Programmer's Guide to the *mach32* Registers

Information in this document is **proprietary** and **confidential.**

Technical Reference Manuals

P/N: REG688000-15

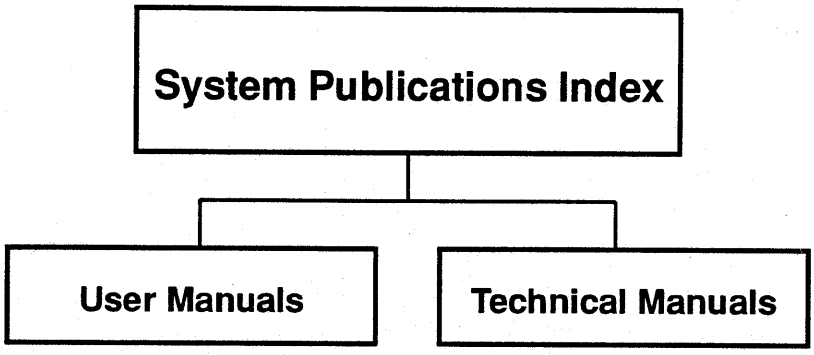
ATI Technologies Inc.
3761 Victoria Park Ave.
Scarborough, Ontario
Canada M1W 3S2

User Support: 416-756-0711
Offices: 416-756-0718
Fax: 416-756-0720
BBS: 416-756-4591

Perfecting the PC

P/N: REG688000-15
Release 1.5

The information contained in this document has been carefully checked and is believed to be entirely reliable. No responsibility is assumed for inaccuracies. ATI reserves the right to make changes at any time to improve design and supply the best product possible.



- Graphics Accelerators User's Guide
- Graphics Ultra Pro User's Guide
- Graphics Ultra + User's Guide

- mach32 Graphics Controller Specifications (GCS688xxx-xx)
- Programmer's Guide to the ***mach32*** Registers (REG688000-xx)
- Graphics Accelerators and VGA BIOS Kits (BIO688000-xx)
- Programmer's Guide to the ***mach32*** Adapter Interface (AIF688000-xx)

Record of Revisions

Release	Date	Description of changes
1.0	92-06	Original release.
1.1	92-06	Reprinted with minor, typographical changes.
1.2	92-07	Reprinted with minor, typographical changes.
1.21	92-12	Registers 46E8h-W, 12EEh-R, 16EEh-R revised.
1.3	93-03	Reprinted with minor, typographical changes.
1.4	93-03	ATI68800 "LX", "-6" and "AX" updates.
1.5	93-05	PCI updates — 22EE-RW, 32EE-RW, 36EE-RW, 5EEE-RW, 62EE-W, 6AEE-RW, and configuration registers.

Table of Contents

This page intentionally left blank.

Part I

1

Introduction

About This Manual

This manual is a register description written for OEM designers and developers who wish to integrate ATI's ***mach32*** graphics controller accelerators in their hardware. It is also written for programmers who wish to program directly to hardware registers to optimize graphics performance. A working knowledge of the 80x86 family of PCs, Assembler, "C", and 2D graphics is assumed.

The contents of this manual cover the following controllers — ATI68800AX, ATI68800-6, ATI68800LX, and ATI68800-3. In general, the "LX" and "AX" are quite similar to the ATI68800-6 — the "LX" is basically a "-6" without VRAM support, the "AX" a "-6" with support for an additional local bus type, PCI. Therefore register descriptions marked as "-6 specific" are equally applicable to both these controllers unless otherwise indicated. (Refer to the appendixes for controller differences.) The registers of these controllers are backward-compatible with the ATI68800-3.

This manual has three parts. The first part contains an introduction and a programmer's overview. Part II describes the VGA controller, the display modes, memory organization, and both VGA-compatible and ATI-extended registers. Part III describes the coprocessor engine, pixel data paths, draw engine functions, and both 8514/A-compatible and ATI-extended registers. Functional and bit-field differences are indicated in the register descriptions (Part III) as well as summarized for each product family in the appendixes (G and H).

Features

- Low cost, high performance single chip graphics solution suitable for board or system level implementations
- 100% register-level hardware compatible to the IBM VGA as well as the IBM 8514/A graphics adapter
- I/O Bus Types: ISA (8 and 16 bit), EISA (32 bit), and Microchannel (16-bit and 32-bit)
- Local Buses: 486, 386DX, 386SX, VESA VL-bus, and PCI

- Fully programmable direct memory interface
- Memory Types: VRAM 256Kx4, 256Kx16; DRAM 256Kx4, 256Kx16
- Memory Sizes: 512KB, 1MB, 2MB, 4MB (using VRAM 256Kx16 only)
- Colors/Resolutions:
  4 and 8 bpp (Bits Per Pixel) to 1280x1024
  16 bpp to 1024x768
  24 bpp to 800x600
- High-speed point-to-point line draw; coprocessor supports 16 bpp modes
- Supports memory mapped registers (except ATI68800-3)
- Hardware support for ATI's ***Crystal Font*** technology
- Support for overscan monitors in both VGA and 8514/A modes
- Hardware cursor: 64x64x2; XGA function compatible
- High-speed polygon fill
- Hardware assisted line and polygon pre-clipping
- Support for packed bitmap data transfers
- 32 8-bit pixel color pattern registers
- 32 1-bit monochrome pattern registers
- Enhanced bit block transfer (blit) operation to allow for better off-screen memory management
- Extended 16-entry data FIFO
- Improved FIFO status registers providing dramatic improvements in data throughput
- Card ID feature supports up to seven display adapters simultaneously in a system
- 0.7 micron CMOS VLSI technology
- 208-pin PQFP

Notation Convention

Mnemonics are used throughout this manual in place of hardware register names. The naming convention for registers and/or bit fields is as follows:

- **Register_Mnemonic**
- **Register_Mnemonic[Bit_Numbers or Field_Name]**

The example below is the mnemonic for the Miscellaneous Output register.

GENMO

The examples below describe the same entity in two different ways — as bits 2 and 3 or as the Clock_Select field of the GENMO register. Note that *square brackets* are used.

GENMO[3:2] or
GENMO[CLOCK_SELECT]

The convention for naming signals is similar to that used for naming hardware registers.

- **Signal_Name**

When several signals of an identical function are described, the part of the signalname that differs may be shown in brackets. For example, the four Select signals — SEL0#, SEL1#, SEL2#, and SEL3# — may be represented as follows. Note that *round brackets* are used.

SEL(0:3)#

This page intentionally left blank.

2

Programmer's Overview

Overview

The ***mach32*** is a VLSI graphics controller chip consisting of an 8514/A-compatible accelerator and a VGA-compatible graphics controller. The accelerator is also known as the coprocessor or the draw engine. This chip may be used to implement board or system level solutions supporting a range of I/O bus types, memory types, memory sizes, screen resolutions, and color depths. In addition the chip supports a number of extended registers and enhancements.

The built-in VGA controller can be disabled to allow the accelerator to co-exist with an alternate external VGA. This manual will concentrate mostly on the programming of the draw engine.

The draw engine is 100% 8514/A-compatible in 4bpp and 8bpp display modes, and is mostly 8514/A-compatible in 16bpp modes. All ATI draw engine extensions are available in 4, 8, and 16bpp modes. The draw engine supports the 16bpp modes in the following RGB weightings — 555, 565, 655, and 664. It supports 24 or 32bpp modes as follows — RGB, BGR, RGBa, and aBGR.

A configurable linear memory aperture is available for all modes, including VGA. It may be configured to 64K paged at A000h or to 1M paged or 4M linear on any 1MB boundary. The aperture is primarily used for increasing throughput on host-to-screen and screen-to-host data transfers. Screen memory can be shared between the VGA and the draw engine. ***mach32***-aware applications may re-configure the memory boundary (register) dynamically to give more or less memory to the VGA or the accelerator.

Since ***mach32*** boards may be equipped with a number of different RAMDACs, a protocol has been established for accelerator mode switching by ROM calls. VGA mode switching is accomplished with the standard INT 10h interface. The VGA and accelerator should be treated as two logically separate entities. Bus type and memory type should be transparent to applications programmers.

A variable size hardware cursor is available for all non-VGA modes. This cursor may be composed of 2 colors, transparent, or complement. It is XGA function compatible.

All I/O writes to the draw engine are written to the 16-word command FIFO. Since standard 8514/A adapters only have an 8-word command FIFO, this draw engine use the FIFO status extensions to poll the state of the eight extra words.

8514/A-compatible draw primitives include rectangle fill, blit, line draw, short stroke vector, and in-place polygon fill. ATI extensions provide for non-conforming rectangle fill, non-conforming blit, point-to-point line draw, line draw, short stroke vector, horizontal raster draw, and blit polyfill. Extended data paths are available for packed monochrome and color patterns in addition to existing color, host, RAM, or monochrome pattern selections. The pixel ALU has 16 logical and 16 arithmetic functions available in 4bpp and 8bpp modes; and 16 logical functions in 16bpp modes.

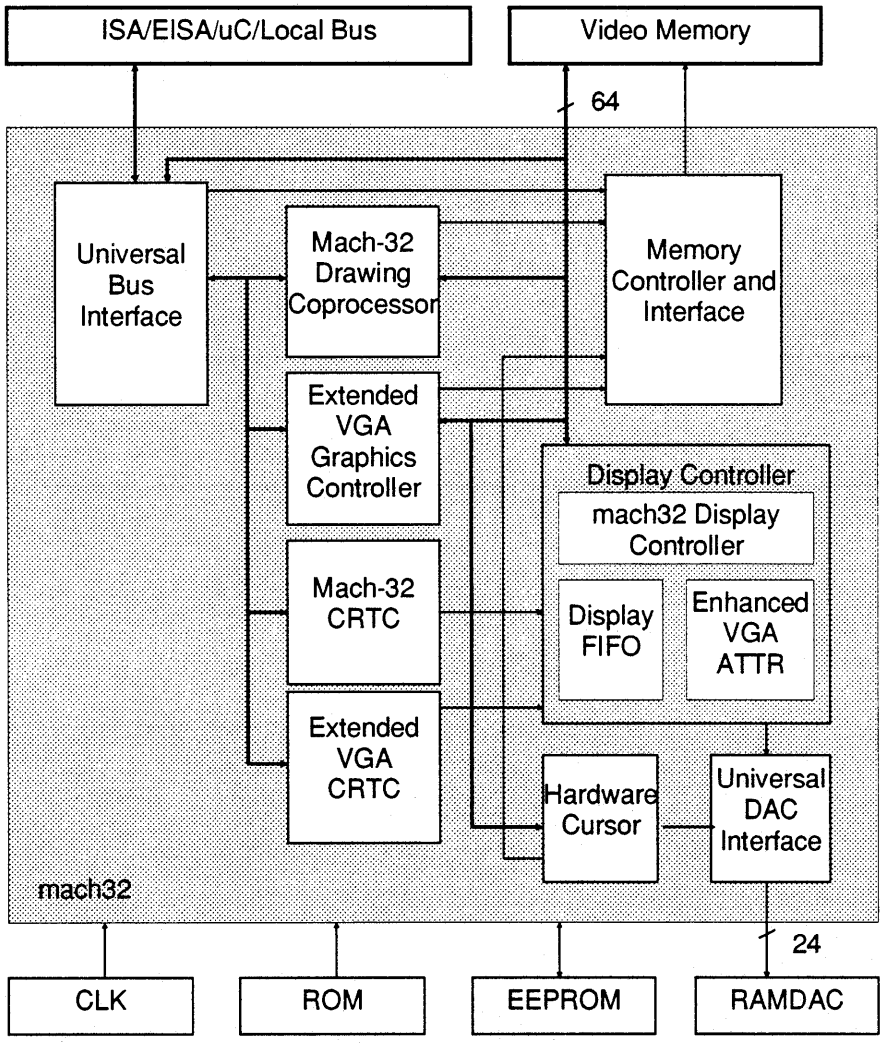


Figure 2-1. Mach32 Drawing Coprocessor

Power On Setup Registers

Microchannel POS Registers

POS register addresses for the Microchannel configuration (uC) are from 100h to 105h inclusive. The input signal -CD SETUP must be active for these registers to be accessed.

uC Setup Mode ID Byte 1 Register					
0100 (R)	SETUP_ID1, uC				
	VGA	-	-	Mach 32	
Bit	Mnemonic	Description			
7:0	SETUP_ID1(uC)	Setup Identification byte 1 is as follows: ATI68800-3 is 89h. ATI68800-6 is 89h. (8Ah when VGA is disabled by external strap bit.)			

uC Setup Mode ID Byte 2 Register				
0101 (R)	SETUP_ID2, uC			
	VGA	-	-	Mach 32
Bit	Mnemonic	Description		
7:0	SETUP_ID2(uC)	Setup Identification byte 2 is 80h.		

uC Setup Mode Option Select Register				
0102 (R/W)	SETUP_OPT, uC			
	VGA	-	-	Mach 32
Bit	Mnemonic	Description		
0	SETUP_OPT(uC)	**0** = Video adapter disabled — responds only to POS registers. **1** = Video adapter enabled		
7:1	-	Reserved		

uC ROM Base Address Setup Register				
0103 (W)	**ROM_SETUP, uC**			
	VGA	**-**	**-**	**Mach 32**
Bit	**Mnemonic**	**Description**		
0	-	**0** = BIOS ROM decodes disabled **1** = BIOS ROM decodes enabled This bit is readable in all except ATI68800-3.		
7:1	-	Setting of ROM base address corresponding to CPU addresses A17:A11, in 8514 only mode.		

uC Setup Byte 1 Register				
0104 (W)	**SETUP_1, uC**			
	VGA	**-**	**-**	**Mach 32**
Bit	**Mnemonic**	**Description**		
7:0	-	General purpose R/W register. The value of this register is read back at 52EEh.		

uC Setup Byte 2 Register				
0105 (W)	**SETUP_2, uC**			
	VGA	**-**	**-**	**Mach 32**
Bit	**Mnemonic**	**Description**		
7:0	-	General purpose R/W register. The value of this register is read back at 52EFh.		

EISA Bus POS Registers

POS register address for EISA are from 0zC80h to 0zC87h inclusive. The input signal AEN must be "0" for these registers to be accessed. The "z" in the address represents the bus slot number.

EISA Setup Mode ID Byte 1 Register				
0zC80 (R)	SETUP_ID1, EISA			
	VGA	-	-	Mach 32
Bit	**Mnemonic**	**Description**		
7:0	SETUP_ID1(EISA)	Setup Identification byte 1 is 06h.		
Note: The "z" in the address is the bus slot number.				

EISA Setup Mode ID Byte 2 Register				
0zC81 (R)	SETUP_ID2, EISA			
	VGA	-	-	Mach 32
Bit	**Mnemonic**	**Description**		
7:0	SETUP_ID2(EISA)	Setup Identification byte 2 is 89h.		
Note: The "z" in the address is the bus slot number.				

EISA Setup Mode ID Byte 3 Register				
0zC82 (R)	SETUP_ID3, EISA			
	VGA	-	-	Mach 32
Bit	**Mnemonic**	**Description**		
7:0	SETUP_ID3(EISA)	Setup Identification byte 3 is 44h.		
Note: The "z" in the address is the bus slot number.				

EISA Setup Mode ID Byte 4 Register					
0zC83 (R)	**SETUP_ID4, EISA**				
	VGA	**-**	**-**	**Mach 32**	
Bit	**Mnemonic**	**Description**			
7:0	SETUP_ID4(EISA)	Setup Identification byte 4 is as follows: ATI68800-3 is 00h ATI68800LX is 01h ATI68800-6: 01h ATI68800AX: 00h.			
Note: The "z" in the address is the bus slot number.					

EISA Setup Mode Option Select Register					
0zC84 (R/W)	**SETUP_OPT, EISA**				
	VGA	**-**	**-**	**Mach 32**	
Bit	**Mnemonic**	**Description**			
0	SETUP_OPT(uC)	**0** = Video adapter disabled — responds only to POS registers. **1** = Video adapter enabled			
7:1	-	Reserved			
Note: The "z" in the address is the bus slot number.					

EISA ROM Base Address Setup Register					
0zC85 (W)	**ROM_SETUP, EISA**				
	VGA	**-**	**-**	**Mach 32**	
Bit	**Mnemonic**	**Description**			
6:0	-	Setting of ROM base address corresponding to CPU addresses A17:A11, in 8514 only mode.			
7	-	**0** = BIOS ROM decodes disabled **1** = BIOS ROM decodes enabled			
Note: The "z" in the address is the bus slot number.					

EISA Setup Byte 1 Register				
0zC86 (W)	SETUP_1, EISA			
	VGA	-	-	Mach 32
Bit	**Mnemonic**	**Description**		
7:0	-	General purpose R/W register. The value of this register is read back at 52EEh.		
Note: The "z" in the address is the bus slot number.				

EISA Setup Byte 2 Register				
0zC87 (W)	SETUP_2, EISA			
	VGA	-	-	Mach 32
Bit	**Mnemonic**	**Description**		
7:0	-	General purpose R/W register. The value of this register is read back at 52EFh.		
Note: The "z" in the address is the bus slot number.				

PCI Configuration Registers

The ATI68800AX ***mach32*** accelerator supports the Intel Peripheral Component Interconnect (PCI) local bus. Please refer to the *PCI Local Bus Specification* for detailed descriptions of the PCI Configuration Space registers. A brief summary is listed as follows:

Address	Bits	R/W	Functions	Power Up Default
0h	15:0	R	Vendor ID	1002
2h	15:0	R	Device ID	4158 (AX)
4h	0	R/W	I/O Access Enable	0 (Disabled)
	1	R/W	Memory Access Enable	0 (Disabled)
	2	R	Bus Master Enable	0 Always (Disabled)
	3	R	Special Cycle Enable	0 Always (Disabled)
	4	R	Mem. Write & Invalidate Enable	0 Always (Disabled)
	5	R/W	Palette Snooping Enable	0 (Disabled)
	6	R	Parity Error Enable	0 Always (Disabled)
	7	R/W	Read Wait Cycle Control	1 (Add 1WS)
	8	R	SERR# Enable	0 Always (Disabled)
	15:9	-	Reserved	0
6h	8:0	-	Reserved	0
	10:9	R	DEVSEL Timing	1 (Medium)
	11	R/W	Signalled Target Abort	0 (No Target Abort)
	12	R	Received Target Abort	0 Always (Inactive)
	13	R	Received Master Abort	0 Always (Inactive)
	14	R	Signalled System Error	0 Always (Inactive)
	15	R	Parity Error Detected	0 Always (Inactive)
8h	7:0	R	Revised ID	0
9h	7:0	R	Register-Level Prog. Interface	0
Ah	7:0	R	Sub-Class Code/Programmable Interface	00 (VGA Compat. Device) 80 (VGA Disabled)
Bh	7:0	R	Base-Class Code	03 (Display Controller)
Ch	7:0	R	Cache Line Size	0 (Not Used)
Dh	7:0	R	Latency Timer	0 (Not Used)
Eh	7:0	R	Header Type	0

Address	Bits	R/W	Functions	Power Up Default
Fh	7:0	R	BIST	0 (Not Used)
10h	21:0	R	Mem. Aperture Base Address	0
	31:22	R/W		
2F:14h	-	-	Reserved	0 Always
30h	0	R/W	BIOS ROM Enable	0
	7:1	R	Reserved	0 Always
31h	7:0	R	Reserved	0 Always
32h	15:0	R/W	BIOS ROM Base Address	0
3B:34h	-	-	Reserved	0
3Ch	7:0	R/W	Interrupt Line	0
3Dh	7:0	R	Interrupt Pin	1
3F:3Eh	-	-	Reserved	0
FF:40h	-	-	Reserved	0

This page intentionally left blank.

Part II

3

VGA Controller

Overview

The VGA Controller registers are completely hardware-compatible with the registers of the IBM Video Graphics Array (VGA) adapter. They are also fully hardware-register-compatible with IBM's Enhanced Graphics Adapter (EGA), Color Graphics Adapter (CGA), and Monochrome Display (MDA) adapters, as well as the Hercules graphics adapters. In addition, it provides a full set of Extended registers for enhanced features and performance.

The ATI controller is not only fast, it is also capable of displaying colors and resolutions beyond VGA in interlaced and non-interlaced modes. It supports 8-/16-bit ROM and I/O operations, 8-/16-bit video RAM data transfers, and zero wait-state BIOS and video memory operations.

This chapter provides specific details on each mode — resolution and color support, horizontal/vertical sync and polarities, pixel clock rates, and interlacing (all modes are non-interlaced unless otherwise indicated).

Integrated within the VGA portion of the ***mach 32*** are the following VGA functional blocks:

- Address Decoder
- Sequencer Controller
- CRT Controller
- Graphics Controller
- Attribute Controller.

IBM-Compatible Modes:

• MDA	80x25	(monochrome text)
• CGA	40x25	(16-color text)
	80x25	(16-color text)
	320x200	(2 sets of 4-color graphics)
	640x200	(2-color graphics)
• EGA	40x25	(16/64 color text)
	80x25	(2/ or 16/64 color text)
	320x200	(4/ or 16/64 color graphics)
	640x200	(2/ or 16/64 color graphics)
	640x350	(2/ or 16/64 color graphics)
• VGA	40x25	(16/256K color text)
	80x25	(2/ or 16/256K color text)
	320x200	(4/, 16/, or 256/256K color graphics)
	640x200	(2/, 4/, or 16/256K color graphics)
	640x350	(2/ or 16/256K color graphics)
	640x480	(2/ or 16/256K color graphics)

High Resolution and Wide-Column Graphics/Text Modes:

• Graphics Modes		
	640x400	(mono, Hercules compatible)
	720x348	(mono, Hercules compatible)
	640x480	16 or 256*/256K)
	800x600	(16/ or 256*/256K)
	1024x768	(16* or 256**/256K color)
• Text Modes		
	132x25*	(2 or 16/64 color)
	132x44*	(2 or 16 color)

* Requires 512KB video memory

** Requires 1MB video memory

Functional Blocks

The **Mach32** has five major functional blocks, illustrated below, including the control and data paths. Also shown are the connections to the video memory and video DAC. Data and address enter via the CPU bus on the left side. Video output in the form of RGB analog signals is available at the video DAC on the right side. The five functional blocks of this chip are described in detail as follows:

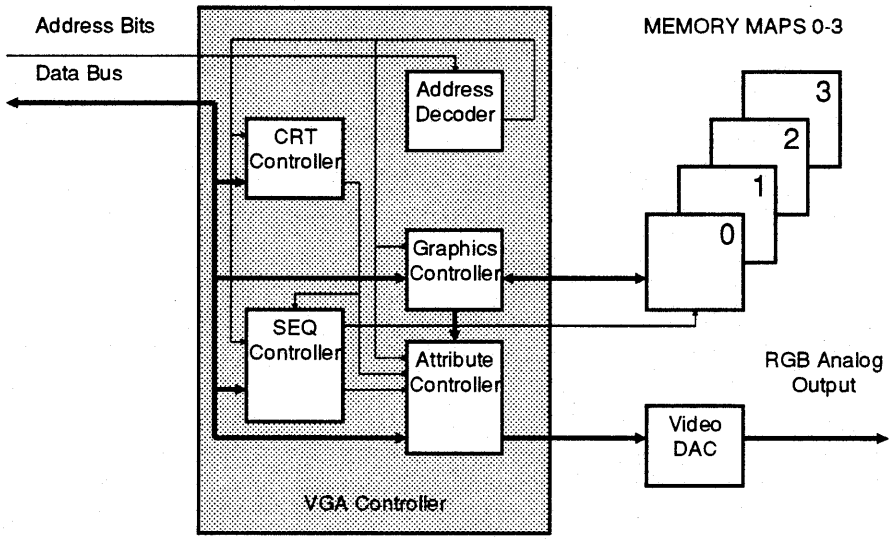


Figure 3-1. VGA Functional Block Diagram

Address Decoder

The address decoder is the top-level interface between the CPU and the Controller. Addresses and data from the input bus are decoded for each module. The address decoder also incorporates PS/2 interface circuitry. The starting address of video memory is programmable as VGA/EGA, MDA, or CGA, for 100% adapter compatibility.

Sequencer Controller

The sequencer controller generates all timing signals for the video RAMs and all control signals for the other modules within the Controller. It prioritizes CRT and CPU accesses to the video memory.

Video memory is protected from alteration by selectively masking out CPU writes to memory planes through the map mask register (also called the plane mask register).

CRT Controller

The CRT controller generates horizontal and vertical sync signals for the monitor interface, timings for cursor and underline attributes, timings for addressing the regenerative display buffer, and timings for refreshing the video RAM. The CRT controller registers are user programmable — for controlling of the display screen size, cursor type and position, character size, split screen, byte panning, and smooth scrolling.

Graphics Controller

The graphics controller handles data transfer between the CPU and video memory with different types of pixel/data mappings for read and write operations. It is also the interface for latching display data from video memory to the attribute controller during the display cycles. Video memory data is sent to the attribute controller as 8-bit parallel data on alphanumeric modes, but as converted serial bit-plane data in graphics modes.

The graphics controller supports 8- and 16-bit bus operations. It also provides color comparators for applications such as color filling and boundary detection.

Attribute Controller

The attribute controller receives data from the graphics controller. In text modes, it generates video signals using the character generator and the attribute code. In graphics modes, it formats the video data into either a 1-, 2-, 4-, or 8-pixel streams, as required by the selected mode. In either case, the video data is then passed through the internal 16/64 color palette registers and further processed to select a color value from the color register.

The output of the color register is then converted by a Digital to Analog Converter (DAC) to signals of the three primary colors, to drive a display device. The attribute controller also supports logics for blinking, underline, and horizontal pixel panning.

VGA Display Modes

The ***mach32*** chip supports alphanumeric or graphics display modes. Modes 0 to 6 are emulations of CGA modes. (Modes 0, 2, and 4 are identical to modes 1, 3, and 5 except that on the CGA adapter, modes 0, 2, and 4 have color burst turned off — Color burst is not supported in the ATI 68800 controller.)

Modes with an asterisk (*) following their mode numbers are enhanced EGA modes, namely 0* to 3*. Modes with a plus symbol ($^{+}$) are enhanced VGA modes. They are 0^{+}, 1^{+}, 2^{+}, 3^{+}, and 7^{+}. These and other modes are fully described starting on page 3-6. Descriptions include character box size, screen resolution, and color/palettes.

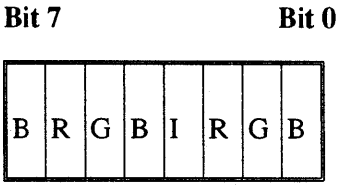
VGA Alphanumeric Modes (A/N)

This section describes the supported A/N modes: IBM compatible modes 0, 1, 2, 3, 7, and ATI extended modes 23, 27, 33, and 37. Mode numbers are expressed as hexadecimal numbers.

In A/N mode, the CPU transfers the character code for that mode into map 0, and attribute data into map 1. The CPU also transfers from ROM character patterns for that mode into map 2. Data from maps 0 and 1 are combined one byte at a time and read by the CRT controller. The CRTC then addresses the character generators in map 2. Dot patterns generated there are sent to the palette register where a color value is assigned. According to this value, the DAC then produces the three RGB analog signals to drive the display.

The character byte describes the displayed character; the attribute byte describes color, intensity, etc. The four most significant bits (MSB) of the attribute byte define the background; the other four bits describe the foreground (character). Bit 3 of the attribute byte may also be used together with two 3-bit pointers in the Character Map Select register to produce a total of 512 characters, and two separate character sets. If the video palette is changed, different colors will be generated. The attribute byte description is as follows:

BIT	"1"	"0"
7	Foreground Blinking*	Background Highlighted
6	Background Red on	Background Red off
5	Background Green on	Background Green off
4	Background Blue on	Background Blue off
3	Foreground Highlighted* Enable SEQ03[5,3,2]	Foreground Normal Enable SEQ03[4,1,0]
2	Foreground Red on	Foreground Red off
1	Foreground Green on	Foreground Green off
0	Foreground Blue on	Foreground Blue off



* Default value on mode is set by BIOS. SEQ03[xx] refers to bits xx of the SEQ03 register in the VGA graphics controller.

Each character on the screen is defined using two bytes of read/write memory. For example, on a 40x25 character page, a buffer memory of 2,000 bytes is required. If each of the 40x25 characters has a resolution (box size) of 9x16 pixels, the page has a resolution of 360x400.

Video data in A/N modes is addressed one character at a time. See tables below for descriptions of each supported mode.

Modes 0+ and 1+

STANDARD	BOX SIZE	CHARxROW	COLORS
VGA	9x16	40x25	16/256K

Modes 0* and 1*

STANDARD	BOX SIZE	CHARxROW	COLORS
EGA	8x14	40x25	16/64

Modes 0 and 1

STANDARD	BOX SIZE	CHARxROW	COLORS
CGA	8x8	40x25	16

Modes 2+ and 3+

STANDARD	BOX SIZE	CHARxROW	COLORS
VGA	9x16	80x25	16/256K

Modes 2* and 3*

STANDARD	BOX SIZE	CHARxROW	COLORS
EGA	8x14	80x25	16/64

Modes 2 and 3

STANDARD	BOX SIZE	CHARxROW	COLORS
CGA	8x8	80x25	16

Mode 7+

STANDARD	BOX SIZE	CHARxROW	COLORS
VGA	9x16	80x25	2/256K

Mode 7

STANDARD	BOX SIZE	CHARxROW	COLORS
EGA	9x14	80x25	2/64
MDA	9x14	80x25	Mono

Modes 23, 27, 33, and 37

STANDARD	BOX SIZE	CHARxROW	COLORS
23	8X16	132x25	16/64
27	8X16	132x25	Mono
33	8X8	132x44	16
37	8X8	132x44	Mono

VGA Graphics Modes (APA)

This section describes the supported APA modes: IBM compatible modes 4, 5, 6, D, E, F, 10, 11, 12, 13; Hercules compatible Hercu1 and Hercu2; and ATI extended modes 54, 55, 62, 63, and 64. Mode numbers are expressed in hexadecimal numbers.

