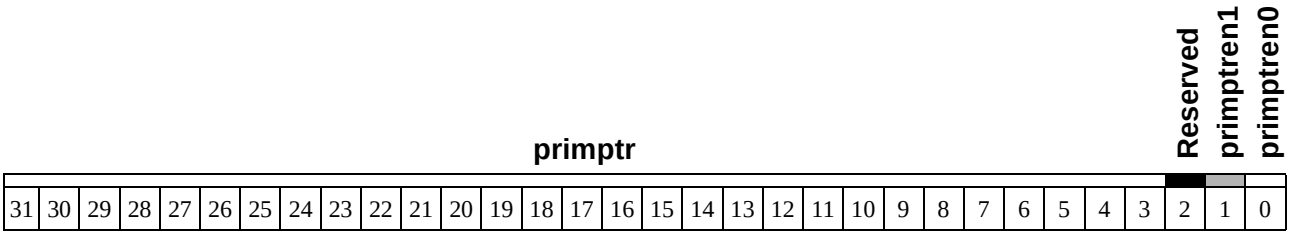
Writing to this field will ***start*** the transfers from the primary DMA channel in the MGA-G200 (*unless* **primnostart** = 0), therefore, **PRIMEND** is the *last* register to be written to (*unless* **primnostart** = 0).

Refer to 'Programming Bus Mastering for DMA Transfers' on page 4-11 for more details.

Address MGABASE1 + 1E50h (MEM)

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

Reset Value ???? ???? ???? ???? ???? ???? ???? ?000b



primptren0 <0> Primary list status fetch Pointer Enable 0. When set to '1', a qword of status data information is written to the system memory (using PCI cycle) at the address corresponding to **primptr** every time a Softrap or Secend or Setupend register write occurs.

Status data information:

- 1st dword: **PRIMADDRESS** register
- 2nd dword: **DWGSYNC** register

primptren1 <1> Primary list status fetch Pointer Enable 1. When set to '1', a qword of status data information is written to the system memory (using PCI cycle) at the address corresponding to **primptr** every time a **DWGSYNC** register write occurs.

primptr <31:3> Primary list status fetch Pointer. This is the qword address where the status data will be placed in system memory.

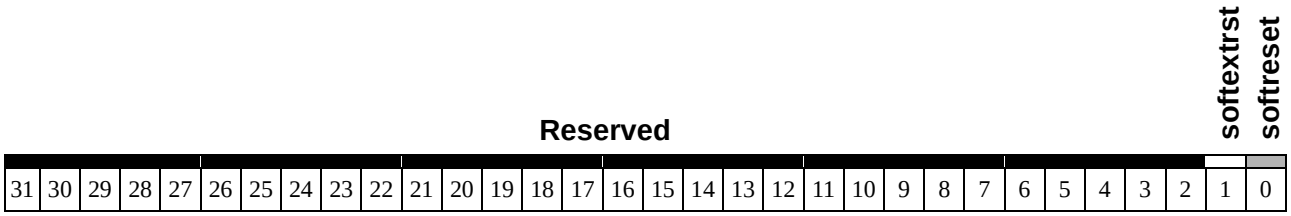
Note: This address *must* be in a PCI accessible range

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

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 allow software resets. This has the effect of flushing the BFIFO and the direct access read cache, and aborting the current drawing instruction. A soft reset will ***not*** generate invalid memory cycles; memory contents are preserved. The **softreset** signal takes place at the end of the PCI write cycle. The reset bit must be maintained at '1' for a minimum of 10 µs to ensure correct reset. After that period, a '0' must be programmed to remove the soft reset. This will:

- reset the set-up engine and set-up engine fifo
- terminate any bus mastering or pseudo-dma transfer
- return some register bits to their soft-reset values (see individual registers).

Refer to Section 4.3.3 on page 4-24 for instructions on when to use this field.

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

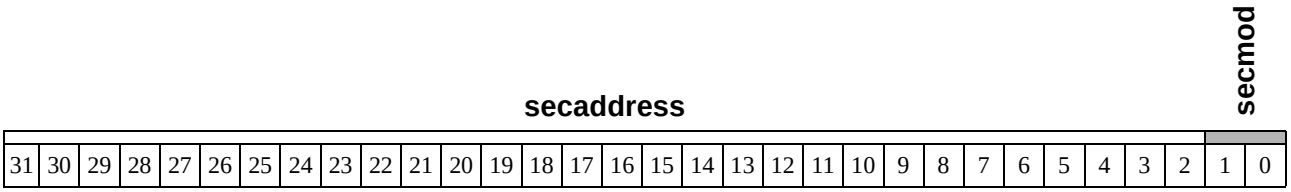
softextrst <1> External Software Reset. When set to '1', this will activate the external reset pin (EXTRSTN). The external reset will remain active until this bit is reset to '0'.

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

Address MGABASE1 + 2C40h (MEM)

Attributes R/W, FIFO, DYNAMIC, DWORD

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



secmod <1:0>

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

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

secaddress <31:2>

Secondary DMA Address. This field indicates the address to be used to access the secondary DMA channel in the system memory when the MGA-G200 is performing bus mastering.

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

The field secaddress is increased by one every time the MGA-G200 terminates a read access at **secaddress** in the system memory.

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

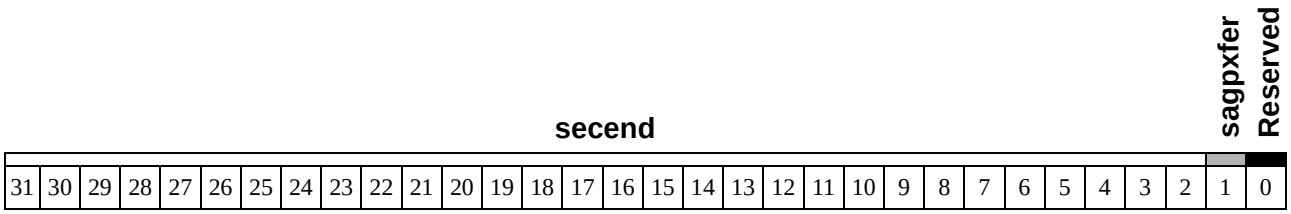
The data that is written to the **SOFTRAP** register will also be loaded into the **secaddress** field.

Note: It is ***not*** possible to write to this register directly. Write access must ***absolutely*** be performed through mastering mode. That is, a primary display list ***must be programmed***. Refer to 'Programming Bus Mastering for DMA Transfers' on page 4-11 for more details.

Address MGABASE1 + 2C44h (MEM)

Attributes R/W, FIFO, STATIC, DWORD

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



sagpxfer <1> Secondary AGP Transfer. When ’1’, the AGP bus cycle will be used for secondary DMA data transfer, otherwise, the PCI cycle will be used.

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

Writing to this field will ***start*** the secondary DMA transfers by the MGA-G200 using bus mastering. The **SECEND** register must always be written to after **SECADDRESS**.

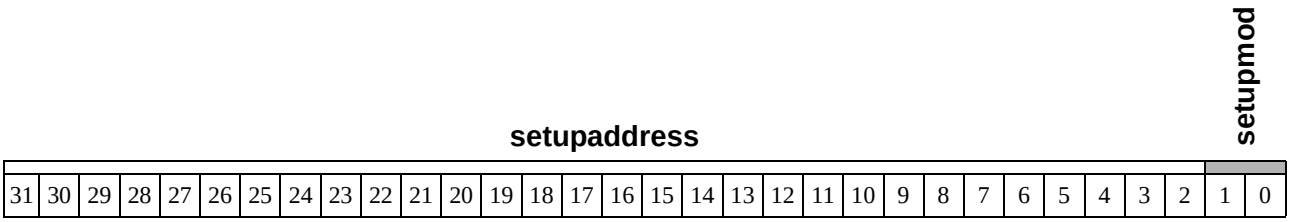
Note: It is ***not*** possible to write to this register directly. Write access must ***absolutely*** be performed through mastering mode. That is, a primary display list ***must be programmed***. Refer to ‘Programming Bus Mastering for DMA Transfers’ on page 4-11 for more details.

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

Address **MGABASE1** + 2CD0h (MEM)

Attributes R/W, FIFO, DYNAMIC, DWORD

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



setupmod <1:0>

Setup Pseudo-DMA mode. This static field indicates the Pseudo-Dma mode to be used to transfer data from system memory to the MGA in mastering mode through the setup DMA channel.

setupmod	*Setup DMA Transfer Mode*
'00'	DMA Vertex Fixed Length Setup List
'01'	Reserved
'10'	Reserved
'11'	Reserved

setup address <31:2>

Setup DMA Address. This field indicates the addresses to be used to access the setup DMA channel in system memory when the MGA-G200 is performing bus mastering.

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

The **setupaddress** increases by one every time the MGA-G200 terminates a read access at **setupaddress** in the system memory.

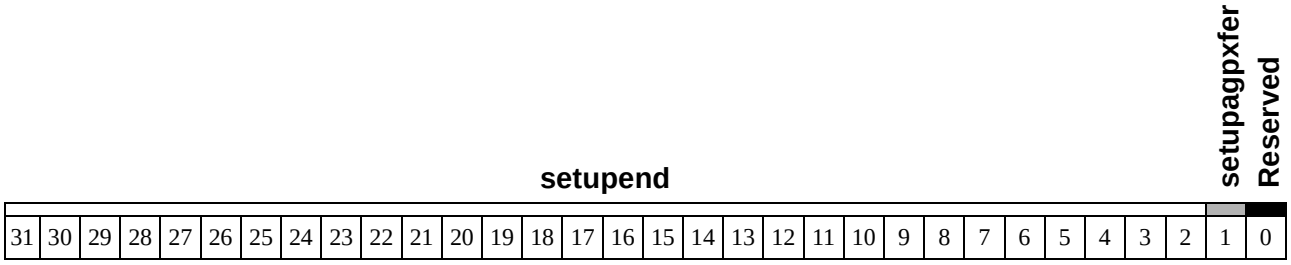
If, after being incremented, **setupaddress** becomes equal to **setupend**, the setup channel is empty. When the last setup DMA generated by the current setup list is complete, bus mastering continues using the primary channel.

Note: It is ***not*** possible to write to this register directly. Write access must ***absolutely*** be performed through mastering mode. That is, a primary display list ***must be programmed***. Refer to 'Programming Bus Mastering for DMA Transfers' on page 4-11 for more details.

Address **MGABASE1** + 2CD4h (MEM)

Attributes R/W, FIFO, STATIC, DWORD

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



setupagpxfer <1> Setup AGP transfer. When '1', the AGP bus cycle will be used for setup DMA data transfer, otherwise, the PCI cycle will be used.

setupend <31:2> Setup End address. The **setupend** field holds the end address + 1 of the setup DMA channel in the system memory, when bus mastering.

Writing to this field will start the setup DMA transfer by the MGA-G200 using bus mastering. The **SETUPEND** register must always be written to after **SETUPADDRESS**.

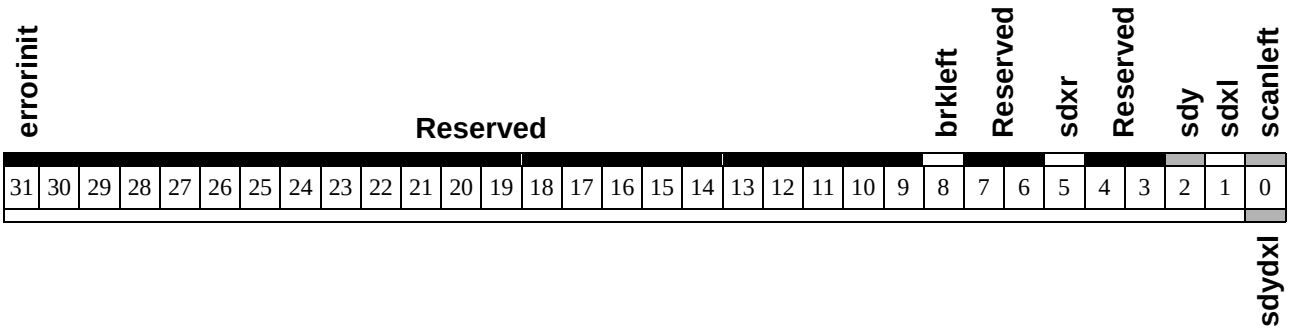
Note: It is ***not*** possible to write to this register directly. Write access must ***absolutely*** be performed through mastering mode. That is, a primary display list ***must be programmed***. Refer to 'Programming Bus Mastering for DMA Transfers' on page 4-11 for more details.

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

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

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value 0??? ???? ???? ???? ???? ???0 ???? ????b



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 3-99). 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

brkleft <8>

Broken left. For trapezoid with subpixel positioning, the start value of the bottom-trap must be adjusted due to the change of the left slope.

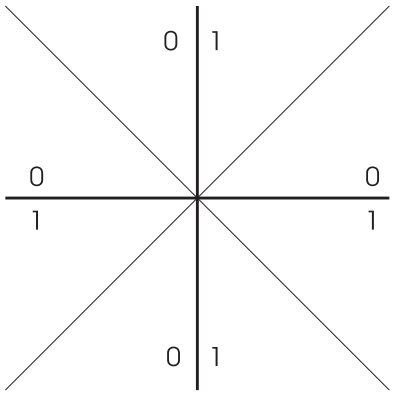
- 0: No pixel adjustment
- 1: Adjust the left edge with the new FXLEFT value

errorinit <31>

This bit is used when **opcod** = AUTOLINE_OPEN or AUTOLINE_CLOSE. It specifies the content of AR1 at the end of the initialization sequence.

- 0: AR1 holds 2x’b’-‘a’- sdy
- 1: AR1 holds 2x’b’-’a’-MGAQuadrantError

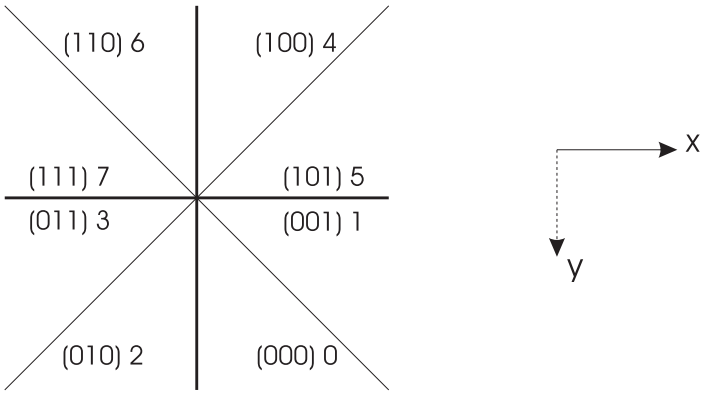
where MGAQuadrantError takes the following values:



"1"- Indicates the quadrants in which the error term should be biased.

The MGA's convention for numbering the quadrants is as follows, where 'sdydxl_Major_X' = 1, 'sdxl_SUB' = 2, 'sdy_SUB' = 4

then:



MGAQuadrantError (sdydx1 + (sdx1 << 1) + (sdy <<2)) = { 1,1,0,1,1,0,0,0 }

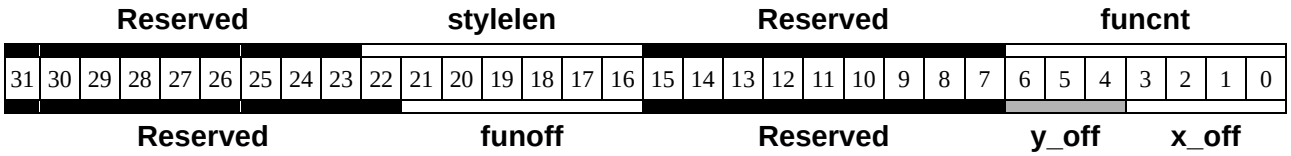
Reserved **<4:3> <7:6> <30:9>**

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



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.

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.

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.

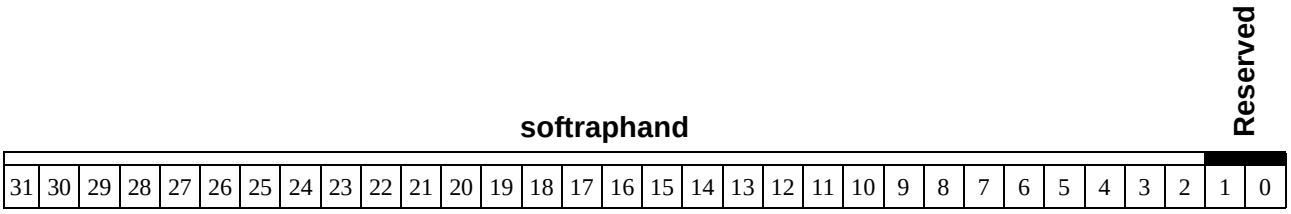
funoff <21:16> Funnel shifter offset. For Blit operations, this field is used to specify a bit offset in the funnel shifter count. In this case **funoff** is interpreted as a 6-bit signed value.

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 3-153), so its value is the number of bits in the complete pattern minus one. For 3D vectors, this field must be initialized to ‘0’.

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

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

Address MGABASE1 + 2C48h (MEM)
Attributes R/W, FIFO, DYNAMIC, DWORD
Reset Value unknown



softraphand <31:2> Soft trap handle. When this field is written, a soft trap interrupt is generated, and the primary DMA channel is stopped (the **softrapen** and **endprdmasts** fields of **STATUS** are set to '1'). To restart the primary DMA channel, **PRIMEND** must be written.

SOFTRAP can be written by either the primary or secondary DMA channel. If part of the secondary channel, both the primary and the secondary channel will stop.

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

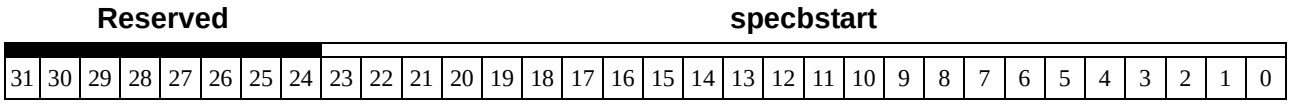
Note: It is ***not*** possible to write to this register directly. Write access must ***absolutely*** be performed through mastering mode. That is, a primary display list ***must be programmed***. Refer to 'Programming Bus Mastering for DMA Transfers' on page 4-11 for more details.

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

Address MGABASE1 + 2C98h (MEM)

Attributes R/W, FIFO, DYNAMIC, DWORD

Reset Value unknown



specbstart <23:0> Specular Lighting Blue Start value. This field holds a signed 9.15 value in two's complement notation.

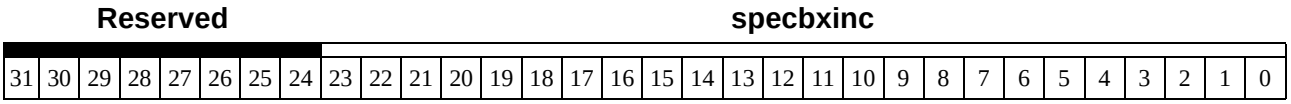
For TEXTURE_TRAP primitives, the SPECBSTART register must be initialized with the starting specular light value for the blue component.

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

Address MGABASE1 + 2C9Ch (MEM)

Attributes R/W, FIFO, STATIC, DWORD

Reset Value unknown



specbxinc <23:0> Specular Lighting Blue X Increment value. This field holds a signed 9.15 value in two's complement notation.

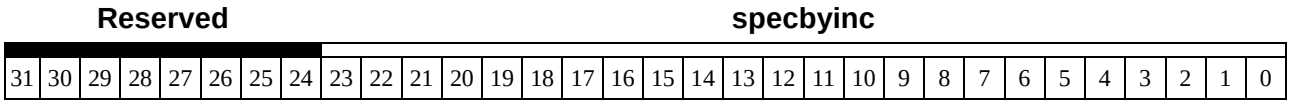
For TEXTURE_TRAP primitives, the SPECBXINC register holds the blue increment value along the x-axis.

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

Address MGABASE1 + 2CA0h (MEM)

Attributes R/W, FIFO, STATIC, DWORD

Reset Value unknown



specbyinc <23:0> Specular Lighting Blue Y Increment value. This field holds a signed 9.15 value in two's complement notation.

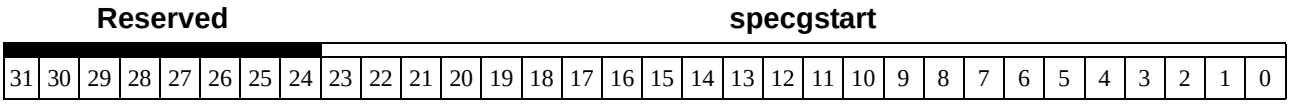
For TEXTURE_TRAP primitives, the SPECBYINC register holds the blue increment value along the y-axis.

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

Address MGABASE1 + 2C8Ch (MEM)

Attributes R/W, FIFO, DYNAMIC, DWORD

Reset Value unknown



specgstart <23:0> Specular Lighting Green Start value. This field holds a signed 9.15 value in two's complement notation.

For TEXTURE_TRAP primitives, the SPECGSTART register must be initialized with the starting specular light value for the green component.

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

Address MGABASE1 + 2C90h (MEM)

Attributes R/W, FIFO, STATIC, DWORD

Reset Value unknown

Reserved	specgxinc

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

specgxinc <23:0> Specular Lighting Green X Increment value. This field holds a signed 9.15 value in two's complement notation.

For TEXTURE_TRAP primitives, the SPECGXINC register holds the green increment value along the x-axis.

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

Address MGABASE1 + 2C94h (MEM)

Attributes R/W, FIFO, STATIC, DWORD

Reset Value unknown

Reserved	specgyinc

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

specgyinc <23:0> Specular Lighting Green Y Increment value. This field holds a signed 9.15 value in two's complement notation.

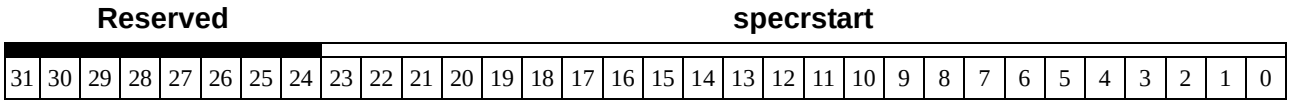
For TEXTURE_TRAP primitives, the SPECGYINC register holds the green increment value along the y-axis.

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

Address MGABASE1 + 2C80h (MEM)

Attributes R/W, FIFO, DYNAMIC, DWORD

Reset Value unknown



specrstart <23:0> Specular Lighting Red Start value. This field holds a signed 9.15 value in two's complement notation.

For TEXTURE_TRAP primitives, the SPECRSTART register must be initialized with the starting specular light value for the red component.

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

Address MGABASE1 + 2C84h (MEM)

Attributes R/W, FIFO, STATIC, DWORD

Reset Value unknown

Reserved	specrxinc

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

specrxinc <23:0> Specular Lighting Red X Increment value. This field holds a signed 9.15 value in two's complement notation.

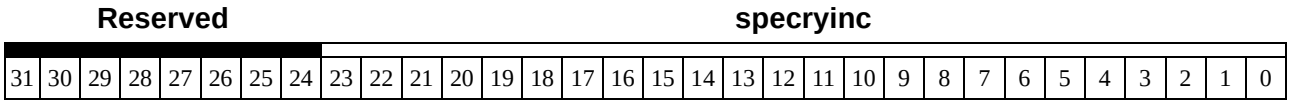
For TEXTURE_TRAP primitives, the SPECRXINC register holds the red increment value along the x-axis.

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

Address MGABASE1 + 2C88h (MEM)

Attributes R/W, FIFO, STATIC, DWORD

Reset Value unknown



specryinc <23:0> Specular Lighting Red X Increment value. This field holds a signed 9.15 value in two's complement notation.

For TEXTURE_TRAP primitives, the SPECRYINC register holds the red increment value along the y-axis.

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

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

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value unknown

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

srcreg <127:0>

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

For LINE with the RPL or RSTR attribute, the source register is used to store the line style. The **funcnt** field of the **SHIFT** register points to the selected source register bit being used as the linestyle for the current pixel. Refer to Section 4.5.4.3 on page 4-33 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 4-41 for more details.

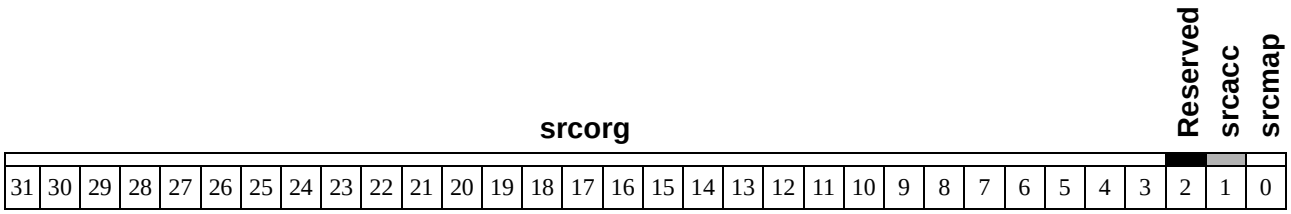
For all BLIT operations, and for TRAP or LINE using depth mode, the source register is used internally for intermediate data.

A write to the **PAT** registers (see page 3-128) will load the **SRC** registers.

Address MGABASE1 + 2CB4h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



srcmap <0> Source Map. A memory space indicator, this field indicates the map location.

- 0: the source surface is in the frame buffer memory.
- 1: the source surface is in the system memory.

srcacc <1> Source Access type. This field specifies the mode used to access the map.

- 0: PCI access.
- 1: AGP access.

Note: This field is *not* considered if the source resides in the frame buffer space.

srcorg <31:3> Source Origin. This field provides an offset value for the position of the first pixel for a source surface. The **srcorg** field is used during BitBlit operations. The **srcorg** field corresponds to a qword address in memory.

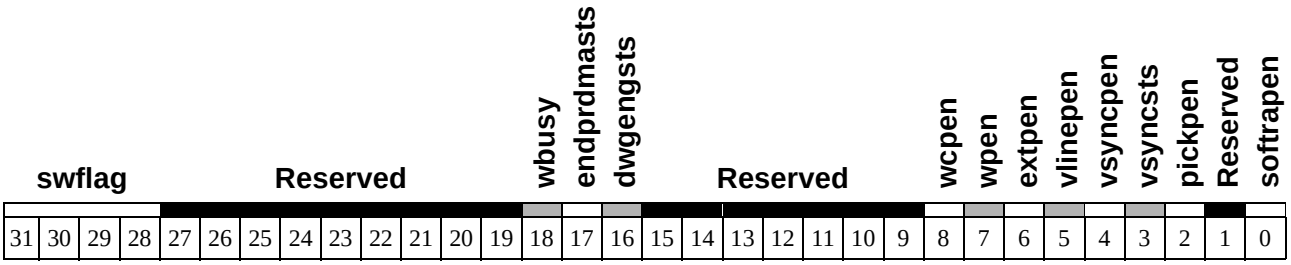
Note: **srcorg** [4:3] must always be loaded with ‘00’.

Reserved <2> Reserved. When writing to this register, the bits in this field ***must*** be set to ‘0’.

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

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

Reset Value 0000 0000 0000 0010 0000 0000 0?00 0000b



softrapen
RO <0>

Soft trap interrupt pending. When set to '1', this field indicates that the MGA-G200 has stopped reading through the primary channel.

This field is set to '1' when the **SOFTRAP** register is written. This field is cleared through the **softrapiclr** field (see **ICLEAR** on page 3-116) or upon soft or hard reset.

pickpen
RO <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 3-116) or upon soft or hard reset.

vsyncsts
RO <3>

VSYNC status. Set to '1' during the VSYNC period. This bit follows the VSYNC signal.

vsyncpen
RO <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
RO <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 3-265. This bit is cleared through the **vlineiclr** bit (see **ICLEAR** on page 3-116) or upon soft or hard reset.

extpen
RO <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.

wpen
RO <7>

WARP interrupt Pending. When set to '1', indicates that a WARP interrupt has occurred. This bit is cleared through the **wiclr** bit (see **ICLEAR**) or upon soft or hard reset.

wcpen
RO <8>

WARP Cache interrupt Pending. When set to '1', indicates that a WARP cache interrupt has occurred. This bit is cleared through the **wciclr** bit (see **ICLEAR**) or upon soft or hard reset.

dwgengsts
RO <16>

Drawing engine status. Set to '1' when the drawing engine is busy. A busy condition will be maintained during any of these conditions:

- bfifo is ***not*** empty
- warpfifo is ***not*** empty
- the drawing engine is still processing and sending commands

- the memory has not completed the last memory access (from the drawing engine).
- The AGP chipset has not completed the last memory access (from the drawing engine).

endprdmasts RO <17>

End of primary DMA channel status. When set to ‘1’, this bit indicates that the MGA-G200 has completed its DMA transfers (**primaddress** = **primend** and **secaddress** = **secend** and **setupaddress** = **setupend**), or when a soft trap interrupt occurs. Restarting the primary DMA by accessing **PRIMEND** will reset **endprdmasts** to ‘0’.

Note: Refer to ‘Programming Bus Mastering for DMA Transfers’ on page 4-11 for more details.

wbusy RO <18>

WARP Busy. When set to ‘1’, indicates that the WARP is ***not*** idle; it may be RUNning, WAITing, STALLed or loading microcode (cachemiss).

swflag R/W <31:28>

Software Flag. These bits have no effect on the chip.

Reserved **<1> <15:9> <27:19>**

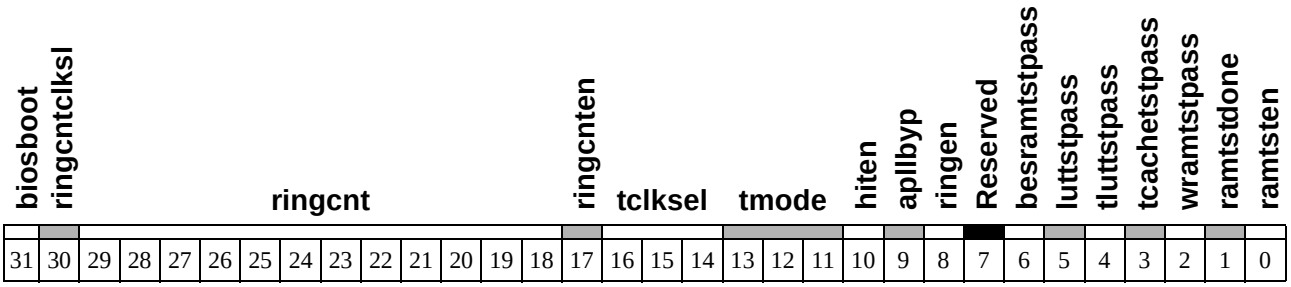
Reserved. When writing to this register, the bits in these fields ***must*** be set to ‘0’. Reading will return ‘0’s.

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 + 1E48h (MEM)

Attributes R/W, DYNAMIC, WORD/DWORD

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



ramtsten <0>

Memory self Test Enable (valid when tmode = ‘010’)

- 0: Disable memory self test
- 1: Enable memory self test

ramtstdone RO <1>

Memory self Test Done

- 0: The memory self test is ***not*** finished yet
- 1: The memory self test is done

wramtstpass RO <2>

WARP Ram self Test Pass. This field is composed of the PASS output of the ram macro of the WARP engine.

- 0: Test failed
- 1: Test passed

tcachetst pass RO <3>

Texture Cache ram self Test Pass. This field is composed of the PASS output of the ram macro of the texture cache.

- 0: Test failed
- 1: Test passed

tluttstpass RO <4>

Texture LUT self Test Pass. This field is composed of the PASS output of the ram macro of the texture LUT.

- 0: Test failed
- 1: Test passed

luttstpass RO <5>

LUT self Test Pass. This field is composed of the PASS output of the ram macro of the RAMDAC LUT.

- 0: Test failed
- 1: Test passed

besramtst pass RO <6>

Back End Scaler Ram self Test Pass. This field is composed of the PASS output of the ram macro of the backend scaler.

- 0: Test failed
- 1: Test passed

ringen
<8>

Ring oscillator enable (valid when **tmode** = ‘001’)

• 0: Disable the ring oscillator
• 1: Enable the ring oscillator

apllbyp
<9>

AGP PLL bypass mode

• 0: The AGP PLL is ***not*** bypassed. The output of the PLL is used as the agp 2x clock.
• 1: The AGP PLL is bypassed by **VDCLK** (VDCLK is used as the agp 2x clock).

hiten
RO
<10>

Hit Enable. Cache hit signal

tmode
<13:11>

Extra Test Mode. This field is used to reconfigure the pins to access internal signals for test purposes.

• ‘000’: Normal mode
• ‘001’: Observe mode
• ‘010’: Memory self-test mode
• (others): Reserved

tclksel
<16:14>

Test Clock Select (valid when **tmode** = ‘001’). This field selects which clock to output on DCC_0 when in observe mode (**tmode** = ‘001’)

• 000: pixpll
• 001: pixpll / 4
• 010: syspll
• 011: syspll / 4
• 100: agppll
• 101: agppll / 4
• 110: rsrv
• 111: rsrv

ringcnten
<17>

Ring Oscillator Counter Enable.

• 0: disables the ring counter
• 1: enables the ring counter

ringcnt
RO dynamic
<29:18>

Ring Count (value in the ring counter). Number of cycles of the ring oscillator while **pixclk** goes through 2048 cycles.

ringcntclksl
<30>

Ring Count Clock Select. Select the clock to use for frequency measurement.

• 0: use the ring oscillator
• 1: use the PCI clock

biosboot
RO
<31>

Bios Boot. Indicates the size of the serial eeprom, if present

• 0: indicates either the absence of a serial eeprom, or the presence of a 128 Byte (up to 512 Bytes) serial eeprom on board.
• 1: indicates the presence of a 32KB or 64KB serial eeprom containing the BIOS.

Reserved **<7> <30>**

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

‘0’s when read.

Attributes MGABASE1 + 2C5Ch (MEM)
Reset Value WO, FIFO, STATIC, DWORD
Reset Value unknown

texbordercol

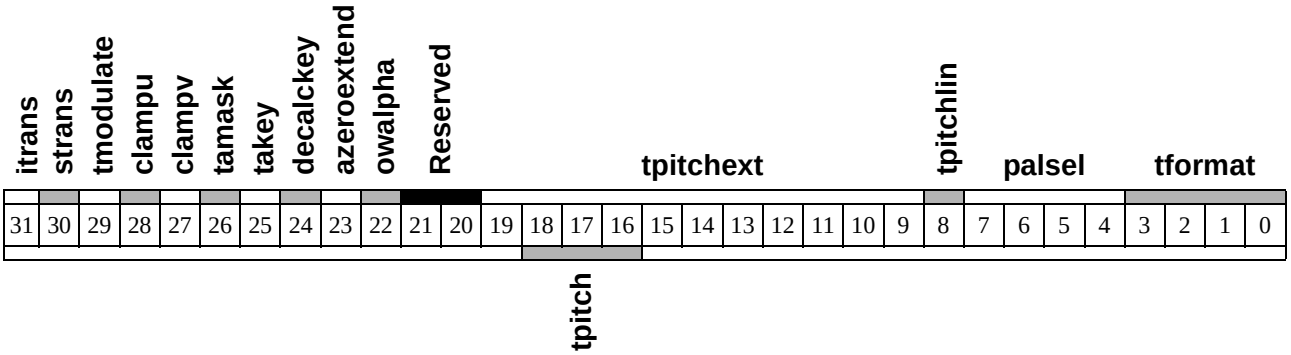
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

texbordercol <31:0> Texture Border Color. The texture's border color in 32-bit ARGB format.

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

Attributes WO, FIFO, STATIC, DWORD

Reset Value unknown



tformat <3:0> Texel Format. Specifies the texture's texel format.

tformat			
Value	***Mnemonic***	***Mode***	***Format***
'0000'	TW4	4 bits/texel	(Goes through the LUT)
'0001'	TW8	8 bits/texel	(Goes through the LUT)
'0010'	TW15	1:5:5:5	(A:R:G:B)
'0011'	TW16	5:6:5	(R:G:B)
'0100'	TW12	4:4:4:4	(A:R:G:B)
'0110'	TW32	8:8:8:8	(A:R:G:B)
'1010'	TW422	4:2:2	(VYUY)

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

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

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

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

Note: If **tformat** is set to TW422, **tpitchext** ***must*** be a multiple of 8.

Note: In repeat mode (clampu or clampv = '0' **tpitchext** ***must*** be programmed with a power of 2 value.

Note: When using a linear pitch with mip-mapping, **tpitchext** ***must*** be programmed with a multiple of 16 - 1 value.

➨ ***Note:*** texture pitch must be greater or equal than the texture width (**twmask** + 1)

tpitch <18:16>

When the **tpitchlin** bit is set to ‘0’, the **tpitch** is set according to the table below:

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

➨ ***Note:*** texture pitch must be greater or equal than the texture width (**twmask** + 1)

owalpha <22>

Over-Write Alpha. When this bit is set to ‘1’, color keying can overwrite the alpha from the texture.

azeroextend <23>

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

- 0: Replicate the tamask and takey
- 1: Zero extend

decalckey <24>

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

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

takey <25>

Texture alpha key. This field indicates which polarity is defined as transparent.