Modes 4 and 5

STANDARD	PELS	COLORS
VGA	320x200	4/256K
EGA	320x200	4/64
CGA	320x200	4 (2 sets)

Mode 6

STANDARD	PELS	COLORS
VGA	640x200	2/256K
EGA	640x200	2/64
CGA	640x200	2

Mode D

STANDARD	PELS	COLORS
VGA	320x200	16/256K
EGA	320x200	16/64

Mode E

STANDARD	PELS	COLORS
VGA	640x200	16/256K
EGA	640x200	16/64

Mode F

STANDARD	PELS	COLORS
VGA	640x350	2/256K
EGA	640x350	2/64

Mode 10

STANDARD	PELS	COLORS
VGA	640x350	16/256K
EGA	640x350	16/64

Mode 11

STANDARD	PELS	COLORS
VGA	640x480	2/256K

Mode 12

STANDARD	PELS	COLORS
VGA	640X480	16/256k

Mode 13

STANDARD	PELS	COLORS
VGA	320x200	256/256K (mono - 64 shades of gray)

Mode Hercu1

STANDARD	PELS	COLORS
Hercu1	720X348	Mono

Mode Hercu2

STANDARD	PELS	COLORS
Hercu2	640X400	Mono

Mode 54

STANDARD	PELS	COLORS
54	800X600	16/256K

Mode 55

STANDARD	PELS	COLORS
55	1024X768	16/256K

Mode 62

STANDARD	PELS	COLORS
62	640X480	256/256K

Mode 63

STANDARD	PELS	COLORS
63	800X600	256/256K

Mode 64

STANDARD	PELS	COLORS
64	1024X768	256/256K

Display Mode Specifications - PS/2 (Analog) Monitors

MODE	BOX SIZE	PELS	COLORS	HOR. SYNC (KHz)	VERT. SYNC (Hz)	H/V POLAR-ITIES	DOT CLOCK (MHz)
0+/VGA	9x16	360x400	16/256K	31.47	70	-/+	28.3
1+/VGA	9x16	360x400	16/256K	31.47	70	-/+	28.3
0*/EGA	8x14	320x350	16/64	31.47	70	+/-	25.2
1*/EGA	8x14	320x350	16/64	31.47	70	+/-	25.2
0/CGA	8x8	320x200	16	31.47	70	-/+	25.2
1/CGA	8x8	320x200	16	31.47	70	-/+	25.2
2+/VGA	9x16	720x400	16/256K	31.47	70	-/+	28.3
3+/VGA	9x16	720x400	16/256K	31.47	70	-/+	28.3
2*/EGA	8x14	640x350	16/64	31.47	70	+/-	25.2
3*/EGA	8x14	640x350	16/64	31.47	70	+/-	25.2
2/CGA	8x8	640x200	16	31.47	70	-/+	25.2
3/CGA	8x8	640x200	16	31.47	70	-/+	25.2
4/VGA	8x8	320x200	4/256K	31.47	70	-/+	25.2
4/EGA	8x8	320x200	4/64	31.47	70	-/+	25.2
4/CGA	8x8	320x200	4 (2 sets)	31.47	70	-/+	25.2
5/VGA	8x8	320x200	4/256K	31.47	70	-/+	25.2
5/EGA	8x8	320x200	4/64	31.47	70	-/+	25.2
5/CGA	8x8	320x200	4 (2 sets)	31.47	70	-/+	25.2
6/VGA	8x8	640x200	2/256K	31.47	70	-/+	25.2
6/EGA	8x8	640x200	2/64	31.47	70	-/+	25.2
6/CGA	8x8	640x200	2	31.47	70	-/+	25.2
7+/VGA	9x16	720x400	2/256K	31.47	70	-/+	28.3
7/EGA	9x14	720x350	2/64	31.47	70	+/-	28.3
7/MDA	9x14	720x350	Mono	31.47	70	+/-	28.3
D/VGA	8x8	320x200	16/256K	31.47	70	-/+	25.2
D/EGA	8x8	320x200	16/64	31.47	70	-/+	25.2
E/VGA	8x8	640x200	16/256K	31.47	70	-/+	25.2
E/EGA	8x8	640x200	16/64	31.47	70	-/+	25.2
F/VGA	8x14	640x350	2/256K	31.47	70	+/-	25.2
F/EGA	8x14	640x350	2/64	31.47	70	+/-	25.2
10/VGA	8x14	640x350	16/256K	31.47	70	+/-	25.2
10/EGA	8x14	640x350	16/64	31.47	70	+/-	25.2
11/VGA	8x16	640x480	2/256K	31.47	60	-/-	25.2
12/VGA	8x16	640x480	16/256K	31.47	60	-/-	25.2
12/VGA	8x16	640x480	16/256K	37.73	72.15	-/-	32
13/VGA	8x8	320x200	256/256K	31.47	70	-/+	25.2

PS/2 (Analog) Monitors (continued)

MODE	BOX SIZE	PELS	COLORS	HOR. SYNC (KHz)	VERT. SYNC (Hz)	H/V POLAR-ITIES	DOT CLOCK (MHz)
Hercu1	-	720x348	Mono	31.47	70	+/-	28.3
Hercu2	-	640x400	Mono	31.47	70	-/+	25.2
23	8x16	1056x400	16/64	31.47	70	-/+	40
27	8x16	1056x400	Mono	31.47	70	-/+	40
33	8x8	1056x352	16	31.47	70	-/+	40
37	8x8	1056x352	Mono	31.47	70	-/+	40
55[I]	8x16	1024x768	16/256K	35.52	86	+/+	45
55	8x16	1024x768	16/256K	48.36	60	-/-	65
55	8x16	1024x768	16/256K	56.47	70	-/-	75
55	8x16	1024x768	16/256K	57.87	72	-/-	75
62	8x16	640x480	256/256K	31.47	60	-/-	25
62	8x16	640x480	256/256K	72.15	72	-/-	32
64[I]	8x16	1024x768	256/256K	35.52	87	+/+	45
64	8x16	1024x768	256/256K	48.36	60	+/+	65
64	8x16	1024x768	256/256K	56.48	70	-/-	75
64	8x16	1024x768	256/256K	57.87	72	-/-	75

NOTE:

I = Interlaced mode, 8514/A monitors only

Display Mode Specifications - Multisync Monitors

MODE	BOX SIZE	PELS	COLORS	HOR. SYNC (KHz)	VERT. SYNC (Hz)	H/V POLAR-ITIES	DOT CLOCK (MHz)
0+/VGA	9x16	360x400	16/256K	31.47	70	-/+	28.3
1+/VGA	9x16	360x400	16/256K	31.47	70	-/+	28.3
0*/EGA	8x14	320x350	16/64	31.47	70	+/-	25.2
1*/EGA	8x14	320x350	16/64	31.47	70	+/-	25.2
0/CGA	8x8	320x200	16	31.47	70	-/+	25.2
1/CGA	8x8	320x200	16	31.47	70	-/+	25.2
2+/VGA	9x16	720x400	16/256K	31.47	70	-/+	28.3
3+/VGA	9x16	720x400	16/256K	31.47	70	-/+	28.3
2*/EGA	8x14	640x350	16/64	31.47	70	+/-	25.2
3*/EGA	8x14	640x350	16/64	31.47	70	+/-	25.2
2/CGA	8x8	640x200	16	31.47	70	-/+	25.2
3/CGA	8x8	640x200	16	31.47	70	-/+	25.2
4/VGA	8x8	320x200	4/256K	31.47	70	-/+	25.2
4/EGA	8x8	320x200	4/64	31.47	70	-/+	25.2
4/CGA	8x8	320x200	4 (2 sets)	31.47	70	-/+	25.2
5/VGA	8x8	320x200	4/256K	31.47	70	-/+	25.2
5/EGA	8x8	320x200	4/64	31.47	70	-/+	25.2
5/CGA	8x8	320x200	4 (2 sets)	31.47	70	-/+	25.2
6/VGA	8x8	640x200	2/256K	31.47	70	-/+	25.2
6/EGA	8x8	640x200	2/64	31.47	70	-/+	25.2
6/CGA	8x8	640x200	2	31.47	70	-/+	25.2
7+/VGA	9x16	720x400	2/256K	31.47	70	-/+	28.3
7/EGA	9x14	720x350	2/64	31.47	70	+/-	28.3
7/MDA	9x14	720x350	Mono	31.47	70	+/-	28.3
D/VGA	8x8	320x200	16/256K	31.47	70	-/+	25.2
D/EGA	8x8	320x200	16/64	31.47	70	-/+	25.2
E/VGA	8x8	640x200	16/256K	31.47	70	-/+	25.2
E/EGA	8x8	640x200	16/64	31.47	70	-/+	25.2
F/VGA	8x14	640x350	2/256K	31.47	70	+/-	25.2
F/EGA	8x14	640x350	2/64	31.47	70	+/-	25.2
10/VGA	8x14	640x350	16/256K	31.47	70	+/-	25.2
10/EGA	8x14	640x350	16/64	31.47	70	+/-	25.2
11/VGA	8x16	640x480	2/256K	31.47	60	-/-	25.2
12/VGA	8x16	640x480	16/256K	31.47	60	-/-	25.2
12/VGA	8x16	640x480	16/256K	37.73	72.15	-/-	32
13/VGA	8x8	320x200	256/256K	31.47	70	-/+	25.2

Multisync Monitors (continued)

MODE	BOX SIZE	PELS	COLORS	HOR. SYNC (KHz)	VERT. SYNC (Hz)	H/V POLAR-ITIES	DOT CLOCK (MHz)
Hercu1	-	720x348	Mono	31.47	70	+/-	28.3
Hercu2	-	640x400	Mono	31.47	70	-/+	25.2
23	8x16	1056x400	16/64	31.47	70	-/+	40
27	8x16	1056x400	Mono	31.47	70	-/+	40
33	8x8	1056x352	16	31.47	70	-/+	40
37	8x8	1056x352	Mono	31.47	70	-/+	40
54	8x14	800x600	16/256K	35.16	56	-/-	36
54V	8x14	800x600	16/256K	35.16	56	+/+	36
54	8x14	800x600	16/256K	37.87	60	+/+	40
54	8x14	800x600	16/256K	44.19	70	+/+	45
55I	8x16	1024x768	16/256K	35.52	86	+/+	45
55	8x16	1024x768	16/256K	48.36	60	-/-	65
55	8x16	1024x768	16/256K	56.47	70	-/-	75
55	8x16	1024x768	16/256K	57.87	72	-/-	75
62	8x16	640x480	256/256K	31.47	60	-/-	25
62	8x16	640x480	256/256K	72.15	72	-/-	32
63	8x14	800x600	256/256K	35.16	56	-/-	36
63V	8x14	800x600	256/256K	35.16	56	+/+	36
63	8x14	800x600	256/256K	37.88	60	+/+	40
63	8x14	800x600	256/256K	48.04	72	+/+	50
64I	8x16	1024x768	256/256K	35.52	87	+/+	45
64	8x16	1024x768	256/256K	48.36	60	+/+	65
64	8x16	1024x768	256/256K	56.48	70	-/-	75
64	8x16	1024x768	256/256K	57.87	72	-/-	75

NOTES:

I = Interlaced mode

V = Applicable to VESA compatible monitors.

This page intentionally left blank.

4

VGA Memory Organization

Overview

The base address of video display buffers in the ***mach32*** graphics controller is configurable as required by the emulated video standard. The size of this buffer depends on the selected display mode — higher resolution requires a larger memory buffer.

STD	SIZE	MEM SEGMENT BASE ADDRESS
VGA/EGA	128k	B8000 (Color Text) B0000 (Mono Text) A0000 (Graphics)
CGA	32k	B8000 (Color Text & Graphics)
MDA	32k	B0000 (Mono Text)
Hercules	32K	B0000, B8000 (Mono Graphics)

Memory organization affects how the video data of each supported display mode is to be written or read. A/N mode is alphanumeric (text); APA mode is All-Planes-Addressable (graphics). Modes 0 - 13 are 100% compatible with the modes available in the various IBM display adapters. Hercu1 and Hercu2 are compatible with Hercules Graphics adapters. All other modes are extended modes, supported by ATI, to provide users with enhanced display speed and better screen resolutions.

The display buffer can be stored as *page* or *map* memory. Page memory access is reading and writing the video data one map (byte) at a time. Map memory access is applying the same CPU address to multiple maps (parallel locations) of video memory for data on one pixel — each map contributes its portion of color/attribute specifications. The processing of map data uses a set of four 8-bit latches which correspond to the maps. Mode 13 uses a type of page memory access called packed pixel format.

Memory Maps

The size of each memory buffer shown as blocks in the illustration is shown in KBs. The total amount to be reserved for a full screen is marked by arrows beside the blocks.

For example, in display mode 0, the buffer size is 2KB and the actual amount required for a full screen is 2000 bytes. The base address of the video buffer can start at either B8000, B8800, or B9000, etc. Mode 0 uses two bytes to describe each pixel, as follows — the first byte of the buffer is for character code, the second byte is for attribute, the third byte is for character code, and the forth byte is for attribute, and so on.

When a display mode requires more than 64KB of data to paint a full screen, as in display mode 62, see page 23, then the display buffer will map to multiple pages. Graphically these pages are shown as **m_page** blocks in the illustrations. For this example, the buffer has mapped to five m_pages, (for a total of 307,200 actual data bytes) to support one full screen.

The following table lists page numbers for the VGA memory maps described in this chapter.

A/N Modes 0 and 1 (16-color 40x25)

- Page memory. For explanation see *Memory Maps* on page 4-2.
- Two bytes per display character.
- Character/Attribute Format:

B_B = Blink / Intensity $_{Background}$
I_F = Intensity $_{Foreground}$ / Character Select
R, G, B = Color / Attribute

- First display character (upper-left corner of display) is at B8000/1, B8800/1, or B9000/1 etc. The display buffer is mapped as follows:

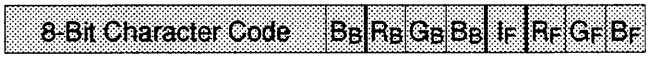
B800:0000 Page 0	B800:0001 Page 0
B880:0000 Page 1	B880:0001 Page 1
B900:0000 Page 2	B980:0001 Page 2
B980:0000 Page 3	B980:0001 Page 3
BA00:0000 Page 4	BA00:0001 Page 4
BA80:0000 Page 5	BA80:0001 Page 5
BB00:0000 Page 6	BB00:0001 Page 6
BB80:0000 Page 7	BB80:0001 Page 7
MAP 0 Character Code	MAP 1 Attribute Data

2KB (2,000 bytes)

Figure 4-1. Memory Organization - Modes 0, 1

A/N Modes 2 and 3 (16-color 80x25)

- Page memory. For explanation see *Memory Maps* on page 4-2.
- Two bytes per display character.
- Character/Attribute Format:



B_B = Blink / Intensity $_{Background}$
I_F = Intensity $_{Foreground}$ / Character Select
RGB = Color / Attribute

- First display character (upper-left corner of display) is at B8000/1, B9000/1, or BA000/1 etc. The display buffer is mapped as follows:

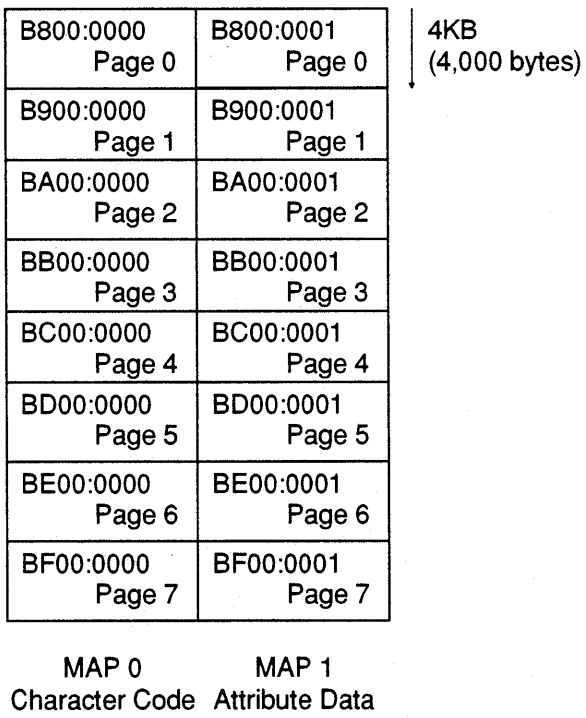


Figure 4-2. Memory Organization - Modes 2, 3

APA Modes 4 and 5 (4-color 320x200)

- Page memory. For explanation see *Memory Maps* on page 4-2.
- Four PELs per byte. First PEL (shaded) is in the MSB position.

C_1	C_0	C_1	C_0	C_1	C_0	C_1	C_0

  C_1, C_0 of the PEL specify one of four colors - selectable by a video BIOS call.

- Display buffer is organized into two blocks of 8KB each. First four PELs (upper-left corner of display) are at B8000. Odd scan lines are offset from even scan lines by 8K. It is mapped as follows:

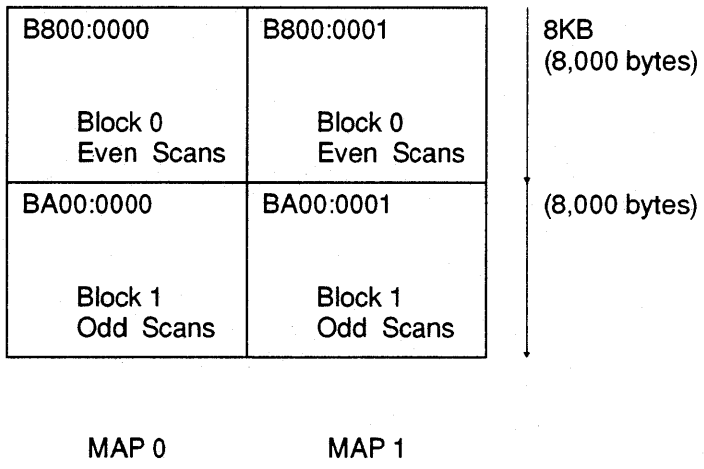


Figure 4-3. Memory Organization - Modes 4, 5

APA Mode 6 (2-color 640x200)

- Map memory. For explanation see *Memory Maps* on page 4-2.
- Eight PELs per byte. First PEL (shaded) is in the MSB position.

C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0

  C_0 specifies the PEL color.

- Display buffer is organized into two blocks of 8KB each. First eight PELs (upper-left corner of display) are at B8000. Odd scan lines are offset from even scan lines by 8K. It is mapped as follows:

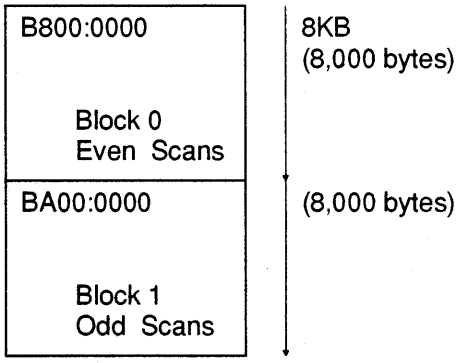


Figure 4-4. Memory Organization - Mode 6

A/N Mode 7 (Mono or 2-color 80x25)

- Page memory. For explanation see *Memory Maps* on page 4-2.
- Two bytes per display character.
- Character/Attribute Format:

8 Bit Character Code	B_B	R_B	G_B	B_B	I_F	R_F	G_F	B_F

B_B = Blink / Intensity $_{Background}$
I_F = Intensity $_{Foreground}$ / Character Select
RGB = Color / Attribute

- First display character (upper-left corner of display) is at B0000/1, B1000/1, or B2000/1 etc. The display buffer is mapped as follows:

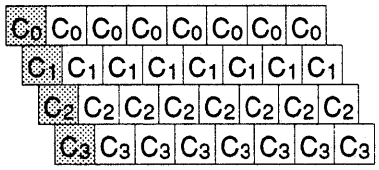
MAP 0 Character Code	MAP 1 Attribute Data
B000:0000 Page 0	B000:0001 Page 0
B100:0000 Page 1	B100:0001 Page 1
B200:0000 Page 2	B200:0001 Page 2
B300:0000 Page 3	B300:0001 Page 3
B400:0000 Page 4	B400:0001 Page 4
B500:0000 Page 5	B500:0001 Page 5
B600:0000 Page 6	B600:0001 Page 6
B700:0000 Page 7	B700:0001 Page 7

4KB
(4,000 bytes)

Figure 4-5. Memory Organization - Mode 7

APA Mode D (320x200)

- Map memory. For explanation see *Memory Maps* on page 4-2.
- Eight PELs per byte. First PEL (shaded) is in the MSB position.



C_0, C_1, C_2, C_3 from bit planes 0-3 specify the PEL color.

- First eight PELs (upper-left corner of display) are at A0000, A2000, or A4000 etc. The display buffer is mapped as follows:

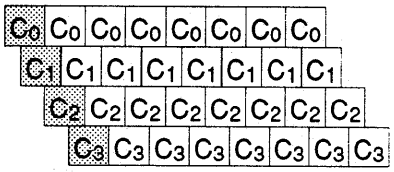
A000:0000 Page 0	A000:0000 Page 0	A000:0000 Page 0	A000:0000 Page 0
A200:0000 Page 1	A200:0000 Page 1	A200:0000 Page 1	A200:0000 Page 1
A400:0000 Page 2	A400:0000 Page 2	A400:0000 Page 2	A400:0000 Page 2
A600:0000 Page 3	A600:0000 Page 3	A600:0000 Page 3	A600:0000 Page 3
A800:0000 Page 4	A800:0000 Page 4	A800:0000 Page 4	A800:0000 Page 4
AA00:0000 Page 5	AA00:0000 Page 5	AA00:0000 Page 5	AA00:0000 Page 5
AC00:0000 Page 6	AC00:0000 Page 6	AC00:0000 Page 6	AC00:0000 Page 6
AE00:0000 Page 7	AE00:0000 Page 7	AE00:0000 Page 7	AE00:0000 Page 7
MAP 0 Blue	MAP 1 Red	MAP 2 Green	MAP 3 Intensity

↓ 8KB (8,000 bytes)

Figure 4-6. Memory Organization - Mode D

APA Mode E (640x200)

- Map memory. For explanation see *Memory Maps* on page 4-2.
- Eight PELs per byte. First PEL (shaded) is in the MSB position.



C_0, C_1, C_2, C_3 from bit planes 0-3 specify the PEL color.

- First eight PELs (upper-left corner of display) are at A0000, A4000, or A8000 etc. The display buffer is mapped as follows:

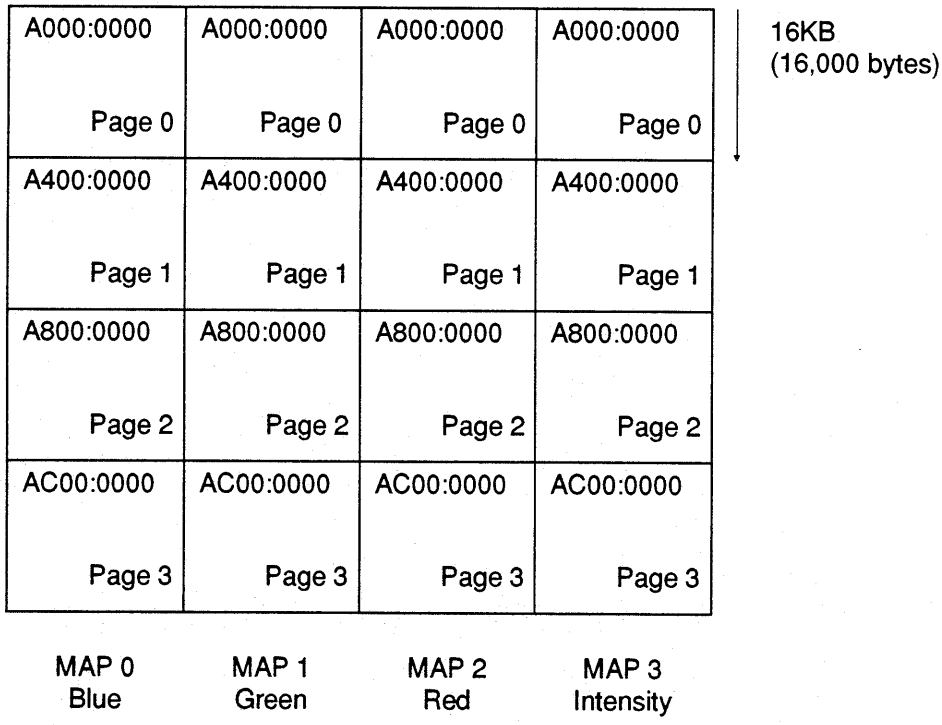
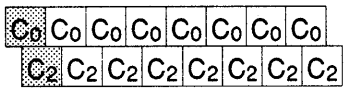


Figure 4-7. Memory Organization - Mode E

APA Mode F (640x350)

- Map memory. For explanation see *Memory Maps* on page 4-2.
- Eight PELs per byte. First PEL (shaded) is in the MSB position.



  C_0, C_2 from bit planes 0, 2 specify the PEL color.

- First eight PELs (upper-left corner of display) are at A0000 or A8000. The display buffer is mapped as follows:

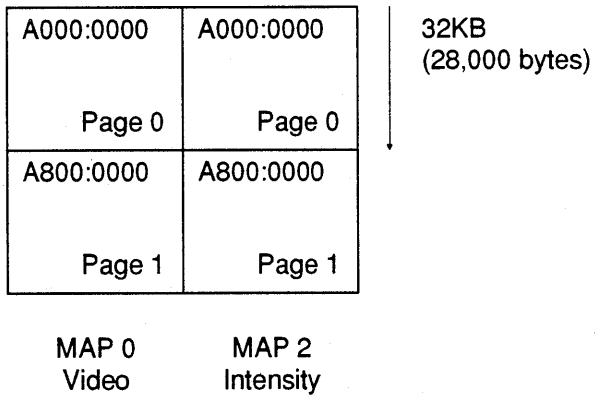
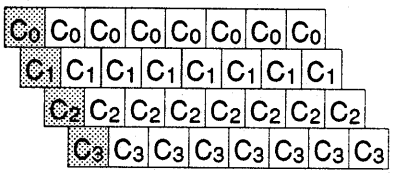


Figure 4-8. Memory Organization - Mode F

APA Mode 10 (640x350)

- Map memory. For explanation see *Memory Maps* on page 4-2.
- Eight PELs per byte. First PEL (shaded) is in the MSB position.



  C_0, C_1, C_2, C_3 from bit planes 0-3 specify the PEL color.

- First eight PELs (upper-left corner of display) are at A0000 or A8000. The display buffer is mapped as follows:

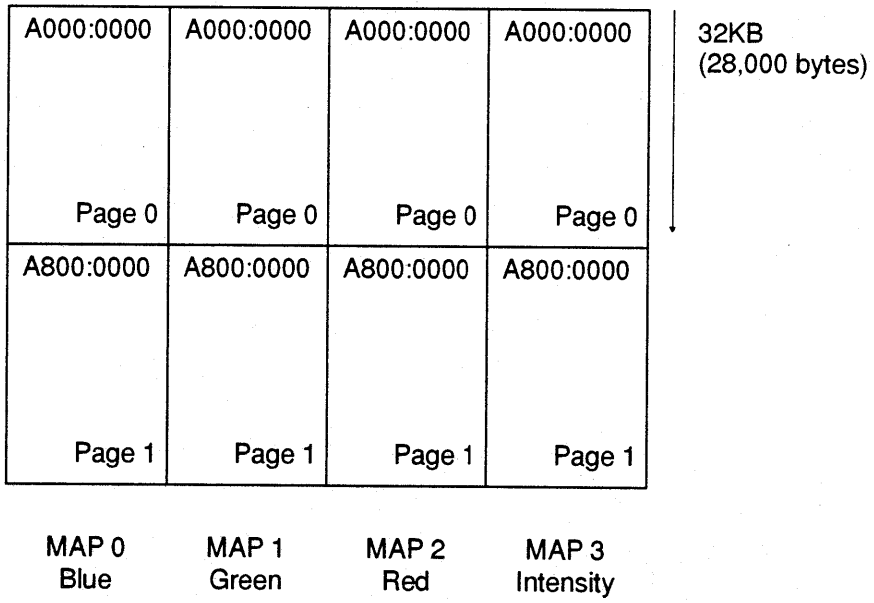


Figure 4-9. Memory Organization - Mode 10

APA Mode 11 (2-color 640x480)

- Page memory. For explanation see *Memory Maps* on page 4-2.
- Eight PELs per byte. First PEL (shaded) is in the MSB position.

C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0

  C_0 specifies the PEL color.
- First eight PELs (upper-left corner of display) are at A0000. The display buffer is mapped as follows:

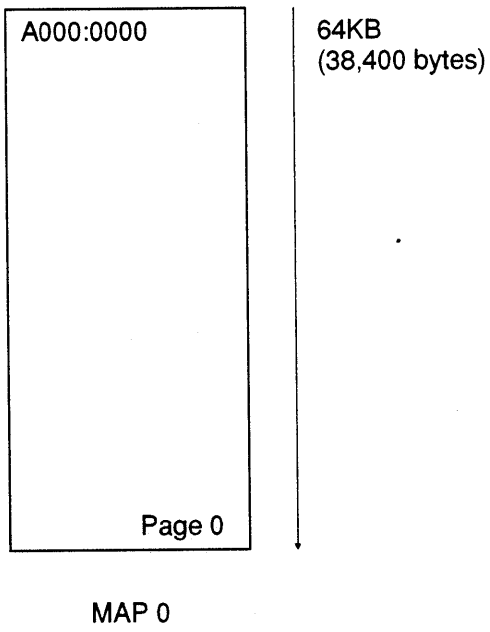
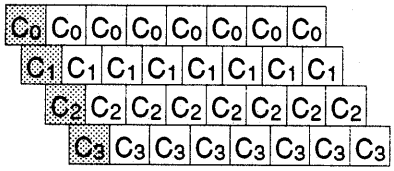


Figure 4-10. Memory Organization - Mode 11

APA Mode 12 (16-color 640x480)

- Map memory. For explanation see *Memory Maps* on page 4-2.
- Eight PELs per byte. First PEL (shaded) is in the MSB position.



C_0, C_1, C_2, C_3 from bit planes 0-3 specify the PEL color.

- First eight PELs (upper-left corner of display) are at A0000. The display buffer is mapped as follows:

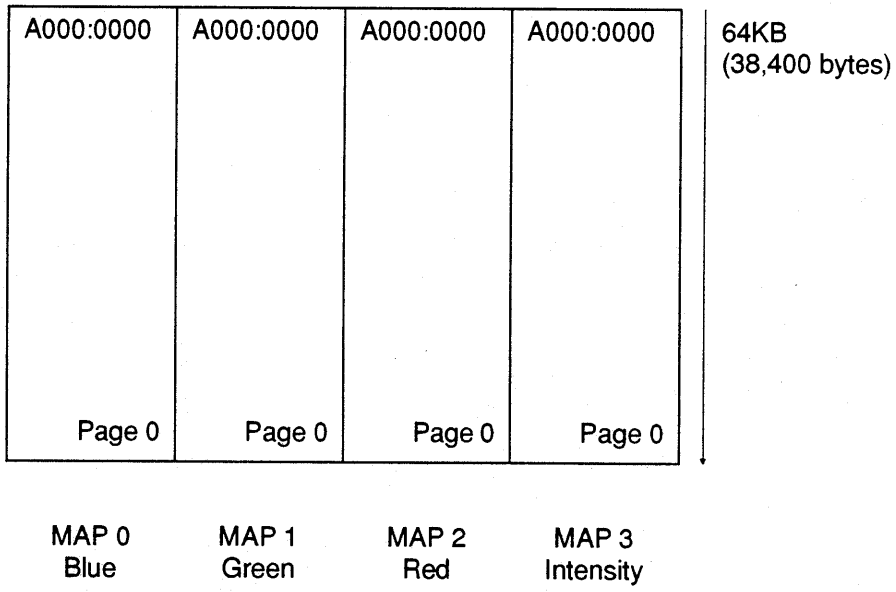


Figure 4-11. Memory Organization - Mode 12

APA Mode 13 (256-color 320x200)

- Page memory. For explanation see *Memory Maps* on page 4-2.
- One PEL per byte.

C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0

  C_{7-0} specify the PEL color

- Display buffer is linear. Bit image data is stored in four bit planes which are sampled twice to produce eight bit planes that address the video DAC. First PEL (upper-left corner of display) is at A0000, second at A0001. Video data is stored in four bit planes and mapped as follows:

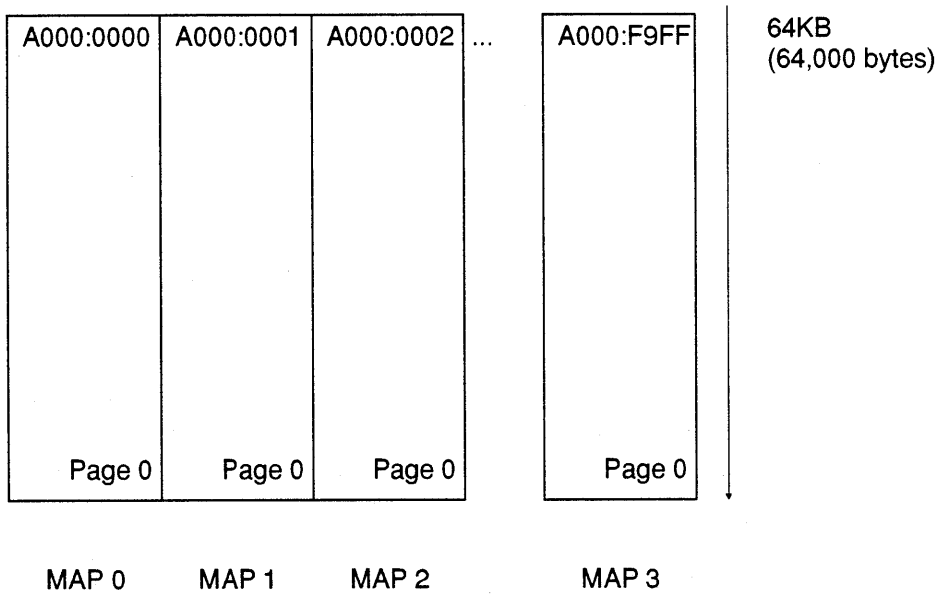


Figure 4-12. Memory Organization - Mode 13

APA Mode Hercu1 (720x348)

- Page memory. For explanation see *Memory Maps* on page 4-2.
- Eight PELs per byte. First PEL (shaded) is in the MSB position.

C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0

  C_0 specifies the PEL color.

- First eight PELs (upper-left corner of display) are at B0000 or B8000. The display buffer is mapped as follows:

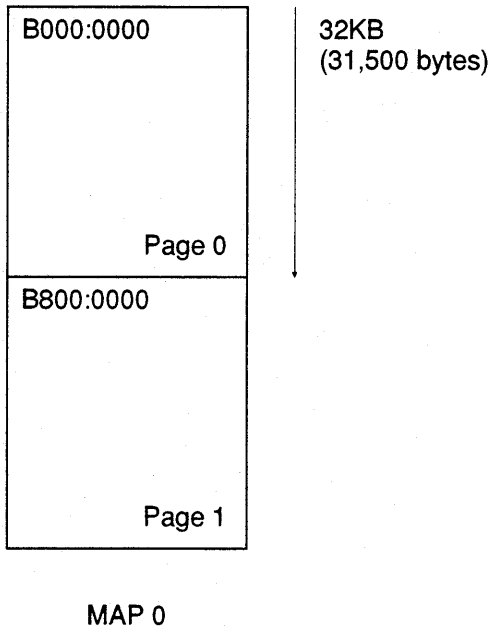


Figure 4-13. Memory Organization - Mode Hercu1

APA Mode Hercu2 (640x400)

- Page memory. For explanation see *Memory Maps* on page 4-2.
- Eight PELs per byte. First PEL (shaded) is in the MSB position.