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

➽ ***Note:*** Texture alpha-keying is possible only if **twidth<1:0>** = TW15

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

- 0: repeat
- 1: clamp

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

- 0: repeat
- 1: clamp

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

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

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

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

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

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

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

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

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

```
alpha = texel<15>

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

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

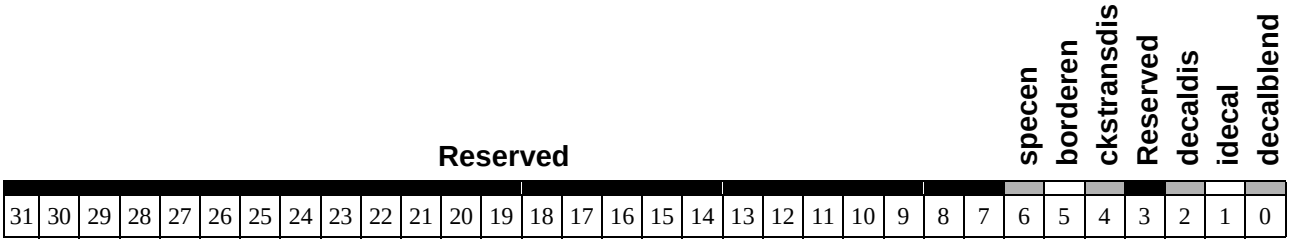
Reserved <21:20>

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

Address **MGABASE1** + 2C3Ch (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



decalblend <0> True Decal Enable. When enabled, with the decal function selected (tmodulate = ‘0’), true decal will perform a blend between the texture (texel) and the surface RGB ALU.

idecal <1> Invert Decal information.

decaldis <2> Decal Disable.

- 0: decal is made between the texture and the surface color (RGB ALU).
- 1: the texel is always chosen.

ckstransdis <4> Color Key Surface Transparency Disabled. Disables surface transparency from color keying.

borderen <5> Border Enable. The constant border color (**texbordercol**) is used instead of duplicating the texel.

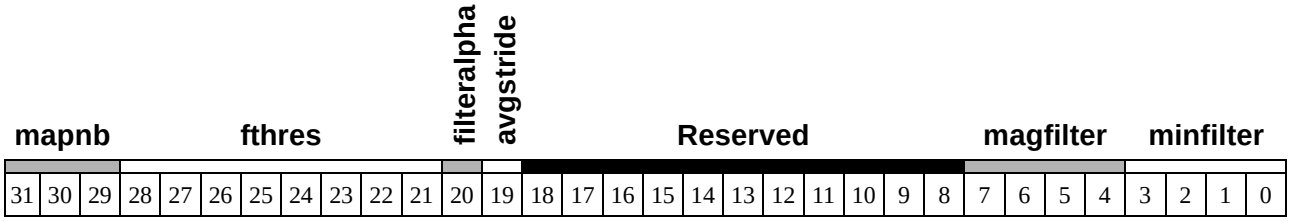
specen <6> Specular Lighting Enable. Specular lighting can be performed on any 3D operation.

Reserved <31:7> Reserved. When writing to this register, the bits in this field ***must*** be set to ‘0’.

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

Attributes WO, FIFO, STATIC, DWORD

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



minfilter <3:0> Minify Filtering Mode. If the texture is minified (based on the comparison between the stride and the '**fthres**' value) the filter mode is selected by the **minfilter** value.

minfilter	*mnemonic*	*filter mode*
'0000'	NRST	Nearest
'0001'		RSVD
'0010'	BILIN	Bi-linear
'0011'	CNST	Constant Bi-linear (0.25)
'0100'		RSVD
'0101'		RSVD
'0110'		RSVD
'0111'		RSVD
'1000'	MM1S	1 sample mip-mapping
'1001'	MM2S	2 sample mip-mapping
'1010'	MM4S	4 sample mip-mapping
'1011'		RSVD (5 sample)
'1100'	MM8S	8 sample mip-mapping

magfilter <7:4> Magnify Filtering Mode. If the texture is magnified (based on the comparison between the stride and the '**fthres**' value) the filter mode is selected by the **magfilter** value.

magfilter	*mnemonic*	*filter mode*
'0000'	NRST	Nearest
'0001'		RSVD
'0010'	BILIN	Bi-linear
'0011'	CNST	Constant Bi-linear (0.25)
'0100'		RSVD
'0101'		RSVD
'0110'		RSVD
'0111'		RSVD

avgstride <19> Average Stride. Takes the average between the x and y strides instead of the maximum.

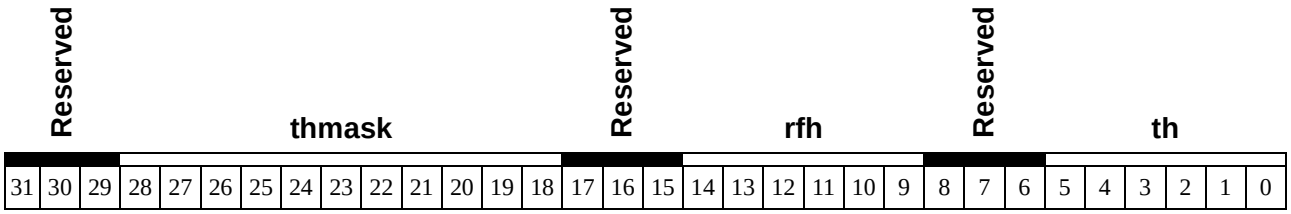
filteralpha <20> Apply filtering on the alpha (texture alpha).

fthres <28:21>	Filter Threshold. The **fthres** field *must* be programmed with the square value of the step wanted as the threshold between minify and magnify. If the actual step in the texture is bigger than fthres, the engine will be minifying the texture and will apply the **minfilter** on the texture. Otherwise, the **magfilter** will be used. ➽ ***Note:*** This field holds an unsigned 4.4 value.
mapnb <31:29>	Map Number. Specifies how many maps are used for mip-mapping. The valid range is 0 to 4.
Reserved <18:8>	Reserved. When writing to this register, the bits in this field ***must*** be set to ‘0’.

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

Attributes WO, FIFO, STATIC, DWORD

Reset Value unknown



th <5:0>

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

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

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

rfh <14:9>

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

thmask <28:18>

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

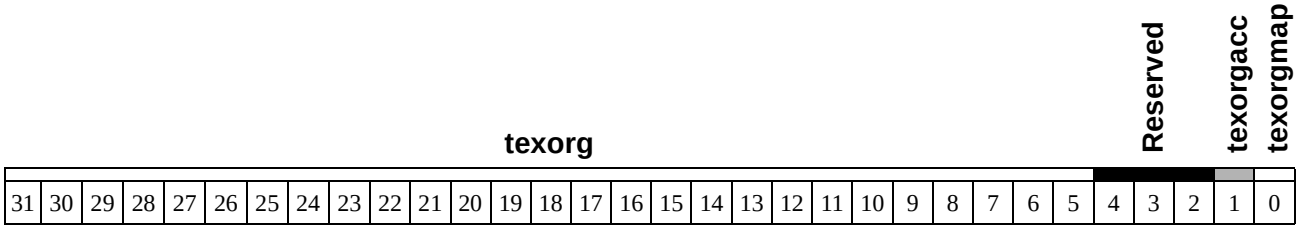
Note: The repeat mode (**clampv** = 0) will work properly if **thmask** is set to a value that is a power of two minus one.

Note: The minimum texture height supported (including maps in mip-mapping) is 8. **thmask** ***must*** be >= 7.

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

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

Address MGABASE1 + 2C24h (MEM)
Attributes WO, FIFO, STATIC, DWORD
Reset Value unknown



texorgmap <0> Memory Space indicator. This field indicates the map location.

- 0: texture is in FB
- 1: texture is in system memory

texorgacc <1> Access Type. This field specifies the mode used to access the map.

- 0: PCI access
- 1: AGP access

Note: If the texture resides in the frame buffer space, this field is ***not*** considered.

texorg <31:5> Origin of map 0. The **texorg** field provides an offset value (the base address), to the position the first texel in the texture.

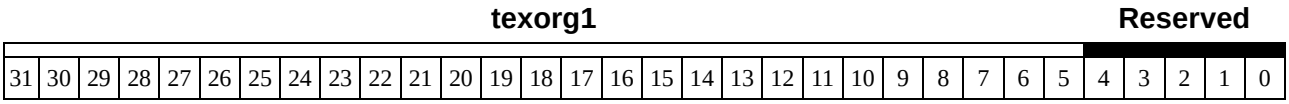
The **texorg** field corresponds to a 256-bit aligned address in memory. This register must be set so that there is no overlap with either the frame buffer or the Z-depth buffer.

Reserved <4:2> Reserved. When writing to this register, the bits in this field ***must*** be set to ‘0’.

Address MGABASE1 + 2CA4h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value unknown



texorg1 <31:5> Origin of map 1. The **texorg1** field provides an offset value (the base address), to the position the first texel in the texture.

The **texorg1** field corresponds to a 256-bit aligned address in memory. This register must be set so that there is no overlap with either the frame buffer or the Z-depth buffer.

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

Note: Fields **texorgmap** and **texorgacc**, in the **TEXORG** register, apply to **TEXORG1** (see page 3-169 for more information).

Address **MGABASE1** + 2CA8h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value unknown

texorg2																										Reserved				
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

texorg2 <31:5> Origin of map 2. The **texorg2** field provides an offset value (the base address), to the position the first texel in the texture.

The **texorg2** field corresponds to a 256-bit aligned address in memory. This register must be set so that there is no overlap with either the frame buffer or the Z-depth buffer.

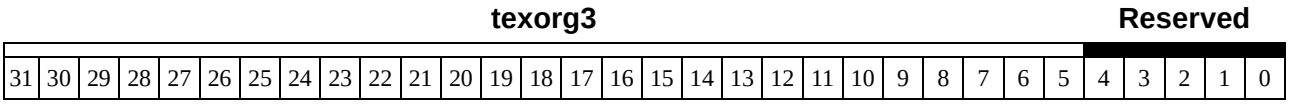
Reserved <4:0> Reserved. When writing to this register, the bits in this field ***must*** be set to ‘0’.

➽ ***Note:*** Fields **texorgmap** and **texorgacc**, in the **TEXORG** register, apply to **TEXORG2** (see page 3-169 for more information).

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

Attributes WO, FIFO, STATIC, DWORD

Reset Value unknown



texorg3 <31:5> Origin of map 3. The **texorg3** field provides an offset value (the base address), to the position the first texel in the texture.

The **texorg3** field corresponds to a 256-bit aligned address in memory. This register must be set so that there is no overlap with either the frame buffer or the Z-depth buffer.

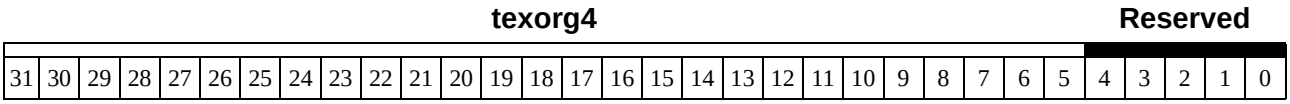
Reserved <4:0> Reserved. When writing to this register, the bits in this field ***must*** be set to ‘0’.

Note: Fields **texorgmap** and **texorgacc**, in the **TEXORG** register, apply to **TEXORG3** (see page 3-169 for more information).

Address MGABASE1 + 2CB0h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value unknown



texorg4 <31:5> Origin of map 4. The **texorg4** field provides an offset value (the base address), to the position the first texel in the texture.

The **texorg4** field corresponds to a 256-bit aligned address in memory. This register must be set so that there is no overlap with either the frame buffer or the Z-depth buffer.

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

➽ ***Note:*** Fields **texorgmap** and **texorgacc**, in the **TEXORG** register, apply to **TEXORG4** (see page 3-169 for more information).

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

Attributes WO, FIFO, STATIC, DWORD

Reset Value unknown

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

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

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

twidth	15 tkmask 0
TW4	0 0 0 0 0 0 0 0 0 0 0 0 m a s k
TW8	0 0 0 0 0 0 0 0 - - m a s k - -
TW12	- - - - - - m a s k - - - - - -
TW15	- - - - - - m a s k - - - - - -
TW16	- - - - - - m a s k - - - - - -
TW32	- - - - - - m a s k - - - - - -
TW422	- - - - - - m a s k - - - - - -

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

Address MGABASE1 + 2C38h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b

tkmaskh																tckeyh															
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

tckeyh <15:0> Texture Color Key High. High portion (16 MSB) of the color key.

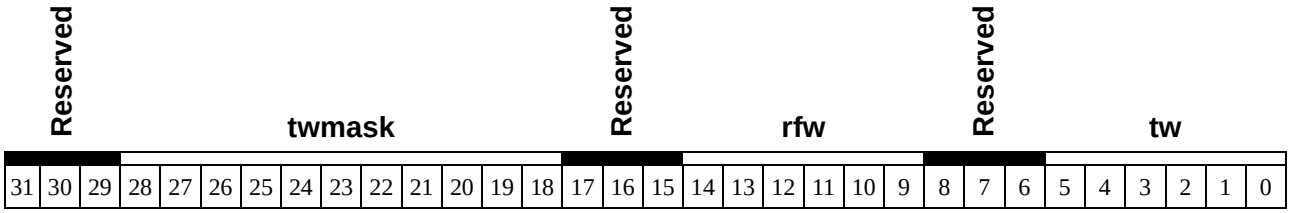
tkmaskh <31:16> Texture Key Masking High. High portion (16 MSB) of the keying mask.

tformat	31 tkmaskh 16
TW4	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
TW8	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
TW12	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
TW15	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
TW16	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
TW32	- - - - - - m a s k - - - - - -
TW422	- - - - - - m a s k - - - - - -

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

Attributes WO, FIFO, STATIC, DWORD

Reset Value unknown



tw <5:0>

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

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

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

rfw <14:9>

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

twmask <28:18>

Width Mask. Determining the usable width of the texture (select the texel that the repeat or clamping will be performed on). This field holds an 11-bit unsigned value usually set to: (texture width) - 1.

Note: The repeat mode (**clampu** = 0) will work properly *only* if the **twmask** is set to a value that is a power of two minus one.

Note: The minimum texture width supported (including maps in mip-mapping) is 8. **twmask *must*** be >= 7.

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

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

Address MGABASE1 + 2C00h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value unknown

tmr0

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

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

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

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

Attributes WO, FIFO, STATIC, DWORD

Reset Value unknown

tmr1

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

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

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

Address MGABASE1 + 2C08h (MEM)
Attributes WO, FIFO, STATIC, DWORD
Reset Value unknown

tmr2

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

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

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

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

Attributes WO, FIFO, STATIC, DWORD

Reset Value unknown

tmr3

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

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

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

Address MGABASE1 + 2C10h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value unknown

tmr4

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

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

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

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

Attributes WO, FIFO, STATIC, DWORD

Reset Value unknown

tmr5

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

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

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

Address MGABASE1 + 2C18h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value unknown

tmr6

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

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

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

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

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value unknown

tmr7

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

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

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

Address MGABASE1 + 2C20h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value unknown

tmr8

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

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

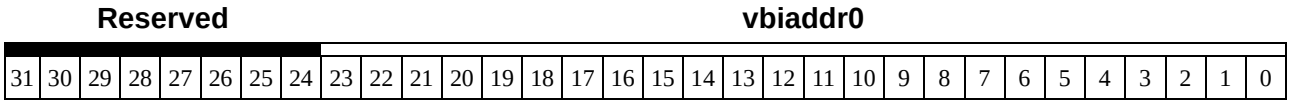
For texture mapping, the **TMR8** register is used to scan the left edge of the trapezoid for the q/wc parameter. This register must be initialized with the starting q/wc-value.

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

Address MGABASE1 + 3E08h (MEM)

Attributes WO, BYTE/WORD/DWORD, STATIC

Reset Value unknown



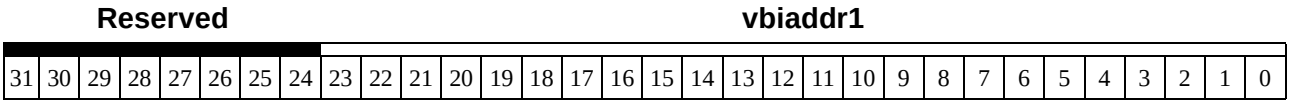
vbiaddr0 <23:0> VBI Data Start Address Window 0. Start address in bytes in the frame buffer of VBI data for Window 0. This field must be loaded with a multiple of 512 (the 9 LSBs = ‘0’).

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

Address MGABASE1 + 3E0Ch (MEM)

Attributes WO, BYTE/WORD/DWORD, STATIC

Reset Value unknown



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

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

Address MGABASE1 + 1E20h (MEM)

Attributes RO, WORD/DWORD, DYNAMIC

Reset Value unknown

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

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

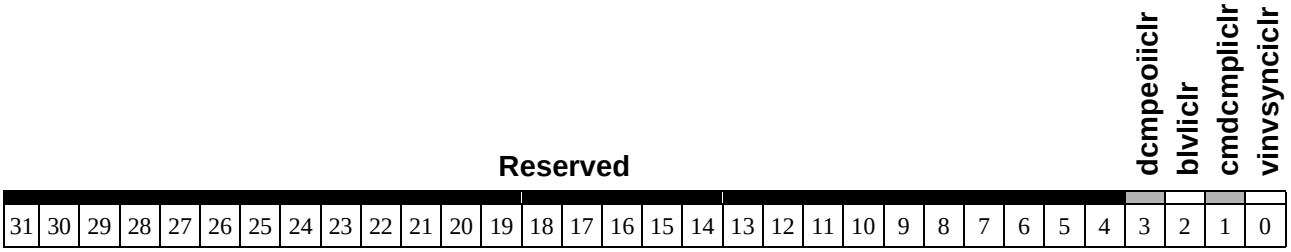
> ***Note:*** This register must be read using a word or dword access, because the value might change between two byte accesses. A sample and hold circuit will ensure a stable value for the duration of one PCI read access.

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

Address MGABASE1 + 3E34h (MEM)

Attributes WO, BYTE/WORD/DWORD, STATIC

Reset Value unknown



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

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

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

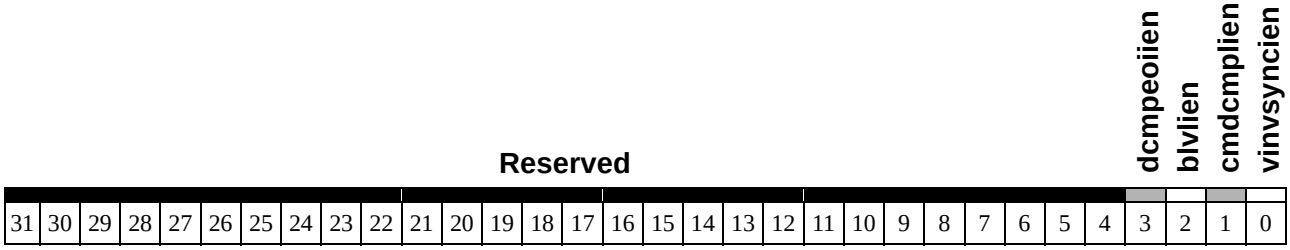
dcmpeoiiclr <3> Codec decompression end of image interrupt clear. When writing a ‘1’ to this bit, the end of image interrupt pending flag is cleared.

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

Address MGABASE1 + 3E38h (MEM)

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

Reset Value XXXX XXXX XXXX XXXX XXXX XXXX XXXX 0000b



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

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

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

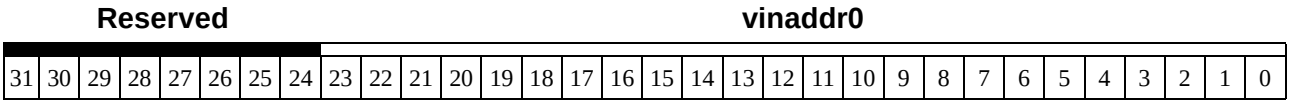
dcmpeoiien <3> Codec Decompression End Of Image Interrupt Enable. When set to a ‘1’, an interrupt will be generated when the Codec Interface is performing decompression and the end of image marker is detected in the stream.

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

Address MGABASE1 + 3E10h (MEM)

Attributes WO, BYTE/WORD/DWORD, STATIC

Reset Value unknown



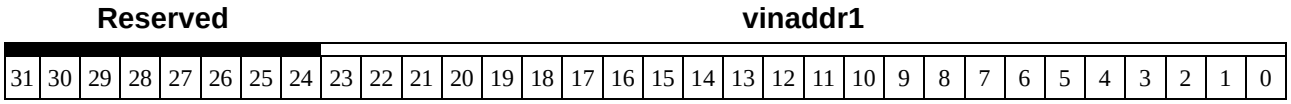
vinaddr0 <23:0> Video write start Address for window 0. Start address in frame buffer of window 0.(byte boundary). This field *must* be loaded with a multiple of 8 (the 3 LSBs = ‘0’).

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

Address MGABASE1 + 3E14h (MEM)

Attributes WO, BYTE/WORD/DWORD, STATIC

Reset Value unknown



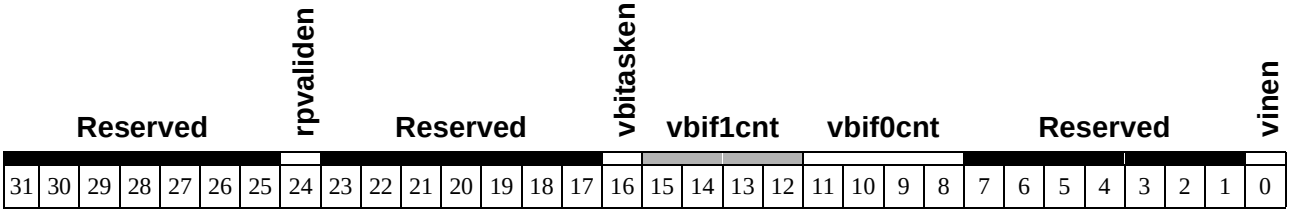
vinaddr1 <23:0> Video write start Address for window 1. Start address in frame buffer of window 0.(byte boundary). This field *must* be loaded with a multiple of 8 (the 3 LSBs = ‘0’).

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

Address MGABASE1 + 3E1Ch (MEM)

Attributes WO, BYTE/WORD/DWORD, STATIC

Reset Value 0000 0001 0000 0001 1010 1011 0000 0000b



vinen <0> Video In Enable. When this bit is '0' the video-in macro is disabled, the video input registers, status fields, and interrupt enables are *reset*.

- 0: reset/disable
- 1: enable video-in macro

vbif0cnt <11:8> VBI Field 0 Count. This field is used to initialize a counter with the number of VBI lines available in field 0 (ODD field) of a video frame.

Note: This field is set to 11 (NTSC: Optional VBI for odd field) during a hard reset.

- '0000': minimum = 0 lines
- '1111': maximum = 15 lines

vbif1cnt <15:12> VBI Field 1 Count. This field is used to initialize a counter with the number of VBI lines available in field 1 (EVEN field) of a video frame.

Note: This field is set to 10 (NTSC: Optional VBI for even field) during a hard reset.

- '0000': minimum = 0 lines
- '1111': maximum = 15 lines

vbitasken <16> VBI range Task bit Enable. Enables the task bit of SAV/EAV codes to be used to determine which lines of the Vertical Blanking Interval contain valid VBI data.

- 0: IGNORE task bit and capture ALL data in VBI range in the manner programmed by the **VINCTL0**: **vbicap0** and **VINCTL1**: **vbicap1** fields
- 1: enable VBI data detection using task bit (Task B)

rpvaliden <24> RP byte Validation Enable. Enables validation of the RP byte of the SAV/EAV codes.

- 0: Do not check RP byte for valid hamming code. Process all tasks, fields, vertical and horizontal blanking bits in the RP byte of an SAV/EAV when encountered.
- 1: Check RP byte for valid hamming code.

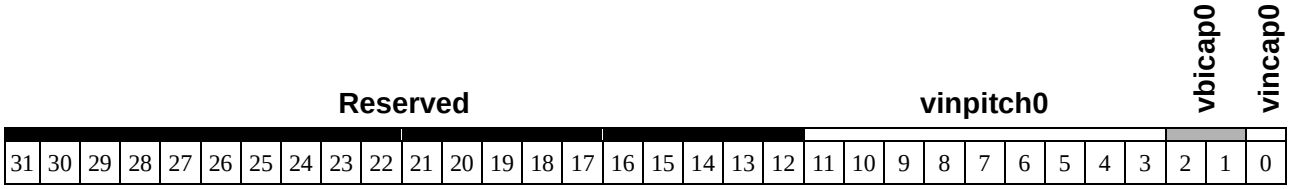
Reserved <31:25><23:17><7:1>

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

Address MGABASE1 + 3E00h (MEM)

Attributes WO, BYTE/WORD/DWORD, STATIC

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



vincap0 <0> Video Input Capture for window 0.

- 0: disable
- 1: enable

vbicap0 <2:1> VBI Capture for window 0. Enables and defines the type of VBI data to be captured for window 0.

- '00': ***no*** VBI captured
- '01': raw VBI captured
- '10': Reserved
- '11': sliced VBI captured

vinpitch0 <11:3> Video Input Pitch for window 0. **vinpitch0** field is mod 4. The incoming video is in YCbCr 4:2:2 which results in 4 pixels stored in every qword memory location. The actual line pitch is **vinpitch0** * 4. This allows up to 2048 pixels when **vinpitch0** = '000000000'.

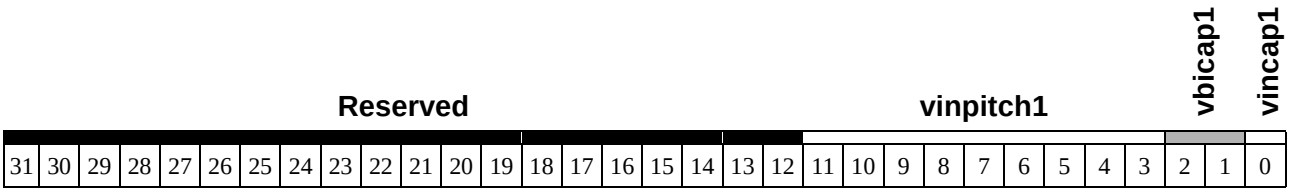
Reserved <31:12>

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

Address MGABASE1 + 3E04h (MEM)

Attributes WO, BYTE/WORD/DWORD, STATIC

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



vincap1 <0> Video Input Capture for window 1.

- 0: disable
- 1: enable

vbicap1 <2:1> VBI Capture for window 1. Enables and defines the type of VBI data to be captured for window 1.

- '00': ***no*** VBI capture
- '01': raw VBI captured
- '10': Reserved
- '11': sliced VBI captured

vinpitch1 <11:3> Video Input Pitch for window 1. The **vinpitch1** field is mod 4. The incoming video is in YCbCr 4:2:2 which results in 4 pixels stored in every qword memory location. The actual line pitch is **vinpitch1***4. This allows up to 2048 pixels when vinpitch1= '000000000'

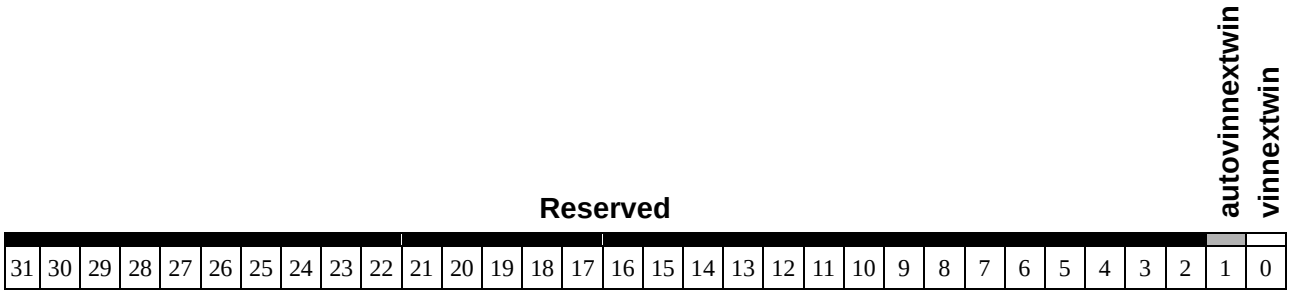
Reserved <31:12>

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

Address MGABASE1 + 3E18h (MEM)

Attributes WO, BYTE/WORD/DWORD, STATIC

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



vinnextwin <0> Video Input Next active Window trigger.

- 0: grab window 0 next
- 1: grab window 1 next

autovin nextwin <1> Automatic Next active Window trigger. Enables the video-in macro to continuously switch between window 0 and window 1. Since writing to the **vinnextwin** generates the initial trigger, the **vinnextwin** field ***must*** be written at the same time or after writing to this field.

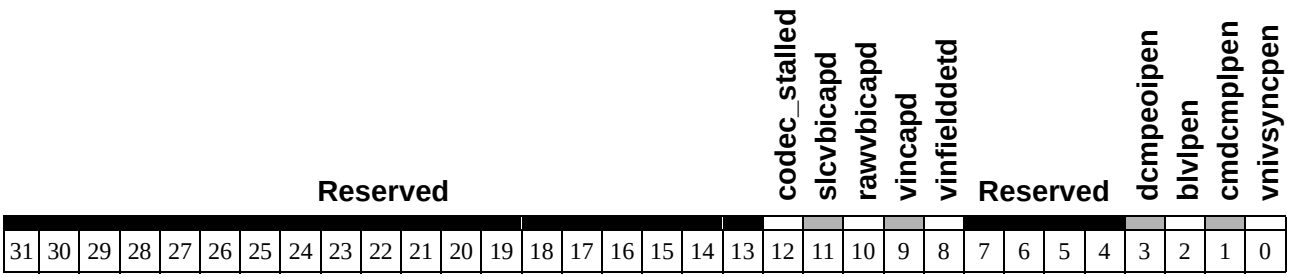
- 0: manual window trigger when **vinnextwin** is written to.
- 1: windows are continuously grabbed, automatically alternating between window 0 and window 1, starting with the window programmed in **vinnextwin**.

Reserved <31:2> Reserved. When writing to this register, the bits in this field ***must*** be set to '0'.

Address MGABASE1 + 3E30h (MEM)

Attributes RO, BYTE/WORD/DWORD, DYNAMIC

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



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

This bit is cleared through the **vinvsynciclr** bit (see **VICLEAR** on page 3-189) or upon a video in soft or a hard reset.

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

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

dcmpeoipen <3> Codec Decompression End Of Image Interrupt Pending. When set to a '1', the Codec Interface has detected an end of image marker in the decompression stream.

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

- 0: odd field
- 1: even field

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

This bit is cleared through the **vinvsynciclr** bit (see **VICLEAR** on page 3-189) or upon a video in soft or a hard reset.

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

This bit is cleared through the **vinvsynciclr** bit (see **VICLEAR** on page 3-189) or upon a video in soft or a hard reset.

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

This bit is cleared through the **vinvsynciclr** bit (see **VICLEAR** on page 3-189) or upon a video in soft or a hard reset.

codec _stalled <12>	Codec Stalled. Data transfer is suspended when we set the **codectransen** bit (within the **CODECCTL** register) to '0'. The Codec Interface will then assert the **codec_stalled** after it has finished transferring data into its 32 byte fifo. The **codec_stalled** bit is ***only*** valid and operational when performing data transfers (compression or decompression) and **codectransen** is set to '0'.
Reserved	**<7:4> <31:13>** Reserved. When writing to this register, the bits in these fields ***must*** be set to '0'.

Address MGABASE1 + 1E6Ch (MEM)

Attributes RO, DYNAMIC, DWORD

Reset Value unknown

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

wcodeaddr <31:8> Warp microcode Address. Specifies the address where the microcode can be found. The field **wcodeaddr** is 256 byte-aligned address. See 'Cache Operation' on page 6-16.

Reserved <7:0> Reserved. Reading will return '0'.

Address MGABASE1 + 1DC4h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b

wprgflag																walucfgflag								walustsflag							
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

walustsflag <7:0> WARP ALUs Status Flags.

bit	*flag*	*description*
<0>	C	Carry
<1>	V	Overflow
<2>	Z	Zero
<3>	N	Negative
<4>	X	Extended Flag
<5>	F	Current Flag
<6>	—	Reserved
<7>	—	Reserved

walucfgflag <15:8> WARP ALUs Configuration Flags.

bit	*flag*	*description*
<8>	L	Left edge sign
<9>	R	Right edge sign
<10>	S	Saturate Integer
<11>	U	Enable culling
<12>	T	Swap edge
<13>	M	GetMSB without ‘nnnnn’ subtract
<14>	H	Double dword result
<15>	—	Reserved

wprgflag <31:16> WARP Program Available Flags. These flags are available to the program. They are specified as flag (F16), … , flag (F31).

Bits 31 to 24 are *accumulated*. (See ‘Pipeline Operation’ on page 6-6.)

Address MGABASE1 + 1E64h (MEM)

Attributes R/W, DYNAMIC, DWORD

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b

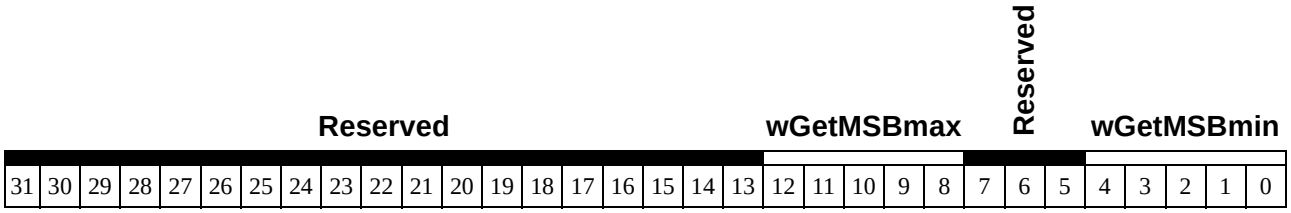
The **WFLAGNB** register is an alternate way to load the **WFLAG** register, bypassing the BFIFO. (See **WFLAG** for field descriptions.)

Note: The WARP flags are ***not stable*** when the WARP engine is running.

Address **MGABASE1** + 1DC8h (MEM)

Attributes WO, FIFO, STATIC, DWORD

Reset Value unknown



wGetMSBmin <4:0> GetMSB minimum value

wGetMSBmax <12:8> GetMSB maximum value

Note: These values are used by the GETMSB instruction to restrict the range of the results. Refer to the GETMSB instruction for more information.

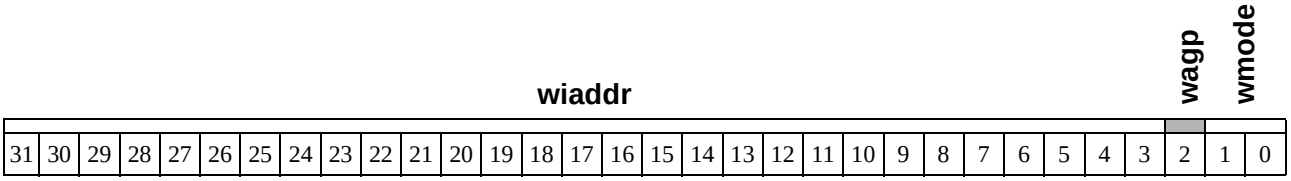
Reserved **<31:13> <7:5>**

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

Address MGABASE1 + 1DC0h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b



wmode <1:0> WARP Mode of operation.

wmode	*Operation*
'00'	Suspend
'01'	Resume
'10'	Jump
'11'	Start

- Suspend:

  In this mode, all execution in the WARP is stopped. Writing to the **wiaddr** and **wagp** field is ignored.
- Resume:

  In this mode the microcode is allowed to resume from where it was STOPped (suspended or STOP (instruction)). Writing to the **wiaddr** and **wagp** field is ignored. This mode ***must*** be set ***only*** when the engine is idle.
- Jump:

  In this mode, the microcode starts executing from where the **wiaddr** field specifies it. A reset is ***not*** performed in the engine; the software ***must*** be aware of the current state of the engine (pending store back) and must check the **wbusy** bit. This mode ***must*** be set ***only*** when the engine is idle.
- Start:

  This mode operates like Jump mode, *but* it resets the engine.

wagp <2> Specifies what host cycle should be used to load the microcode from the system memory when mastering is enabled (see WMISC).

wagp	*cycle*
'0'	PCI
'1'	AGP

wiaddr <31:3>

WARP Instruction Address.

- Bits 31 to 3 represent the current address of the microcode to be fetched.
- The WARP engine can only modify bits 17 to 3.
- When caching is disabled, bit 31 to 11 must be set to ‘0’ (the microcode must reside within the 2 kbyte instruction memory).

➽ ***Note***: The microcode has to be aligned on a 256 byte boundary.

Address MGABASE1 + 1E60h (MEM)

Attributes R/W, DYNAMIC, DWORD

Reset Value 0000 0000 0000 0000 0000 0000 0000 0000b

The **WIADDRNB** register is simply an alternate means of loading the **WIADDR** register, bypassing the BFIFO. (See **WIADDR** for the field descriptions).

Note: Bits 2 to 0 are write only, reading will give '0'.

Note: The WARP Instruction Address register is ***not stable*** when the WARP engine is running.

Address MGABASE1 + 1E68h (MEM)

Attributes WO, STATIC, DWORD

Reset Value unknown

Reserved																						wimemaddr							
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

wimemaddr <7:0> WARP Instruction Memory Address. This register is used to address the Instruction Memory to load the microcode. This register is incremented every 2 writes to the WIMEMDATA space. When this address increments beyond the last Instruction Memory location, it will wrap around to location 0. Writing to this register will reset the modulo 2 counter to 0. Reading to the WIMEMDATA space will ***not*** increment **WIMEMADDR**

Reserved <31:8> Reserved. When writing to this register, the bits in this field ***must*** be set to '0'.

➾ ***Note:*** The WARP engine ***must*** be idle before writing to this register.

Address MGABASE1 + 2000h to MGABASE1 + 207Fh

Attributes R/W, STATIC, DWORD

Reset Value unknown

wimemdata

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

wimemdata <31:0>

WARP Instruction Memory Data. This range is used to load data in the WARP Instruction Memory. Since the Instruction Memory is 64 bits wide, two accesses are required to this range in order to write one complete location. The address in the WARP Instruction Memory to be written is determined by the value in the **WIMEMADDR** register. When **wucodecache** = '0' there is ***no*** caching.

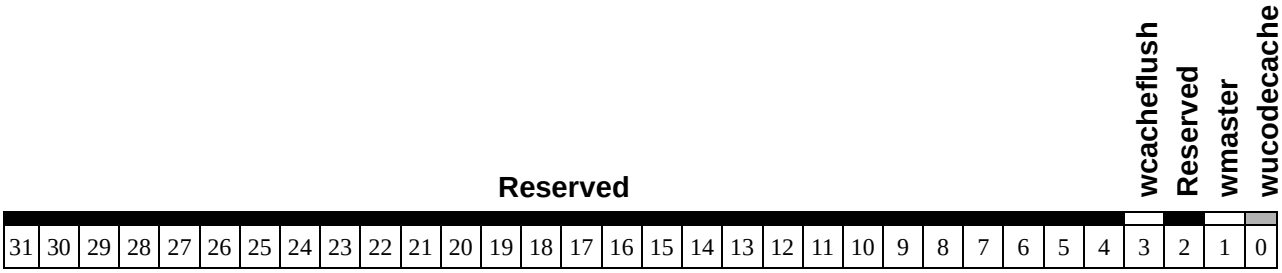
To read one complete Instruction memory location (64 bits), **WIMEMADDR** must be written with the location to be read. This data can then be read at MGABASE1+2000h for DW0, and at MGABASE1+2004h for DW1.

Note: The **wmaster** field in the **WMISC** register must be set '0' when writing to this range (**MGABASE1** + 2000h to **MGABASE1** + 207Fh).

Address **MGABASE1** + 1E70 (MEM)

Attributes R/W, DWORD, STATIC

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



wucode cache <0> WARP microcode Caching Enable. When set to '1', microcode caching is enabled. It is ***not*** necessary to load the microcode via the **WIMEMDATA** space before starting the WARP engine; the caching system will fetch the microcode when needed. When at '0', caching of the microcode will ***not*** be performed; the entire microcode must be loaded into the Instruction Memory via the **WIMEMDATA** space before starting the engine.

wmaster <1> When at '1', this bit indicates that microcode loading will be done through bus mastering; when at '0', microcode loading will be handled by interrupt. (See 'Cache Operation' on page 6-16). This bit is only used when **wucodecache** = '1'.

wcacheflush <3> WARP microcode Cache Flush. When this bit is set to '1' the WARP cache tags are reset: the contents of the Instruction memory are invalidated. Reading will give '0's.

Reserved **<31:4> <2>**

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

Address MGABASE1 + 2D00h (MEM) (WR0)
…
MGABASE1 + 2DFCh (MEM) (WR63)

Attributes WO, DB, DYNAMIC, DWORD, FIFO

Reset Value unknown

wr

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

These are the 64 multi-purpose WARP registers (WR0, WR1, …WR63). They can represent an IEEE single precision floating point number, a fixed point 32-bit integer, two 16-bit words or four bytes. It is up to the software to define what to put in these registers and how the microcode will interpret them.

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

Attributes WO, FIFO, STATIC, DWORD

Reset Value ???? ???? ???? ???? ???? ???? ??11 1111b

Reserved																								wvrtxsz					
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

wvrtxsz <5:0> WARP Vertex Size. This is the number of registers (minus one) to be written sequentially in the **WR** register before switching to the next bank. This information is used to align the vertices in the register bank boundaries and is only used by the ACCEPT command. When **WVRTXSZ** + 1 registers have been transferred to the **WR** register, the pointer will jump to the beginning of the next bank.

This field is also used by the Setup DMA Channel to determine the dword length of each DMA vertex fixed length setup list entry (setupmod = ‘00’).

Reserved <31:6> Reserved. When writing to this register, the bits in this field *must* be set to ‘0’.

Address **MGABASE1** + 1CB0h (MEM)

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value unknown

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

xdst <15:0> The x-coordinate of destination address. The **xdst** field contains the running x-coordinate of the destination address. It is a 16-bit signed value in two's complement notation.

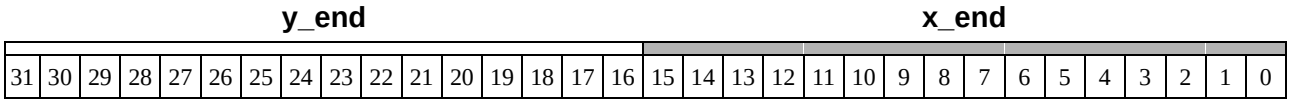
- Before starting a vector draw, **xdst** must be loaded with the x-coordinate of the starting point of the vector. At the end of a vector, **xdst** contains the address of the last pixel of the vector. This can also be done by accessing the **XYSTRT** register.
- This register does ***not*** require initialization for polyline operations.
- For BLITs, this register is automatically loaded from **fxleft** (see **FXLEFT** on page 3-114) and **fxright** (see **FXRIGHT** on page 3-115), 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 3-75.)

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.

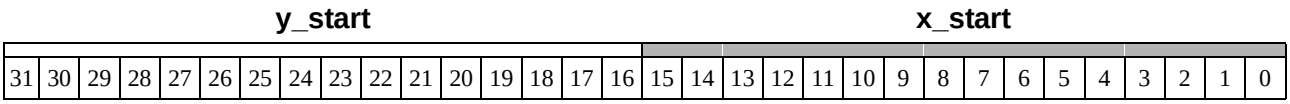
When **XYEND** is written, the following registers are affected:

- **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)

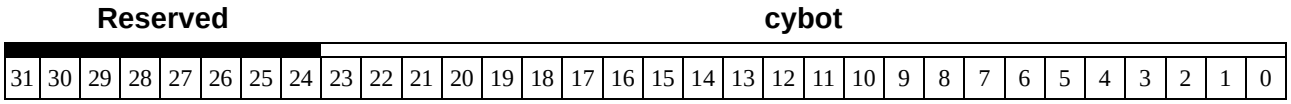
x_start <15:0> The **x_start** field contains the x-coordinate of the starting point of the vector. It is a 16-bit signed value in two's complement notation.

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 3-215). 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).

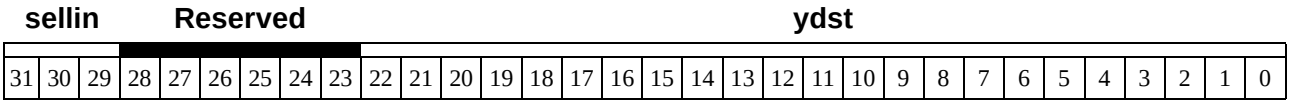
Note: Clipping can be disabled by the **clipdis** bit in **DWGCTL** without changing **cybot**.

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

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

Attributes WO, FIFO, DYNAMIC, DWORD

Reset Value unknown



ydst <22:0>

The y destination. The **ydst** field contains the current y-coordinate (in pixels) of the destination address as a signed value in two's complement notation. Two formats are supported: linear format and xy format. The current format is selected by **ylin** (see **PITCH** on page 3-129).

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 4096. Pitch is also a multiple of 32.

- Before starting a vector draw, **ydst** must be loaded with the y-coordinate of the starting point of the vector. This can be done by accessing the **XYSTRT** register. This register does ***not*** require initialization for polyline operations.
- Before starting a BLIT, **ydst** must be loaded with the y-coordinate of the starting corner of the destination rectangle.
- For trapezoids, this register must be loaded with the y-coordinate of the first scanned line of the trapezoid.
- To load the texture color palette, **ydst** must be loaded with the position in the color palette (0 to 255) at which the texture color palette will begin loading.

Note: When **ylin** = 0 (even when the y-coordinate range is 16-bit signed), the sign must be extended until bit 22.

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** (y-coordinate). If no linearization occurs, then those bits must be initialized correctly if one of the above-mentioned functions is to be used.

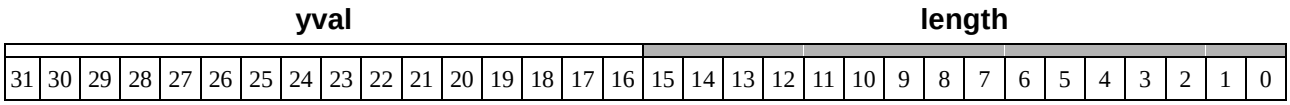
Reserved <28:23>

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 3-118.

yval <31:16>

The y destination value. See the **YDST** register on page 3-215. 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.

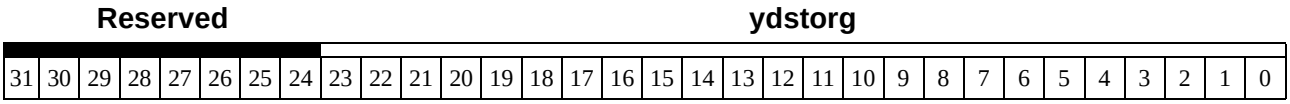
ydst<22:0> <= sign extension (**yval**<31:16>)

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 or 64 according to the table below, due to a restriction involving block mode. See 'Constant Shaded Trapezoids / Rectangle Fills' on page 4-40. See page 3-100 for additional restrictions that apply to block mode (**atype** = BLK).

pwidth	*value:*
PW8	64
PW16	32
PW24	64
PW32	32

Note: It is ***recommended*** to use **DSTORG** or **SRCORG** in place of **YDSTORG**.

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 3-215). 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:

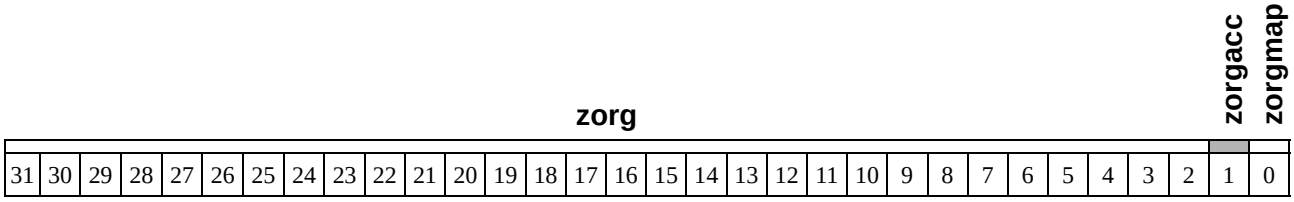
cytop = (top line number) × **PITCH** + **YDSTORG**

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

Note: Clipping can be disabled by the **clipdis** bit in **DWGCTL** without changing **cytop**.

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



zorgmap <0>
Z-depth Origin Map. This field indicates the map location.

- 0: depth buffer is in the frame buffer
- 1: depth buffer is in the system memory

zorgacc <1>
Z-depth Origin Access type. This field specifies the mode used to access the map.

- 0: PCI access
- 1: AGP access

Note: This field is ***not*** considered if, the depth buffer resides in the frame buffer space.

zorg <31:2>
Z-depth origin. The **zorg** field is a 30-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 dword 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 128 (the seven LSBs = 0).

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

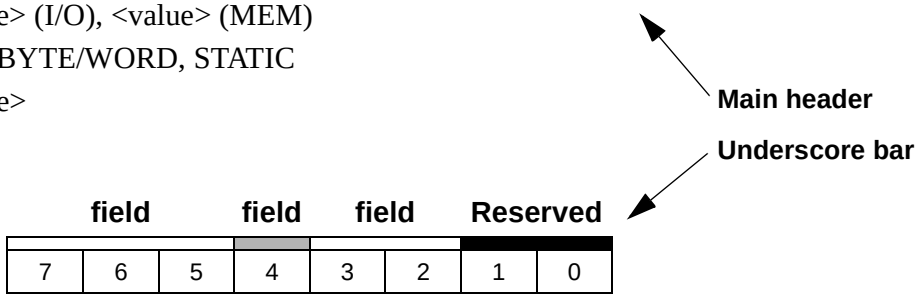
3.2 VGA Mode Register Descriptions

3.2.1 VGA Mode Register Descriptions

The MGA-G200 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.

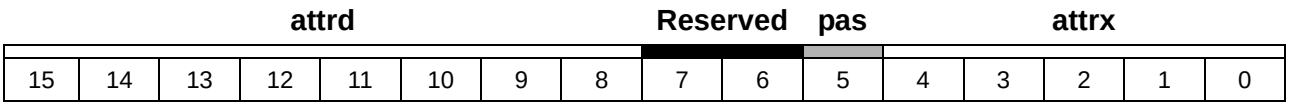
Reset Value

- `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



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.

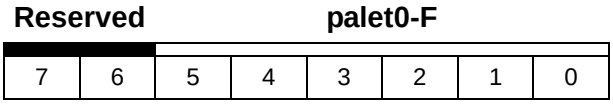
Retrieve or write the contents of the register pointed to by the **attrx** field.

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



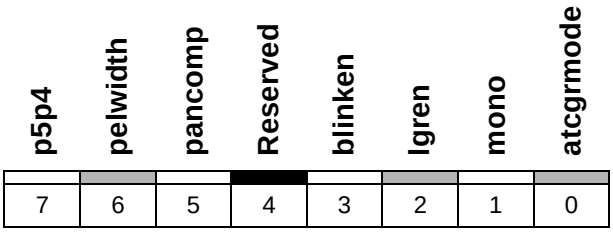
palet0-F <5:0> Internal palette data. VGA.

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 modeis 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 3-234.

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

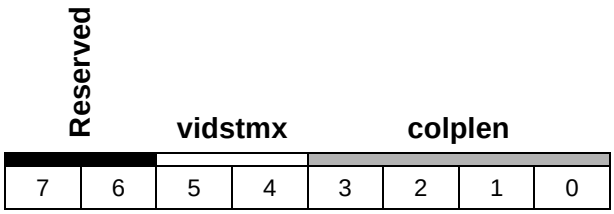
ovscol <7:0>

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 3-293.

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.

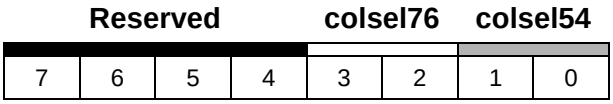
This 4-bit value specifies the number of picture elements to shift the video data horizontally to the left, according to the following table (values 9 to 15 are reserved):

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>

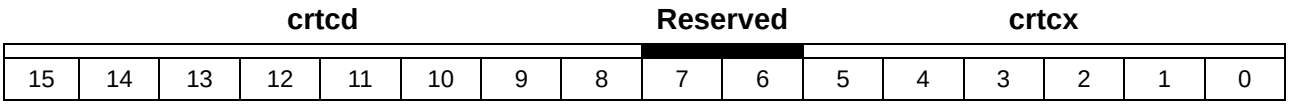
Flush the cache. Writes to this register will flush the cache. For additional details, refer to 'Direct Access Read Cache' on page 4-5.

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 - 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>

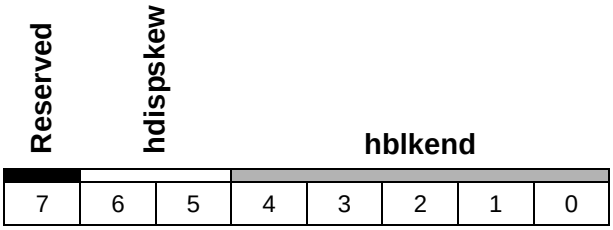
Start horizontal blanking. VGA/MGA.

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>

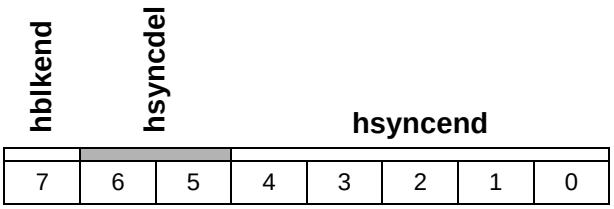
Start horizontal retrace pulse. VGA/MGA.

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 3-241.

Index **crtcx** = 06h

Reset Value 0000 0000b

vtotal

7	6	5	4	3	2	1	0

vtotal <7:0>

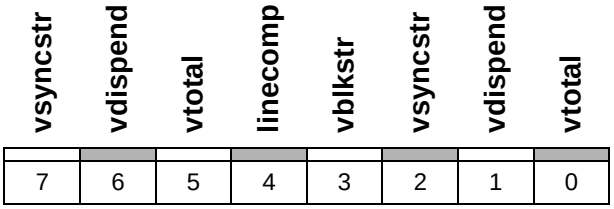
Vertical total. VGA/MGA.

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 3-244.

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 3-256.

vsyncstr <2> Vertical retrace start bit 8. VGA/MGA.
Contains bit 8 of the Vertical Retrace Start. See the **CRTC10** register on page 3-254.

vblkstr <3> Start vertical blank bit 8. VGA/MGA.
Contains bit 8 of the Start Vertical Blank. See the **CRTC15** register on page 3-259.

linecomp <4> Line compare bit 8. VGA/MGA.
Line compare bit 8. See the **CRTC18** register on page 3-265. This bit is not write-protected by **crtcprotect** (**CRTC11**<7>).

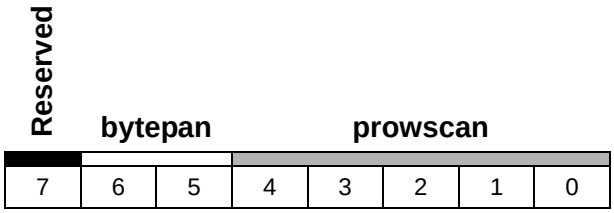
vtotal <5> Vertical total bit 9. VGA/MGA.
Contains bit 9 of the Vertical Total. See the **CRTC6** register on page 3-244.

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 3-256.

vsyncstr <7> Vertical retrace start bit 9. VGA/MGA.
Contains bit 9 of the Vertical Retrace Start. See the **CRTC10** register on page 3-254.

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

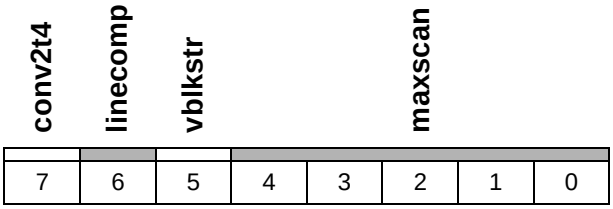
bytepan <6:5> Byte panning control. VGA/MGA.

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 3-259.

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

Bit 9 of the Line Compare register. See the **CRTC18** register on page 3-265.

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

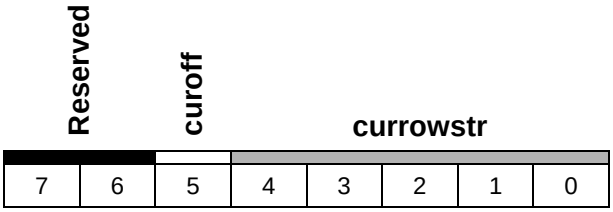
Controls the row scan counter clock and the time when the start address latch loads a new memory address:

- **conv2t4** (**CRTC9**<7>) = 0: HS
- **conv2t4** = 1: HS/2

This feature allows a low resolution mode (200 lines, for example) to display as 400 lines on a display monitor. This lowers the requirements for sync capability of the monitor.

Index **crtcx** = 0Ah

Reset Value 0000 0000b



currowstr <4:0> Row scan cursor begins. VGA.

These bits specify the row scan of a character line where the cursor is to begin.

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

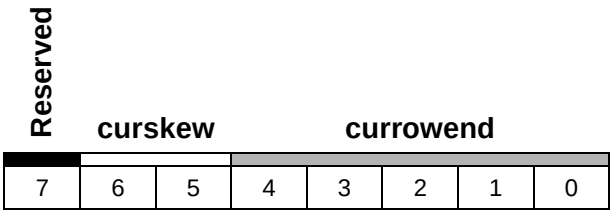
curoff <5> Cursor off. VGA.

- Logical ‘1’: turn off the cursor
- Logical ‘0’: turn on the cursor

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

Index **crtcx** = 0Bh

Reset Value 0000 0000b



currowend <4:0> Row scan cursor ends. VGA.

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

curskew <6:5> Cursor skew control. VGA.

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

curskew	*Skew*
‘00’	0 additional character delays
‘01’	Move the cursor right by 1 character clock
‘10’	Move the cursor right by 2 character clocks
‘11’	Move the cursor right by 3 character clocks

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

Index **crtcx** = 0Ch

Reset Value 0000 0000b

startadd

7	6	5	4	3	2	1	0

startadd <7:0>

Start address, bits<15:8>. VGA/MGA.

These are the middle eight bits of the start address. The 21-bit value from the **startadd** (**CRTCEXT0**<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 4-65 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 3-250.

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 3-252.

Index **crtcx** = 10h

Reset Value 0000 0000b

vsyncstr

7	6	5	4	3	2	1	0

vsyncstr <7:0>

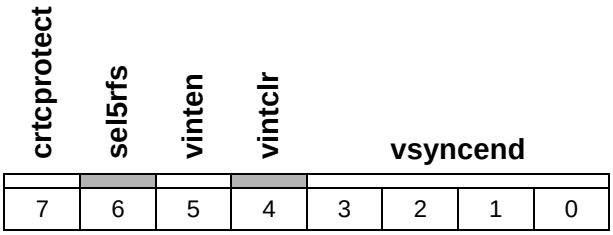
Vertical retrace start bits 7 to 0. VGA/MGA.

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

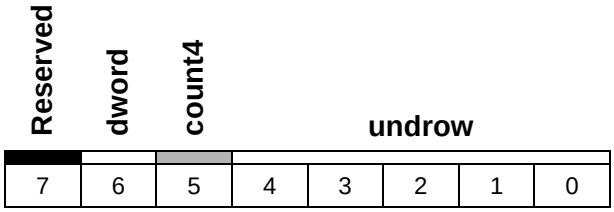
offset <7:0> Logical line width of the screen. VGA/MGA.

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 4-65 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’.

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

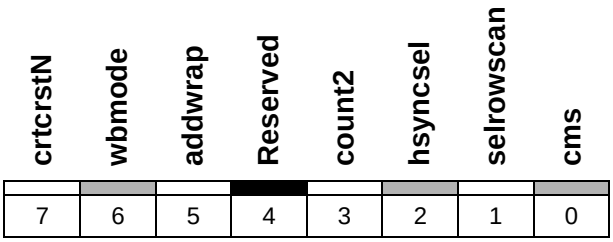
vblkend <7:0>

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.

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

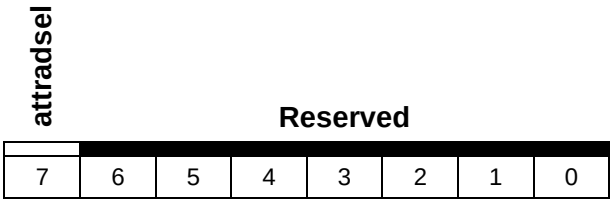
cpudata <7:0>

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’ (The **chain4** (**SEQ4** <3>)and **gcoddevmd** (**GCTL5** <4>) fields must be ‘0’). This register contains color compare data in Read Mode 1.

Index **crtcx** = 24h

Reset Value 0000 0000b



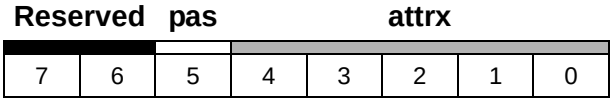
attradsel <7> Attributes address/data select. VGA.

- 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.

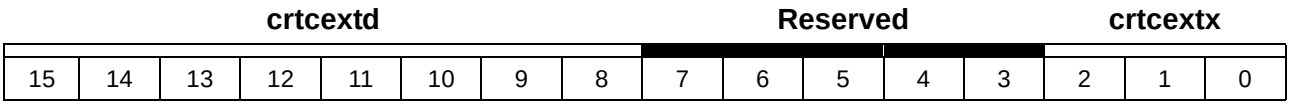
Reserved <7:6> Reserved. When writing to this register, the bits in this field ***must*** be set to ’0’.

•See the **ATTR** register on page 3-226.

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

Attributes R/W, BYTE/WORD, STATIC

Reset Value nnnn nnnn 0000 0000b



crtcextx <2:0> CRTC extension index register.

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

Register Name	*Mnemonic*	**crtcextx** *address*
Address Generator Extensions	**CRTCEXT0**	00h
Horizontal Counter Extensions	**CRTCEXT1**	01h
Vertical Counter Extensions	**CRTCEXT2**	02h
Miscellaneous	**CRTCEXT3**	03h
Memory Page register	**CRTCEXT4**	04h
Horizontal Video Half Count	**CRTCEXT5**	05h
Priority Request Control	**CRTCEXT6**	06h
Request for Control	**CRTCEXT7**	07h

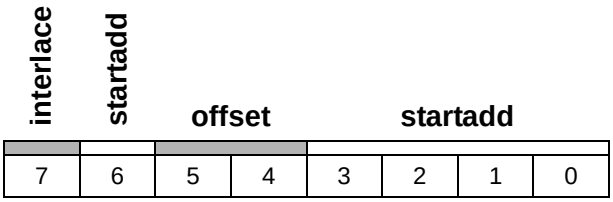
crtcextd <15:8> CRTC extension data register.

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 20, 19, 18, 17, and 16.

These are the five most significant bits of the start address. See the **CRTCC** register on page 3-250.

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 3-257.

startadd <6> Start address bits 20.

This is the most significant bit of the start address. See the **CRTCC** register on page 3-250.

interlace <7> Interlace enable.

Indicates if interlace mode is enabled.

- 0: Not in interlace mode.
- 1: Interlace mode.

Index **crtcextx** = 01h

Reset Value 0000 0000b

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

htotal <0>
Horizontal total bit 8.

This is the most significant bit of the **htotal** (horizontal total) register. See the **CRTC0** register on page 3-238.

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 3-240.

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 3-242.

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**).

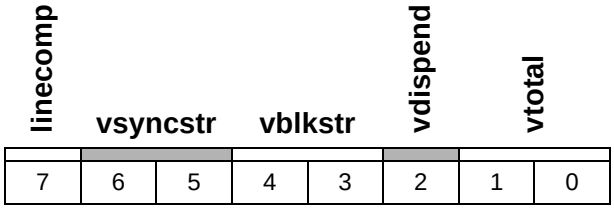
Bit 6 of the End Horizontal Blanking value. See the **CRTC3** register on page 3-241.

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 3-244.

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 3-256.

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 3-259.

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 3-254.

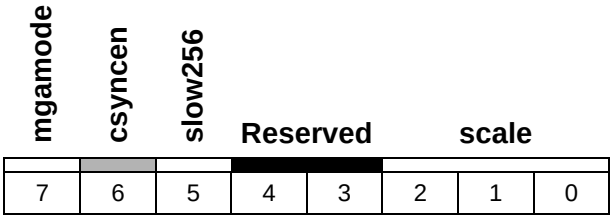
linecomp <7>

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 3-265.

Index **crtcextx** = 03h

Reset Value 0000 0000b



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

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

slow256 <5> 256 color mode acceleration disable.

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

csyncen <6> Composite Sync Enable.

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

Reserved <4:3>

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

Index **crtcextx** = 04h

Reset Value 0000 0000b

page

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

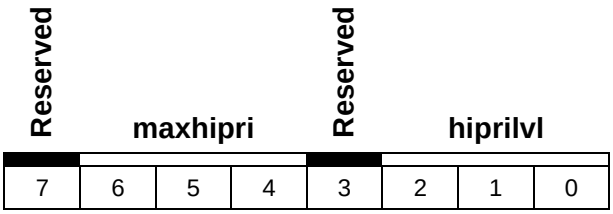
hvidmid <7:0> Horizontal video half count.

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:

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

Index **crtcextx** = 06h

Reset Value 0000 0000b



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

hiprilvl	*high priority request level*
‘000’	1
‘001’	2
‘010’	3
‘011’	4
‘100’	5
‘101’	6
‘110’	7
‘111’	8

maxhipri <6:4> Maximum High Priority requests. This register indicates the minimum number of high priority requests to be made.

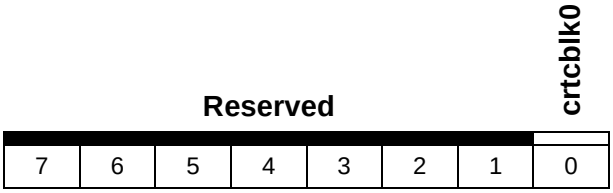
maxhipri	*maximum high priority request level*
‘000’	1
‘001’	2
‘010’	3
‘011’	4
‘100’	5
‘101’	6
‘110’	7
‘111’	8

Reserved **<3><7>**

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

Index **crtcextx** = 07h

Reset Value 0000 0000b



crtcblk0 <0> This field is used to enable a bandwidth-saving mode that blocks the CRTC requester when video window 0 is displayed.

- 0: normal mode
- 1: block request mode

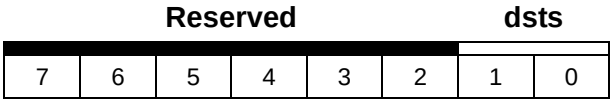
Note: This field has effect ***only*** when **HZOOM** equals ’00’.

Reserved <7:1> Reserved. When writing to this register, the bits in this field ***must*** be set to ’0’.

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

Attributes RO, BYTE, STATIC

Reset Value 0000 0000b



dsts <1:0>

This port returns the last access cycle to the palette.

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

Reserved <7:2>

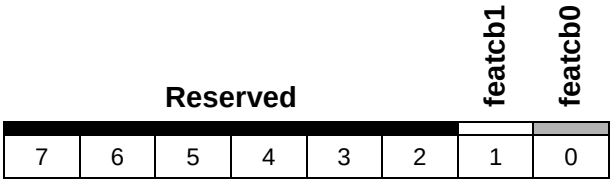
This field returns zeroes when read.

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 0000`b



featcb0 <0> Feature control bit 0. VGA. General read/write bit.

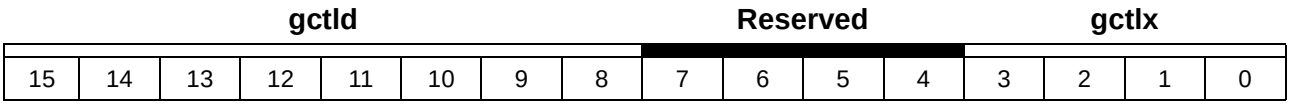
featcb1 <1> Feature control bit 1. VGA. General read/write bit.

Reserved <7:2> 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.

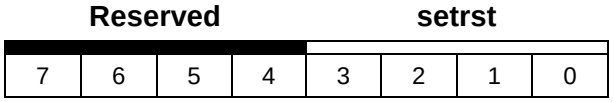
Retrieve or write the contents of the register pointed to by the **gctlx** field.

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:

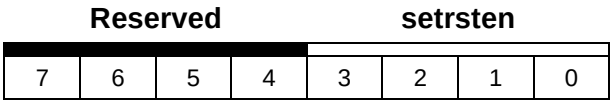
- 1: Set the byte, assuming the corresponding set/reset enable bit is ‘1’.
- 0: Reset the byte, assuming the corresponding set/reset enable bit is ‘0’.

➽ ***Note:*** 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 this field ***must*** be set to ’0’.

Index **gctlx** = 01h

Reset Value 0000 0000b



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

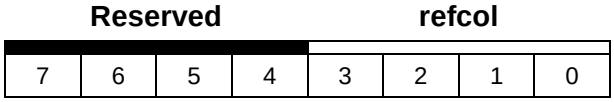
When a set/reset plane is enabled (the corresponding bit is '1') and the write mode is 0 (**wrmode** (GCTL5<1:0>) = 00), the value written to all eight bits of that plane represents the contents of the set/reset register. Otherwise, the rotated CPU data is used.

Note: This register has no effect when ***not*** in Write Mode 0.

Reserved <7:4> Reserved. When writing to this register, the bits in this field ***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 3-290) register will be tested.

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

Index **gctlx** = 03h

Reset Value 0000 0000b

Reserved			funsel		rot		
7	6	5	4	3	2	1	0

rot <2:0>

Data rotate count bits 2 to 0. VGA.

These bits represent a binary encoded value of the number of positions to right-rotate the host data before writing in Mode 0 (**wrmode** (**GCTL5**<1:0>) = 00).

The rotated data is also used as a mask together with the **GCTL8** (‘Bit Mask’, page 3-291) 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).

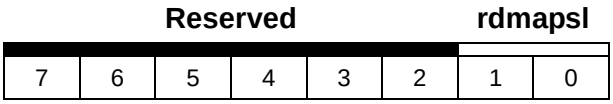
funsel	*Function*
‘00’	Source unmodified
‘01’	Source AND latched data
‘10’	Source OR latched data
‘11’	Source XOR latched data

Reserved <7:5>

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

Index **gctlx** = 04h

Reset Value 0000 0000b



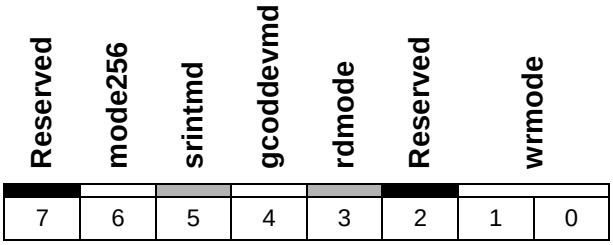
rdmapsl <1:0> Read map select. VGA.

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 this field ***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 IBLU 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 3-291) 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.

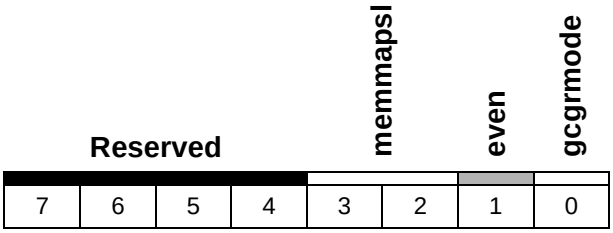
- 0: The loading of the shift registers is controlled by the **srintmd** field.
- 1: The shift registers are loaded in a manner which supports 256-color mode.

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 3-294).

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

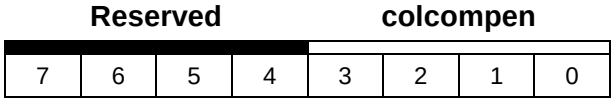
These bits select where the video memory is mapped, as shown below:

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 this field ***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 this field ***must*** be set to ’0’.

Index **gctlx** = 08h

Reset Value 0000 0000b

wrmask

7	6	5	4	3	2	1	0

wrmask <7:0>

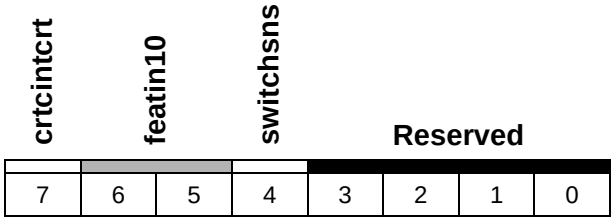
Data write mask for pixels 7 to 0. VGA.

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.

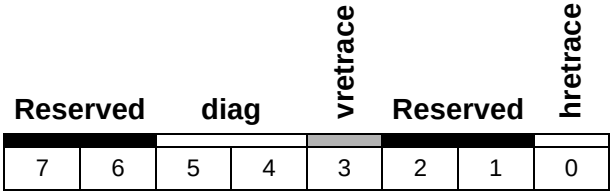
- 0: Vertical retrace interrupt is cleared.
- 1: Vertical retrace interrupt is pending.

Reserved <3:0> Reserved. When writing to this register, the bits in this field ***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

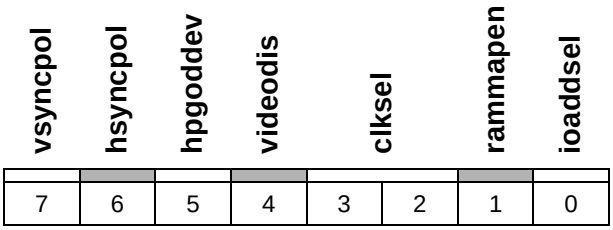
Reserved <2:1> <7:6>

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/MGA.

- 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>

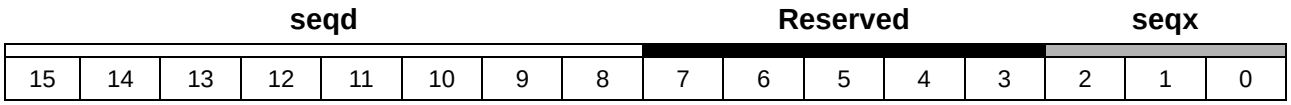
Vertical sync polarity. VGA/MGA.

• 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.

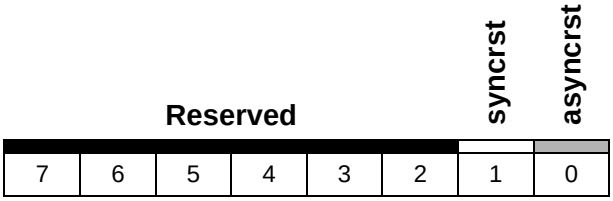
seqd <15:8> Sequencer data register.

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 this field ***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.

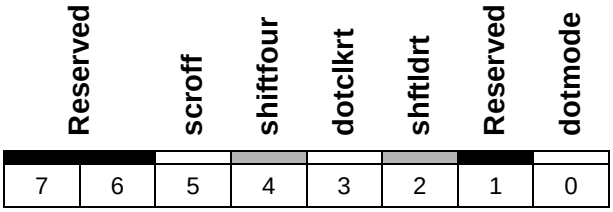
syncrst <1> Synchronous reset. VGA.

- 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-G200 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 this field ***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.

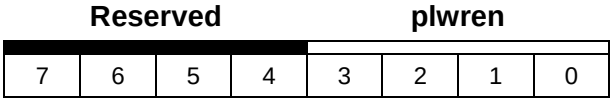
- 0: Normal video operation.
- 1: Turns off the video, and maximum memory bandwidth is assigned to the system. The display is blanked, however, all sync pulses are generated normally.

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 this field ***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.

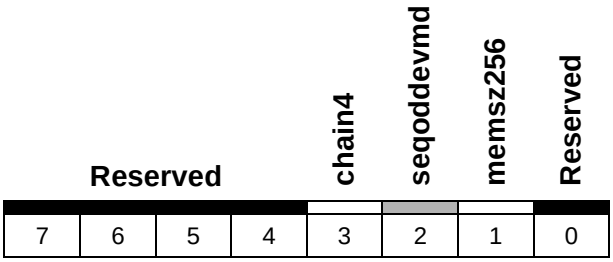
These bits are used for alpha character generation when the character's attribute bit 3 is '1', according to the following table:

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 this field ***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

3.3 DAC Registers

3.3.1 DAC Register Descriptions

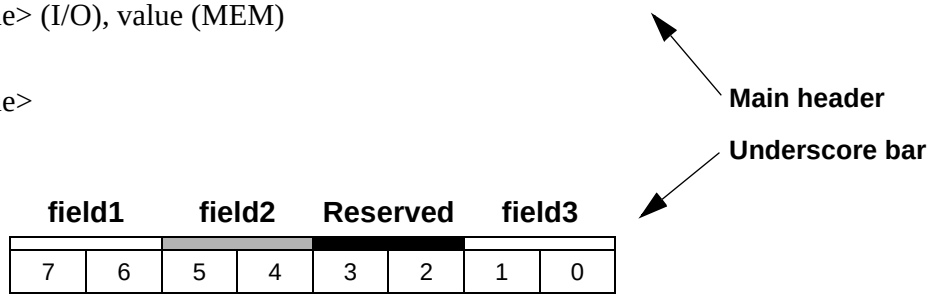
The MGA-G200 DAC register descriptions contain a (gray single-underlined) main header which indicates the register's name and mnemonic. Below the main header, the memory address or index, attributes, and reset value are indicated. Next, an illustration of the register identifies the bit fields, which are then described in detail below the illustration. The reserved bit fields are underscored by black bars, and all other fields are delimited by alternating white and gray bars.

Sample DAC Register Description **SAMPLE_DAC**

Address <value> (I/O), value (MEM)

Attributes R/W

Reset Value <value>



field1 <7:6> Field 1. Detailed description of the **field1** field of the **SAMPLE_DAC** register, which comprises bits 7 to 6. ***Font and case changes within the text indicate a register or field.***

field2 <5:3> Field 2. Detailed description of the **field2** field of **SAMPLE_DAC**, which comprises bits 5 to 3.

field3 <1:0> Field 3. Detailed description of the **field3** field of **SAMPLE_DAC**, which comprises bits 1 to 0.

Reserved<2> Reserved. When writing to this register, this field must be set to '0'. (Reserved registers always appear at the end of a register description.)

Address

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

Index

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

Attributes

The DAC register attributes are:

- RO: There are no writable bits.
- WO: There are no readable bits.
- R/W: The state of the written bits can be read.
- BYTE: 8-bit access to the register is possible.
- WORD: 16-bit access to the register is possible.
- DWORD: 32-bit access to the register is possible.

Reset Value

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

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

Address **CURPOSXL MGABASE1** + 3C0Ch (MEM)
CURPOSXH MGABASE1 + 3C0Dh (MEM)
CURPOSYL MGABASE1 + 3C0Eh (MEM)
CURPOSYH MGABASE1 + 3C0Fh (MEM)

Attributes R/W, BYTE, WORD, DWORD

Reset Value unknown

Reserved	curposy	Reserved	curposx

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

curposx <11:0> X Cursor position. Determines the position of the last column of the hardware cursor on the display screen.

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

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

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

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

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

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

curposy <27:16> Y Cursor position. Determines the position of the last row of the hardware cursor on the display screen.

Reserved **<15:12> <31:28>**

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

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

Attributes R/W, BYTE

Reset Value unknown

paldata

7	6	5	4	3	2	1	0

paldata <7:0>

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

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

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

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

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

Address 03C7h (I/O, W), **MGABASE1** + 3C03h (MEM, R/W)

Attributes BYTE

Reset Value unknown

palrdadd

7	6	5	4	3	2	1	0

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

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

Address 03C8h (I/O), **MGABASE1** + 3C00h (MEM)

Attributes R/W, BYTE

Reset Value unknown

palwtadd

7	6	5	4	3	2	1	0

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

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

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

Address 03C6h (I/O), **MGABASE1** + 3C02h (MEM)

Attributes R/W, BYTE

Reset Value 1111 1111h

pixrdmsk

7	6	5	4	3	2	1	0

pixrdmsk <7:0> Pixel read mask. The pixel read mask register is used to enable or disable a bit plane from addressing the palette RAM. This mask is used in all modes which access the palette RAM (not just the 8 bit/pixel modes).

Each palette address bit is logically ANDed with the corresponding bit from the read mask register before going to the palette RAM.

Note: Direct pixels (pixels that do not go through the palette RAM) are not masked in any mode.

Address MGABASE1 + 3C0Ah (MEM)

Attributes R/W, BYTE

Reset Value unknown

indd

7	6	5	4	3	2	1	0

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

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

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

indd *Address*	*Register Addressed*	*Mnemonic*
30h-37h	Reserved	—
38h	Zoom Control	**XZOOMCTRL**
39h	Reserved	—
3Ah	Sense Test	**XSENSETEST**
3Bh	Reserved	—
3Ch	CRC Remainder Low	**XCRCREML**
3Dh	CRC Remainder High	**XCRCREMH**
3Eh	CRC Bit Select	**XCRCBITSEL**
3Fh	Reserved	—
40h	Color Key Mask	**XCOLMSK**
41h	Reserved	—
42h	Color Key	**XCOLKEY**
43h	Reserved	—
44h	PIXPLL M Value Set A	**XPIXPLLAM**
45h	PIXPLL N Value Set A	**XPIXPLLAN**
46h	PIXPLL P Value Set A	**XPIXPLLAP**
47h	Reserved	—
48h	PIXPLL M Value Set B	**XPIXPLLBM**
49h	PIXPLL N Value Set B	**XPIXPLLBN**
4Ah	PIXPLL P Value Set B	**XPIXPLLBP**
4Bh	Reserved	—
4Ch	PIXPLL M Value Set C	**XPIXPLLCM**
4Dh	PIXPLL N Value Set C	**XPIXPLLCN**
4Eh	PIXPLL P Value Set C	**XPIXPLLCP**
4Fh	PIXPLL Status	**XPIXPLLSTAT**
50h	Reserved	—
51h	KEYING Operating Mode	**XKEYOPMODE**
52h	Color Mask 0 Red	**XCOLMSK0RED**
53h	Color Mask 0 Green	**XCOLMSK0GREEN**
54h	Color Mask 0 Blue	**XCOLMSK0BLUE**
55h	Color Key 0 Red	**XCOLKEY0RED**
56h	Color Key 0 Green	**XCOLKEY0GREEN**
57h	Color Key 0 Blue	**XCOLKEY0BLUE**
58h-5Fh	Reserved	—
60h	Cursor Color 3 Red	**XCURCOL3RED**
61h	Cursor Color 3 Green	**XCURCOL3GREEN**
62h	Cursor Color 3 Blue	**XCURCOL3BLUE**
63h	Cursor Color 4 Red	**XCURCOL4RED**
64h	Cursor Color 4 Green	**XCURCOL4GREEN**
65h	Cursor Color 4 Blue	**XCURCOL4BLUE**
66h	Cursor Color 5 Red	**XCURCOL5RED**
67h	Cursor Color 5 Green	**XCURCOL5GREEN**

indd *Address*	*Register Addressed*	*Mnemonic*
68h	Cursor Color 5 Blue	**XCURCOL5BLUE**
69h	Cursor Color 6 Red	**XCURCOL6RED**
6Ah	Cursor Color 6 Green	**XCURCOL6GREEN**
6Bh	Cursor Color 6 Blue	**XCURCOL6BLUE**
6Ch	Cursor Color 7 Red	**XCURCOL7RED**
6Dh	Cursor Color 7 Green	**XCURCOL7GREEN**
6Eh	Cursor Color 7 Blue	**XCURCOL7BLUE**
6Fh	Cursor Color 8 Red	**XCURCOL8RED**
70h	Cursor Color 8 Green	**XCURCOL8GREEN**
71h	Cursor Color 8 Blue	**XCURCOL8BLUE**
72h	Cursor Color 9 Red	**XCURCOL9RED**
73h	Cursor Color 9 Green	**XCURCOL9GREEN**
74h	Cursor Color 9 Blue	**XCURCOL9BLUE**
75h	Cursor Color 10 Red	**XCURCOL10RED**
76h	Cursor Color 10 Green	**XCURCOL10GREEN**
77h	Cursor Color 10 Blue	**XCURCOL10BLUE**
78h	Cursor Color 11 Red	**XCURCOL11RED**
79h	Cursor Color 11 Green	**XCURCOL11GREEN**
7Ah	Cursor Color 11 Blue	**XCURCOL11BLUE**
7Bh	Cursor Color 12 Red	**XCURCOL12RED**
7Ch	Cursor Color 12 Green	**XCURCOL12GREEN**
7Dh	Cursor Color 12 Blue	**XCURCOL12BLUE**
7Eh	Cursor Color 13 Red	**XCURCOL13RED**
7Fh	Cursor Color 13 Green	**XCURCOL13GREEN**
80h	Cursor Color 13 Blue	**XCURCOL13BLUE**
81h	Cursor Color 14 Red	**XCURCOL14RED**
82h	Cursor Color 14 Green	**XCURCOL14GREEN**
83h	Cursor Color 14 Blue	**XCURCOL14BLUE**
84h	Cursor Color 15 Red	**XCURCOL15RED**
85h	Cursor Color 15 Green	**XCURCOL15GREEN**
86h	Cursor Color 15 Blue	**XCURCOL15BLUE**

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

colkey

7	6	5	4	3	2	1	0

colkey <7:0>

Color key bits for alpha overlay. The color key is ***only*** used to perform color keying between graphics and the alpha overlay buffer in 32 bits/pixel single frame buffer mode (**depth** = ‘100’). The equation is:

```
if (COLMSK AND ALPHA == COLKEY)     ;show graphic stream
else                                ;show the overlay color LUT(ALPHA))
```

✒ ***Note:*** The **depth** field is located in the **XMULCTRL** register.

where:

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

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

Index	55h	**XCOLKEY0RED**
	56h	**XCOLKEY0GREEN**
	57h	**XCOLKEY0BLUE**
Attributes	R/W, BYTE	
Reset Value	unknown	

colkey0

7	6	5	4	3	2	1	0

colkey0 <7:0> Color Key bits 7 to 0 for window 0. The color key is used to perform color keying between graphics and the video buffer.

Index 40h

Attributes R/W, BYTE

Reset Value unknown

colmsk

7	6	5	4	3	2	1	0

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

The mask is ***only*** used in 32 bits/pixel single frame buffer modes (**depth** = ‘100’) for overlay enable/disable.

See “XCOLKEY” on page 312 for more information.

Index	52h	**XCOLMSK0RED**
	53h	**XCOLMSK0GREEN**
	54h	**XCOLMSK0BLUE**
Attributes	R/W, BYTE	
Reset Value	unknown	

colmsk0

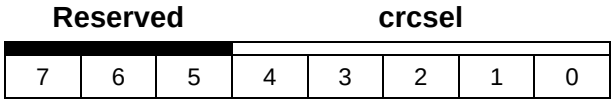
7	6	5	4	3	2	1	0

colmsk0 <7:0> Color Key mask bits 7 to 0 for window 0. To prevent a particular bit plane from participating in a keying comparison, the corresponding color key mask bit should be set to ‘0’.

Index 3Eh

Attributes R/W, BYTE

Reset Value unknown



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

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

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

Index 3Dh

Attributes RO, BYTE

Reset Value unknown

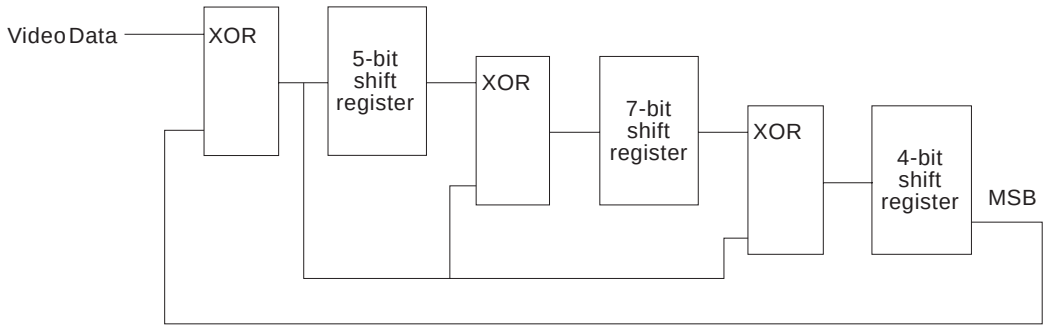
crcdata

7	6	5	4	3	2	1	0

crcdata <7:0> High-order CRC remainder. This register is used to read the results of the 16-bit CRC calculation. **XCRCREMH** corresponds to bits 15:8 of the 16-bit CRC.

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

The CRC is calculated as follows:



Index 3Ch

Attributes RO, BYTE

Reset Value unknown

crcdata

7	6	5	4	3	2	1	0

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

Index 05h

Attributes R/W, BYTE

Reset Value unknown

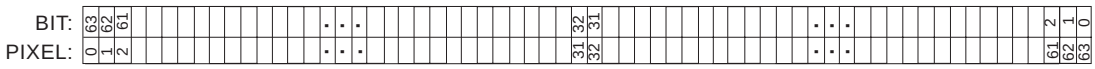
Reserved		curadrh					
7	6	5	4	3	2	1	0

curadrh <5:0> Cursor address high. These are the high-order bits of the cursor map address.

The 14-bit value from the high and low order cursor address locations is the base address (bits 23:10) of the frame buffer where the cursor maps are located. The cursor maps must be aligned on a 1 KByte boundary.

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

Two cursor maps are possible depending on the cursor mode selected (see the curmode field in XCURCTRL). These are based on either a two-slice mode or a six-slice mode. In two-slice mode, each pixel of the cursor is defined by two bits (bit plane 1 and bit plane 0). The cursor data is stored in 64-bit slices in memory (each slice contains all the data for one plane of one cursor scanline). One scanline-plane is stored in memory as follows::



In six-slice mode, each pixel color is defined by 4 bits, with two additional bits: one for transparency and another for complement. The 4-bit color is linearly written into the 4 slices and is binary encoded to select one of 16 cursor colors (see XCURCOL). The memory map of this mode for one scanline is stored in memory as follows:

Address	*Data*
Base + 0	Slice 0 cursor pixels 0 to 15
Base + 1	Slice 1 cursor pixels 16 to 31
Base + 2	Slice 2 cursor pixels 34 to 48
Base + 3	Slice 3 cursor pixels 49 to 63
Base + 4	Complement pixels 0 to 63
Base + 5	Slice 4 transparent pixels 0 to 63

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

Index 04h

Attributes R/W, BYTE

Reset Value unknown

curadrl

7	6	5	4	3	2	1	0

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

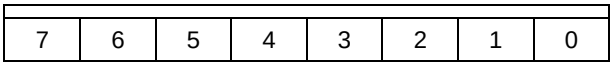
Index

08h	**XCURCOL0RED**	6Fh	**XCURCOL8RED**
09h	**XCURCOL0GREEN**	70h	**XCURCOL8GREEN**
0Ah	**XCURCOL0BLUE**	71h	**XCURCOL8BLUE**
0Ch	**XCURCOL1RED**	72h	**XCURCOL9RED**
0Dh	**XCURCOL1GREEN**	73h	**XCURCOL9GREEN**
0Eh	**XCURCOL1BLUE**	74h	**XCURCOL9BLUE**
10h	**XCURCOL2RED**	75h	**XCURCOL10RED**
11h	**XCURCOL2GREEN**	76h	**XCURCOL10GREEN**
12h	**XCURCOL2BLUE**	77h	**XCURCOL10BLUE**
60h	**XCURCOL3RED**	78h	**XCURCOL11RED**
61h	**XCURCOL3GREEN**	79h	**XCURCOL11GREEN**
62h	**XCURCOL3BLUE**	7Ah	**XCURCOL11BLUE**
63h	**XCURCOL4RED**	7Bh	**XCURCOL12RED**
64h	**XCURCOL4GREEN**	7Ch	**XCURCOL12GREEN**
65h	**XCURCOL4BLUE**	7Dh	**XCURCOL12BLUE**
66h	**XCURCOL5RED**	7Eh	**XCURCOL13RED**
67h	**XCURCOL5GREEN**	7Fh	**XCURCOL13GREEN**
68h	**XCURCOL5BLUE**	80h	**XCURCOL13BLUE**
69h	**XCURCOL6RED**	81h	**XCURCOL14RED**
6Ah	**XCURCOL6GREEN**	82h	**XCURCOL14GREEN**
6Bh	**XCURCOL6BLUE**	83h	**XCURCOL14BLUE**
6Ch	**XCURCOL7RED**	84h	**XCURCOL15RED**
6Dh	**XCURCOL7GREEN**	85h	**XCURCOL15GREEN**
6Eh	**XCURCOL7BLUE**	86h	**XCURCOL15BLUE**

Attributes R/W, BYTE

Reset Value unknown

curcol



curcol <7:0> Cursor color register. The desired color register (0-15) is chosen according to both the cursor mode and cursor map information. (See the **XCURCTRL** register for more information.) Each color register is 24 bits wide and contains an 8-bit red, 8-bit green, and 8-bit blue field.

Index 06h

Attributes R/W, BYTE

Reset Value 0000 0000b

Reserved					curmode		
7	6	5	4	3	2	1	0

curmode <2:0> Cursor mode select. This field is used to disable or select the cursor mode, as indicated below:

- '000': cursor disabled (default)
- '001': three-color cursor
- '010': XGA cursor
- '011': X-Windows cursor
- '100': 16-color palettized cursor
- '101': Reserved
- '110': Reserved
- '111': Reserved

In cursor modes other than 16 color palettized (**curmode** = '100'), there are four possible ways to display each pixel of the cursor. The following table shows how the encoded pixel data is decoded, based on the cursor mode (set by **curmode**):

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

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

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

In 16 color palettized cursor mode, the cursor bit map is placed in memory as described in **XCURADDH.** The following table demonstrates how the bit map data is encoded**:**

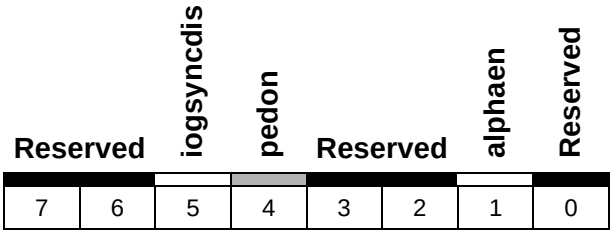
Display	*Slice*					
	5	*4*	*3*	*2*	*1*	*0*
Pixel color	0	0	C_3	C_2	C_1	C_0
Complement	0	1	X	X	X	X
Transparent	1	0	X	X	X	X
Transparent	1	1	X	X	X	X

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

Index 1Dh

Attributes R/W, BYTE

Reset Value 0000 0000b



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

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

pedon <4> Pedestal control. This field specifies whether a 0 or 7.5 IRE blanking pedestal is to be generated on the video outputs.

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

iogsyncdis <5> Green channel sync disable. This field specifies if sync (from the internal signal HSYNC) information is to be sent to the output of the green DAC.

- 0: enable (default)
- 1: disable (sync information is sent to the output of the green DAC)

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

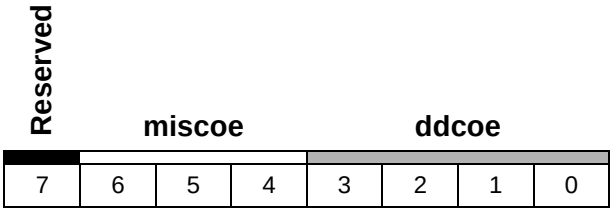
Reserved **<0> <3:2> <7:6>**

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

Index 2Ah

Attributes R/W, BYTE

Reset Value 0000 0000b



ddcoe <3:0> DDC pin output control. Controls the output enable of the driver on pins DDC<3:0>, respectively.

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

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

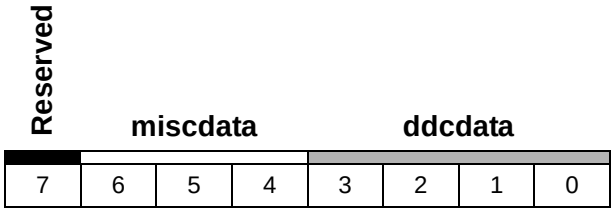
- 0: disable the output driver
- 1: enable the output driver

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

Index 2Bh

Attributes R/W, BYTE

Reset Value 0000 0000b



ddcdata <3:0> DDC pin output state. Controls the output state of the driver on pins DDC<3:0>, respectively. On read, this field returns the state of the DDC<3:0> pins.

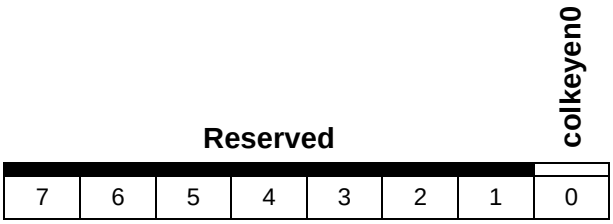
miscdata <6:4> MISC pin output state. Controls the output state of the driver on pins MISC<2:0> during a write operation. On read, this field returns the state of the MISC<2:0> pins.

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

Index 51h

Attributes R/W, BYTE

Reset Value 0000 0001b



colkeyen0 <0> This field selects the type of display for video window ‘0’.

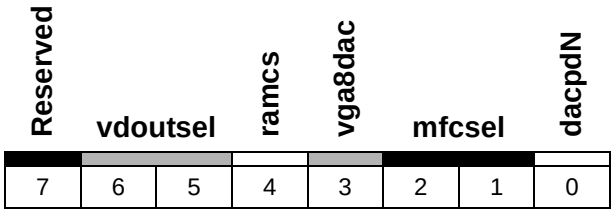
- 0: overlay
- 1: color key

Reserved <7:1> Reserved. When writing to this register, the bits in this field ***must*** be set to ’0’.

Index 1Eh

Attributes R/W, BYTE

Reset Value 0000 0110b



dacpdN <0> DAC power down. This field is used to remove power from the DACs, to conserve power.

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

mfcsel <2:1> Matrox advanced feature connector (MAFC) Function Select.

- '00': Reserved
- '01': Matrox Advanced Feature Connector. In MAFC mode, the data just before the DAC is output to the 12-bit feature connector using both edges of the clock.
- '10': Panel Link Mode. In Panel Link Mode, the data just before the DAC, is output to the 12-bit feature connector using both edges of the clock.
- '11': Disable Feature Connector. When the feature connector is disabled, the VOBLANK/, VDOCLK, and VDOUT<11:0> pins are driven low.

vga8dac <3> VGA 8-bit DAC. This field is used for compatibility with standard VGA, which uses a 6-bit DAC.

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

ramcs <4> LUT RAM chip select. Used to power up the LUT.

- 0: LUT disabled
- 1: LUT enabled

vdoutsel <6:5> Video Out Select.

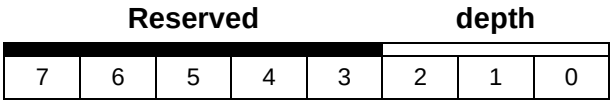
- '00': In MAFC12 mode, the outputs are taken just before DAC and are output 12 bits at a time on both edges of the clock. This effectively transfers one 24 bit pixel (8 bits per channel) per clock. This works in all MGA display modes.
- '01': Reserved
- '10': In BYPASS656 mode, the output is taken directly from the Video-In bus and passed directly through to the Video-Out bus at 1 byte per clock cycle.
- '11': Reserved

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

Index 19h

Attributes R/W, BYTE

Reset Value 0000 0000b



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

Value	*Color Depth Used*	
'000'	8 bits/pixel	(palettized) (default)
'001'	15 bits/pixel	(palettized) + 1-bit overlay
'010'	16 bits/pixel	(palettized)
'011'	24 bits/pixel	(packed, palettized)
'100'	32 bits/pixel	(24 bpp direct, 8 bpp overlay palettized)
'101'	Reserved	
'110'	Reserved	
'111'	32 bits/pixel	(24 bpp palettized, 8 bpp unused)

When at '0', the **mgamode** field of the **CRTCEXT3** VGA register sets the DAC to VGA mode. The **depth** field should be programmed to '000' when in VGA mode.

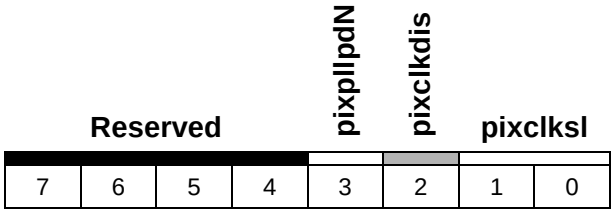
Reserved <7:3> Reserved. When writing to this register, the bits in this field ***must*** be set to '0'.

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

Index 1Ah

Attributes R/W, BYTE

Reset Value 0000 0000b



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

- '00': selects the output of the PCI clock
- '01': selects the output of the pixel clock PLL
- '10': selects external source (from the **VDOCLK** pin)
- '11': Reserved

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

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

pixpllpdN<3> Pixel PLL power down.

- 0: power down
- 1: power up

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

➨ ***Note:*** See "Programming the PLLs" on page 79 for information on modifying the clock parameters.

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

Attributes R/W, BYTE

Reset Value 15h **XPIXPLLAM**
1Eh **XPIXPLLBM**
unknown **XPIXPLLCM**

Reserved			pixpllm				
7	6	5	4	3	2	1	0

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

There are three sets of PIXPLL registers:

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

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

clksel	*Pixel Clock PLL Frequency*	*Reset Value*
‘00’	Register Set A (25.172 MHz)	M = 21
‘01’	Register Set B (28.361 MHz)	M = 30
‘1X’	Register Set C	Unknown

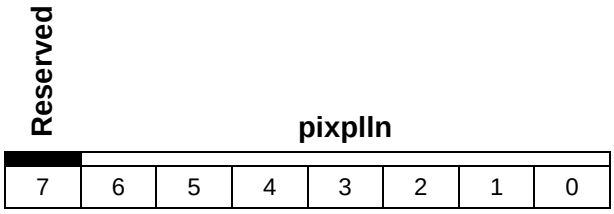
Note: The **pixpllm** value *must* be in the range of 1 to 31.

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

Index 45h **XPIXPLLAN**
49h **XPIXPLLBN**
4Dh **XPIXPLLCN**

Attributes R/W, BYTE

Reset Value 28h **XPIXPLLAN**
40h **XPIXPLLBN**
unknown **XPIXPLLCN**



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

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

clksel	*Pixel Clock PLL Frequency*	*Reset Value*
‘00’	Register Set A (25.172 MHz)	N = 40
‘01’	Register Set B (28.361 MHz)	N = 64
‘1X’	Register Set C	Unknown

Note: The **pixplln** value must be in the range of 1 (1h) to 127 (7Fh) inclusive.

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

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

Attributes R/W, BYTE

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

Reserved			pixplls		pixpllp		
7	6	5	4	3	2	1	0

pixpllp <2:0> Pixel PLL P value register. The ‘p’ value is used by the VCO clock divider circuit. The permitted values are:

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

pixplls <4:3> Pixel PLL S value register. The ‘s’ value controls the loop filter bandwidth.

50 MHz <= Fvco < 100 MHz S=0
100 MHz <= Fvco < 140 MHz S=1
140 MHz <= Fvco < 180 MHz S=2
180 MHz <= Fvco < 250 MHz S=3

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

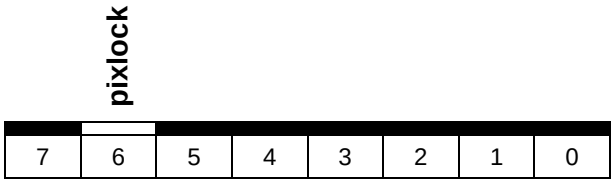
clksel	*Pixel Clock PLL Frequency*	*Reset Value*
‘00’	Register Set A (25.172 MHz)	P = 1, S = 0
‘01’	Register Set B (28.361 MHz)	P = 1, S = 0
‘1X’	Register Set C	Unknown

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

Index 4Fh

Attributes RO, BYTE

Reset Value unknown



pixlock <6> Pixel PLL lock status.

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

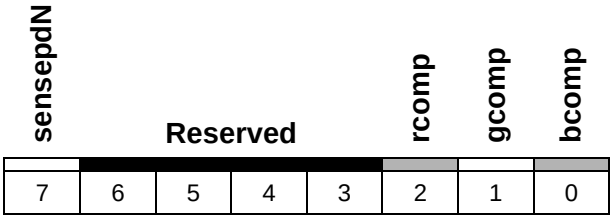
Reserved <5:0> <7>

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

Index 3Ah

Attributes R/W, BYTE

Reset Value 0XXX XXXXb



bcomp RO<0> Sampled blue compare. Verifies that the blue termination is correct.

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

gcomp RO<1> Sampled green compare. Verifies that the green termination is correct.

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

rcomp RO<2> Sampled red compare. Verifies that the red termination is correct.

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

sensepdN <7> Sense comparator power down

- 0: power down
- 1: power up

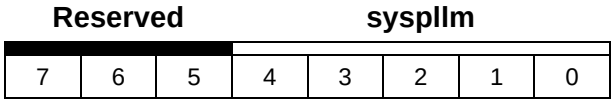
Reserved <6:3> Reserved. When writing to this register, the bits in this field ***must*** be set to ’0’.

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

Index 2Ch

Attributes R/W, BYTE

Reset Value 0000 0101b



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

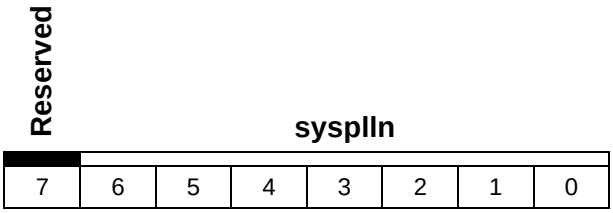
➽ ***Note:*** The **syspllm** value *must* be in the range of 1 to 31.

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

Index 2Dh

Attributes R/W, BYTE

Reset Value 0001 1111b



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

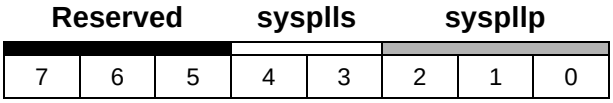
Note: The **sysplln** value must be in the range of 1 (1h) to 127 (7Fh) inclusive.

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

Index 2Eh

Attributes R/W, BYTE

Reset Value 0001 0000b



syspllp <2:0> System PLL P value register. The ‘p’ value is used by the VCO post-divider circuit.

The permitted values are:

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

Other values are reserved.

syspllS <4:3> System PLL S value register. The ‘s’ value controls the loop filter bandwidth.

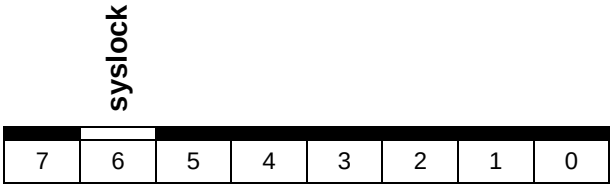
50 MHz <= Fvco < 100 MHz → S=0
100 MHz <= Fvco < 140 MHz → S=1
140 MHz <= Fvco < 180 MHz → S=2
180 MHz <= Fvco < 250 MHz → S=3

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

Index 2Fh

Attributes RO, BYTE

Reset Value unknown



syslock <6> System PLL lock status.

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

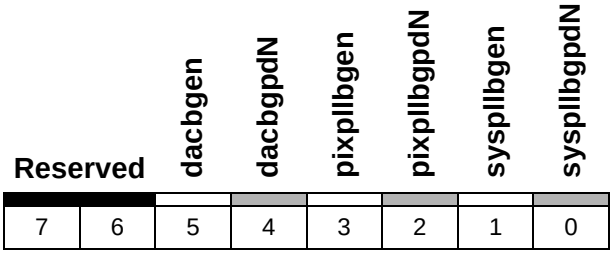
Reserved **<5:0> <7>**

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

Index 18h

Attributes R/W, BYTE

Reset Value 0000 0000b



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

- 0: power down
- 1: power up

syspllbgen <1> System PLL voltage reference enable.

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

pixpllbgpdN <2> Pixel PLL voltage reference block power down.

- 0: power down
- 1: power up

pixpllbgen <3> Pixel PLL voltage reference enable.

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

dacbgpdN <4> DAC voltage reference block power down.

- 0: power down
- 1: power up

dacbgen <5> DAC voltage reference enable.

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

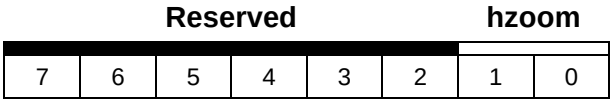
Reserved <7:6> Reserved. When writing to this register, the bits in this field ***must*** be set to ’0’.

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

Index 38h

Attributes R/W, BYTE

Reset Value 0000 0000b



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

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

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

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

Chapter 4: Programmer's Specification

4.1 HOST Interface

4.1.1 Introduction

The MGA-G200-PCI chip interacts directly with the PCI interface, while the MGA-G200-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.

CACHE This is a 8-location cache, which is used for direct frame buffer MGA read accesses.

ROMFIFO This is a 2-entry FIFO used for ROM accesses.

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

Access	*Type*	*BFIFO*	*MIFIFO*	*CACHE*	*ROMFIFO*
Configuration registers Host Registers VGA Registers DAC Registers Backend Scaler Registers Video-In and CODEC Registers WARP Instruction Memory	R W	—	—	—	—
ROM	R W	—	—	—	W W
DMAWIN or **MGABASE3**	WO	W	—	—	—
Drawing registers	R W	— W	—	—	—
Host registers +DRWI(1)	R W	— W	—	—	—
VGA frame buffer	R W	—	W W	—	—
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 Frame Buffer is read when the MIFIFO is not empty or when the data is not available.
- The VGA registers are written to when the MIFIFO is not empty.
- The ROMFIFO is written to when it is full.
- The Serial EEPROM is read and the data is not available.

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 used:

Access	*Access Type*
MGABASE1 + DMAWIN range	W
MGABASE1 + drawing register range	W
MGABASE1 + host reg. range +DRWI range	W
MGABASE1 + WARP instruction memory range	W
MGABASE3 range	W
VGA frame buffer range	W
VGA frame buffer range (mgamode = 1)	R (cache hit)
MGABASE2 range	W
MGABASE2 range	R (cache hit)

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.

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 the VGA frame buffer range, the MGABASE2 range (when there is a cache miss), or the ROMBASE range will activate the delayed transaction mechanism (when the **noretry** field of **OPTION** = '0').

4.1.4 PCI Target-Abort Generation

The MGA-G200 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-G200.

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

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

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

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-G200'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 eight-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 first access will be completed in 2 pclks, and the data phase of the next accesses (in the case of a burst) will be completed in 1 pclk.

The following situations will cause a *cache flush*, in order to maintain data coherency:

1. A write access to the frame buffer (**MGABASE2** or VGA frame buffer).
2. A write to the VGA registers (either I/O or memory).
3. 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-G200 to correct this situation.

The exact action that will be taken depends on the data size (the MGA-G200 must be aware of the data size when processing Big-Endian data). The data size depends on the location of the data (the specific memory space), and the pixel size (when the data is a pixel).

There are ***six*** distinct memory spaces:

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

Configuration space

Each register in the configuration space is 32 bits, and should be addressed using dword accesses. For these registers, no byte swapping is done, and bytes will appear in different positions, depending on the endian mode of the host processor. Keep in mind that the MGA-G200 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-G200 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-G200 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-G200 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.

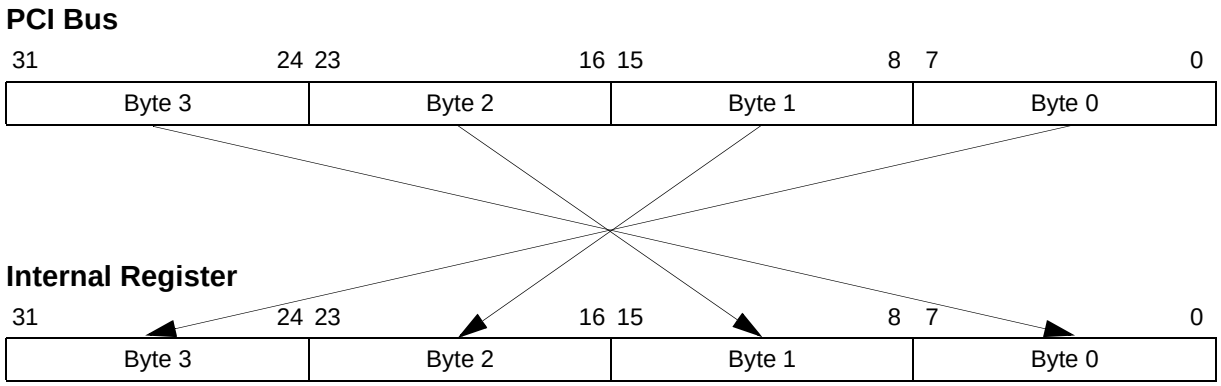
Byte swapping for Big-Endian processors can be enabled in the register space by setting the **OPTION** configuration space register's **powerpc** bit to 1.

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.

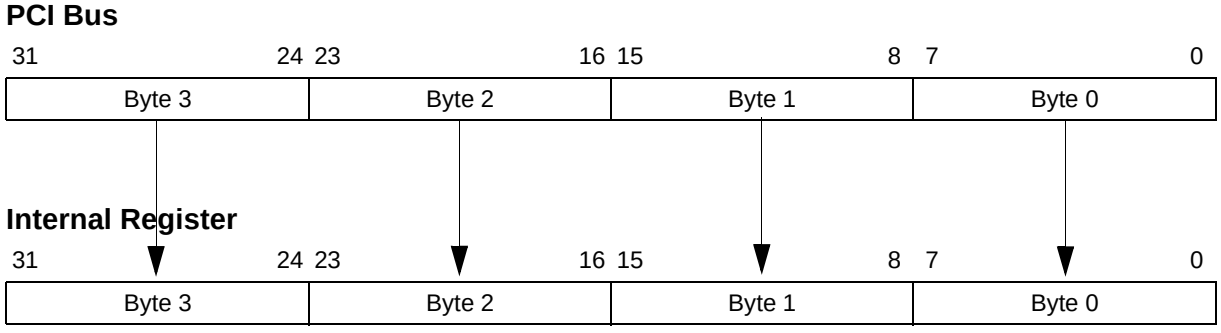
Note: Byte and word accesses will not return the same data on both Little and Big-Endian processors.

In the register mapping tables in Chapter 3, all addresses are given for a Little-Endian processor.

powerpc = 1



powerpc = 0

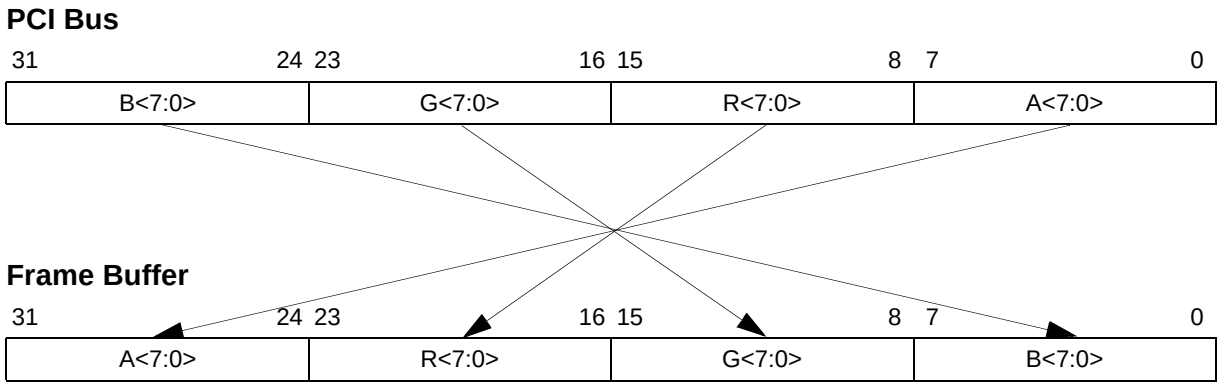


Frame Buffer Space

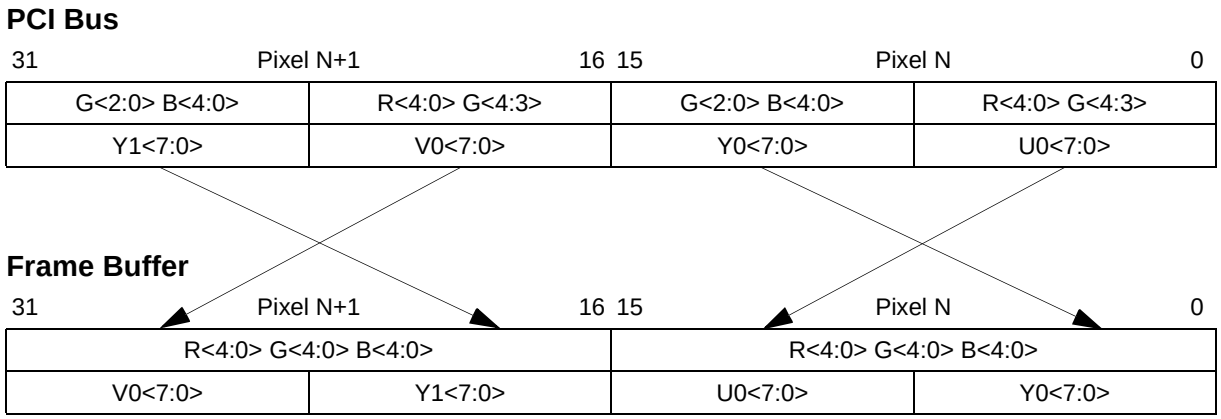
The frame buffer is organized in Little-Endian format, and byte swapping depends on the size of the pixel. As usual, addresses are not modified.

Swapping mode is directed by the **dirDataSiz** field of the **OPMODE** host register. This field is used for direct access either through the VGA frame buffer window or the full memory aperture. The only exception is 24 bits/pixel mode, which is correctly supported only by Little-Endian processors.

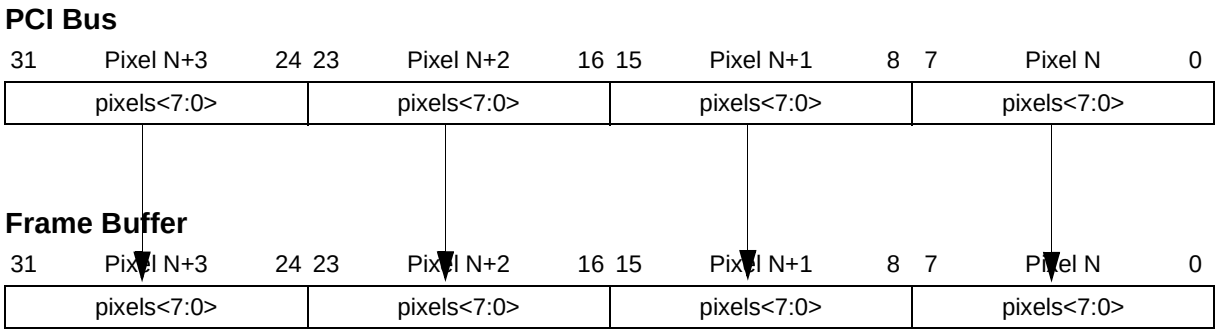
32 bits/pixel, dirDataSiz = 10



16 bits/pixel, dirDataSiz = 01



8 bits/pixel, dirDataSiz = 00



ILOAD 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 ILOAD operations in DMA BLIT WRITE mode). Other DMA modes - DMA General Purpose, DMA Vector Write, or DMA Vertex Write - should set **dmaDataSiz** to '10' for Big-Endian or '00' for Little-Endian processor.

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 and ILOAD are explained in their respective sections of this chapter.

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

Pixel Format (From the Processor's Perspective)

8-bit A Little-Endian 8-bit (see the **powerpc** field of **OPTION**) is used in ILOAD operations. Refer to Table 4-3 on page 4-58.

	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 operations. Refer to Table 4-3 on page 4-58.

	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 operations. Refer to Table 4-3 on page 4-58.

	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 operations. Refer to Table 4-3 on page 4-58.

	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 operations. Refer to Table 4-3 on page 4-58.

	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 operations. Refer to Table 4-3 on page 4-58.

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

24-bit A 24-bit RGB packed pixel, used in ILOAD operations. Refer to Table 4-3 on page 4-58.

	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 operations. Refer to Table 4-4 on page 4-66.

	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	:	:	:	:

MONO A Little-Endian 1-bit used in ILOAD and BITBLT operations. Refer to Table 4-4 on page 4-59.

	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 4-4 on page 4-59.

	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 4-4 on page 4-59.

	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	:

4.1.9 Programming Bus Mastering for DMA Transfers

When the busmaster field bit of the DEVCTRL register is ‘0’(disabled), the MGA-G200 will not perform PCI bus mastering, even if the other bus mastering registers are accessed. Disabling busmaster does not preclude writing to the host registers related to bus mastering. To enable AGP bus mastering, the following register must be correctly set: **data_rate**, **agp_enable** and **sba_enable**.

➽ ***Note:*** In order to enable AGP2X transfers, **agp2xpllen *must*** be set

The bus mastering feature allows the MGA-G200 to access system memory through a DMA channel (used in conjunction with Pseudo-DMA mode). In order to send 3D commands to the chip, General Purpose Pseudo-DMA should be used. For texture mapping transfers between system memory and the off-screen area, ILOAD Pseudo-DMA mode should be used. For Vertex transfer to the WARP engine, the Vertex write Pseudo-DMA mode should be used.

➽ ***Note:*** This is the *recommended* usage - *any* Pseudo-DMA mode can actually be used for either case.

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

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

4.1.9.1 DMA Registers

Primary Current Address (PRIMADDRESS)

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

Primary End Address (PRIMEND)

This register must be initialized with the end address of the primary DMA channel in the system memory area. Bit 1 is used to specify the system memory type where the channel reside in (PCI/AGP). And bit 0 can be used to prevent that the primary list execution restart when there is a Soft Trap Interrupt pending.

Secondary Current Address (SECADDRESS)

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

Secondary End Address (SECEND)

This register must be initialized with the end address of the secondary DMA channel in the system memory area. It is accessed using General Purpose Pseudo-DMA mode (primary DMA transfer) in order

to be able to start secondary DMA transfers while the primary DMA channel is active. Bit 1 is used to specify the system memory type where the channel reside in (PCI/AGP).

Setup Current Address (SETUPADDRESS)

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

Setup End Address (SETUPEND)

This register must be initialized with the end address of the setup DMA channel in the system memory area. It is accessed using General Purpose Pseudo-DMA mode (primary DMA transfer) in order to start setup DMA transfers while the primary DMA channel is active. Bit 1 is used to specify the system memory type where the channel reside in (PCI/AGP).

Soft Trap Interrupt (SOFTRAP)

This register is used when the secondary DMA channel is unavailable (due to a system or memory constraint, or other reason). When **SOFTRAP** is written, the MGA-G200 will generate an interrupt (if the **IEN** register is set). In the context of texture mapping, this register must be written with the handle of the texture so that the interrupt handler can know where the texture is located in system memory. Writing this register stops primary DMA transfers. To restart the primary DMA channel, the **PRIMEND** register must be re-written with bit 0 set to ‘0’.

Interrupt Clear (ICLEAR)

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

Interrupt Enable (IEN)

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

Status (STATUS)

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

4.1.9.2 Using the DMA Channel

To use the DMA channel, follow this sequence:

1. Write a list of commands to a buffer in main memory.
2. Write the starting address of the primary buffer in memory to the **PRIMADDRESS** register. Since this is a 32-bit pointer, the two LSBs are not used as an address but rather as an indication of the type of Pseudo-DMA transfer to be used.
3. Write the address of the first dword after the end of the primary buffer to the **PRIMEND** register, and the memory type the list resides in (PAGPXFER).
4. As soon as the **PRIMEND** register is accessed, the primary DMA channel will be activated (if there is no **SOFTRAP** pending or if bit 0 (**PRIMNOSTART**) of **PRIMEND** is set to ‘0’).
5. A read access will be performed on the PCI bus at the location pointed to by **PRIMADDRESS**. The data that is read will be fed to the 7K Pseudo-DMA window (which is internal to the chip). **PRIMADDRESS** will be advanced to point to the next dword.
6. If, within this process the host requires the second level of Pseudo-DMA, then the **SECADDRESS** register must be written with the starting address of the secondary buffer in memory and the Pseudo-DMA mode to be used, then the **SECEND** register must be written. In this case, steps 7 to 9 will be taken; if not, operations resume at step 10.

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

➧ ***Note:*** If the primary list status fetch pointer enable (**primptren**) is set, the DMA engine will send the status info to the system memory at the address programmed in **PRIMPTR**.

7. Read accesses will be performed on the PCI/AGP bus at the location pointed to by **SECADDRESS** with the method indicated by SAGPXFER. The data that is read will be fed to the DMAWIN window (which is internal to the chip). The secondary current address will be advanced to point to the next dword.
8. The **SECADDRESS** and **SECEND** registers cannot be accessed while they are being used by the secondary DMA channel. This will produce unpredictable results.
9. **SECADDRESS** and **SECEND** are compared. If they are different, the secondary DMA continues (refer to step 7). If they are equal, the secondary DMA is finished and the primary DMA continues. It should be noted that when the primary DMA continues, the selected Pseudo-DMA mode restarts. The General Purpose Pseudo-DMA mode is selected, the first dword fetch will be interpreted as a set of four register indexes.
10. If, within this process the host requires a level of Setup-DMA, then the **SETUPADDRESS** register must be written with the starting address of the setup buffer in memory and the Setup-DMA mode to be used, then the **SETUPEND** register must be written. In this case, steps 11 to 13 will be taken; if not, operations resume at step 14.

➧ ***Note:*** It is not permitted to set **SETUPEND** to the same value as **SETUPADDRESS**.

➧ ***Note:*** If the primary list status fetch pointer enable (**primptren**) is set, the DMA engine will send the status info to the system memory at the address programmed in **PRIMPTR**.

11. Read accesses will be performed on the PCI/AGP bus at the location pointed by **SETUPADDRESS** with the method indicated by **SETUPEND**. The data that is read will be fed to the Setup-DMA engine which will generate virtual secondary DMA based on the Setup-DMA mode (SETUPMOD) (refer to section 4.1.9.3 for Setup-DMA mode description). The setup current address will be advanced to point to the next dword.

➧ ***Note:*** The **SETUPADDRESS** and **SETUPEND** registers can only be accessed by the primary DMA channel.

12. **SETUPADDRESS** and **SETUPEND** are compared. If they are different, the setup DMA continues (refer to step 10). If they are equal, the setup DMA is finished and the primary DMA continues. It should be noted that when the primary DMA continues, the selected Pseudo-DMA mode restarts. The General Purpose Pseudo-DMA mode is selected, the first dword fetch will be interpreted as a set of four register indexes.
13. **PRIMADDRESS** is compared with **PRIMEND**. If they are different, the DMA transfer continues (refer to step 5). If they are equal, the DMA transfer is complete.
14. If the **SOFTRAP** register is accessed in the primary DMA channel, the primary DMA transfer will stop and an interrupt will be generated (see the softrapen field). In the context of texture mapping, the **SOFTRAP** register must be written with the handle of the texture so that the interrupt routine will know what action to take. To restart the primary DMA channel, the **PRIMEND** register must be written with the **primnostart** field set to ‘0’. Until **PRIMEND** is written, any operation can be undertaken with the MGA-G200. If the DMAWIN window is accessed before **PRIMEND** is written, it

will be controlled by the **dmamod** field of the **OPTION** register (when the **SOFTRAP** register is written, a DMA reset will occur).

4.1.9.3 Setup-DMA Channel Operation

The Setup-DMA channel (access via **SETUPADDRESS** and **SETUPEND**) is an intermediate level of DMA used to build a virtual secondary list based on the **setupmod** programmed.

The only **setupmod** available is the DMA Vertex Fixed Length Setup List mode (see page 3-137) which uses the information in the list to build a virtual secondary list of DMA Vertex Write for each entry of the list.

Each entry of the list is used as a pointer to the start of a Vertex list in system memory, the length of each list correspond to the **wvrtxsz** field of the **WVRTXSZ** register.

So,

virtual**SECADDRESS**(n)	= data read from the ‘n’ entry of the Setup-DMA list
virtual **SECMOD**(n)	= DMA Vertex Write
virtual **SECEND**(n)	= data read from the ‘n’ entry of the Setup-DMA list + WVRTXSZ + 1
virtual **SAGPXFER**(n)	= SETUPAGPXFER

4.1.9.4 Special Cases

The PCI Specification indicates that when the MGA-G200 acts as a master on the PCI bus, two particular circumstances can arise (aside from the regular transfer of data):

1. The first case is a ***master-abort***, which occurs when an access is attempted and no device responds (often due to a glitch in programming). When this happens, the **recmastab** bit of the **DEVCTRL** register will be set (and will remain set until a ‘1’ is written to it).
2. The second case is a ***target-abort***. This is a target termination that is used when the target detects a problem with an access generated by the MGA-G200. The MGA-G200 always generates accesses in the correct form, so this situation depends on the target. Target-aborts should *not* occur, since the PCI Specification indicates that they occur with I/O accesses; the MGA-G200 generates ***memory*** accesses. There is no way to change the MGA-G200 or its programming to prevent a target-abort from occurring. If a target-abort occurs, the **rectargab** bit of the **DEVCTRL** register will be set (and will remain set until a ‘1’ is written to it).

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

Reset of the Pseudo-DMA sequence:

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

- When the **PRIMADDRESS** register is written.
- When the **SECEND** register is written, assuming **SECEND** is ***not*** equal to **SECADDRESS**).
- When the **SETUPEND** register is written, assuming **SETUPEND** is ***not*** equal to **SETUPADDRESS**).
- When the **SOFTRAP** register is written.
- When secondary DMA transfers end (that is, when **SECADDRESS** becomes equal to **SECEND** at the end of the secondary DMA, and not when **SECADDRESS** is written with the **SECEND** value).
- When setup DMA transfers end (that is, when **SETUPADDRESS** becomes equal to **SETUPEND** at the end of the setup DMA, and ***not*** when **SETUPADDRESS** is written with the **SETUPEND** value).
- When a master-abort or target-abort is detected.

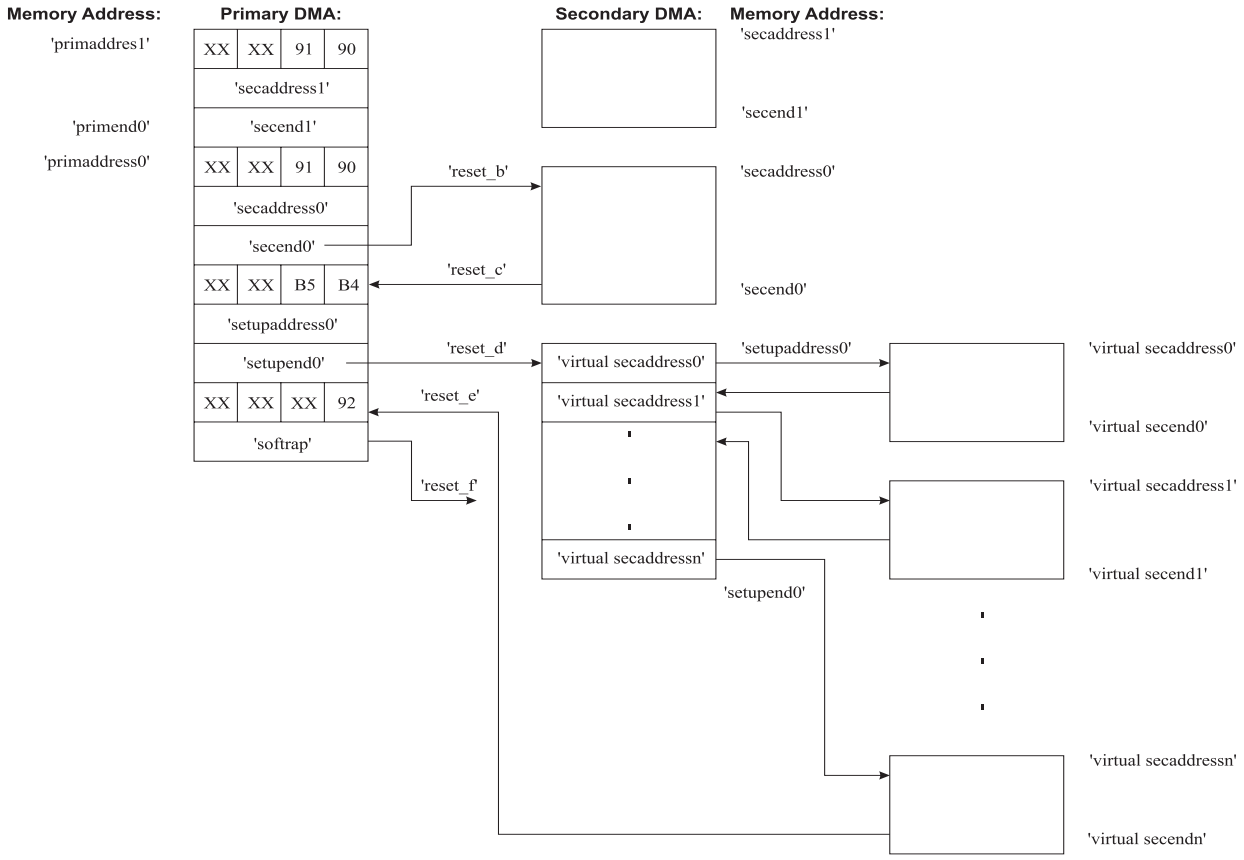
Note: There is *no* reset of the Pseudo-DMA sequence when **PRIMEND** is written (since **PRIMEND** starts the primary DMA transfers); writing can happen more than once to extend the list (even while the list is still being transferred).

Note: There is *no* reset of the Pseudo-DMA sequence when primary DMA transfers end. If commands are added to the primary display list, **PRIMEND** has to be written with its new value to restart the primary DMA transfers.

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

Example:

- When **PRIMADDRESS** is written with ‘primaddress0’, a reset of the Pseudo-DMA sequence is executed.
- When **SOFTRAP** is written (through a primary DMA transfer), another reset is executed (‘reset_f’).
- When **SECEND** is written, and when the secondary DMA ends, two other resets are executed (‘reset_b’ and ‘reset_c’).
- When **SETUPEND** is written, and when the setup DMA ends, two other resets are executed (‘reset_d’ and ‘reset_e’).


Memory Address:
Primary DMA:
Secondary DMA:
Memory Address:
'primaddres1'
XX XX 91 90
'secaddress1'
'secend1'
'primend0'
'primaddress0'
XX XX 91 90
'secaddress0'
'secend0'
XX XX B5 B4
'setupaddress0'
'setupend0'
XX XX XX 92
'softrap'
'reset_b'
'reset_c'
'reset_d'
'reset_e'
'reset_f'
'virtual secaddress0'
'virtual secaddress1'
'virtual secaddressn'
'secaddress1'
'secend1'
'secaddress0'
'secend0'
'setupaddress0'
'setupend0'
'virtual secaddress0'
'virtual secend0'
'virtual secaddress1'
'virtual secend1'
'virtual secaddressn'
'virtual secendn'

Additional Information

- When the DMA channel is used (mastering), it is possible to know which parts of the buffer have been executed by the drawing engine.
    - The DMA current pointer is readable by the CPU through the **PRIMADDRESS**, **SECADDRESS** or **SETUPADDRESS** registers.
    - The primary list status fetch pointer (**PRIMPTR**) functionality can be used to allow the DMA engine to send the status info to system memory every time **SECEND**, **SETUPEND** or **SOFTRAP** is written.
- The **endprdmasts** field of the **STATUS** register always indicates whether or not all the DMA channels have been read completely (**PRIMADDRESS** = **PRIMEND** and **SECADDRESS** = **SECEND** and **SETUPADDRESS** = **SETUPEND**). It is set to '1' when a soft trap interrupt occurs. This bit toggles to '0' as soon as the primary DMA channel restarts.

If primnostart is '1' when reprogramming primend:

- To get an interrupt when the primary DMA channel terminates, include a write to the **SOFTRAP** register as the last DMA transfer.

4.2 Memory Interface

4.2.1 Frame Buffer Organization

The MGA-G200 supports up to 16 megabytes (MB) of 16Mb SGRAM memory divided into four 4 MByte banks, up to 16 MB of SDRAM memory divided into two 8 MByte banks, or up to 8 MB of SGRAM memory divided into four 2 MByte banks.

There are two different frame buffer organizations, described below:

- VGA Mode
- Power Graphic Mode

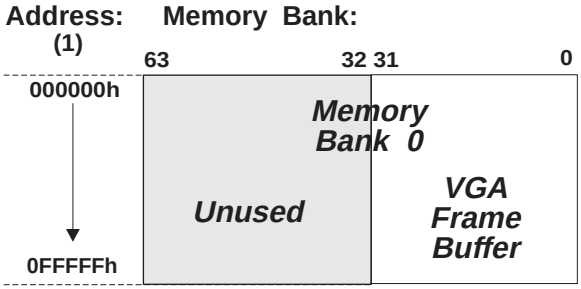
4.2.1.1 Supported Resolutions

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

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: OPTION<12:10> memconfig [2:0] = '000'

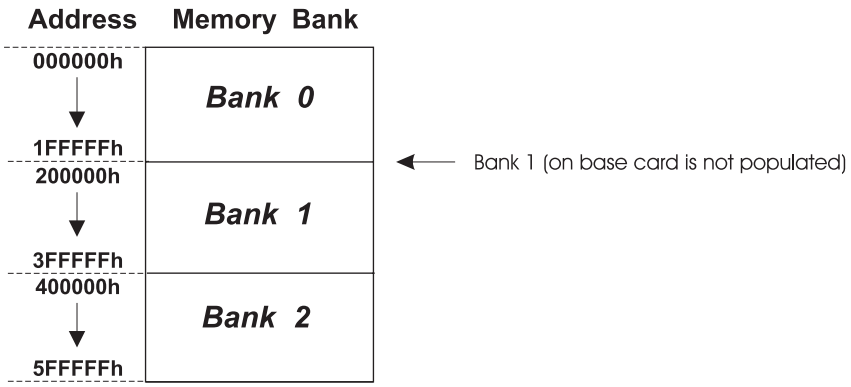


Figure 4-2: OPTION<2:0> memconfig [2:0] = '001'

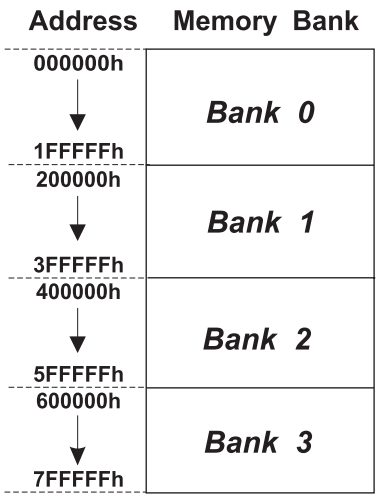


Figure 4-3: OPTION<12:10> memconfig [2:0] = '010' or '110'

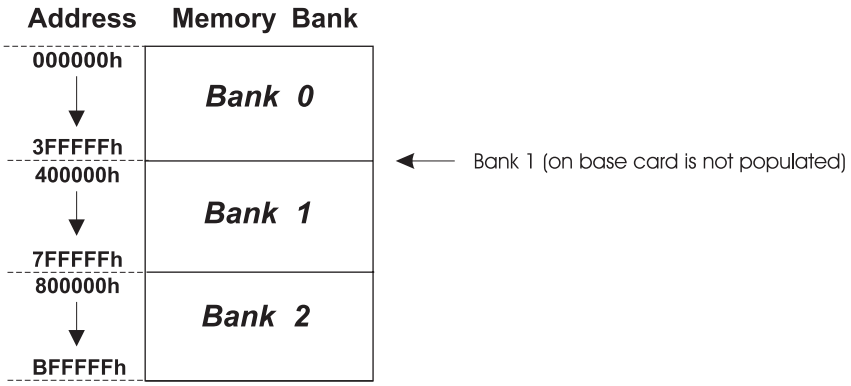


Figure 4-4: OPTION<2:0> memconfig [2:0] = '011' or '111'

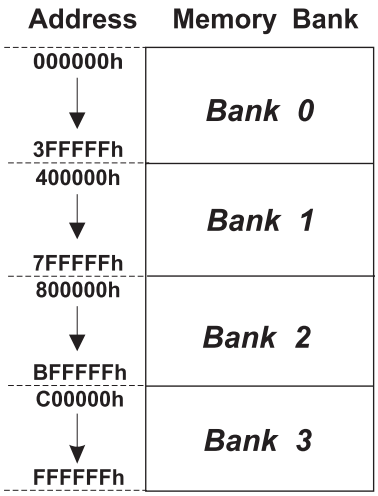
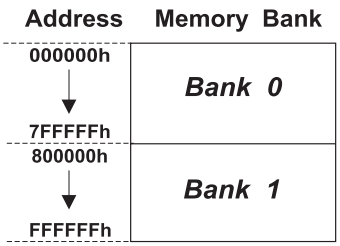


Figure 4-5: OPTION<2:0> memconfig [2:0] = '100' or '101'

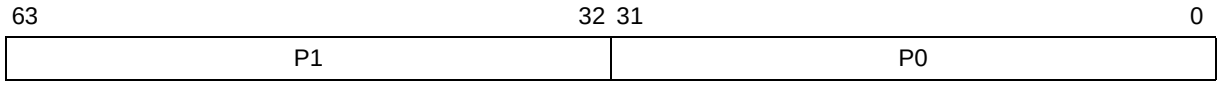


4.2.2 Pixel Format

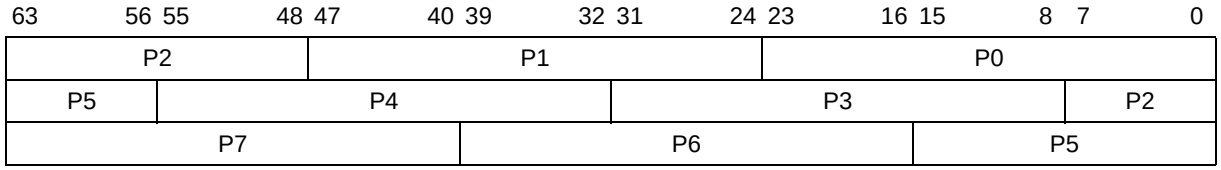
The slice is 64 bits long and is organized as follows. In all cases, the least significant bit is 0. The Alpha part of the color is the section of a pixel that is not used to drive the DAC. Note that the data is always true color, but in 8 bit/pixel formats pseudo color can be used when shading is not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