C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0

  C_0 specifies the PEL color.

- First eight PELs (upper-left corner of display) are at B0000 or B8000. The display buffer is mapped as follows:

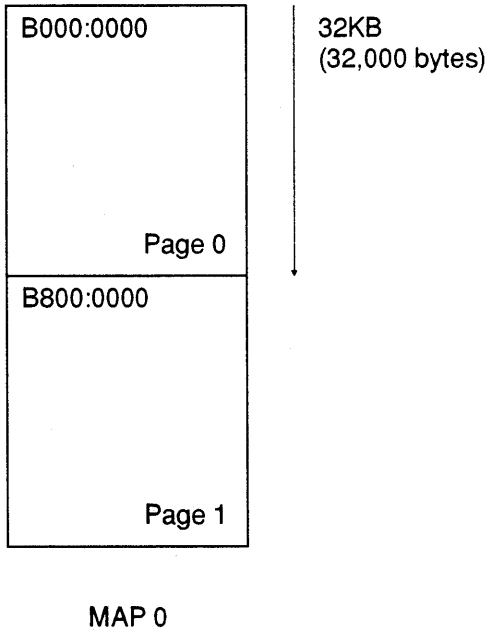


Figure 4-14. Memory Organization - Mode Hercu2

A/N Mode 23 (16-color 132x25)

- Page memory. For explanation see *Memory Maps* on page 4-2.
- Two bytes per display character.
- Character/Attribute Format:

B_B = Blink / Intensity $_{Background}$
I_F = Intensity $_{Foreground}$ / Character Select
RGB = Color / Attribute

- First display character (upper-left corner of display) is at B8000/1, BA000/1, or BC000/1 etc. The display buffer is mapped as follows:

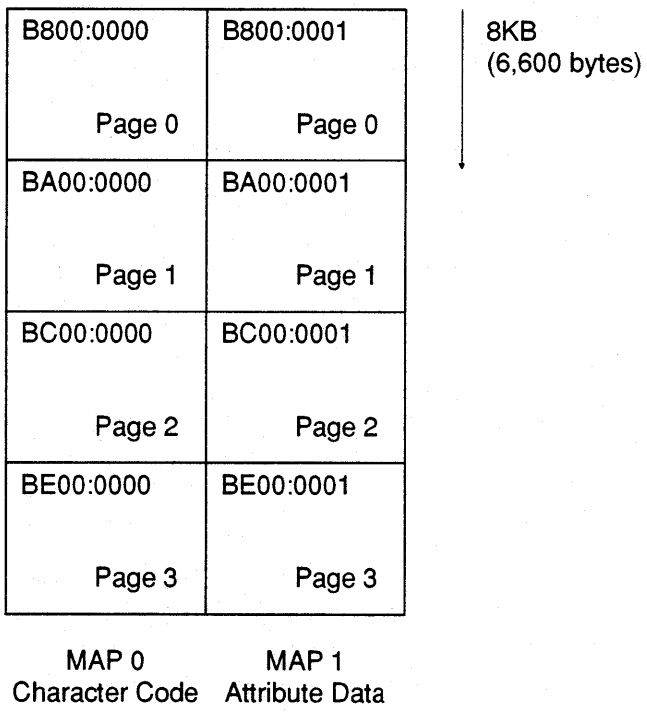


Figure 4-15. Memory Organization - Mode 23

A/N Mode 27 (Mono 132x25)

- Page memory. For explanation see *Memory Maps* on page 4-2.
- Two bytes per display character.
- Character/Attribute Format:

B_B = Blink / Intensity $_{Background}$
I_F = Intensity $_{Foreground}$ / Character Select
RGB = Color / Attribute

- First display character (upper-left corner of display) is at B0000/1, B2000/1, or B4000/1 etc. The display buffer is mapped as follows:

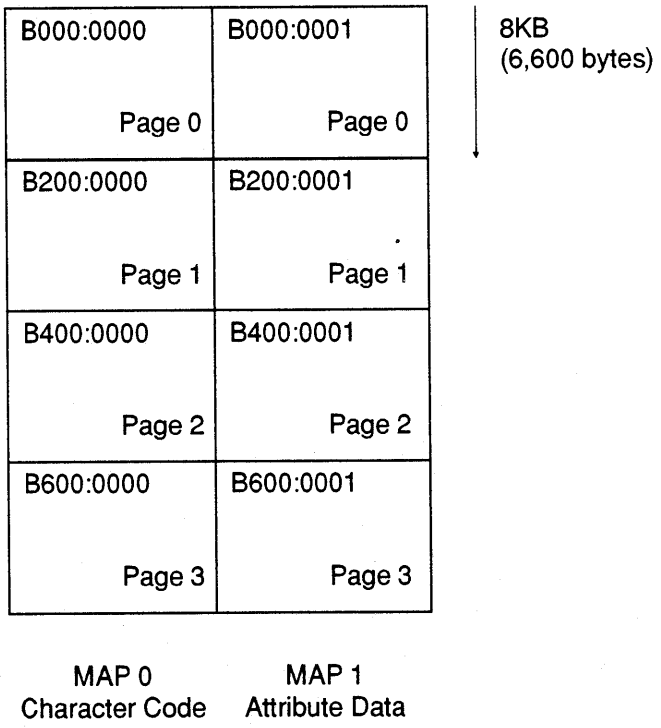


Figure 4-16. Memory Organization - Mode 27

A/N Mode 33 (16-color 132x44)

- Page memory. For explanation see *Memory Maps* on page 4-2.
- Two bytes per display character.
- Character/Attribute Format:

8-Bit Character Code	B_B	R_B	G_B	B_B	I_F	R_F	G_F	B_F

B_B = Blink / Intensity $_{Background}$
I_F = Intensity $_{Foreground}$ / Character Select
RGB = Color / Attribute

- First display character (upper-left corner of display) is at B8000/1 or BC000/1. The display buffer is mapped as follows:

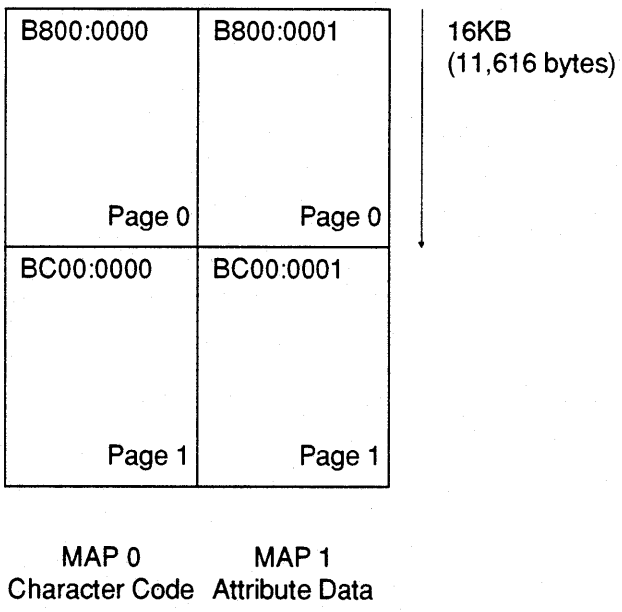


Figure 4-17. Memory Organization - Mode 33

A/N Mode 37 (Mono 132x44)

- Page memory. For explanation see *Memory Maps* on page 4-2.
- Two bytes per display character.
- Character/Attribute Format:

8 Bit Character Code	B_B	R_B	G_B	B_B	I_F	R_F	G_F	B_F

B_B = Blink / Intensity $_{Background}$
I_F = Intensity $_{Foreground}$ / Character Select
RGB = Color / Attribute

- First display character (upper-left corner of display) is at B0000/1 or B4000/1. The display buffer is mapped as follows:

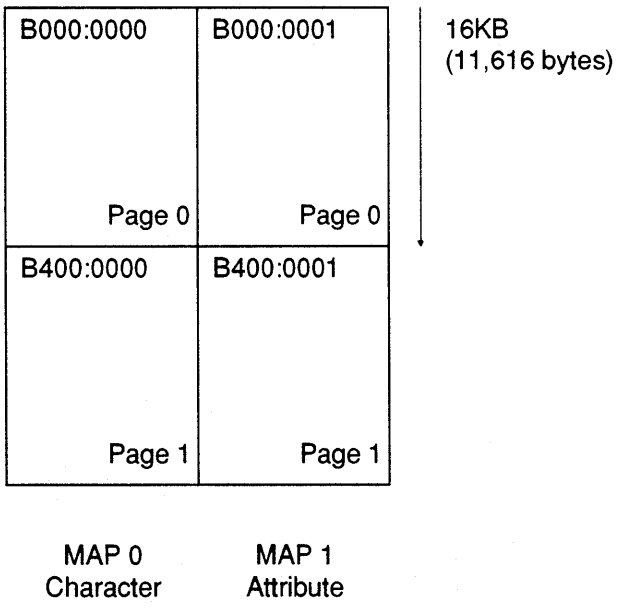
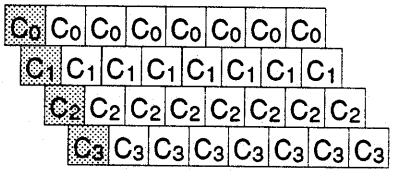


Figure 4-18. Memory Organization - Mode 37

APA Mode 54 (16-color 800x600)

- Map memory. For explanation see *Memory Maps* on page 4-2.
- Eight PELs per byte. First PEL (shaded) is in the MSB position.



  C_0, C_1, C_2, C_3 from bit planes 0-3 specify the PEL color.

- First eight PELs (upper-left corner of display) are at A0000. The display buffer (based on 256KB video memory configuration) is mapped as follows:

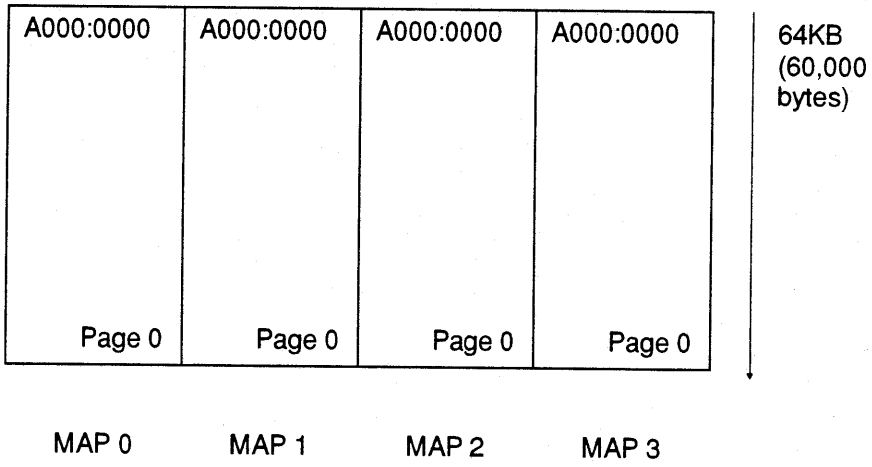
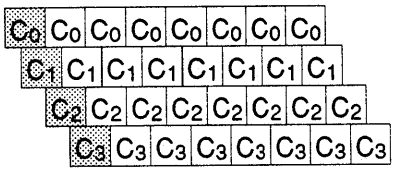


Figure 4-19. Memory Organization - Mode 54

APA Mode 55 (16-color 1024x768)

- Map memory (512KB required). For explanation see *Memory Maps* on page 4-2.
- Eight PELs per byte. First PEL (shaded) is in MSB position.



C_0, C_1, C_2, C_3 from bit planes 0-3 specify the PEL color.

- First eight PELs (upper-left corner of display) are at A0000. The display buffer for a full screen is mapped to two m_pages as follows:

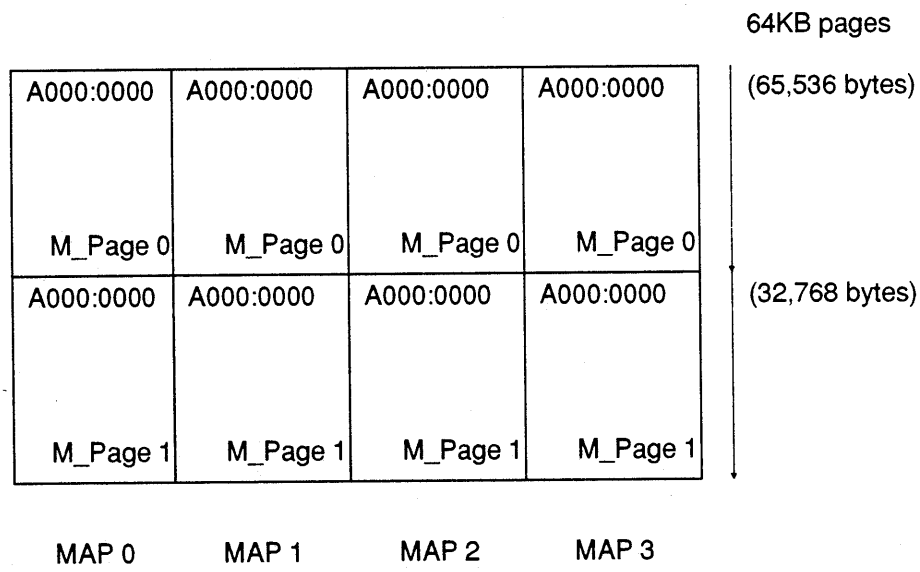


Figure 4-20. Memory Organization - Mode 55

APA Mode 62 (256-color 640x480)

- Page memory (512KB required). For explanation, see *Memory Maps* on page 4-2.
- One PEL per byte.

C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0

  C_{7-0} specify the PEL color.

- First PEL (upper-left corner of display) is at A0000, second at A0001. First 640 bytes form the first raster scan line. The display buffer for a full screen is mapped to five m_pages as follows:

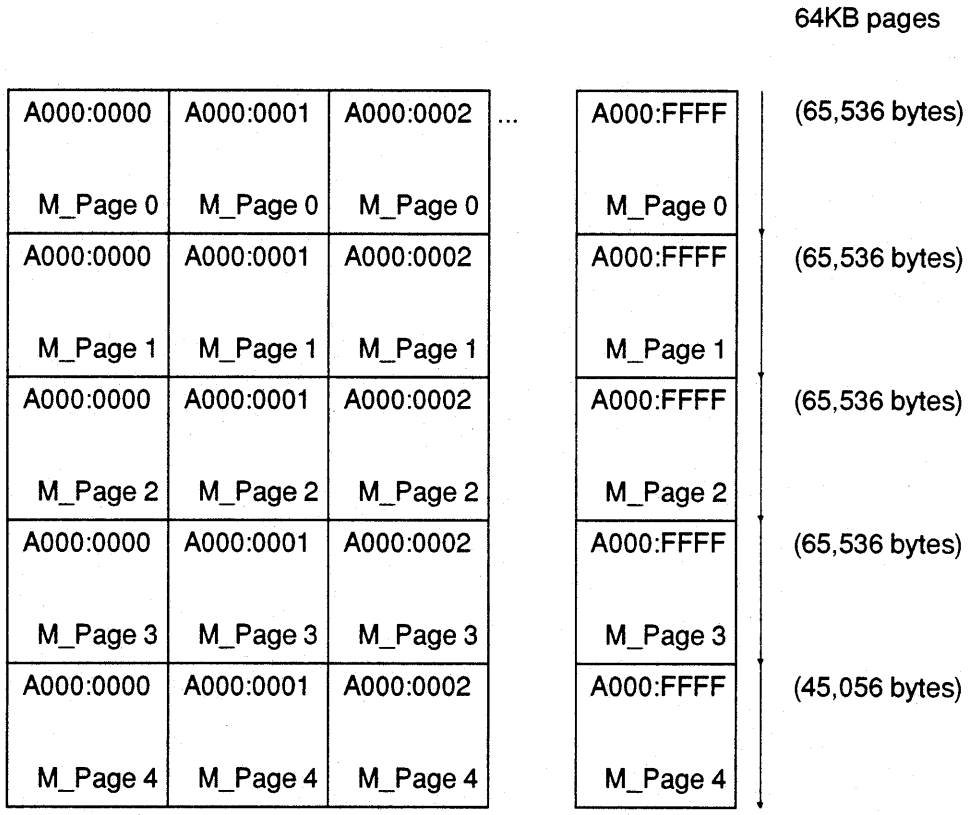


Figure 4-21. Memory Organization - Mode 62

APA Mode 63 (256-color 800x600)

- Page memory (512KB required). For explanation, see *Memory Maps* on page 4-2.
- One PEL per byte.

C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0

  C_{7-0} specify the PEL color.

- First PEL (upper-left corner of display) is at A0000, second at A0001. First 800 bytes form the first raster scan line. The display buffer for a full screen is mapped to eight m_pages as follows:

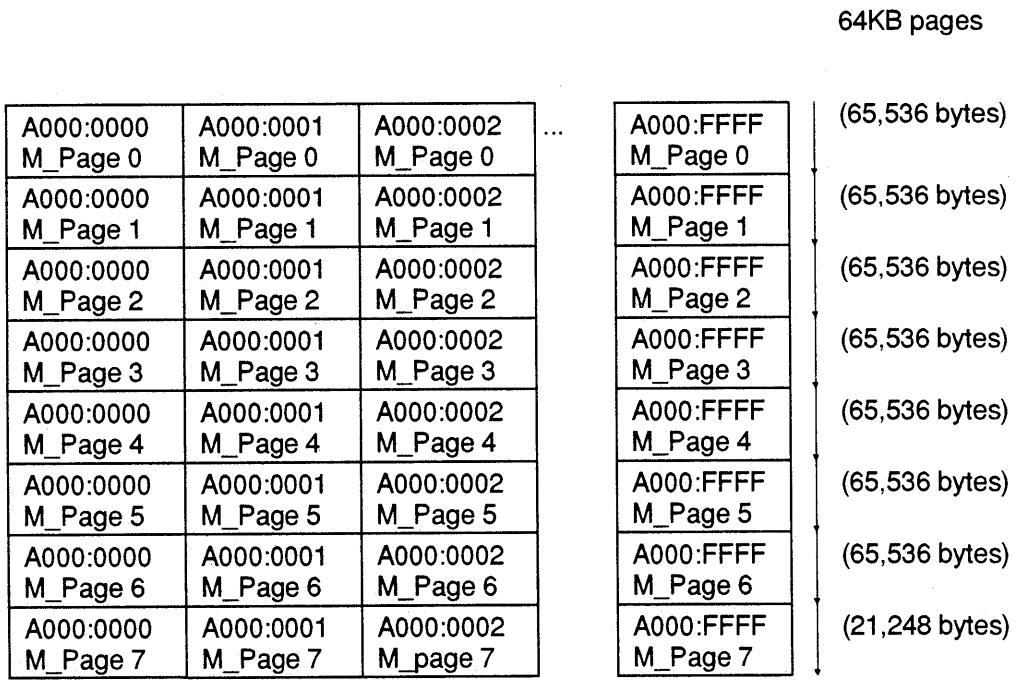


Figure 4-22. Memory Organization - Mode 63

APA Mode 64 (256-color 1024x768)

- Page memory (1MB RAM required) For explanation, see *Memory Maps* on page 4-2.
- One PEL per byte.

C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0

C_{7-0} specify the PEL color

- First PEL (upper-left corner of display) is at A0000, second at A0001. First 1024 bytes form the first raster scan line. The display buffer for a full screen is mapped to twelve m_pages as follows:

A000:0000 M_Page 0	A000:0001 M_Page 0	A000:0002 M_Page 0	...	A000:FFFF M_Page 0	64KB pages (65,536 bytes)
A000:0000 M_Page 1	A000:0001 M_Page 1	A000:0002 M_Page 1		A000:FFFF M_Page 1	(65,536 bytes)
A000:0000 M_Page 2	A000:0001 M_Page 2	A000:0002 M_Page 2		A000:FFFF M_Page 2	(65,536 bytes)
A000:0000 M_Page 3	A000:0001 M_Page 3	A000:0002 M_Page 3		A000:FFFF M_Page 3	(65,536 bytes)
A000:0000 M_Page 4	A000:0001 M_Page 4	A000:0002 M_Page 4		A000:FFFF M_Page 4	(65,536 bytes)
A000:0000 M_Page 5	A000:0001 M_Page 5	A000:0002 M_Page 5		A000:FFFF M_Page 5	(65,536 bytes)
A000:0000 M_Page 6	A000:0001 M_Page 6	A000:0002 M_Page 6		A000:FFFF M_Page 6	(65,536 bytes)
A000:0000 M_Page 7	A000:0001 M_Page 7	A000:0002 M_Page 7		A000:FFFF M_Page 7	(65,536 bytes)
A000:0000 M_Page 8	A000:0001 M_Page 8	A000:0002 M_Page 8		A000:FFFF M_Page 8	(65,536 bytes)
A000:0000 M_Page 9	A000:0001 M_Page 9	A000:0002 M_Page 9		A000:FFFF M_Page 9	(65,536 bytes)
A000:0000 M_Page 10	A000:0001 M_Page 10	A000:0002 M_Page 10		A000:FFFF M_Page 10	(65,536 bytes)
A000:0000 M_Page 11	A000:0001 M_Page 11	A000:0002 M_Page 11		A000:FFFF M_Page 11	(65,536 bytes)

Figure 4-23. Memory Organization - Mode 64

This page intentionally left blank.

5

VGA-Compatible Registers

Overview

VGA registers in ***mach32*** controllers are fully hardware-compatible with registers in the IBM VGA video adapter and Hercules graphics adapters. (See *Section 2, Programmer's Overview.*) In addition to the compatible registers, ATI's controllers also contain many extended-VGA registers that support higher resolutions, faster video modes, and enhanced features. These registers are described in Chapter 6, *VGA Register Extensions.*

Both VGA extended and compatible registers are listed in the Index. For convenience, they are also listed by I/O Port addresses on page 5-2. This chapter contains detailed descriptions on the compatible registers, arranged by functions as follows:

VGA Compatible Registers — by I/O Port

Port	Type	Mnemonic	Register Name	Page
0102	R/W	GENVS	VGA Sleep	5-9
03?4	R/W	CRTX	CRTC Index	5-20
03?5	R/W	CRT00	Horizontal Total	5-20
03?5	R/W	CRT01	Horizontal Display Enable	5-21
03?5	R/W	CRT02	Start Horizontal Blanking	5-21
03?5	R/W	CRT03	End Horizontal Blanking	5-22
03?5	R/W	CRT04	Start Horizontal Retrace	5-23
03?5	R/W	CRT05	End Horizontal Retrace	5-23
03?5	R/W	CRT06	Vertical Total	5-24
03?5	R/W	CRT07	CRTC Overflow	5-25
03?5	R/W	CRT08	Preset Row Scan	5-26
03?5	R/W	CRT09	Maximum Scan Line	5-27
03?5	R/W	CRT0A	Cursor Start	5-28
03?5	R/W	CRT0B	Cursor End	5-29
03?5	R/W	CRT0C	Start Address (High Byte)	5-30
03?5	R/W	CRT0D	Start Address (Low Byte)	5-30
03?5	R/W	CRT0E	Cursor Location (High Byte)	5-31
03?5	R/W	CRT0F	Cursor Location (Low Byte)	5-31
03?5	R/W	CRT10	Start Vertical Retrace	5-32
03?5	R/W	CRT11	End Vertical Retrace	5-33
03?5	R/W	CRT12	Vertical Display Enable End	5-34
03?5	R/W	CRT13	Offset	5-34
03?5	R/W	CRT14	Underline Location	5-35
03?5	R/W	CRT15	Start Vertical Blanking	5-36
03?5	R/W	CRT16	End Vertical Blanking	5-36
03?5	R/W	CRT17	CRT Mode	5-37
03?5	R/W	CRT18	Line Compare	5-39
03CA	W	GENFC	Feature Control	5-5
03?A	R	GENS1	Input Status 1	5-6
03?B	R/W	GENLPC	Light Pen Clear	5-12
03B9	R	GENLPS	Light Pen Set (CGA) - Also see 03DC	5-11
03BF	R/W	GENHP	Hercules Page	5-12
03C0	W	ATTRX	Attribute Controller Index	5-52
03C0	W	ATTR(00:0F)	Palette (00 to 0F)	5-53
03C0	W	ATTR10	Mode Control	5-54
03C0	W	ATTR11	Overscan Color	5-56
03C0	W	ATTR12	Color Map Enable	5-57
03C0	W	ATTR13	Horizontal PEL Panning	5-58
03C0	W	ATTR14	Color Select	5-59
03C1	R	ATTRX	Attribute Controller Index	5-52
03C1	R	ATTR(00:0F)	Palette (00 to 0F)	5-53

Note: ?=B when GENMO[0]=0 (Monochrome emulation)
?=D when GENMO[0]=1 (Color/graphics emulation

Port	Type	Mnemonic	Register Name	Page
03C1	R	ATTR10	Mode Control	5-54
03C1	R	ATTR11	Overscan Color	5-56
03C1	R	ATTR12	Color Map Enable	5-57
03C1	R	ATTR13	Horizontal PEL Panning	5-58
03C1	R	ATTR14	Color Select	5-59
03C2	W	GENMO	Miscellaneous Output	5-4
03C2	R	GENSO	Input Status 0	5-5
03C3	R	GENENB	VGA Subsystem Enable (Board)	5-8
03C4	R/W	SEQX	Sequencer Index	5-15
03C5	R/W	SEQ00	Reset	5-15
03C5	R/W	SEQ01	Clocking Mode	5-16
03C5	R/W	SEQ02	Map Mask	5-17
03C5	R/W	SEQ03	Character Map Select	5-18
03C5	R/W	SEQ04	Memory Mode	5-19
03C6	R/W	DAC_MASK	DAC Mask Register	5-13
03C7	R/W	DAC_R_INDEX	DAC Read Current Color Index Register	5-14
03C8	R/W	DAC_W_INDEX	DAC Write Current Color Index Register	5-14
03C9	R/W	DAC_DATA	DAC Data Register	5-13
03CA	R	GENFC	Feature Control	5-5
03CC	R	GENMO	Miscellaneous Output	5-4
03CE	R/W	GRAX	Graphics Controller Index	5-39
03CF	R/W	GRA00	Set/Reset	5-40
03CF	R/W	GRA01	Enable Set/Reset	5-42
03CF	R/W	GRA02	Color Compare	5-43
03CF	R/W	GRA03	Data Rotate	5-44
03CF	R/W	GRA04	Read Map Select	5-45
03CF	R/W	GRA05	Graphics Mode	5-46
03CF	R/W	GRA06	Graphics Miscellaneous	5-49
03CF	R/W	GRA07	Color Don't Care	5-50
03CF	R/W	GRA08	Bit Mask	5-51
03D9	R/W	GENB	Border - Palette (CGA)	5-10
03DC	W	GENLPS	Light Pen Set (CGA) - Also see 03B9	5-11
46E8	W	GENENA	VGA Subsystem Enable (Add-On)	5-8

GENERAL REGISTERS				
3CCh (R) 3C2h (W)	Miscellaneous Output Register (GENMO)			
	VGA	-	-	-

Bit	Mnemonic	Description
0	I/O Address Select	**0** = Addressing for monochrome emulation **1** = Addressing for color/graphic emulation
1	Video RAM Enable	**0** = Disables CPU access to video RAM **1** = Enables CPU access to video RAM
3:2	Clock Select	**00** = 25.1744 MHz (640 PELs) **01** = 28.3212 MHz (720 PELs) **10** = Reserved **11** = Reserved To change these two bits, SEQ0[0:1] must first be set to zero
4	-	Reserved
5	Even/Odd Mode Page Select	This bit is used in Even/Odd display modes (A/N modes: 0, 1, 2, 3, and 7). This bit is ignored when bit GRA06[1] or SEQ4[3] is enabled. **0** = Selects even (low) video memory locations **1** = Selects odd (high) video memory locations
7:6	-	Dual purpose bits used to select screen size and retrace sync polarity. (x=Bit not used for selection) Screen Size: **00** = Reserved **01** = Screen size is 400 lines **10** = Screen size is 350 lines **11** = Screen size is 480 lines Sync Polarity: **x0** = H Retrace pulse is active high **x1** = H Retrace pulse is active low **0x** = V Retrace pulse is active high **1x** = V Retrace pulse is active low

Note: In VGA mode, this register controls I/O port and video buffer addressing, and selects the dot clock frequency.

GENERAL REGISTERS				
3CAh (R) 3?Ah (W)	Feature Control Register (GENFC)			
	VGA	-	-	-

Bit	Mnemonic	Description
2:0	-	Reserved
3	-	Vertical Sync Select **0** = Normal vertical sync **1** = Sync is 'vertical sync' ORed 'vertical display enable'
7:4	-	Reserved

Note: **?** = B when GENMO[0]=0 (Monochrome emulation)
? = D when GENMO[0]=1 (Color/Graphics emulation)

GENERAL REGISTERS				
3C2h (R)	Input Status 0 Register (GENS0)			
	VGA	-	-	-

Bit	Mnemonic	Description
3:0	-	Reserved
4	-	Switch Sense **0** = Output state of the DAC Lookup Table Comparators is inactive **1** = Output state of the DAC Lookup Table Comparators is active
5	-	Reserved
6	-	Reserved
7	-	CRT Interrupt **0** = Vertical retrace interrupt is cleared **1** = Vertical retrace interrupt is pending

GENERAL REGISTERS				
3?Ah (R)	Input Status 1 Register (GENS1)			
	VGA	-	-	-

Bit	Mnemonic	Description
0	-	Display Enable **0** = Enables display of video data **1** = Disables display of video data
2:1	-	Reserved
3	-	Vertical Retrace Status **0** = V Retrace pulse is inactive **1** = V Retrace pulse is in progress
5, 4	-	Diagnostic Bits 0, 1 respectively • These two bits are connected to two of the eight color outputs (P7:P0) of the attribute controller. Connections are controlled by ATTR12[5,4] as follows: **00** = P2, P0 **01** = P5, P4 **10** = P3, P1 **11** = P7, P6
7:6	-	Reserved

Notes:

1. **?** = B when GENMO[0]=0 (Monochrome emulation)
   ? = D when GENMO[0]=1 (Color/Graphics emulation)

2. Bits 0 and 3 can be used to synchronize the video buffer updates with the screen refresh cycles to minimize interference with the displayed image.

GENERAL REGISTERS				
3?8h (R/W)	Mode Control Register (GENMC)			
	VGA	-	-	-

Bit	Mnemonic	Description
0	-	Monochrome Emulation: Reserved Color/Graphics Emulation: **0** = 40x25 Text **1** = 80x25 Text
1	-	Monochrome Emulation: **0** = Text Mode **1** = Hercules Graphics Mode Color/Graphics Emulation: **0** = Text mode **1** = Graphics mode
2	-	Monochrome Emulation: Reserved Color/Graphics Emulation: **0** = Color mode **1** = B/W mode
3	-	Monochrome Emulation: **1** = Enables video Color/Graphics Emulation: **1** = Enables video
4	-	Monochrome Emulation: Reserved Color/Graphics Emulation: **1** = Enables 640x200 B/W mode
5	-	Monochrome Emulation: **1** = Enables screen blanking Color/Graphics Emulation: **0** = Keeps background intensity attribute bit **1** = Changes background intensity attribute bit
6	-	Monochrome Emulation: Reserved Color/Graphics Emulation: Reserved
7	-	Monochrome Emulation: **0** = Selects Hercules page 0 **1** = Selects Hercules page 1 Color/Graphics Emulation: Reserved

Note: **?** = B when GENMO[0]=0 (Monochrome emulation)
? = D when GENMO[0]=1 (Color/Graphics emulation)

GENERAL REGISTERS				
46E8h (W)	Video Subsystem Enable (Add-On) Register (GENENA)			
	VGA	-	-	-

Bit	Mnemonic	Description
2:0		Reserved
3	-	VGA Enable **0** = Puts VGA video subsystem into sleep mode, during which, the VGA video subsystem only responds to memory read operations to the BIOS ROM, and I/O writes to register 102h. All other I/O or video memory read/write operations are suspended. **1** = Enables I/O and memory address decoding of VGA video subsystem, if GENVS[0] is also a logical one.
4	-	GENVS[0] Enable **0** = Disables I/O write to GENVS (0102). **1** = Enables I/O write to GENVS (0102).
7:5	-	Reserved

GENERAL REGISTERS				
3C3h (R)	Video Subsystem Enable(Board) Register (GENENB)			
	VGA	-		-

Bit	Mnemonic	Description
0	-	VGA Enable • Read back status of GENVS[0] (0102)
7:1	-	Reserved

GENERAL REGISTERS				
102h (R/W)	VGA Sleep Register (GENVS)			
	VGA	-		-

Bit	Mnemonic	Description
0	-	VGA Sleep **0** = Disables VGA video subsystem (controller). The VGA video subsystem only responds to memory read operations to the BIOS ROM. All other I/O or memory read/write operations are suspended. **1** = Enables VGA video subsystem.
7:1	-	Reserved

Notes:

1. Writes to this register are controlled by GENENA[4].

2. This register is readable only in Micro-Channel mode

3. Example — to enable the VGA:

```
MOV  DX, 46E8
MOV  AL, 10
OUT  DX, AL
MOV  DX, 102
MOV  AL, 1
OUT  DX, AL
MOV  DX, 46E8
MOV  AL, 8
OUT  DX, AL
```

GENERAL REGISTERS				
3D9h (R/W)	Border (Palette) Register (GENB)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
0		• Selects a blue border in 40x25, 80x25 text modes; 16-Color 320x200, 16-Color 640x200 graphics modes. • Selects a blue background in 4-Color 320x200, 4-Color 640x200 graphics modes. • Selects blue as foreground in 640x200 B/W mode.		
1		• Selects a green border in 40x25, 80x25 text modes; 16-Color 320x200, 16-Color 640x200 graphics modes. • Selects a green background in 4-Color 320x200, 4-Color 640x200 graphics modes. • Selects green as foreground in 640x200 B/W mode.		
2		• Selects a red border in 40x25, 80x25 text modes; 16-Color 320x200, 16-Color 640x200 graphics modes. • Selects a red background in 4-Color 320x200, 4-Color 640x200 graphics modes. • Selects red as foreground in 640x200 B/W mode.		
3		• Selects an intensified border color in 40x25, 80x25 text modes; 16-Color 320x200, 16-Color 640x200 graphics modes. • Selects an intensified background color in 4-Color 320x200, 4-Color 640x200 graphics modes. • Selects an intensified foreground color in 640x200 B/W mode.		
4		Selects an intensified set of foreground colors in 4-Color 320x200 or 4-Color 640x200 graphics mode.		
5		Selects an active set of colors in 4-Color 320x200 or 4-Color 640x200 graphics mode.		