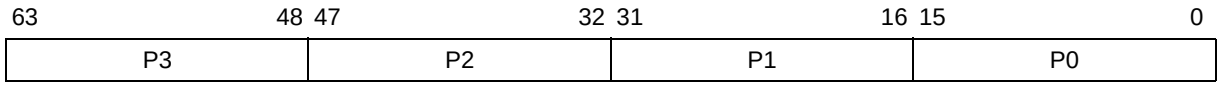
32 bits/pixel



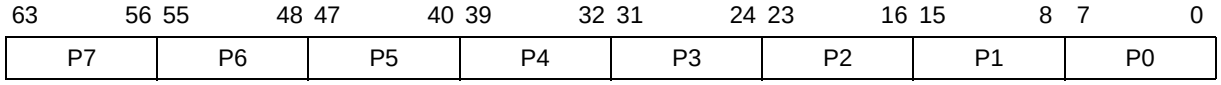
24 bits/pixel



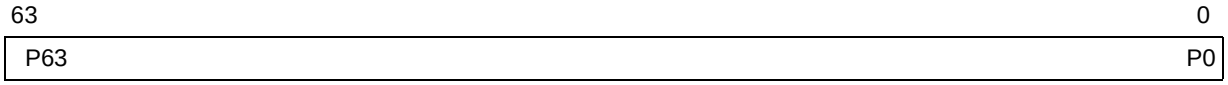
16 bits/pixel



8 bits/pixel

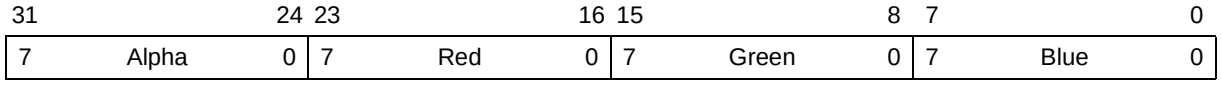


Monochrome

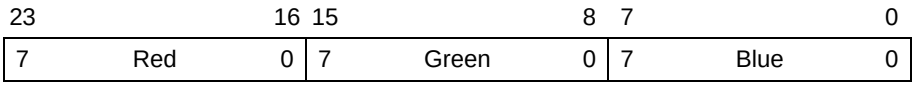


For each of these modes, the pixels are arranged as follows:

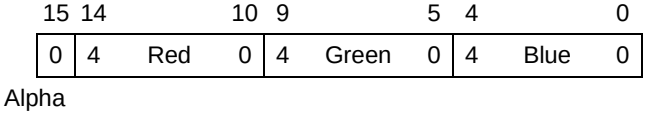
32 bits/pixel



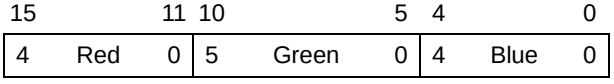
24 bits/pixel



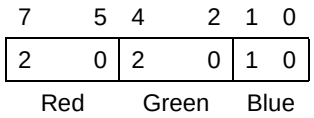
16 bits/pixel (5:5:5)



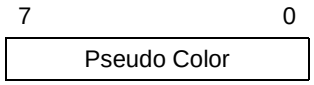
16 bits/pixel (5:6:5)



8 bits/pixel



8 bits/pixel



4.3 Chip Configuration and Initialization

4.3.1 Reset

The MGA-G200 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*
biosboot	HDATA<1>	Indicates whether a ROM is installed (‘1’) or not (‘0’). The biosboot strap also controls the **biosen** field of the **OPTION** register.
modclkp	MDQ<31:29>	Used to determine the frequency at which an LVTTL memory expansion module is designed to operate.
vgaboot	HDATA<0>	Indicates whether the VGA I/O locations are decoded (‘1’) or not (‘0’) only if the **vgaioen** bit has not been written. The vgaboot strap also controls bit 23 of the **CLASS** register, setting the **class** field to ‘Super VGA compatible controller’ (‘1’) or to ‘Other display controller’ (‘0’).

A soft reset is performed by programming a ‘1’ into bit 0 of the **RST** host register. Soft reset will be maintained until a ‘0’ is programmed (see the **RST** register description on page 3-134 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; the softextrst (RST host register) does.

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	'000000'	The refresh counter defines the rate of MGA memory refresh. For a typical 144 MHz MCLK, a value of 22h 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 SGRAM Reset Sequence

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

Analog Macro Power Up Sequence

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

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

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

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

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

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

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

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

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

The PLLs are now set up and oscillating at their reset frequencies, but they are not selected. The following steps will set MCLK to143 MHz, GCLK to 71.5 MHz, and PIXCLK to 25.175 MHz. See 'Programming the PLLs' on page 4-79.

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

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

SGRAM/SDRAM Initialization

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

Step 2. Program all the fields of the **MCTLWTST** register.

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

Step 4. Program the **mbuftype** field of the **MEMRDBK** register.

Step 5. Program the **mrscopcod** field of the **MEMRDBK** register to '0000'.

Step 6. Program the mclkbrd0 and mclkbrd1 fields in the MEMRDBK register with tap delay values appropriate for the SGRAM type and loading.

Step 7. Wait a minimum of 200 µs.

Step 8. Set the **memreset** field of **MACCESS**.

Step 9. Start the refresh by programming the **rfhcnt** field of the **OPTION** register.

The MGA-G200 provides three different display modes: text (VGA or SVGA), VGA graphics, and SVGA graphics. Table 4-1 lists all of the display modes which are available through BIOS calls.

- The text display uses a multi-plane configuration in which a character, its attributes, and its font are stored in these separate memory planes. All text modes are either VGA-compatible or extensions of the VGA modes.
- The VGA graphics modes can operate in either multi-plane or packed-pixel modes, as is the case with standard VGA.
- The SVGA modes operate in packed-pixel mode - they enable use of the graphics engine. This results in very high performance, with high resolution and a greater number of pixel depths.

Table 4-1: Display Modes (Part 1 of 2)

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

Table 4-1: Display Modes (Part 2 of 2)

Mode	Type	Organization	Resolution	No. of colors
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
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 ‘CRTC Programming’ on page 4-61).
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:*** 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:*** The MGA-G200 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).

➧ ***Note:*** Plane write masks are *not* available during direct frame buffer accesses.

4.5 Drawing in Power Graphic Mode

This section explains how to program the MGA-G200's registers to perform various graphics functions. The following two methods are available:

- 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 be used with any command.
  - The DMA Vector Write mode is specifically dedicated to polyline operations.
  - ILOAD 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

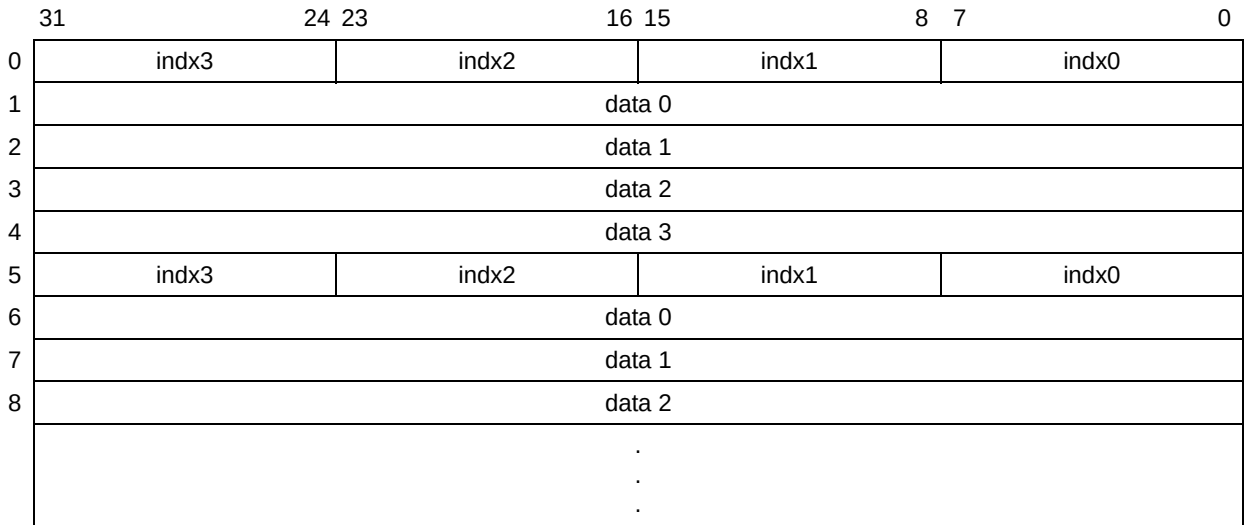
The general purpose Pseudo-DMA operations are performed through the DMAWIN aperture in the MGA control register space, or in the 8 MByte Pseudo-DMA window. It is recommended that host CPU instructions be used in such a way that each transfer increments the address. This way, the PCI bridge can proceed 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. 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 dword written to the DMA window is loaded into the Address Generator. This dword contains indices to the next four drawing registers to be written, and the next four dword transfers contain the data that is to be written to the four registers specified.

When each dword 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 dword. 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 dword 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, lines 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 3-99).

   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 ***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)
DSTORG	Determine screen origin	**DSTORG** should be used instead of **YDSTORG**
MACCESS	Set pixel format (8, 16, 24, 32 bpp) and Z precision (16 or 32 bits) and sets the dithering mode.	Some limitations apply. Dithering mode is used in the following primitives: (i) lines with depth, (ii) Gouraud shaded trapezoids, (iii) texture mapping, (iv) unformatted ILOAD.
CXBNDRY	Left/right clipping limits	Can use **CXLEFT** and **CXRIGHT** instead
YTOP	Top clipping limit	—
YBOT	Bottom clipping limit	—
PLNWT	Plane write mask	—
ZORG	Z origin position	Only required for depth operations

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 4.5.5.1 first.

➽ ***Note:*** In order to start the drawing engine, the last register programmed ***must*** be accessed in the 1D00h-1DFFh range.

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

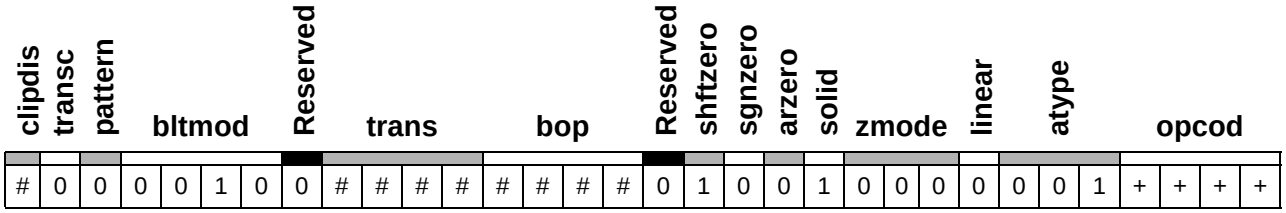
This type of line is initiated when the **DWGCTL** register's **opcod** field is set to either AUTOLINE_OPEN or AUTOLINE_CLOSE. Auto-init vectors ***cannot be used*** when the destination addresses are linear (**ylin** = 1).

➧ ***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:

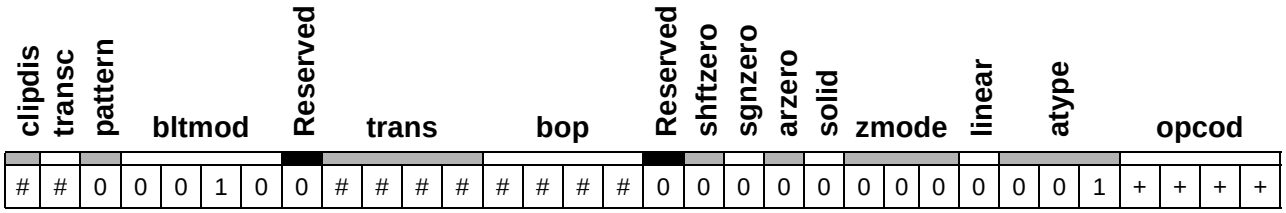


- **opcod**: must be set to LINE_OPEN, LINE_CLOSE, AUTOLINE_OPEN, or AUTOLINE_CLOSE

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

4.5.4.3 Lines That Use a Linestyle

DWGCTL:



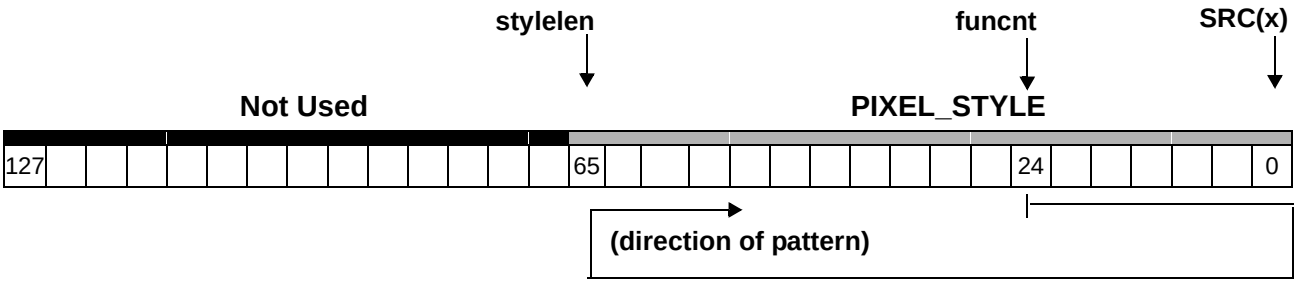
- **opcod**: must be LINE_OPEN, LINE_CLOSE, AUTOLINE_OPEN, or AUTOLINE_CLOSE

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, define the pattern you wish to use, then load it into the 128-bit source register (**SRC3-0**). Next, 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

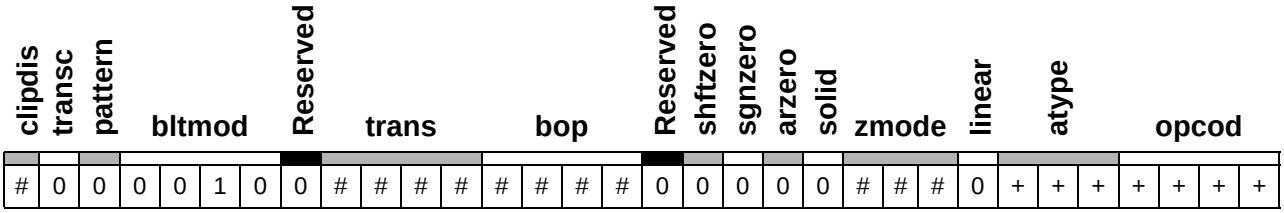
SHIFT : **stylelen** = 65, **funcnt** = 24
SRC0 : **srcreg0** = PIXEL_STYLE(31:0)
SRC1 : **srcreg1** = PIXEL_STYLE(63:32)
SRC2 : **srcreg2** = PIXEL_STYLE(65:64)



- 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:



- **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*
ALPHACTRL	**scrblendf**, **dstblendf**, **alphamode**, **astipple**, **alpha test mode**, **alphasel**	must set aten = '0'
ALPHASTART	The Alpha Start value	(1)
ALPHAXINC	The Alpha Increment on the major axis	(1)
ALPHAYINC	The Alpha Increment on the diagonal axis	(1)
DR0 (if zwidth = 0) **DR2_Z32LSB, DR2_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	"
DR3 (if zwidth = 0) **DR3_Z32LSB, DR3_Z32MSB** (if zwidth = 1)	The z diagonal increment	"
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
FOGSTART	The Fog factor Start value	only if **fogen** = 1
FOGXINC	The Fog factor Increment on major axis	"
FOGYINC	The Fog factor Increment on diagonal axis	"
FOGCOL	The Fog Color	"

(1) Whenever a function in the ALPHACTRL register requires them, Alphastart, alphaxinc, or alphayinc are programmed.

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 Global Initialization (All Operations) on page 4-31) 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.

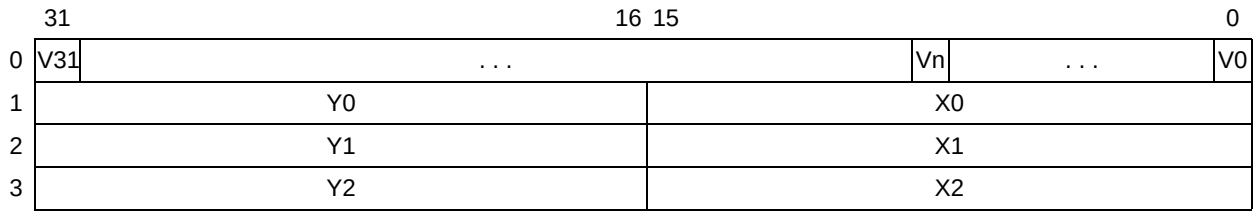
The *first* dword that is transferred is loaded into the Address Generator. This dword contains one bit of 'address select' for each of the next 32 vector vertices to be sent to the drawing registers. These 32 bits are called the vector tags. The next 32 dword transfers contain the xy address data to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



:	:	
n	Yn + 1	Xn + 1
:	:	
31	Y30	X30
32	Y31	X31
33	V31 . . .	Vn . . . V0
34	Y0	X0
35	Y1	X1
36	Y2	X2
:	:	

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.

➧ ***Note:*** In order to start the drawing engine, the last register programmed ***must*** be accessed in the 1D00h-1DFFh range.

4.5.5.1 Slope Initialization

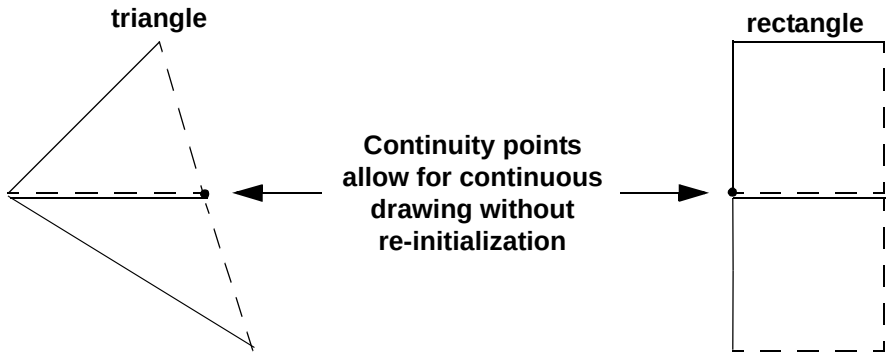
Trapezoids, rectangles, and span lines consist of a flat edge at the top and bottom, with programmable side edge positions at the left and right. When such a primitive is displayed, the pixels at the top and left edge are actually drawn as part of the object, while the bottom and right edges exist just 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.

➧ ***Note:*** For trapezoids with subpixel positioning, the **brkleft** bit in the **SGN** register and the **FXLEFT** must be programmed for the bottom-trap in case of a broken-left trapezoid.

Figure 4-6: Drawing Multiple Primitives

- **solid lines represent left, top edges**
- **dotted lines represent right, bottom edges**



Trapezoids

For trapezoid drawing, initialize the following registers:

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

For rectangle and span line drawing, the following registers must be initialized:

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:

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

- **trans**: if **atype** is BLK (block mode[(1)]), the transparency pattern is not supported - the value of **trans** must be '0000'
- **bop**: uses any Boolean operation if **atype** is RSTR; if **atype** is BLK, **bop** must be loaded with '1100'
- **atype**: can be 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 RSTR
  or
- **forcol**<31:24>, **forcol**<23:16>, **forcol**<15:8>, and **forcol**<7:0> are set to the same value

(1) 'Block mode' refers to the high bandwidth block mode function of SGRAM. It should be used whenever possible for the fastest performance, although certain restrictions apply (see the **atype** field of the **DWGCTL** register on page 3-125).

4.5.5.3 Patterned Trapezoids / Rectangle Fills

DWGCTL:

	clipdis	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	1	0	0
RECT	#	+	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 BLK, **bop** must be loaded with ‘1100’
- **atype**: Can be RSTR, or BLK
- **transc:** if **atype** is BLK, an opaque background is ***not*** supported - the value of **transc** must be ‘1’

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:

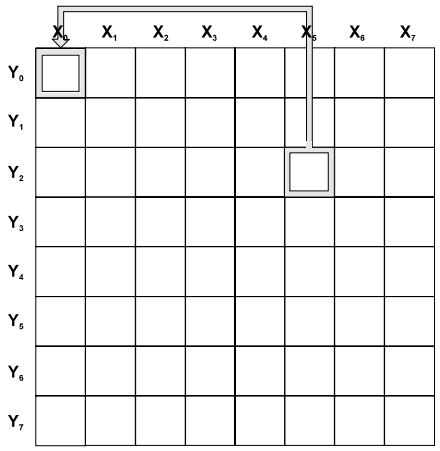
- **atype** is RSTR

  or

- **forcol**<31:24>, **forcol**<23:16>, **forcol**<15:8>, and **forcol**<7:0> are set to the same value, and **backcol**<31:24>, **backcol**<23:16>, **backcol**<15:8>, and **backcol**<7:0> are set to the same value.

Patterns and Pattern Offsets

Patterns can be comprised of one of two 8 x 8 pattern formats (Windows, or Little-Endian). If required, you can offset the pattern origin for the frame buffer within the register (if no offset is required, program the **shftzero** bit to ‘1’).



In the illustration on the left, the offset position is 5, 2. The corresponding register position’s value is moved to the starting point of the pattern array. (This starting point is equivalent to an offset of 0,0.) Refer to the examples on the next page for more details.

Screen Representation

The examples below show how the data stored in the pattern registers is mapped into the frame buffer. The numbers inside the boxes represent the register bit positions that comprise the pattern.

- Windows format (used to drive Microsoft Windows) stores the pattern in the **PAT0** and **PAT1** (page 3-128) 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 (page 3-153). 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:

	clipdis	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	0	0	0	#	#	#	0	+	+	+	0	1	0	0
RECT	#	0	0	0	0	0	0	0	#	#	#	#	#	#	#	#	0	1	1	1	0	#	#	#	0	+	+	+	0	1	0	0

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

Register	*Function*	*Comment / Alternate Function*
ALPHACTRL	**scrblendf**, **dstblendf**, **alphamode**, **astipple**, **alpha test mode**, **alphasel**	must set aten = '0'
ALPHASTART	The Alpha Start value	(1)
ALPHAXINC	The Alpha Increment on the x-axis	(1)
ALPHAYINC	The Alpha Increment on the y-axis	(1)
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	"
DR3 (if zwidth = 0) **DR3_Z32LSB, DR3_Z32MSB** (if zwidth = 1)	The z increment for y	"
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
FOGSTART	The Fog factor Start value	only if **fogen** = 1
FOGXINC	The Fog factor Increment on major axis	"
FOGYINC	The Fog factor Increment on diagonal axis	"
FOGCOL	The Fog Color	"

(1) Whenever a function in the ALPHACTRL register requires them, Alphastart, alphaxinc, or alphayinc are programmed.

➽ ***Note:*** When drawing Gouraud shaded trapezoids, the **MACCESS** register's **pwidth** field must *not* be set to 24 bits per pixel (PW24).

4.5.5.5 Texture Mapping

DWGCTL

	clipdis	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	0	0	0	#	#	#	0	+	+	+	0	1	1	0
RECT	#	0	0	0	0	0	0	0	#	#	#	#	#	#	#	#	0	1	1	1	0	#	#	#	0	+	+	+	0	1	1	0

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

Register	*Function*	*Comment / Alternate Function*
ALPHASTART	The Alpha Start value	see note
ALPHAXINC	The Alpha Increment on the x-axis	see note
ALPHAYINC	The Alpha Increment on the y-axis	see note
ALPHACTRL	**srcblendf**, **dstblendf**, **alphamode**, **astipple**, **alpha test mode**, **alphasel**	—
DR0 (if zwidth = 0) **DR0_Z32LSB, DR0_Z32MSB** (if zwidth = 1)	Z start position	Only if **zmode** <> NOZCMP or **atype** = ZI
DR2 (if zwidth = 0) **DR2_Z32LSB, DR2_Z32MSB** (if zwidth = 1)	Z increment for x	"
DR3 (if zwidth = 0) **DR3_Z32LSB, DR3_Z32MSB** (if zwidth = 1)	Z increment for y	"
DR4	Red start value	Only if modulate or decal is used
DR6	Red increment on x axis	"
DR7	Red increment on y axis	"
DR8	Green start value	"
DR10	Green increment on x axis	"
DR11	Green increment on y axis	"
DR12	Blue start value	"
DR14	Blue increment on x axis	"
DR15	Blue increment on y axis	"
FCOL	Alpha value	Only if **pwidth** = 32, or **pwidth** = 16 and **dit555** = 1.
FOGCOL	The Fog color	only if **fogen** = 1
FOGSTART	The Fog factor Start value	only if **fogen** = 1
FOGXINC	The Fog factor increment for x	"
FOGYINC	The Fog factor increment for y	"
SPECRSTART	Specular Red Start value	only if **specen** = 1
SPECRXINC	Specular Red Increment on major axis	"
SPECRYINC	Specular Red Increment on diagonal axis	"

Register	*Function*	*Comment / Alternate Function*
SPECGSTART	Specular Green Start value	"
SPECGXINC	Specular Green Increment on major axis	"
SPECGYINC	Specular Green Increment on diagonal axis	"
SPECBSTART	Specular Blue Start value	"
SPECBXINC	Specular Blue Increment on major axis	"
SPECBYINC	Specular Blue Increment on diagonal axis	"
TEXBORDERCOL	Texture Border Color	Only if **borderen** = 1
TEXCTL	Texel width, texture pitch, texture alpha key, texture alpha mask, palette select, decal with color key, **clampmode**, **tmodulate**, **strans**, **itrans, alpha overwrite and keying, alpha extend**	—
TEXCTL2	decalblend, idecal, decaldis, decalmod, ckstransdis, borderen, specen	—
TEXFILTER	minfilter, maxfilter, filteralpha, fthres, mapnb	—
TEXHEIGHT	Texture height, height mask, and round-up factor	—
TEXORG	Texture base address and origin of map 0	—
TEXORG1	Origin of map 1	Only when mip-mapping or planar mode is used.
TEXORG2	Origin of map 2	"
TEXORG3	Origin of map 3	Only when mip-mapping is used.
TEXORG4	Origin of map 4	"
TEXTRANS	Transparency color key, texture keying mask	—
TEXTRANSHIGH	Transparency color key high, texture keying mask high	—
TEXWIDTH	Texture width, width mask, and round-up factor	—
TMR0	s/wc increment for x	—
TMR1	s/wc increment for y	—
TMR2	t/wc increment for x	—
TMR3	t/wc increment for y	—
TMR4	q/wc increment for x	—
TMR5	q/wc increment for y	—
TMR6	s/wc start value	—
TMR7	t/wc start value	—
TMR8	q/wc start value	—

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

Note: If **twidth** = TW4 or TW8, the color palette ***must*** be initialized.

Note: Only clamp mode is supported when programming the **twmask** or **thmask** with a

value which is not a power of 2. The fields **tw** and **th** ***must*** be programmed with a "power-of-2" value but s/wc and t/wc can be scaled accordingly.

➧ ***Note:*** Each texture can be located in the frame buffer or in the system memory.

➧ ***Note:*** Whenever a function in the **ALPHACTRL** register requires them, **alphastart**, **alphaxinc**, **alphayinc** are programmed.

➧ ***Note:*** When using any mip_mapping filtering mode, the **thmask**, **twmask** fields ***must*** be programmed with a multiple of 16 - 1.

➧ ***Note:*** When using any mip_mapping filtering mode, the **tpitchext** field ***must*** be programmed with a multiple of 16.

4.5.5.6 Video Scaler

As a video scaler, the perspective effect must be disabled, this is done by programming the following register.

- TMR4 = 0x00000000
- TMR5 = 0x00000000
- TMR8 = 0x00010000

When used as a video scaler, texture engine quality depends on the filter selected. Bilinear filtering will give quality output at good speed. Mip-mapping will give high quality but with lower speed.

The texture engine supports both up and down scaling. The video source can be any format defined by texformat, but typically a video source will be one of these formats.

4.5.5.7 Video Scaler

When the selected filter includes *bilinear* filtering (a least 4 texels are needed to generate a pixel) and the texture width is *greater* than 512 texels, and the texel format is TW32 or TW422, the texture cache fills before the end of a texture line resulting in cache misses for the next line. Performances can be affected due to the extra memory bandwidth required.

One way to improve performance is to split the operation in two and keep the texels mapped at 512 or less for each TRAP. For the second TRAP, re-adjust the S/wc and T/wc parameters to continue scanning the texture where the first TRAP stopped. Use the same precision as that of the hardware for the second trap's S and T start values or artifacts could occur.

➧ ***Note:*** This technique may ***not*** always improve the drawing speed depending on memory bandwidth available at the time of the operation.

➧ ***Example:*** The texture is a 720x720 YUV422 image that is to be drawn with a 1:1 ratio.

FIRST TEXTURE_TRAP:

Tw	= log2(1024)
Th	= log2(1024)
Sstart	= 0
Sxinc	= 1/1024
Syinc	= 0
Tstart	= 0
Txinc	= 0
Tyinc	= 1/1024
YDST	= 0
LENGTH	= 720
FXLEFT	= 0
FXRIGHT	= 512

SECOND TEXTURE_TRAP:

Sstart	= 512 * Sxinc
Tstart	= 0
YDST	= 0
LENGTH	= 720
FXLEFT	= 512
FXRIGHT	= 720

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**
DSTORG	Origin of the destination	—
SRCORG	Origin of the same source surface	—
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**
SRCORG	Origin of the source surface	—
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**)

Note: **AR0** comprises 18 bits, therefore, a maximum of 256 Kpixels can be blitted.

Patterning Operations

Register	*Function*	*Comment / Alternate Function*
FXBNDRY	**FXLEFT** = destination boundary left-4 **FXRIGHT** = destination boundary 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**)
CXLEFT	Destination boundary left	—

4.5.6.2 Two-operand Bitblts

DWGCTL:

	clipdis	transc	pattern	bltmod				Reserved	trans				bop				Reserved	shftzero	sgnzero	arzero	solid	zmode				linear	atype			opcod			
XY	#	+	0	0	0	1	0	0	#	#	#	#	#	#	#	#	0	1	+	0	0	0	0	0	0	0	0	1	1	0	0	0
LIN.	#	+	0	0	1	1	1	0	#	#	#	#	#	#	#	#	0	1	1	0	0	0	0	0	1	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)

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: For *rolling blit* primitives AR5 (source pitch) must be programmed to ‘0’.

4.5.6.3 Color Patterning 8 x 8

DWGCTL:

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

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 + $(x\ offset - 4)_{mod8}$ + (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 4-42; 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

4.5.6.4 BitBlts With Expansion (Character Drawing) 1 bpp

DWGCTL:

clipdis	transc	pattern	bltmod	Reserved	trans	bop	Reserved	shftzero	sgnzero	arzero	solid	zmode	linear	atype	opcod
#	+	0	+	+	+	+	0	#	#	#	#	#	#	#	#	0	1	1	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 BLK, ***must*** be loaded with '1100'
- **bltmod:** can be BMONOLEF ***or*** BMONOWF
- **atype**: can be RSTR ***or*** BLK
- **transc:** if **atype** is BLK, an opaque background is ***not*** supported - the value of **transc** must be '1'

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 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.

4.5.6.5 BitBlts With Expansion (Character Drawing) 1 bpp Planar

DWGCTL:

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

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 4-2: ILOAD Source Size

bltmod	pwidth	psiz
BFCOL	PW8	8
	PW16	16
	PW24	24
	PW32	32
BMONOLEF	—	1
BMONOWF	—	1
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 (as in: **ylin** = 1, see **PITCH**)

4.5.7.2 ILOAD of Two-operand Bitblts

DWGCTL:

clipdis	transc	pattern	bltmod	Reserved	trans	bop	Reserved	shftzero	sgnzero	arzero	solid	zmode	linear	atype	opcod
#	+	0	+	+	+	+	0	#	#	#	#	#	#	#	#	0	1	+	0	0	0	0	0	+	0	0	1	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, BU32BGR, BU32RGB, BU24BGR, or BU24RGB.
- **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’

	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 4-3 shows all the valid format combinations. The structure of the buffers to be transferred is defined for each data format (as shown in the 'Pixel Formats' illustrations starting on page 4-8).

Table 4-3: ILOAD Supported Formats

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

4.5.7.3 ILOAD with Expansion (Character Drawing)

DWGCTL:

clipdis	transc	pattern	bltmod	Reserved	trans	bop	Reserved	shftzero	sgnzero	arzero	solid	zmode	linear	atype	opcod
#	+	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 BLK, **bop** must be loaded with ‘1100’
- **atype**: RSTR, or BLK
- **transc:** if **atype** is BLK, an opaque background is ***not*** supported - the value of **transc** must be ‘1’

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 4-4 shows all the valid format combinations. The structure of the buffers to be transferred is defined for each data format (as shown in the ‘Pixel Formats’ illustrations starting on page 4-8).

Table 4-4: Bitblt with Expansion Supported Formats

Processor Type	bltmod	dmaDataSiz	*Data Format*
Little-Endian	BMONOLEF	‘00’	MONO A
	BMONOWF	‘00’	MONO B
Big-Endian	BMONOWF	‘00’	MONO C

4.5.8 Loading the Texture Color Palette

This operation is similar to a normal BITBLT or ILOAD operation. In this case, a source texture color palette is loaded into the destination 256 x 1 x 16 bpp LUT (Look-Up Table) that is used in texture mapping. Any portion of the LUT can be programmed independently.

This color palette is used to perform color expansion during texture mapping when textures are in 4 or 8 bpp format. When 4 bpp textures are used, 16 palettes are available in the LUT. The choice of the palette used to do color expansion is determined by the **palsel** field of the **TEXCTL** register.

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

Register	*Function*	*Comment / Alternate Function*
AR0	Source end address	—
AR3	Source start address	—
PITCH	**iy** = 1024	—
YDSTLEN	**yval**: Start position in the LUT (0 to 255) **length**: Number of locations to fill in the LUT (1 to 256)	—
SRCORG	Origin of the source	For BLIT only
YDSTORG	0	—
FXBNDRY	0	—
MACCESS	**pwidth** = PW16, **tlutload** = 1	—
OPMODE	**dmamod** = 01, **dmadatsiz** = 00 (Little-Endian) **dmadatsiz** = 01 (Big-Endian)	For ILOAD only

Note: The **PITCH** and **MACCESS** registers are not normally modified during drawing operations, and may have to be re-programmed after this operation in order to return the drawing engine to its original state.

4.6 CRTC Programming

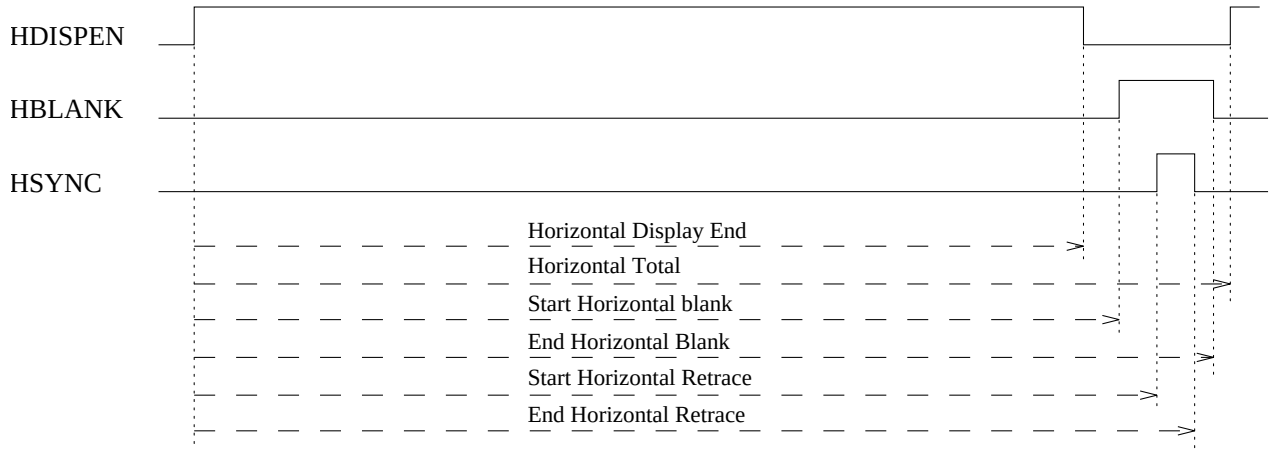
The CRTC can be programmed in one of two modes: VGA Mode or Power Graphic Mode. The **mgamode** field of the **CRTCEXT3** register is used to select the operating mode.

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 4-7: 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'.

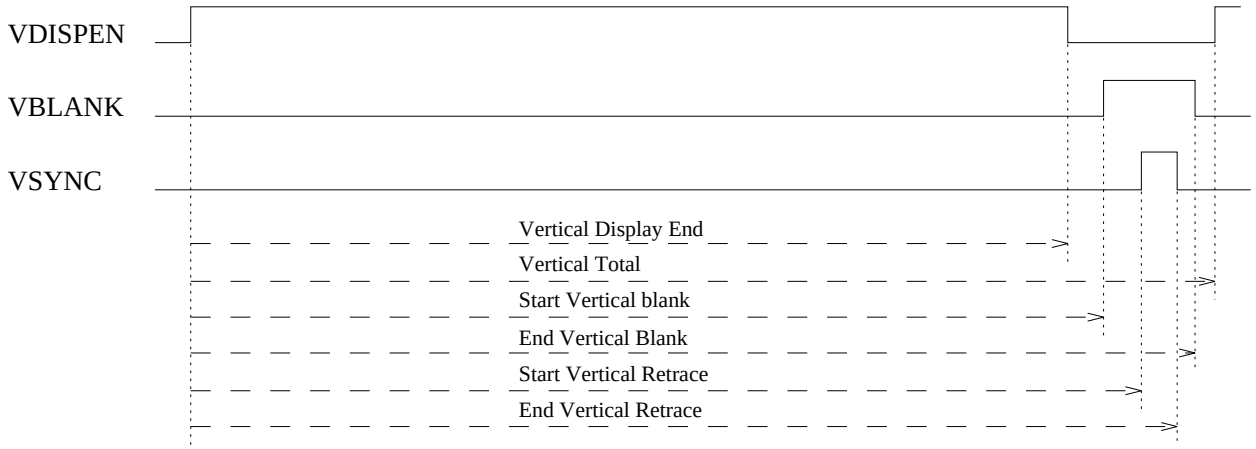
The units of the horizontal counter are *character clocks* for VGA Mode, or 8 pixels in Power Graphic Mode. The **scale** field of the **CRTCEXT3** register is used to bring the VCLK clock down to an *8 pixel* clock.

The scale factor settings are shown in the following table:

Bits/Pixel	scale
8	'000'
16	'000'
24	'010'
32	'001'

4.6.2 Vertical Timing

Figure 4-8: 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.
curloc<15:0>	Cursor location
prowscan<4:0>	Preset row scan
maxscan<4:0>	Maximum scan line
currowstr<4:0>	Row scan cursor begins
currowend<4:0>	Row scan cursor ends
curoff<4:0>	Cursor off
undrow<4:0>	Horizontal row scan where underline will occur
curskew<1:0>	Cursor skew control

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

In Power Graphic Mode, the following fields are extended in order to support both a larger display, and up to 16 megabytes of memory.

startadd<20:0>	Start address.
offset<9:0>	Logical line width of the screen. This is programmed with the number of double slices in one display line.

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

4.6.4 Programming in VGA Mode

The VGA CRTC of the MGA-G200 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-G200'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:

$$\textbf{offset} = \frac{\text{videopitch} \times \text{bpp} \times \text{fsplit}}{128}$$

Where:

bpp is the pixel width, expressed in bits per pixel

videopitch is the number of pixels per line in the frame buffer (including pixels that are not visible

fsplit = 2 in split mode; 1 in all other modes

For example, 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	*multiple of*
8	16
16	8
24	16
32	4

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

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

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

bpp	*factor*
8	8
16	4
24	8
32	2

For example, to program **startadd** to use an offset of 64 with a 16 bit/pixel frame buffer, **startadd** = 64/4 = 16. With a 24 bit/pixel frame buffer, **startadd** = 64/8 = 8.

➧ ***Note:*** 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.

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.

There is no overscan in Power Graphic mode, therefore:

htotal + 5 == **hblkend** +1
hdispend + 1 == **hblkstr** +1

The End Horizontal Blank value must always be greater that **hsyncstr** + 1, so that the start address latch can be loaded before the memory address counter.

A composite sync (block sync) can be generated on the HSYNC pin of the chip if the **csyncen** field of the **CRTCEXT3** register is set to ‘1’. The VSYNC pin will continue to carry the vertical retrace signal.

➧ ***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 using the hzoom field of the **XZOOMCTRL** register. To implement the zoom function, pixles are duplicated within the DAC, and the memory address generator advances at a reduced rate.

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

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

***Limitations*:**

- **htotal** ***must*** be greater than 0 (because of the delay registers on the **htotal** comparator)
- **htotal** - **hdispend** must be greater than 0
- In interlace mode, **htotal** must be equal to or greater than **hsyncend** + 1
- **htotal** - **bytepan** + 2 must be greater than **hdispend**
- **hsyncstr** must be greater than **hdispend** + 2
- **vtotal** ***must*** be greater than 0 (because of the delay registers on the **vtotal** comparator)
- In interlace mode, **vtotal** must be an even number
- In HZOOM = 00, (Horizontal Total) MOD 8 ***must not*** be ‘7’
- In HZOOM = 01, (Horizontal Total) MOD 16 ***must not*** be ‘15’
- In HZOOM = 1X, (Horizontal Total) MOD 32 ***must not*** be ‘31’

CRTC Latency Formulas

This section presents several rules that must be followed in Power Graphic Mode in order to adhere to the latency constraints of the 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. (VC) (Tpixclk) >= Tmclk
2. ((8) (**htotal** - **hsyncstr** +1) +3(VC)) (Tpixclk) >= (124) (Tmclk)
3. ((7 - **hiprilvl**) (VC) (8) + ((1)VC - 1)) (Tpixclk) >= (MP) (Tmclk) + (11) (VC) (Tpixclk)

MAXHIPRI = MIN(HIPRILVL, (**SCALE** + 1)(Tmclk / Tpixclk)) (round to the nearest whole number)

where:

SCALE = value programmed into the **SCALE** register.
VC = 8 /DF (which is the number of pixels per slice)
8 in BPP8
4 in BPP15 / BPP16
8/3 in BPP24 packed pixel
2 in BPP32PAL / BPP32DIR

Tmclk = The period of MCLK in ns

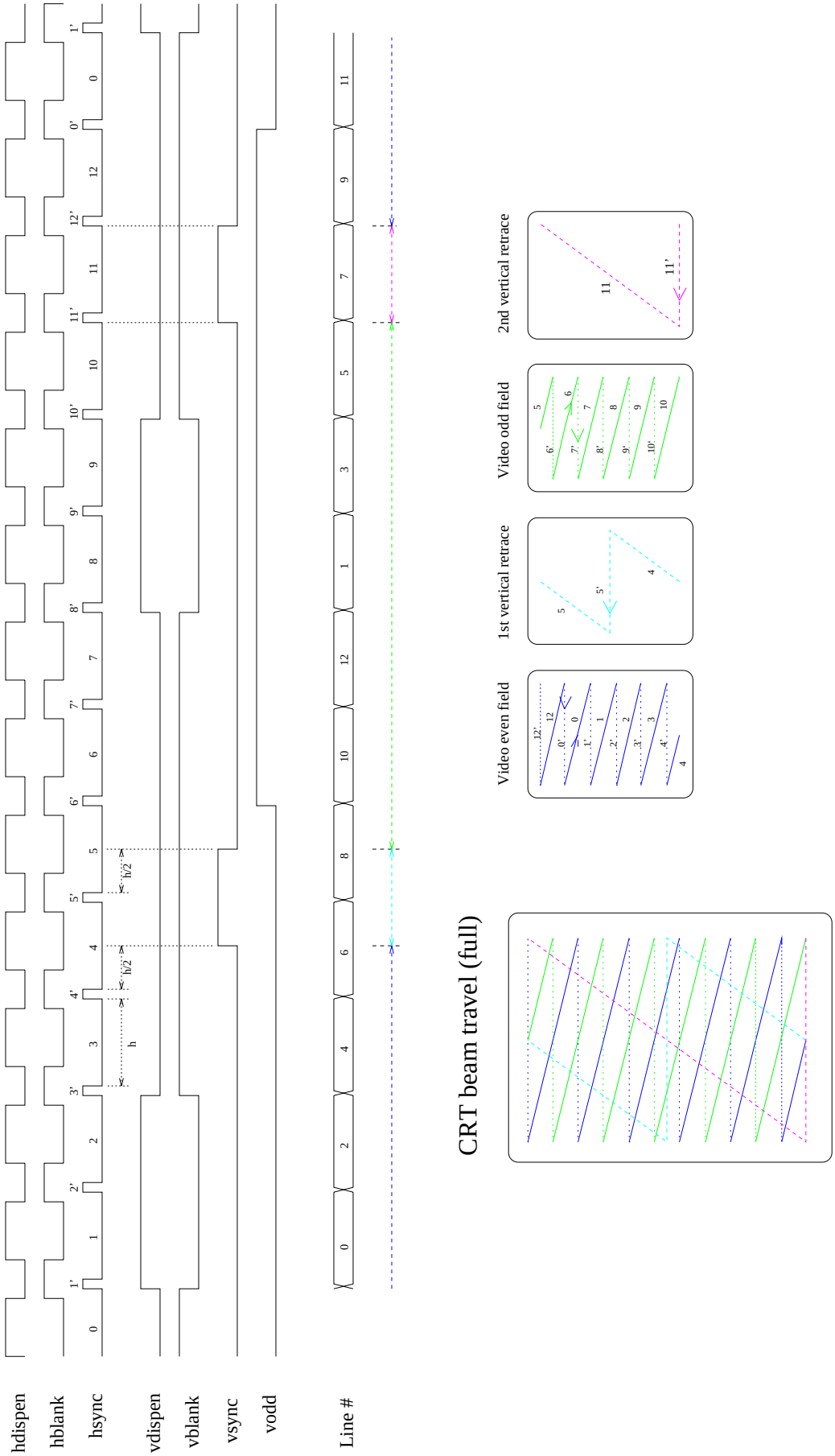
Tpixclk = The period of the pixel clock in ns

MP = Memory Controller Pipe Depth

strmfctl = The Streamer Pipe Blocking Field

if	*MP with CODEC or* **VIN** *operating*	*MP without CODEC or* **VIN** *operating*
strmfctl = 2	32	32
strmfctl = 1	41	41
strmfctl = 0	59	52

Figure 4-9: Video Timing in Interlace Mode



4.7 Video Interface

4.7.1 Operation Modes

The MGA-G200's RAMDAC can operate in one of five modes, depending on the values of the **mgamode** field of the **CRTCEXT3** register and the **depth** field of the **XMULCTRL** RAMDAC register, as shown below:

mgamode	depth	*Mode Selected*
0	000	VGA
1	000	Pseudo Color (BPP8)
1	001	True Color (BPP15)
1	010	True Color (BPP16)
1	011	True Color (BPP24)
1	100	Direct Color (BPP32DIR)
1	101	True Color (BPP32PAL)

4.7.1.1 VGA Mode

In VGA mode, the data to be displayed comes from the MGA-G200's VGA Attribute Controller (ATC). The data from the ATC is used as an address for the three-LUT RAM. The pixel read mask can be applied to the data before it passes through the palette. The hardware cursor and keying functions are not supported in VGA mode.

Red LUT	P7	P6	P5	P4	P3	P2	P1	P0
Green LUT	P7	P6	P5	P4	P3	P2	P1	P0
Blue LUT	P7	P6	P5	P4	P3	P2	P1	P0

- The frequency of the pixel PLL is determined by the **clksel** field of the VGA **MISC** register, which selects the proper set of registers for the PLL. The frequency can also be changed via the **XPIXPLLM**, **XPIXPLLN**, and **XPIXPLLP** registers.
- Horizontal zooming is not supported in VGA mode.

4.7.1.2 Pseudo Color Mode

In Pseudo Color mode (BPP8), the data from the memory is sent to the RAMDAC's internal FIFO via the memory controller. The data is then used as an address for the three-LUT RAM (as in VGA mode). A hardware cursor is available.

- The pixel PLL should use register set 'C' register set, so as to not change the frequency of the VGA PLL sets.
- Horizontal zooming is supported in pseudo color mode.

4.7.1.3 True Color Mode

The four true color modes supported by MGA-G200 are BPP15, BPP16, BPP24, and BPP32PAL. In these modes, the pixel data from the internal RAMDAC's FIFO is mapped to the LUT addresses as shown in the following illustrations:

15-Bit True Color (BPP15)

Red LUT	P15	0	0	P14	P13	P12	P11	P10
Green LUT	P15	0	0	P9	P8	P7	P6	P5
Blue LUT	P15	0	0	P4	P3	P2	P1	P0

- Bit 15 can be used as an overlay color (it selects another LUT table in the RAM) and can be masked out by the **alphaen** field of the **XGENCTRL** register (when at '0').

16-bit True Color (BPP16)

Red LUT	0	0	0	P15	P14	P13	P12	P11
Green LUT	0	0	P10	P9	P8	P7	P6	P5
Blue LUT	0	0	0	P4	P3	P2	P1	P0

24-bit True Color (BPP24 and BPP32PAL)

Red LUT	P23	P22	P21	P20	P19	P18	P17	P16
Green LUT	P15	P14	P13	P12	P11	P10	P9	P8
Blue LUT	P7	P6	P5	P4	P3	P2	P1	P0

In BPP24, the pixel data in the FIFO is unpacked before it enters the pixel pipeline, since each slice contains 2 2/3 24-bit pixels. In BPP32PAL, each pixel is 32 bits wide, but the eight MSBs are not used since they do not contain any color information.

- The hardware cursor and horizontal zooming are supported.
- Register set 'C' should be used to program the pixel PLL.

4.7.1.4 Direct Color Mode (BPP32DIR)

In direct color mode, each pixel in the FIFO is composed of a 24-bit color portion and an 8-bit alpha portion. The 24-bit portion is sent directly to the RAMDAC (that is, each color is directly applied on each DAC input). The alpha portion of the pixel can be used for color keying (refer to the **XCOLKEY** register description) and may be displayed as a pseudo color pixel, depending on the outcome of the color comparison.

- As in all non-VGA modes, the hardware cursor and horizontal zooming are available.
- Register set 'C' should be used to program the pixel PLL.

4.7.2 Palette RAM (LUT)

The MGA-G200's RAMDAC uses three 256x8 dual-ported RAMs for its color LUT. The use of a dual-ported RAM allows for asynchronous operation of the RAM, regardless of the current display state. The RAM is addressed by an 8-bit register/counter (**PALWTADD**) and selection among the three LUTs is done using a modulo 3 counter.

To write the red, green, and blue components of a pixel to a location in the RAM, three writes to the **PALDATA** RAMDAC register must occur. Each byte will be transferred to the RAM when it is written. The modulo 3 counter will track the color being written. When the last byte (the blue component) of a RAM location is written, the address register is incremented, the modulo 3 counter is cleared, and the circuit is ready to write the red component of the next location. This allows the entire RAM to be updated with only one access to the **PALWTADD** register.

To read a complete location in the palette RAM, three reads of the **PALDATA** RAMDAC register must

occur. The palette address register will then be incremented to the next location. As with writes, the RAM can be completely read with only one write to the **PALRDADD**.

Note: When changing the **ramcs** bit of the **XMISCCTRL** RAMDAC register, the pixel clock ***must*** be disabled (that is, **pixclkdis** = '1').

4.7.3 Hardware Cursor

A hardware cursor has been defined for all non-VGA modes. This cursor will be displayed over any other display information on the screen, either video or graphics.

The cursor position is relative to the end of the blanking period. Refer to the **curposx** and **curposy** field descriptions (**CURPOS**). The cursor is not zoomed when horizontal and/or vertical zooming is selected. The cursor pattern is stored in the off-screen memory of the frame buffer at the location defined by the **XCURADDH** and **XCURADDL** RAMDAC registers. In Big Endian mode, the cursor pattern must be swapped according to section 4.1.7 before being written to the frame buffer.

The **CURPOSX** and **CURPOSY** registers are double-buffered (that is, they are updated at the end of the vertical retrace period). They ***must not*** be programmed when the **vsyncsts** field of the **STATUS** register is '1'. (They can be updated at any other time.) The **XCURADDH** and **XCURADDL** registers are ***not double buffered***, so changes to this register may produce unwanted artifacts on the screen.

In interlaced mode, if the cursor Y position is greater than 64, the first line of the cursor to appear on the screen will depend on the state of the internal field signal.

- If the value of **curposy** is an odd number, the data for row 0 of the cursor will be displayed in the odd field. Rows 2, 4, ... 62 will then be displayed on the subsequent lines. The data for row 1 of the cursor will be displayed in the even field, followed by rows 3, 5, ... 63.
- If the value of **curposy** is an even number, the data for row 0 of the cursor will be displayed in the even field. Rows 2, 4, ... 62 will then be displayed on the subsequent lines. The data for row 1 of the cursor will be displayed in the odd field, followed by rows 3, 5, ... 63.
- If the value of **curposy** is less than 64, the cursor is partially located off the top of the screen. The first cursor row (row N) to be displayed will always be on scan line 0, which is the first line of the even field, and therefore the topmost scan line of the screen. Rows N+2, N+4, and so on will follow. The data in cursor row N+1 will be displayed on the first line of the odd field, followed by row N+3, N+5, and so on.

In order for the cursor to function properly, the following rules must be respected:

Hblank_width (ns) >= 7 * Tmclk + 48 * Tmclk where:
Tmclk=MCLK cycle time (ns)

4.7.4 Keying Functions

Color keying can occur in any non-VGA color depth and resolution. The color key comparison occurs on the index of the palette. The corresponding color key and mask ***must*** match the pixel format used by the RAMDAC. These formats are shown in section 4.7.1.2 - 4.7.1.4. Refer to the **XCOLKEY** and **XCOLMSK** sets of registers for more details and to the Backend Scaler Programmer's Guide, section 4.10.4.

In True Color BPP15 (1:5:5:5), the alpha bit can be disabled or enabled by using the ALPHAEN field of the XGENCTRL register.

In Direct Color BPP32DIR (RGB-alpha), color keying comparisons canbe done on the alpha byte and on the RGB bytes. The order of precedence is as follows:

```
if (alpha byte <> XCOLKEY) then
    display ‘alpha-byte’;
elseif (RGB=XCOLKEY0) then
    display ‘Back End Scaler Video’;
else
    display ‘RGB direct’;
end if;
```

The keying on the alpha byte is not restricted to the Back End Scaler Video Window. Refer to the **XCOLKEY** and **XCOLMSK** registers for more details.

4.7.5 Zooming

Horizontal zooming is achieved by changing the **hzoom** field of the **XZOOMCTRL** register. The CRTC Memory Address Counter clock will automatically be changed accordingly. No other CRTC register need be changed. The supported zoom factors are x1 (no zoom), x2, and x4.

Vertical zooming is performed by the CRTC and nothing need be done in the RAMDAC section of the MGA-G200.

4.7.6 Feature Connector

The MGA-G200’s MAFC (Matrox Advanced Feature Connector) Video Output Port is supported for all non-VGA display modes upto a dot clock of 65 MHz. The MAFC Port is 12 bits wide and also provides **VVSYNC/, VHSYNC/, VOBLANK/ and VDOCLK** signals depending on the mode of operation. All of the connector pins ca be disabled (reset to zero) except for the sync signals which must be active for the monitor.

The following table describes the output from the pins of the MAFC Video Output Port.

Table 4-5: MAFC Video Output Port Pins

Operating Mode	*VDOUT*	*VDOCLK*	*VOBLANK/*
MAFC12	Multiplexed 12 bit data 12 bits per clock edge dual edge	Input clock from master device	Gated input clock with BLANK signal
PANELLINK	Multiplexed 12 bit data 12 bits per clock edge dual edge	MGA-G200 drives PIXEL clock	BLANK signal
BYPASS656	8 bit data on the LSB portion of the data bus	Not used	Blank signal

When the MGA-G200 is in PANELLINK or MAFC12 output mode, the data sent through the VDOUT data bus, is taken before the DACs. When the MAFC Port is disabled, the VDOUT bus, VDOCLK signal and the VOBLANK/ signal output is ‘0’.

For more details, refer to the **mfcsel** and **vdoutsel** fields of the **XMISCCTRL** register.

4.7.7 Test Functions

A 16-bit CRC is provided to verify video integrity at the input o f the DACs. The CRC can be read via the **XCRCREMH** and **XCRCREML** registers when the vertical sync is active. The CRC is cleared at the end of the vertical retrace period, and calculated only when the video is active. The **crcsel** field determines which of the 24 bits will be used in the calculation.

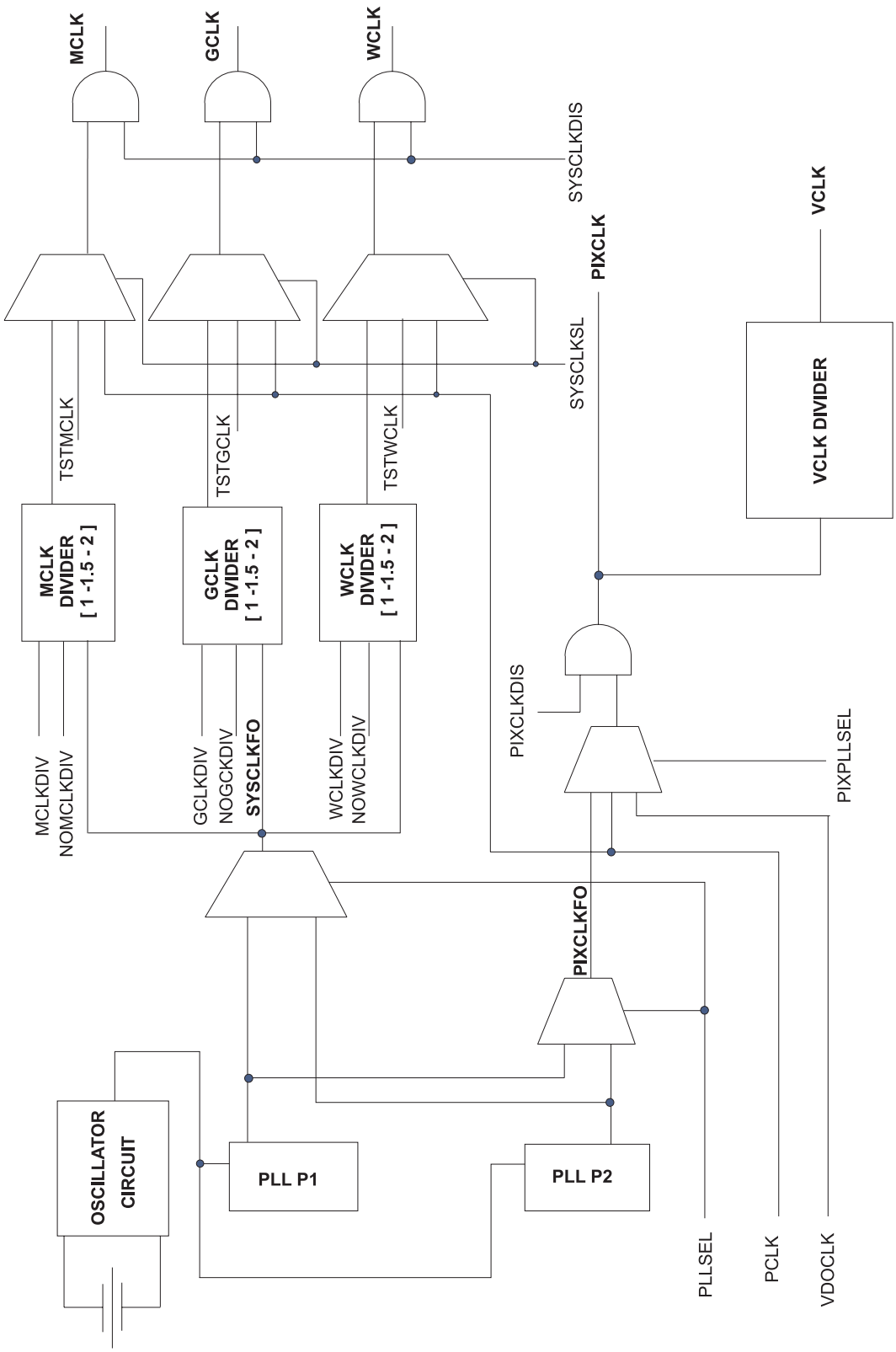
The output of the sense comparator can be read via the **XSENSETEST** RAMDAC register. This provides a means to check for the presence of the CRT monitor and determine if the termination is correct. The sense bits are latched at the start of the blanking interval. In order to ensure a stable value at the input of the comparator, the input of the DACs should remain constant during the visible display period. The sense amplifiers can be powered down by setting the **sensepdN** bit to ‘0’.

4.7.8 PLL Clock Generators

The MGA-G200's RAMDAC has two independent programmable Phase Locked Loops (PLLs), named P1 and P2. These PLLs are used as frequency sources for clock generation. One source (SYSTEM PLL) is used for **GCLK**, **MCLK** and **WCLK**. The second source (PIXEL PLL) is used to generate PIXCLK and VCLK. Either PLL can be used as source through the **pllsel** field fo the **OPTION** register. Therefore a reference to the SYSTEM PLL or the PIXEL PLL will mean the PLL chosen through the **pllsel** field, to be the source of that clock.

GCLK	=	Graphics Engine Clock
MCLK	=	Memory Clock
PIXCLK	=	Ramdac Clock
VCLK	=	CRTC Clock
WCLK	=	Warp Engine Clock

Figure 4-10: Clock Division Scheme



4.7.8.1 System PLL

The system PLL is programmed through the **XSYSPLLM**, **XSYSPLLN** and **XSYSPLLP** registers. The frequency of the Voltage Controlled Oscillator (VCO) is defined by:

Fvco = Fref * (**XPIXPLLN** + 1) / (**XSYSPLLM** + 1)

Where Fref = 14.31818 MHz.

7 <= N <= 127 (feedback divider)

1 <= M <= 6 (input divider)

P = {0,1,3,7} (post-divider)

0 <= S <= 3

The PLL output frequency is then:

Fo = Fvco / (**SYSPLLP** + 1)

On reset, the system PLL is bypassed and the system clock is derived from the PCI bus clock. This permits the MGA-G200 to boot-up properly. The system PLL resets to its oscillating frequency when the **syspllpdN** bit is set to '1'.

The memory clock (MCLK) can be selected to be the PCI bus clock, the MCLK pin, or the system PLL clock output. The GLCK and WCLK can be (independently of each other) the PCI bus clock or the system PLL. All clocks also have another possibility but this is for testing only.

Although all the system clocks share a single clock source, they all have independently controlled clock dividers on the SYSTEM PLL frequency. the divider works according to the following tables.

NOMCLKDIV	*MCLKDIV*	*MCLK*
'1'	'X'	SYSTEM PLL frequency
'0'	'0'	1/2 * SYSTEM PLL frequency
'0'	'1'	2/3 * SYSTEM PLL frequency

It works the same for the GCLK and the WCLK.

➽ ***Note:*** Refer to the **SYSCLKSL**, and division fields of **OPTION** and **OPTION2** registers for more details.

The system clocks can be gated off when SYSCLKDIS is '1', when reprogramming the SYSTEM PLL (see section 5.7.8.3). Power consumption can be reduced by programming syspllpdN to '0'. This will shut down the SYSTEM PLL however, all memory contents will be lost.

4.7.8.2 Pixel PLL

The pixel PLL contains three independent sets of registers: sets A, B, and C. The **clksel** field of the VGA MISC register will determine which set will define the operating frequency of the pixel PLL (see the pixpllan register description). The frequency of the Voltage Controlled Oscillator (VCO) is defines by the following formula:

Fvco = Fref * (**XPIXPLLN** + 1) / (**XPIXPLLM** + 1)

Where Fref = 14.31818 MHz.

7 <= N <= 127 (feedback divider)

1 <= M <= 6 (input divider)

P = {0,1,3,7} (post-divider)

0 <= S <= 3

The PLL output frequency is then:

Fo = Fvco / (**XPIXPLLP** + 1)

On reset, the pixel clock (PIXCLK) is generated from the PCI bus clock. The pixel PLL will run with the register set that is selected by the **clksel** field when the **pixpllpdN** field is set to '1'. After a reset, **clksel** is '00', so the pixel PLL will oscillate at 25.175 MHz and VCLK will be the same frequency (since the DAC wakes up in VGA mode).

The video clock (VCLK) is function of the display mode of the DAC:

mgamode	depth	*Video Clock*
0	xxx	PIXCLK
1	000	PIXCLK/8
1	001	PIXCLK/4
1	010	PIXCLK/4
1	011	PIXCLK*8/3
1	100	PIXCLK/2
1	101	PIXCLK/4
1	110	PIXCLK/2
1	111	PIXCLK/2

The maximum supported pixel clock frequency is 250 MHz (1600 x 1200 resolution @ 85 Hz. The minimum period of the VCLK signal is 14.8 ns (1280 x 1024, 24-bit packed pixel at a 75 Hz vertical refresh rate).

The pixel clock can obtain its source from three different places: the Pixel PLL (normal operation), the PCI bus clock (at boot-up), or the VDOCLK pin The selection is done via the **pixclksl** field of the **XPIXCLKCTRL** RAMDAC register. PIXCLK and VCLK can also be shut off by setting the **pixclkdis** bit to '1'. Again, as for the system PLL, the pixel PLL can be powered down by resetting the **pixpllpdN** bit to '0' to lower power consumption.

4.7.8.3 Programming the PLLs

To change the frequency of one of the PLLs or the source of a clock, the following procedure ***must*** be followed:

(A) Changing the Pixel Clock Frequency or Source

To program any of the **XPIXPLLM**, **XPIXPLLN**, **XPIXPLLP**, or **XPIXCLKCTRL** registers, the memory clock ***must*** be running and enabled (**sysclkdis** = '0').

1. Force the screen off.
2. Set **pixclkdis** to '1' (disable the pixel and video clocks).
3. Re-program the desired pixel PLL registers by changing the values of the registers, by changing the **clksel** field of the VGA **MISC** register, or by selecting another source for the pixel clock.
4. Wait until the clock source is locked onto its new frequency (the **pixlock** bit is '1') for the pixel PLL, or for the **VDCLK** pin to become stable.
5. Set **pixclkdis** to '0' (enable the pixel and video clocks).
6. Resume normal operations (re-enable the screen display).

No special procedures need to be followed when changing the frequency of the video clock since the MGA-G200's hardware will not generate glitches on the video clock when the **mgamode** or **depth** fields are changed.

(B) Changing the System PLL Frequency

Special care must be taken when changing the frequency of the system PLL. Since the **XSYSPLLM**, **XSYSPLLN**, and **XSYSPLLP** registers are clocked on the memory clock, the system PLL must always be running.

1. Set **sysclkdis** to '1' (disable the system clocks).
2. Select the PCI bus clock for the system clocks (**sysclksl** = '00').
3. Set **sysclkdis** to '0' (enable the system clocks).
4. Re-program the desired system PLL registers.
5. Wait until the **syslock** bit is '1'.
6. Set **sysclkdis** to '1' (disable the system clocks).
7. Select the system PLL clock for the system clocks (**sysclksl** = '01').
8. Set **sysclkdis** TO '0' (enable the system clocks).
9. Resume normal operations.

(C) Changing the System Clock Source, MCLK, GCLK or WCLK Division Factor

1. Set **sysclkdis** to '1' (disable the system clocks).
2. Select the new clock source or change the **mclkdiv** and/or **gclkdiv** and/or **wclkdiv** fields. Make sure that the new clock source is stable before continuing.
3. Set **sysclkdis** to '0' (enable the system clocks).
4. Resume normal operations.

➽ ***Note:*** Steps (B) and (C) must be executed in an order which keeps MCLK and GCLK within their specified values.

DAC external components:

The magnitude of the full scale current can be controlled by a resistor using the following calculation:

R (ohm) = K * 1000 * REF(V) / Iout (mA)

Pedestal	*K factor*	
	With sync	*No sync*
7.5 IRE	3.415	2.439
0.0 IRE	3.231	2.255

This resistor should be placed between the RSET pin and the analog GND. A 0.1 uF capacitor should be placed between the COMP pin and the analog VDD. The voltage applied to the Vref pins is 1.235 V.

4.8 Video Input Interface

4.8.1 Overview of the Video-Grabber

The MGA-G200's field based Video-Grabber captures the incoming video data and writes it into the frame buffer. There are two sets of registers that act as a double-buffered set: one may be active during a field while the other is programmed. VBI data, either raw or decoded, may also be captured and written to the framebuffer. Active video data in 4:2:2 format, may be written directly into the frame buffer. The Video-Grabber works in a 'one-shot' mode. Software needs to program for every field that needs to be captured.

If both even and odd fields are desired, the pitch of both windows (**vinpitch0** and **vinpitch1**) should be set to twice the anticipated line width, and the start address (**vinaddrX**) of the second window should be set to a value of 1 line width higher that the first windows's start address.

4.8.2 MAFC Mode Selection

See the **XMISCCTRL** register to allow video in data to be driven back out the **VDOUT**(7:0) pins. The video-in data is registered once with **VDCLK**, so there will be a one cycle delay between the input data and the output data. The Video In interface does not have to be enabled in order for this pass-through mode to work, but it can be enabled if the stream is desired to be captured.

4.8.3 Programming sequence

Since the Video-Grabber is a field based grabber, the sequence is the same for all types of captures: odd only, even only, both odd and even, and VBI captures.The grabber registers are programmed between **vsync** for capture of the field following the next **vsync**.

> ***Note:*** The registers for window0 cannot be reprogrammed while window0 is active, and registers for window1 cannot be reprogrammed while window1 is active. It is legal to reprogram window1 registers when window0 is active, and reprogram window0 registers when window1 is active. If a particular window's registers are reprogrammed while that window is active, then data corruption will occur.

Before Programming the Video-Grabber:

Reset the Video In interface using the **VINCTL** register.

Programming steps:

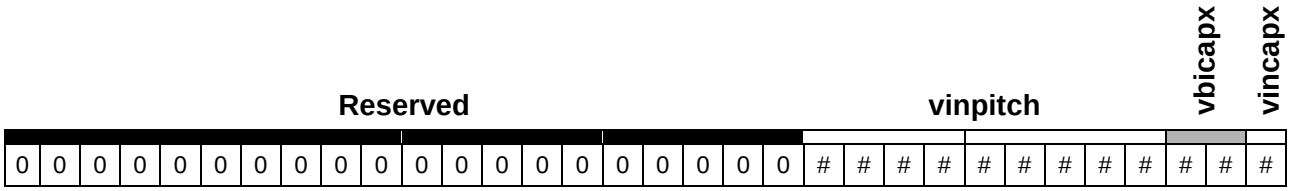
Step 1. Clear the **vinvsyncpen** flag in the **VSTATUS** register to clear the previous vsync status.

Step 2. Enable the video input vsync interrupt (**vinvsyncien**).

Step 3. At the next **vsync** interrupt read the **VSTATUS** register. If the **vinvsyncpen** bit is active clear the flag like in step 1. If the completed **vinfielddetd** bit indicates the field desired to capture go to step 4. Otherwise repeat this step.

Step 4. Program all the Video In window registers ‘0’ or ‘1’ related to video capture.

VINCTLX



Register	*Function*	*Comments/Alternate Function*
VBIADDRX	VBI Write Address	if **vbicapx** is not ‘00b’
VINADDRX	Video Write Address	if **vincapx** is ‘1’

Step 5. Program the **vinnextwin** bit in the **VINNEXTWIN** register to select the same window that was chosen in Step 4.

Step 6. If another field is desired following the field just programmed go to step 3, otherwise disable interrupts.

4.9 CODEC Interface

A CODEC can be used in conjunction with the MGA-G200 chip to compress and decompress a video stream in real time.

➽ ***Note:*** When programming **CODECHOSTPTR** for compression/decompression, the Codec Interface will *not* stop transferring data when the PTR value is reached. Software should suspend the Codec Interface's memory accesses until more data is put into memory (or more space is available) by setting **codectransen** of **CODECCTL** to a '0'

4.9.1 Memory Organization

Three main sections of memory are reserved for Codec Interface usage. The **CODECADDR** register is used to set the location of the buffer in the off-screen memory area. Table 4-11 shows the organization of the CODEC interface

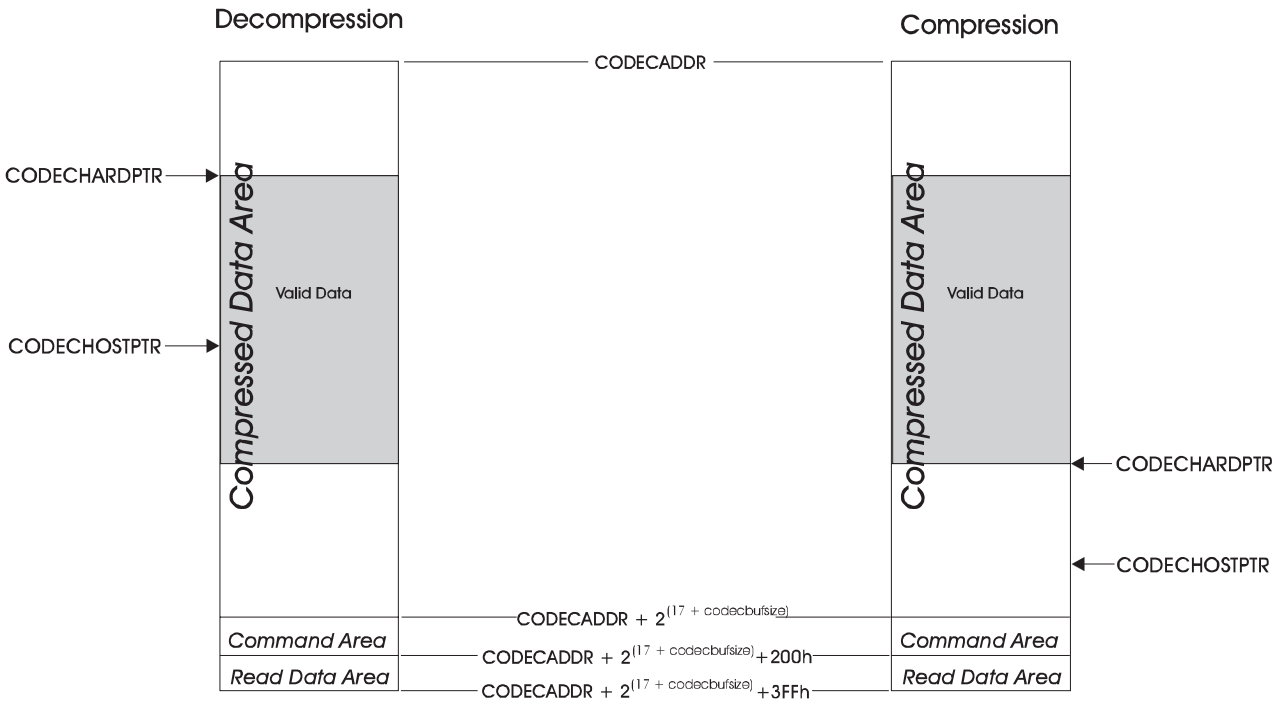
Compressed data area

The compressed data area is used both for compression and decompression operations. This buffer size is selectable between 128Kbytes or 256Kbytes. The functions of **CODECHARDPTR** differ for compression and decompression (refer to the register definitions in Chapter 3 for more details).

The compressed data area acts as a circular queue. Data from the beginning of one field is loaded into the dword which follows the last data from the previous field. It is the sole responsibility of the software to ensure that the compressed data area never overflows. The Codec Interface engine does not verify that there is valid compressed data in the buffer before reading, nor does it check to see that there is enough free space in the buffer before writing. The buffer level interrupt has been provided to help the software to ensure that overflows never occur.

.

Figure 4-11: CODEC Interface Organization



Command area

The *command area* is used to store the commands to be sent to the CODEC. The organization of these commands and their execution is explained in more detail in the 4.9.2 section. The command area is set to a fixed size of 512 bytes.

Read data area

The *read data area* is used to store the data which has been read from the CODEC. When more than one location is read with a single command, the data is packed to take full advantage of the 8-byte wide memory locations. However, if only a single CODEC location is read, the remaining 7 bytes of the qword will be unused.The read data is always stored beginning with the LSB. The read data area is set to a fixed size of 512 bytes.

4.9.2 Command Execution

Register read and write commands are stored in an off-screen command buffer. Each qword in the buffer may contain either a command or, in the case of a write command, write data. Each command, with its accompanying data, is stored one after the other in the command buffer.

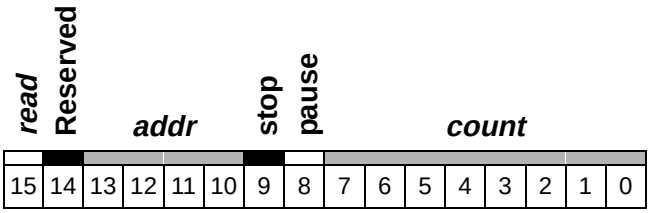
> ***Note:*** The first *qword* in the queue ***must always*** contain a command. The format of a command, with its accompanying data, is shown in Figure 4-12.

Figure 4-12: CODEC Command Format

63 ... 16	15 ... 0
	Command
write data 4	write data3	write data2	Write data1
:	:	:	:
:	:	:	:
write data n	write data n-1	write data n-2	write data n-3

The command word itself is composed of several control bits which affect command execution:

Command Word Definition:



count **<7:0>**

The ***count*** specifies the number of registers to be written to or read from the CODEC. When executing writes, the count will refer to how many words in Write Data qword(s) will be processed. When executing reads, the count will refer to how many times the address **addr** in the Command Word will be read.

pause **<8>**

The **pause** bit is used for *compression* only. It is active high. The command word in which it is contained is executed. Once this command is completed, register read/writes are suspended until the next end-of-field marker is detected in the compressed data stream.

Note: The pause bit does ***not*** reset the command buffer pointer.

stop **<9>**

The **stop** bit is used to halt *register accesses*. It is active high. The command word in which it is contained is executed. Once this is completed, register accesses are complete, and the **cmdcmplpen** field in the **VSTATUS** register is set. When software triggers command execution once again, the command buffer pointer is reset and execution begins from the first qword in the command buffer.

Here is an example of how the *pause* and *stop* bits are used in compression:

Pre-compression:
During the vertical blanking interval, software loads the off-screen command buffer with register transfer commands. The first set of instructions are used to setup the CODEC for the next field. The last instruction in the register setup sequence has its pause bit set high. The CODEC Interface Engine will execute all of the register read/write commands until it reaches the pause bit. At this point, the CODEC begins data compression transfers.

Post-compression:
In the command buffer, the pre-compression sequence should be followed by a set of post-compression instructions (to read field status information from CODEC or setup the next compression). When the end-of-field indication is detected in the compressed data stream, register read/write execution automatically resumes at the command

immediately following the last pre-compression command (the one with the pause bit set high). Commands will be executed until the next active pause bit or stop bit is encountered.

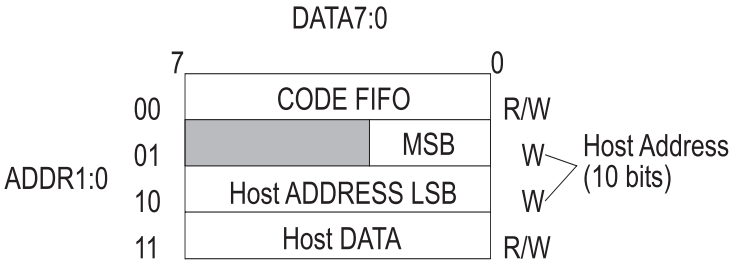
In the case where an active stop bit is encountered, register transfers are considered complete and software is interrupted.

addr <13:10>

Address to access for register *reads*

When executing read commands, **addr**<13:10> will indicate which address the Codec Interface will read from in the Codec address space. This address will be read as many times as was programmed in **count<7:0>**. When executing register writes, these 4 bits are unused. When transferring compressed data, the address asserted is programmed by software in the **CODECCTL** register. When in VMI mode, <13:10> are output. When in I33 mode, <11:10> are output.

Figure 4-13: Address Space of I33 CODEC in Code Slave Mode

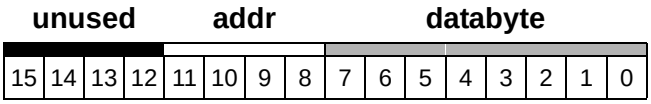


read <15>

This bit indicates the direction of transfer for reads or writes, with respect to the CODEC. Read and write commands may be interleaved in any manner.

- 0: write data to the CODEC registers
- 1: read data from the CODEC registers

Write data:



databyte <7:0>

Data byte to be written to the indirect register.

addr <11:8>

Indicates which segment of the CODEC address space the CODEC Interface will write too.

1. The Codec Interface engine begins executing commands when the **CODECCTL** register is written with an access that sets the **cmdexectrig** field (the command execution trigger field does not need to be cleared by software). When software triggers command execution, the Codec Interface engine resets its Command Area pointer and Read Data Area pointer.
   - The *first* pointer (command area pointer) selects the next qword to be read. This pointer is reset to point to the beginning of the command area when the **cmdexectrig** field is set.
   - The *second* pointer (read data area pointer) selects the next qword to be written in the read data area. This pointer is reset to the beginning of the read data area when the **cmdexectrig** field is set.
2. The Codec Interface engine then fetches the *first* command.
   - *If* the command is a *write*, the appropriate number of qwords are read from the command area and written to the CODEC. After one qword of data is read from the command area, the data is written to the CODEC one word at a time, starting with word 0, then word 1, and so on to word 3. If the write command completes before all 4 words are written, the remaining data is dropped and not used. On the next write command, data is again fetched and sent to the CODEC, starting with word 0.
   - *If* the command is a *read*, the appropriate number of bytes are read and stored in the read data area. The data is read from the CODEC one byte at a time, and is accumulated until a complete qword has been received. The first byte read is loaded into byte 0, the second into byte 1, and so on to byte 7 (the eighth byte). The resulting qword is then written to the next available location of the read data area. If the read command completes before all 8 bytes are filled, the data is written to the off-screen buffer as is (unfilled). On the next read command, data is again filled, starting with byte 0.
3. Upon completing the read or write command, the Codec Interface engine fetches the next command from the command buffer; the process repeats until a STOP command is executed.

Examples

Table 4-6 shows the contents of a command area, while Table 4-7 shows the contents of the read data area after the execution of all commands has taken place.

Table 4-6: Contents of the Command Area

Location	*Contents*	*Meaning*
+ 8000h	XXXXXXXXXXXX8401h	Read address “0001”, count=1
+ 8008h	XXXXXXXXXXXX0002h	Write 2 locations
+ 8010h	XXXXXXXX03FF02AAh	Write addresses and data
+ 8018h	XXXXXXXXXXXX0003h	Write 3 locations
+ 8020h	XXXX06BB057504EEh	Write addresses and data
+ 8028h	XXXXXXXXXXXX0104h	Write 4 locations, pause
+ 8030h	07DD06CC05BB04AAh	Write addresses and data
+ 8038h	XXXXXXXXXXXX9E04h	Read address “0111”, count=4, stop

Table 4-7: Contents of the Read Data Area

Location	*Contents*	*Meaning*
+ 8200h	XXXXXXXXXXXXXX00h	Read data from address “0001”
+ 8208h	XXXXXXXXDDDDDDDDh	Read data from address “0111”

The first command read 1 byte from address 01h. The data, 00h, was written to memory at address +8200h. The next command was a write of 2 locations. The data is fetched in memory and written as data AAh to address 02h, and data FFh to address 03h. The next command is a write to 3 locations. The data is fetched from memory and written as data EEh to address 04h, data 75h to address 05h, and data BBh to address 06h. The next command is a write to 4 locations with a pause. The data is fetched and written as data AAh to address 04h, data BBh to address 05h, data CCh to address 06h, and data DDh to address 07h.

Since the pause bit was asserted in this command, the Codec Interface engine will ***not*** fetch its next command until it receives the end-of-field indication from the CODEC. *At this point, compressed data transfers begin.* When the end-of-field is detected, the interface engine proceeds with the next command, which is a read from address 07h, count of 4, with stop. The data is read as DDh, DDh, DDh, DDh, and put into location +8208h. Since the stop bit was asserted in this command, the interface engine will not send any further commands to the CODEC and will assert the **cmdcmplpen** field of **VSTATUS**.

4.9.3 Output mode

The Codec Interface may operate in 3 possible output modes: VMI Mode A, VMI Mode B, and Zoran I33 compatible mode. The mode is programmed in the **CODECCTL** register. All examples set forth assume I33 mode. All programming procedures are identical in all modes (except for setting the proper mode in the **CODECCTL** register.

4.9.4 Codec Interface IDLE State

In order to have the Codec Interface enabled but in and IDLE state, the following fields need to be set as indicated *after* the Codec Interface is disabled:

field	*setting after Codec disable*
codecen	1
cmdexectrig	0
codectransen	0
all other fields	X

4.9.5 Recovery Width Programming

The strobe recovery pulse width in the Codec Interface engine is programmable in the **CODECCTL** register. The **codecrwidth** should be programmed before the Codec Interface is enabled (with **codecen**) to begin transfers to the CODEC. If the **codecrwidth** is reprogrammed during data transfers, data corruption may occur. The formula to compute the optimal recovery time, Trec (in *ns*), is:

$$Trec = N * Tgclkbuf$$

and Trec > Tcodecrec

Where:

Trec = minimum recovery time
Tgclkbuf = the period of gclkbuf
Tcodecrec = the CODEC's minimum required recovery width

Given:

gclkbuf (internal graphic clock) = 72 Mhz, thus Tgclkbuf = 13.9 *ns*
for I33, Tcodecrec = 55.5 *ns*

Then:

Trec = 55.6 *ns*, which is less that Tcodecrec = 55.5 *ns*

Thus, Trec should be greater than 55.5 *ns*, corresponding to 4 gclkbuf cycles.

Thus, codecrwidth should be programmed with "00" 4 gclkbuf cycles.

4.9.6 Miscellaneous Control Programming

The **miscctl** byte located in the **CODECCTL** register is used to program an 8 bit flip-flop on the graphics card. The values of the **miscctl** field are used to set various inputs to the CODEC and MPEG2 chips like SLEEP, START, etc. By writing to the **miscctl** field, software triggers a sequence to program the on-board flip-flop with the corresponding data.

Note: In order for this automated sequence to be executed, the Codec Interface engine must be enabled and in one of the following modes/states: IDLE, COMPRESSION, or DECOMPRESSION. For example, if the chip select for a CODEC is connected to bit 0 of the on-board flip-flop, and is active low, then software could enable the CODEC as follows:

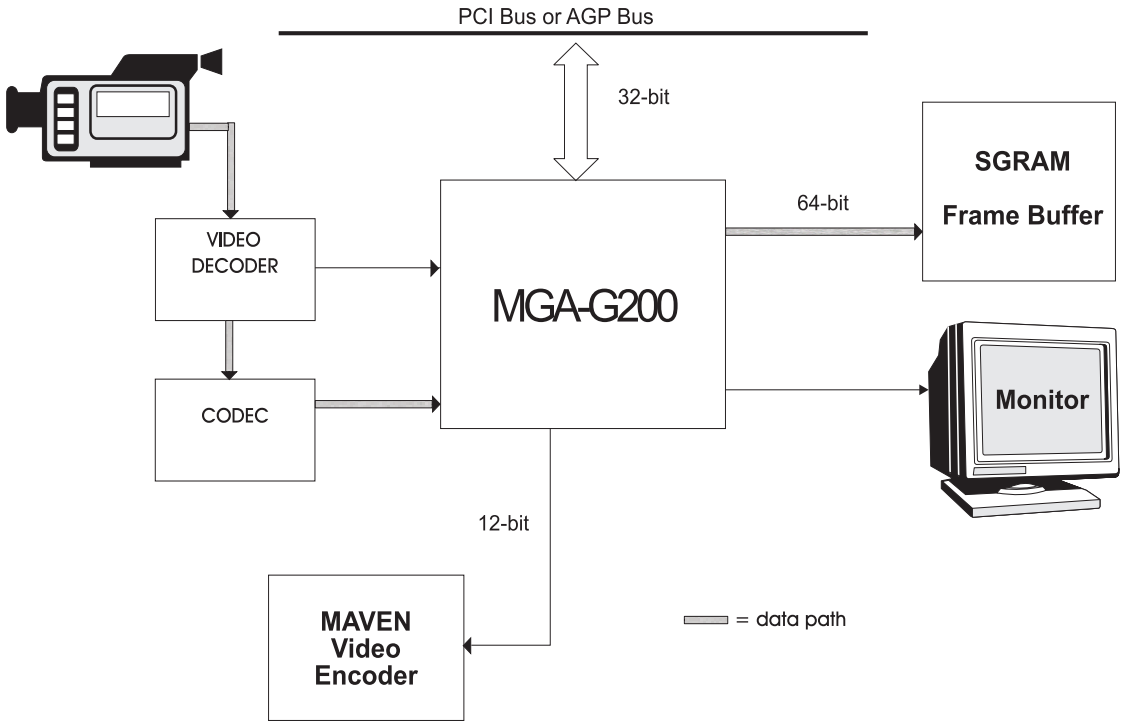
1. write “00000000”b to the lower byte of the **CODECCTL** register (to reset it, optional)
2. write “00000001”b to the lower byte of the **CODECCTL** register (to enable it)
3. write “11111110”b to the **miscctl** field of the **CODECCTL** register (to set the chip select to the CODEC).

If the Codec Interface is in the process of compression or decompression when the **miscctl** field is written to, then the Interface will wait until the current byte transfer is complete. At that point the on-board flip-flop will be programmed, and then data transfers will resume with the next byte. No data will be lost or corrupted during this process.

4.9.7 Compressing data

Data being compressed comes from the video decoder. The compressed video frame is returned to the frame buffer through the Codec Interface channel.

Figure 4-14: Compression of a Live Video Source



Two interrupts are used while the CODEC is compressing data. The *buffer level* interrupt is used to tell software the current fill state of the frame buffer. The *command execution completed* interrupt is used to request host services in order to adjust the compression factors.

The following must be performed to compress video:

Step 1. Program the video decoder according to its specification.

Step 2. Software must reset the Codec Interface engine by writing ‘0’ followed by a ‘1’ to **codecen** field bit <0>:

Register	*Function*	*Comment / Alternate Function*
CODECCTL	Reset the Codec Interface engine	Write 00h to the lower byte
CODECCTL	Reactivate the Codec Interface	Write 01h to the lower byte

Step 3. Software must then initialize the following registers in the Codec Interface engine:

Register	*Function*	*Comment / Alternate Function*
CODECADDR	Address of off-screen buffer	—

Step 4. The CODEC must be initialized with the mode and other parameters shown below. To do this, software must transfer CODEC register write commands into off-screen memory. Typically, the registers to be written are:

- **Mode Control**
- **FIFO Control**
- **HSTART**
- **HEND**
- **VSTART**
- **VEND**