GENERAL REGISTERS				
3D9h (R/W)	Border (Palette) Register (GENB)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
7:6	-	Reserved		
Note: This register is used in CGA emulation mode.				

GENERAL REGISTERS				
3B9h (R) 3DCh (W)	Light Pen Set Register (GENLPS)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
7:0	-	Reading or writing at this I/O location sets the Light Pen Set register.		
Note: This register is used in CGA emulation mode.				

GENERAL REGISTERS				
3?Bh (R/W)	Light Pen Clear Register (GENLPC)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
7:0	-	Reading or writing at this I/O location clears the Light Pen Clear register.		

Notes:

1. ? = B when GENMO[0]=0 (Monochrome emulation)
   ? = D when GENMO[0]=1 (Color/Graphics emulation)

2. This register is used in CGA and Hercules emulation modes.

GENERAL REGISTERS				
3BFh (R/W)	Hercules Page Register (GENHP)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
0	-	**0** = Prevents graphics mode. **1** = Allows graphics mode.		
1	-	**0** = Disables upper 32K of graphics mode buffer **1** = Enables upper 32K of graphics mode buffer at B8000h.		
7:2	-	Reserved		

Note: This register is used in Hercules emulation mode.

DAC REGISTERS				
03C9h (R/W)	DAC Data Register (DAC_DATA)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
7:0	DAC_DATA	DAC data		

DAC REGISTERS				
03C6h (R/W)	DAC Mask Register (DAC_MASK)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
7:0	DAC_MASK	Participating bit positions in the mask for DAC lookup are set to one.		

DAC REGISTERS				
03C7h (R/W)	DAC Read Current Color Index Register (DAC_R_INDEX)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
7:0	DAC_R_INDEX	The current read index for a DAC operation — increments after every third read of DAC_DATA (03C9h). Also see DAC_W_INDEX (03C8h)		
Note: Only bit 0 of this register is readable. Writing the DAC at 03C8h results in a read-back value of 0. Writing the DAC at 03C7h results in a read-back value of 1.				

DAC REGISTERS				
03C8h (R/W)	DAC Write Current Color Index Register (DAC_W_INDEX)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
7:0	DAC_W_INDEX	The current write index for a DAC operation. Also see DAC_R_INDEX (03C7h)		

SEQUENCER REGISTERS				
3C4h (R/W)	Sequencer Index Register (SEQX)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
2:0	-	This index points to one of the sequencer registers (SEQ) at I/O port address 3C5h, for the next SEQ read/write operation. These registers are described on the following pages.		
7:3	-	Reserved		

SEQUENCER REGISTERS				
3C5h (R/W)	Reset Register (SEQ00)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
0	-	Synchronous Reset Bit 0 **0** = Follows SEQ00[1]. **1** = Allows the sequencer to run unless SEQ00[1] is zero.		
1	-	Synchronous Reset Bit 1 **0** = Disable character clock, and display requests to the video memory and H/V sync signals. **1** = Allows the sequencer to run unless SEQ00[0] is zero.		
7:2	-	Reserved		

Notes:

1. Bits 0 and 1 must both be zero (sequencer halted) before any clock select bits are changed; for example, clock selects GENMO[3:2] or SEQ01[0:3].

2. The SEQ00[0:1] bits must both be set to one for normal operation.

SEQUENCER REGISTERS				
3C5h (R/W)	Clock Mode Register (SEQ01)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
0	-	8/9 Dot Clocks **0** = Selects 9-dot character clocks. **1** = Selects 8-dot character clocks. • Modes 0, 1, 2, 3, 7 use 9-dot characters. • To change bit 0, GENVS[0] must be logical zero.		
1	-	Reserved		
4,2	-	Shift 4, Shift Load bits: **00** = Loads video serializers every character clock **01** = Loads video serializers every other character clock. **10** = Loads video serializers every fourth character clock. **11** = Loads video serializers every fourth character clock.		
3	-	Dot Clock **0** = Indicates dot clock is Master clock. **1** = Indicates dot clock is Master clock divided by 2. • Typically, 320 and 360 horizontal modes use divide-by-2 to provide 40-column displays. • To change this bit, SEQ00[0:0] must first be set to zero.		
5	-	Screen Off **0** = Allows CPU:CRT interleaved access to video memory. **1** = Blanks the screen and disables video-generation logic access to video memory. Allows CPU uninterrupted access to video memory.		
7:6	-	Reserved		
Note: To change this register, SEQ00[1 or 0] must first be logical zero.				

SEQUENCER REGISTERS				
3C5h (R/W)	Map Mask Register (SEQ02)			
	VGA	-	-	-

Bit	Mnemonic	Description
0		Enable Map 0 **0** = Disables write access to memory map 0. **1** = Enables write access to memory map 0.
1		Enable Map 1 **0** = Disables write access to memory map 1. **1** = Enables write access to memory map 1.
2	-	Enable Map 2 **0** = Disables write access to memory map 2. **1**= Enables write access to memory map 2.
3	-	Enable Map 3 **0** = Disables write access to memory map 3. **1** = Enables write access to memory map 3.
7:4	-	Reserved

Notes:

1. In 4 bit per PEL graphics modes, when the value of this register is set to '1111' (0Fh), the processor can complete a 32-bit write operation in one memory cycle.
2. In text modes, CPU only needs to access maps 0 and 1; therefore, this register should have a value of 03h.
3. When in odd/even modes, map mask value for maps 0 and 1 should be same as the map mask value for maps 2 and 3.
4. Memory map updating such as bit map layering can be selectively enabled or disabled using bits in this register. For pixel-oriented operations, the graphics controller provides better control.

SEQUENCER REGISTERS				
3C5h (R/W)	Character Map Select Register (SEQ03)			
	VGA	-	-	-

Bit	Mnemonic	Description
4,1,0		Character Map Select B Bits 2:0
5,3,2		Character Map Select A Bits 2:0
7:6	-	Reserved

Notes:

1. Extended memory SEQ04[1] must be logical one to enable this select function; otherwise, the first 8K of map 2 is always selected.

2. Any changes made to this register take effect at the start of the next character line on the display.

3.

*Bit Pattern	Map Selected	Offset into Map
0 0 0	0	0K
0 0 1	1	16K
0 1 0	2	32K
0 1 1	3	48K
1 0 0	4	8K
1 0 1	5	24K
1 1 0	6	40K
1 1 1	7	56K

SEQUENCER REGISTERS				
3C5h (R/W)	**Memory Mode Register (SEQ04)**			
	VGA	**-**	**-**	**-**

Bit	Mnemonic	Description
0		Reserved
1		Extended Memory • Indicates 256KB of video memory is present. Also enables character map selection in SEQ03.
2	-	Odd/Even **0** = Uses the LSB CPU address bit A0 to select which memory map to access. Even CPU addresses access maps 0 and 2; odd addresses access maps 1 and 3. **1** = Enables sequential access to video data maps for odd/even modes. Map Mask register bits SEQ02[0:3] identify which maps are to be accessed for each CPU address.
3	-	Chain **0** = Enables sequential data access within a bit map. Map Mask register bits SEQ02[0:3] identify which maps are to be accessed at any one time. **1** = In 256-color modes map select is by CPU address bits A0 and A1: **A1, A0 Map Selected** 0 0 0 0 1 1 1 0 2 1 1 3 • When Chain is logical one, it takes priority over odd/even mode bits SEQ04[2] and GRA05[4]. Unlike odd/even mode, SEQ04[2] is the only bit used to enable chain mode (double odd/even). • Chain does not affect CRTC access to video memory. • Odd/even bit SEQ04[2] should be the opposite of GRA05[4].
7:4	-	Reserved

CRT CONTROLLER REGISTERS				
3?4h (R/W)	CRTC Index Register (CRTX)			
	VGA	-	-	-

Bit	Mnemonic	Description
4:0	-	This index points to one of the internal registers of the CRT controller (CRTC) at address 3?5h, for the next CRTC read/write operation. These registers are described on the following pages.
7:5	-	Reserved

Note: ? = B when GENMO[0]=0 (Monochrome emulation)
? = D when GENMO[0]=1 (Color/Graphics emulation)

CRT CONTROLLER REGISTERS				
3?5h (R/W)	Horizontal Total Register (CRT00)			
	VGA	-	-	-

Bit	Mnemonic	Description
7:0	-	These bits define the active horizontal display in a scan line, including the retrace period. The value is five less than the total number of displayed characters in a scan line.

Note: ? = B when GENMO[0]=0 (Monochrome emulation)
? = D when GENMO[0]=1 (Color/Graphics emulation)

CRT CONTROLLER REGISTERS				
3?5h (R/W)	Horizontal Display Enable End Register (CRT01)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
7:0	-	These bits define the active horizontal display in a scan line. The value is one less than the total number of displayed characters in a scan line.		
Note: ? = B when GENMO[0]=0 (Monochrome emulation) ? = D when GENMO[0]=1 (Color/Graphics emulation)				

CRT CONTROLLER REGISTERS				
3?5h (R/W)	Start Horizontal Blanking Register (CRT02)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
7:0	-	These bits define the horizontal character count that represents the character count in the active display area plus the right border. In other words, the count is from the start of active display to the start of triggering of the H blanking pulse.		
Note: ? = B when GENMO[0]=0 (Monochrome emulation) ? = D when GENMO[0]=1 (Color/Graphics emulation)				

CRT CONTROLLER REGISTERS				
3?5h (R/W)	End Horizontal Blanking Register (CRT03)			
	VGA	-	-	-

Bit	Mnemonic	Description
4:0	-	H Blanking End bits 4-0, respectively • These are the five low-order of six bits of horizontal character count for triggering the end of the horizontal blanking pulse. The sixth bit is CRT05[7]. The character count is equal to the sum of "H blanking start" plus "H blanking pulse width".
6:5	-	Display Enable Skew: **00** = Zero-character-clock skew. **01** = One-character-clock skew. **10** = Two-character-clock skew. **11** = Three-character-clock skew.
7	-	Compatibility Read **0** = Enables write-only to CRT10 and CRT11. **1** = Enables read/write access to both vertical retrace start/end registers CRT10 and CRT11.

Note: **?** = B when GENMO[0]=0 (Monochrome emulation)
? = D when GENMO[0]=1 (Color/Graphics emulation)

CRT CONTROLLER REGISTERS				
3?5h (R/W)	Start Horizontal Retrace Register (CRT04)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
7:0	-	These bits define the horizontal character count at which the horizontal retrace pulse becomes active.		
Note: ? = B when GENMO[0]=0 (Monochrome emulation) ? = D when GENMO[0]=1 (Color/Graphics emulation)				

CRT CONTROLLER REGISTERS				
3?5h (R/W)	End Horizontal Retrace Register (CRT05)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
4:0	-	H Retrace End bits • These are the 5-bit result from the sum of CRT04 plus the width of the horizontal retrace pulse, in character clock units.		
6:5	-	H Retrace Delay bits • These two bits skew the Horizontal Retrace pulse. **00** = Zero character clocks **01** = One character clocks **10** = Two character clocks **11** = Three character clocks		
7	-	H Blanking End Bit 5 • This is bit 5 of the 6-bit character count for the H blanking end pulse. The other five low-order bits are CRT03[4:0].		
Note: ? = B when GENMO[0]=0 (Monochrome emulation) ? = D when GENMO[0]=1 (Color/Graphics emulation)				

CRT CONTROLLER REGISTERS				
3?5h (R/W)	Vertical Total Register (CRT06)			
	VGA	-	-	-

Bit	Mnemonic	Description
7:0	-	These are the eight low-order bits of the 10-bit vertical total register. The two high-order bits are CRT07[0:5] in the CRTC overflow register. The value of this register represents the total number of H raster scans plus vertical retrace (active display, blanking), minus two scan lines.

Note: **?** = B when GENMO[0]=0 (Monochrome emulation)
? = D when GENMO[0]=1 (Color/Graphics emulation)

CRT CONTROLLER REGISTERS				
3?5h (R/W)	CRTC Overflow Register (CRT07)			
	VGA	-	-	-

Bit	Mnemonic	Description
0	-	V Total Bit 8 (CRT06) • Bit 8 of 10-bit vertical count for V Total (for functional description see CRT06 register).
1	-	End V Display Bit 8 (CRT12) • Bit 8 of 10-bit vertical count for V Display enable end (for functional description see CRT12 register).
2	-	Start V Retrace Bit 8 (CRT10) • Bit 8 of 10-bit vertical count for V Retrace start. For functional description see CRT10 register.
3	-	Start V Blanking Bit 8 (CRT15) • Bit 8 of 10-bit vertical count for V Blanking start (for functional description see CRT15 register).
4	-	Line Compare Bit 8 (CRT18) • Bit 8 of 10-bit vertical count for Line Compare (for functional description see CRT18 register).
5	-	V Total Bit 9 (CRT06) • Bit 9 of 10-bit vertical count for V Total (for functional description see CRT06 register).
6	-	End V Display Bit 9 (CRT12) • Bit 9 of 10-bit vertical count for V Display enable end (for functional description see CRT12 register).
7	-	Start V Retrace Bit 9 (CRT10) • Bit 9 of 10-bit vertical count for V Retrace start (for functional description see CRT10 register).

Notes:

1. **?** = B when GENMO[0]=0 (Monochrome emulation)
   ? = D when GENMO[0]=1 (Color/Graphics emulation)

2. Various bits in this register are functionally part of other CRTC registers.

CRT CONTROLLER REGISTERS				
3?5h (R/W)	Preset Row Scan Register (CRT08)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
4:0	-	Preset Row Scan bits 4:0 • This register is used for software-controlled vertical scrolling in text or graphics modes. The value specifies the first line to be scanned after a V Retrace (in the next frame.) • Each H Retrace pulse increments the counter by 1, up to the Maximum Scan Line value programmed by CRT09, then the counter is cleared.		
6:5	-	Byte Panning Control Bits 1 and 0, respectively • Bits 6 and 5 extend the capability of byte panning (shifting) by up to three characters (for description see *H PEL Panning register ATTR13*).		
7	-	Reserved		
Note: ? = B when GENMO[0]=0 (Monochrome emulation) ? = D when GENMO[0]=1 (Color/Graphics emulation)				

CRT CONTROLLER REGISTERS				
3?5h (R/W)	Maximum Scan Line Register (CRT09)			
	VGA	-	-	-

Bit	Mnemonic	Description
4:0	-	Maximum Scan Line bits • These bits define a value that is the actual number of scan lines per character minus one.
5	-	Start V Blanking bit 9 (CRT15) • Bit 9 of 10-bit vertical count for V blanking start (for functional description see *CRT15 register*).
6	-	Line Compare bit 9 (CRT18) • Bit 9 of 10-bit vertical count for line compare (for functional description see *CRT18 register*).
7	-	200-/400-Line Scan **0** = Counter is incremented per scan line. **1** = Clock pulses to the row scan counter are divided by two. Effectively, this allows the lines in 200-line mode to be displayed twice before the row scan counter is incremented once. **Note:** H/V display and blanking timings etc. (in CRT00-CRT06 registers) are not affected by these bits. • In ATI extended modes, scan function is configured in ATI31[5:3]

Note: **?** = B when GENMO[0]=0 (Monochrome emulation)
? = D when GENMO[0]=1 (Color/Graphics emulation)

CRT CONTROLLER REGISTERS				
3?5h (R/W)	Cursor Start Register (CRT0A)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
4:0		Cursor Start bits 4-0, respectively • These bits define a value that is the starting scan line (on a character row) for the line cursor. The five-bit value is equal to the actual number minus one. This value is used together with Cursor End bits CRT0B[4:0] to determine the height of the cursor. • The cursor height in VGA does not wrap around (as in EGA) and is actually absent when the 'end' value is less than the 'start' value. In EGA, when the 'end' value is less, the cursor is a full block cursor the same height as the character cell.	
5	-	Cursor On/Off **0** = Cursor on. **1** = Cursor off.		
7:6	-	Reserved		
Note: **?** = B when GENMO[0]=0 (Monochrome emulation) **?** = D when GENMO[0]=1 (Color/Graphics emulation)				

CRT CONTROLLER REGISTERS				
3?5h (R/W)	Cursor End Register (CRT0B)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
4:0	-	Cursor End Bits 4-0, respectively • These bits define the ending scan row (on a character line) for the line cursor. In EGA, this 5-bit value is equal to the actual number of lines plus one. • The cursor height in VGA does not wrap around (as in EGA) and is actually absent when the 'end' value is less than the 'start' value. In EGA when the 'end' value is less, the cursor is a full block cursor the same height as the character cell.		
6:5	-	Cursor Skew Bits 1 and 0, respectively • These bits define the number of characters the cursor is to be shifted to the right (skewed) from the character pointed at by the cursor location registers (CRT0E and CRT0F), in VGA mode. Skew values when in EGA mode are enclosed in brackets. **00** = Zero(zero) character skew. **01** = One (zero) character skew. **10** = Two (one) character skew. **11** = Three (two) character skew.		
7	-	Reserved		
Note: **?** = B when GENMO[0]=0 (Monochrome emulation) **?** = D when GENMO[0]=1 (Color/Graphics emulation)				

CRT CONTROLLER REGISTERS				
3?5h (R/W)	Start Address (High Byte) Register (CRT0C)			
	VGA	-	-	-

Bit	Mnemonic	Description
7:0	-	SA bits 15:8 • These are the eight high-order bits of the 16-bit display buffer start location. The low-order bits are contained in CRT0D. • In split screen mode, CRT0C + CRT0D points to the starting location of screen A (top half.) The starting address for screen B is always zero.

Notes:

1. ? = B when GENMO[0]=0 (Monochrome emulation)
   ? = D when GENMO[0]=1 (Color/Graphics emulation)

2. *ATI Extended modes have additional SA bits: ATI30[6], ATI23[4].

CRT CONTROLLER REGISTERS				
3?5h (R/W)	Start Address (Low Byte) Register (CRT0D)			
	VGA	-	-	-

Bit	Mnemonic	Description
7:0	-	SA bits 7:0 These are the eight low-order bits of the 16-bit display buffer start location. The high-order bits are contained in CRT0C. • In split screen mode, CRT0C+CRT0D points to the starting location of screen A (top half.) The starting address for screen B is always zero.

Notes:

1. ? = B when GENMO[0]=0 (Monochrome emulation)
   ? = D when GENMO[0]=1 (Color/Graphics emulation)

2. *ATI Extended modes have additional SA bits: ATI30[6], ATI23[4].

CRT CONTROLLER REGISTERS				
3?5h (R/W)	Cursor Location (High Byte) Register (CRT0E)			
	VGA	-	-	-

Bit	Mnemonic	Description
7:0	-	CA bits 15:8 • These are the eight high-order bits of the 16-bit cursor start address. The low-order CA bits are contained in CRT0F. This address is relative to the start of physical display memory address pointed to by CRT0C+CRT0D. In other words, if CRT0C+CRT0D is changed, the cursor still points to the same character as before.

Notes:

1. ? = B when GENMO[0]=0 (Monochrome emulation)
   ? = D when GENMO[0]=1 (Color/Graphics emulation)

2. ATI Extended modes use additional CA bits: ATI30[2], ATI23[3].

CRT CONTROLLER REGISTERS				
3?5h (R/W)	Cursor Location (Low Byte) Register (CRT0F)			
	VGA	-	-	-

Bit	Mnemonic	Description
7:0	-	CA bits 7:0 • These are the eight low-order bits of the 16-bit cursor start address. The high-order CA bits are contained in CRT0E. This address is *relative* to the start of physical display memory address pointed to by CRT0C+CRT0D. In other words, if CRT0C+CRT0D is changed, the cursor still points to the same character as before.

Notes:

1. ? = B when GENMO[0]=0 (Monochrome emulation)
   ? = D when GENMO[0]=1 (Color/Graphics emulation)

2. ATI Extended modes use additional CA bits: ATI30[2], ATI23[3].

CRT CONTROLLER REGISTERS				
3?5h (R/W)	Start Vertical Retrace Register (CRT10)			
	VGA	-	-	-

Bit	Mnemonic	Description
7:0	-	Bits CRT10[7:0] are the eight low-order bits of the 10-bit vertical retrace start count. The two high-order bits are CRT07[2:7], located in the CRTC overflow register. • These bits define the horizontal scan count that triggers the V retrace pulse.

Notes:

1. **?** = B when GENMO[0]=0 (Monochrome emulation)
   ? = D when GENMO[0]=1 (Color/Graphics emulation)

2. This register is read/write enabled if CRT03[7] is set to one. It is write-only enabled if CRT03[7] is set to zero.

CRT CONTROLLER REGISTERS				
3?5h (R/W)	End Vertical Retrace Register (CRT11)			
	VGA	-	-	-

Bit	Mnemonic	Description
3:0	-	V Retrace End Bits 3-0 • Bits CRT11[0-3] define the horizontal scan count that triggers the end of the V Retrace pulse.
4	-	V Retrace Interrupt Set **0** = V Retrace interrupt cleared.
5	-	V Retrace Interrupt Disabled **0** = Enable V Retrace interrupt.
6	-	Reserved
7	-	Write Protect (CRT00-CRT07) **0** = Enables normal read/write of CRT00 to CRT07. **1** = Write protect registers CRT00 to CRT07 when in VGA mode as follows: All register bits except CRT07[4] are write protected.

Notes:

1. **?** = B when GENMO[0]=0 (Monochrome emulation)
   ? = D when GENMO[0]=1 (Color/Graphics emulation)

2. This register is read/write enabled if CRT07[7] is set to one. It is write-only enabled if CRT03[7] is set to zero.

CRT CONTROLLER REGISTERS				
3?5h (R/W)	Vertical Display Enable End Register (CRT12)			
	VGA	-	-	-

Bit	Mnemonic	Description
7:0	-	These are the eight low-order bits of the 10-bit register containing the horizontal scan count indicating where the active display on the screen should end. The high-order bits are CRT07[1:6] in the CRT overflow register.

Note: ? = B when GENMO[0]=0 (Monochrome emulation)
? = D when GENMO[0]=1 (Color/Graphics emulation)

CRT CONTROLLER REGISTERS				
3?5h (R/W)	Offset Register (CRT13)			
	VGA	-	-	-

Bit	Mnemonic	Description
7:0	-	• These bits define an offset value, equal to the logical line width of the screen (from the first character of the current line to the first character of the next line). • Memory organization is dependent on the video mode. Bit CRT17[6] selects byte or word mode. Bit CRT14[6], which overrides the byte/word mode setting, selects Double-Word mode when it is logical one. • The first character of the next line is specified by the start address (CRT0C+CRT0D) plus the offset. The offset for byte mode is 2x CRT13; for word mode, 4x; for double word mode, 8x.

Note: ? = B when GENMO[0]=0 (Monochrome emulation)
? = D when GENMO[0]=1 (Color/Graphics emulation)

CRT CONTROLLER REGISTERS				
3?5h (R/W)	Underline Location Register (CRT14)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
4:0	-	H Row Scan Bits 4-0 • These bits define the horizontal scan row, from the top of the character line, that should be used for underlining. The 5-bit value is equal to the actual number minus one.		
5	-	Count-by-4 **0** = Character clock is used unmodified at the memory address counter, for byte addressing. **1** = Character clock is dividedby4 at the clock input to the Memory address counter. Countby4 clocks are used for double-word addressing. This bit overrides divide-by-2 bit CRT17[3].		
6	-	Double-Word Mode **0** = Allows addressing mode to be selected by CRT17[6]. **1** = Enables double-word addressing mode. This bit overrides byte/word bit CRT17[6].		
7	-	Reserved		
Note: **?** = B when GENMO[0]=0 (Monochrome emulation) **?** = D when GENMO[0]=1 (Color/Graphics emulation)				

CRT CONTROLLER REGISTERS				
3?5h (R/W)	Start Vertical Blanking Register (CRT15)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
7:0	-	• These are the eight low-order bits of the 10-bit vertical blanking start register. Bit 9 is CRT09[5]; bit 8 is CRT07[3]. • The 10 bits specify the starting location of the vertical blanking pulse, in units of horizontal scan lines. The value is equal to the actual total number of displayed lines minus one.		
Note: ? = B when GENMO[0]=0 (Monochrome emulation) ? = D when GENMO[0]=1 (Color/Graphics emulation)				

CRT CONTROLLER REGISTERS				
3?5h (R/W)	End Vertical Blanking Register (CRT16)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
6:0	-	• These bits define the point at which to trigger the end of the vertical blanking pulse. The location is specified in units of horizontal scan lines. • The value to be stored in this register is the seven low-order bits of the sum of "pulse width count" plus the content of Start Vertical Blanking register (CRT15) minus one.		
7		Reserved		
Note: ? = B when GENMO[0]=0 (Monochrome emulation) ? = D when GENMO[0]=1 (Color/Graphics emulation)				

CRT CONTROLLER REGISTERS				
3?5h (R/W)	CRT Mode Register (CRT17)			
	VGA	-	-	-

Bit	Mnemonic	Description
0	-	Compatibility Mode **0** = Substitutes row scan counter bit 0 as bit 13 of CRTC output during active display time. This allows for compatibility with, e.g. the 6845 controller or CGA APA modes. **1** = Enables row scan counter bit 13 as bit 13 of CRTC output.
1	-	Select Row Scan Counter **0** = Selects row scan counter bit 1 (RA1) as bit 14 at the CRTC output during active display time. This substitution allows for compatibility with Hercules graphics and other 400-line graphics modes. **1** = Selects row scan counter bit 14 (RA14) as bit 14 at the CRTC output.
2	-	Vertical-by-2 **0** = Increments the vertical timing counter every horizontal retrace. **1** = Increments the vertical timing counter every two horizontal retrace pulses. It effectively doubles the vertical resolution by two, for example, to 2048 horizontal scan lines in VGA, and 1024 in EGA. **Note:** When bit 2 is logical one, other vertical register values should be adjusted as well (CRT06, CRT10, CRT12, CRT15, and CRT18).
3	-	Count-by-2 **0** = Increments the memory address counter every character clock. **1** = Increments the memory address counter every two character clocks.
4	-	Reserved

CRT CONTROLLER REGISTERS				
3?5h (R/W)	CRT Mode Register (CRT17)			
	VGA	-	-	-

Bit	Mnemonic	Description
5	-	Address Wrap **0** = Indicates only 64K video memory is to be addressed: In word mode, address counter bits are left shifted once, so bit 13 (MA13) is wrapped around to bit 0 position at the CRTC output. **1** = Enables 256K video memory addressing: In word mode, address counter bits are rotated left by one, so bit 15 (MA15) is wrapped around to bit 0 position at the CRTC output.
6	-	Byte/Word Mode **0** = Selects word mode memory addressing. The memory address is rotated left by one. **1** = Selects byte mode memory addressing.
7	-	H/V Retrace Enable **0** = Disables horizontal and vertical retrace. **1** = Enables horizontal and vertical retrace.

Notes:

1. **?** = B when GENMO[0]=0 (Monochrome emulation)
   ? = D when GENMO[0]=1 (Color/Graphics emulation)

2. 640x200 mode is programmed for 100 horizontal scan lines with two row scan addresses per character row. Odd scan lines are offset in the display memory by 8K bytes.

CRT CONTROLLER REGISTERS				
3?5h (R/W)	Line Compare Register (CRT18)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
7:0	-	• These bits are the eight low-order of the 10-bit line compare register. Bit 8 is CRT07[4], bit 9 is CRT09[6]. The value of this register is used to disable scrolling on a portion of the display screen, as when split screen is active. When the vertical counter reaches this value, the memory address and row scan counters are cleared. • The screen area above the line specified by this register is commonly called screen A. The screen below is screen B. Screen B cannot be scrolled, but it can be panned only together with screen A, controlled by the PEL panning compatibility bit ATTR10[5]. (For a description of this control bit see *ATTR10[5].*)		
Note: ? = B when GENMO[0]=0 (Monochrome emulation) ? = D when GENMO[0]=1 (Color/Graphics emulation)				

GRAPHICS CONTROLLER REGISTERS				
3CEh (R/W)	Graphics Controller Index Register (GRAX)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
3:0	-	• This index is used to address one of the internal registers of the graphics controller (GRAC) at I/O port 3CFh. These are described on the following pages.		
7:4	-	Reserved		

GRAPHICS CONTROLLER REGISTERS				
3CFh (R/W)	Set/Reset Register (GRA00)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
0	-	Set/Reset Map 0 **0** = All eight bits of buffer map 0 are to be written with zeros during CPU write if write mode is 0 (See write mode bits GRA05[1:0]), and if the enable set/reset bit GRA01[0] is a logical one. **1** = All eight bits of buffer map 1 are to be written with ones during CPU write if write mode is 0 or 3 (See write mode bits GRA05[1:0]), and if the enable set/reset bit GRA01[0] is a logical one.		
1	-	Set/Reset Map 1 **0** = All eight bits of buffer map 1 are to be written with zeros during CPU write if write mode is 0 (See write mode bits GRA05[1:0]), and if the enable set/reset bit GRA01[1] is a logical one. **1** = All eight bits of buffer map 1 are to be written with ones during CPU write if write mode is 0 or 3 (See write mode bits GRA05[1:0]), and if the enable set/reset bit GRA01[1] is a logical one.		
2	-	Set/Reset Map 2 **0** = All eight bits of buffer map 2 are to be written with zeros during CPU write if write mode is 0 (See write mode bits GRA05[1:0]), and if the enable set/reset bit GRA01[2] is a logical one. **1** = All eight bits of buffer map 2 are to be written with ones during CPU write if write mode is 0 or 3 (See write mode bits GRA05[1:0]), and if the enable set/reset bit GRA01[2] is a logical one.		

GRAPHICS CONTROLLER REGISTERS				
3CFh (R/W)	Set/Reset Register (GRA00)			
	VGA	-	-	-

Bit	Mnemonic	Description
3	-	Set/Reset Map 3 **0** = All eight bits of buffer map 3 are to be written with zeros during CPU write if write mode is 0 (See write mode bits GRA05[1,0]), and if the enable set/reset bit GRA01[3] is a logical one. **1** = All eight bits of buffer map 3 are to be written with ones during CPU write if write mode is 0 or 3 (See write mode bits GRA05[1:0]), and if the enable set/reset bit GRA01[3] is a logical one.
7:4	-	Reserved

GRAPHICS CONTROLLER REGISTERS				
3CFh (R/W)	**Enable Set/Reset Register (GRA01)**			
	VGA	**-**	**-**	**-**
Bit	**Mnemonic**	**Description**		
0	-	Enable Set/Reset Map 0 **0** = If write mode is 0 (GRA05[1:0]=0) CPU data is written to memory map 0. **1** = If write mode is 0 (GRA05[1:0]=0) GRA00[0] is written to all eight bits of memory map 0.		
1	-	Enable Set/Reset Map 1 **0** = If write mode is 0 (GRA05[1:0]=0) CPU data is written to memory map 1. **1** = If write mode is 0 (GRA05[1:0]=0) GRA00[1] is written to all eight bits of memory map 1.		
2	-	Enable Set/Reset Map 2 **0** = If write mode is 0 (GRA05[1:0]=0) CPU data is written to memory map 2. **1** = If write mode is 0 (GRA05[1:0]=0) GRA00[2] is written to all eight bits of memory map 2.		
3	-	Enable Set/Reset Map 3 **0** = If write mode is 0 (GRA05[1:0]=0) CPU data is written to memory map 3. **1** = If write mode is 0 (GRA05[1:0]=0)GRA00[3] is written to all eight bits of memory map 3.		
7:4	-	Reserved		
Note: This register has no effect on data source select when video memory map write mode is 1, 2, or 3.				