- **Compression Ratio** (See ‘Command Word Definition:’ on page 4-85 to set the **stop** bit)

Note: The CODEC specification will have detailed information regarding which registers need to be programmed.

Step 5. Trigger the execution of commands to the CODEC:

Example:

- Codec mode = ‘1’ I33 mode
- Codecdatain = ‘1’ Compression (receiving data from the codec)

Register	*Function*	*Comment / Alternate Function*
CODECCTL	Enable the Codec Interface engine	00001111b

Step 6. At the completion of this command, the Codec Interface engine will set the **cmpcmplpen** field A of **VSTATUS**. The host will read the status in order to know when the command has been executed. The host must also clear the **cmdcmplpen** flag.

Register	*Function*	*Comment / Alternate Function*
VICLEAR	Clear interrupt	02h

Step 7. The host must then transfer the following commands to off-screen memory:

- program registers in CODEC (set the ***pause*** bit)
- read field information from CODEC
- read the FIFO status for the error conditions (if necessary, see CODEC specification) (set the ***stop*** bit)

Step 8. The host must then enable all interrupt bits.

Register	*Function*	*Comment / Alternate Function*
CODECHOSTPTR	Next level interrupt	value desired by the software

Step 9. Trigger the execution of commands to the CODEC and enable the transfer of compressed data:

Register	*Function*	*Comment / Alternate Function*
CODECCTL	Reset the Codec Interface engine	01001111b

Step 10. The Codec Interface engine will then begin to transfer data from the CODEC to the compressed data area.

Note: Since the first field will probably be corrupted, software should discard it.

Step 11. The Codec Interface engine will interrupt the host every time **CODECHARDPTR** is equal to **CODECHOSTPTR**. The host can detect this situation by reading the **blvlpen** field of the **VSTATUS** register. As part of the interrupt routine, the host must perform transfers from the compressed data area to the system memory or hard disk. The host must also clear the **blvlpen** flag and update its pointer.

Register	*Function*	*Comment / Alternate Function*
CODECHOSTPTR	Next level interrupt	—
VICLEAR	Clear Codec Status Register	04h

Step 12. When the CODEC asserts EOI (connected to misc[2] when codec engine is enabled), this informs the Codec Interface engine that its current read is the last byte of the field. The Codec Interface engine will write all remaining data in its memory interface buffer to the compressed data area and resume command execution.

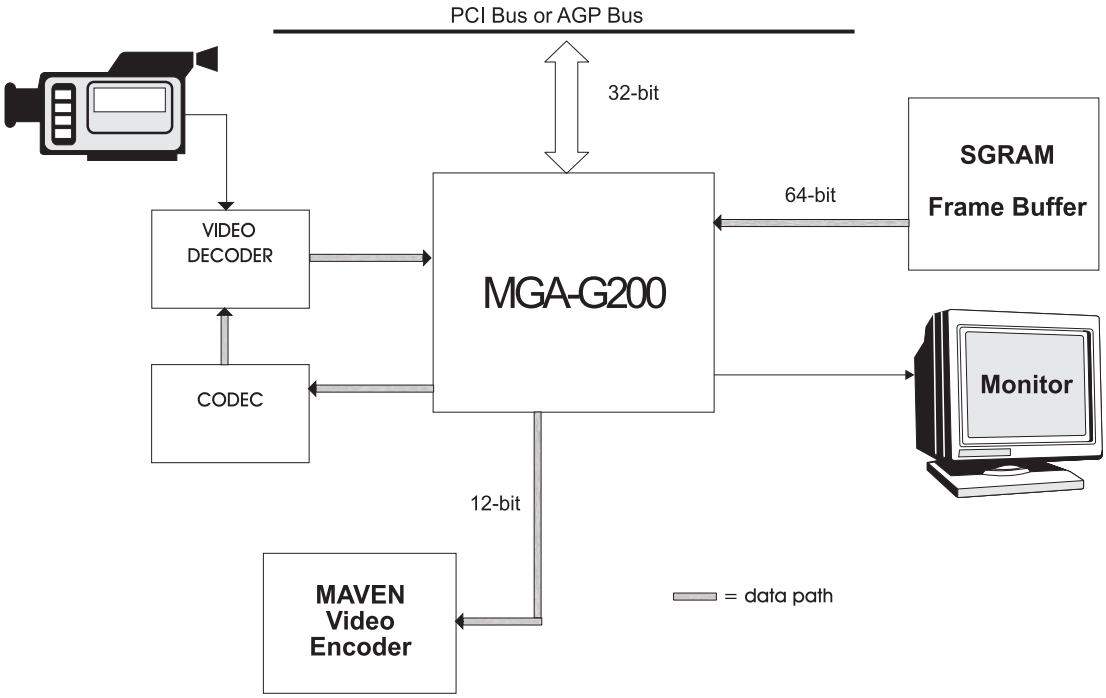
Step 13. The Codec Interface engine will then interrupt software when the command is completed. The host can detect this situation by reading the **cmdcmplpen** field of the **VSTATUS** register. Based on the statistics, software must calculate new specs for compression field, and update the write commands in the off-screen buffer (see Step 6). The host must also clear the **cmdcmplpen** flag and restart command execution.

Register	*Function*	*Comment / Alternate Function*
VSTATUS	Status of memory commands	00000010b
VICLEAR	Clear all CODEC interrupts	06h
CODECCTL	Reset the Codec Interface engine	00000000b

4.9.8 Decompressing data

When data is being decompressed, the compressed information is sent to the CODEC through the Codec Interface port. From there, the data is sent to the decoder and back to the MGA-G200 to be displayed on the monitor or TV (through MAVEN):

Figure 4-15: Decompressing Data from the Memory



Two interrupts are used when the Codec Interface is decompressing data. The *buffer leve*l interrupt is used to request more data from the host. The *decompression end of image* interrupt is used to notify software when the end of the field has been detected.

When **stopcodec** is set to a ‘1’ in the **CODECCTL** register, and decompression is enabled, the Codec Interface will look for the FFD9h end of image marker in the compressed data. When the end of image is detected, the Codec Interface will stall and post the **dcmpeoipen** interrupt (if enabled).

At this point, software has 2 options:

- reset the Codec DMA engine
- use **cmdexectrig** of the **CODECCTL** to trigger command execution. If this method is used, the software *must* set **codectransen** to a ‘0’, preventing decompression transfers from continuing after the commands have been completed.

If **stopcodec** is set to a ‘0’ for decompression transfers, then the Codec Interface will ***not*** detect the FFD9h end of image marker in the stream. It is up to software to stop the Codec Interface engine when it has determined that the desired field is complete (by polling the buffer level interrupt).

The following steps *must* be performed in order to decompress video:

Step 1. Program the video decoder according to its specification.

Step 2. Software *must* reset the Codec Interface engine by writing '0' followed by a '1' to **codecen** field bit <0>:

Register	*Function*	*Comment / Alternate Function*
CODECCTL	Reset the Codec Interface engine	Write 00h to the lower byte
CODECCTL	Reactivate the Codec Interface engine	Write 00h to the lower byte

Step 3. Software must then initialize the following registers in the Codec Interface engine:

Register	*Function*	*Comment / Alternate Function*
CODECADDR	Start address of buffer in off-screen memory	—

Step 4. The CODEC must be initialized with the mode and other parameters shown below. To do this, software must transfer CODEC register write commands into off-screen memory. Typically, the registers to be written are:

- **Mode Control**
- **FIFO Control**
- **HSTART**
- **HEND**
- **VSTART**
- **VEND** (See 'Command Word Definition:' on page 4-85 to set the **stop** bit)

Note: The CODEC specification will have detailed information regarding which registers need to be programmed.

Step 5. Trigger the execution of commands to the CODEC:

Register	*Function*	*Comment / Alternate Function*
CODECCTL	Enable the Codec Interface engine	00000111b

Step 6. The host must then fill at least half of the compressed data area and write its pointer

Example:

- Codec mode = '1' I33 mode

Register	*Function*	*Comment / Alternate Function*
CODECHOSTPTR	Next level interrupt	—

Step 7. At the completion of the command, the Codec Interface engine will set the **cmpcmplpen** field. The host will read the status in order to know when the command has been executed. The host must also clear the **cmdcmplpen** flag.

Register	*Function*	*Comment / Alternate Function*
VICLEAR	Clear interrupt	02h

Step 8. The host must then enable the buffer level interrupt and enable the transfer of compressed data.

Register	*Function*	*Comment / Alternate Function*
CODECCTL	Enable the Codec Interface engine	01000011b

Step 9. The Codec Interface engine will then begin to transfer data from the compressed data area to the CODEC.

Step 10. The Codec Interface engine will interrupt the host every time **CODECHARDPTR** is equal to **CODECHOSTPTR**. At that time, the host should add more data to the compressed data area. The host must also clear the **blvlpen** flag and update its pointer.

Register	*Function*	*Comment / Alternate Function*
CODECHOSTPTR	Next level interrupt	—
VICLEAR	Status of memory commands	04h

4.9.9 Error Recovery

The Codec Interface gives *highest priority* to command execution. When transferring data, if software determines there is an error condition with the CODEC, software can trigger command execution and preempt data transfers. If the Codec Interface is triggered to execute commands while it is the process of transferring data, the Codec Interface engine will disregard any data presently in its 4 qword fifo and load and execute the first command in the command data buffer.

When the interface engine has completed all the commands (signalled by a STOP in the last command), it will resume data transfers wherever it left off (the **CODECHARDPTR** does not reset under these conditions).

➽ ***Note: Any*** data already loaded in the 4 qword fifo (either from the CODEC or from the frame buffer) when command execution is triggered will be *lost*.

4.10 Backend Scaler

4.10.1 Introduction

4.10.1.1 Overview

The Backend Scaler supports one window. The supported formats of source data taken from the frame buffer are: YCbCr in 4:2:2;and 4:2:0 (2 planes).

The window performs independent horizontal and vertical scaling. Bilinear filtering is ***only*** available on Y component. ***However***, horizontally interpolated upsampling of the chroma component is available. For magnification: replicate or bilinear filters are possible. For *minification*: drop, fixed 0.25 coefficient or normal bilinear filters are possible. The normal bilinear filter uses the next adjacent pixel or line to perform interpolation.

The Backend Scaler also supports: complete source cropping; video de-interlace conversion with subpixel compensation; and horizontal mirroring.

The Backend Scaler registers are double-buffered: they are internally updated once per frame when the **besvcnt** field compares with the CRTC vertical counter. If more than one register of a window must be modified, ***ensure*** that they are all reprogrammed during the same frame: the vertical counter should ***not*** reach the programmed **besvcnt** in the middle of reprogramming (this can result in unexpected intermediate video images or unwanted artifacts appearing on the screen).

Four offscreen buffers are available to the window. In *software manual mode*, the field selection is under the control of software. In *hardware automatic field select*, the video input port toggles the circular select after each field received so that it is not necessary for the software to be interrupted at each field.

The Backend Scaler takes data from the selected buffer, performs scaling, then sends it to the ramdac. The keying circuitry then decides to display it to the screen. ***No*** memory writes are made. Two modes of keying are available: color keying and (inside the Backend Scaler window) coordinates keying. Color keying is done on the graphic color.

The scaling factors are limited to 1/32 in downscaling and 16384/(source width) in upscaling.

The Backend Scaler includes a YCbCr-to-RGB converter which allows to send full 24 bit RGB colors to the screen.

The Backend Scaler is ***not*** supported when the CRTC is programmed in interlace mode. Virtual desktop and hardware zoom are supported ***only*** in software (which means that software must reprogram the Backend Scaler accordingly with the changing desktop situation).

4.10.1.2 Notation

[real number]	:	A fixed-point representation; it is important to apply this specification immediately where specified. Do ***not*** use more precision than specified. Do ***not*** round-off ***any*** value.
>>	:	A logical shift to the right.

4.10.1.3 Backend Scaler Control

Backend Scaler control results from the following registers:

Register	*Function*
BESGLOBCTL	This register sets the following ***global*** controls: **beshzoom** accelerated 2X horizontal zoom **beshzoomf** accelerated 2X horizontal zoom filtering **bescorder** chroma sample order **besreghup** update on horizontal sync. for test **besvcnt** vertical counter register update
BESCTL	This register sets the following ***window*** controls: **besen** Backend Scaler enable **beshfen** horizontal filtering enable **besvfen** vertical filtering enable **beshfixc** horizontal fixed coefficient enable **bescups** chroma upsampling enable **bes420pl** 4:2:0 planar data format **besdith** dither enable **beshmir** horizontal mirror enable **besbwen** black and white enable **besblank** blank enable **besfselm** field select mode **besfsel** field select

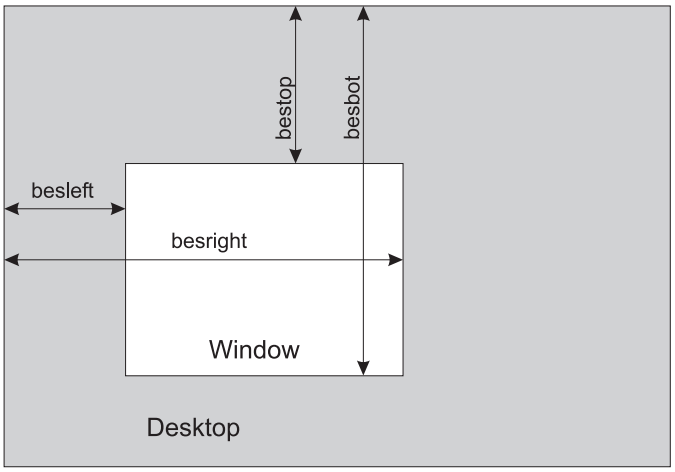
4.10.1.4 Backend Scaler Status

The status indicates the buffer currently being displayed (A1, A2, B1, B2)

Register	*Fields*	*Comment / Alternate Function*
BESSTATUS	**besstat**	status of window

4.10.1.5 Window Coordinates

The horizontal and vertical coordinates of the window in the desktop are defined as follows:



The fields are combined in registers:

Register	*Fields*	*Comment / Alternate Function*
BESHCOORD	**besleft** **besright**	**besright** *must* be greater than **besleft** 0 to max. desktop
BESVCOORD	**bestop** **besbot**	**besbot** *must* greater than **bestop** 0 to max. desktop

4.10.1.6 Bases Address and Origin Address

The *base address* is the first byte of the source image in the frame buffer without source cropping or desktop offset. The *origin address* is the first byte of the first line (source image) used to produce the destination image (with source cropping and desktop offset).

The source image (below) is displayed in xy, this occurs even if stored linearly in the frame buffer.

	…	A00AFD	A00AFE	A00AFF
base_address	A00B00	A00B01	A00B02	A00B03
	A00B04	A00B05	A00B06	A00B07
	A00B08	A00B09	A00B0A	A00B0B
origin_address	A00B0C	A00B0D	A00B0E	A00B0F
	A00B10	A00B12	A00B13	…

Legend:

Data dropped by source cropping and /or desktop offset.

Data used to produce destination image.

Note: The base and origin addresses are useful for vertical source positioning.

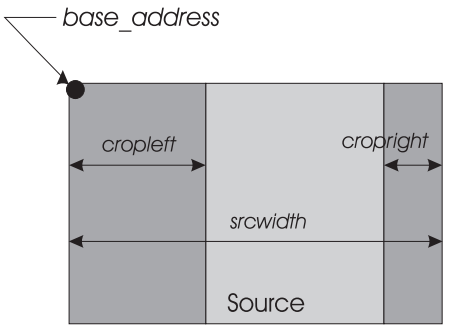
4.10.1.7 Pitch

The *pitch* is the offset (in numbers of pixels) from the beginning of one line to the next in the field currently read from the source data.

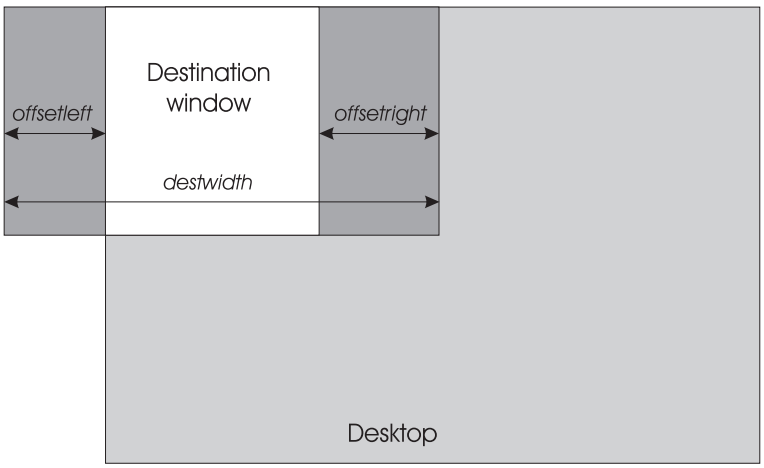
The pitch must be programmed in the **BESPITCH** register, must be a multiple of 4 in 4:2:2 format and a multiple of 8 in 4:2:0 planar format. The *maximum* value for pitch is 4092 pixels in 4:2:2 format and 4088 pixels in 4:2:0 format.

4.10.2 Horizontal Scaling

The following illustrations demonstrate the source and desktop considerations for horizontal scaling:



Source with left and right cropping



Destination window with left and right offset

The last horizontal coordinate of the source must be programmed in the **BESHSRCLST** register:

Register	*Field*	*Function*
BESHSRCLST	**beshsrclst**	*srcwidth* - 1

4.10.2.1 Horizontal Inverse Scaling Factor

The *Horizontal Inverse Scaling Factor* is the ratio of the source width to the destination width. To calculate this inverse factor (taking into account the above parameters), do the following:

Step 1. Set the interval representation of source and destination variable.

	Filtering mode ON **beshfen** = 1	Filtering mode OFF **beshfen** = 0	
		Downscaling(1)	Upscaling(1)
Interval representation value (*intrep*)	1(2)(3)	1(3)	0

(1) Upscaling is used when the destination width is greater than or equal to the source width (with source width reduced by cropping values)

(2) If the destination width is equal to the source width, *intrep* = 0 (with source width reduced by cropping values).

(3) If the destination width is less than 2, *intrep* = 0.

Step 2. Inverse Scaling Factor

Inverse Scaling Factor:

$$ifactor = \left[\frac{srcwidth - cropleft - cropright - intrep}{destwidth - intrep} \right]_{5.14}$$

Inverse Scaling Factor with better precision:

$$ifactorbetter = \left[\frac{srcwidth - cropleft - cropright - intrep}{destwidth - intrep} \right]_{5.20} + (intrep >> 20)$$

Step 3. Set the Round-off Variable

Condition: $(ifactorbetter * (destwidth - 1))_{FLOOR} > (ifactor * (destwidth - 1))_{FLOOR}$:

	Filtering mode ON **beshfen** = 1	Filtering mode OFF **beshfen** = 0	
		Condition:	
		True	False
Round-off value (*roundoff*)	0	1	0

Step 4. Set the Accelerated 2X Zoom Variable

	Accelerated 2x zoom ON **beshzoom** = 1	Accelerated 2x zoom OFF **beshzoom** = 0
Accelerated 2x zoom (*acczoom*)	2	1

Step 5. Program the Inverse Scaling Factor

The Inverse Scaling Factor ***must*** be programmed with the following formula:

$$\textbf{beshiscal} = (acczoom * ifactor) + (roundoff >> 14)$$

The **beshiscal** value ***must*** be in the following interval:

$$\frac{srcwidth}{16384} <= \textbf{beshiscal} < 32$$

Register	*Field*
BESHISCAL	beshiscal

4.10.2.2 Horizontal Source Positioning

The horizontal starting source position (**BESHSRCST**) is the first source pixel that will contribute to the left first destination pixel.

Without horizontal mirroring **(beshmir** = 0**)**:

beshsrcst = *cropleft* + *offsetleft* * (*ifactor* + (*roundoff* >>14))

With horizontal mirroring **(beshmir** = 1):

beshsrcst = *cropright* + *offsetleft* * (*ifactor* + (*roundoff* >> 14))

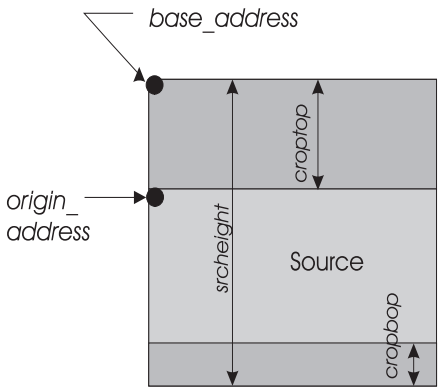
The horizontal ending source position (**BESHSRCEND**) is the last source pixel that will contribute to the last right destination pixel.

$$\textbf{beshsrcend} = \textbf{beshsrcst} + ((destwidth - offsetleft - offsetright - 1) / acczoom)_{FLOOR} * (acczoom * ifactor + (roundoff >>14))$$

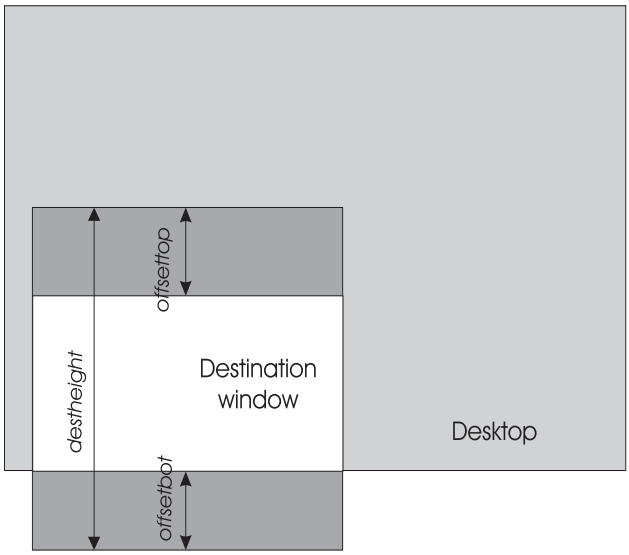
Register	*Field*
BESHSRCST	beshsrcst
BESHSRCEND	beshsrcend

4.10.3 Vertical Scaling

The following illustration demonstrates the source and desktop considerations for vertical scaling:



Source with top and bottom cropping



Destination window with top and bottom offset

4.10.3.1 Vertical Inverse Scaling Factor

The *Vertical Inverse Scaling Factor* is the ratio of the source height to the destination height. To calculate this inverse factor, (taking into account the above parameters) do the following:

➧ ***Note:*** For ***de-interlace*** conversion, the source is a field

Step 1. Set the interval representation of source and destination.

	Filtering mode ON **besvfen** = 1	Filtering mode OFF **besvfen** = 0	
		Downscaling(1)	Upscaling(1)
Interval representation value (*intrep*)	1(2)(3)	1(3)	0

(1) Upscaling is used when the destination height is greater than or equal to the source height (with source height reduced by cropping values)

(2) If the destination height is equal to the source height, *intrep* = 0 (with source height reduced by cropping values).

(3) If the destination height is less than 2, *intrep* = 0.

Step 2. Inverse Scaling Factor

Inverse Scaling Factor:

$$ifactor = \left[\frac{srcheight - croptop - cropbot - intrep}{destheight - intrep} \right]_{5.14}$$

Inverse Scaling Factor with better precision:

$$ifactorbetter = \left[\frac{srcheight - croptop - cropbot - intrep}{destheight - intrep} \right]_{5.20} + (intrep >> 20)$$

Step 3. Set the Round-off Variable

Condition: $(ifactorbetter * (destheight - 1))_{FLOOR} > (ifactor * (destheight - 1))_{FLOOR}$:

	Filtering mode ON **beshfen** = 1	Filtering mode OFF **beshfen** = 0	
		Condition:	
		True	False
Round-off value (*roundoff*)	0	1	0

Step 4. Program the Inverse Scaling Factor

The Inverse Scaling Factor must be programmed with the following formula:

$$\textbf{besviscal} = ifactor + (roundoff >> 14)$$

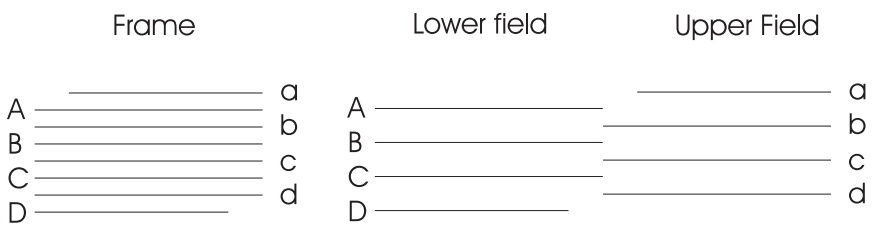
The **besviscal** value must be in the following interval:

$$\frac{srcheight}{16384} <= \textbf{besviscal} < 32$$

Register	*Field*
BESVISCAL	**besviscal**

4.10.3.2 Vertical Subpixel Compensation

For de-interlaced conversion, subpixel compensation is applied on the lower field.



Subpixel Compensation Value (S_C):

Downscaling:

Upper Field:

$$S_C = 0$$

Lower Field:

$$S_C = \left[\frac{2 * besviscal - 1}{2} \right]_{5.14}$$

Upscaling:

Upper Field:

$$S_C = 0$$

Lower Field:

$$S_C = \left[\frac{destheight - intrep}{2 * (srcheight - croptop - cropbot - intrep)} \right]_{5.14}$$

4.10.3.3 Vertical Source Positioning

Vertical Source Positioning is created in relation to subpixel compensation, source cropping, desktop offset and data format. Upscaling and Downscaling are presented separately.

The following table shows how to set the round-off variable affecting the inverse scaling factor calculation:

	Data format 4:2:0 **bes420pl** = 1	Data format 4:2:2 **bes420pl** = 0
Data Format (*dataformat*)	1	2

4.10.3.3.1 Downscaling

Origin Address

The Origin Address of buffer A and B for fields 1 and 2 is achieved by:

$$origin\_address = [croptop + offsettop * \mathbf{besviscal} + S_C]_{24.0} * \mathbf{BESPITCH} * dataformat + base\_address + \mathbf{beshmir} * (srcwidth * dataformat - 1)$$

Register	*Function*	*Comment*
BESA1ORG **BESA2ORG** **BESB1ORG** **BESB2ORG**	$[\ origin\_address\ ]_{24.0}$	When *de-interlace conversion* is desired, the subpixel compensation value (S_C) is applied to the registers that contain the lower field.

Chroma Plane Origin Address

The Chroma Plane Origin address of buffer A and B for fields 1 and 2 is achieved by

$$chroma\_plane\_origin\_address = [(croptop + offsettop * \mathbf{besviscal} + S_C) / 2]_{24.0} * \mathbf{BESPITCH} + chroma\_plane\_base\_address + \mathbf{beshmir} * (srcwidth * dataformat - 1)$$

Register	*Function*	*Comment*
BESA1CORG **BESA2CORG** **BESB1CORG** **BESB2CORG**	$[\ chroma\_plane\_origin\_address\ ]_{24.0}$	When *de-interlace conversion* is desired, the subpixel compensation value (S_C) is applied to the registers that contain the lower field.

Vertical Weight Starting Value

The Vertical Weight Starting Value registers are programmed as follows (the weight is the same for buffer A and B):

If de-interlaced conversion is desired and is the lower field then:

If *offsettop* = 0

Then *weight* = -(0.5 + [**besviscal**]$_{0.14}$)

Else *weight* = -0.5 + *offsettop* * **besviscal**

Else *weight* = *offsettop* * **besviscal**

Register	*Fields*	*Function*	*Comment*
BESV1WGHT	**bes1wght**	[*weight*] $_{0.14}$	When *de-interlace conversion* is desired, the subpixel compensation is applied to the registers that contain the lower field.
	bes1wghts	sign of *weight*	
BESV2WGHT	**bes2wght**	[*weight*] $_{0.14}$	
	bes2wghts	sign of *weight*	

Vertical Source Last Position

The vertical source last position for field 1 or 2 is achieved by:

$$vsrclst = srcheight - 1 - [croptop + offsettop * \mathbf{besviscal} + S_C]_{10.0}$$

Register	*Fields*	*Function*	*Comment*
BESV1SRCLST	**besv1srclst**	[*vsrclst*]$_{10.0}$	When *de-interlace conversion* is desired, the subpixel compensation value (S_C) is applied to the registers that contain the lower field.
BESV2SRCLST	**besv2srclst**		

Vertical Source Start Polarity

The vertical source start polarity for field 1 ***or*** 2 is achieved by:

Register	*Fields*	*Function*
BESCTL	**besv1srcstp**	Set if [*croptop* + *offsettop* * **besviscal** + S_C]$_{FLOOR}$ is odd.
	besv2srcstp	

4.10.3.3.2 Upscaling

Origin Address

The Origin Address of buffer A and B for fields 1 and 2 is achieved by:

$$origin\_address = [croptop + (offsettop - S_C)_{abs} * \mathbf{besviscal}]_{24.0} * \mathbf{BESPITCH} * dataformat + base\_address + \mathbf{beshmir} * (srcwidth * dataformat - 1)$$

Register	Function	*Comment*
BESA1ORG **BESA2ORG** **BESB1ORG** **BESB2ORG**	$[\ origin\_address\ ]_{24.0}$	When *de-interlace conversion* is desired, the subpixel compensation value (S_C) is applied to the registers that contain the lower field.

Chroma Plane Origin Address

The Chroma Plane Origin address of buffer A and B for fields 1 and 2 is achieved by:

$$chroma\_plane\_origin\_address = [(croptop + (offsettop - S_C)_{abs} * \mathbf{besviscal}) / 2]_{24.0} * \mathbf{BESPITCH} + chroma\_plane\_base\_address + \mathbf{beshmir} * (srcwidth * dataformat - 1)$$

Register	*Function*	*Comment*
BESA1CORG **BESA2CORG** **BESB1CORG** **BESB2CORG**	$[\ chroma\_plane\_origin\_address\ ]_{24.0}$	When *de-interlace conversion* is desired, the subpixel compensation value (S_C) is applied to the registers that contain the lower field.

Vertical Weight Starting Value

The Vertical Weight Starting Value registers are programmed as follows (the weight is the same for buffer A and B):

If de-interlaced conversion is desired and is the lower field, then

If *offsettop* * **besviscal** ≥ 0.5

Then *weight* = -0.5 + *offsettop*

Else *weight* = -(0.5 + *offsettop* * **besviscal**)

Else *weight* **=** *offsettop* *** besviscal**

Register	*Fields*	*Function*	*Comment*
BESV1WGHT	**bes1wght**	$[weight]_{0.14}$	When *de-interlace conversion* is desired, the subpixel compensation value (S_C) is applied to the registers that contain the lower field.
	bes1wghts	sign of *weight*	
BESV2WGHT	**bes2wght**	$[weight]_{0.14}$	
	bes2wghts	sign of *weight*	

Vertical Source Last Position

The vertical source last position for field 1 or 2 is achieved by:

$$vsrclst = srcheight \text{ -1- } [croptop + (offsettop - S_C)_{abs} * \textbf{besviscal}]_{10.0}$$

Register	*Fields*	*Function*	*Comment*
BESV1SRCLST	**besv1srclst**	$[vsrclst]_{10.0}$	When *de-interlace conversion* is desired, the subpixel compensation value (S_C) is applied to the registers that contain the lower field.
BESV2SRCLST	**besv2srclst**		

Vertical Source Start Polarity

The vertical source start polarity for field 1 or 2 is achieved by:

Register	*Fields*	*Function*
BESCTL	**besv1srcstp**	Set if $[croptop + (offsettop - S_C)_{abs} * \textbf{besviscal}]_{FLOOR}$ is odd.
	besv2srcstp	

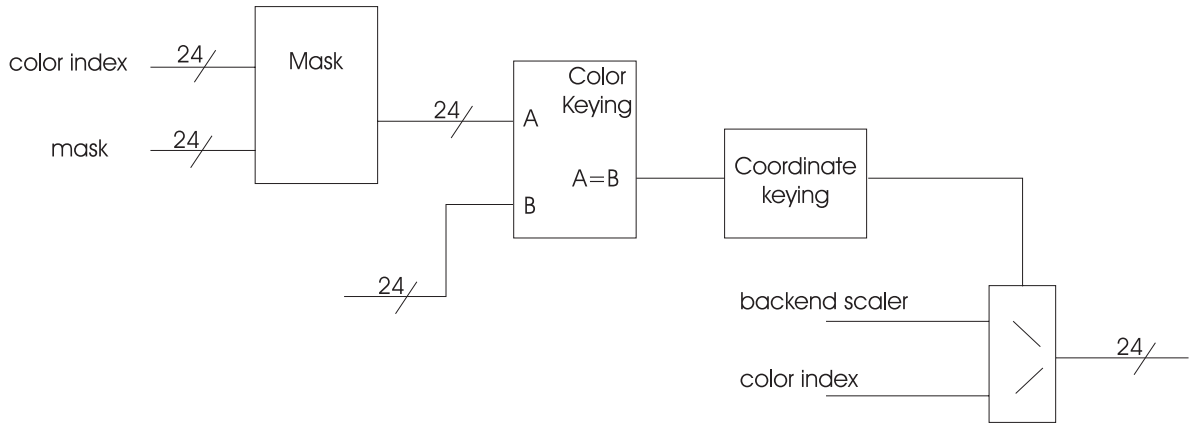
4.10.4 Keying

Note: Refer to the CRTC section for bandwidth usage when enabling the Backend Scaler.

4.10.4.1 Color Keying

The field **colkeyen** of **XKEYOPMODE** register selects the type of display for the video window. In color keying mode **colkeyen** = 1.

Color keying works by first comparing to see if the pixel color should be used or masked. If the pixel color matches the key color; a single bit is generated signifying a match or no match. Coordinate keying is used at the end to verify that the current coordinate is inside the Backend Scaler window.



The color key mask registers (**XCOLMSKRED**, **XCOLMSKGREEN**, **XCOLMSKBLUE**) mask bit-to-bit with the color index of the palette. Some LSB of the mask value can be set to ‘0’ when range keying is required.

The color key registers (**XCOLKEYRED**, **XCOLKEYGREEN**, **XCOLKEYBLUE**) must be programmed with the appropriate color.

4.10.4.2 Overlay Keying

The field **colkeyen** of **XKEYOPMODE** register selects the type of display for the video window. In overlay keying mode **colkeyen** = 0.

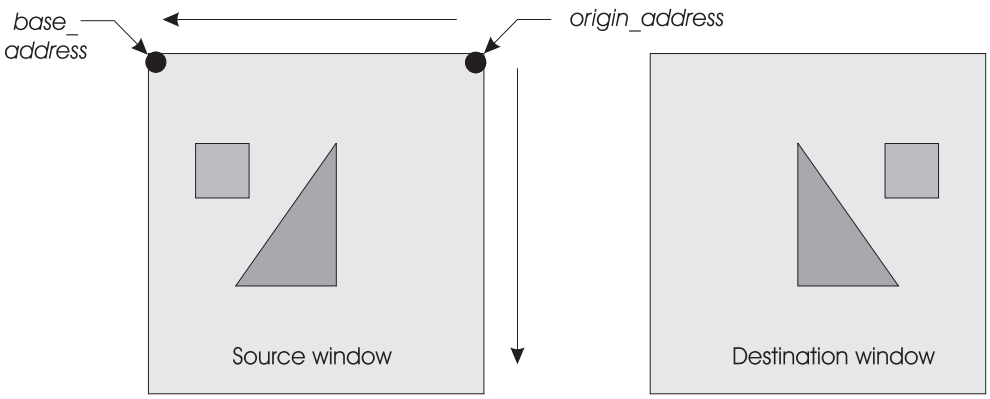
The overlay keying uses only the coordinates of the Backend Scaler window to perform keying between graphic and video data, regardless of keying color.

4.10.5 Miscellaneous Functions

4.10.5.1 Horizontal Mirroring

Horizontal mirroring is set by **beshmir** control bit and the origin address must be pointing to the top right corner of the source.

Horizontal Mirroring:



Horizontal mirroring affects the horizontal source positioning registers **BESHSRCST** and **BESHSRCEND,** and horizontal inverse scaling factor register **BESHISCAL.**

4.10.5.2 Automatic Field Select

The Automatic Field Select must be initialized using the following procedure:

Step 1. Set the starting field for the window:

Field select mode in software mode:

besfselm = 0

Starting field in field select register:

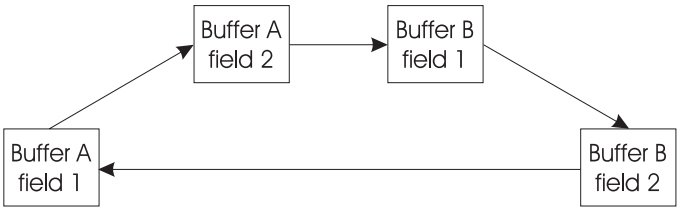
besfsel = 00: buffer A field 1 is the starting field
01: buffer A field 2 is the starting field
10: buffer B field 1 is the starting field
11: buffer B field 2 is the starting field

Step 2. Start the automatic field selection

Field select mode in hardware mode:

besfselm = 1

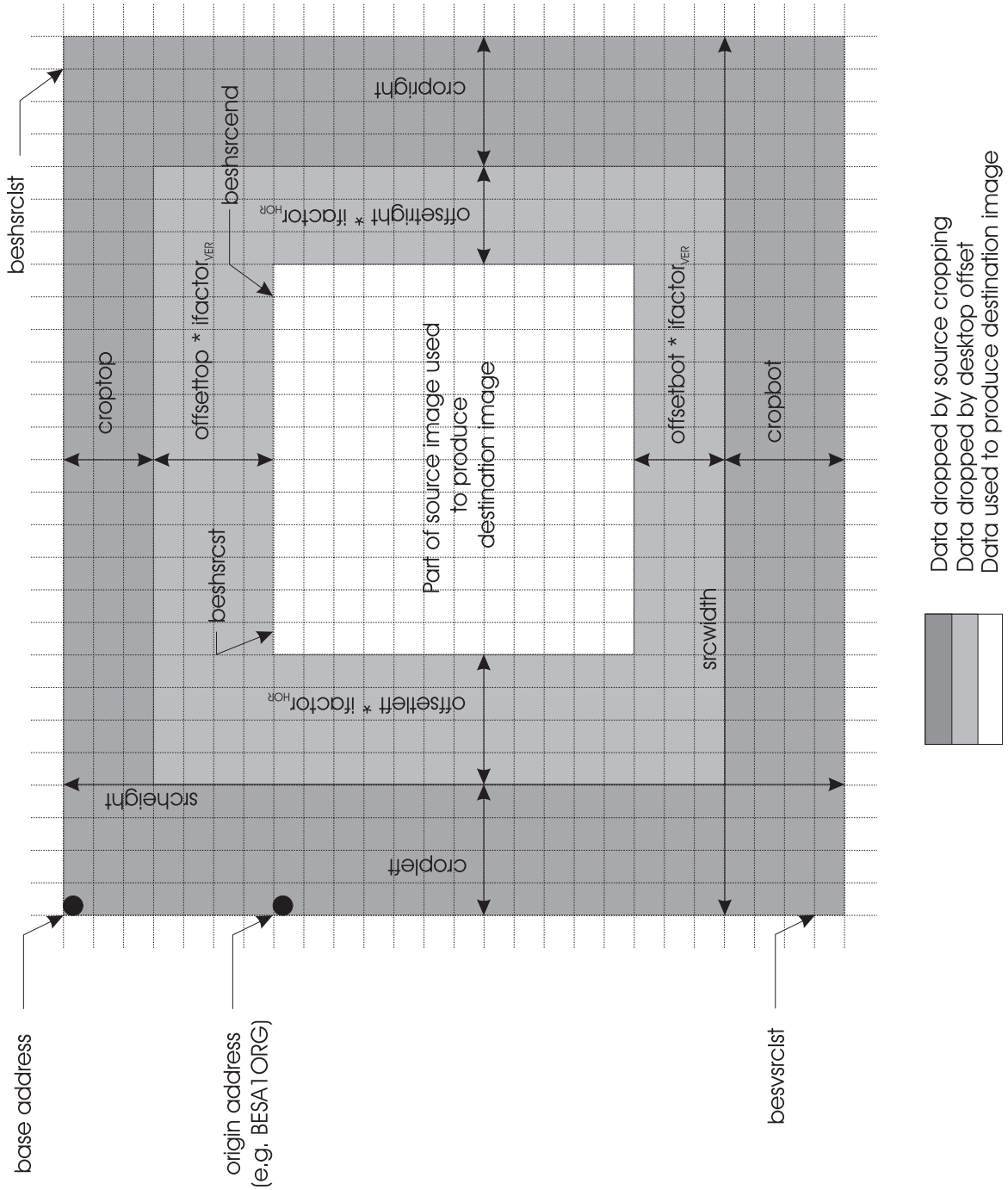
Buffer cycle used in hardware mode:



4.10.6 BES Parameter Example:

The following figure is an example of a source window with full source cropping and full destination offset:

Source Image:



4.11 Interrupt Programming

The MGA-G200 has 11 interrupt sources: 7 Graphics Engine Interrupts and 4 Video Interrupts.

Graphics Engine Interrupts:

1. Soft Trap interrupt

   This interrupt is generated when a write to the **SOFTRAP** register is executed (refer to 'Programming Bus Mastering for DMA Transfers' on page 4-11 and to the **SOFTRAP** register description).

2. Pick interrupt

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

3. Vertical Sync interrupt

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

4. Vertical Line interrupt

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

5. External interrupt

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

6. WARP interrupt

   This interrupt is generated when the WARP executes an IRQ instruction.

7. WARP cache interrupt

   This interrupt is generated when there is a WARP cache miss.

Video Interrupts:

1. Video In Vertical Sync interrupt

   This interrupt is generated when a video input vsync is detected. This interrupt is located in the **VSTATUS** register.

2. Codec Command Complete interrupt

   This interrupt is generated when the codec interface has completed executing the commands in the command buffer. The interrupt is located in the **VSTATUS** register

3. Codec Buffer Level interrupt

   This interrupt is generated when the Codec interface's hardware pointer (**CODECHARDPTR**) is equal to the value set in the **CODECHOSTPTR**. This interrupt is located in the **VSTATUS** register.

4. Decompression End of Image interrupt

   This interrupt is posted when **stopcodec** is a '1'. The Codec Interface is decompressing data, and the FFD9h end of image marker was detected in the data stream. This interrupt is located in the **VSTATUS** register.

Table 4-8: Supported Functionality for each Interrupt Source

Type	*Interrupt*	*STATUS*	*EVENT*	*ENABLE*	*CLEAR*
Graphic Engine	Soft trap	— —	**softrapen** **STATUS**<0>	**softrapien** **IEN**<0>	**softrapiclr** **ICLEAR**<0>
	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>	— —
	WARP	— —	**wpen** **STATUS**<7>	**wien** **IEN**<7>	**wiclr** **ICLEAR**<7>
	WARP cache	— —	**wcpen** **STATUS**<8>	**wcien** **IEN**<8>	**wciclr** **ICLEAR**<8>
Video	Video In vsync	— —	**vinvsyncpen** **VSTATUS**<0>	**vinvsyncien** **VIEN**<0>	**vinvsynciclr** **VICLEAR**<0>
	Codec command done	— —	**cmdcmplpen** **VSTATUS**<1>	**cmdcmplien** **VIEN**<1>	**cmdcmpliclr** **VICLEAR**<1>
	Codec buffer level	— —	**blvlpen** **VSTATUS**<2>	**blvlien** **VIEN**<2>	**blvliclr** **VICLEAR**<2>
	Codec decompression end of image marker	— —	**dcmpeoipen** **VSTATUS**<3>	**dcmpeoiien** **VIEN**<3>	**dcmpeoiiclr** **VICLEAR**<3>

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

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

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

IEN Each interrupt source may or may not take part in activating the PINTA/ hardware interrupt line. The EVENT and STATUS flags are not affected by interrupt enabling or disabling.

VSTATUS Indicates which bit reports the current state of the video interrupt source.

VICLEAR A pending bit remains set until it is cleared by the associated clear bit.

VIEN Each interrupt source may or may not take part in activating the PINTA/hardware interrupt line. The **VSTATUS** flags are only set when the enable bit in **VIEN** for each respective source is on.

Note:

- Clear interrupts before enabling them
- **vsyncpen** is set on the rising edge of vsync
- **pickpen** is set on the first pixel within the clipping box
- **vlinepen** is set at the beginning of the line
- **vinvsyncpen** is set after a vsync is detected and the video in unit has completed writing the data to memory.

4.12 Power Saving Features

4.12.1 Entering Power Saving Mode

The MGA-G200 supports three power conservation features:

- DPMS is supported directly, through the following control bits:
  - Video can be disabled using **scroff** blanking bit (**SEQ1**<5>)
  - Vertical sync can be forced inactive using **vsyncoff** (**CRTCEXT1**)
  - Horizontal sync can be forced inactive using **hsyncoff** (**CRTCEXT1**)
- The video section can be powered down using the following steps:
  1. Set bits **scroff**, **hsyncoff** and **vsyncoff** to '1'.
  2. Disable the cursor (set the **curmode** field to '00').
  3. Set the **pixclkdis** field of **XPIXCLKCTRL** to '1'.
  4. Power down the DAC.
  5. Power down the LUT.
  6. Power down the Pixel PLL.
- Chip power consumption can be further reduced by shutting-down the drawing engine and slowing-down the system clocks.The procedure below ***must*** be followed:
  1. Power down the video section following the procedure above.
  2. Wait for **dwgengsts** to become '0'.
  3. If the contents of the frame buffer must be preserved, MCLK must be running and the **rfhcnt** field of the **OPTION** register must be re-programmed according to the new MCLK frequency (normally, set rfhcnt to '0001').
  4. Program the memory clock to the desired value following the procedure in section 4.7.8.3 (see (B) Changing the System PLL Frequency on page 4-79). The recommended PLL oscillation frequency is 6.66MHz (N=107, M=28, P=7, S=0).
  5. Set mclkdiv to '1' (gclkdiv should already be '0' and must be set to '0' if that is not already the case) following the procedure in section 4.7.8.3 (see (C) Changing the System Clock Source, MCLK, or GCLK Division Factor on page 4-79).

Note: In Power Saving mode, ***do not*** use, or initialize, the drawing engine.

Note: MGA-G200 supports PCI Bus Power Management Interface specification 1.0.(See the **PM_CSR** register on page 4-23.)

4.12.2 Coming Out of Power Saving Mode

When coming out of Power Saving Mode, do the following:

Step 1. Set **mclkdiv** to '0' following the procedure in section (See '(C) Changing the System Clock Source, MCLK, GCLK or WCLK Division Factor' on page 4-79).

Step 2. Program the System PLL to normal frequency following the procedure in section (See '(B) Changing the System PLL Frequency' on page 4-79).

Step 3. Program **rfhcnt** to its normal value.

Step 4. Power up the **Pixel PLL**.

Step 5. Power up the **LUT**.

Step 6. Power up the **DAC**.

Step 7. Set the **pixclkdis** field of **XPIXCLKCTRL** to '0', or reprogram the **Pixel PLL** to a new operating frequency if desired by following the procedure in section (See '(A) Changing the Pixel Clock Frequency or Source' on page 4-79).

Step 8. Reset bits **scroff**, **vsyncoff** and **hsyncoff** to '0'.

4.13 Accessing the Serial EEPROM

The page write sizes of serial eeproms may vary between manufactuers.The MGA-G200 is designed to support up to 16 bytes for a small serial eeprom (128 to 512 bytes) and up to 128 bytes for a large serial eeprom (32 or 64 Kbytes).

It is possible to access any combination of bytes in a dword of the serial eeprom (reading or writing). There is a read or a write cycle for each non-consecutive byte in a dword, and for every dword, except when a page write is possible (in this case there will be one write cycle for the entire page write).

To access the serial eeprom, **biosen** and **romen** must be set to '1', and **rombase** must be mapped.

To write the serial eeprom, **eepromwt** must also be set to '1'.

The process of writing the serial eeprom is as follows:

Step 1. Find the size of the serial eeprom

Step 2. Find the size of the page write

Step 3. Decide how to reorder the data to be written

Step 4. Execute the writes

Step details:

1. The size of the serial eeprom is indicated by the reset value of **biosen** (biosboot).
   - '0': small serial eeprom
   - '1': big serial eeprom
2. To determine the size of the page write:

   If biosboot = '0':

   a) Write a DW at address 0x04 of the serial eeprom.
   b) Write a DW at address 0x08 of the serial eeprom.
   c) Read a DW at address 0x08 of the serial eeprom.
   d) If the data is the one written in b), then the page write size is 16 bytes. If not, the page write size is 8 bytes.

   If biosboot = '1':

   a) Write a DW at address 0x3C of the serial eeprom.
   b) Write a DW at address 0x40 of the serial eeprom.
   c) Read a DW at address 0x40 of the serial eeprom.
   d) If the data is the one written in b), then the page write size is 128 bytes. If not, the page write size is 64 bytes.

   At present, 8/16 bytes and 64/128 bytes are the only known page write sizes. If there was a smaller page size, the procedure would continue with 5), 6), 7), 8) at half the addresses to determine the proper page size.

   In the event of a bigger page size, MGA-G200 could not support a page write only version of that serial eeprom.

   If the page size is 16 bytes (for a small serial eeprom) or 128 bytes (for a big serial eeprom), proceed to step 4.

3. The size of the page write is smaller than the one hard-coded in the MGA-G200. That means that care must be taken when sending the writes to the MGA-G200's serial eeprom. The MGA-G200 has a 2 dword ROMFIFO serving as a dword accumulator. The MGA-G200 will perform a page write when the ROMFIFO is full, the addresses are consecutive, both dwords are writes, and all the bytes within the dwords are enabled. If the transferred dword is the last before the boundary of the page write size (0x7C for a big serial eeprom), then, after transferring the data, the MGA-G200 will stop the page write. If the real page write size of the serial eeprom is smaller, the second half of the page size will always be written over the first half.

   ➽ ***Note:*** To avoid this write the serial eeprom starting from the end, by blocks of its page write size.

Example:

128 byte serial eeprom with an 8 byte page write size.

Access	*beN*	*Address*	
write	0x0	0X78	one page write (8 bytes)
write	0x0	0x7c	
write	0x0	0x70	a second page write
write	0x0	0x74	
write	0x0	0x68	
write	0x0	0x6C	
…	…	…	
write	0x0	0x08	
write	0x0	0x0C	
write	0x0	0x00	
write	0x0	0x04	

Other possible solutions are inserting a delay or reading from the serial eeprom between the write transfers at the end of the page write. The solution displayed above will always prove to be faster since accessing the serial eeprom is a long process (the clock runs at less than 5 MHz per bit). In reality, the blocks could be re-arranged in any way ***except*** consecutively. This can be done even when the page write size is the biggest one, as long as the blocks are at least the size of the page write. If the blocks are smaller, there will still be a page write, but not as much data will be transferred, and you will still have to wait for the write cycle time to complete before being able to transfer more data.

4. Transfer the data to the MGA-G200. When a page write occurs, no accesses are transferred to the serial eeprom until the write cycle time is over (up to 10 ms), therefore, the ROMFIFO will remain full for a long time and will force retrys on the PCI bus when accessed. There is ***no*** way to determine if the ROMFIFO is full. If retrys are an issue, insert delays between page write transfers, or program the **noretry** bit to '1' in the **OPTION** register.

Chapter 5: Hardware Designer's Notes

5.1 Introduction

The MGA-G200 chip has been designed to minimize the amount of external logic required to build a board. Included among its features are:

- Direct interface to the PCI bus or AGP bus
- All necessary support for external devices such as ROM
- Direct connection to the RAM
- Direct interface with the video and feature connectors

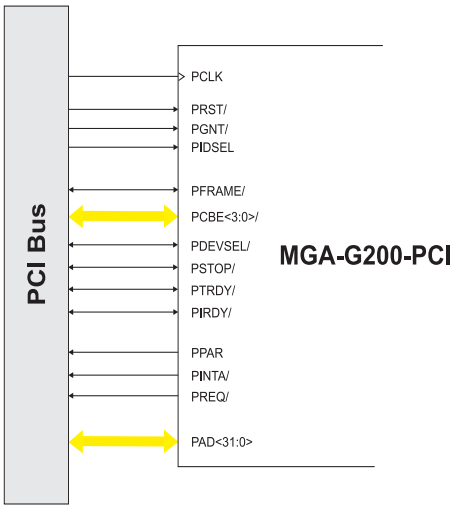
5.2 Host Interface

5.2.1 PCI Interface

The MGA-G200-PCI interfaces directly with PCI as shown in Figure 5-1. The MGA-G200-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-G200-PCI can also act as a master on the PCI bus - refer to Section 4.1.9 for more information.

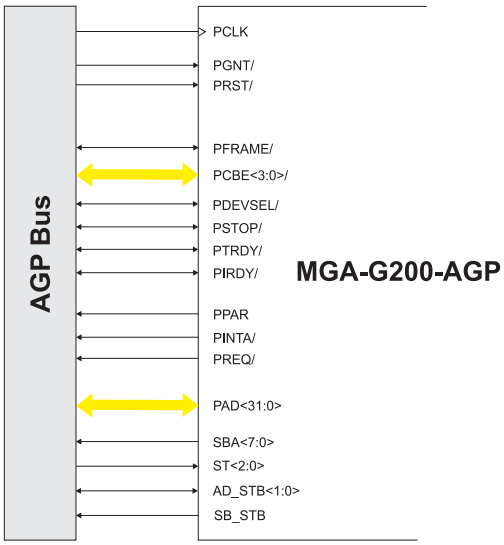
Figure 5-1: PCI Interface



5.2.2 AGP Interface

The MGA-G200-AGP interfaces with the AGP bus as shown in Figure 5-2. The MGA-G200-AGP acts as an AGP master and a PCI target. The AGP master uses the AGP sideband signal in 1X and 2X modes addressing mechanism. The PCI target is a medium speed device (it responds with PDEVSEL/ during the second clock after PFRAME/ is asserted).

Figure 5-2: AGP Interface



5.3 Snooping

The MGA-G200 performs snooping when VGA I/O is enabled *and* snooping is turned *on*. In this case, two things may occur when the DAC is written to:

1. If the MGA-G200 is *unable* to process the access immediately, it takes control of the bus and performs a retry cycle.
2. If the MGA-G200 is *able* to process the access, the access is snooped, and the MGA-G200 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.

5.4 EEPROM Devices

The MGA-G200 supports a few external devices (the SPI-EEPROM is a standard expansion device that is supported by the MGA-G200).

Figure 5-3 shows how to connect the serial eeprom devices to the MGA-G200.

BIOS EPROM

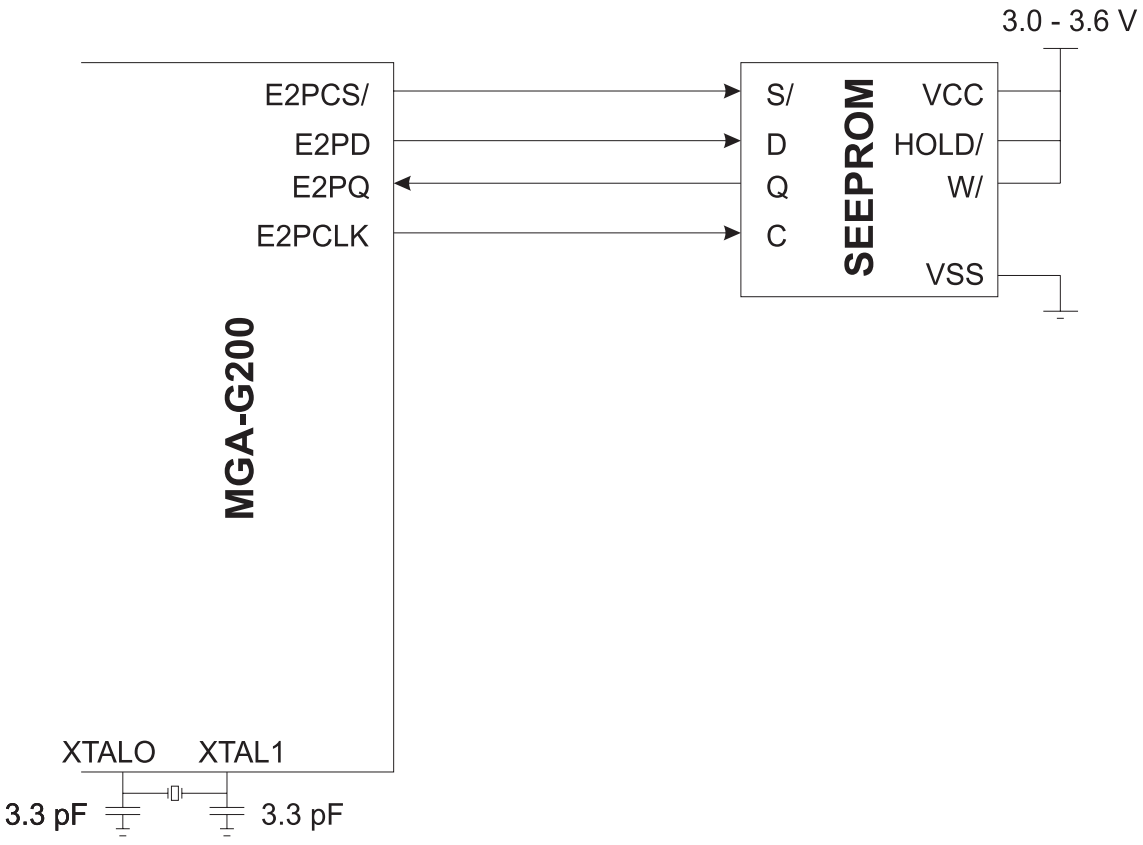
The MGA-G200 supports 32K x 8 EEPROMs, 64K x 8 EEPROMS, and 128 byte EEPROMS. The following table lists specific EEPROM devices that have been verified to work with the MGA-G200.

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

Note, however, that sequencing of operations to write the memory must be performed by software. Additionally, some requirements must be guaranteed by software (refer to the device specification and 'Accessing the Serial EEPROM' on page 4-118).

	Serial EEPROM		
Manufacturer	***32K x 8***	***64K x 8***	***128 x 8***
Atmel	AT 25256		AT 25010
SGS -Thomson	M35560		ST95010

Figure 5-3: External Device Configuration



Note: If a local oscillator is used instead of crystal, it is connected to XTAL0, and XTAL1 is left unconnected.

5.5 Memory Interface

5.5.1 SGRAM Configurations

The principal characteristics of the MGA-G200's SGRAM interface are provided in table 5-1, which identifies the cycles that are supported by the chip, and lists all of the commands generated by the MGA-G200.

Table 5-1: Supported SGRAM/SDRAM Commands

Command[1]	***Mnemonic***	***MCS/***	***MRAS/***	***MCAS/***	***MWE/***	***MDSF***	***MDQMi***	***BS***	***AP***	***Address***
Mode Register Set	MRS	L	L	L	L	L	X	L	L	opcode1[2]
Special Mode Register Set	SMRS	L	L	L	L	H	X	L	L	opcode2[3]
Auto Refresh	REF	L	L	L	H	L	X	X	X	X
Bank Activate / Row Address (Mask Disabled)	ACTV	L	L	H	H	L	X	X	Row Address[4]	
Bank Activate / Row Address (Mask Enabled)	ACTM	L	L	H	H	H	X	V	Row Address[4]	
Read / Column Address (Auto-Precharge Disabled)	READ	L	H	L	H	L	X	V	L	Column Address[5]
Write / Column Address (Auto-Precharge Disabled)	WRITE	L	H	L	L	L	X	V	L	Column Address[5]
Block Write / Column Address (Auto-Precharge Disabled)	BWRIT	L	H	L	L	H	X	V	L	Column Address [5][6]
Precharge (Single Bank)	PRE	L	L	H	L	L	X	V	L	X
Precharge (Both Banks)	PALL	L	L	H	L	L	X	X	H	X
No Operation	NOP	L	H	H	H	L	X	X	X	X
Device De-select	DESL	H	X	X	X	X	X	X	X	X
Mask Write Data / Disable Read Output	X	X	X	X	X	X	H	X	X	X[7]
Write Data / Enable Read Output	X	X	X	X	X	X	L	X	X	X[7]
Legend: H = Logical High, L = Logical Low, V = Valid, X = "Don't Care", '/' Indicates an active low signal.										

[1] MCS = MCS<3:0>

(2) The MGA-G200 supports CAS latency (CL=2, CL=3, CL=4, CL=5); burst type = sequential; burst length = 4.
opcode1 = mrsopcod[3:0]: CLbits: '0': '010'. (Usually '00000110010')

CLbits	Caslatency
'010'	2
'011'	3
'100'	4
'101'	5

The mrsopcod value is a programmable field in the MEMRDBK register.

(3) A5 = 1 for a mask register access, and A6 = 1 for a color register access. Both registers cannot be accessed simultaneously.
opcode2 = '00000100000' <- load mask register
opcode2 = '00001000000' <- load color register

(4) For 2-bank, 8 MBit SGRAM device: Row Address = MA<x> for MA<8:0>
For 2-bank, 16 MBit SGRAM device: Row Address = MA<x> for MA<9:0>
For 4-bank, 16 MBit SGRAM device: Row Address = MA<x> for MA<8:0>
For 2-bank, 16 MBit SDRAM device: Row Address = MA<x> for MDSF and MA<9:0>

(5) The MGA-G200 does not support the auto-precharge function, so AP will always be forced low for READ/WRITE/BWRIT commands.
Column Address = MA<7:0>

(6) MA<2:0> are 'don't care' for block write commands.

(7) Not a command - "MDQ mask enable'

➧ ***Note:*** The MGA-G200 does not drive CKE: it should be driven high externally.

➧ ***Note:*** The number of address bits depends on the memory type (selected by the memconfig field of OPTION). The addresses are mapped as follows:

Table 5-2: 10-Bit Address Configuration (memconfig <2:0>= 00x for 2-bank 8Mb(x32) device

MCS<3:0>/		MA										
		10	*9*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
1110 or 1101	Row	A11	A20	'0'	A19	A18	A17	A16	A15	A14	A13	A12
	Column	A11	'0'	'0'	A10	A9	A8	A7	A6	A5	A4	A3
1011 or 0111	Row	'0'	A11	A20	A19	A18	A17	A16	A15	A14	A13	A12
	Column	'0'	A11	'0'	A10	A9	A8	A7	A6	A5	A4	A3

➧ ***Note:*** The BA0 and AP bits are remapped for the expansion for the memory SO-DIMM.
MA10 and MA9 = = BA0 and AP bits for base memory.
MA9 and MA8 = = BA0 and AP bits for expansion memory.

Table 5-3: 11-Bit Address Configuration (memconfig <2:0>=01x) for 2-bank, 16Mb(x32) device

	MA										
	10	*9*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Row	A11	A21	A20	A19	A18	A17	A16	A15	A14	A13	A12
Column	A11	'0'	'0'	A10	A9	A8	A7	A6	A5	A4	A3

Table 5-4: 12-Bit Address Configuration (memconfig <2:0>=10x) for 2-bank, 16Mb(x16) device

	MDSF	***MA***										
		10	***9***	***8***	***7***	***6***	***5***	***4***	***3***	***2***	***1***	***0***
Row	A22	A11	A21	A20	A19	A18	A17	A16	A15	A14	A13	A12
Column	‘0’	A11	‘0’	‘0’	A10	A9	A8	A7	A6	A5	A4	A3

➧ ***Note:*** No remapping is performed for SO-DIMM access
MA10 & MA9 = = BA0 & AP bits
For SDRAM(x16), connect as follows:

Eclipse:	MA10	MA9	MDSF	MA8	…	MA0
	↓	↓	↓	↓		↓
SDRAM:	A11	A10	A9	A8	…	A0

Table 5-5: 11-Bit Address Configuration (memconfig<2:0>=11x) for 4-bank, 16Mb(x32) device

MCS<3:0>/		***MA***										
		10	***9***	***8***	***7***	***6***	***5***	***4***	***3***	***2***	***1***	***0***
1110 or 1101	Row	A11	A20	A21	A19	A18	A17	A16	A15	A14	A13	A12
	Column	A11	‘0’	A21	A10	A9	A8	A7	A6	A5	A4	A3
1011 or 0111	Row	A21	A11	A20	A19	A18	A17	A16	A15	A14	A13	A12
	Column	A21	A11	‘0’	A10	A9	A8	A7	A6	A5	A4	A3

➧ ***Note:*** The BA* and AP bits are remapped for the expansion memory SO-DIMM
MA10, MA9 and MA8 = = BA0, AP and BA1 bits for base memory
MA10, MA9 and MA8 = = BA1, BA0 and AP bits for expansion memory

Figure 5-4: Memory Interface Connection (memconfig = '00X')

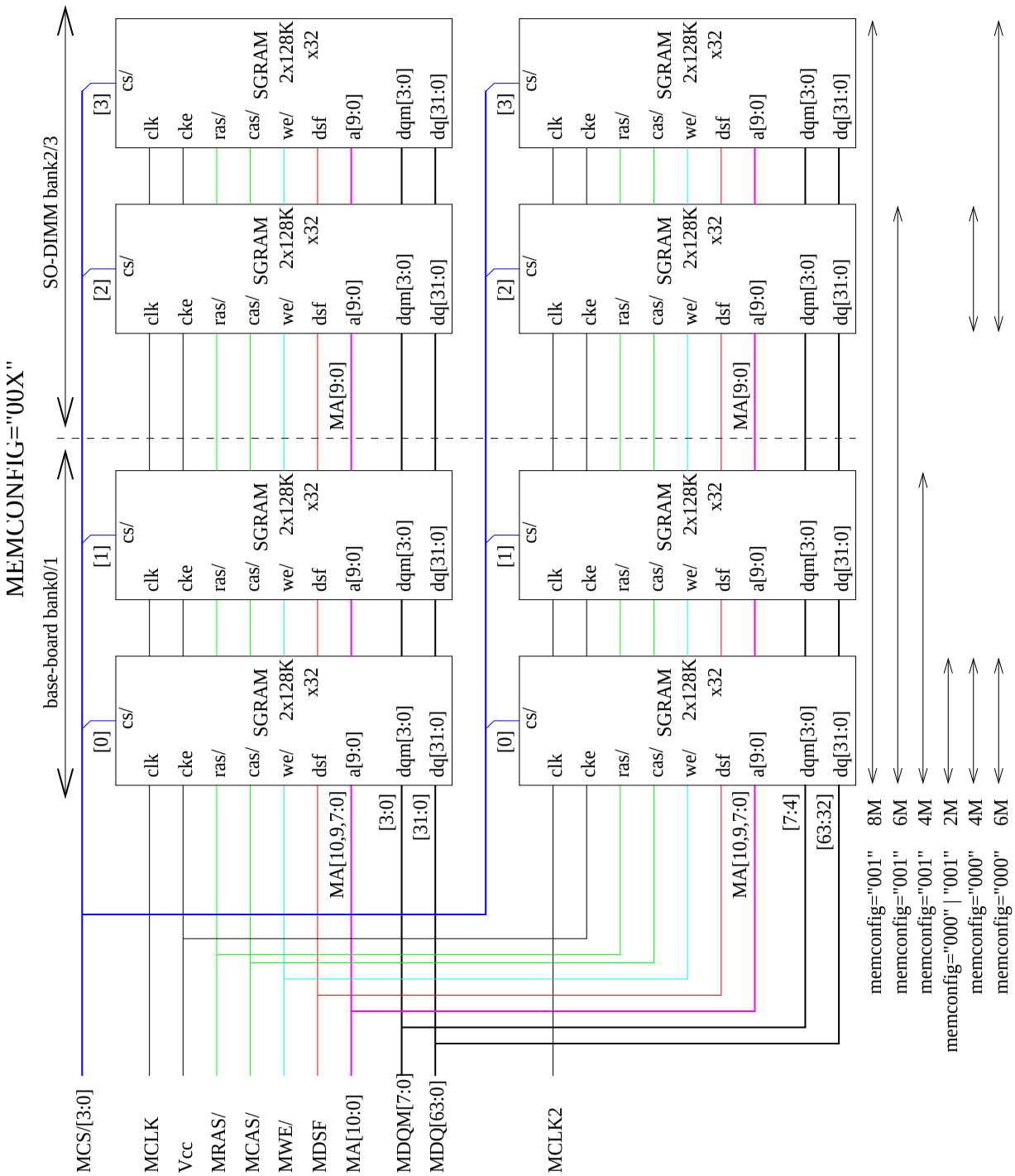


Figure 5-5: Memory Interface Connection (memconfig = '01X')

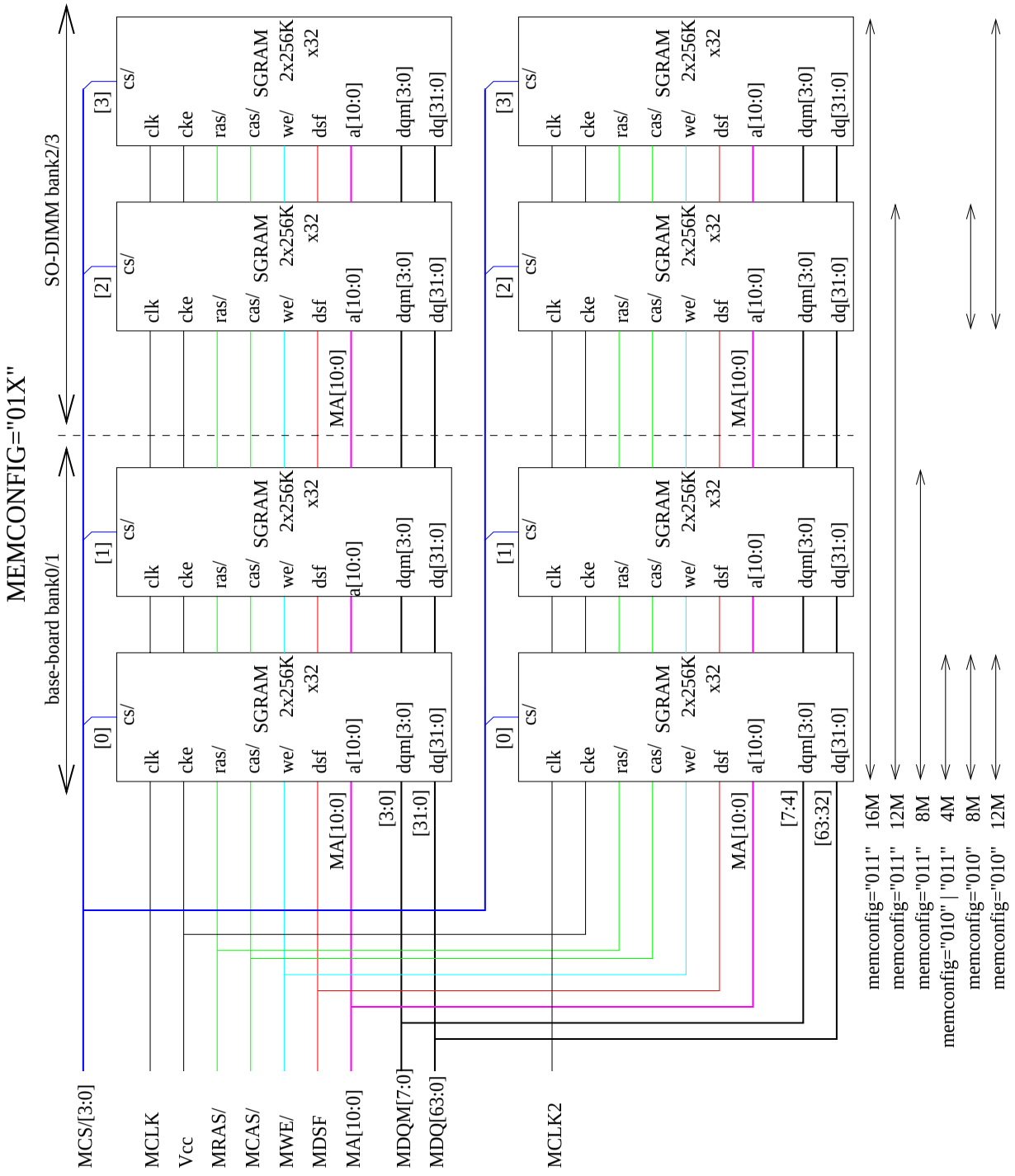


Figure 5-6: Memory Interface Connection (memconfig = '10X')

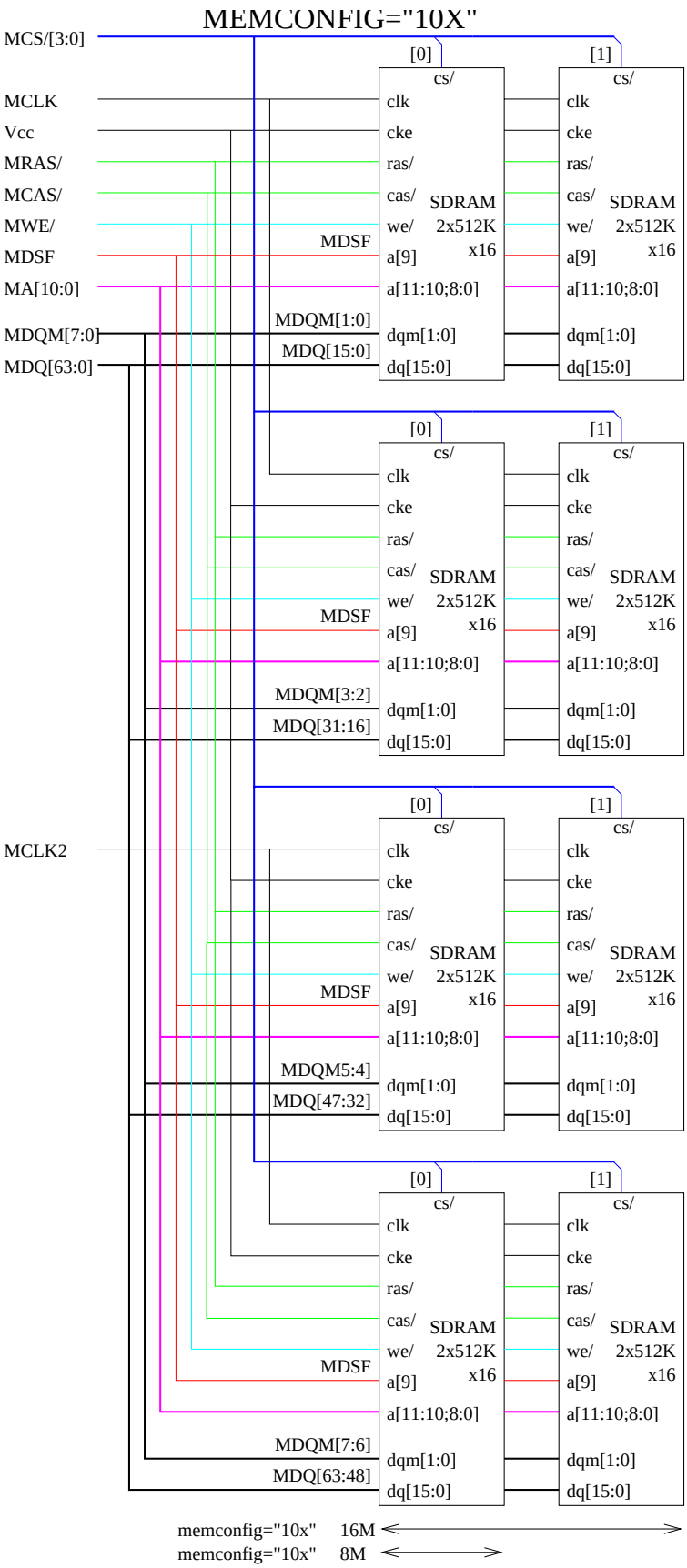
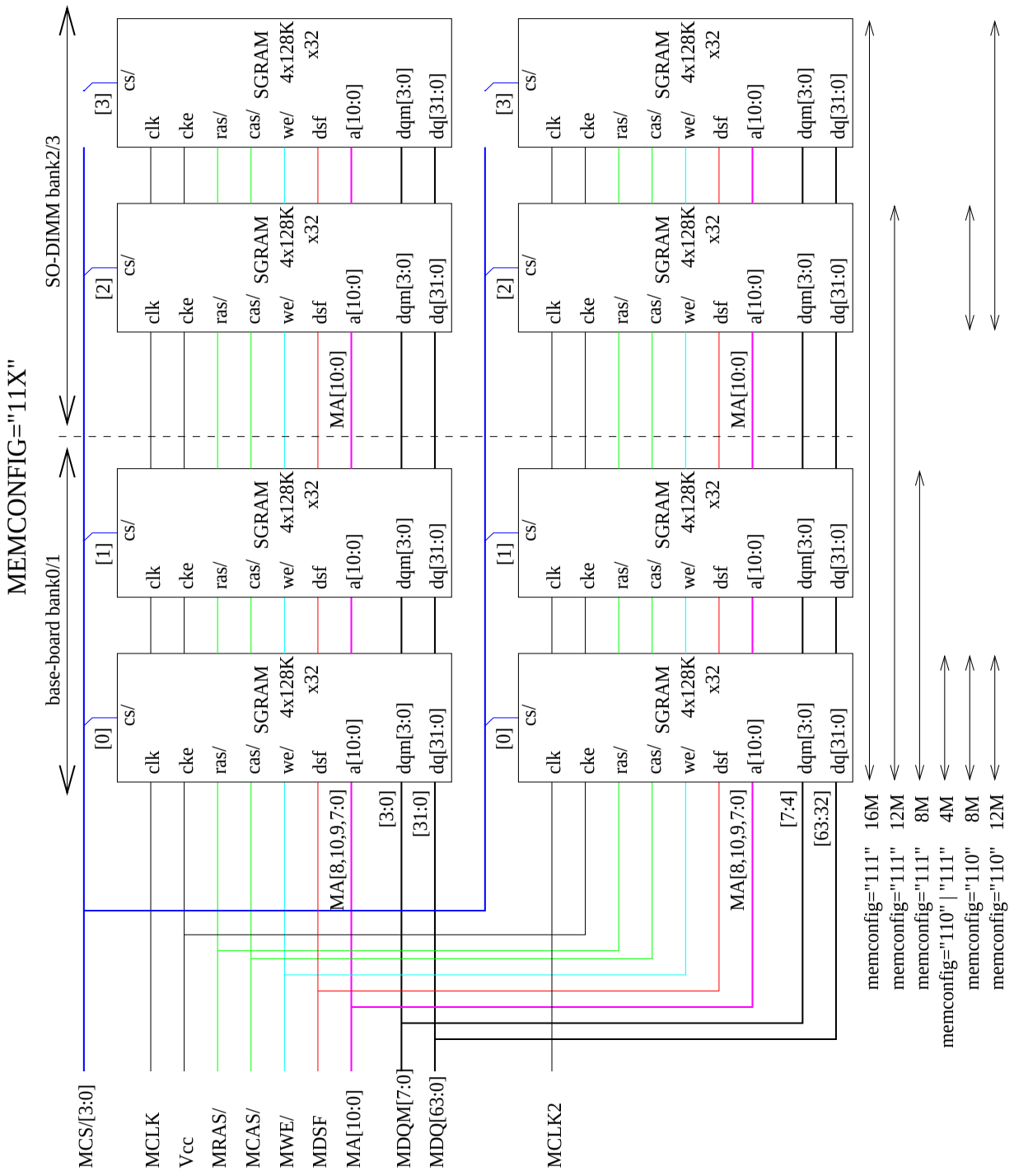


Figure 5-7: Memory Interface Connection (memconfig = '11X')



5.6 Video interface

5.6.1 Slaving the MGA-G200

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

The first time that the MGA-G200's CRTC is synchronized, the data may be corrupted for up to one complete frame. However, when the CRTC is already synchronous and a reset occurs, the CRTC will behave as if there was no VIDRST.

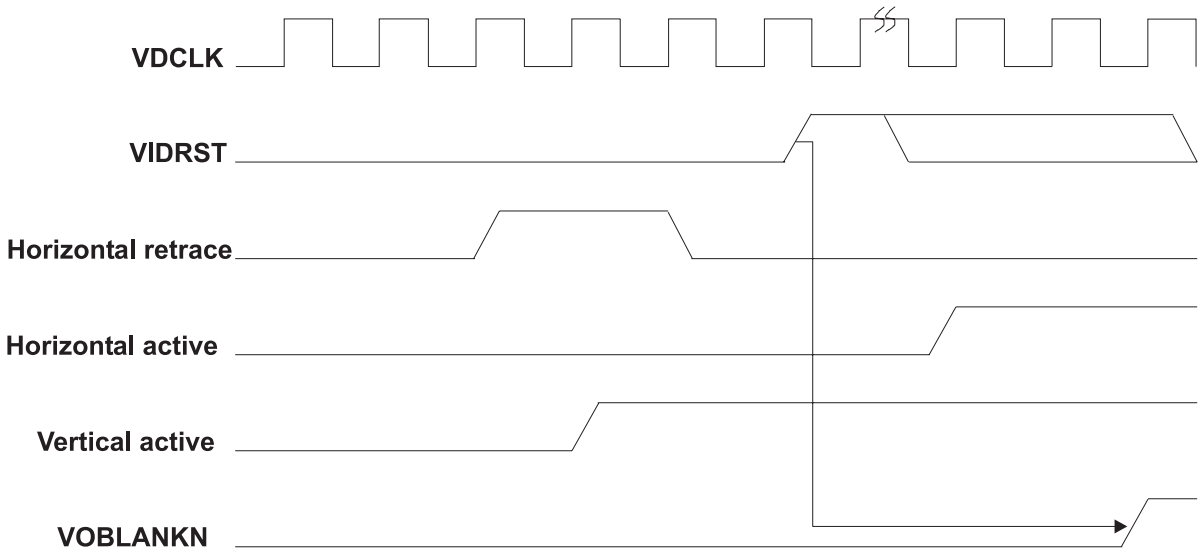
Note: In order for the MGA-G200 to be synchronous with any other source, the MGA-G200 CRTC must be programmed with the same video parameters as that other source. VDOCLK can also be modulated in order to align both CRTCs.

The **hrsten** field of the **CRTCEXT1** register is used to enable the horizontal reset, which sets the horizontal and character counters to the beginning of the horizontal SYNC.

The **vrsten** field of the **CRTCEXT1** register is used to enable the vertical reset, which sets the vertical counter to the beginning of the vertical SYNC in the even field.

Horizontal active and horizontal retrace are not affected by VIDRST when only the vertical reset is active. Figure 5-8 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 5-8: VIDRST, Internal Horizontal/Vertical Active, and VOBLANKN



5.6.2 Genlock Mode

The **VIDRST** pin can be used to reset the **CRTC** horizontally and/or vertical counters. The **VIDRST** ***must*** be maintained for at least 1 **VDOCLK** cycle for the reset to take effect in the MGA-G200. When it is ***not*** used, the **VIDRST** pin ***must*** be maintained low (there is ***no*** enable/disable control bit for the **VIDRST** pin).

If the timing on the VIDRST pin is respected, the reset operation on the chip will be completed (the VBLANK/ pin is set to '1'), according to the number of VDOCLKs shown in the following table:

Pixel Width	*Delay to Video Pin (VDOCLKs)*
BPP8	31
BPP15	21
BPP16	21
BPP24	17
BPP32	16

➽ ***Note:*** Genlocking is ***not*** supported in VGA mode.

5.6.3 Crystal Resonator Specification

Frequency	27 MHZ
Equivalent series resistance (Rs)	35 - 200 Ω
Load capacitance (Cl)	18 or 20 pF (series ***or*** parallel)
Shunt capacitance (Co)	7 pF max.
Drive level	100 - 1,000 µW
Temperature stability	50 ppm

Figure 5-9: Video Interface

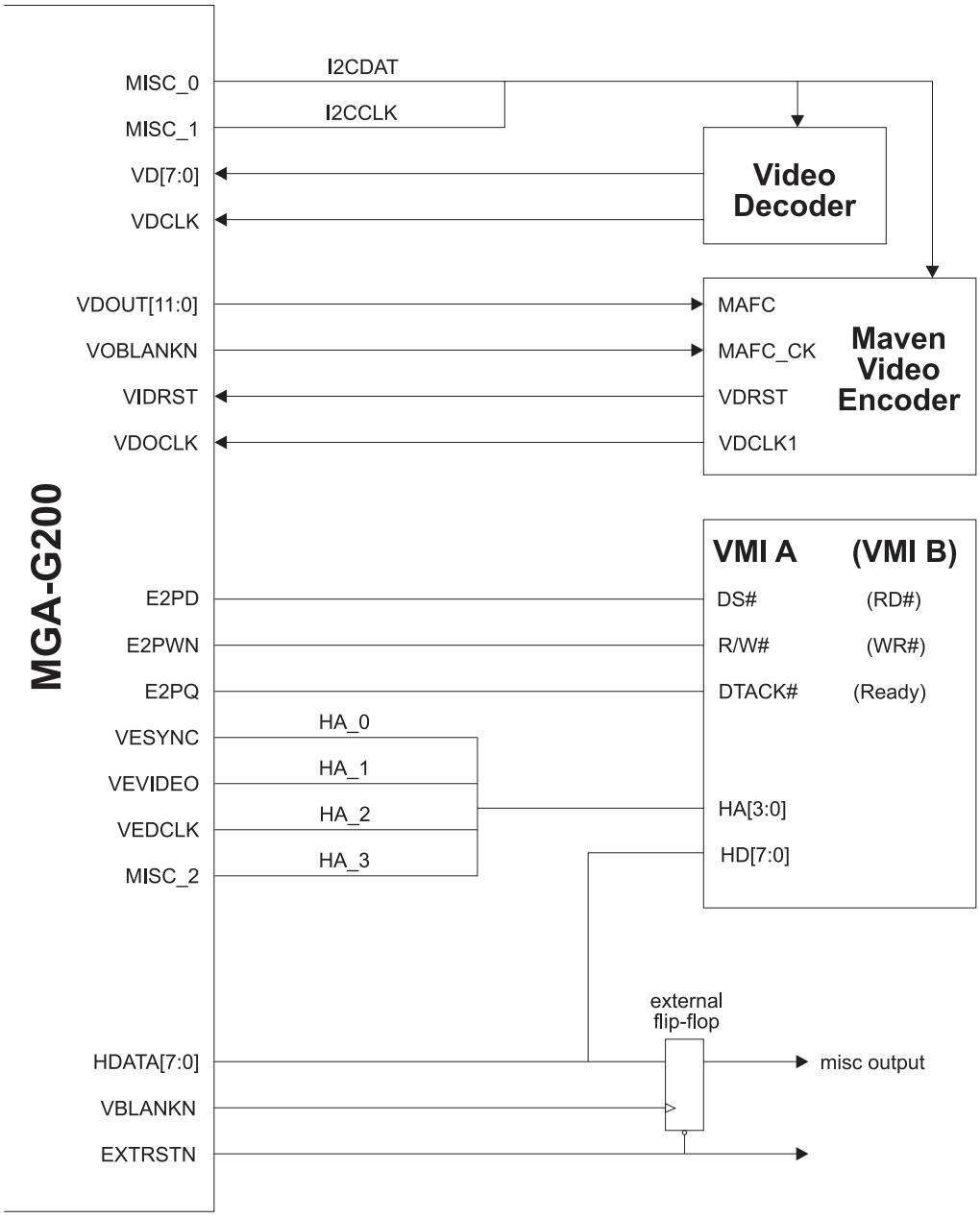
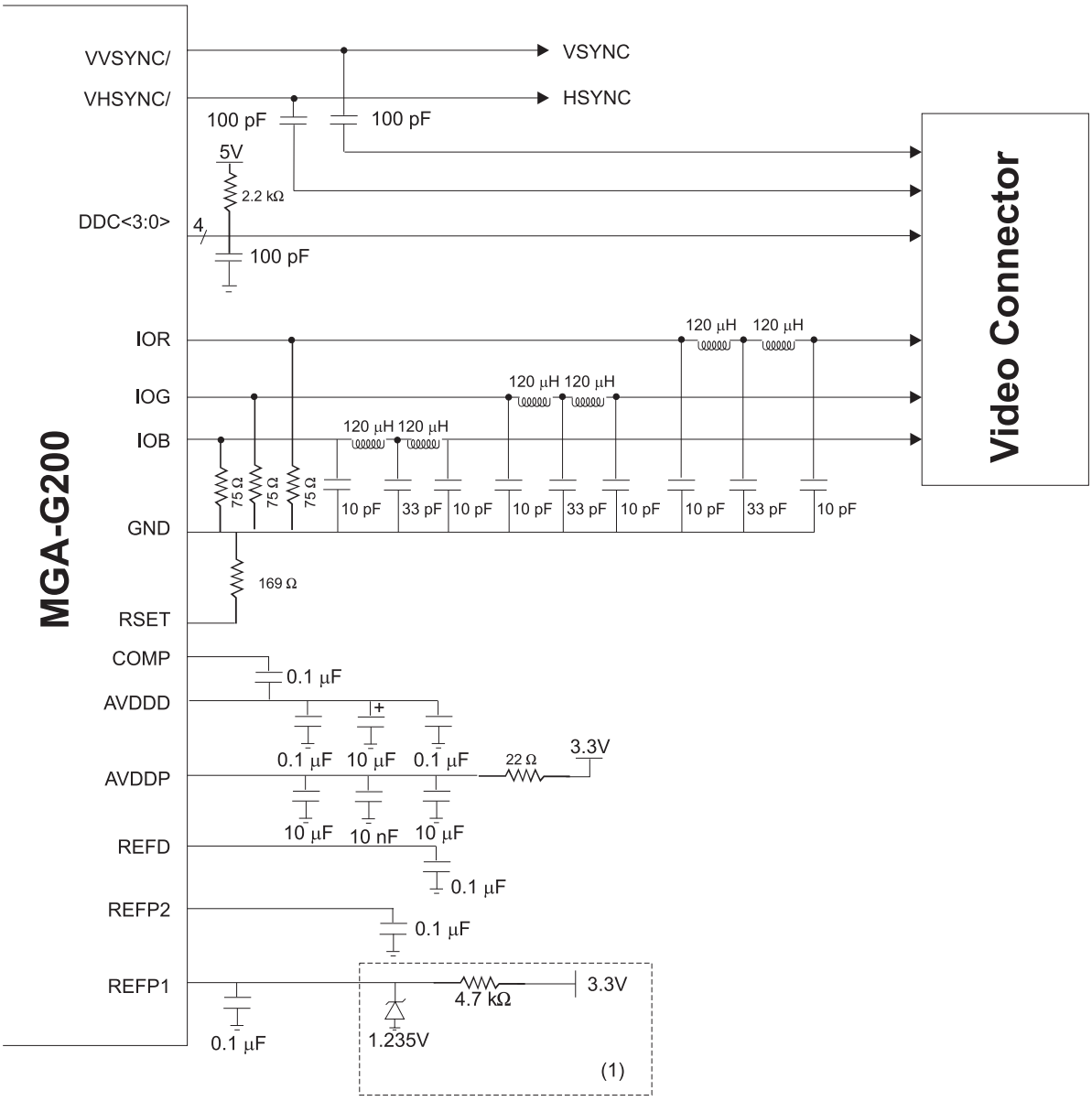


Figure 5-10: Video Connector



Appendix A: Technical Information

A.1 Pin List

Table A-1: Pin Count Summary MGA-G200-PCI

Group	*Total*	*I*	*O*	*I/O*
Host PCI	48	4	3	41
Host Local	2	1	1	
Memory	93	—	28	65
Video Display	6	—	2	4
Video In	9	9	—	—
MAFC/Video Out	15	1	13	1
CODEC	13	2	3	8
SEEPROM	4	1	3	—
Analog Signals	11	6	5	—
Miscellaneous Functions	3	—	—	3
Test	3	3	—	—
VCC/GND	102	—	—	—
Reserved	11	—	—	—
PBGA Total	320	—	—	—

Table A-2: Pin Count Summary MGA-G200-AGP

Group	*Total*	*I*	*O*	*I/O*
Host PCI	61	6	12	43
Host Local	2	1	1	—
Memory	93	—	8	65
Video Display	6	—	2	4
Video In	9	9	—	—
MAFC/Video Out	15	1	13	1
CODEC	13	2	3	8
SEEPROM	4	1	3	—
Analog Signals	12	7	5	—
Miscellaneous Functions	3	—	—	3
Test	3	3	—	—
VCC/GND	98	—	—	—
Reserved	1	—	—	—
PBGA Total	320	—	—	—

A.1.1 Host (PCI / AGP)

Name	*# Pins*	*Type*	*Description*
PAD<31:0>	32	I/O	PCI address and data bus. During the address phase of a PCI transaction, PAD contains a physical address. During the data phase, it contains the data that is read or written.
SB_STB	1	O	AGP Sideband address strobe. While in AGP 2X mode, the strobe is used to qualify the information on the SBA bus on each of its edges. MGA-G200-AGP only.
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. Is asserted when a transaction is within the MGA address range and space.
PFRAME/	1	I/O	Cycle frame. Indicates the beginning of an access and its duration.
AD_STB<1:0>	2	I/O	AGP Data Strobe.While in AGP 2X mode, the strobe is used to qualify the information on the PAD bus on each of its edges. MGA-G200-AGP only.
PGNT/	1	I	Grant. Indicates to the MGA-G200 that access to the PCI bus has been granted.
PIDSEL	1	I	Initialization device select. Used as a chip select during configuration read and write transactions. MGA-G200-PCI only.
PINTA/	1	O	Interrupt request signal.
PIRDY/	1	I/O	Initiator ready. Indicates the initiating agent's ability to complete the current data phase of the transaction (used in conjunction with PTRDY/). Wait cycles are inserted until both PIRDY/ and PTRDY/ are asserted together.
PPAR	1	O	PCI even parity bit for the PAD<31:0> and PCBE<3:0>/ lines. Parity is generated during read data phases and during the address phase throughout the PCI mastering cycle.
PREQ/	1	O	Request. Indicates to the arbiter that the MGA-G200 wishes to use the bus.
PRST/	1	I	PCI reset. This signal is used as the chip's hard reset.
PSTOP/	1	I/O	Stop. Forces the current transaction to terminate.
PTRDY/	1	I/O	Target ready. When asserted, indicates that the current data phase of the transaction can be completed (used in conjunction with PIRDY/). Wait cycles are inserted until both PIRDY/ and PTRDY/ are asserted together. In target mode, PTRDY/ is used as an input for snooping operations.
SBA<7:0>	8	O	Sideband Address port. Provides an additional bus to pass addresses and commands.(***only*** for the MGA-G200-AGP).
ST<2:0>	3	I	Status. Provides additional information from the arbitor indicating what the MGA-G200-AGP may do when it receives a PGNT/.

A.1.2 Host (Local Mode)

Name	*# Pins*	*Type*	*Description*
EXTINT/	1	I	External interrupt pin. Can be used by a companion chip to generate interrupts on the PCI bus. Interrupt is an active low level interrupt.
EXTRST/	1	O	External reset signal. Used to reset the external devices.

A.1.3 Memory Interface

Name	*# Pins*	*Type*	*Description*
MCS<3:0>/	4	O	Memory Chip Select
MA<10:0>	11	O	Memory addresses (row, column, bank: multiplexed). MA<10:0> are used as either bank select or memory addresses, depending on the type of memory.
MRAS/	1	O	Memory Row Address Strobe
MCAS/	1	O	Memory Column Address Strobe
MWE/	1	O	Memory Write Enable
MDSF	1	O	Memory special function eable.
MDQ<63:0>	64	I/O	Memory data inputs/outputs
MDQM<7:0>	8	O	Memory Data mask enable
MCLK	1	I/O	Memory Clock for MDQ<31:0>
MCLK2	1	O	Memory Clock for MDQ<63:32>

A.1.4 Video Display Interface

Name	*# Pins*	*Type*	*Description*
DDC<3:0>	4	I/O	Display Data Channel. Used to communicate with monitor.
VHSYNC/	1	O	Horizontal Sync.
VVSYNC/	1	O	Vertical Sync.

A.1.5 Video In Interface

Name	*# Pins*	*Type*	*Description*
VD<7:0>	8	I	Video In Data
VDCLK	1	I	Video In Clock. Maximum: 27 MHz.

A.1.6 Video Out Interface

Name	*# Pins*	*Type*	*Description*
VDOCLK	1	I/O	Video Out Clock. (Input for mafc, output in Panel link mode.)
VDOUT<11:0>	12	O	Video Out Data.
VOBLANK/	1	O	Video Out Blank. Gated clock feedback in mafc mode.
VIDRST	1	I	Video Reset Input. Used to synchronize the CRTC on an external source. The VIDRST pin must be forced inactive when not in use.

A.1.7 CODEC Interface

Name	*# Pins*	*Type*	*Description*
VBLANK/	1	O	Data strobe for external register
VEDCLK	1	O	CODEC address <2>
VEVIDEO	1	O	CODEC address <1>
VESYNC	1	O	CODEC address <0>
HDATA <7:0>	8	I/O	CODEC data port. Also used for chip strapping HDATA <0> VGA boot strap HDATA <1> BIOS EEPROM installed strap
E2PW/	1	O	CODEC interface write signal when CODEC interface is enabled

A.1.8 SEEPROM

Name	*# Pins*	*Type*	*Description*
E2PCLK	1	O	Clock for the Serial EEPROM.
E2PCS/	1	O	Serial EEPROM chip select.
E2PD	1	O	Serial EEPROM write data. Also, the Codec Interface read signal when Codec interface is enabled.
E2PQ	1	I	Serial EEPROM read data. Also, the Codec Interface read/write acknowledge signal when Codec interface is enabled.

A.1.9 Analog Signals

Name	*# Pins*	*Type*	*Description*
COMP	1	O	Compensation. COMP provides compensation for the interanl reference amplifier. A 0.1 uF ceramic capacitor is required between COMP and analog ground. The capacitor must be as close to the device as possible to avoid noise pick-up.
IOB	1	O	Analog current Output Blue. This output can drive a 37.5Ω load directly (doubly-terminated 75Ω load).
IOG	1	O	Analog current Output Green. This output can drive a 37.5Ω load directly (doubly-terminated 75Ω load).
IOR	1	O	Analog current Output Red. This output can drive a 37.5Ω load directly (doubly-terminated 75Ω load).
REFAGP	1	I	Voltage reference for AGP PLLs.
REFD REFP1 REFP2	3	I	Voltage reference for DACs and PLLS. An internal voltage reference of nominally 1.234 V may or may not be provided, which would require an external 0.1 uF ceramic capacitor between REF ans analog GND. However, the internal reference voltage can be overdirven by an externally supplied reference voltage.
REFSSTL	1	I	Reference voltage for the SSTL/LVTTL I/O buffers.
RSET	1	I	Full-scale adjustment pin. A resistor connected between this pin and ground controls the full-scale range of the DACs.
XTAL1 XTAL0	1 1	O I	Connection for a series of resonant crystals as a reference for the frequency synthesizer PLLs. XTAL0 may be used as a TTL reference clock (local oscillator) input, in which case XTAL1 is left unconnected.

A.1.10 Miscellaneous Functions

Name	*# Pins*	*Type*	*Description*
MISC<2:0>	3	I/O	Miscellaneous Control. General purpose I/O pins. MISC <0> can be used for I2CDAT MISC <1> can be used for I2CCLK MISC <2> can be used for CODEC Address <3> or EOI/

A.1.11 TEST

Name	*# Pins*	*Type*	*Description*
TST<2:0>	3	I	These pins place the chip in test mode. They should be tied to a pull-up during normal operation.

A.1.12 AGP VDD/GND

Name	*# Pins*	*Type*	*Description*
AVDDD	3	-	Analog; attaches to +3.3 volts
AVDDP	3	-	Analog; attaches to +3.3 volts
GND	77	-	Attaches to ground
AGNDP	3	-	Analog Ground
VDD	12	-	Attaches to +3.3 volts.

A.1.13 PCI VDD/GND

Name	*# Pins*	*Type*	*Description*
AVDDD	3	-	Analog; attaches to +3.3 volts.
AVDDP	2	-	Analog; attaches to +3.3 volts
GND	77	-	Attaches to ground
AGNDP	2	-	Analog Ground
VDD	12	-	Attaches to +3.3 volts
VDD5	6	-	Attaches to +5 volts

A.2 PCI Pinout Illustration and Table

The illustration below shows the locations of the MGA-G200's 320 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

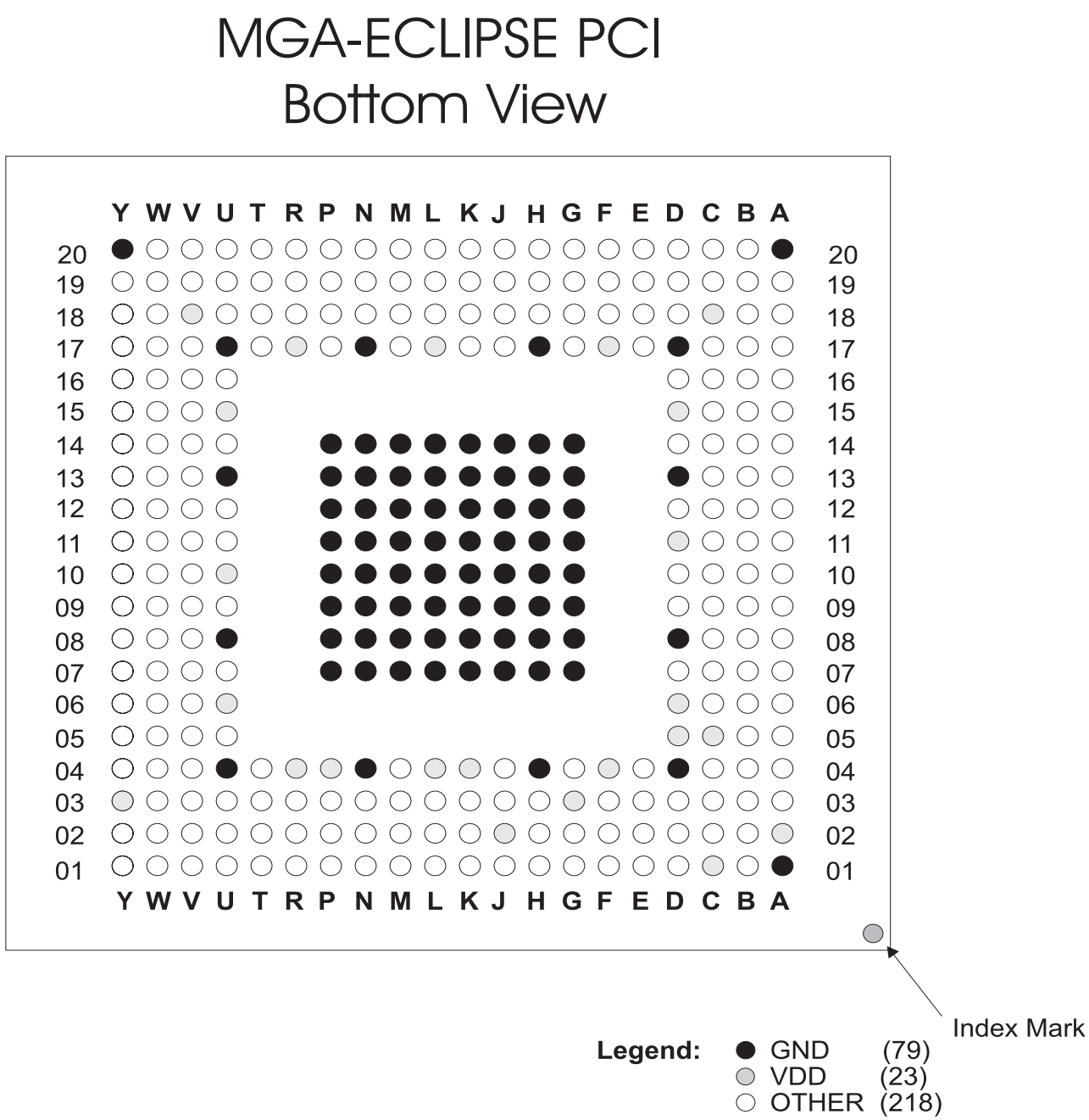
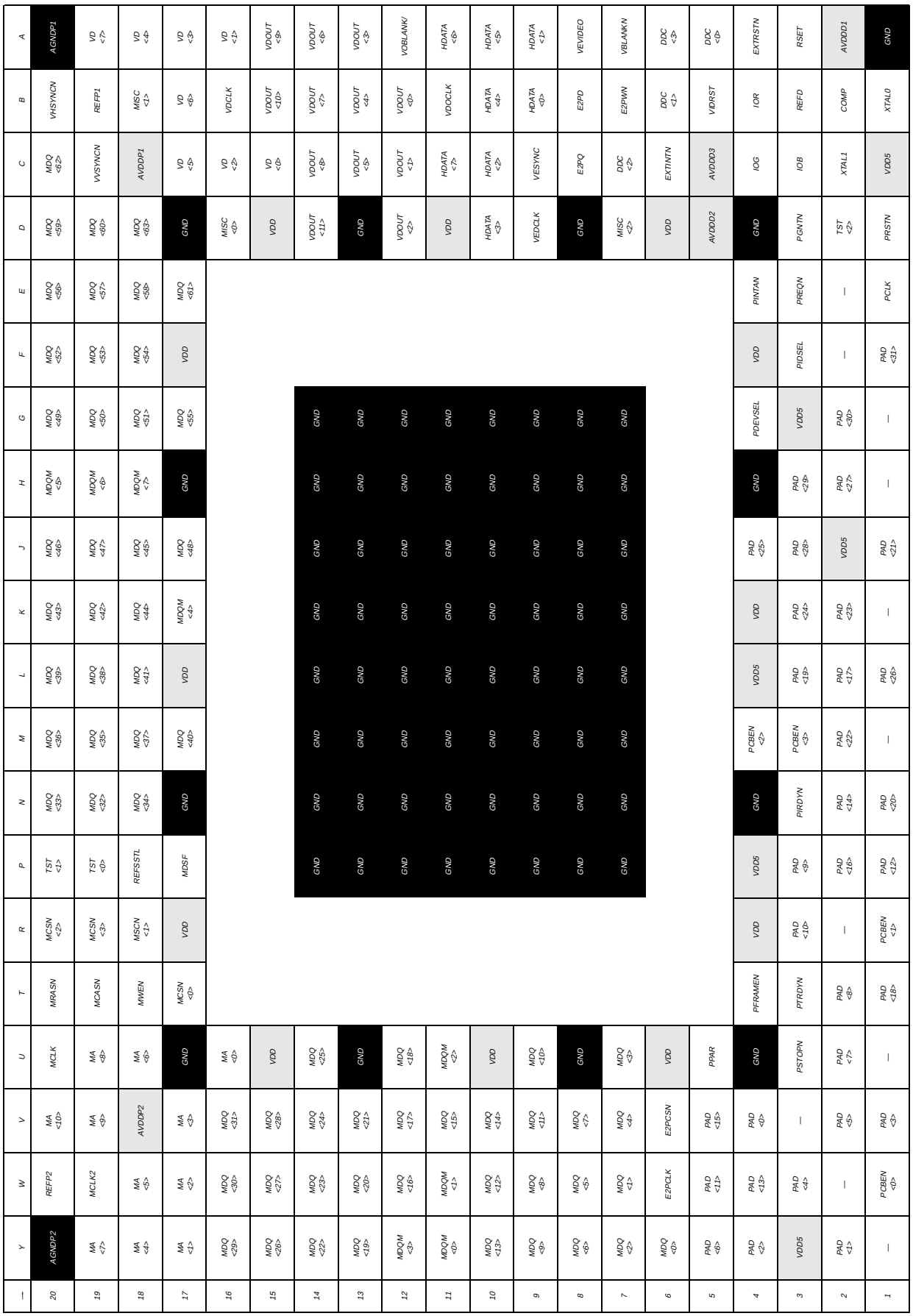


Table A-3: PCI Pinout Legend (Bottom View)



A.3 AGP Pinout Illustration and Table

The illustration below shows the locations of the MGA-G200-AGP's 320 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

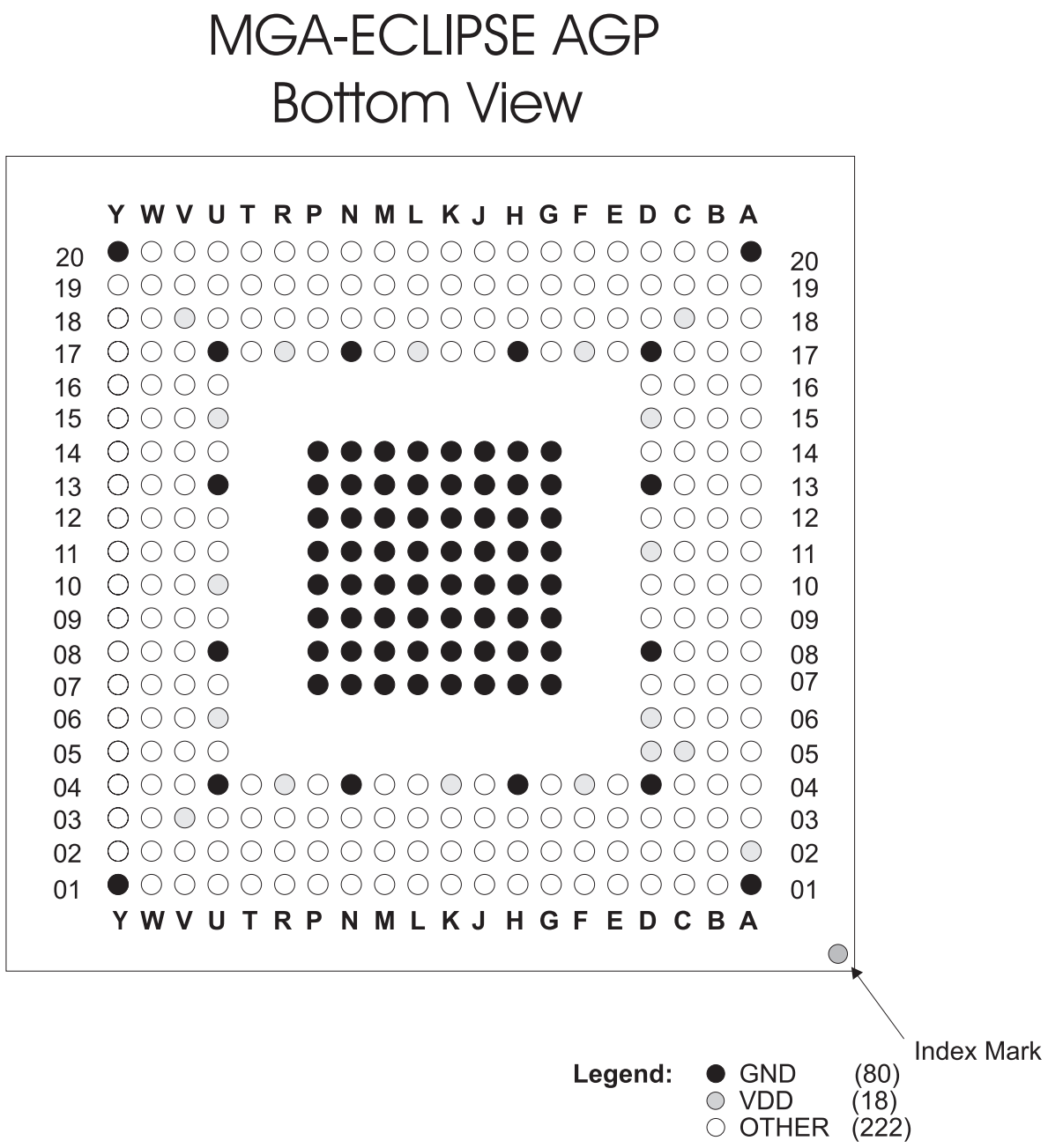
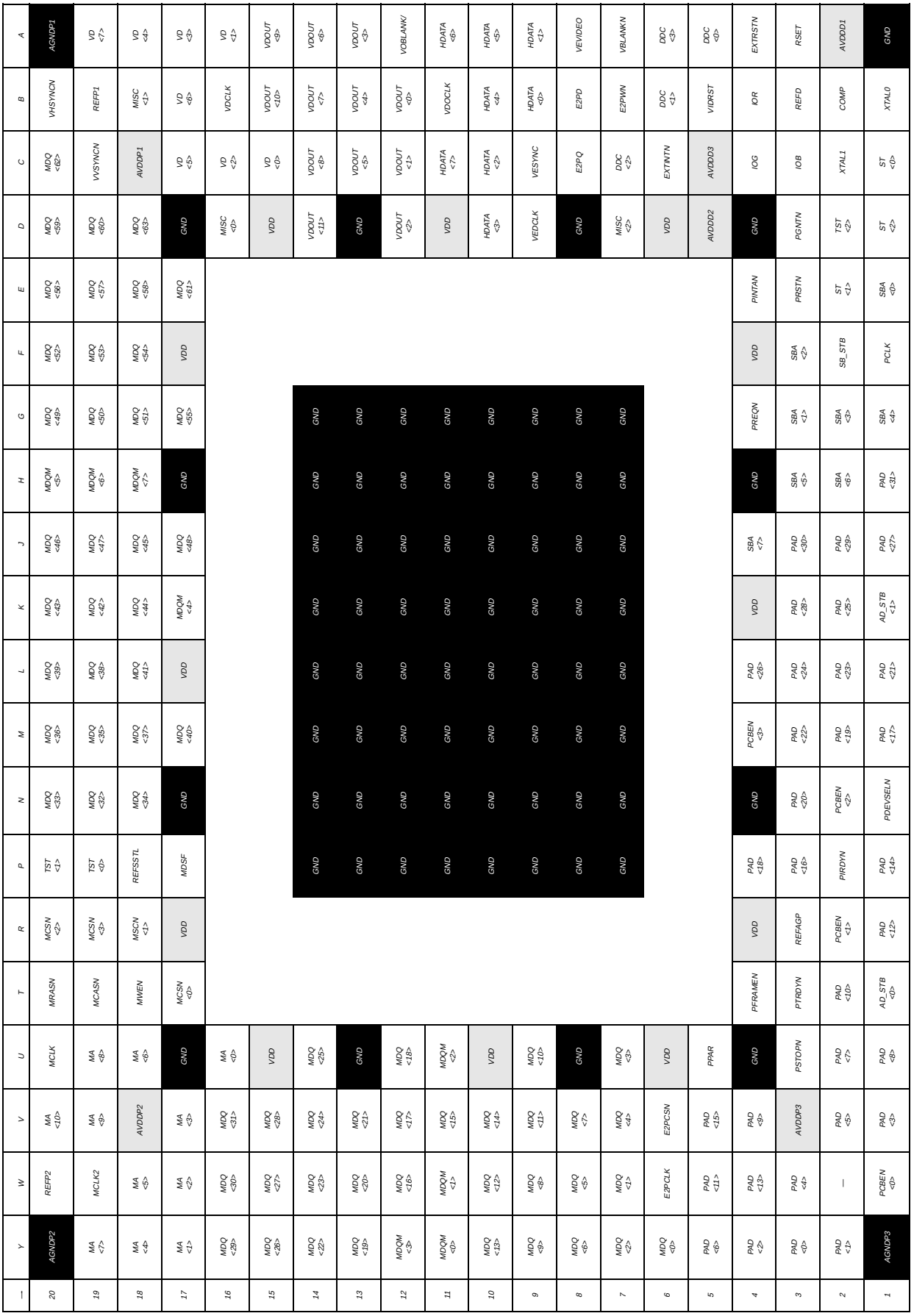


Table A-4: AGP Pinout Legend (Bottom View)



—	Y	W	V	U	T	R	P	N	M	L	K	J	H	G	F	E	D	C	B	A
20	AGNDP2	REFP2	MA <10>	MCLK	MRASN	MCSN <2>	TST <1>	MDQ <33>	MDQ <36>	MDQ <39>	MDQ <43>	MDQ <46>	MDQM <5>	MDQ <49>	MDQ <52>	MDQ <55>	MDQ <59>	MDQ <62>	VHSYNCN	AGNDP1