GRAPHICS CONTROLLER REGISTERS				
3CFh (R/W)	Color Compare Register (GRA02)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
3:0	-	Color Compare Map bits 3-0 • In Read mode (GRA05[3] being logical one), the four bits from this register are compared with the 4-bit PEL value (made up of one bit from each map), from bit positions 0 to 7. • As long as the Color Don't Care bits (GRA07[0:3]) for the respective maps are logical ones, the compare takes place only on those bits of the PEL value, and CPU reads a one for a match in that bit position. • If Color Don't Care bit for one map is logical zero, the latched data from that map is excluded from the compare, and only the remaining three bits are compared to generate the bus data.		
7:4	-	Reserved		

GRAPHICS CONTROLLER REGISTERS				
3CFh (R/W)	Data Rotate Register (GRA03)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
2:0	-	Rotate Count Bits 2-0 • Specifies the number of bit positions the CPU data is to be rotated to the right, before doing the function selected by bits 3 and 4 above and subsequent bit mask select and write operations. • Rotation is carried out only in write modes 0 and 3. In these two modes, the CPU data is rotated first, then operated on by the function bits GRA03[4:3], then updated by the bit mask register GRA05.		
4:3	-	Function Select Bits 1 and 2 **00** = CPU data replaces latched data. **01** = CPU data ANDed with latched data. **10** = CPU data ORed with latched data. **11** = CPU data XORed with latched data. These functions are performed on the CPU data before the selected bits are updated by the bit mask register, and then written to the display buffers.		
7:5	-	Reserved		

GRAPHICS CONTROLLER REGISTERS				
3CFh (R/W)	Read Map Select Register (GRA04)			
	VGA	-	-	-

Bit	Mnemonic	Description
1:0	-	Bits 1 and 2, respectively • Read mode 0 only: GRA controller returns the contents of one of the four latched buffer bytes to CPU each time a CPU read loads the latches. These two bits (0 and 1) define a value that represents the bit map where CPU is to read data — useful in transferring bit map data between the maps and system RAM.
7:2	-	Reserved

Notes:

1. In Odd/Even modes, the value may be binary 00 or 01 for chained bit maps 0 and 1.
2. In mode 13h, where all maps are chained to form one map, and in read mode 1, this register is ignored.

GRAPHICS CONTROLLER REGISTERS				
3CFh (R/W)	Graphic Mode Register (GRA05)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
1:0	-	Write Mode **00** = The CPU data byte can be written to video buffers map data latches in two dimensions: 1) Byte-oriented: to update any or all maps. 2) Pixel-oriented: to update any or all eight pixels using a predefined pixel value. • Updates are controlled using values in the internal registers of this graphics controller, namely GRA00-GRA08. • If enable set/reset bits are all zeros, CPU data updates the latches according to the function bits GRA03[4:3], and each map is updated as masked by GRA08[7:0]. **01** = Each map is written with the contents of its respective latches. These latches are loaded by a previous CPU memory read operation. **10** = Pixel-oriented: The four low-order bits of the CPU data are combined with the pixel values from the maps according to the functions specified by GRA03[4:3], and each map is updated as masked by GRA08[7:0]. **11** = Pixel-oriented, write mode 3 involves the following data manipulations: 1) CPU data is rotated by GRA03[2:0], then logical ANDed with the Bit Mask register bits GRA08[7:0]. The result is an 8-bit mask for use in write mode 3, to determine which pixels (from step 2 below) are to be updated by the set/reset value, and which pixels are updated directly from the latches. 2) The set/reset pixel values are produced as follows: The set/reset bits GRA00[0:3] are compared with each pixel value from the latches according to function bits GRA03[4:3].		
2	-	Reserved		

GRAPHICS CONTROLLER REGISTERS				
3CFh (R/W)	Graphic Mode Register (GRA05)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
3	-	Read Mode **0** = Byte-oriented: CPU reads the memory map specified by the Read Map Select register GRA04 unless SEQ04[3] is logical one (Chain). In case SEQ04[3] is logical one, CPU address bits A0 and A1 are used to read the specified memory map. **1** = Pixel-oriented, 4-bit value: The value is made up of one bit from each map. CPU reads the result of the comparison of this pixel value ANDed with the 4-bit color compare register value. If a bit in the Color Don't Care register (GRA07) is zero, that bit position is excluded from the compare. A match causes that position in the byte to be read out by CPU as a one. This process is repeated for all eight pixels.		
4	-	Odd/Even Addressing Enable • Used to enable CGA emulation, this bit enables odd/even addressing mode when it is logical one. Normally, this bit and memory mode bit SEQ04[2] are set to agree with each other in enabling odd/even mode emulation.		

GRAPHICS CONTROLLER REGISTERS				
3CFh (R/W)	Graphic Mode Register (GRA05)			
	VGA	-	-	-

Bit	Mnemonic	Description
6:5	-	Bit 6 = 256-color Mode Bit 5 = Shift Register Mode • These bits control how data from memory is loaded into the shift registers. M0D0:M0D7, M1D0:M1D7, M2D0:M2D7, and M3D0:M3D7 are representations of this data. The LSB bits are shifted out first: **00** = MSB LSB O/P M0D0 M0D1 M0D2 M0D3 M0D4 M0D5 M0D6 M0D7 →C0 M1D0 M1D1 M1D2 M1D3 M1D4 M1D5 M1D6 M1D7 → C1 M2D0 M2D1 M2D2 M2D3 M2D4 M2D5 M2D6 M2D7 →C2 M3D0 M3D1 M3D2 M3D3 M3D4 M3D5 M3D6 M3D7 →C3 **01** = MSB LSB O/P M1D0 M1D2 M1D4 M1D6 M0D0 M0D2 M0D4 M0D6 →C0 M1D1 M1D3 M1D5 M1D7 M0D1 M0D3 M0D5 M0D7 →C1 M3D0 M3D2 M3D4 M3D6 M2D0 M2D2 M2D4 M2D6 →C2 M3D1 M3D3 M3D5 M3D7 M2D1 M2D3 M2D5 M0D7 →C3 **10** = When GRA05[6] = 1, bit 5 is ignored — maps 0:3 data is consequently read as packed pixels. **11** = When GRA05[6] = 1, bit 5 is ignored — maps 0:3 data is consequently read as packed pixels.
7	-	Reserved

GRAPHICS CONTROLLER REGISTERS				
3CFh (R/W)	Graphics Miscellaneous Register (GRA06)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
0	-	Graphics/*Alphanumeric Mode **0** = Selects A/N (alphanumeric mode): display data bypasses the graphics controller and latches into the attribute controller. **1** = Selects APA (graphics) mode: color data is serialized in the shift registers before it is passed to the attribute controller .		
1	-	Chain Odd Maps to Even **1** = CPU address bit A0 is replaced by a higher order address bit. Even maps (0 and 2) are selected when A0 = zero; odd maps are selected when A0 = one.		
3:2	-	Memory Map Read Bits 1 and 0, respectively **00** = Maps the display buffer into processor address A0000h for 128K bytes. **01** = Maps the display buffer into processor address A0000h for 64K bytes. **10** = Maps the display buffer into processor address B0000h for 32K bytes. **11** = Maps the display buffer into processor address B8000h for 32K bytes.		
7:4	-	Reserved		

GRAPHICS CONTROLLER REGISTERS				
3CFh (R/W)	Color Don't Care Register (GRA07)			
	VGA	-	-	-

Bit	Mnemonic	Description
0	-	Ignore Map 0
1	-	Ignore Map 1
2	-	Ignore Map 2
3	-	Ignore Map 3
7:4	-	Reserved

Notes:

1. A byte is latched from each memory map in a CPU read, mode 1. The color value of a pixel (PEL) is made up of a bit from each map. The 4-bit PEL value is ANDed with the four bits from this register.

2. Any bit (map x) indicated by a logical zero in this register causes the corresponding bit in the PEL value to exclude itself from the comparison with the color compare bits. The remaining bits are ANDed with the 4-bit color compare register, where a match produces a logical one for that bit position in the CPU data byte as read data.

3. For example, if register value is "1111", the entire 4-bit PEL value is compared with the color compare bits. If any bit position matches, a logical one in the corresponding bit position is generated, as CPU read data.

GRAPHICS CONTROLLER REGISTERS				
3CFh (R/W)	Bit Mask Register (GRA08)			
	VGA	-	-	-

Bit	Mnemonic	Description
7:0	-	**0** = Data is from latches: Logical zero in a bit position preserves the memory content of the four maps in the same bit position. **1** = Data is from CPU byte: Logical one in a bit position allows updating of the four map bits that are in the same bit position. This register is used directly in write modes 0-2 only. Bit masking in write mode 3 involves the CPU data, which is described in *register GRA05.*

ATTRIBUTE CONTROLLER REGISTERS				
3C1h (R) 3C0h (W)	**ATTR Index Register (ATTRX)**			
	VGA	**-**	**-**	**-**
Bit	**Mnemonic**	**Description**		
4:0	-	ATTR Index Bits 4-0 • This index points to one of the internal registers of the attribute controller (ATTR) at addresses 3C1h/3C0h, for the next ATTR read/write operation. Since both the index and data registers are at the same I/O port, a pointer to the registers is necessary. This pointer can be initialized to point to the index register by a read instruction to the GENS1 register.		
5	-	Palette Address Source **0** = Allows the processor to load the color palette registers. **1** = Allows memory data to access the color palette registers. After loading the color palette, this bit should be set to logical one.		
7:6	-	Reserved		

Notes:

1. After initialization, OUT commands toggle between writing to the ATTRX and the indexed Attribute registers.

2. The Attribute registers operate with the Palette registers to establish the video DAC PEL definition.

ATTRIBUTE CONTROLLER REGISTERS				
3C1h (R) 3C0h (W)	Palette Registers 0-F (ATTR00-0F)			
	VGA	-	-	-

Bit	Mnemonic	Description
5:0	-	Color Bits 5-0 • Bits 0-5 map the text attribute or graphic color input value to a display color on the screen. Color is disabled for those bits that are set to logical zero; enabled for those bits set to logical one.
7:6	-	Reserved

Notes:

1. The two high-order bits of a 6-bit palette register content are stored in ATTR14[3:2].
2. Color bits 4 and 5 are substituted by ATTR14[1:0] when color source select ATTR10[7] is logical one.
3. In all modes except 256-color mode, pre-mapped 4-bit pixel values are used as addresses into the 16 ATTR palette registers. These internal registers allow 16 colors to be displayed simultaneously. The actual color output is the content of these registers.
4. In 256-color mode, where 256 colors can be displayed simultaneously, these registers are used only to index into the external registers, also called the DAC color table, where the color output values are stored.
5. Modification of these 16 internal palette registers enables the user to access 64 different addresses in the DAC color table.

ATTRIBUTE CONTROLLER REGISTERS				
3C1h (R) 3C0h (W)	Mode Control Register (ATTR10)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
0	-	Graphics/*Alphanumeric Mode **1** = Selects monochrome display. **0** = Selects A/N: alphanumeric mode. **1** = Selects APA: graphics mode.		
1	-	Monochrome/*Color Attributes Select **0** = Selects color display. **1** = Graphics/*Alphanumeric Mode		
2	-	Line Graphics Enable **0** = Sets the ninth dot to the background color: mandatory for character fonts that do not use the line graphics character codes (C0h-DFh). **1** = Enables the special line graphics character codes for monochrome emulation, and sets the ninth dot of a line graphics character to be the same as the eighth dot.		
3	-	Blink Enable/*Background Intensity **0** = Allows bit 7 of the character attribute to control background intensity. **1** = Allows bit 7 of the character attribute to control blinking.		
4	-	Reserved		
5	-	PEL Panning Compatibility **0** = Allows both halves of a split screen to pan together by preventing a line compare split screen function from affecting the output of PEL panning register ATTR13 and byte panning bits CRT08[6:5] **1** = For panning only the top half of a split screen: by forcing ATTR13 output to zero until the start of the next V sync pulse when line compare condition is "true".		
6	-	PEL Clock Select **0** = Shift registers are clocked every dot clock. **1** = For 256-color mode 13h: eight bits of video data are packed to form a pixel.		

ATTRIBUTE CONTROLLER REGISTERS				
3C1h (R) 3C0h (W)	Mode Control Register (ATTR10)			
	VGA	-	-	-

Bit	Mnemonic	Description
7	-	Alternate Color Source **0** = Selects palette register bits 4 and 5 (in ATTR00-0F) as source for color output bits P4 and P5. **1** = Selects ATTR14[1:0] as source for color output bits P4 and P5, respectively.

ATTRIBUTE CONTROLLER REGISTERS				
3C1h (R) 3C0h (W)	Overscan Color Register (ATTR11)			
	VGA	-	-	-

Bit	Mnemonic	Description
7:0	-	Bits 7 - 0

Notes:

1. These bits define the color of the border (overscan) area in 80-column modes. Overscan borders are not supported in 40-column modes.

2. Refer to the description and notes for registers ATTR00-0F for information regarding how the color bits are substituted: bits 6 and 7, ATTR14[3:2], and bits 4 and 5, ATTR14[1:0].

ATTRIBUTE CONTROLLER REGISTERS				
3C1h (R) 3C0h (W)	Color Map Enable Register (ATTR12)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
3:0	-	Enable Color Map bits 3-0 **0** = Disables data from maps 3-0 to be used for video output. **1** = Enables data from a specific map, maps 3-0, to be used for video output.		
5:4	-	Video Status Mux bits 0-1 • These are control bits for the multiplexer on color bits P0-P7. The bit selection is also indicated at GENS1[5,4] as follows: **00** = P2, P0. **01** = P5, P4. **10** = P3, P1. **11** = P7, P6.		
7:6	-	Reserved		

ATTRIBUTE CONTROLLER REGISTERS

3C1h (R) 3C0h (W)	Horizontal PEL Panning Register (ATTR13)			
	VGA	-	-	-

Bit	Mnemonic	Description
3:0	-	Shift Count Bits 3-0 • The shift count value (0-8) indicates how many pixel positions to shift left.

COUNT VALUE	MODES 0+, 1+, 2+, 3+, 7, 7+	MODE 13	ALL OTHER MODES
0	1	0	0
1	2	-	1
2	3	1	2
3	4	-	3
4	5	2	4
5	6	-	5
6	7	3	6
7	8	-	7
8	0	-	-

7:4	-	Reserved

Note: A/N modes 0+, 1+, 2+, 3+, and 7+ are enhanced modes with 9x16 box size resolution. A/N mode 7 has a 9x14 box size. APA mode 13 has a 320x200 screen resolution.

ATTRIBUTE CONTROLLER REGISTERS				
3C1h (R) 3C0h (W)	Color Select Register (ATTR14)			
	VGA	-	-	-

Bit	Mnemonic	Description
1:0	-	Color bits P5 and P6, respectively. These bits are the color output bits (instead of bits 5 and 4 of the internal palette registers ATTR00-0F) when alternate color source, bit ATTR10[7], is logical one.
3:2	-	Color bits P7 and P6, respectively. These two bits are the two high-order bits of the 8-bit color, used for rapid color set switching (addressing different parts of the DAC color lookup table). The lower order bits are in registers ATTR00-0F.
7:4	-	Reserved

This page intentionally left blank.

6

VGA Register Extensions

Configuring VGA Extended Registers

I/O address and offset for ATI's VGA extended registers are user-selectable. This feature allows users to pick an unoccupied I/O address for the extended registers when there is I/O port conflict with other peripheral hardware. Normally, on system reset, the BIOS writes the values for *I/O Address* (A11:A0) and *Offset* (O1:O0) to locations 3CEh offset 50h and 51h. These registers are write-only.

The default values for address and offset are "1CE" and "2" respectively, as illustrated below:

Address: 3CEh Offset: 50h

Bit 7							Bit 0
A7	A6	A5	A4	A3	A2	A1	A0

Address: 3CEh Offset: 51h

Bit 7							Bit 0
O1	O0	"0"	"0"	A11	A10	A9	A8

The following code will program "1CE" as the I/O address, and "2" as the index offset:

```
MOV  DX, 3CE
MOV  AX, CE50      ;CE = address bits A7 to A0
OUT  DX, AX
MOV  AX, 8151      ;81 = offset bits 1, 0 + two "0", and address bits A11 to A8
OUT  DX, AX
```

VGA Extended Registers — by Name

Name	Type	Description	Page
ATIX	R/W	ATI Index	6-3
ATI01	R/W	ATI Register 1	6-5
ATI02	R/W	ATI Register 2	6-6
ATI03	R/W	ATI Register 3	6-7
ATI04	R/W	ATI Register 4	6-8
ATI05	R/W	ATI Register 5	6-9
ATI06	R/W	ATI Register 6	6-10
ATI20	R/W	ATI Register 20	6-11
ATI23	R/W	ATI Register 23	6-12
ATI24	R/W	ATI Register 24	6-13
ATI25	R/W	ATI Register 25	6-14
ATI26	R/W	ATI Register 26	6-15
ATI28	R	ATI Register 28	6-16
ATI29	R	ATI Register 29	6-17
ATI2B	R/W	ATI Register 2B	6-18
ATI2D	R/W	ATI Register 2D	6-19
ATI2E	R/W	ATI Register 2E	6-20
ATI30	R/W	ATI Register 30	6-21
ATI31	R/W	ATI Register 31	6-22
ATI32	R/W	ATI Register 32	6-23
ATI33	R/W	ATI Register 33	6-24
ATI34	R/W	ATI Register 34	6-25
ATI35	R/W	ATI Register 35	6-26
ATI36	R/W	ATI Register 36	6-27
ATI37	R	ATI Register 37	6-28
ATI38	R/W	ATI Register 38	6-29
ATI39	R/W	ATI Register 39	6-30
ATI3A	R/W	ATI Register 3A	6-31
ATI3B	R/W	ATI Register 3B	6-32
ATI3C	R	ATI Register 3C	6-33
ATI3D	R/W	ATI Register 3D	6-34
ATI3E	R/W	ATI Register 3E	6-35
ATI3F	R/W	ATI Register 3F	6-36

ATI EXTENDED REGISTERS				
*(R/W)	ATI Index Register (ATIX)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
0	-	Index Bit I0		
1	-	Index Bit I1		
2	-	Index Bit I2		
3	-	Index Bit I3		
4	-	Index Bit I4		
5	-	Index Bit I5		
6	-	Offset Bit O0		
7	-	Offset Bit O1		

*The I/O address and offset are programmed as illustrated on page 1. ***ATI Extended registers must have both I/O address and offset configured before being accessed.***

Note:

1. Assuming that the I/O address for the extended registers has been configured as illustrated in the example at the beginning of this chapter, i.e., an address of 1CEh with an index offset of 2h, the steps for writing a value of "FF" to the extended register at index "3E" (ATI3E) are as follows:

   a) Obtain the offset for ATI3E by concatenating the index offset "2" (bits O1-O0) to the register index "3E" (bits I5-I0). For this example, it is "1011 1110" (BE):

   Address: 1CEh **Offset:** 2h

   Bit 7 **Bit 0**

O1	O0	I5	I4	I3	I2	I1	I0

   b) Write "FF" to the extended register ATI3E by specifying "FFBE" in the program code as follows:

```
MOV  DX, 1CE
MOV  AX, FFBE
OUT  DX, AX
```

ATI EXTENDED REGISTERS				
*(R/W)	ATI Register 0 (ATI00)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
7:0	-	Scratch Pad Bits 7:0		

Note:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

ATI EXTENDED REGISTERS				
*(R/W)	ATI Register 1 (ATI01)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
7:0	-	Scratch Pad Bits 7:0		

Note:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

ATI EXTENDED REGISTERS				
*(R/W)	ATI Register 2 (ATI02)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
7:0	-	Scratch Pad Bits 7:0		

Note:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

ATI EXTENDED REGISTERS				
*(R/W)	ATI Register 3 (ATI03)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
7:0	-	Scratch Pad Bits 7:0		

Notes:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

ATI EXTENDED REGISTERS				
*(R/W)	ATI Register 4 (ATI04)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
7:0	-	Scratch Pad Bits 7:0		

Notes:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

ATI EXTENDED REGISTERS				
*(R/W)	ATI Register 5 (ATI05)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
3:0	-	CPUCLK Select bits 3:0. • Selects number of CPU clocks for the basic command cycle in the local bus. Power-up default is 6 CPU clocks.		
5:4	-	Delay memory read ready control bits 1:0. **00** Read Ready signal is 1 MCLK before memory data is available. **01** Read Ready signal is active at the same time as memory data. **10** Read Ready signal is 1 MCLK after memory data is available. **11** Read Ready signal is 2 MCLK after memory data is available.		
6	-	Delay latch memory read data in VGA planar mode by half the period of the memory clock.		
7	-	Cursor blink rate select **0** = Normal blink rate (VGA standard) **1** = Half normal blink rate.		

Notes:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

2. See Appendix G for register definition for the ATI 68800LX controller.

3. ATI05[3:0] can be programmed only when ATI2E[4] is logical zero.

ATI EXTENDED REGISTERS				
*(R/W)	ATI Register 6 (ATI06)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
2:0	-	Text mode character FIFO depth. Power-up default is 2.		
3	-	Reserved		
6:4	-	Programs the number of CPU CLK in the BIOS ROM read cycle in local bus. Power-up default is 3.		
7	-	Reserved		

Notes:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

2. See Appendix G for register definition for the ATI 68800LX controller.

3. Bits 6:4 of this register can only be programmed when bit 4 of register ATI24 is a zero.

ATI EXTENDED REGISTERS				
*(R/W)	ATI Register 20 (ATI20)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
3:0	-	Display FIFO Bits 3:0 • These bits select the video FIFO depth at which Display Request changes from low priority to high priority in the memory controller. Power-up default is 8.		
4	-	16-bit ROM Access **1** = Enables 16-bit ROM access.		
6:5	-	RAMDAC extended address select bits RS3:2		
7	-	Reserved		

Note:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

2. See Appendix G for register definition for the ATI 68800LX controller.

ATI EXTENDED REGISTERS				
*(R/W)	ATI Register 23 (ATI23)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
2:0	-	16-Bit ROM Access Bits 2:0 • ROM Access Time bit (in single ROM, 16-bit mode).		
3	-	ATI-Ext CRTC CA Bit 17 • This is bit 17 of the cursor start address (CA) when in ATI extended modes. See CRT0E and CRT0F for descriptions.		
4	-	ATI-Ext CRTC SA Bit 17 • This is bit 17 of the display buffer start address (SA) when in ATI extended modes. See CRT0E and CRT0F for descriptions.		
7:5	-	Reserved.		

Note:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

ATI EXTENDED REGISTERS				
*(R/W)	ATI Register 24 (ATI24)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
0	-	ROM Page 0 Bit 0		
1	-	ROM Page 0 Bit 1		
2	-	ROM Page 0 Bit 2		
3	-	ROM Page 0 Bit 3		
4	-	ROM Page 1 Bit 0		
5	-	ROM Page 1 Bit 1		
6	-	ROM Page 1 Bit 2		
7	-	ROM Page 1 Bit 3		

Note:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

ATI EXTENDED REGISTERS				
*(R/W)	ATI Register 25 (ATI25)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
0	-	ROM Page 2 Bit 0		
1	-	ROM Page 2 Bit 1		
2	-	ROM Page 2 Bit 2		
3	-	ROM Page 2 Bit 3		
4	-	ROM Page 3 Bit 0		
5	-	ROM Page 3 Bit 1		
6	-	ROM Page 3 Bit 2		
7	-	ROM Page 3 Bit 3		

Note:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

ATI EXTENDED REGISTERS				
*(R/W)	ATI Register 26 (ATI26)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
0	-	Display Enable Skew-by-2 **1** = Skews "Display Enable" by 2 characters.		
2:1	-	Reserved		
3	-	General Purpose R/W Bit		
5:4	-	Reserved		
6	-	Solid Underline **0** = Dashed underline in monochrome text mode. **1** = Solid Underline in monochrome text mode.		
7	-	Forced Read 3CCh **0** = Normal read back operation. **1** = Force data bits GENMO[7:1] to logical zero during read operation of 3CCh.		

Note:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

ATI EXTENDED REGISTERS				
*(R)	ATI Register 28 (ATI28)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
0	-	Vertical Line Counter Bit 8		
1	-	Vertical Line Counter Bit 9		
7:2	-	Reserved		

Note:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

ATI EXTENDED REGISTERS				
*(R)	ATI Register 29 (ATI29)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
7:0	-	Vertical Line Counter Bits 7:0		

Note:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

ATI EXTENDED REGISTERS				
*(R/W)	ATI Register 2B (ATI2B)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
0	-	Video Zero Wait-State Enable **1** = Enable zero wait-state support for video memory write.		
1	-	BIOS Zero Wait-State Enable **1** = Enable zero wait-state support for BIOS read.		
2	-	I/O Zero Wait State Enable **1** = Enable I/O zero wait state.		
3	-	Double scan lock enable **1** = Lock the CRTC9[7] bit from being altered.		
4	-	Lock DAC Write **1** = Lock RAMDAC write select signal.		
5	-	Reserved		
6	-	Memory data delay latch in text mode **1** = delay latching of memory data from the DRAM port by 1/2 MCLK in text mode.		
7	-	Video data delay latch in text mode **1** = Delay internal latching of video data from the serial port by 1/2 MCLK in text mode.		

Note:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

2. See Appendix G for register definition for the ATI 68800LX controller.

ATI EXTENDED REGISTERS				
*(R/W)	ATI Register 2D (ATI2D)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
1:0	-	Bits 19:18 of the Extended Cursor Address.		
3:2	-	Bits 19:18 of the Extended Start Address.		
7:4	-	Bits 19:16 of the Extended Character Map Address.		

Note:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

ATI EXTENDED REGISTERS				
*(R/W)	ATI Register 2E (ATI2E)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
1:0	-	CPU Write Page Pointer bits 5:4		
3:2	-	CPU Read Page Pointer bits 5:4		
4	-	**0** = Enables programming of the CPUCLK Select **1** = Locks CPUCLK Select bits ATI05[3:0]		
7:5	-	Horizontal Sync Skew 2:0 by pixel clock		

Note:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

ATI EXTENDED REGISTERS				
*(R/W)	ATI Register 30 (ATI30)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
0	-	**1** = Skews Display Enable by one character clock		
1	-	Reserved		
2	-	ATI-Ext CRTC CA Bit 16 • This is bit 16 of the cursor start address (CA) when in ATI extended modes. See CRT0E and CRT0F for descriptions.		
3	-	1MB Memory Support **1** = Indicates 1MB video RAM is reserved for VGA.		
4	-	Video Memory Size **0** = 256KB video RAM is reserved for VGA **1** = 512KB video RAM is reserved for VGA		
5	-	ATI-Ext 256-Color Mode Select **1** = Enables 256-color mode in ATI extended mode		
6	-	ATI-Ext CRTC SA Bit 16 • This is bit 16 of the display buffer start address (SA) when in ATI extended modes. See CRT0C and CRT0D for descriptions.		
7	-	Reserved		

Note:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

ATI EXTENDED REGISTERS				
*(R/W)	ATI Register 31 (ATI31)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
0	-	EGA I/O Address Compatibility **1** = Forces all VGA I/O port addresses to be EGA-compatible.		
1	-	EGA Register Compatibility **1** = Forces all VGA registers to be EGA-register-compatible.		
2	-	General Purpose R/W Bit • Read/write bit. Any status can be stored here and read back later.		
5:3	-	Scan Function Bits 2:0 **000** = Normal scanning. **001** = APA mode: Enables double scanning in lieu of CRT09[7]. **010** = APA mode: Enables 3 of 4 scanning. **101** = A/N mode: Enables double scanning in lieu of CRT09[7]. **110** = A/N mode: Enables 3 of 4 scanning.		
6	-	V Timings Divide-by-2 **0** = Vertical timings at current clock rate. **1** = Divides vertical timing parameters by 2.		
7	-	Reserved		

Note:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

ATI EXTENDED REGISTERS				
*(R/W)	ATI Register 32 (ATI32)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
0	-	CPU Address Read Paging bit — used with bits 7:5, must be set to logical zero.		
4:1	-	**0000** = 1st 64K block or 1st 128K block. **0001** = 2nd 64K block or 2nd 128K block. **0010** = 3rd 64K block or 3rd 128K block. **0011** = 4th 64K block or 4th 128K block. **0100** = 5th 64K block. **0101** = 6th 64K block. **0110** = 7th 64K block. **0111** = 8th 64K block. **1000** = 9th 64K block. • CPU Address Write Paging bits 3:0. • If ATI3E[3] is logical zero, bits ATI32[3:1] are used to control both CPU read and write paging.		
7:5	-	**000** = 1st 64K block or 1st 128K block. **001** = 2nd 64K block or 2nd 128K block. **010** = 3rd 64K block or 3rd 128K block. **011** = 4th 64K block or 4th 128K block. **100** = 5th 64K block. **101** = 6th 64K block. **110** = 7th 64K block. **111** = 8th 64K block. • CPU Address Read Paging bits 2:0. • If ATI3E[3] is logical one, bits ATI32[5:7] are used to control CPU read paging; while ATI32[1:3] are used to control CPU write paging. • Read/write bit. Any status can be stored here and read back later.		

Notes:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

2. CPU page addressing control is affected by ATI3D[2], which is for enabling 128K CPU addressing

ATI EXTENDED REGISTERS				
*(R/W)	ATI Register 33 (ATI33)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
0	-	EEPROM data input bit.		
1	-	EEPROM clock input bit.		
2	-	EEPROM Interface Enable		
3	-	EEPROM chip select input **0** = Disable EEPROM interface. **1** = Enable EEPROM interface.		
4	-	Reserved		
5	-	ISA bus 8-/16-Bit Video Memory Operation • *When pin RMCE1B is **not** grounded through a 2KΩ resistor:* **0** = Defaults to 8-bit video memory operation. **1** = Enables 16-bit video memory operation. • *When pin RMCE1B **is** grounded through a 2KΩ resistor:* **0** = Selects 16-bit video memory operation. **1** = Selects 8-bit video memory operation.		
6	-	4-bit PEL, 1 bit/map **1** = Enables map memory mapping for 16-color 1024x768 mode 55h.		
7	-	Double Scan Enable **0** = Scans at the selected clock rate. **1** = Enables double scanning when the screen size is programmed to be between 150 to 250 lines.		

Note:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

2. See Appendix G for register definition for the ATI 68800LX controller.

ATI EXTENDED REGISTERS				
*(R/W)	ATI Register 34 (ATI34)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
0	-	Enable CGA Emulation **1** = Enables CGA emulation.		
1	-	Enable Hercules Emulation **1** = Enables Hercules emulation.		
2	-	Write Protect CRT09[0:4, 7] **1** = Write protects registers CRT09[0:4, 7].		
3	-	Write Protect V Timing Registers **1** = Write protects all vertical timing related registers: CRT06 CRT07[0-3,5-7] CRT09[5] CRT10, CRT11[0-3] CRT12 CRT15 CRT16.		
4	-	Write Protect CRT0A-CRT0B **1** = Write protects the cursor start/end registers CRT0A/CRT0B.		
6	-	Write Protect CRT00-CRT07 **1** = Replaces CRT11[7] as the write protect bit for registers CRT00-CRT07 (except bit 4 of CRT07, which is not protected).		
5	-	Write Protect CRT08[0-6], CRT14[0-4] **1** = Write protects registers CRT08[0:6] and CRT14[0:4].		
7	-	CRT11[7] Override **1** = Write protect bit CRT11[7] is ignored. See *CRT11[7]* for list of registers to protect from a write operation.		

Note:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

ATI EXTENDED REGISTERS				
*(R/W)	ATI Register 35 (ATI35)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
0	-	Blanking Signal Select **1** = Selects the Display Enable signal as the output blanking signal.		
1	-	Blanking Polarity Invert **1** = Inverts polarity of blanking signal BLANKB.		
2	-	Display Enable Signal Skewed **1** = Skews the Display Enable signal by one PEL clock.		
3	-	General Purpose R/W Bit **1** = Read/write bit. Any status can be stored here and read back later.		
4	-	Anti-alias Fonts Enable **1** = Enables eight simultaneous fonts.		
5	-	Cursor Blinking Disable **1** = Disables cursor blinking		
6	-	CGA Cursor Start/End Address **1** = Adds a value of '5' to both cursor start/end registers, for CGA emulation.		
7	-	VGA Overscan Output Enable **1** = Used as an overlay input signal to the RAMDAC to generate an overscan feature that is independent of the RAMDAC palette values. (ATI68800-6)		

Note:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

2. See Appendix G for register definition for the ATI 68800LX controller.

ATI EXTENDED REGISTERS				
*(R/W)	ATI Register 36 (ATI36)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
0	-	CRTC Display Address Counter Enable 1 = Enables VGA CRTC display address counter above 64KB address space.		
1	-	640x400 Hercules Graphics Emulation 1 = Enables 640x400 Hercules graphics mode.		
2	-	Linear Addressing, 256-color Mode 1 = Selects linear addressing, 256-color mode.		
3	-	General Read/Write Bit		
4	-	16-Color Enable, APA Mode 1 = Enables high resolution graphics modes with 16 simultaneous colors.		
5	-	Vertical Interrupt Enable 1 = Enables vertical interrupt.		
6	-	Linear Addressing, Text Mode 1 = Selects linear addressing, text mode.		
7	-	Screen Blanking Disable (CGA/Hercules Modes) 1 = Disables function of bit 3 in the CGA and Hercules mode control registers GENMC (at I/O ports 3D8h & 3B8h respectively) to prevent screen blanking when in these modes.		

Note:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

ATI EXTENDED REGISTERS				
*(R)	ATI Register 37 (ATI37)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
0	-	Status of ISA bus 16-bit Operation Select **0** = Selects 8-bit operation. **1** = Selects 16-bit operation.		
2:1	-	Reserved		
3	-	Output Data - EEPROM/input		
4	-	Status of ROM Address Decode Enable/Disable Bit **0** = Disables ROM address decode bit. **1** = Enables ROM address decode bit.		
7:5	-	Reserved		

Notes:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

2. On a read, all above described bits except bit 3 are external inputs during system reset.

ATI EXTENDED REGISTERS				
*(R/W)	ATI Register 38 (ATI38)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
0	-	Write Protect ATTR00 - 0F **1** = Disable I/O writes to ATTR00 - 0F palette registers.		
1	-	Write Protect ATTR11 **1** = Disable I/O writes to ATTR11 overscan register.		
2	-	Write protect VGA registers **1** = Disable I/O writes to all VGA registers except: • CRTC0C, CRTC0D (Start Address registers). • CRTC0A (Cursor Start register) • CRTC0B (Cursor End register).		
3	-	Write Protect Register at I/O port 3C2h **1** = Disable I/O writes to 3C2h register.		
4	-	Reserved		
5	-	640x300 Hercules graphics emulation **1** = enable 640x300 Hercules graphics mode.		
6	-	**0** = Input video clock undivided. **1** = Input video clock divided by 2.		
7	-	General Purpose Read/Write bit.		

Note:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

ATI EXTENDED REGISTERS				
*(R/W)	ATI Register 39 (ATI39)			
	VGA	-	-	-

Bit	Mnemonic	Description
0	-	Reserved
1	-	General Purpose Read/Write bit.
3:2	-	ROM Address Decode Bits 1 and 0, respectively **00** = Enables 32K BIOS size, starting at C0000h. **01** = Enables 28K BIOS size, starting at C0000h. **10** = Enables 24K BIOS size, starting at C0000h. **11** = Enables 24K BIOS size, starting at C0000h.
5:4	-	General Purpose Read/Write bit.
6	-	16-bit I/O Operation **0** = Enables 8-bit I/O operation. **1** = Enables 16-bit I/O operation.
7	-	Write Protects CRT18 (Line Compare) **1** = Disables write to CRT18 (Line Compare register)

Note:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

ATI EXTENDED REGISTERS				
*(R/W)	ATI Register 3A (ATI3A)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
2:0	-	General Purpose Read/Write bits		
7:3	-	Reserved		

Note:

1. The I/O port addresses* of ATI Extended registers are user programmable. Refer to *Note 1* in the ATIX register description.

ATI EXTENDED REGISTERS				
*(R/W)	ATI Register 3B (ATI3B)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
7:0	-	General Purpose Read/Write bits 7 to 0		

Notes:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

ATI EXTENDED REGISTERS				
*(R)	ATI Register 3C (ATI3C)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
7:0	-	Reserved, all bits must be 0		

Notes:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

ATI EXTENDED REGISTERS				
*(R/W)	ATI Register 3D (ATI3D)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
0	-	Composite Sync Polarity Select **1** = Selects Composite Sync Polarity output		
1	-	Reserved		
2	-	128KB CPU Address **1** = Enables CPU addressing of 128KB from A0000h to BFFFFh; all CPU Adrs Read page control bits in ATI32 have 128KB per block operations.		
3	-	Composite Sync Select **1** = Selects Composite Sync output instead of Horizontal Sync.		
7:4	-	SW3, SW2, SW1, SW0, respectively. • These bits emulate DIP switches SW3-SW0 on the IBM EGA adapter.		

Note:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

ATI EXTENDED REGISTERS				
*(R/W)	ATI Register 3E (ATI3E)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
0	-	R/W V Display End Register **1** = Allows programming of register CRT12 (Vertical Display End); even in Double Scan mode.		
1	-	Interlace Operation **1** = Enables interlace operation.		
2	-	Select Internal EGA DIP Switches **1** = Selects the internal EGA DIP switches ATI3B[4:7] as Switch Sense: instead of GENS0[4].		
3	-	Read/Write Paging Select **0** = Selects CPU read/write paging, as defined in ATI32[1:3]. **1** = Selects CPU read/write paging, as defined in ATI32[1:3, 5:7].		
4	-	General-purpose read/write bit		
7:5	-	Reserved		

Note:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

ATI EXTENDED REGISTERS				
*(R/W)	ATI Register 3F (ATI3F)			
	VGA	-	-	-
Bit	**Mnemonic**	**Description**		
0	-	**1** = Disables zero wait state in planar mode.		
3:1	-	**1** = Selects the number of MCLK delay in 16-bit Planar mode memory operation before latching the first CPU data.		
7:4	-	Reserved		

Note:

1. *The port address of ATI Extended registers are user programmable. Refer to *Note 1* on page 6-3 for details.

Part III

7

Coprocessor Functions

Logical Register Groupings

The coprocessors described in this chapter are ATI68800-3, ATI68800LX, ATI68800-6, and ATI68800AX. The differences among them is in the number of functions that they support — starting with the low-end "-3" leading up to the high-end "AX", all these accelerators are backward-compatible and they are all 8514/A-compatible.

- RAMDAC Control Registers — configure the DAC for specific pixel depths of 4, 8, 16, 24, or 32, and read/write the color lookup table for 4bpp and 8bpp modes.
- CRT Controller Registers — set up the CRT controller for display in a particular mode. They are based on the 8514/A CRT controller registers with extensions to select up to 16 pixel clocks.
- Status Registers — provide the current state of the FIFO, draw engine, CRT controller, or memory controller.
- Draw Engine Setup Registers — are set by the boot ROM according to the current adapter and system configuration. Once set, they should not be modified, with the exception of the memory boundary register, which is used to increase or decrease the memory available to the accelerator.
- Draw Engine Control Registers — are used for setting up the current engine context. They control the data path, scissor region, ALU function, and various draw options.
- Draw Registers — determine the size, trajectory, and type of draw operation to be performed. All draw commands are performed through a 16-word FIFO.
- Extended Features Registers — control various hardware features such as overscan, hardware cursor, and memory aperture.
- Access Control Registers — control the EEPROM access and custom configuration for specific platforms such as EISA.

Pixel Data Path

The output of the monochrome mux determines the color source and ALU functions. A logical one selects foreground color source and ALU function. A logical zero selects background color source and ALU function. This provides innate color expansion abilities.

When using monochrome host data in an ATI extended operation, data is packed into 8 pixels/byte. This differs from the 4 pixels/byte data format in an 8514/A-compatible operation. When using monochrome pattern data in an ATI extended drawing operation,

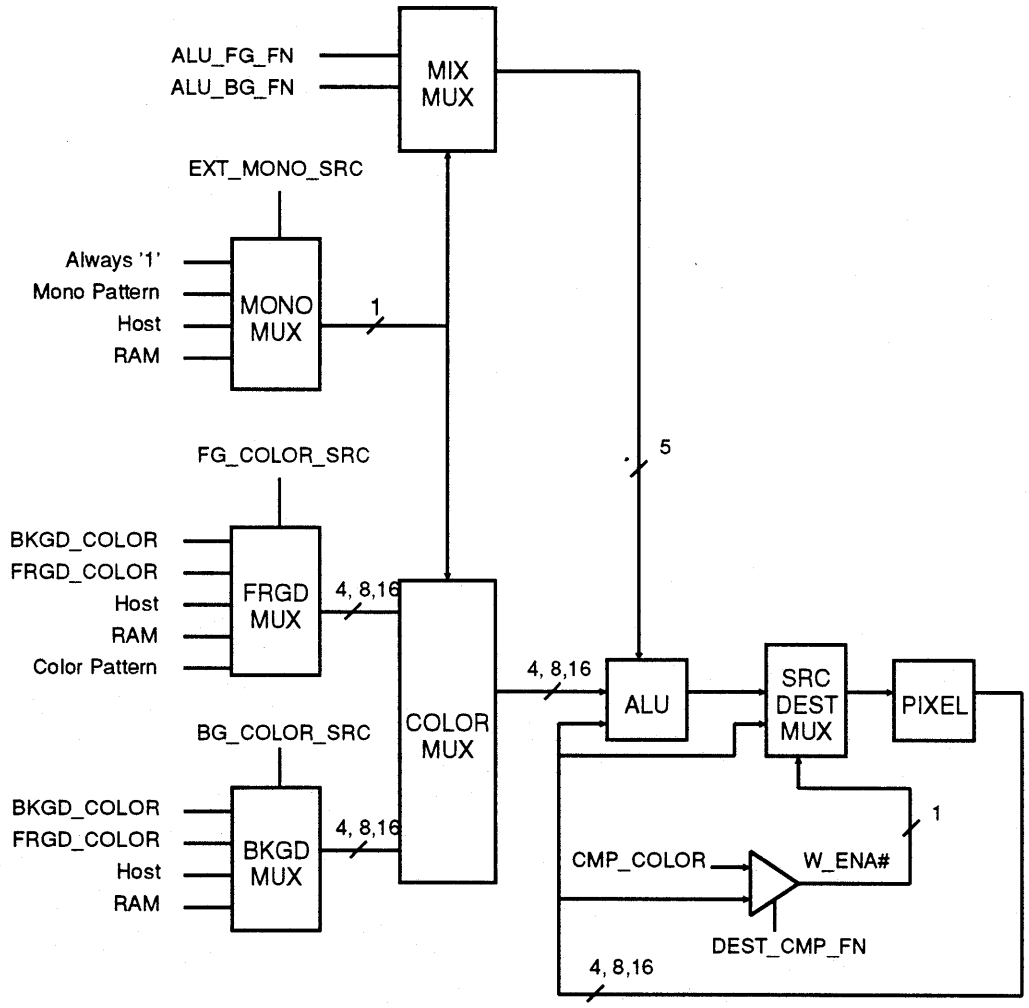


Figure 7-1. Pixel Data Path

pattern length is variable. This differs from the 8-pixel fixed-length data pattern in an 8514/A-compatible operation.

The 16bpp mode is both a drawing and display mode. It is similar to the 8bpp mode with the following exceptions:

- Color host data is defined in terms of 16 bit pixels. Therefore the DATAWIDTH field must be set for color host data transfers; and DATAORDER to swap bytes within each 16 bit pixel rather than swap the order of pixels.
- Three new destination color compare functions are defined for each of the RED, GREEN, and BLUE color fields within each 16 bit pixel. The result of the three tests are ANDed together and if TRUE will effect the drawing operation.
- The monochrome blit source, monochrome reads, and polygon outline fill operations use all 16 bit planes within each pixel after the 16 bit write mask and read mask have been applied.
- Display and drawing memory offset registers are defined in terms of bytes rather than pixels.
- 8514/A compatible monochrome reads are not available, use extended mono reads.
- VPIX and VNIB style rectangle fills are not available.

CRT Controller

The 18811-x clock chip has 16 selectable pixel clocks for maximum flexibility. The CRT controller can support any video mode that uses a dot clock from this list. VESA standard modes of 640x480, 800x600, 1024x768, and 1280x1024 are selectable.

For compatibility reasons, there are two logical CRT controllers on the ***mach32*** — they are **VGA*Wonder*** compatible and 8514/A-compatible with extensions. These extensions allow two custom video modes to be locked into the CRTC shadow sets, which are then selected using the 8514/A pixel clock select bit ADVFUNC_CNTL[2] (4AE8).

See Appendix E for pixel clock specifications.

Pixel Transfer ALU

The Pixel Transfer ALU has 16 logical and 16 arithmetic functions in 4bpp and 8bpp modes and 16 logical functions in 16bpp modes with the exception of (A+B)/2 on ATI68800-6 parts. The destination compare function operates on the individual RGB components of the color. Accelerator modes have 16 logical functions only. Function code definitions are listed on page 8-24.

Destination Pixel

For 4bpp and 8bpp modes, the destination pixel is overwritten when the result of the comparison function is false. For the 16bpp mode, the destination pixel is overwritten when the result of R, G, and B comparison functions ANDed together is false.

Comparison Functions	
0	False
1	True
2	Pixel ≥ CMP_COLOR
3	Pixel < CMP_COLOR
4	Pixel != CMP_COLOR
5	Pixel == CMP_COLOR
6	Pixel ≤ CMP_COLOR
7	Pixel > CMP_COLOR

(A+B)/2 Function

The (A+B)/2 ROP function is supported in 4bpp and 8bpp for all chip versions. For the ATI68800-6, it has been implemented in 16bpp drawing modes as well. This function is used for enhancing image quality in motion video applications. It is applied separately to the red, green, and blue components. It will produce different results depending on which of the 16bpp DAC mode is used.

Command FIFO

All 8514/A-compatible registers with an address greater than 0x8000 and all ATI extended registers are written to a 16-word command FIFO. Before performing an I/O write to any of these registers, the application must first check the FIFO for available entries.

For generic 8514/A applications, the developer may use the GE_STAT (9AE8) register to check for FIFO status. Since only the first eight entries are polled in this way, the developer is encouraged to use EXT_FIFO_STATUS (9AEE) to increase throughput to the draw engine.

Overrunning the FIFO or attempting to read from the host data transfer register (PIX_TRANS) when no data is available causes the FIFO to lock. A locked FIFO requires remedial action by the host to unlock it. Read operations do not use the FIFO.

Scissor Registers

The scissor registers are used for simple draw clipping. The 8514/A-compatible scissors have a range of [-1024..1023] for the top and left scissors and [0..2047] for the right and bottom scissors. The extended scissors have a range of [-2048..2047]. Drawing is inhibited outside the scissor region.

In certain situations, the application developer may be able to get performance improvements by pre-clipping the draw operation instead of using the scissor registers. All draw operations will take the same amount of time to complete even if they are mostly or entirely outside the scissor region, with the exception of trivially rejected line draws and clip exceptions (see *Line Clipping* on page 7-11).

All draw operations that are entirely or partially outside the maximum device coordinates [-512..1535], must be pre-clipped unless special line clipping provisions have been made (see *Line Clipping* on page 7-11).

Drawing Operations

All drawing operations are performed by first setting up the necessary control registers, then the trajectory control registers, and then initiating the drawing operation. Draw Engine Control and ALU Control registers as well as the drawing operations are summarized in the four tables as follows:

8514/A Compatible Draw Control Registers		
Register	**Description**	**Applies to**
FRGD_COLOR (A6E8)	Foreground color.	Draw operations which use this as a color source.
BKGD_COLOR (A2E8)	Background color.	Draw operations which use this as a color source.
FRGD_MIX (BAE8)	Set foreground source and ALU function.	All draw operations.
BKGD_MIX (B6E8)	Set background source and ALU function.	Draw operations when monochrome source is non-trivial.
WRT_MASK (AAE8)	Enable/disable write operations to individual bit planes.	All draw operations.
RD_MASK (AEE8)	Enable/disable planes for read operations.	Monochrome read, monochrome blit, and polygon fill types A and C.
SCISSOR_T (BEE8[1])	Top scissor.	All draw operations.
SCISSOR_L (BEE8[2])	Left scissor.	All draw operations.
SCISSOR_B (BEE8[3])	Bottom scissor.	All draw operations.
SCISSOR_R (BEE8[4])	Right scissor.	All draw operations.
PIX_CNTL (B2E8[A])	Set monochrome source, destination compare function, monochrome read or polygon fill, and polygon fill type.	Monochrome source and destination compare function applies to all 8514/A compatible draw operations. Mono_read/polyfill and polygon fill type only apply to monochrome reads or polygon fills.
COLOR_CMP (B2E8)	Comparison color.	Draw operations when destination compare function is non-trivial.
PATTERN_L (BEE8[8])	Low 4 bits of monochrome pattern.	Draw operations which use pattern as a monochrome source for an 8514/A compatible draw operation.
PATTERN_H (BEE8[9])	High 4 bits of monochrome pattern.	Draw operations which use pattern as a monochrome source for an 8514/A compatible draw operation.
CMD (9AE8[NO_OP])	Draw option bits, line option bits, host transfer option bits.	Short stroke vector operations.

ATI-Extended Draw Control Registers		
Register	**Description**	**Applies to**
DP_CONFIG (CEEE)	Set foreground and background color sources, monochrome source, draw option bits, host transfer option bits, polygon fill option bit.	Extended draw operations.
LINEDRAW_OPT (A2EE)	Line draw options.	Line draw operations.
SCISSOR_TOP (DEEE)	Extended top scissor.	All draw operations.
SCISSOR_LEFT (DAEE)	Extended left scissor.	All draw operations.
SCISSOR_BOTTOM (E6EE)	Extended bottom scissor.	All draw operations.
SCISSOR_RIGHT (E2EE)	Extended right scissor.	All draw operations.
DEST_CMP_FN (EEEE)	Destination compare function.	All draw operations.
ALU_FG_FN (BAEE)	Foreground ALU function.	All draw operations.
ALU_BG_FN (B6EE)	Background ALU function	Draw operations when monochrome source in non-trivial.
PATT_DATA[0:F] (8EEE)	Color pattern shift registers. Use PATT_DATA_INDEX to index.	Extended draw operations which use pattern as a color source.
PATT_DATA[10:11] (8EEE)	Monochrome pattern shift registers. Use PATT_DATA_INDEX to index.	Extended draw operations which use pattern as a monochrome source
PATT_INDEX (D6EE)	Pattern index.	Extended draw operations which use color or monochrome patterns.
PATT_LENGTH (D2EE)	Pattern length.	Extended draw operations which use color or monochrome patterns.

8514/A Compatible Drawing Operations		
Description	**Draw Initiator**	**Trajectory Control Registers**
Line draw. Draws a line either using Bresenham's algorithm or draw in a direction which is a multiple of 45 degrees. User must supply Bresenham parameters if used.	CMD (9AE8)	CUR_X, CUR_Y, MAJ_AXIS_PCNT, if Bresenham, then ERR_TERM, DESTX_DIASTP, DESTY_AXSTP.
Rectangle fill (FILL_RECT_HOR). Rectangle is filled using horizontal lines. Polygon fill can be performed with this operation when the appropriate option bit is set.	CMD (9AE8)	CUR_X, CUR_Y, MAJ_AXIS_PCNT, MIN_AXIS_PCNT.
Rectangle fill (FILL_RECT_VPIX). Rectangle is filled using vertical lines. Not available in 16-bit per pixel modes.	CMD (9AE8)	CUR_X, CUR_Y, MAJ_AXIS_PCNT, MIN_AXIS_PCNT.
Rectangle fill (FILL_RECT_VNIB). Rectangle is filled using vertical nibbles which alternate direction. Not available in 16-bit per pixel modes.	CMD (9AE8)	CUR_X, CUR_Y, MAJ_AXIS_PCNT, MIN_AXIS_PCNT.
Polygon boundary line. Lines are guaranteed to write only one pixel per scan line. Lines are clamped to the left scissor when drawn to the left of it.	CMD (9AE8)	CUR_X, CUR_Y, MAJ_AXIS_PCNT, if Bresenham, then ERR_TERM, DESTX_DIASTP, DESTY_AXSTP.
Bitblt. Screen to screen copy of a rectangular area. Source and destination areas track each other.	CMD (9AE8)	CUR_X, CUR_Y, MAJ_AXIS_PCNT, MIN_AXIS_PCNT, DESTX_DIASTP, DESTY_AXSTP.
Short stroke vector. Quick line draw for vectors less than 16 pixels long. Generally used in degree mode. Can also operate in Bresenham mode but this has limited usefulness.	SHORT_STROKE (9EE8)	CUR_X, CUR_Y.

ATI-Extended Drawing Operations		
Description	**Draw Initiator**	**Trajectory Control Registers**
Extended line draw. Same as line draw except that extended data paths are permitted. Can also draw polygon boundary lines when the appropriate option bit is set.	BRES_COUNT (96EE)	CUR_X, CUR_Y, LINEDRAW_OPT. If Bresenham, then ERR_TERM, DESTX_DIASTP, DESTY_AXSTP.
Horizontal raster draw. Draws a horizontal line to specified X coordinate. useful for object rasterization algorithms.	SCAN_TO_X (CAEE)	CUR_X, CUR_Y.
Point-to-point line draw. Draws a line given the end points. Can also draw polygon boundary lines when the appropriate option bit is set.	LINEDRAW[3] (FEEE)	LINEDRAW[0-2]. Use LINEDRAW_INDEX to index.
Move to. Move to specified X,Y coordinate.	LINEDRAW[4-5] (FEEE)	Use LINEDRAW_INDEX to index.
Extended short stroke vector. Same as short stroke vector except that extended data paths are permitted.	EXT_SHORT_STROKE (C6EE)	CUR_X, CUR_Y
Non-conforming bitblt. Bitblt with independent source and destination trajectories. Useful for efficient off-screen memory management. Polygon fill can be performed with this operation when the appropriate option bit is set.	DEST_Y_END (AEEE)	CUR_X, CUR_Y, SRC_X, SRC_Y, SRC_X_START, SRC_X_END, SRC_Y_DIR. DEST_X_START, DEST_X_END
Gray scale font rendering assist	-	-

Polygon Fills

8514/A-compatible polygon fill operations are accomplished by first drawing the polygon outline into off-screen memory, rectangle fill the polygon area with the polygon fill bit enabled, and blitting the result to the display area. With extended polygon fills, the outline is drawn into off-screen memory as before, and then a non-conforming blit is performed to the destination area with the polygon fill bit enabled. Drawing large polygons with small memory configurations can be accomplished by banding the polygon into narrow strips.

Polygon fills use the alternate fill algorithm. An internal polygon fill state flag is reset at the beginning of every scan line. When a polygon outline has been detected the fill state flag is toggled. There are 3 flavors of fill operation. Type A and B are 8514/A-compatible only, and fill type C is extended only.

Fill State	Write Mask	Read Mask	Polygon Fill Type		
			A	B	C
Off	0	0	Destination	Destination	Destination
Off	0	1	Destination	Destination	Destination
Off	1	0	Destination	Destination	Destination
Off	1	1	0	Destination	Destination
On	0	0	Destination	Destination	Destination
On	0	1	Destination	Destination	Destination
On	1	0	ALU	ALU	ALU
On	1	1	0	ALU	ALU
Left Outlline Edge			Inclusive	Inclusive	Inclusive
Right Outline Edge			Exclusive	Inclusive	Exclusive
Outline Qualifier			RD_MASK	WRT_MASK	RD_MASK
Outline Region			Destination	Destination	Source
Fill Trajectory			Destination	Destination	Destination
Polygon Enable			PIX_CNTL	PIX_CNTL	DP_CONFIG

Line Clipping

The ***mach32*** has extensions to do fast line pre-clipping within the range [-32768..32767], as follows: The host bursts a set of vectors through the LINEDRAW (FEEE) register. If a clip exception is detected, the host backs up and pre-clips that vector then it continues. In actual applications, clip exceptions are very rare. There are 3 flavors of pre-clip modes, summarized below:

Clip Mode	Trivial Accept	Trivial Reject	Exception
Disabled.	Always.	Never.	Never.
Normal.	Both endpoints within device coordinates and not trivially rejected.	Both endpoints outside of one of the scissor boundaries.	Not trivially rejected and not trivially accepted. One or both of the endpoints must be outside device coordinates.
Polygon boundary lines.	Both endpoints within device coordinates and not trivially rejected.	Endpoints both above or both below or both right of scissor region.	One or both endpoints are above, below, or right of device coordinates, or left endpoint is outside of device coordinates and right endpoint within scissor region.
Patterned lines.	Endpoints within device coordinates.	Never.	One or both endpoints outside device coordinate space.

Polygon boundary clip exception detection is only slightly modified from normal clip exception detection. Left edges are normally clamped to the left scissor so the draw engine cannot be allowed to trivially reject any line to the left of the scissor region.

Patterned lines are never trivially rejected so that the pattern index registers are updated properly.

When a clip exception occurs, all writes to the LINEDRAW (FEEE) register are ignored and the CLIP_OVERRUN register (found in EXT_GE_STATUS (62EE)) is incremented. The host may then back up to the vector which generated the exception.

Off-Screen Memory Management

The ***mach32*** offers superior off-screen memory management with its non-conforming blits and programmable Pitch and Offset registers. Cursors, font caches, icon caches, bitmap save areas, and polygon fill work areas are more effectively implemented by treating off-screen memory as a linear block of memory. The programmable pitch registers allow objects to be drawn directly into linearized cache areas without an intermediate linearization step.

Far-Blit

Far-Blit allows unsymetrical blit operations between any areas of memory and is particularly useful for efficient linearized cache management, rendering images directly into linearized cache and 24-bit drawing operations. For the ATI68800-6, ATI68800LX, and ATI68800AX ***mach32*** accelerators, the graphics engine offset and pitch registers have been split into separate destination and source offset and pitch registers. These new registers are accessed through the GE_OFFSET and PITCH registers, combined with a 2-bit pointer in SHADOW_SET[9:8] 5AEE-W. A pointer value of "1" causes the GE_OFFSET and PITCH values to write to the destination registers; a pointer value of "2" causes the register values to write to the source registers. For compatibility, if the pointer is set to "0" both destination and source registers are written.

Destination Color Compare Mask

The write mask has been split into separate write mask (AAE8) and destination color compare mask (F2EE). This feature allows any plane or group of planes to be used as an alpha channel which is particularly useful in motion video applications.

For compatibility, both registers are loaded when writing to the write mask.

Linear Memory Aperture

The configurable memory aperture is a region in the system address space where screen memory can be accessed directly. If the on-board VGA is enabled, the standard 64KB VGA aperture, pageable on 64KB boundary through VGA page at A0000, will become available and pageable through the entire video address space (up to 4MB). Typical ***mach32*** configurations include a pageable 1MB aperture addressed on 1MB boundary or a linear 4MB aperture. The aperture can access both VGA and accelerator memories.

The aperture may not always function in ISA systems. See *Memory Interface* on page 9-75 for DMA details.

Scalable Gray Scale Fonts

The ***mach32*** has direct hardware support for high speed rendering of scalable anti-aliased fonts. These fonts are anti-aliased in 2 dimensions using a rectangular super-sampling technique, as opposed to single-dimension edge enhancement methods used by other vendors.

Mono Pattern

An 8x8x1 destination-aligned mono pattern has been implemented in the ATI68800-6, ATI68800LX, and ATI68800AX ***mach32*** accelerators. It is enabled by setting D2EE[7]. This new mono pattern uses the four existing mono pattern registers and four additional mono pattern registers. The relationship between the register bits, the mono pattern, and the destination is as follows:

Register Index	DST_Y Coordinate	DST_X Coordinate 0 1 2 3 4 5 6 7
10h	0	7 6 5 4 3 2 1 0
11h	1	7 6 5 4 3 2 1 0
12h	2	7 6 5 4 3 2 1 0
13h	3	7 6 5 4 3 2 1 0
14h	4	7 6 5 4 3 2 1 0
15h	5	7 6 5 4 3 2 1 0
16h	6	7 6 5 4 3 2 1 0
17h	7	7 6 5 4 3 2 1 0

When the 8x8x1 mono pattern is used, the bits of the pattern will be drawn aligned to the destination coordinates. This behaviour differs from that of the 32x1 linear mono pattern, which has no absolute relationship between the pattern and the destination coordinates. The 32x1 linear mono pattern therefore is not considered destination-aligned. See page 9-58 for details.

Block Write and 64-Bit Fill Draw

Block Write

Block write is available only in the ATI68800-6 and ATI68800AX accelerators. Most VRAM devices have a block write mode in which they can perform a pseudo mono data expansion using an internal color register — Block Write Mono Pattern mode. Data bits which are set to "1" enable an internal 4-bit color register to be drawn into a nibble of memory. Data bits which are "0" do not alter the addressed nibble of memory. Since there is only one internal color register, this mode can only be used for foreground color paint operations. A mono pattern may be applied if the background ALU function is transparent. The more common mono pattern which uses the foreground and background color registers may be performed in two passes by first painting the background and then applying a mono pattern with the foreground color.

This feature is only supported with memory type 5 or 6. See CONFIG_STATUS[6:4] (12EE). Block Write mode is enabled by setting MISC_OPTIONS[10] (36EE) to 1. Block Write Mono Pattern mode is enabled by setting PATT_LENGTH[15] (D2EE) to "1".

The foreground color register is written into the VRAM internal color register whenever a block write operation is initiated after the foreground color register has been written. Block write enable is set by the BIOS and should not be changed. A driver must read the state of this bit to determine if this feature is enabled. The block write mode will only be activated under the following conditions:

- The write mask is all "1"s
- The destination compare function is false (Mode = 0)
- The foreground select is set to the foreground color register (Mode = 0)
- The mono select is true (Mode = 0)
- The foreground ALU function is Paint (Function = 7)
- The polygon file mode is disabled

The drawing operation is horizontal degree line, horizontal degree short stroke vector (IBM or ATI compatible), scanline, or fill (IBM H1H4 or ATI compatible).

64-Bit Fill Draw

The feature is supported by the ATI68800LX, ATI68800-6, and ATI68800AX accelerators. A 2MB video memory configuration is required for 64-bit fill draws. Fill drawing operations under certain conditions will fully utilize the 64-bit data bus. The BIOS must set MISC_OPTIONS[11] (36EE-W) to "1" to enable this mode. A driver must read the state of this bit to determine if the feature is enabled. For 2MB DRAM video memory configurations, this mode may be enabled all the time.

Accelerated painting will be enabled when all of the following conditions are true:

- Foreground color select is the foreground color register or the background color register.
- If the mono select is pattern then the background color select must be either foreground color register or background color register.
- The mono select is "always 1" or pattern. If the type of operation is ATI then the 8x8 mono pattern mode must be enabled — for this mode only supports destination-aligned mono patterns.
- The CPUDATA bit must be "false" if the type of operation is IBM.
- The FGALU function must be one of: 0, 1, 4, or 7.
- If the mono select is pattern then the BGALU function must be one of: 0, 1, 4, or 7.
- All color compare functions must be set to zero.
- Polygon fill must be disabled.
- The write mask must be all "1"s.
- The pixel size must be 8- or 16-bit.
- The block write function must be disabled.

Split Transfer Cycle

Split transfer cycle is supported by the ATI68800-6 and ATI68800AX accelerators. It has been added to allow decoupling of the memory clock and serial clock. This allows higher memory clocks in VRAM implementations with a corresponding increase in performance. This feature is enabled if the memory type is set to 5 or 6. See CONFIG_STATUS[6:4] (12EE).

Separate Display/Drawing Pixel Sizes

This feature is available in the ATI68800LX. ATI68800-6, and ATI68800AX accelerators only. Both the display and drawing pixel sizes can be written independently. The display pixel size is written when EXT_GE_CONFIG[11] is set to "1". The drawing pixel size is written when EXT_GE_CONFIG[15] is set to "1".

For compatibility, both pixel sizes are written if bits 11 and 15 are both set to "0".

Memory Mapped Registers

Memory mapping of registers is supported by the ATI68800LX, ATI68800-6, and ATI68800AX ***mach32*** accelerators. This feature is enabled when bit 32EE[5] is "1". Since the feature is enabled by the BIOS, 32EE[5] should not be changed. A driver must read the state of this bit to determine if it can use the feature.

All of the coprocessor registers may be mapped into the upper 128 dwords of a 4MB linear aperture. The offset from the start of the aperture to the base of the register space is FFE00 bytes. See diagram of memory mapped registers in Appendix A. Each 16-bit register will map into the lower 16 bits of a 32-bit dword. The upper 16 bits are currently unused and therefore should be ignored.

The 64 IBM-compatible registers from 2E8h to FFE8h are mapped into the 64 dwords from FFE00 to FFEFC. The 64 ATI-extended registers from 2EEh to FFEEh are mapped into the 64 dwords from FFF00 to FFFFF.

To the extent that I/O registers are sparse for either reads or writes the memory mapped registers will also be sparse. Note that the 128 dwords reserved for registers are not available for memory operations. Examples of both IBM and ATI register mappings are described for your reference in Appendix A.