19	MA <7>	MCLK2	MA <9>	MA <8>	MCASN	MCSN <3>	TST <0>	MDQ <32>	MDQ <35>	MDQ <38>	MDQ <42>	MDQ <47>	MDQM <6>	MDQ <50>	MDQ <53>	MDQ <57>	MDQ <60>	VVSYNCN	REFP1	VD <7>
18	MA <4>	MA <5>	AVDDP2	MA <6>	MWEN	MSCN <1>	REFSSTL	MDQ <34>	MDQ <37>	MDQ <41>	MDQ <44>	MDQ <45>	MDQM <7>	MDQ <51>	MDQ <54>	MDQ <58>	MDQ <63>	AVDDP1	MISC <1>	VD <4>
17	MA <1>	MA <2>	MA <3>	GND	MCSN <0>	VDD	MDSF	GND	MDQ <40>	VDD	MDQM <4>	MDQ <48>	GND	MDQ <55>	VDD	MDQ <61>	GND	VD <5>	VD <6>	VD <3>
16	MDQ <29>	MDQ <30>	MDQ <31>	MA <0>													MISC <0>	VD <2>	VDCLK	VD <1>
15	MDQ <26>	MDQ <27>	MDQ <28>	VDD													VDD	VD <0>	VDOUT <10>	VDOUT <9>
14	MDQ <22>	MDQ <23>	MDQ <24>	MDQ <25>													VDOUT <11>	VDOUT <8>	VDOUT <7>	VDOUT <6>
13	MDQ <19>	MDQ <20>	MDQ <21>	GND													GND	VDOUT <5>	VDOUT <4>	VDOUT <3>
12	MDQM <3>	MDQ <16>	MDQ <17>	MDQ <18>													VDOUT <2>	VDOUT <1>	VDOUT <0>	VOBLANK/
11	MDQM <0>	MDQM <1>	MDQ <15>	MDQM <2>													VDD	HDATA <7>	VDOCLK	HDATA <6>
10	MDQ <13>	MDQ <12>	MDQ <14>	VDD													HDATA <3>	HDATA <2>	HDATA <4>	HDATA <5>
9	MDQ <9>	MDQ <8>	MDQ <11>	MDQ <10>													VEDCLK	VESYNC	HDATA <0>	HDATA <1>
8	MDQ <6>	MDQ <5>	MDQ <7>	GND													GND	E2PQ	E2PD	VEVIDEO
7	MDQ <2>	MDQ <1>	MDQ <4>	MDQ <3>													MISC <2>	DDC <2>	E2PWN	VBLANKN
6	MDQ <0>	E2PCLK	E2PCSN	VDD													VDD	EXTINTN	DDC <1>	DDC <3>
5	PAD <6>	PAD <11>	PAD <15>	PPAR													AVDDD2	AVDDD3	VIDRST	DDC <0>
4	PAD <2>	PAD <13>	PAD <9>	GND	PFRAMEN	VDD	PAD <18>	GND	PCBEN <3>	PAD <26>	VDD	SBA <7>	GND	PREQN	VDD	PINTAN	GND	IOG	IOR	EXTRSTN
3	PAD <0>	PAD <4>	AVDDP3	PSTOPN	PTRDYN	REFAGP	PIRDYN	PCBEN <2>	PAD <22>	PAD <24>	PAD <28>	PAD <30>	SBA <5>	SBA <1>	SBA <2>	PRSTN	PGNTN	IOB	REFD	RSET
2	PAD <1>	—	PAD <5>	PAD <7>	PAD <10>	PCBEN <1>	PIRDYN	PCBEN <2>	PAD <19>	PAD <23>	PAD <25>	PAD <29>	SBA <6>	SBA <3>	SB_STB	ST <1>	ST <2>	XTAL1	COMP	AVDDD1
1	AGNDP3	PCBEN <0>	PAD <3>	PAD <8>	AD_STB <0>	PAD <12>	PAD <14>	PDEVSELN	PAD <17>	PAD <21>	AD_STB <1>	PAD <27>	PAD <31>	SBA <4>	PCLK	SBA <0>	ST <0>	ST <0>	XTAL0	GND

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-3, IO-6, IO-9, IO-12, I-0, I-S	V_I < VDD3 + 0.5V	-0.5	4.6	V	
	IO-6-5V, IO-9-5V	V_I < VDD3 + 3.0V	-0.5	6.6	V	
	I-PCI 33, IO-PCI 33	V_I < VDD5 + 0.5V	-0.5	6.6	V	(1)
	IO_SSTL1, IO_SSTL2	V_I < VDD3 + 0.3V	-0.3	4.6	V	
V_O	Output Voltage					(2)
	IO-3, IO-6, IO-9, IO-12	V_O < VDD3 + 0.5V	-0.5	4.6	V	
	IO-6-5V, IO-9-5V	V_O < VDD3 + 3.0V	-0.5	6.6	V	
	IO-PCI 33	V_O < VDD5 + 0.5V	-0.5	6.6	V	(1)
	IO_SSTL1, IO_SSTL2	V_O < VDD3 + 0.3V	-0.3	4.6	V	
I_O	Output Current					(2)
	IO-3			10	mA	
	IO-6			20	mA	
	IO-9			30	mA	
	IO-12			40	mA	
	IO-PCI33			?	mA	
	IO_SSTL1		-8	8	mA	
	IO_SSTL2		-16	16	mA	
T_A	Operating Temperature		0	55	^oC	
T_{STG}	Storage Temperature		-65	150	^oC	

(1) MGA-G200-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 (PCI)		3.0	3.6	V
VDD3 (AGP)		3.15	3.45	V
REFAGP	AGP buffer voltage reference (V_{ref})	0.39 VDD3	0.41 VDD3	V
	Vref pin input current (I_{ref})	+10		µA
V_{IH}	High-Level Input Voltage			
	IO-AGP, I-AGP	V_{ref} +0.2	VDD3	V
	IO-12, I-0	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	0	0.8	V
	IO-AGP, I-AGP	0	V_{ref} -0.2	
	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°)(PCI)
(VDD3 = 3.3 ±0.15V, VDD5 = 5.0 ±0.25V, TA = 0 to 55°)(AGP)

Symbol	*Parameter*	*Conditions*	*Min.*	*Max.*	*Units*	*Notes*
I_{OS}	Output Short-Circuit Current	V_O = 0V		-250	mA	(1)
I_i	Input Leakage Current	V_i = VDD3 or 0V		±10	µA	
I_{OL}	IO-AGP, O-AGP	V_{OL} = 0.18VDD3				(2)
	Low-Level Output Current	V_{OL} = 0.4V				
	O-3		3		mA	
	O-6		6		mA	
	O-9		9		mA	
	O-12, IO-12		12		mA	
	O-24		24		mA	
	IO-PCI33, O-PCI33				mA	(2)
	IO-SSTL1		8		mA	
	IO-SSTL2		16		mA	
I_{OH}	IO-AGP, O-AGP	V_{OH} = 0.7 VDD3				(2)
	High-Level Output Current	V_{OH} = 2.4V				
	O-3		-3		mA	
	O-6		-6		mA	
	O-12, IO-12		-12		mA	
	O-24		-24		mA	
	IO-PCI33, O-PCI33				mA	(2)
	IO-SSTL1		-8		mA	
	IO-SSTL2		-16		mA	
V_{OL}	IO-AGP, O-AGP	I_{out}=1500µA		0.1 VDD3	V	
	Low-Level Output Voltage	I_{OL} = 0 mA		0.1	V	
	IO-SSTL			VTT-0.6	V	
	IO-SSTL2			VTT-0.8	V	
V_{OH}	IO-AGP, O-AGP	I_{out}=1500µA	0.9 VDD3		V	
	High-Level Output Voltage	I_{OH} = 0 mA	VDD3 - 0.1		V	
	IO-SSTL1		VTT+0.6		V	
	IO-SSTL2		VTT+0.8		V	
θ_{JA}	Junction-to-Air Thermal Coefficient	No Air Flow		?	°c/w	(3)
C_{PIN}	Pin Capacitance	F = 1 MHz		7	pF	
ICC3	VDD3 Supply Current		?		mA	
ICC5	VDD5 Supply Current		?		mA	

(1) The Output Short-Circuit time is less than one second for one pin only.

(2) AGP and PCI buffers are characterized by their V/I curves (see the following figures).

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

Figure A-3: SSTL Buffer I/V Curve Pull-Up (Class 1)

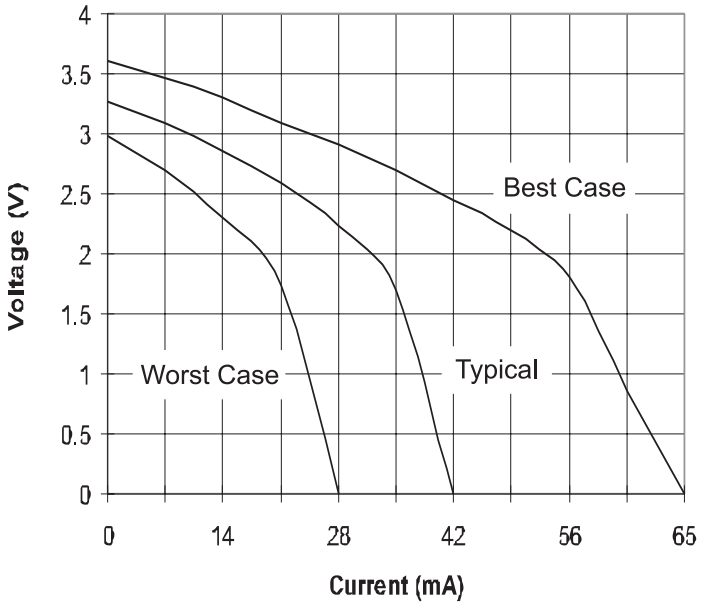


Figure A-4: SSTL Buffer I/V Curve Pull-Up (Class 2)

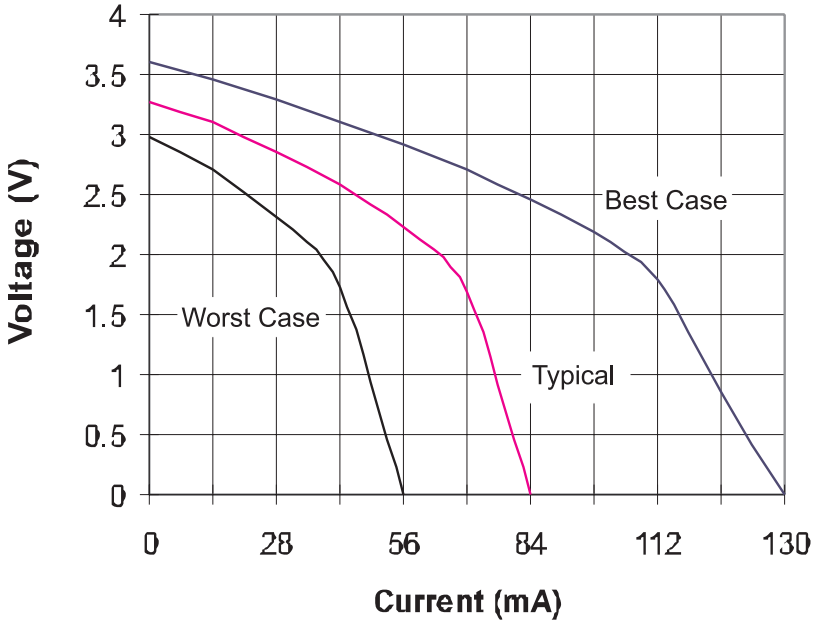


Figure A-5: SSTL Buffer I/V Curve Pull-Down (Class 1)

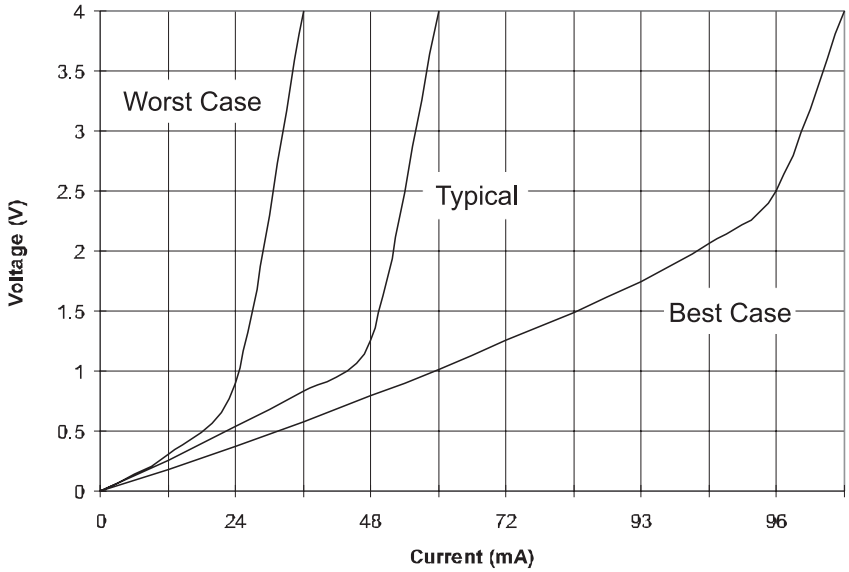


Figure A-6: SSTL Buffer I/V Curve Pull-Down (Class 2)

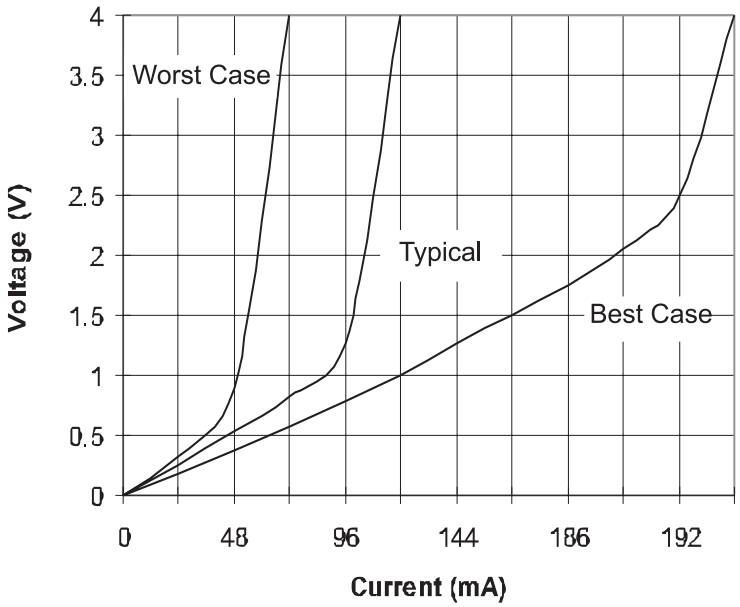


Figure A-7: V/I Curves for AGP Buffers (Best Case)

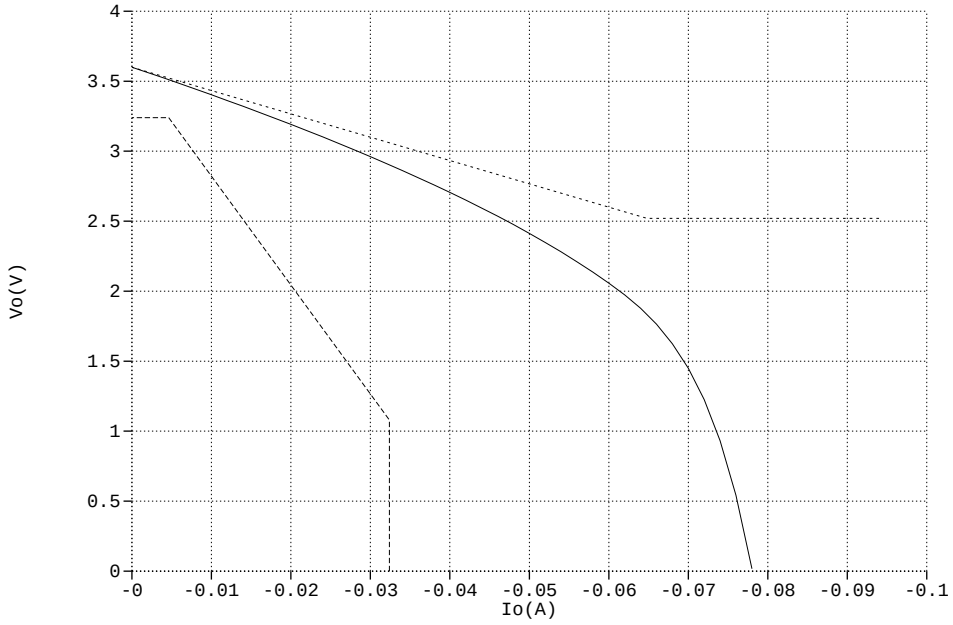


Figure A-8: V/I Curves for AGP Buffers (Worst Case)

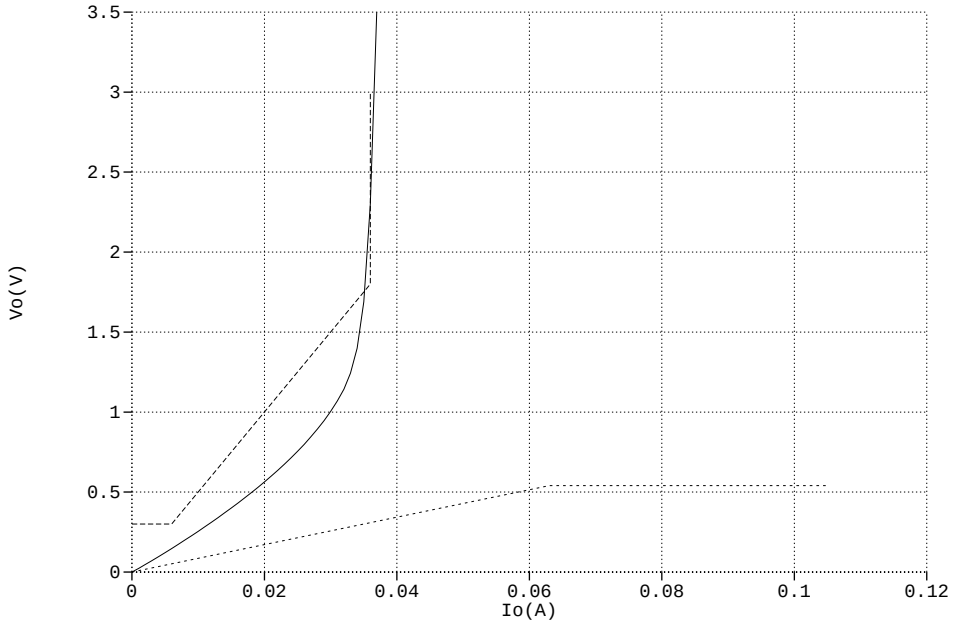


Figure A-9: VV/I Curves for AGP Buffers (Typical Case)

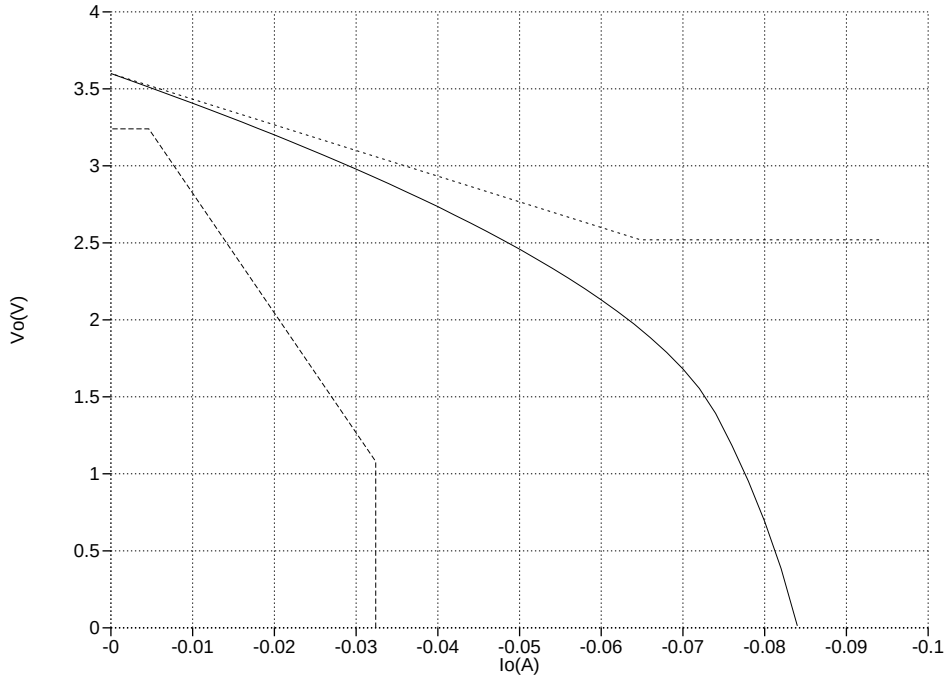


Table A-8: PCI Buffer Type and Pin Load (Part 1 of 2)

Signal Name	Buffer Type	Load (pF)		Damping (ohms)	Notes
		Min.	Max.		
PAD<31:0>	IO_PCI33	0	50	-	(1)
PCBE<3:0>/	IO_PCI33	0	50	-	(1)
PCLK	I_PCI33	-	-	-	(1)
PDEVSEL/	IO_PCI33	0	50	-	(1)
PFRAME/	IO_PCI33	0	50	-	(1)
PGNT/	I_PCI33	-	-	-	(1)
PIDSEL	I_PCI33	-	-	-	(1)
PINTA/	O_PCI33	0	50	-	(1)
PIRDY/	IO_PCI33	0	50	-	(1)
PPAR	IO_PCI33	0	50	-	(1)(2)
PREQ/	IO_PCI33	0	50	-	(2)
PRST/	I_PCI33	-	-	-	(1)
PSTOP/	IO_PCI33	0	50	-	(1)
PTRDY/	IO_PCI33	0	50	-	(1)
EXTINT/	I_O	-	-	-	
EXTRST/	IO_6	50	50	-	(2)
E2PCLK	IO_3	12	16	-	(2)
E2PD	IO_9	16	96	-	(2)
E2PQ	I_O_5V	-	-	-	
E2PW/	IO_6	16	80	-	(2)
E2PCS/	IO_3	14	18	-	(2)
MCS<3:0>/	IO_SSTL1	15	65		
MA<10:0>	IO_SSTL2	15	65	-	(2)
MCAS/	IO_SSTL2	15	65	-	(2)
MCLK	IO_SSTL2	12	25	-	
MCLK2	IO_SSTL2	12	25	-	
MCS<3:0>/	IO_SSTL1	10	40	-	(2)
MDQ<63:0>	IO_SSTL1	10	40	-	
MDQM<7:0>	IO_SSTL1	15	65	-	(2)
MDSF	IO_SSTL2	15	65	-	(2)
MRAS/	IO_SSTL2	15	65	-	(2)
MWE/	IO_SSTL2	15	65	-	(2)
VBLANK/	IO_6	15	35	33	(2)
VD<7:0>	I_O_5V	-	-	-	
VDCLK	I_O_5V	-	-	-	
VDOCLK	IO_9	10	35	-	
VDOUT<11:0>	IO_6	10	35	-	(2)
VEDCLK	IO_6	16	80	-	(2)
VESYNC	IO_6	16	80	-	(2)
VEVIDEO	IO_6	16	80	-	(2)
HDATA<7:0>	IO_6_5V	26	90		

Table A-8: PCI Buffer Type and Pin Load (Part 2 of 2)

Signal Name	Buffer Type	Load (pF)		Damping (ohms)	Notes
		Min.	Max.		
VHSYNC/	IO_6	10	20	33	(2)
VIDRST	I_O	-	-	-	
VOBLANK/	IO_9	10	35	-	(2)
VVSYNC/	IO_6	10	20	33	(2)
DDC<0>	IO_9_5V	50	50	-	
DDC<3:1>	IO_6_5V	50	50	-	
MISC<2:0>	IO_6_5V	16pF<2:0>	100	-	
TST<1:0>	I_O	-	-	-	
RBFN_LFT	I_O	-	-	-	

(1) See Table A-9

(2) input mode only when TST<2:0> = '000' (HiZ mode for NAND Tree test)

Table A-9: AGP Buffer Type and Pin Load

Signal Name	Buffer Type	Load (pF)		Damping (ohms)	Notes
		Min.	Max.		
PAD<31:0>	IO_AGP	0	10	-	
PCBE<3:0>/	IO_AGP	0	10	-	
PCLK	I_AGP	-	-	-	
PDEVSEL/	IO_AGP	0	10	-	
PFRAME/	IO_AGP	0	10	-	
PGNT/	I_AGP	-	-	-	
PINTA/	O_AGP	0	10	-	
PIRDY/	IO_AGP	0	10	-	
PPAR	IO_AGP	0	10	-	(1)
PREQ/	IO_AGP	0	10	-	(1)
PRST/	I_AGP	-	-	-	
PSTOP/	IO_AGP	0	10	-	
PTRDY/	IO_AGP	0	10	-	
SBA<7:0>	IO_AGP	0	10	-	(1)
ST<2:0>	I_AGP	-	-	-	
SB_STB	IO_AGP	0	10	-	(1)
AD_STB<1:0>	IO_AGP	0	10	-	

(1) Input mode only when TST<2:0. = '000' (HiZ mode for NAND Tree test)

A.4.2 AC Specifications

The following tables indicate the timing parameters a device (SGRAM, BIOS ROM, or other external device) must meet to work properly with the MGA-G200 chip.

A.4.2.1 DAC AC Parameters

Table A-10: DAC Parameter List

Parameter	*Min.*	*Typ.*	*Max.*	*Units*
Analog output delay (Td) (1)	1.5	-	4.4	ns
Analog output settling time (Ts) (2)	4.2	-	5.5	ns
Analog output rise/fall time (Tr/f) (3)	1.8	-	2.3	ns
Glitch impulse	-	100	-	pV/s
DAC to DAC crosstalk	-	-	-	-
Analog output skew	-	-	-	ns

(1) Measured from the 90% point of the rising clock edge to 50% of full-scale transition.

(2) Measured from 50% of full-scale transition to settled output, within +/- 1 LSB.

(3) Measured between 10% and 90% of full-scale transition.

Table A-11: PLL Parameter List

Parameter	*Min.*	*Typ.*	*Max.*	*Units*
Pixel clock PLL frequency	6.25	-	-	MHz
PLL duty cycle variation	45	-	55	%
System PLL frequency	6.25	-	-	MHz
Frequency select to PLL output stable	-	1.00	-	ms
PLL phase jitter	-	-	-	-
PLL duty cycle jitter	-	-	-	-

A.4.2.2 Host Interface Timing

Figure A-10: PCI 33 MHz Waveform (MGA-G200-PCI only)

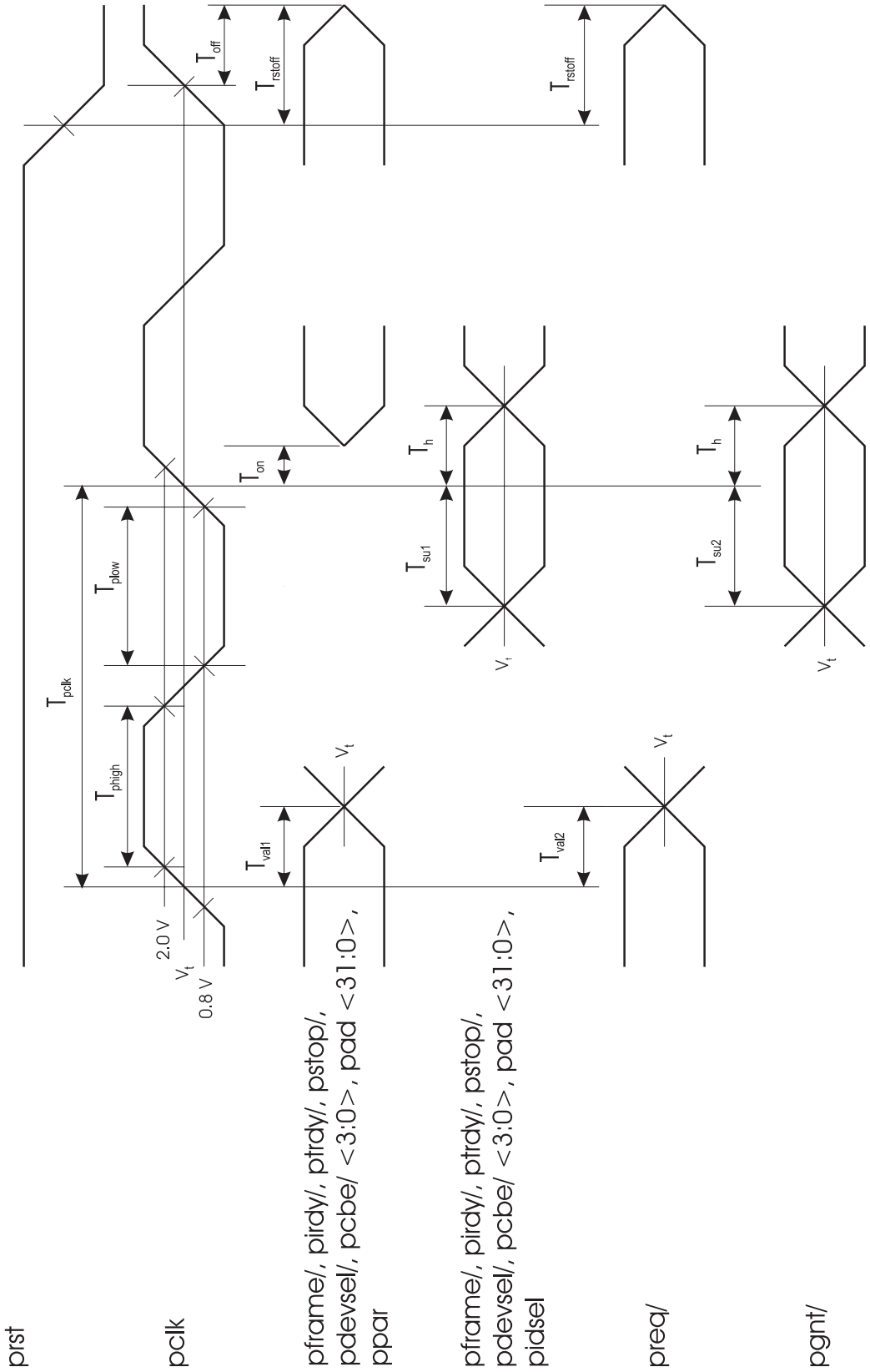


Table A-12: PCI 33 MHz 5V Signaling Environment Timing (MGA-G200-PCI only)(1)

Symbol	*Parameter*	*Min*	*Max*	*Unit*	*Notes*
T_{pclk}	PCLK cycle time	30	—	ns	—
T_{plow}	PCLK low time	11	—	ns	—
T_{phigh}	PCLK high time	11	—	ns	—
T_{on}	Float to active delay	2	—	ns	—
T_{val1}	PCLK to signal valid delay	2	11	ns	(2),(3)
T_{val2}	PCLK to signal valid delay	2	12	ns	(3),(4)
T_{off}	Active to float delay		28	ns	(5)
T_{rstoff}	Reset active to output float delay		40	ns	(5)
T_{su1}	Input setup time to PCLK	7	—	ns	(6)
T_{su2}	Input setup time to PCLK	10	—	ns	(7)
T_h	Input hold time from PCLK	0	—	ns	—

(1) $V_t = 1.5V$

(2) Applies only to pframe/, pridy/, ptrdy/, pstop/, pdevsel/, pcbe/ <3:4>, pad <31:0>, ppar

(3) Minimum times are evaluated with 0 pF lumped load.
Maximum times are evaluated with 50 pF lumped load.

(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-11: AGP 1X Timing (MGA-G200-AGP only)

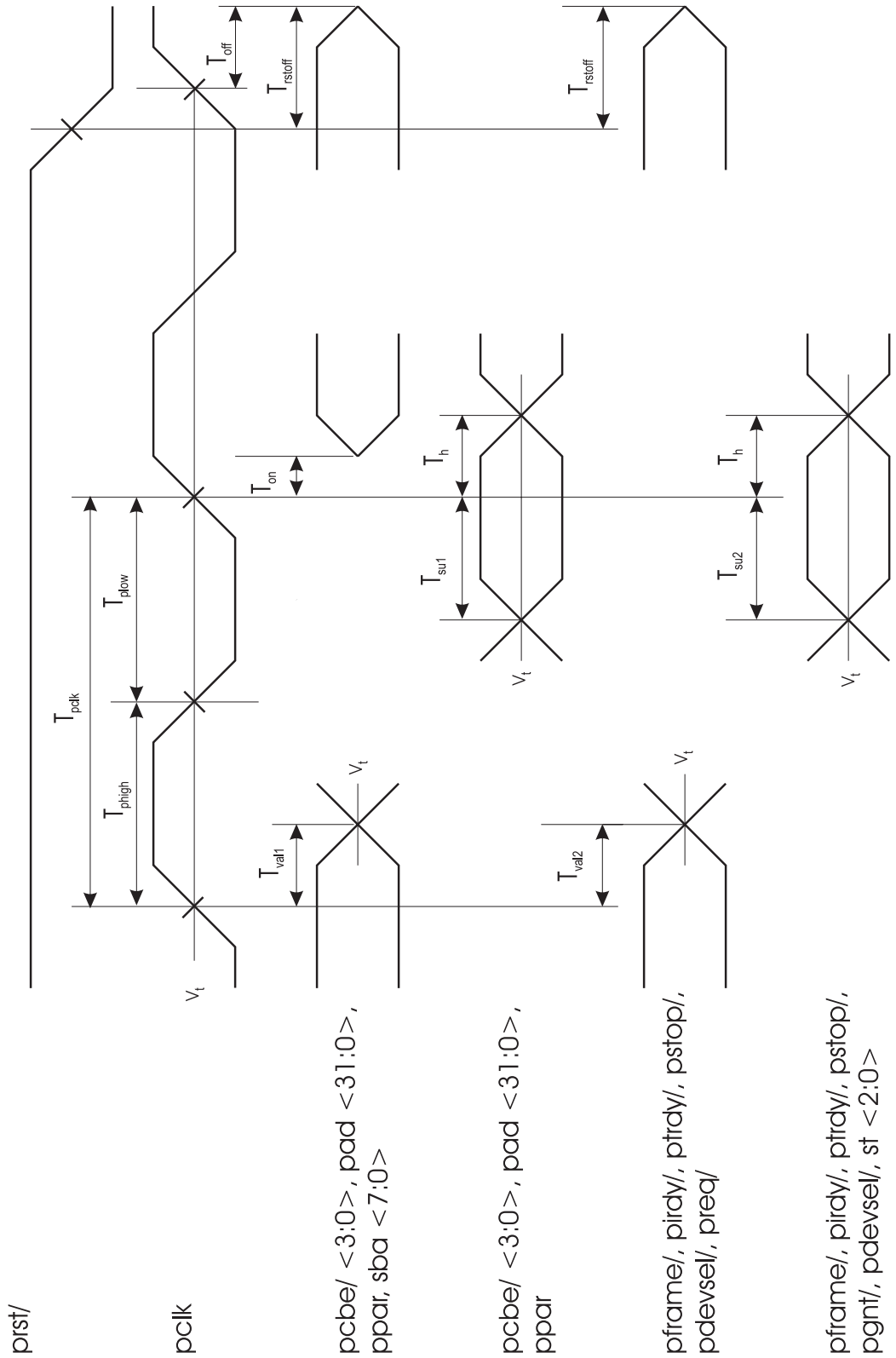


Table A-13: AGP 1X Timing (MGA-G200-AGP only)(1)

Symbol	*Parameter*	*Min*	*Max*	*Unit*	*Notes*
T_{pclk}	PCLK cycle time	15.0	30	ns	—
T_{plow}	PCLK low time	6.0	—	ns	—
T_{phigh}	PCLK high time	6.0	—	ns	—
T_{on}	Float to active delay	1.0	6.0	ns	—
T_{val1}	PCLK to signal valid delay (Data)	1.0	6.0	ns	(2)
T_{val2}	PCLK to signal valid delay (Control)	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 (Data)	5.5	—	ns	(5)
T_{su2}	Input setup time to PCLK (Control)	6.0	—	ns	(6)
T_h	Input setup time from PCLK	0	—	ns	—

(1) Timings are evaluated with a 10pF lumped load.
Vt = 0.4 V_{DD}

(2) Applies only to pframe/, pirdy/, ptrdy/, pstop/, pdevsel/, preq/, and ppar.

(3) Applies only to pad<31:0> and pcbe/<3:0>, SBA<7:0> and ppar.

(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 pad<31:0>, pcbe/<3:0> and ppar.

(6) Applies only to pframe/, pirdy/, ptrdy/, pstop/, pdevsel/, pgnt/ and st<2:0>.

Figure A-12: AGP 2X Timing Diagram

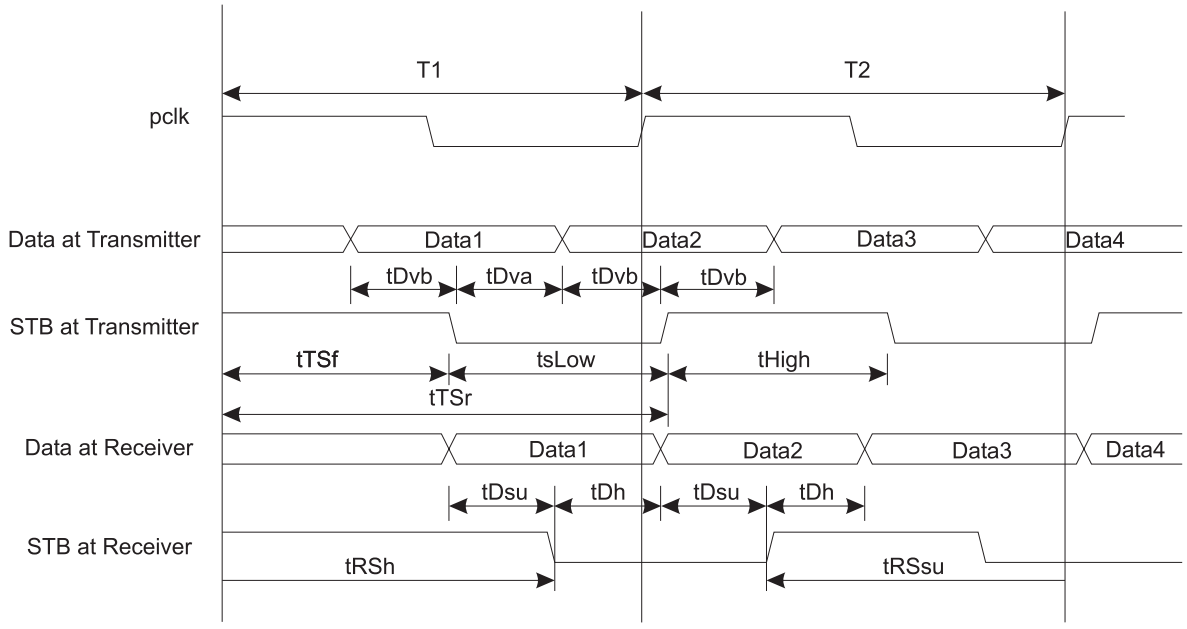


Figure A-13: AGP 2X Strobe/Data Turnaround Timing Diagram

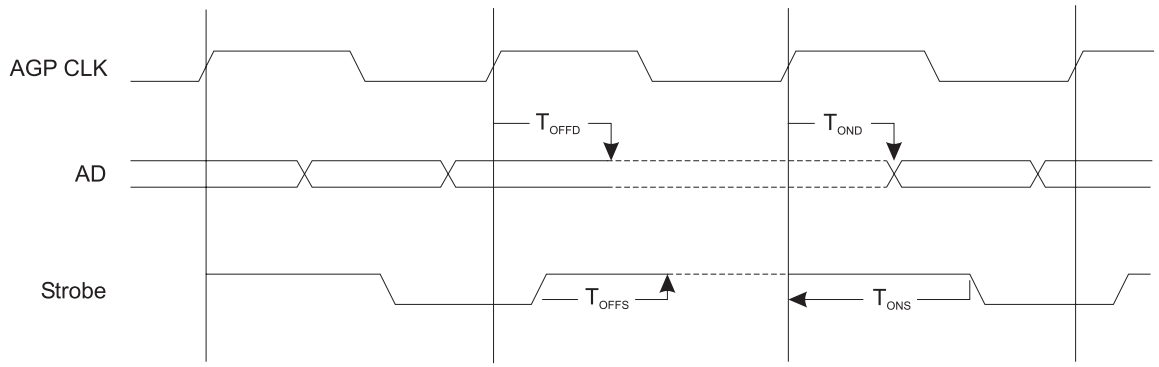


Table A-14: AGP 2X Timing (MGA-G200-AGP only)(1)

Symbol	*Parameter*	*Min*	*Max*	*Unit*	*Notes*
Transmitter Output Signals:					
T_{TSf}	CLK to transmit strobe falling	2	12	ns	—
T_{TSr}	CLK to transmit strobe rising		20	ns	—
T_{Dvb}	Data valid before strobe	1.7	—	ns	—
T_{Dva}	Data valid after strobe	1.9	—	ns	—
T_{ONd}	Float to Active Delay	-1	9	ns	—
T_{OFFd}	Active to Float Delay	1	12	ns	—
T_{ONS}	Strobe active to strobe falling edge setup	6	10	ns	—
T_{OFFS}	Strobe rising edge to strobe float delay	6	10	ns	—
Receiver Output Signals:					
T_{RSsu}	Receive strobe setup tome to CLK	6	—	ns	—
T_{RSh}	Receive strobe hold time from CLK	1	—	—	—
T_{Dsu}	Data strobe setup time	1	—	ns	—
T_{Dh}	Strobe to data hold time	1	—	ns	—
Ts_{low}	Strobe minimum low and high time	5.0	—	ns	—
Ts_{high}		5.0	—	ns	—
$Tpll_{lock}$	AGP 2X pll lock time		2.5	ns	—

(1) Timings are evaluated with a 10pF lumped load.
Vt = 0.4 V_{DD}

A.4.2.3 Serial EEPROM Device Timing

Figure A-14: Serial EEPROM Waveform

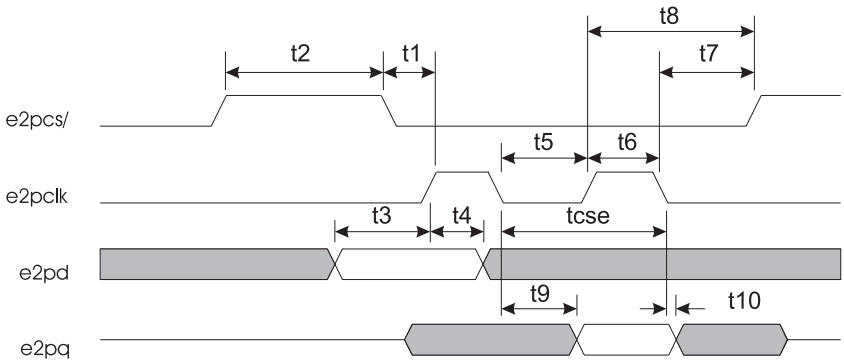


Table A-15: Serial EEPROM Clock Period Cycle

biosboot	*serial eeprom size*	*MGA-G200*	*tcp min (ns)*	*tcse (ns)*	*tcse*	
					min (ns)	*max (MHz)*
0	128 bytes	AGP	15	tcp * 64	960	1.04
		PCI	30	tcp * 32		
1	32KB or 64KB	AGP	15	tcp * 16	240	4.17
		PCI	30	tcp * 8		

➧ ***Note:*** tcp is the PCI clock cycle period.

➧ ***Note:*** tcse is the serial EEPROM clock cycle period.

Table A-16: Serial EEPROM Parameter List

Name	*Min. (ns)*	*Max. (ns)*	*Commnet*
t1	tcse/2-tcp-3		Chip select falling to clock rising
t2	tcse-1		Chip select high time
t3	tcse/2-9		Input data setup time to clock rising
t4	tcse/2-2		Input data hold time from clock rising
t5	tcse/2		Clock low time
t6	tcse/2-1		Clock high time
t7	tcp-3		Clock falling time to chip select rising
t8	tcse/2+tcp-3		Clock rising to chip select rising
t9		tcse-19	Clock falling to valid output data
t10	3		Output data hold from clock falling

A.4.2.4 MAFC Feature Connector

Figure A-15: MAFC Waveform

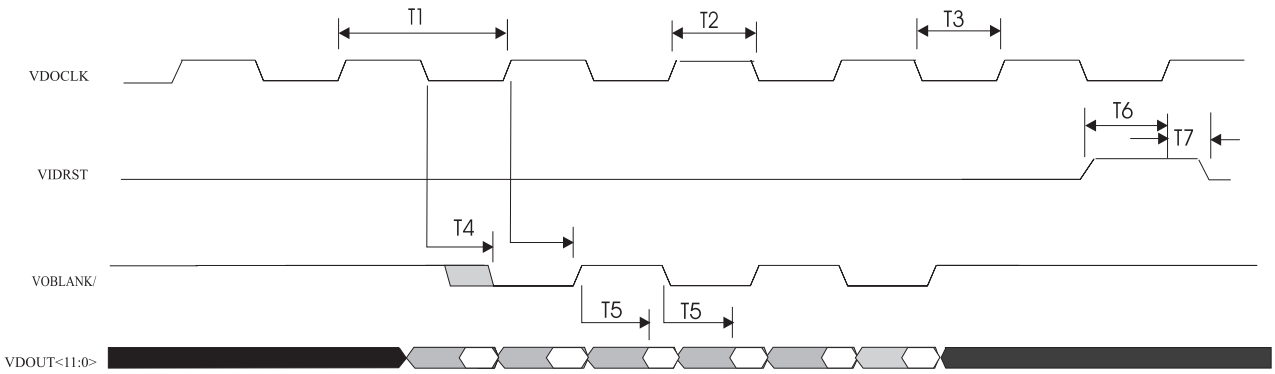


Figure A-16: Panel Link Mode

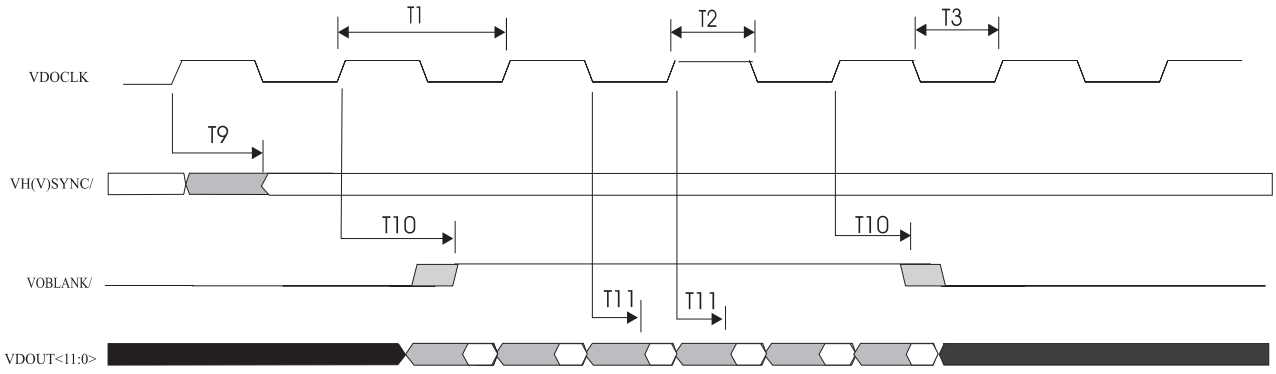


Figure A-17: Bypass Mode

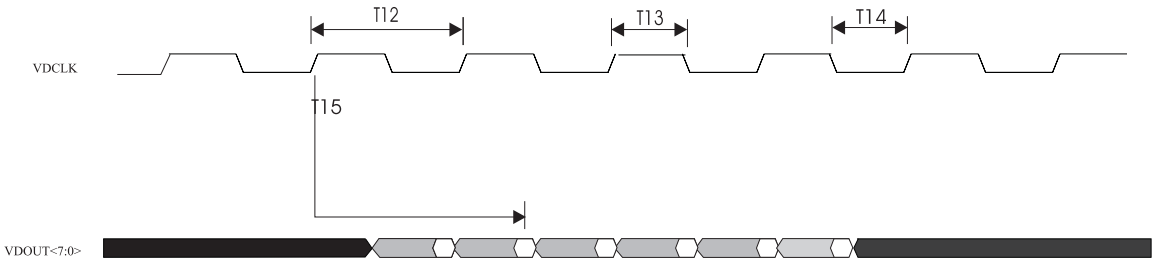


Table A-17: MAFC Waveforms data information

Name	*Min. (ns)*	*Max. (ns)*	*Comment*	
T1	15.4	-	VDOCLK period Mode A	
T2	6.7	-	VDOCLK high Mode A	
T3	6.7	-	VDOCLK low Mode A	
T5	- 0.4	4.5	Valid data time (VOBLANK/ => VDOUT)	MAFCLK
T6	6	-	Setup time (VIDRST => VDOCLK)	
T7	0	-	Hold time (VIDRST => VDOCLK)	
T9	0.3	5.7	VDOCLK => VH(V)SYNC	Panel Link
T10	0.3	5.7	VDOCLK => VOBLANK/	Panel Link
T11	0.3	5.7	Valid data time (VDOCLK => VDOUT)	Panel Link
T12	35	-	VDCLK period	Bypass mode
T13	13.5	-	VDCLK high	Bypass mode
T14	13.5	-	VDCLK low	Bypass mode
T15	6	20	VDCLK -> VDOUT<7:0>	Bypass mode

A.4.2.5 Memory Interface Timing

Figure A-18: Memory Interface Waveform

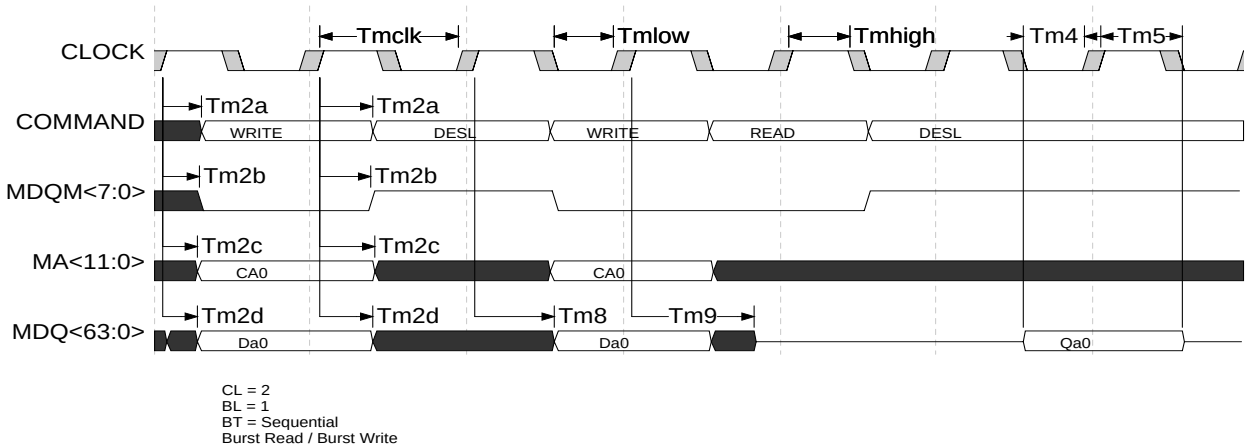


Table A-18: Memory Interface Parameter List

Timing	*Description*	*Min. (ns)*	*Max. (ns)*
Tmclk	MCLK period	10.0	-
Tmlow	MCLK low	3.0	-
Tmhigh	MCLK high	3.0	-
Tm2a	MCLK --> Command (MRAS/, MCAS/, MWE/, MDSF, MCS/<3:0>) valid	1.86	3.64
Tm2b	MCLK --> MDQM<7:0> valid	1.86	3.64
Tm2c	MCLK --> MA<11:0> valid	1.86	3.64
Tm2d	MCLK --> MDQ<63:0> valid	1.86	3.64
Tm4	MDQ<63:0> setup --> MCLK	0.64	-
Tm5	MDQ<63:0> hold --> MCLK	2.36	-
Tm8	MCLK --> MDQ<63:0> low-z	1.86	-
Tm9	MCLK --> MDQ<63:0> high-z	-	3.64

Figure A-19: Read Followed by Precharge (Tcl=3)

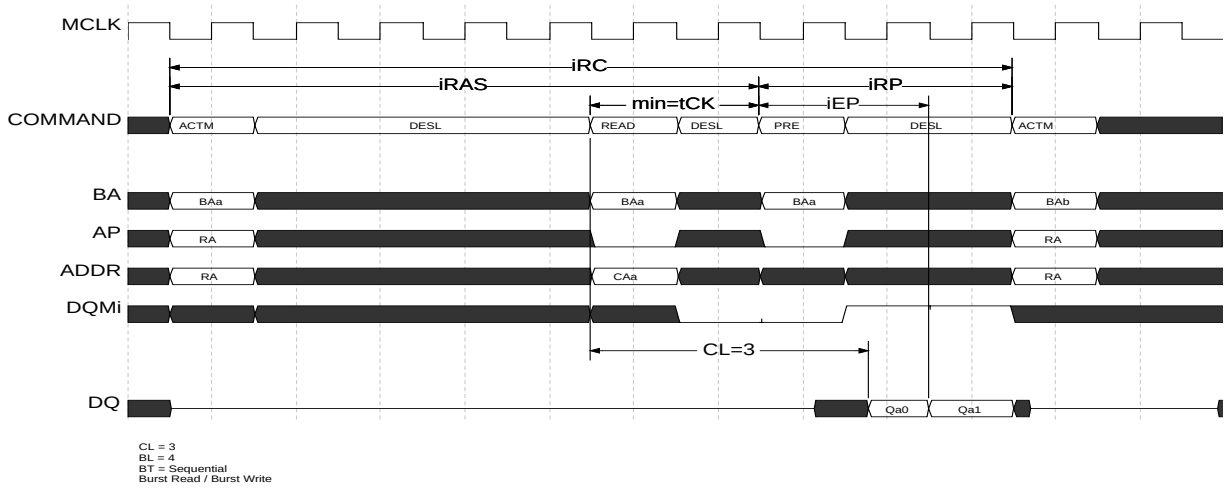
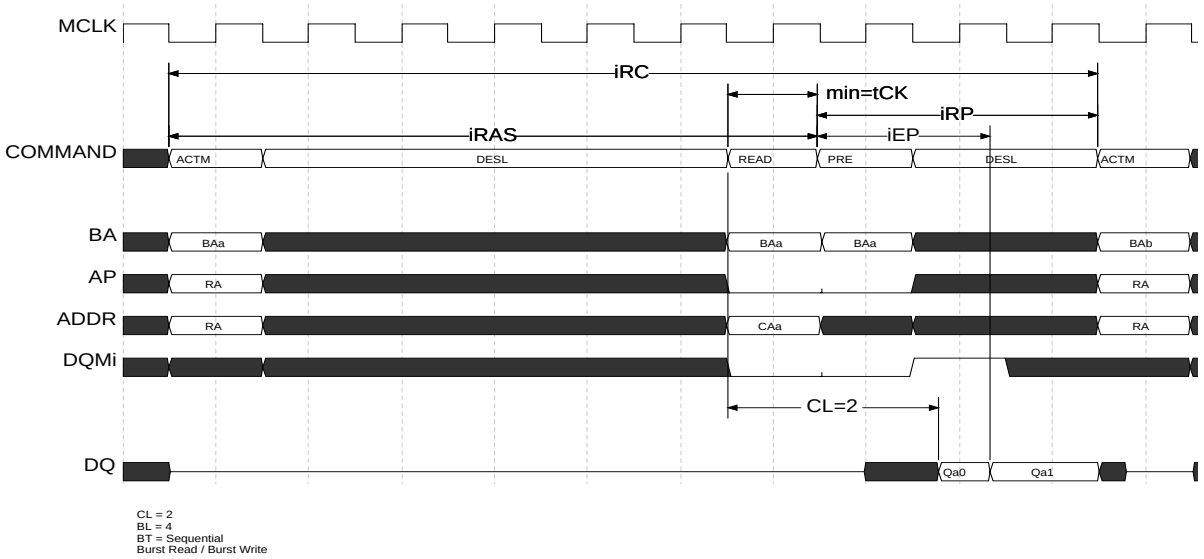
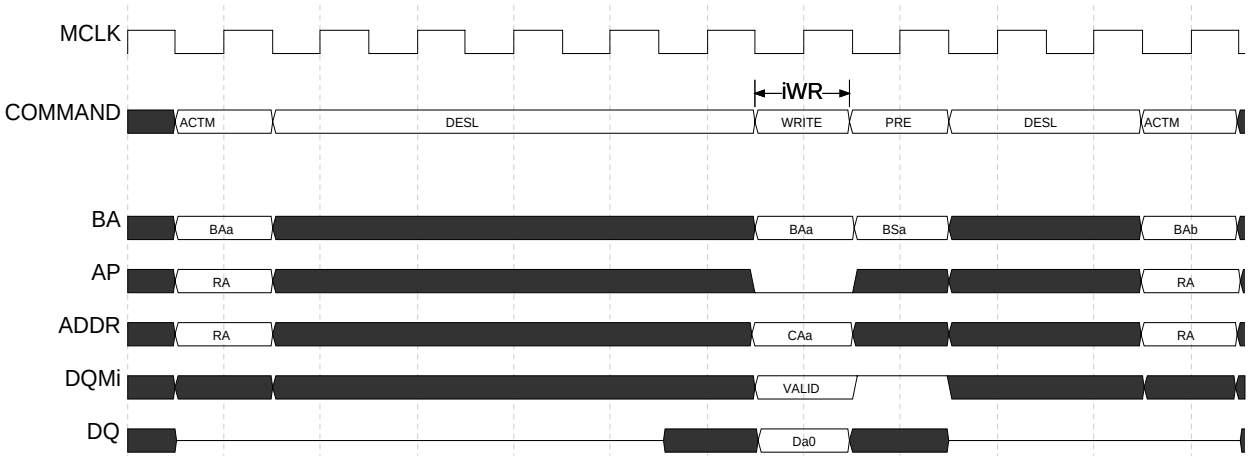


Figure A-20: Read Followed by a Precharge (Tcl = 2)



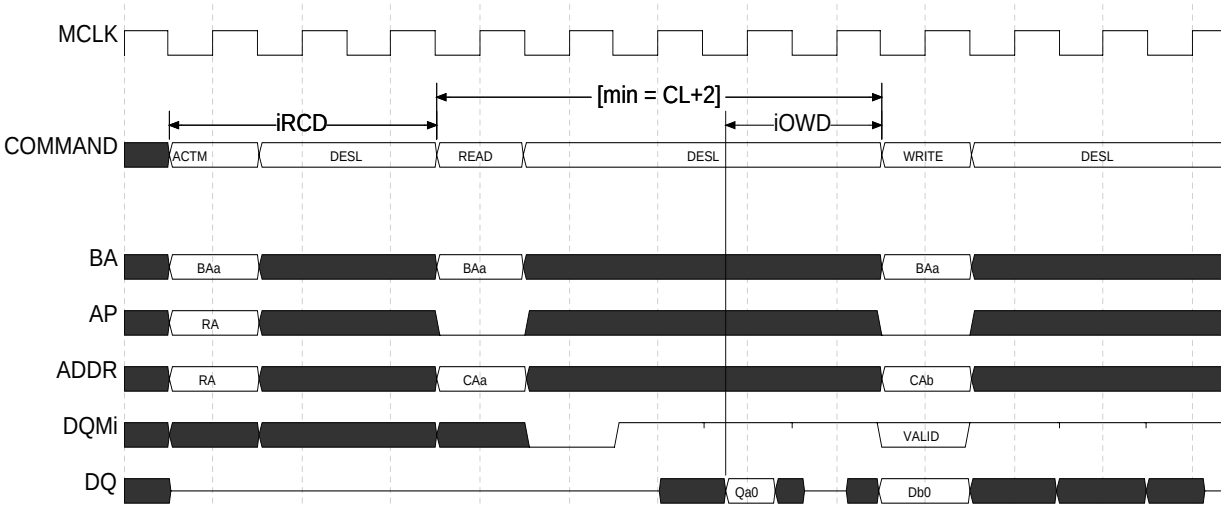
CL = 2
BL = 4
BT = Sequential
Burst Read / Burst Write

Figure A-21: Write Followed by Precharge



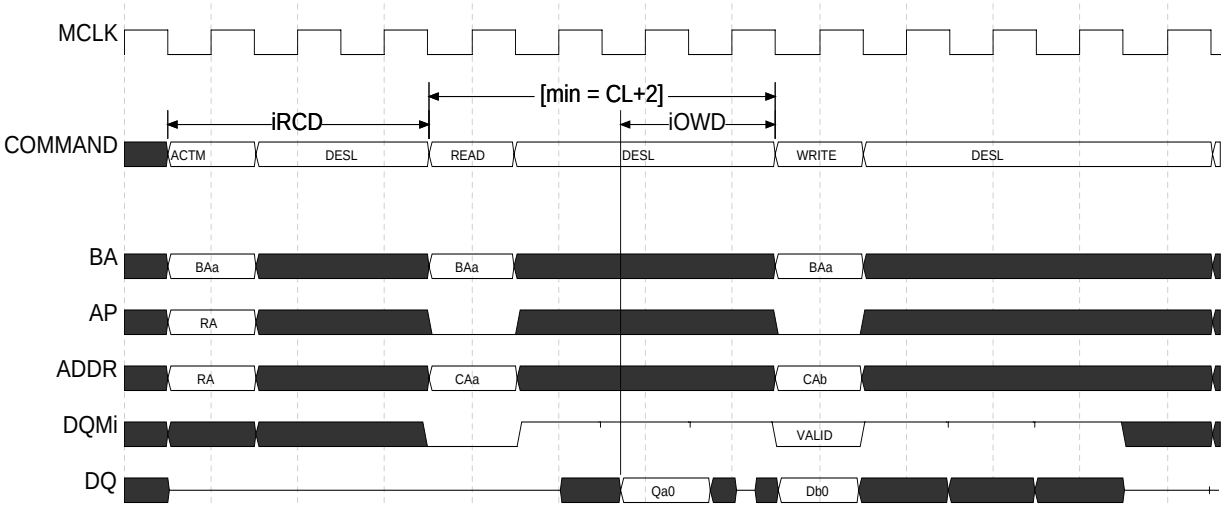
BL = 4
BT = Sequential
Burst Read / Burst Write

Figure A-22: Read Followed by Write (Tcl =3)



CL = 3
BL = 4
BT = Sequential
Burst Read / Burst Write

Figure A-23: Read Followed by Write (Tcl =2)



CL = 2
BL = 4
BT = Sequential
Burst Read / Burst Write

Figure A-24: Write Followed by Read (same chip select)

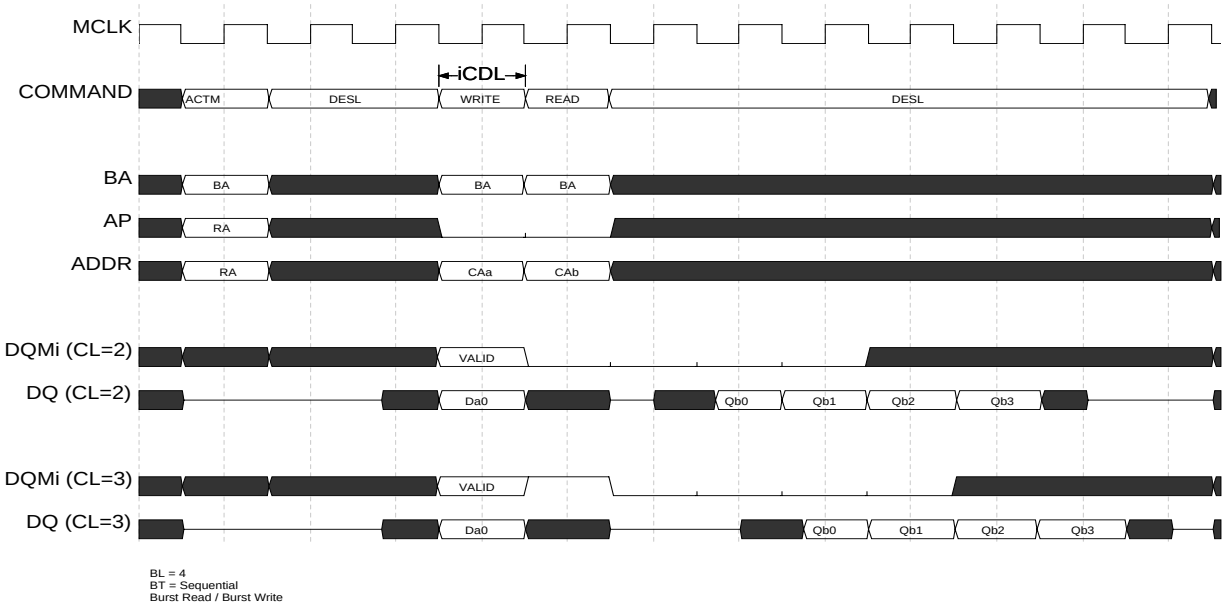


Figure A-25: Write Followed by Read (different chip select)

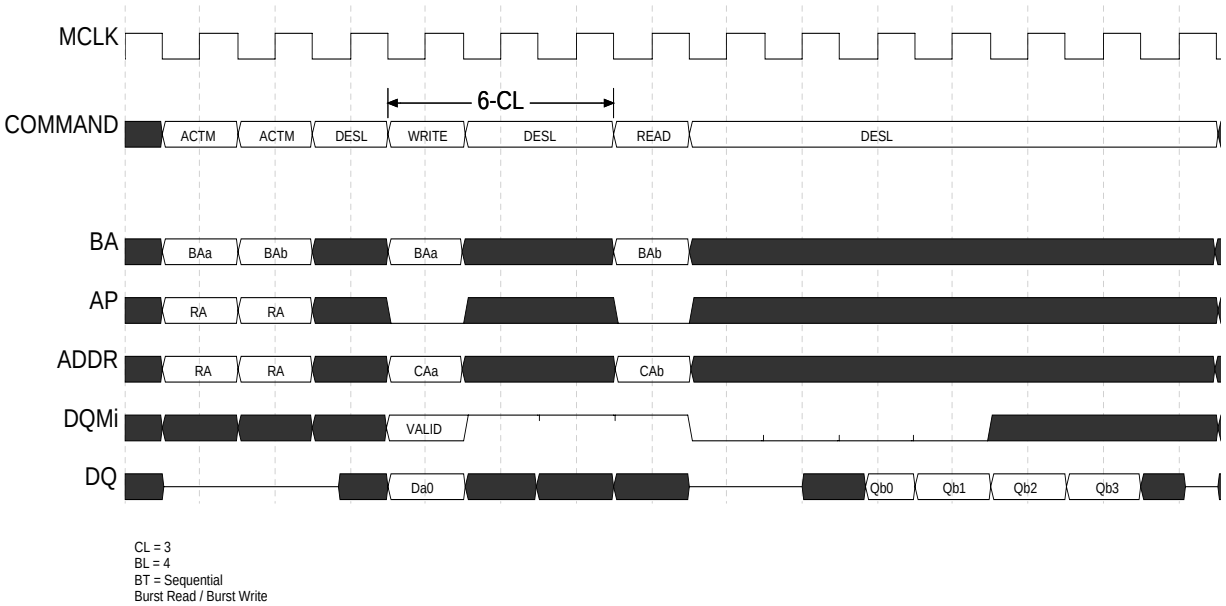
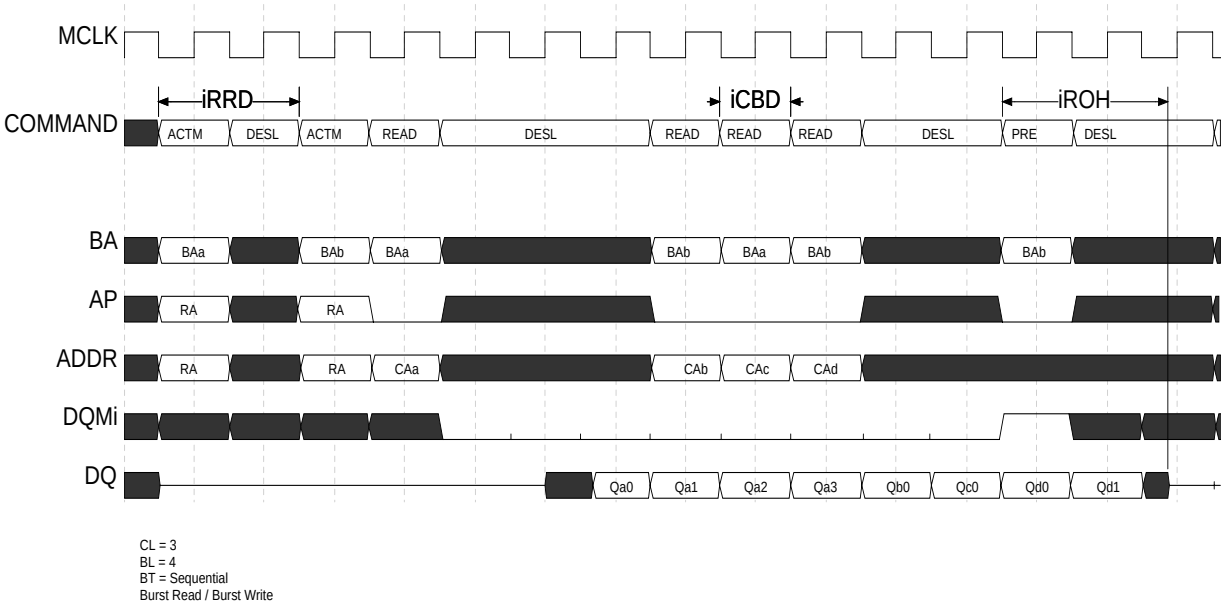
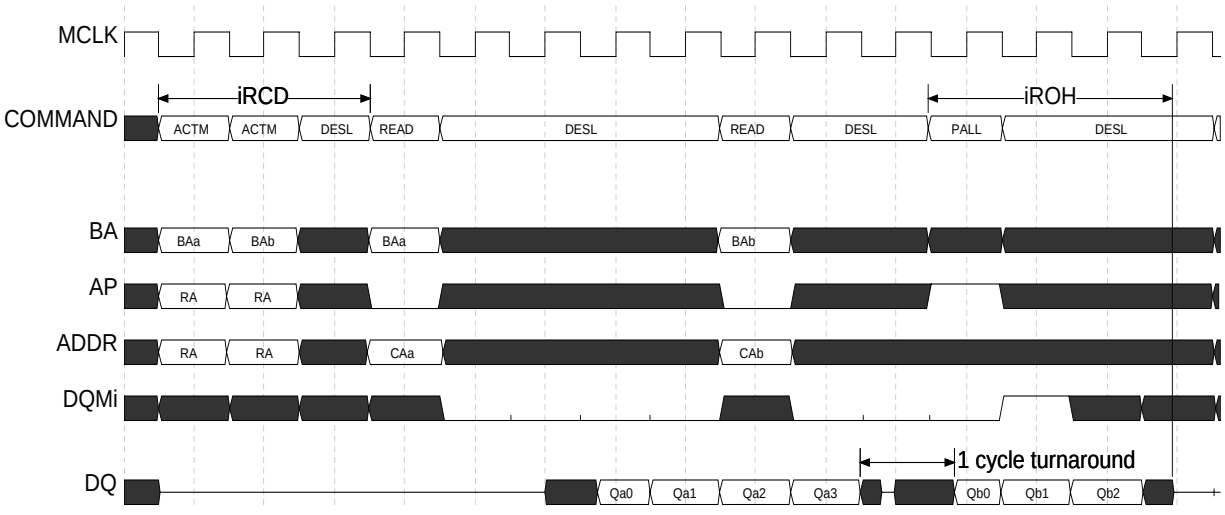


Figure A-26: Read to Both Banks(same chip select)



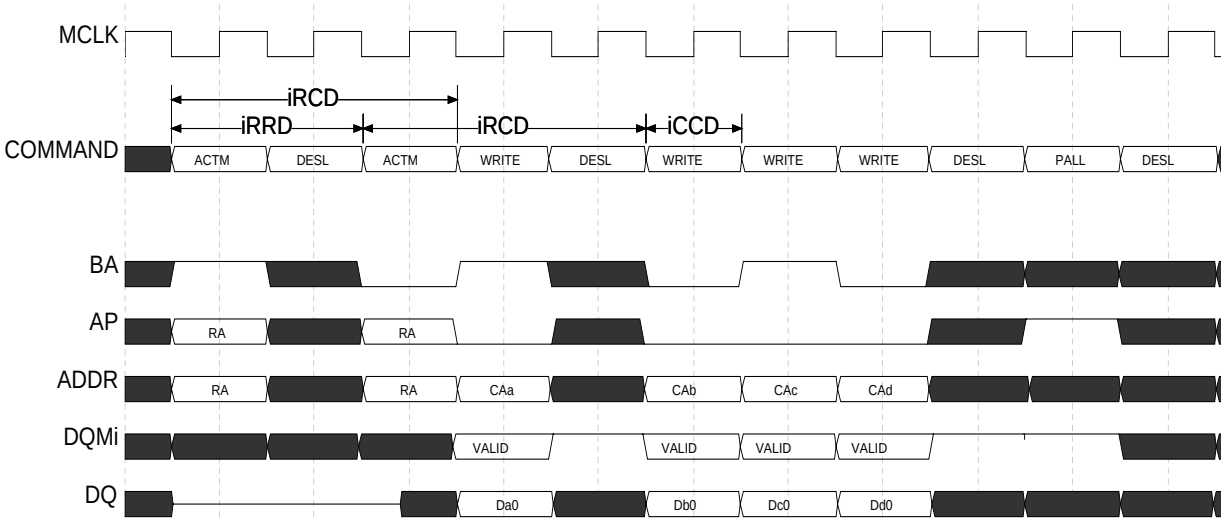
CL = 3
BL = 4
BT = Sequential
Burst Read / Burst Write

Figure A-27: Read to Different Banks (different chip select)



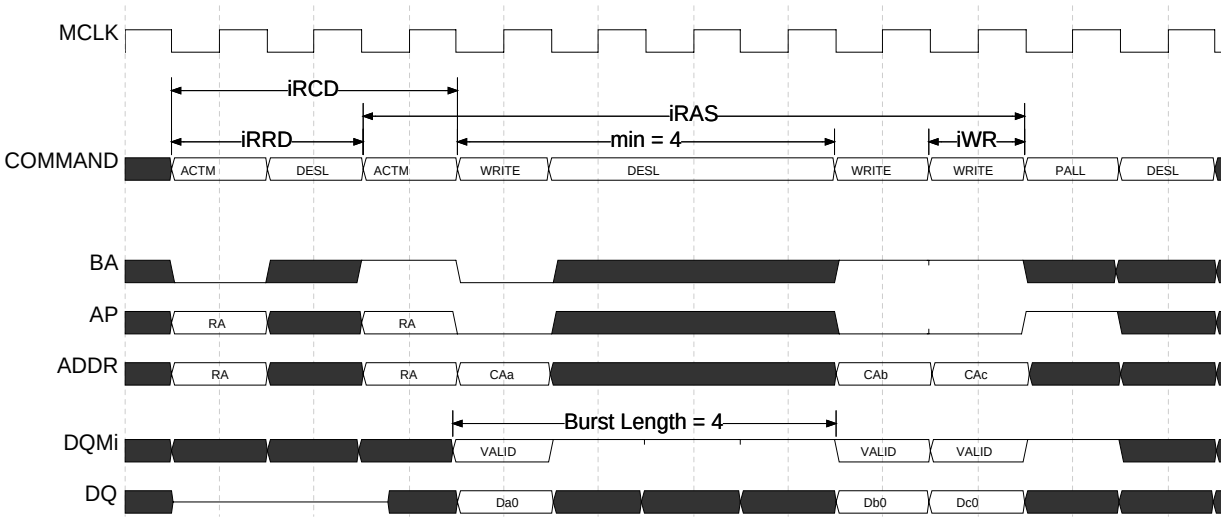
CL = 3
BL = 4
BT = Sequential
Burst Read / Burst Write

Figure A-28: Write to Both Banks(same chip select)



BL = 4
BT = Sequential
Burst Read / Burst Write

Figure A-29: Write to Different Banks (different chip select)



CL = 2 or 3
BL = 4
BT = Sequential
Burst Read / Burst Write

Figure A-30: Power-On Sequence

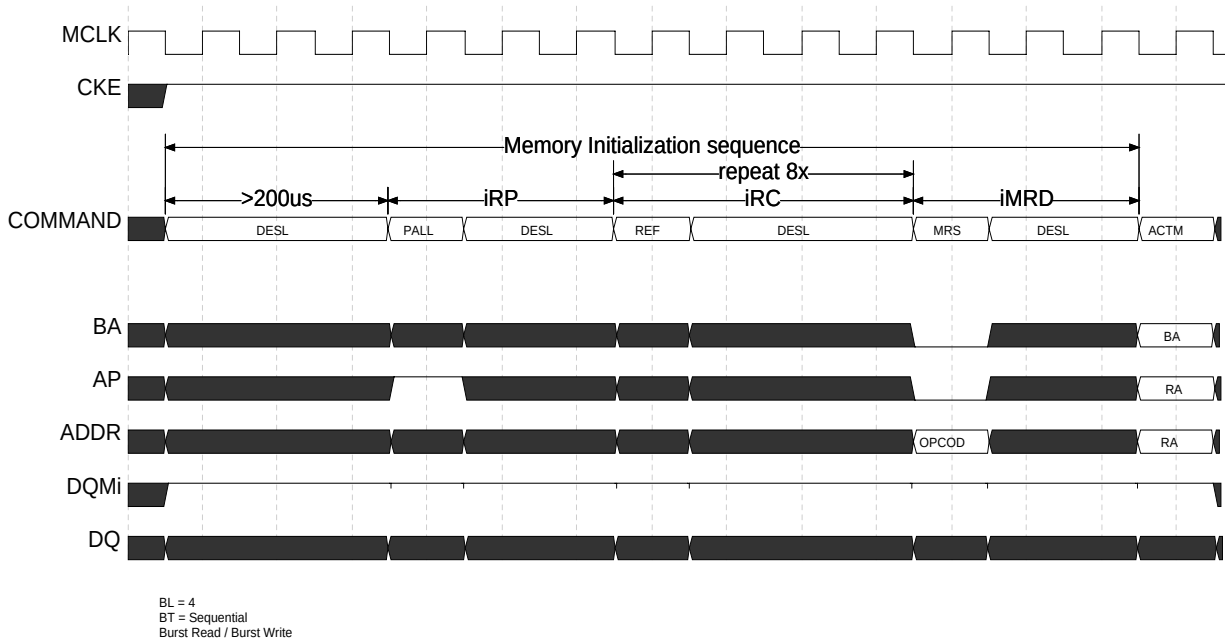


Figure A-31: Block Write and Special Mode Register command

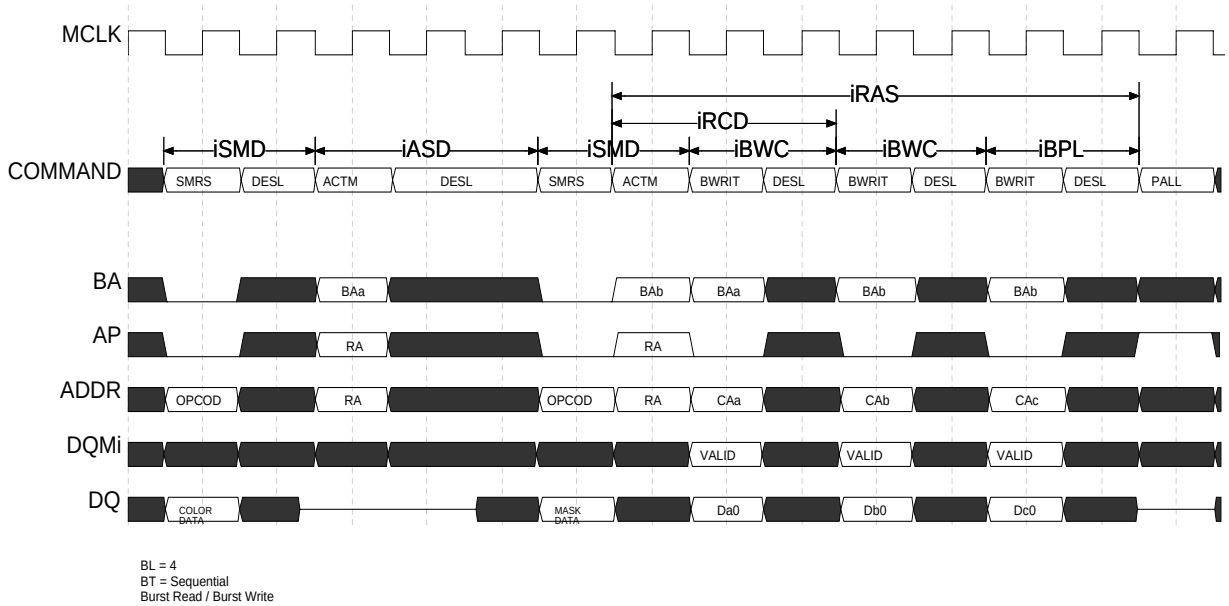


Figure A-32: Memory Refresh Sequence

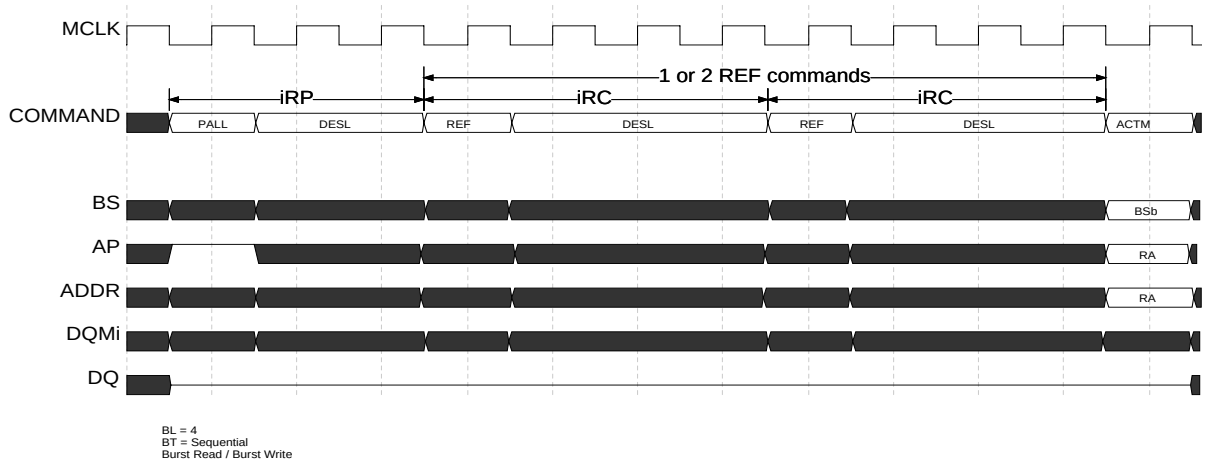


Table A-19: MGA-G200 Sync. RAM Clock-Based Parameter Table

Symbol	*Number of Clocks*	*Paramter*	*Notes(1)*
CL	(2,3,4,5)	CAS Latency	(2)
iRRD	(1,2,3)	Active command to active command (other bank)	(3)
iRCD	(2,3,4)	Active command to column address command (min.)	(3)
iRAS	(3…10)	Row Active time (min.)	(3)
iRP	(2…5)	Row Precharge time (min.)	(3)
iRC	iRAS + iRP	Row Cycle time (min.)	(3)
iWR	(1, 2)	Last data in to precharge command (write recovery time)	(3)
iEP	(-1, 1 - CL)	Last data out to early prechearge command	
iCCDrd	(1)	Read command to read command.	
iCCDwr	(1)	Write command to write command	
iCDL	(1)	Last data in to read command	
iOWD	(2)	Last data out to write command	
iMRD	(3)	MRS to row active command	
iASD	iRCD	Active command to SMRS command	(3) (4)
iSMD	(1,2)	SMRS tro block write command	(3) (4)
iBWC	(1,2)	Block write cycle time (min)	(3) (4)
iBPL	(1…5)	Block write command to precharge command	(3) (4)

(1) tCk operates in the range [7,10] ns. or [100, 143] MHz.

(2) CAS Latency is dependent upon tCK and memory device rating.

(3) Programmable paramters based on device rating and tck. For a given A.C. paramter tXXX; iXXX = tXXX/tCK rounded to the next largest integer.
As in: tck =13.3 ns, tRAS = 84 ns => iRAS = 84/13.33 = 6.32, therefore, iRAS = 7.

(4) Paramters for SGRAM

A.4.2.6 CODEC

Figure A-33: I33 Mode, Writes

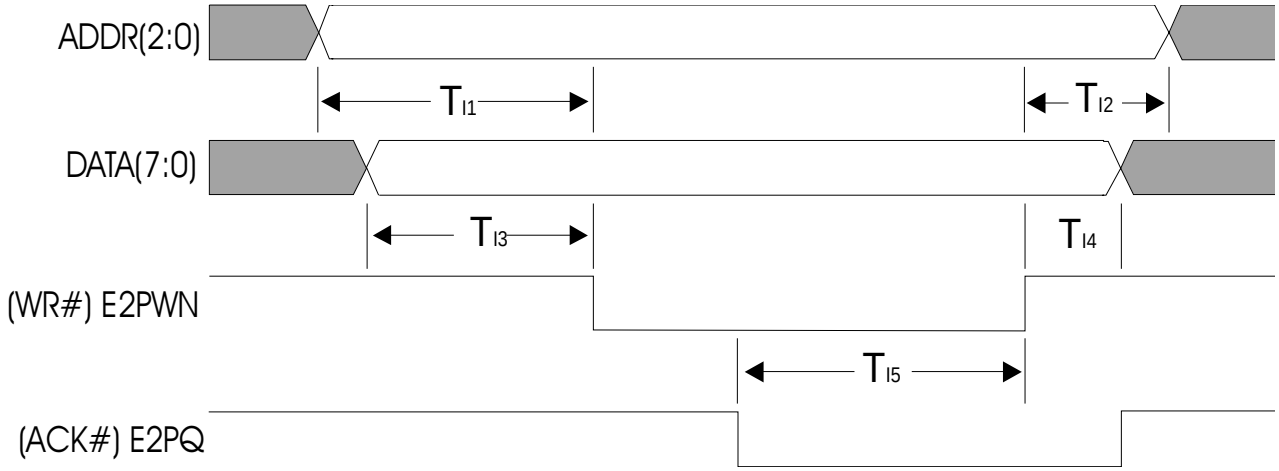


Figure A-34: I33 Mode, Reads

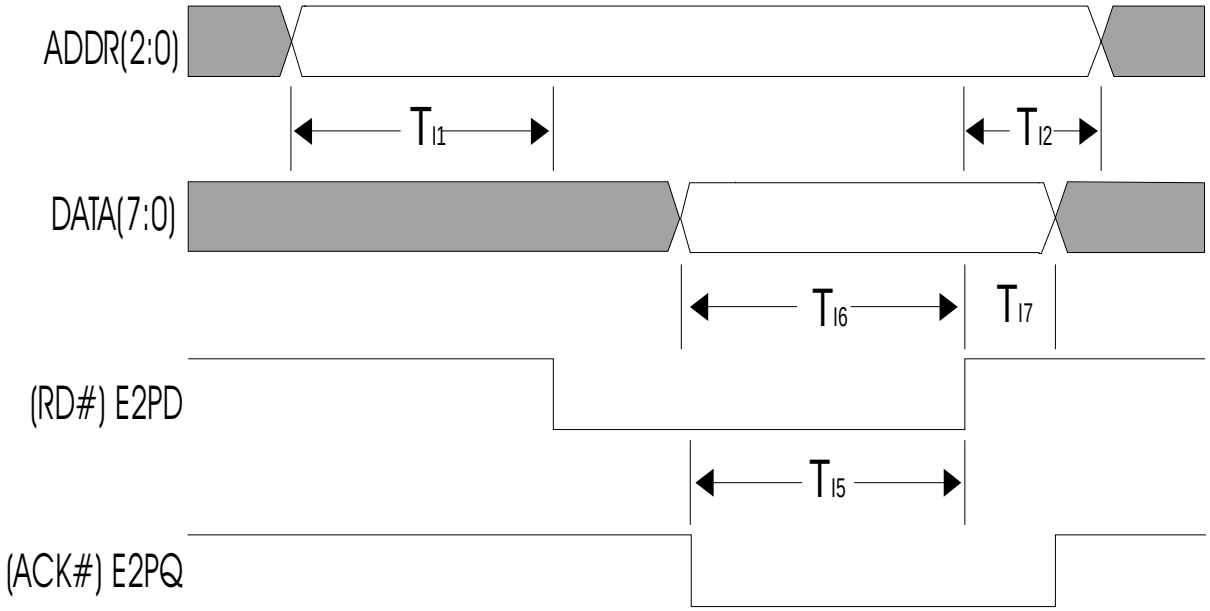


Figure A-35: VMI Mode A Writes

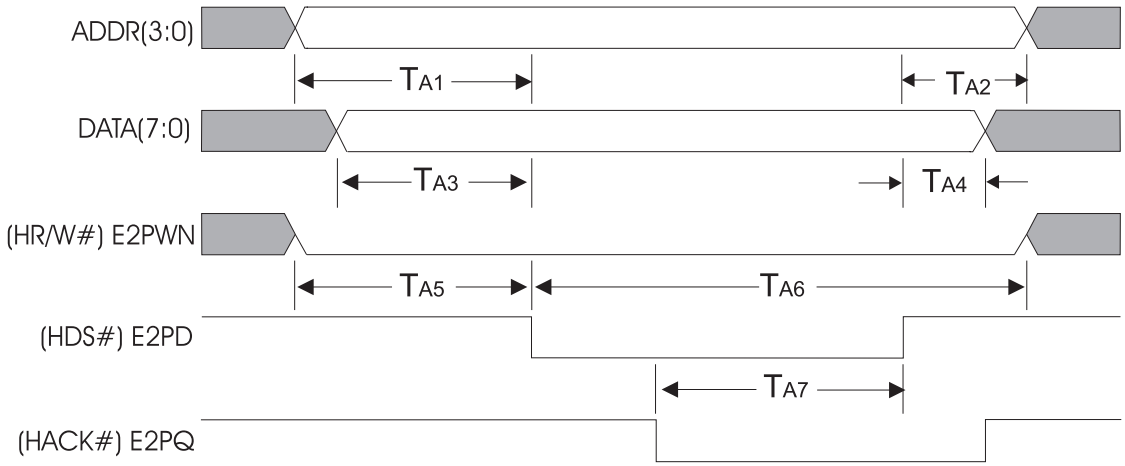


Figure A-36: VMI Mode A, Reads

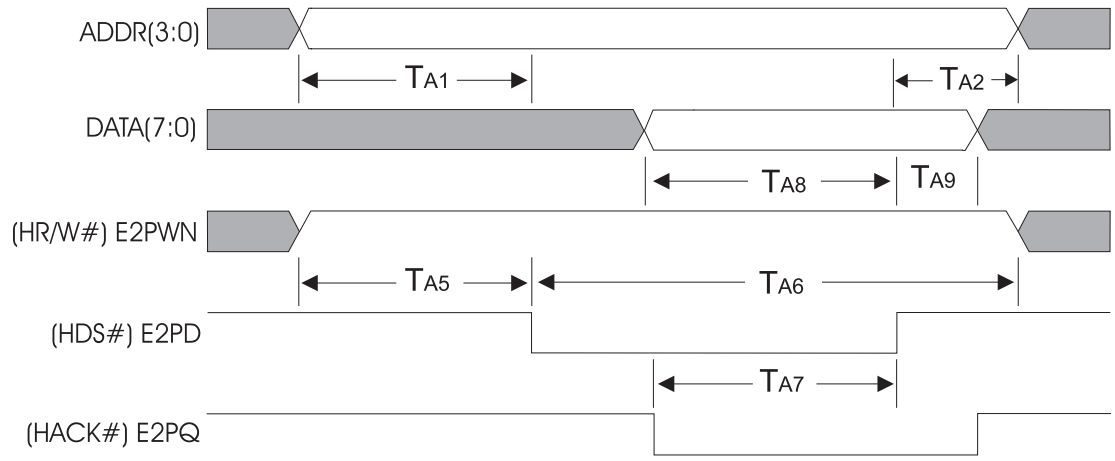


Figure A-37: VMI Mode B, Writes

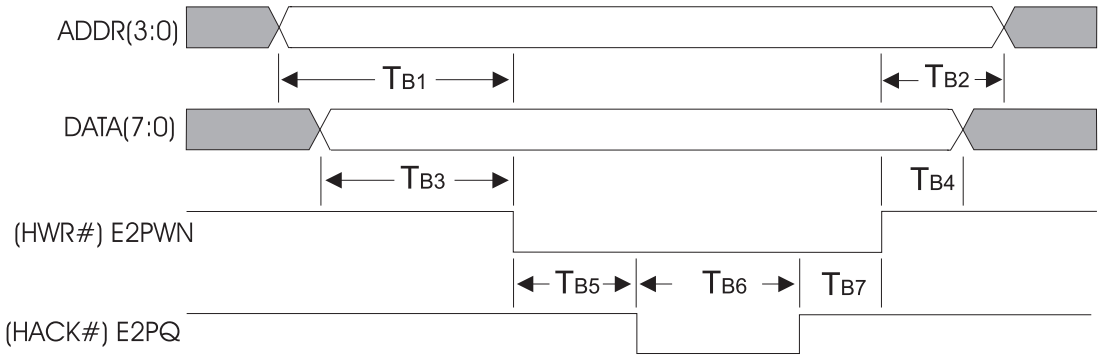


Figure A-38: VMI Mode B, Reads

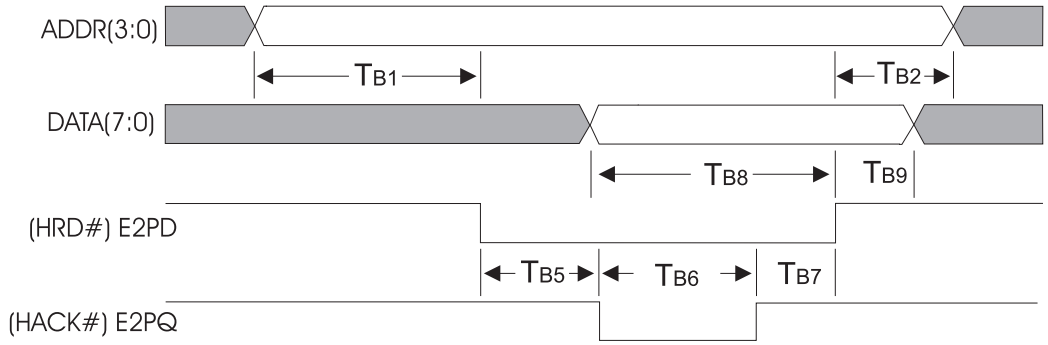


Table A-20: Codec Parameters(1)

name	*Min (ns)*	*Max (ns)*	*Comment*
TI1	13.3	-	Address setup times to falling edge of HRD#/HWR# strobe
TI2	10.0	-	Address hold time from rising edge of HRD#/HWR# strobe
TI3	11.9	-	Data setup time to falling edge of HWR# strobe
TI4	13.3	-	Data hold time form rising edge of HWR# strobe
TI5	11.9 + 2 * GCLK	17.0 + 3 * GCLK	HRD#/HWR# strobe rising edge from ACK# falling edge
TI6	12.7	-	Required data setup time to rising HRD# strobe
TI7	0	-	Required data hold time from rising HRD# strobe
TA1	14.3	-	Address setup time to falling edge of HDS# strobe
TA2	9.2	-	Address hold time from rising edge of HDS# strobe
TA3	15.2	-	Data setup time to falling edge of HDS# strobe
TA4	10.6	-	Data hold time from rising edge of HDS# strobe
TA5	15.7	-	HR/W# setup time to falling edge of HDS# strobe
TA6	11.4	-	HR/W# hold time from falling edge of HDS# strobe
TA7	12.3 + 2 * GCLK	19.7 + 3 * GCLK	HDS# strobe rising edge from HACK# falling edge
TA8	12.7	-	Required data setup time to rising edge of HDS# strobe
TA9	0	-	Required data hold time from rising edge of HDS# strobe
TB1	11.0	-	Address setup time from falling edge of HRD#/HWR#
TB2	10.0	-	Address hold time from rising edge of HRD#/HWR# strobe
TB3	11.9	-	Data setup time to falling edge of HWR# strobe
TB4	13.3	-	Data hold time form rising edge of HWR# strobe
TB5	-	12.2	HACK# falling edge from HWR#/HRD# falling edge
TB6	0	-	HACK# rising edge from HACK# falling edge
TB7	10.4 + 2 * GCLK	18.9 + 3 * GCLK	HRD#/HWR# strobe rising edge from HACK# rising edge
TB8	12.7	-	Required data setup time to rising HRD# strobe
TB9	0	-	Required data hold time from rising HRD# strobe

(1) ***Note:*** Parameters valid for GCLK = 14ns (71.4 MHz)

A.4.2.7 Video In

Figure A-39: Video In Timings

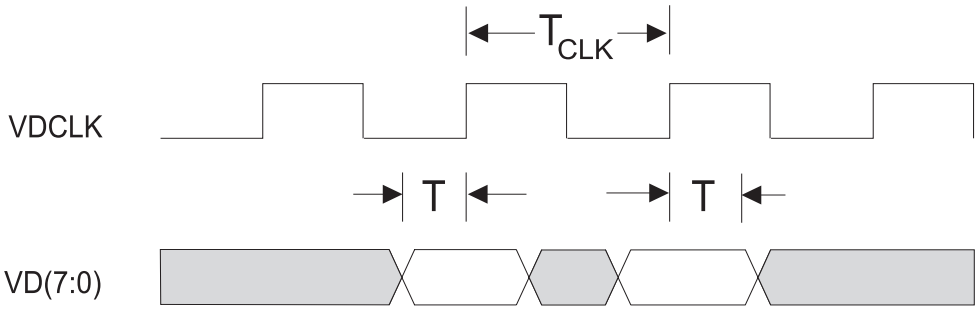
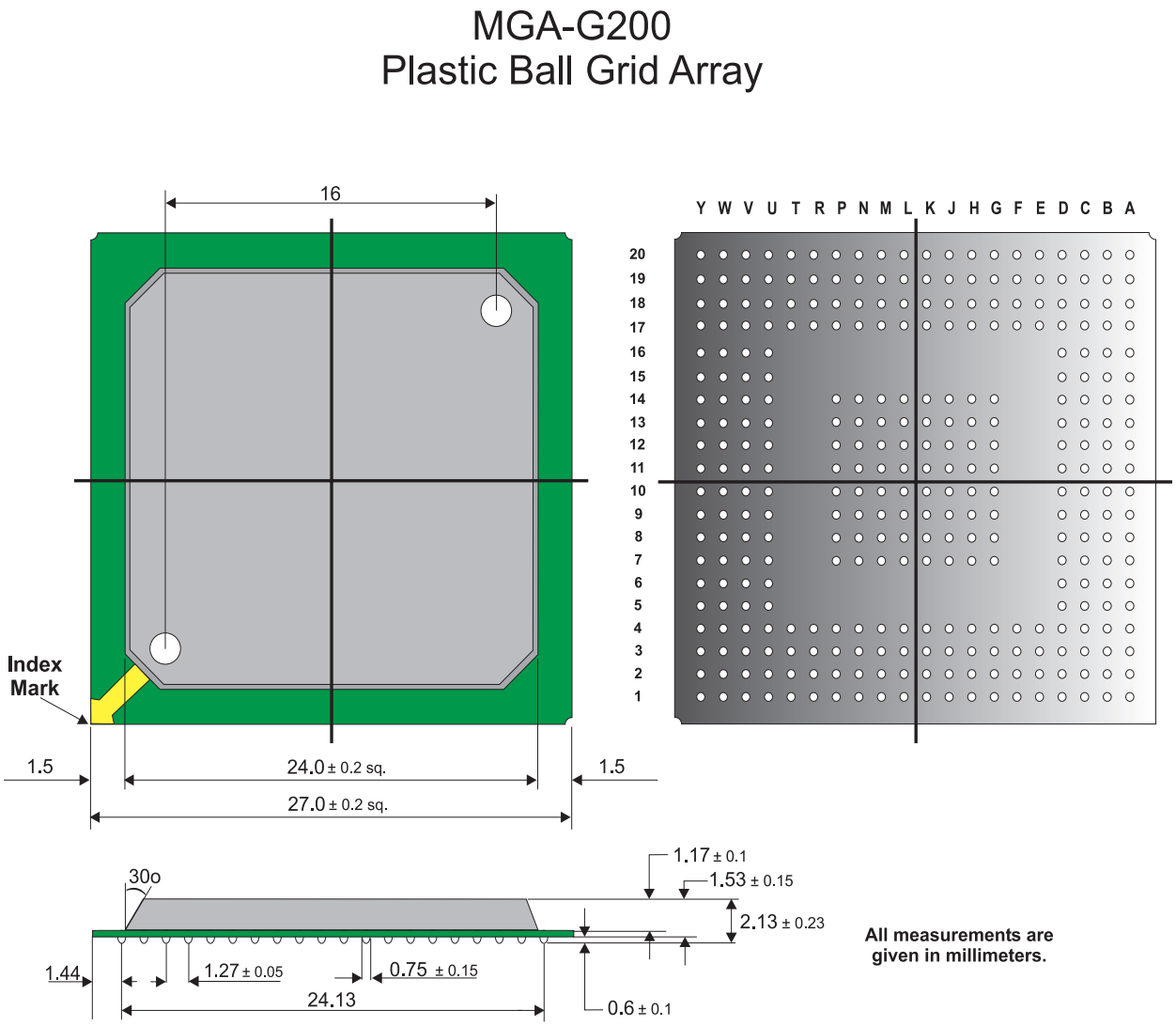


Table A-21: Video In Parametrs

Name	*min (ns)*	*Max (ns)*	*Comment*
Tclk	30	-	Input clock cycle time
Tds	4	-	Required data setup time to rising edge of VDCLK
Tdh	3	-	Required data hold time to rising edge of VDCLK

A.5 Mechanical Specification

Figure A-40: MGA-G200 Mechanical Drawing



A.6 Test Feature

The MGA-G200 is equipped with a ***nand tree*** to allow the lead connections to be verified by production test equipment. The test procedure is as follows:

1. Force the TST pins to ‘000’ to enter test mode and maintain that value during the entire test (for normal operations, the TST pins are tied to pull-ups). This will disable all output drivers except the PINTA/ pin. All pins (except PINTA/, the analog pins, and TST<2:0>) are used as input for the nand tree operation. In test mode, PINTA/ acts as a normal driver and is used for the nand tree output (for normal operations, PINTA/ is an open drain).
2. Force all signal pins to logical ‘1’. PINTA/ should read ‘1’.
3. Next, apply a ‘0’ to the first pin in the nand tree. The PINTA/ output should toggle to ‘0’.
4. Maintain the first pin at ‘0’ and toggle the next pin to ‘0’. The output should toggle again.
5. Continue the shift-in of ‘0’, following the nand tree order and monitoring the toggling of the PINTA/ pin for each new test vector.

A.6.1 Nand Tree Order (for MGA-G200-AGP only)

Table A-22: AGP Nand Tree Order (Part 1 of 2)

Tree Order	*Pin Name*	*Tree Order*	*Pin Name*	*Tree Order*	*Pin Name*
1 (1st pin)	ST<0>	50	PAD<4>	99	MA<2>
2	ST<2>	51	PAD<1>	100	MA<4>
3	PRST/	52	PAD<13>	101	MA<0>
4	ST<1>	53	PAD<9>	102	MA<3>
5	SBA<0>	54	PPAR	103	MA<5>
6	SBA<2>	55	PAD<0>	104	MA<7>
7	PREQ/	56	PAD<2>	105	MA<9>
8	SB_STB	57	PAD<15>	106	MA<8>
9	PCLK	58	PAD<11>	107	MCLK2
10	SBA<1>	59	PAD<6>	108	MA<6>
11	SBA<3>	60	E2PCS/	109	MCS<0>/
12	SBA<4>	61	MDQ<3>	110	MA<10>
13	SBA<5>	62	E2PCLK	111	MCLK
14	SBA<6>	63	MDQ<0>	112	MWE/
15	PAD<31>	64	MDQ<4>	113	MCAS/
16	SBA<7>	65	MDQ<1>	114	MRAS/
17	PAD<30>	66	MDQ<2>	115	MCS<1>/
18	PAD<29>	67	MDQ<7>	116	MDSF
19	PAD<27>	68	MDQ<5>	117	MCS<3>/
20	PAD<25>	69	MDQ<6>	118	MCS<2>/
21	PAD<28>	70	MDQ<10>	119	MDQ<34>
22	AD_STB<1>	71	MDQ<11>	120	MDQ<32>
23	PAD<21>	72	MDQ<8>	121	MDQ<33>
24	PAD<23>	73	MDQ<9>	122	MDQ<40>
25	PAD<24>	74	MDQ<12>	123	MDQ<37>
26	PAD<26>	75	MDQ<14>	124	MDQ<35>
27	PAD<17>	76	MDQ<13>	125	MDQ<36>
28	PAD<19>	77	MDQM<0>	126	MDQ<38>
29	PAD<22>	78	MDQM<1>	127	MDQ<41>
30	PCBE<3>/	79	MDQ<15>	128	MDQ<39>
31	PDEVSEL/	80	MDQM<2>	129	MDQ<43>
32	PCBE<2>/	81	MDQM<3>	130	MDQ<42>
33	PAD<20>	82	MDQ<16>	131	MDQ<44>
34	PAD<14>	83	MDQ<17>	132	MDQM<4>
35	PIRDY/	84	MDQ<18>	133	MDQ<46>
36	PAD<12>	85	MDQ<19>	134	MDQ<47>
37	PAD<16>	86	MDQ<20>	135	MDQ<45>
38	PCBE<1>/	87	MDQ<21>	136	MDQ<48>
39	AD_STB<0>	88	MDQ<22>	137	MDQM<5>
40	PAD<18>	89	MDQ<23>	138	MDQM<6>
41	PAD<10>	90	MDQ<26>	139	MDQM<7>
42	PAD<8>	91	MDQ<24>	140	MDQ<49>
43	PTRDY/	92	MDQ<27>	141	MDQ<50>
44	PAD<7>	93	MDQ<29>	142	MDQ<52>
45	PAD<3>	94	MDQ<25>	143	MDQ<51>
46	PFRAME/	95	MDQ<28>	144	MDQ<53>
47	PSTOP/	96	MDQ<30>	145	MDQ<56>
48	PAD<5>	97	MA<1>	146	MDQ<55>
49	PCBE<0>/	98	MDQ<31>	147	MDQ<54>

Table A-22: AGP Nand Tree Order (Part 2 of 2)

Tree Order	Pin Name	Tree Order	Pin Name	Tree Order	Pin Name
148	MDQ<57>	171	VDOUT<8>	194	E2PQ
149	MDQ<59>	172	VDOUT<7>	195	VBLANK/
150	MDQ<58>	173	VDOUT<6>	196	E2PW/
151	MDQ<60>	174	VDOUT<5>	197	DDC<3>
152	MDQ<62>	175	VDOUT<4>	198	DDC<2>
153	MDQ<61>	176	VDOUT<3>	199	DDC<1>
154	MDQ<63>	177	VDOUT<2>	200	DDC<0>
155	VVSYNC/	178	VDOUT<1>	201	MISC<2>
156	VHSYNC/	179	VDOUT<0>	202	EXTINT/
157	VD<7>	180	VOBLANK/	203	VIDRST
158	MISC<1>	181	VDOCLK	204	EXTRST/
159	VD<6>	182	HDATA<7>	205	PGNT/
160	VD<5>	183	HDATA<6>		
161	MISC<0>	184	HDATA<5>		
162	VD<4>	185	HDATA<4>		
163	VD<3>	186	HDATA<2>		
164	VD<2>	187	HDATA<3>		
165	VDCLK	188	HDATA<1>		
166	VD<1>	189	HDATA<0>		
167	VD<0>	190	VESYNC		
168	VDOUT<11>	191	VEDCLK		
169	VDOUT<10>	192	VEVIDEO		
170	VDOUT<9>	193	E2PD		

A.6.2 Nand Tree Order (for MGA-G200-PCI only)

Table A-23: PCI Nand Tree Order (Part 1 of 2)

Tree Order	*Pin Name*	*Tree Order*	*Pin Name*	*Tree Order*	*Pin Name*
1 (1st pin)	PRST/	50	MDQ<0>	99	MWE/
2	PREQ/	51	MDQ<4>	100	MCAS/
3	PCLK	52	MDQ<1>	101	MRAS/
4	PIDSEL	53	MDQ<2>	102	MCS<1>/
5	PDEVSEL/	54	MDQ<7>	103	MDSF
6	PAD<31>	55	MDQ<5>	104	MCS<3>/
7	PAD<30>	56	MDQ<6>	105	MCS<2>/
8	PAD<29>	57	MDQ<10>	106	MDQ<34>
9	PAD<27>	58	MDQ<11>	107	MDQ<32>
10	PAD<25>	59	MDQ<8>	108	MDQ<33>
11	PAD<28>	60	MDQ<9>	109	MDQ<40>
12	PAD<21>	61	MDQ<12>	110	MDQ<37>
13	PAD<23>	62	MDQ<14>	111	MDQ<35>
14	PAD<24>	63	MDQ<13>	112	MDQ<36>
15	PAD<26>	64	MDQM<0>	113	MDQ<38>
16	PAD<17>	65	MDQM<1>	114	MDQ<41>
17	PAD<19>	66	MDQ<15>	115	MDQ<39>
18	PAD<22>	67	MDQM<2>	116	MDQ<43>
19	PCBE<3>/	68	MDQM<3>	117	MDQ<42>
20	PCBE<2>/	69	MDQ<16>	118	MDQ<44>
21	PAD<20>	70	MDQ<17>	119	MDQM<4>
22	PAD<14>	71	MDQ<18>	120	MDQ<46>
23	PIRDY/	72	MDQ<19>	121	MDQ<47>
24	PAD<12>	73	MDQ<20>	122	MDQ<45>
25	PAD<16>	74	MDQ<21>	123	MDQ<48>
26	PCBE<1>/	75	MDQ<22>	124	MDQM<5>
27	PAD<9>	76	MDQ<23>	125	MDQM<6>
28	PAD<18>	77	MDQ<26>	126	MDQM<7>
29	PAD<10>	78	MDQ<24>	127	MDQ<49>
30	PAD<8>	79	MDQ<27>	128	MDQ<50>
31	PTRDY/	80	MDQ<29>	129	MDQ<52>
32	PAD<7>	81	MDQ<25>	130	MDQ<51>
33	PAD<3>	82	MDQ<28>	131	MDQ<53>
34	PFRAME/	83	MDQ<30>	132	MDQ<56>
35	PSTOP/	84	MA<1>	133	MDQ<55>
36	PAD<5>	85	MDQ<31>	134	MDQ<54>
37	PCBE<0>/	86	MA<2>	135	MDQ<57>
38	PAD<4>	87	MA<4>	136	MDQ<59>
39	PAD<1>	88	MA<0>	137	MDQ<58>
40	PAD<13>	89	MA<3>	138	MDQ<60>
41	PAD<0>	90	MA<5>	139	MDQ<62>
42	PPAR	91	MA<7>	140	MDQ<61>
43	PAD<2>	92	MA<9>	141	MDQ<63>
44	PAD<15>	93	MA<8>	142	VVSYNC/
45	PAD<11>	94	MCLK2	143	VHSYNC/
46	PAD<6>	95	MA<6>	144	VD<7>
47	E2PCS/	96	MCS<0>/	145	MISC<1>
48	MDQ<3>	97	MA<10>	146	VD<6>
49	E2PCLK	98	MCLK	147	VD<5>

Table A-23: PCI Nand Tree Order (Part 2 of 2)

Tree Order	*Pin Name*	*Tree Order*	*Pin Name*	*Tree Order*	*Pin Name*
148	MISC<0>	163	VDOUT<3>	178	VEDCLK
149	VD<4>	164	VDOUT<2>	179	VEVIDEO
150	VD<3>	165	VDOUT<1>	180	E2PD
151	VD<2>	166	VDOUT<0>	181	E2PQ
152	VDCLK	167	VOBLANK/	182	VBLANK/
153	VD<1>	168	VDOCLK	183	E2PW/
154	VD<0>	169	HDATA<7>	184	DDC<3>
155	VDOUT<11>	170	HDATA<6>	185	DDC<2>
156	VDOUT<10>	171	HDATA<5>	186	DDC<1>
157	VDOUT<9>	172	HDATA<4>	187	DDC<0>
158	VDOUT<8>	173	HDATA<2>	188	MISC<2>
159	VDOUT<7>	174	HDATA<3>	189	EXTINT/
160	VDOUT<6>	175	HDATA<1>	190	VIDRST
161	VDOUT<5>	176	HDATA<0>	191	EXTRST/
162	VDOUT<4>	177	VESYNC	192	PGNT/

A.7 Ordering Information

- To receive an AGP version of the MGA-G200, order: **MGA-G200A**
- To receive an PCI version of the MGA-G200, order: **MGA-G200P**

Appendix A: Changes

B.1 Changes in the Document Since Revision 0300

This appendix contains the revision history of the MGA-G200 Specification. This section is meant for customers who need to identify the functional changes that took place between various versions of the chip. These changes may or may not be reflected elsewhere in this manual.

B.1.1 Chip Revision Notes and Changes

The changes are in reference to rev. C of the chip (document number 10581-MS-0301) and may or may not be reflected elsewhere in this manual.

B.1.1.1 Register Modification

Bit 18 of the **OPMODE** (MGABASE1 + 1E54h) is set to '1' instead of '0' (rev A and B) upon a hard or soft reset (see page 3-126).

B.1.1.2 CRTC Latency Formula

The **MAXHIPRI** formula has been changed to read: MIN(HIPRILVL, (**SCALE** + 1)(Tmclk / Tpixclk)), (see'CRTC Latency Formulas' on page 4-67).

B.1.1.3 Codec Data Decompression Explanation

The section of the Programmer's Guide explaining the codec data decompression with the **stopcodec** field of the **CODECCTRL** register enabled has been modified (see 'Decompressing data' on page 4-93).

The test has been modified to refelect the fact that if the **stopcodec** bit is set during decompression then the codec ***cannot*** be re-enabled solely by toggling the **stopcodec** field.

B.1.1.4 Index Modification

The index for **XCOLMSK** has been changed to 40h (see page 3-314).

B.2 Changes in the Document

This appendix contains the revision history of the MGA-G200 Specification. Although this is the first official release (10581-MS-0300, dated May 11, 1998), a preliminary release (10581-MS-0200, dated Dec 23, 1997) was made that contains information that has since been updated. This section is meant for customers who need to identify the functional changes that took place between various versions of the chip. These changes may or may not be reflected elsewhere in this manual.

B.2.1 Chip revision Notes and Changes

B.2.1.1 Revision Change

The revision change to the chip is reflected in the revision field of the **CLASS** register:

00h	rev A
01h	rev B

Note: Revision A chips were prototypes; they are no longer supported.

B.2.1.2 Bus Mastering

In revision A chips, bus_mastered (both AGP and PCI) DMA cycles may not work reliably in all systems.

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

Power Graphic (cont)

Power Graphic (cont)

Power Graphic (cont)

Power Graphic (cont)

VGA Mode Register Fields

VGA (cont)

Alphabetical List of Register Fields

DAC Register Fields

Notes

Notes