FIFO discipline

When sending data to the 16-entry ***mach8*** or ***mach32*** FIFO, you should poll the EXT_FIFO_STATUS register to ensure there is available FIFO space, especially when FIFOed registers are to be written. FIFO operation is not always as simple as it sounds. It is quite possible to have a situation requiring more than 16 FIFO entries. Such operations must be divided into multiple sequences of polling for FIFO space, followed by writing to the FIFO. Generally, experimentation with various configurations rather than simply waiting for 16 spaces and then filling the FIFO repeatedly can prove beneficial.

It appears that a subtle placement of status reads can save a read from the card and result in more optimal concurrency between the host processor and graphics coprocessor. However, it has been our experience that this kind of FIFO tuning is rarely worthwhile due to the fact that timings can vary greatly — depending on the speed of the host and coprocessor, number of wait-states on the bus, speed of the bus, speed of the coprocessor memory, etc.

In certain situations, performance can improve by preserving the value from the FIFO status register for interpretation later. For example, an operation requires three FIFO entries and polling of the EXT_FIFO_STATUS register indicates 14 available FIFO spaces. It follows that after the operation there would be at least 11 spaces available. This method allows the next write to the FIFO to proceed as if the EXT_FIFO_STATUS register had just been polled and 11 FIFO spaces were discovered. This is particularly useful for operations which are

divided into several logical components, or for many small operations that are performed separately.

A practical example is the repeated line drawing for scan conversion of irregular solids which requires three I/O instructions (i.e., only three FIFO spaces) per scanline. Keeping track of the FIFO status is therefore more efficient than waiting/polling for each set of FIFO writes.

Host FIFO Discipline

Host data transfers involve read and write operations. Special considerations for each type are as follows:

Reading from the PIX_TRANS register — After the operation is triggered, you must poll for the GE_DATA_READY bit in the EXT_GE_STATUS register in order to read data at full host speed. A REP prefix may be used. Memory mapped I/O also works if your card supports it.

Although it is possible to use standard ATI FIFO discipline when performing I/O to the host data ports (PIX_TRANS, SHORT_STROKE and EXT_SHORT_STROKE), these registers have support for wait-stated I/O operations which are enhancements but can be somewhat trickier. Wait-stating is a process whereby the time required for an I/O operation is extended by an I/O device.

Wait-stating is normally used by devices to ensure safe inter-device communications; but the ***mach8*** and ***mach32*** use it to avoid the need for FIFO discipline, for some of their operations. Wait-stating incurs a fixed time delay for all I/Os. When the FIFO is more than half full, the fixed delay may not be sufficient. In the event that a large operation is in the FIFO prior to this host data transfer, the FIFO should be flushed and the graphics engine made idle before wait stating can be trusted for adequate delays. A simple way of doing this is to use proper FIFO discipline for operation setup, then wait for enough spaces in the FIFO to ensure that no other operation is pending.

For example, if an operation requires six FIFO entries, you can be assured that it is being processed only when fewer than six entries remain in the FIFO. This will not be the case if the previous operation is in the engine but not complete.

A single additional FIFO space is also required to place the first wait-stated register in the FIFO and cause the waitstate to occur. Thus, if your operation requires six I/Os before the host data, you should wait for six spaces in the FIFO, set up the operation, then wait for nine entries in the FIFO (16-(6+1)) before performing host I/O with no additional FIFO discipline, regardless of the amount of data being transferred.

This page intentionally left blank.

8

8514/A-Compatible Coprocessor Registers

DAC Operations

Most DAC types today are compatible with the industry-standard 256 color triple-6 DAC. In a configuration where the VGA within the ***mach32*** is disabled, and there is a seperate external VGA, then the two DACs may be programmed independently. In the special case where a passthrough cable is connected and PASS_THRU for VGA is set, then any programming made to the VGA DAC would also program the accelerator DAC.

DAC registers are writable either at the 8514/A-compatible address of 02EAh-02EDh or VGA-compatible address of 03C6h-03C9h. But they are readable only from the 8514/A compatible address. If Advance Function Control sub-register PASS_THRU is set to VGA, then access to the VGA DAC register would also affect the 8514/A DAC.

Currently available DACs do not use a "DAC Ready" indicator. Some DAC types may even require at least 240ns between successive writes to DAC registers. In practice, this means string I/O transfers, e.g., REP OUTSB, should be avoided when reading or writing DAC registers. Once a read/write DAC index is written, multiple read/write operations will not require further setting of the index for subsequent entries. The following sequence of instructions may be used to ensure acceptable timing on fast CPUs such as 33MHz 80486s.

```
LoopTop:
    LODSB
    OUT DX,AL
            * Place any required delay instructions here for
              faster CPU or slower RAMDAC
    LOOP LoopTop
```

See the register specifications on the target RAMDAC for exact timing requirements. Readers may also refer to *The Programmer's Guide to the EGA and VGA Cards* by R.F. Ferraro for additional general information.

To write a range of palette entries to the DAC, first write the starting color index to DAC_W_INDEX, then for each color, write three bytes (R,G,B values) to DAC_DATA. (The color index auto increments after every third write.)

To read a range of palette entries from the DAC, write the starting color index to DAC_R_INDEX and then, for each color, read three bytes (R,G,B values) from DAC_DATA (the color index auto increments after every third read).

See Appendix F for details on DAC programming for extended modes — 16bpp, 24bpp, and 1280x1Kx8 etc.

DAC Registers

DAC Data Register				
02ED (R/W)	DAC_DATA			
	-	8514/A	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
7:0	DAC_DATA	DAC data		

DAC Mask Register				
02EA (R/W)	DAC_MASK			
	-	8514/A	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
7:0	DAC_MASK	Participating bit positions in the mask for DAC lookup are set to one.		

DAC Read Current Color Index Register				
02EB (R/W)	DAC_R_INDEX			
	-	8514/A	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
7:0	DAC_R_INDEX	The current read index for a DAC operation — increments after every third read of DAC_DATA (02ED). Also see DAC_W_INDEX (02EC).		

DAC Write Current Color Index Register				
02EC (R/W)	DAC_W_INDEX			
	-	8514/A	Mach 8	Mach 32

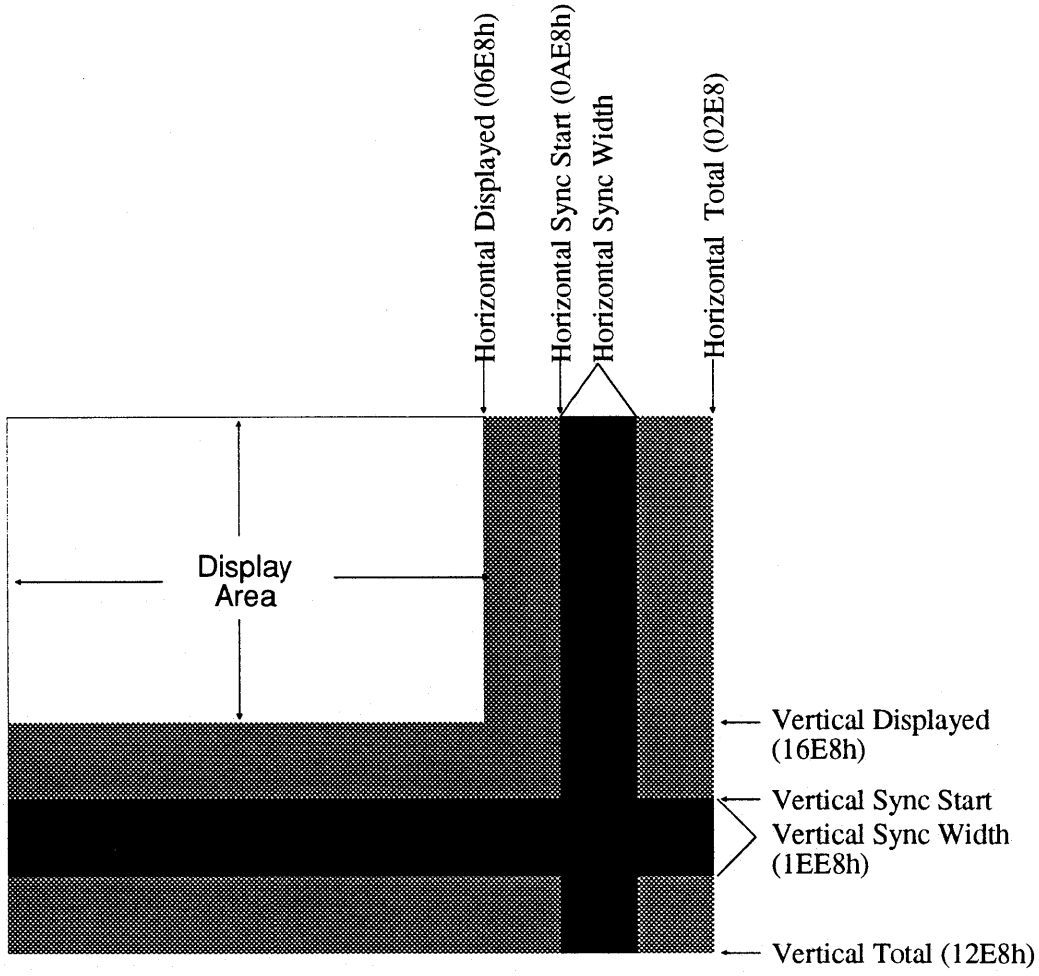
Bit	Mnemonic	Description
7:0	DAC_W_INDEX	The current write index for a DAC operation. Also see DAC_R_INDEX (02EB).

CRT Operations

The CRT registers must be set according to memory configuration and desired resolution. The controller and the graphics engine must be configured separately. A sub-register of Multi-Function Control, MEM_CNTL, must be set to correspond to the video control parameters and memory configuration for proper drawing to proceed correctly. See CRT parameters in Appendix D for details.

WARNING:
To avoid possible monitor damage, the CRT Controller should be put into Reset state while CRT parameters are being changed. The correct sequence to do it is as follows:

1. Set DISP_CNTL[5:6] to 2 (RESET).
2. Set CRT parameters.
3. Set DISP_CNTL[5:6] to 1 (ENABLE)



CRT Configuration and Status

A standard 8514/A would select a 25MHz or 45MHz pixel clock from the ADFUNC_CNTL (4AE8) register. To maintain compatibility without sacrificing available display modes, the ***mach32*** uses the same control bit, but selects a CRT shadow set instead. Default shadow sets are loaded with 640x480 and 1024x768 modes, although they may contain any valid mode.

All CRT timing parameters (H_DISP, H_TOTAL, H_SYNC_STRT, H_SYNC_WID, V_TOTAL, V_DISP, and V_SYNC_WID) are inclusive. For example, a value of 5 for H_TOTAL implies an actual horizontal width of 6 characters (i.e., 48 pixels).

VGA Controller Configurations

The ***mach32*** accelerator can be configured with the internal VGA controller either enabled (with a single monitor attached), or disabled (and use a companion VGA with a single monitor and a passthrough cable, or dual monitor.)

Internal VGA Controller Enabled

The internal VGA enabled configuration requires only a single DAC. Writing to the VGA DAC register will also write the (logical) accelerator DAC and vice versa. Dual adapter systems have independent DACs. Generally, when writing one DAC, e.g., the VGA DAC, the other accelerator DAC will not be written. However, there is one exception — if (and only if) VGA passthrough is enabled and the passthrough cable is attached — programming the VGA DAC will also write the accelerator DAC.

The VGA features connector is bidirectional. When VGA passthrough is enabled, the VGA will supply pixel data to both DACs. Even if VGA passthrough is disabled, the ***mach32*** will drive both DACs. Therefore, dual monitor systems should not have the passthrough cable connected to display both VGA and accelerator simultaneously. Note that the features connector is only 8 bits wide and will not pass through HiColor or TrueColor data.

In a dual adapter system with no passthrough cable, and the accelerator is in VGA passthrough mode, normally the pixel clock is not present. To program the accelerator DAC pixel clocks must be present. In this situation, a PASSTHROUGH_OVERRIDE bit is provided to drive the pixel clock line of the features connector to allow the DAC to be programmed. This bit should not be set with the passthrough cable attached, and should be set only by the boot ROM on power up. A VGA enabled adapter uses a 32K BIOS which resides at segment C000h.

Internal VGA Controller Disabled

A VGA disabled adapter uses a 4K BIOS and the location is determined by the on-board straps. A VGA disabled adapter cannot use the VGA registers to program the DAC nor the EEPROM. Nor does it have a 64K memory aperture.

CRT Registers

Advanced Function Control Register				
4AE8 (W)	ADVFUNC_CNTL			
	-	8514/A	Mach 8	Mach 32

Bit	Mnemonic	Description
0	PASS_THROUGH	**0** = Enabled — Video output is VGA, driven by data from the features connector. **1** = Disabled — Video output is 8514. This bit is over-written by CLOCK_SEL[0] (4AEE).
1	-	Reserved
2	RESOLUTION	**0** = Shadow register set 1, 640x480 **1** = Shadow register set 2, 1024x768
15:3	-	Reserved

Register Description:

1. The Advanced Function Control register is provided for compatibility with the IBM 8514/A. Fields indicate 8514 functions.

2. Writing to the Advanced Function Control register forces the CRT controller to 8514-compatible mode. (Writing to CLOCK_SEL (4AEE) forces the CRT controller to ATI mode.)

3. Only ***mach8*** will reset CRT_PITCH, GE_PITCH, CRT_OFFSET, and GE_OFFSET registers when ADFUNC_CNTL is written, not ***mach32***.

Display Control Register				
22E8 (W)	DISP_CNTL			
	-	8514/A	Mach 8	Mach 32

Bit	Mnemonic	Description
0	-	Reserved
2:1	Y_CONTROL	Line skip control: **0** = Bits 1:2 skipped **1** = Bit 2 skipped **2** = Reserved **3** = Reserved See V_TOTAL (12E8), V_DISP (16E8), and V_SYNC_STRT (1AE8).
3	DOUBLE_SCAN	**0** = Single scan **1** = Double scan
4	INTERLACE	**0** = No interlace **1** = Interlace
6:5	ENA_DISPLAY	**0** = No change **1** = 8514 CRT Controller enabled **2** = 8514 CRT Controller reset **3** = 8514 CRT Controller reset
7	-	Reserved

Display Status Register				
02E8 (R)	DISP_STATUS			
	-	8514/A	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
0	RGB_TEST	**0** = R,G,B less than or equal to 0.3V **1** = R,G,B greater than 0.3V		
1	VERT_SYNC	In 640x480 mode, typically 0 during vertical sync In 1024x768 mode, typically 1 during vertical sync		
2	LINE_SYNC	This bit toggles on every horizontal retrace. **0** = odd scan line **1** = even scan line		
7:3	-	Reserved		
Note: Sync polarity is set in V_SYNC_WID[5] (1EE8).				

Horizontal Displayed Register				
06E8 (W)	H_DISP			
	-	8514/A	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
7:0	H_DISP	Displayed portion of the screen — measured horizontally in units of eight pixels. Also see V_DISP (16E8).		
15:8	H_TOTAL (Alt)	Alternate total horizontal area of the screen — see H_TOTAL (2E8-W).		

Horizontal Sync Start Register				
0AE8 (W)	H_SYNC_STRT			
	-	8514/A	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
7:0	H_SYNC_STRT	Screen position of the H sync specified in units of eight pixels — measured horizontally from the start of the display area.		

Horizontal Sync Width Register				
0EE8 (W)	H_SYNC_WID			
	-	8514/A	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
4:0	H_WIDTH	Width of the sync width measured horizontally in units of eight pixels		
5	H_POLARITY	**0** = Positive H sync polarity **1** = Negative H sync polarity		
7:6	-	Reserved		

Horizontal Total Register				
02E8 (W)	H_TOTAL			
	-	8514/A	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
7:0	H_TOTAL	Total horizontal area of the screen in units of eight pixels — pixel count measured from the beginning of one row to the beginning of the next row. Also see V_TOTAL (12E8) and alternate H_TOAL at H_DISP[15:8] (6E8-W).		

Vertical Displayed Register				
16E8 (W)	V_DISP			
	-	8514/A	Mach 8	Mach 32

Bit	Mnemonic	Description
11:0	V_DISP	Displayed portion of the screen — measured vertically in lines. Also see H_DISP (06E8).
15:12	-	Reserved

Register Description:

1. Registers that specify vertical amounts are specified with respect to the Y Counter of the CRT controller. This Y counter is not necessarily linear: it may skip bits according to the setting of DISP_CNTL[Y_CONTROL] (22E8).

2. The different skip representations for the Vertical Displayed register are as follows:

   SKIP_1_2 Representation (only used in min mode):
   Bits 1 and 2 are skipped if Y_CONTROL is set to zero,
   i.e., if DISP_CNTL[1,2] at 22E8h is 00.

   SKIP_2 Representation (used in normal mode):
   ((linear<<1) & 0xFFF8) | (linear & 0x3) | ((linear & 0x80)>>5).

   Linear Representation (not used):
   ((SKIP_2>>1) & 0xFFFC) | (SKIP_2 & 0x3).

Vertical Sync Start Register					
1AE8 (W)		V_SYNC_STRT			
		-	8514/A	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**			
11:0	V_SYNC_STRT	Screen position of the V sync specified in lines — measured from the top edge of the display.			
15:12	-	Reserved			

Register Description:

1. Registers that specify vertical amounts are specified with respect to the Y Counter of the CRT controller. This Y counter is not necessarily linear: it may skip bits according to the setting of DISP_CNTL[Y_CONTROL] (22E8).

2. The different skip representations for the Vertical Sync Start register are as follows:

   SKIP_1_2 Representation (only used in min mode):
   Bits 1 and 2 are skipped if Y_CONTROL is set to zero,
   i.e., if DISP_CNTL[1,2] at 22E8h is 00.

   SKIP_2 Representation (used in normal mode):
   ((linear<<1) & 0xFFF8) | (linear & 0x3) | ((linear & 0x80)>>5).

   Linear Representation (not used)
   ((SKIP_2>>1) & 0xFFFC) | (SKIP_2 & 0x3).

Vertical Sync Width Register				
1EE8 (W)	V_SYNC_WID			
	-	8514/A	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
4:0	V_WIDTH	Width of the sync pulse measured vertically in lines		
5	V_POLARITY	**0** = Positive V sync polarity **1** = Negative V sync polarity		
15:6	-	Reserved		

Vertical Total Register				
12E8 (W)	V_TOTAL			
	-	8514/A	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
11:0	V_TOTAL	Size of the screen specified in lines — line count measured from the first line of one frame to the first line of the next frame. Also see H_TOTAL (02E8).		
15:12	-	Reserved		

Register Description:

1. Registers that specify vertical amounts are specified with respect to the Y Counter of the CRT controller. This Y counter is not necessarily linear: it may skip bits according to the setting of DISP_CNTL[Y_CONTROL] (22E8).

2. The different skip representations for the Vertical Total register are as follows:

   SKIP_1_2 Representation (only used in min mode
   Bits 1 and 2 are skipped if Y_CONTROL is set to zero,
   i.e., if DISP_CNTL[1,2] at 22E8h is 00.

   SKIP_2 Representation (used in normal mode)
   ((linear<<1) & 0xFFF8) | (linear & 0x3) | ((linear & 0x80)>>5).

   Linear Representation (not used):
   ((SKIP_2>>1) & 0xFFFC) | (SKIP_2 & 0x3).

Engine Control

Engine Control Registers

The 8514/A-compatible engine control registers are divided into five classes:

- Draw Engine Setup
- Data Path Control
- ALU Control
- Draw Control
- Special

Draw Engine Setup Registers

The following register is normally not used.

- MEM_CNTL (BEE8*5)

Data Path Control Registers

These registers control the color and monochrome sources and multiplexers.

- FRGD_MIX (BAE8)
- FRGD_COLOR (A6E8)
- BKGD_MIX (B6E8)
- BKGD_COLOR (A2E8)
- PATTERN_L (BEE8*8)
- PATTERN_H (BEE8*9)
- PIX_CNTL (BEE8*A)

ALU Control Registers

These registers control ALU and comparator function.

- FRGD_MIX (BAE8)
- BKGD_MIX (B6E8)
- PIX_CNTL (BEE8*A)
- CMP_COLOR (B2E8).

Inclusion Control Registers

These control various aspects of the draw engine.

- WRT_MASK (AAE8)
- RD_MASK (AEE8)
- SCISSOR_T (BEE8*1)
- SCISSOR_L (BEE8*2)
- SCISSOR_B (BEE8*3)
- SCISSOR_R (BEE8*4)

Both the Write Mask and Read Mask registers are used to control which bit planes are to be involved in the operation. Scissor registers specify the pixels for the operation.

Special Register

These are only used for special functions such as polygon filling.

- PIX_CNTL (BEE8, Index A).

Interrupts

The 8514/A can generate host interrupts (if enabled). Each interrupt has corresponding ENA and ACK fields in SUBSYS_CNTL, and an INT field in SUBSYS_STAT. Writing a logical "1" to an ACK field clears the corresponding INT field for a minimum of one clock cycle. The INT fields operate even if the corresponding ENA field is zero. The 8514/A interrupt request line is generated by OR'ing the result of each INT field ANDed with its corresponding ENA field.

FIFO Status Register

Graphics processor registers are writeable registers that have a "high" bit A15 in their address. All host writes to these registers are queued in a 16-word command FIFO.

The Command FIFO allows the 8514/A to accept data from the host while the graphics processor is busy. This allows concurrent operation by the host and the processor.

When I/O address bit A14 is set, it has the effect of adding wait states to any read or write operation that otherwise would cause SUBSYS_STAT[2] to be set if R/W were executed immediately.

When executing draw commands with CMD[8] (CPU_WAIT) = 1, the graphics processor consumes entries from the Command FIFO expecting host data. However, only those entries with addresses A2E8(BKGD_COLOR), A6E8(FRGD_COLOR), E2E8(PIX_TRANS), and E6E8 are treated as host data. Any entry not in this group causes CMD[8] (CPU_WAIT) to be ignored, and the draw operation is continued as if CMP[8] is logical 0, without further consumption of FIFO data.

When the drawing engine is not executing a draw command, it interprets E2E8 (PIX_TRANS) as background color BKGD_COLOR (A2E8), and E6E8 as foreground color FRGD_COLOR (A6E8).

Note carefully that the interpretation of PIX_TRANS and BKGD_COLOR is made when the graphics processor consumes FIFO data, and *not* when the host writes data to the FIFO.

PIX_TRANS is just BKGD_COLOR with Address Bit A14 turned on and can be regarded as a convenient fiction that simplifies documentation and programming — Register Alias.

PIX_TRANS(R) is not associated with the Command FIFO. However, both are disabled when SUBSYS_STAT[2] is logical one.

The command FIFO overruns when an I/O write of an 8514/A-compatible register in the address range [82E8-FEE8] or an extended register occurs and all 16 FIFO entries are full. The FIFO can also be locked if GE_STAT[8] = 0 and a read from PIX_TRANS(E2E8) is attempted. All applicable I/O writes should be preceded by checking the FIFO Status register. 8514/A-compatible applications use GE_STAT(9AE8) and ***mach8*** or ***mach32*** aware applications use EXT_FIFO_STATUS(9AEE).

Once locked, the FIFO may be cleared by resetting the draw engine using SUBSYS_CNTL (42E8) or by clearing the INVALID_IO bit of the same register.

Graphics Engine Registers (all queued in the Command FIFO)		
Address	**Mnemonic**	**Description**
82E8	CUR_Y	Current Y Position
86E8	CUR_X	Current X Position
8AE8	DESTY_AXSTP	Destination Y Position/Axial Step Constant
8EE8	DESTX_DIASTP	Destination X Position/Diagonal Step Constant
92E8	ERR_TERM	Error Term
96E8	MAJ_AXIS_PCNT	Major Axis Pixel Count
9AE8	CMD	Command
9EE8	SHORT_STROKE	Short Stroke Vector Transfer
A2E8	BKGD_COLOR	Background Color
A6E8	FRGD_COLOR	Foreground Color
AAE8	WRT_MASK	Write Mask
AEE8	RD_MASK	Read Mask
B2E8	COLOR_CMP	Color Compare
B6E8	BKGD_MIX	Background Mix
BAE8	FRGD_MIX	Foreground Mix
BEE8	MULTIFUNC_CNTL	Multifunction Control
E2E8	PIX_TRANS	Pixel Data Transfer

Engine Control Registers

Memory Control Register				
BEE8*5 (W)	MEM_CNTL			
	-	8514/A	Mach 8	Mach 32

Bit	Mnemonic	Description
1:0	GE_X_CONTROL	Reserved (set to (1,0))
3:2	GE_Y_CONTROL	GE Memory Control (should be equal to DISP_CNTL.Y_CONTROL): **00** = SKIP1 (640x480x8, 512K configuration) (Not Used - ATI68800-6) **01** = Linear (other resolutions and configurations) **10** = Reserved **11** = Reserved
4	PLANE_SELECT	Plane Select (512K configuration only) **0** = select lower 4 planes (Bank 0) **1** = select upper 4 planes (Bank 1) **ATI68800-6:** Not used - min mode no longer supported
11:5	-	Reserved
15:12	MF_INDEX	Multi-Function Index (set to 5)

* Index 5 of port address BEE8.

Register Description:

1. **GE_Y_CONTROL**

   Specifies skip behaviour for the draw controller and should be set to the same value as DISP_CNTL, Y_CONTROL.

   When GE_Y_CONTROL is 00b the CPU must adjust the values it sends to CUR_Y, DESTY_AXSTP, MULTIFUNC_CNTL [SCISSOR_T], AND MULTIFUNC_CNTL[SCISSOR_B]. For example, if (X,Y) is the desired start position for a draw command, the host must write ((2*Y)+(Y&1)) to CUR_Y.

2. **PLANE_SELECT**

   Works for 512K configuration only, and applies to both read and write operations.

Subsystem Control Register				
42E8 (W)	SUBSYS_CNTL			
	-	8514/A	Mach 8	Mach 32

Bit	Mnemonic	Description
0	VBLANK_ACK	**0** = No change **1** = Associated interrupt status bit cleared
1	INSIDE_SCISSOR_ACK	**0** = No change **1** = Associated interrupt status bit cleared
2	INVALID_IO_ACK	**0** = No change **1** = Associated interrupt status bit cleared
3	GE_IDLE_ACK	**0** = No change **1** = Associated interrupt status bit cleared
7:4	-	Reserved
8	VBLANK_ENA	**0** = Interrupt disabled - VBLANK **1** = Interrupt enabled - VBLANK
9	INSIDE_SCISSOR_ENA	**0** = Interrupt disabled - Pixel inside region **1** = Interrupt enabled - Pixel inside region
10	INVALID_IO_ENA	**0** = Interrupt disabled - Invalid I/O **1** = Interrupt enabled - Invalid I/O
11	GE_IDLE_ENA	**0** = Interrupt disabled - Engine idle **1** = Interrupt enabled - Engine idle
13:12	-	Reserved
15:14	GE_RESET	**0** = No change **1** = Normal Engine operation **2** = Engine & FIFO reset

Register Description:

1. The Subsystem Control register is used to clear any selected interrupt sources — If any control bit is enabled in this register, the corresponding interrupt status bits in 42E8-R will be set even if the associated interrupt enable is not set to monitor the condition.

2. The related ACK, ENA, and INT fields bear the following relationship to one another — Writing logical one to an ACK will clear its corresponding INT for at least one period. The INT continues to operate even if the corresponding ENA is logical zero because ***mach32*** interrupt request is generated only if (INT • ENA) is true.

3. Interrupts (INT) may be generated from any of the following sources:

   - VBLANK — generated by the trailing edge of the vertical blanking signal after the last vertical line in a frame is displayed. The line is specified at V_DISP (16E8).

- INSIDE_SCISSOR — generated if the current destination pixel falls inside the scissor region during any draw operation. Destination is specified by DESTY_AXSTP (8AE8) and DESTX_DIASTP (8EE8) for IBM blits, and by CUR_X (86E8) and CUR_Y (82E8) for all other draw operations. Monochrome reads ignore the scissors, but they still trigger the interrupt correctly. Blits trigger the interrupt only for pixels on the destination trajectory.
- INVALID_IO — generated if an attempt to write to the Command FIFO or to read from the CPU Data Transfer register fails even after waiting out the maximum permissable wait-states if A14 is logical one. The illegal I/O may be due to a full FIFO or register not ready.
- GE_IDLE — generated if the FIFO is empty *and* the engine is in transition to idle.

4. See SUBSYS_STATUS (42E8-R) for INT descriptions.

Subsystem Status Register				
42E8 (R)	SUBSYS_STATUS			
	-	8514/A*	Mach 8	Mach 32

Bit	Mnemonic	Description
0	V_BLANK_INT	**0** = Not detected **1** = Detected - generate interrupt if enabled
1	INSIDE_SCISSOR_INT	**0** = Not detected **1** = Detected - generate interrupt if enabled
2	INVALID_IO_INT	**0** = Not detected **1** = Detected - generate interrupt if enabled
3	GE_IDLE_INT	**0** = Not detected **1** = Detected - generate interrupt if enabled
6:4	MONITOR_ID	**2** = IBM 8514 monitor **5** = IBM 8503 monitor **6** = IBM 8513 or 8512 monitor **7** = Monitor not connected Bits correspond to pins 4, 12, and 11 of the analog monitor connector
7	MEM_SIZE	**0** = 512KBytes **1** = 1MBytes
15:8	-	Reserved

*The Subsystem Status register in 8514-compatible mode is only 8-bit wide (7:0).

Register Description:

1. The Subsystem Status register contains various conditions for interrupt generation. See SUBSYS_CNTL (42E8(W)) for ACK and ENA descriptions.

2. For ***mach32***, MEM_SIZE is set to 512K on 512KB boards or 1MB boards with the memory boundary set to 512K; in all other cases MEM_SIZE is set to 1M.

Drawing Control

Draw Control Registers

Registers that control the position, size, and direction of the trajectory are draw control registers. There are two general drawing control registers: CMD (9AE8-W) and GE_STAT (9AE8-R).

- CMD (9AE8)
- CUR_X (86E8)
- CUR_Y (82E8)
- GE_STAT (9AE8)
- ERR_TERM (92E8)
- MAJ_AXIS_PCNT (96E8)
- MIN_AXIS_PCNT (BEE*0)
- SRC_X/DEST_X/DIASTP (8EE8)
- SRC_Y/DEST_Y/AXSTP (8AE8)

The draw registers are used for trajectory control and initiating draw operations. All 8514/A compatible draw operations are initiated with the CMD (9AE8) with the exception of short stroke vectors, which are drawn with SHORT_STROKE (9EE8).

The CMD register defines and initiates draw commands. All draw parameters should be set before this register is used. Writing to the Command register is one of the few actions that actually causes an effect on the display screen. Most other registers are used to "set the stage" for draw commands.

The Command register defines the type of drawing operation used. There are three categories of draw operations - line draws, rectangle fills, and BLITs. (The term BLIT is generally used to describe data transfer, and in particular here refers to the copying of pixel (or monochrome) data from one rectangular screen region to another.) These draw operations define *trajectories*, which are the paths along which pixels are drawn in order to create lines or rectangles.

Several parameters influence trajectory:

- CUR_X and CUR_Y control position
- MAJ_AXIS_PCNT and MULTIFUNC_CNTL [MIN_AXIS_PCNT] control size
- DESTY_AXSTP and DESTX_DIASTP control slope (for lines)
- The fields CMD[3] and CMD[7:5] determine quadrant/octant

- CMD[2] controls whether the last pixel in a line (or the last row/column in some rectangle types) is drawn.

The pixel values that are drawn into the trajectory can come from one of several possible sources, including the FRGD_COLOR register, the host CPU, or BLIT source. When using host CPU data, setting CMD[8] to 1 causes the graphics engine to wait for host I/O.

Scissor Registers

The 8514/A Scissor registers inhibit drawing outside a rectangular area of the screen. Scissor registers do not provide full clipping of drawing operations. This means any coordinate parameters falling outside the range of the graphics engine's coordinate system, which extends from (-512,-512) to (1535, 1535), will be interpreted incorrectly.

Even when the scissor registers are being used, the host application may still have to perform pre-clipping of graphics primitives before passing them to the graphics engine. Despite of this, the scissor registers do have many useful applications when implementing graphics functions.

Scissor registers may also be used to generate interrupts on drawing operations that fall inside the scissor rectangle, the INSIDE_SCISSOR_ENA bit of the SUBSYS_CNTL register enables this feature. INSIDE_SCISSOR interrupts are useful for implementing high-speed pick operations.

The Scissor registers behave as follows. The Left and Top scissors are 11 bit registers interpreted as signed numbers in the range (-1024, 1023). The Right and Bottom scissors are interpreted as unsigned numbers in the range (0, 2047). The X and Y coordinates are interpreted as numbers in the range (-512, 1535).

A point (X,Y) is compared against the Scissor registers and if it is inside the scissors, it is plotted modulo 1024. Memory pitch is always 1024 for 8514/A compatible mode. Specifically:

```
    if ( X >= SCISSOR_L &&
         X <= SCISSOR_R &&
         Y >= SCISSOR_T &&
         Y <= SCISSOR_B ){

         if (write) setpel ( X mod 1024, Y ) ;
         if (read) set_data_ready (getpel ( X mod 1024, Y ));
    }    else {
         if (read) set_data_ready (0xFF) ;
}
```

As a consequence of the above, if scissors are set at (L,R,T,B) = (-512, 1535, -512, 1535) then all possible points, even those off-screen, are drawn and visible (albeit with wrap-around).

Color host read operations return 0xFF outside the scissor region.

The blit source is not affected by the scissors. As would be expected from the above pseudo code, The blit source wraps at the memory pitch in the X direction.

The scissor region has no effect on monochrome reads.

The Scissor registers include SCISSOR_B (BEE8, Index 3), SCISSOR_L (BEE8, Index 2), SCISSOR_T (BEE8, Index 1).

Data Transfer Registers

The Data Transfer registers are PIX_TRANS (E2E8 (R/W)) and SHORT_STROKE (9EE8-W).

Pixel Transfer ALU Control Registers

The Pixel Transfer ALU (Arithmetic Logic Unit) is used to combine each pixel on a drawing trajectory with one or more sources of data.

For each draw operation, one pixel from each of the color and monochrome data sources will be combined with the destination pixel by the pixel transfer ALU.

The pixel transfer ALU provides 16 Boolean functions and 16 arithmetic functions. The ALU will use one of two independent mixes (functions) depending on the value of the monochrome data streams. If the value of the monochrome data stream is one for the current pixel, the foreground mix will be used; if the value is zero, the background mix will be used.

COLOR_SRC	Color Source Select
00	BKGD_COLOR register
01	FRGD_COLOR register
10	PIX_TRANS register
11	BLIT buffer

A destination color comparator allows destination pixels of pre-selected colors to be protected from update by drawing operations.

Mix Control

Registers FRGD_MIX and BKGD_MIX are used to specify foreground and background color sources and mix functions that are used to combine the source and destination. The following tables list the definitions of codes being used.

MIX_FN	Mix Code Description (S = Source, D = Destination)
00	not D
01	Zero
02	One
03	D
04	not S
05	D xor S
06	(Not D) xor S
07	S
08	(not D) or (not S)
09	D or (not S)
0A	(not D) or S
0B	D or S
0C	D and S
0D	(not D) and S
0E	D and (not S)
0F	(not D) and (not S)
10	Min (S, D)
11	D-S
12	S-D
13	D+S
14	Max (S, D)
15	(D-S)/2

MIX_FN	Mix Code Description (S = Source, D = Destination)
16	(S-D)/2
17	(D+S)/2
18	Max (0, D-S)
19	Max (0, D-S)
1A	Max (0, S-D)
1B	Min (0xFF, D+S)
1C	Max (0, D-S)/2
1D	Max (0, D-S)/2
1E	Max (0, S-D)/2
1F	(0xFF<S+D) ? 0xFF:(S+D)/2*

*If function < S+D, then it evaluates to 0xFF; otherwise, it evaluates to (S+D)/2.

Pixel Transfer ALU Control registers include:

- BKGD_COLOR (A2E8-W)
- BKGD_MIX (B6E8-W)
- CMP_COLOR (B2E8-W)
- FRGD_COLOR (A6E8-W)
- FRGD_MIX (BAE8-W)
- PATTERN_H (BEE8*9-W)
- PATTERN_L (BEE8*8-W)
- PIXEL_CNTL (BEE8*A-W)
- RD_MASK (AEE8-W)
- WRT_MASK (AAE8-W)

The Multi-Function Control Register

A multi-function control register (BEE8h) is used to access nine (9) sub-registers by index, which is the four high bits of the register value. The sub-registers are described later in this chapter as follows:

- BEE8*0 — MIN_AXIS_PCNT
- BEE8*1 — SCISSOR_T
- BEE8*2 — SCISSOR_L
- BEE8*3 — SCISSOR_B
- BEE8*4 — SCISSOR_R
- BEE8*5 — MEM_CNTL
- BEE8*8 — PATTERN_L
- BEE8*9 — PATTERN_H
- BEE8*A — PIXEL_CNTL

Drawing Control Registers

Background Color Register				
A2E8 (W)	BKGD_COLOR			
	-	8514/A	Mach 8	Mach 32

Bit	Mnemonic	Description
15:0	BKGD_COLOR	Determines the background color. Upper eight bits (15:8) ignored in 8-bit mode.

Background Mix Register				
B6E8 (W)	BKGD_MIX			
	-	8514/A	Mach 8	Mach 32

Bit	Mnemonic	Description
4:0	MIX_FN	Background Mix Value
6:5	COLOR_SRC	Background Source Select
15:7	-	Reserved

Register Description:

1. The BKGD_MIX register specifies the background mix and color source (see "Mix Control" codes on page 8-24)

Draw Command Register				
9AE8 (W)	**CMD**			
	-	**8514/A**	**Mach 8**	**Mach 32**

Bit	Mnemonic	Description
0	RW	**0** = read (READ) **1** = write (WRITE)
1	PIXEL_MODE	**0** = single pixel mode (SINGLE) **1** = nibble mode (NIBBLE)
2	LAST_PEL_OFF	**0** = draw last pixel **1** = do not draw past pixel
3	DIR_TYPE	**0** = coordinate mode (XY) **1** = degree mode (DEGREE)
4	DRAW	**0** = no draw **1** = draw
7:5	OCTANT (Bit 3=0)	Bit 7 (YPOS): **0** = negative Y direction **1** = positive Y direction Bit 6 (YMAJOR) **0** = Xmajor **1** = Y major Bit 5, XPOS: **0** = negative X direction **1** = positive X direction
	DEGREES (Bit 3=1)	Drawing direction: **0** = 0 degrees **1** = 45 degrees **2** = 90 degrees **3** = 135 degrees **4** = 180 degrees **5** = 225 degrees **6** = 270 degrees **7** = 315 degrees
8	CPU_WAIT	**0** = no wait for host data **1** = wait for host data
9	DATA_WIDTH	**0** = 8 bit data (Bit8) **1** = 16 bit data (Bit16)
11:10	-	Reserved (set to 00)
12	LSB_FIRST	**0** = use MSB of host data first in BIT16 data width **1** = use LSB of host data first in BIT16 data width

Draw Command Register				
9AE8 (W)	CMD			
	-	8514/A	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
15:13	OPCODE	Draw function: **0** = NO_OP — Short Stroke Setup **1** = DRAW_LINE — Draw Line **2** = FILL_RECT_HOR — Fill Rectangle (horizontal direction first) **3** = FILL_RECT_VPIX — Fill Rectangle (vertical direction first) **4** = FILL_RECT_VNIB — Fill Rectangle (nibbles, up and down) **5** = DRAW_POLY_LINE — Draw Polygon Boundary Line **6** = BLIT — Copy Screen Area (Blit) **7** = Reserved		

Register Description:

Some of the register fields are as follows:

RW (CMD[0])

When CMD[0] = 0, pixels are read from the draw trajectory and fed to the host through PIX_TRANS(R). When CMD[0] = 1, data is written to the draw trajectory. Blit is an exception to the rule, in that it reads from the blit source trajectory. Read and write are exclusive (except for BLIT.)

PIXEL_MODE (CMD[1])

A nibble is a horizontal group of four contiguous pixels aligned on a modulo 4 boundary. CMD[1] determines the mapping of host data to pixels.

In SINGLE pixel mode, one byte of host data is produced or consumed for every pixel read or drawn. SINGLE mode is appropriate for color reads and writes. For monochrome-single-write mode, each pixel uses a separate byte of host data, and only the bit of host data corresponding to the pixel's position within its nibble is used.

In NIBBLE mode, one byte of host data is produced or consumed for every nibble (or fraction of a nibble) read or drawn. NIBBLE mode is appropriate for monochrome reads. For monochrome-nibble-write mode, each nibble in the draw trajectory uses a new byte of host data.

If host data is not involved, using NIBBLE mode yields a significant performance improvement.

DIR_TYPE (CMD[3])

This field specifies the type of draw direction as either radial (DEGREE) or coordinate-based (XY). The direction is then specified by CMD[7:5] of the CMD register

DRAW (CMD[4])

When CMD[4] = "no draw", all side effects occur, but no data is read or written from/to either host or VRAM . At first glance, this would appear to be a "move to". However, it is more efficient to achieve this by simply updating CUR_X and CUR_Y, as the graphics processor still calculates the draw trajectory for a "no draw" command.

The real intent behind the "no draw" option is the efficient implementation of "pick" operations in CAD type applications. By setting the scissor rectangle to a small "pick" region near a pointing cursor and "re-drawing" the screen with "no draw", the application can determine which lines fall within the pick region by monitoring SUBSYS_STAT[1].

Note that CMD.DRAW has no effect on the FILL_RECT_VNIB fill rectangle operation or the blit.

CPU_WAIT (CMD[8])

If CMD[8] = "wait for host data", the graphics processor waits for MSB of host I/O.

This is used for host-VRAM image transfer and for patterned draws. Note that either color source or monochrome source must be set to "host" if host data is to affect VRAM writes.

DATA_WIDTH (CMD[9])

All host I/O is 16 bit I/O. If this field is "8-bit data", only the least significant byte (LSB) of PIX_TRANS is used; otherwise, both MSB and LSB are used.

LSB_FIRST (CMD[12])

This field affects the byte order for 16 bit host I/O. When CMD[9] = 1 (16-bit data), the MSB of PIX_TRANS is drawn first if CMD[12] = 0, and LSB is drawn first if CMD[12] = 1. This rule

also applies to host reads and monochrome data. CMD[12] has no effect when data-width is 8-bit (CMD[9] = 0).

Side Effects — Draw operations have side effects such as changing CUR_X, CUR_Y, DESTX_DIASTP, DESTY_AXSTP, ERR_TERM, and MAJ_AXIS_PCNT. These should be carefully noted for each draw operation. The term "side effects" refers only to register side effects. Side effects occur independently of the settings of CMD[0] and CMD[4].

OP_CODE (CMD[15:13]) OP_CODE 7 should never be used. Although it is implemented as a "NO OP" on the ***mach32***, if used, it will set the IBM 8514/A into an indefinite GP_BUSY state which can be cleared only by SUBSYS_CNTL[15:14].

Short Stroke Setup (OP_CODE = 0)

This op code is used to set up Short Stroke Vector drawing. SSV operation is the same as "Draw Line" operation unless otherwise noted. The host sends Short Stroke Vectors to the graphics processor through the SHORT_STROKE register.

DRAW No effect. The "move/draw" bit of a SHORT_STROKE command byte overrides CMD[4].

DATA_WIDTH The graphics processor always processes two SSV commands per word.

CPU_WAIT When CMD[0] = WRITE and host data is being written, the first SSV of every pair does not consume host data correctly. Therefore the first SSV of every pair should be 0 (a NO OP).

LSB_FIRST When CMD[12] = 1 the LSB of SHORT_STROKE is processed first. Otherwise the MSB of SHORT_STROKE is used first.

Side Effects — CUR_X, CUR_Y, and ERR_TERM behave as for "Line Draw". MAJ_AXIS_PCNT is replaced by SHORT_STROKE[3:0].

Draw Line (OP_CODE = 1)

Line draw, polygon boundary draw, and short stroke vector draw all share the same basic behaviour.

LAST_PEL_OFF When CMD[2] = 0, MAJ_AXIS_PCNT + 1 pixels are drawn/read. When CMD[2] = 1, only MAJ_AXIS_PCNT pixels are drawn. This bit does not alter the side effects on CUR_X or CUR_Y or ERR_TERM.

DIR_TYPE OCTANT If CMD[3] = DEGREE, lines are drawn in one of eight directions. If CMD[3] = XY, the current Bresenham parameters are used in the specified octant.

Side Effects — (CUR_X, CUR_Y) becomes (CUR_X+dx, CUR_Y+dy). MAJ_AXIS_PCNT is not changed. ERR_TERM is updated.

Fill Rectangle Type FILL_RECT_HOR (OP_CODE = 2)

In this mode rectangles are filled using horizontal lines. The CMD register bitfields have the following effects.

LAST_PEL_OFF When this bit is set, the last pixel in each scan line is dropped.

DIR_TYPE OCTANT For CMD.DIR_TYPE = XY, the "X direction" and "Y direction" bits are observed, and the "X,Y major" bit is ignored. For example, if CMD[7:5] = 0, the rectangle is drawn above and to the left of (CUR_X, CUR_Y). If CMD[3] = DEGREE, then the quadrant is determined by (CMD[15:13]/2) (0 and 1 map to the upper right quadrant, 2 and 3 map to upper left quadrant, etc).

Side Effects — (CUR_X, CUR_Y) becomes (CUR_X±width, CUR_Y±height)).

Fill Rectangle Type FILL_RECT_VPIX (OP_CODE = 3)

In this mode rectangles are filled using vertical lines. The CMD register bit fields have the following effects.

PIXEL_MODE Effectively ignored. This operation always works in SINGLE pixel mode.

LAST_PEL_OFF When this bit is set, the last pixel in each column is dropped.

DIR_TYPE OCTANT For CMD[3] = XY, the "X direction" and "Y direction" bits are observed, and the "X,Y major" bit is ignored. For example, if CMD[7:5] = 0, the rectangle is drawn above and to the left of (CUR_X, CUR_Y). If CMD[3] = DEGREE, then the quadrant is determined by (CMD[7:5]/2), (0 and 1 map into the upper right quadrant, 2 and 3 into upper left quadrant, etc).

Side Effects — (CUR_X, CUR_Y) becomes (CUR_X±width, CUR_Y±height)).

Fill Rectangle Type FILL_RECT_VNIB (OP_CODE =4)

In this mode, rectangles are filled in a vertical trajectory a nibble wide. The vertical direction alternates between up and down. The CMD register bitfields have the following effects.

PIXEL_MODE Effectively ignored. This operation always works in NIBBLE mode.

LAST_PEL_OFF NO effect on FILL_RECT_VNIB rectangles.

DIR_TYPE OCTANT For CMD[3] = XY, the "X direction" and "Y direction" bits are observed, and the "X,Y major" bit is ignored. For example, if CMD[7:5] = 0, the rectangle is drawn above and to the left of (CUR_X, CUR_Y). If CMD[3] = DEGREE, then the quadrant is determined by (CMD[7:5]/2), (0 and 1 map to the upper right quadrant, 2 and 3 to upper left quadrant, etc).

DRAW CMD[4] has no effect on the FILL_RECT_VNIB rectangle fill command.

Side Effects — For FILL_RECT_VNIB, (CUR_X, CUR_Y) remains unchanged.

Draw Polygon Boundary Line (OP_CODE = 5)

This operation is a variant of DRAW_LINE used to draw polygon boundaries in preparation for polygon filling. In all respects it behaves like "Draw Line", except that only one pixel is drawn/read per scan line. When the X coordinate of the draw trajectory goes to the left of the left scissor value, pixels are plotted on the left scissor boundary line (although CUR_X is not affected).

During a poly line draw, SUBSYS_STAT[1] behaves as during a normal line draw - the condition is tested *before* left scissor clamping.

RW For mono reads see *Draw Line*. For color reads, if CMD[0] = READ, 0xFF is returned for the pixels that are skipped on the draw trajectory. For CMD[0] = WRITE, and CMD[8]=1, host data is consumed for the skipped pixels.

LAST_PEL_OFF For correct polygon fill operation, CMD[2] should be set to logical one so as to disable the last scan line.

BLIT (CMD[15:13] = 6)

Blit copies pixels from a source rectangle located at (CUR_X, CUR_Y) to a destination rectangle located at (DESTX_DIASTP, DESTY_AXSTP). The blit operation is mediated by an 8 element nibble-wide blit buffer between source and destination. Up to 32 pixels are copied into the buffer from source and then are copied (with suitable rotation if necessary) into destination.

For a color blit to operate correctly, the foreground color source must be set to blit source by setting the COLOR_SRC field of FRGD_MIX. For a monochrome blit to operate correctly, the MONO_SRC bitfield of PIX_CNTL must be set to blit source.

RW Read mode ignores CMD[1], behaving as if CMD[1] = NIBBLE, reads from the source trajectory, and still draws to the blit destination.

LAST_PEL_OFF No effect.

DIR_TYPE OCTANT These bitfields are interpreted as for the rectangle fill op codes.

DRAW CMD[4] has no effect on blit. In particular, if CMD[0] = READ, GE_STAT[8] will be set even when CMD[4] = 0.

Side Effects — (DESTX_DIASTP, DESTY_AXSTP) becomes (DESTX_DIASTP, DESTY_AXSTP±dy) depending on quadrant where:

dy = MIN_AXIS_PCNT+1.

(CUR_X, CUR_Y), MAJ_AXIS_PCNT and MIN_AXIS_PCNT remain unchanged.

POLYGON FILL MODES

The 8514/A has two polygon fill modes. Both types are intended to be used during a FILL_RECT_HOR (Fill Rectangle) operation. Both use monochrome reads to detect outlines — whenever detected the Fill State is toggled. During FILL_RECT_HOR Rectangle Fill and BLIT, the Fill State is set to 0 at the beginning of each scan line.

Polygon Fill Type A

This mode is selected when PIX_CNTL is set to XXXXX10X. RD_MASK is used for the monochrome outline detect. That means for a pixel to be recognized as an outline all bits selected by RD_MASK must be high.

WRT_MASK selects the planes to which the filled polygon is written. However, those planes where both read and write masks are "high" are cleared. In planes where the read mask is "on", the write mask is treated as "off" for the purposes of color compare and mix. The right edges of the outline are not included in the fill, but the left edges are.

Outline Plane(s): 00000000100000000100000000010001000000100000

Fill Plane(s): 00000000111111111000000000011110000000111111

Polygon Fill Type B

This mode is selected when MULTIFUNC_CNTL [PIX_CNTL] is set to XXXXX11X. The write mask selects both the outline and the fill planes. For a pixel to be recognized as an outline all bits selected by WRT_MASK must be high. RD_MASK must be non-zero, but is otherwise ignored. Both left and right edges are included in the resulting filled area.

Before (Outline): 00000000100000000100000000010001000000100000

After (Fill): 00000000111111111000000000011111000000111111

All draw commands use polygon fill mode if it is selected, but only FILL_RECT_HOR Rectangle Fill and BLIT reset the Fill State to 0 at the beginning of every scan line. Polygon fill mode is therefore useful only for those two commands.

Fill State	Write Mask	Read Mask	Result	
			Mode A	Mode B
0	0	0	Destination	Destination
0	0	1	Destination	Destination
0	1	0	Destination	Destination
0	1	1	0	Destination
1	0	0	Destination	Destination
1	0	1	Destination	Destination
1	1	0	ALU	ALU
1	1	1	0	ALU

Destination Compare Color Register				
B2E8 (W)	CMP_COLOR			
	-	8514/A	Mach 8	Mach 32

Bit	Mnemonic	Description
15:0	DCC	Destination compare color is compared with the color destination pixel according to DEST_CMP_FN. The upper 8 bits are ignored in 8-bit mode.

Register Description:

1. The COLOR_CMP register specifies the destination comparison color. When a pixel on the draw trajectory is to be updated, it is compared to the value of COLOR_CMP using the function specified in PIX_CNTL[5:3]. If the result is False, the write succeeds; otherwise, the destination pixel is left alone.

2. Color compares are masked through the WRT_MASK register, (the color compare operation ignores unmasked planes).

Current X Register				
86E8 (R/W)	CUR_X			
	-	8514/A	Mach 8	Mach 32

Bit	Mnemonic	Description
10:0	CUR_X	The starting X coordinate for the first row of the destination blit
15:11	-	Reserved

Register Description:

1. The Current X Position register is used to determine the starting X coordinate of all drawing operations. Values written to this register will be considered to be in the range (-512 .. 1535).

2. Bit 12 which is normally reserved when performing 8514-compatible drawing operations is unused when performing extended drawing operations.

Current Y Register				
82E8 (R/W)	CUR_Y			
	-	8514/A	Mach 8	Mach 32

Bit	Mnemonic	Description
10:0	CUR_Y	The starting Y coordinate for the first row of the destination blit
15:11	-	Reserved

Register Description:

1. The Current Y Position register is used to determine the destination starting point of most drawing operations. Values written to this register will be considered to be in the range (-512 .. 1535).

2. Bit 12 which is normally reserved when performing 8514-compatible drawing operations is unused when performing extended drawing operations.

3. (CUR_X, CUR_Y) specifies the starting coordinates of all draw operations, with the exception of BLIT, for which it specifies the starting coordinates of the source rectangle.

4. (CUR_X, CUR_Y) is affected by the execution of draw commands.

5. These registers are interpreted as having a range of (-512 .. 1535)

Error Term Register				
92E8 (R/W)	ERR_TERM			
	-	8514/A	Mach 8	Mach 32

Bit	Mnemonic	Description
12:0	ERR_TERM	The Bresenham error term is within the range of (-4096 ... 4096)
15:13	-	Reserved

Register Description:

1. The ERR_TERM register defines the initial Bresenham error term for the current line draw. Before starting a line draw ERR_TERM should be set to:

   2* min (|dx|, |dy|) - max (|dx|, |dy|) - 1 (if starting x < ending x)

   2* min (|dx|, |dy|) - max (|dx|, |dy|) (if starting x >= ending x)

Foreground Color Register				
A6E8 (W)	FRGD_COLOR			
	-	8514/A	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
15:0	FRGD_COLOR	Determines the foreground color. Upper 8 bits (15:8) ignored in 8-bit mode.		

Foreground Mix Register				
BAE8 (W)	FRGD_MIX			
	-	8514/A	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
4:0	MIX_FN	Foreground Mix Value		
6:5	COLOR_SRC	Foreground Source Select		
15:7	-	Reserved		

Register Description:

1. The FRGD_MIX register specifies the foreground mix and color source. See "Mix Codes" on page 8-24.

Graphics Engine Status Register				
9AE8 (R)	GE_STAT			
	-	8514/A	Mach 8	Mach 32

Bit	Mnemonic	Description
0	FIFO_OCCUPIED0	**1** = FIFO postion 0 occupied
1	FIFO_OCCUPIED1	**1** = FIFO position 1 occupied
2	FIFO_OCCUPIED2	**1** = FIFO position 2 occupied
3	FIFO_OCCUPIED3	**1** = FIFO position 3 occupied
4	FIFO_OCCUPIED4	**1** = FIFO position 4 occupied
5	FIFO_OCCUPIED5	**1** = FIFO postion 5 occupied
6	FIFO_OCCUPIED6	**1** = FIFO position 6 occupied
7	FIFO_OCCUPIED7	**1** = FIFO position 7 occupied
8	DATA_READY	**0** = No data available for reading by host **1** = Data is ready to be read by host
9	GE_BUSY	**0** = graphics processor is idle **1** = graphics engine is busy executing a draw command
15:10	-	Reserved

Register Description:

1. This register provides the FIFO status information, host read data status, and the hardware busy status.

2. Bits 0:7 indicate the status of the eight FIFO entries, whether it has data or not. (position 0 is the first entry, position 1 the second, and so on.

Major Axis Pixel Count Register				
96E8 (W)	MAJ_AXIS_PCNT			
	-	8514/A	Mach 8	Mach 32

Bit	Mnemonic	Description
10:0	COUNT	Defines the major axis pixel count or Bresenham line length. The major axis pixel count is within the range of (0 ... 2047)
15:11	-	Reserved

Register Description:

1. This register defines both the rectangle width for a blit/rectangle draw and the line length for a line draw. Before starting a line draw MAJ_AXIS_PCNT should be set to max(|dx|,|dy|). For rectangle fill or BLIT use, set to horizontal rectangle dimension - 1.

2. CMD[2] has a one pixel effect on the width of FILL_RECT_HOR rectangles.

3. This register is read back at 96EEh.

Minor Axis Pixel Count Register				
BEE8*0 (W)	MIN_AXIS_PCNT			
	-	8514/A	Mach 8	Mach 32

Bit	Mnemonic	Description
10:0	-	Rectangle height (unsigned)
11	-	Reserved
15:12	MF_INDEX	Multi-Function Index (set to 0)

* Index 0 of port address BEE8.

Register Description:

1. This register defines the rectangle height for blit/rectangle fill. Before starting these operations MIN_AXIS_PCNT should be set to vertical rectangle dimension -1.

2. CMD[2] has a one pixel effect on the height of FILL_RECT_VPIX rectangles.

Monochrome Pattern High Register				
BEE8*9 (W)	PATTERN_H			
	-	8514/A	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
0	-	Reserved		
1	-	Monochrome pattern for pixel 3		
2	-	Monochrome pattern for pixel 2		
3	-	Monochrome pattern for pixel 1		
4	-	Monochrome pattern for pixel 0		
11:5	-	Reserved		
15:12	MF_INDEX	Multi-Function Index (set to 9h)		

* Index 9 of port address BEE8.

Register Description:

1. This register defines the monochrome pattern for all odd numbered nibbles in a trajectory when the Pattern registers are selected as monochrome source in MULTIFUNC_CNTL {PIX_CNTL}.

Monochrome Pattern Low Register				
BEE8*8 (W)	PATTERN_L			
	-	8514/A	Mach 8	Mach 32

Bit	Mnemonic	Description
0	-	Reserved
1	-	Monochrome pattern for pixel 3
2	-	Monochrome pattern for pixel 2
3	-	Monochrome pattern for pixel 1
4	-	Monochrome pattern for pixel 0
11:5	-	Reserved
15:12	MF_INDEX	Multi-Function Index (set to 8h)

* Index 8 of port address BEE8.

Register Description:

1. This register defines the monochrome pattern for all even numbered nibbles in a trajectory when the Pattern registers are selected as monochrome source in MULTIFUNC_CNTL {PIX_CNTL}.

Pixel Control Register				
BEE8*A (W)	PIXEL_CNTL			
	-	8514/A	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
0	-	Reserved		
1	POLY_FILL_TYPE	Select Polygon Fill **0** = Select Polygon Fill Mode A **1** = Select Polygon Fill Mode B		
2	POLY_FILL_ENA and MONO_READ	Polygon Fill Enable, Monochrome Read Enable — **1** = Compute monochrome data from the read/blit-source trajectory. One of two special polygon fill modes is used on the draw/destination trajectory.		
5:3	COLOR_CMP_FN	Color Compare Function — when result of comparison is True, destination data is not overwritten: **0** = False (destination overwritten) **1** = True (destination not overwritten) **2** = Destination Color >= COLOR_CMP register **3** = Destination Color < COLOR_CMP register **4** = Destination Color != COLOR_CMP register **5** = Destination Color = COLOR_CMP register **6** = Destination Color <= COLOR_CMP register **7** = Destination Color > COLOR_CMP register		
7:6	MONO_SRC	Monochrome Source **0** = Bit pattern is always 1 **1** = MULTIFUNC_CNTL [PATTERN_L, PATTERN_H] **2** = CPU Data (via PIX_TRANS) **3** = Blit Source (using monochrome read)		
11:8	-	Reserved		
15:12	MF_INDEX	Multi-Function Index (set to Ah)		

* Index A of port address BEE8.

Register Description:

1. MONO_SRC — All monochrome write operations involve not only a foreground and a background color (from sources selected in FRGD_MIX and BKGD_MIX) but also a monochrome pattern. When mono patterns are not desired, set MONO_SRC to 0 ("always 1"), which causes only the foreground source to participate.

   If both COLOR_SRC_BLIT and MONO_SRC_BLIT are selected, then bit 7 of color will be replaced by the mono bit.

Pixel Transfer Register				
E2E8 (R/W)	PIX_TRANS			
	-	8514/A	Mach 8	Mach 32

Bit	Mnemonic	Description
7:0	-	Color Data bits 7:0
		Monochrome Data Bit 0 = Reserved Bit 1 = Data bit 3 Bit 2 = Data bit 2 Bit 3 = Data bit 1 Bit 4 = Data bit 0 Bits 7:5 = Not used
15:8	-	Color Data bits 15:8
		Monochrome Data Bit 8 = Reserved Bit 9 = Data bit 7 Bit 10 = Data bit 6 Bit 11 = Data bit 5 Bit 12 = Data bit 4 Bits 15:13 = Not used

Register Description:

1. Image and pattern data are transferred between the host and adapter through PIX_TRANS. For monochrome image transfer, read/write data is aligned by nibble boundary on the screen. When CMD[9]=1 (bit 16 data width), two nibbles of data can be transferred for each read/write. Also, in 8 bit image transfers, two pixels can be transferred for each read/write operation.

2. For a monochrome read, the bit value returned for each pixel P is logical one if ((P | ~RD_MASK) == 0xFF), otherwwise the value is logical zero.

3. For monochrome writes bit value = 0 will write BKGD_MIX.COLOR_SRC, and bit value = 1 will write FRGD_MIX.COLOR_SRC

4. The exact usage of the high and low bytes of this register depends on the DATA_WIDTH and LSB_FIRST fields of the CMD register.

5. For image or pattern transfers from host to adapter, either monochrome source or color source should be designated as the host.

Read Mask Register				
AEE8 (W)	RD_MASK			
	-	8514/A	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
15:0	RD_MASK	For each Read Mask bit: **0** = Plane Compare Disabled **1** = Plane Compare Enabled		

Register Description:

1. The Read Mask register selects video memory planes as monochrome source data under the following circumstances:

   - Read monochrome data from screen memory
   - VRAM to VRAM blit with blit source data selected as monochrome source data
   - Convert polygon outline data

2. Each byte of the source data is tested for 0xFF. If true, foreground data; if false, background data:

```
(SrcColor OR (NOT ReadPlaneSelect)) == 0xFF
```

3. RD_MASK is never rotated in ATI operations (but is in some IBM operations). For IBM-only monochrome reads, the value of RD_MASK is the "true" mask rotated left one bit — bit 0 = mask for Plane 7, bits 1:7 = mask for Planes 0:6 respectively. The upper 8 bits (15:8) are ignored in non 16-bit color modes.

4. RD_MASK participates only in monochrome reads and poly fills. See PIX_CNTL[2] (BEE8*A). RD_MASK has no effect on color reads.

Bottom Scissor Register				
BEE8*3 (W)	**SCISSOR_B**			
	-	**8514/A**	**Mach 8**	**Mach 32**
Bit	**Mnemonic**	**Description**		
10:0	SCISSOR_B	Bottom Scissor Coordinate		
11	-	Reserved		
15:12	MF_INDEX	Multi-Function Index (set to 3h)		

* Index 3 of port address BEE8.

Left Scissor Register				
BEE8*2 (W)	**SCISSOR_L**			
	-	**8514/A**	**Mach 8**	**Mach 32**
Bit	**Mnemonic**	**Description**		
10:0	SCISSOR_L	Left Scissor Coordinate		
11	-	Reserved		
15:12	MF_INDEX	Multi-Function Index (set to 2h)		

* Index 2 of port address BEE8.

Right Scissor Register				
BEE8*4 (W)	**SCISSOR_R**			
	-	**8514/A**	**Mach 8**	**Mach 32**
Bit	**Mnemonic**	**Description**		
10:0	SCISSOR_R	Right Scissor Coordinate		
11	-	Reserved		
15:12	MF_INDEX	Multi-Function Index (set to 4h)		

* Index 4 of port address BEE8.

Top Scissor Register				
BEE8*1 (W)	**SCISSOR_T**			
	-	**8514/A**	**Mach 8**	**Mach 32**
Bit	**Mnemonic**	**Description**		
10:0	SCISSOR_T	Top Scissor Coordinate		
11	-	Reserved		
15:12	MF_INDEX	Multi-Function Incex (set to 1h)		

* Index 1 of port address BEE8.

Short Stroke Vector Transfer Register				
9EE8 (W)	SHORT_STROKE			
	-	8514/A	Mach 8	Mach 32

Bit	Mnemonic	Description
3:0	LENGTH_LO	Stroke Length
4	DRAW_LO	**0** = No draw **1** = Draw
7:5	OCTANT (when CMD[3]=0)	Bit 7 =YPOS **0** = negative Y direction **1** = positive Y direction Bit 6 = YMAJOR **0** = Xmajor **1** = Y major Bit 5 = XPOS **0** = negative X direction **1** = positive X direction
	DEGREES when CMD[3]=1) DRAW_DIR_LO	Drawing Direction **0** = 0 degrees **1** = 45 degrees **2** = 90 degrees **3** = 135 degrees **4** = 180 degrees **5** = 225 degrees **6** = 270 degrees **7** = 315 degrees
11:8	LENGTH_HI	Stroke Length
12	DRAW_HI	**0** = No draw **1** = Draw
15:13	DRAW_DIR_HI	Drawing Direction **0** = 0 degrees **1** = 45 degrees **2** = 90 degrees **3** = 135 degrees **4** = 180 degrees **5** = 225 degrees **6** = 270 degrees **7** = 315 degrees

Source X Destination X Register					
8EE8 (W)		SRC_X/DESTX_DIASTP			
		-	8514/A	Mach 8	Mach 32
Bit	Mnemonic	Description			
10:0	SRC_X (ATI blits only)	Field is Source X. It determines the starting X coordinate (for the first row) of the blit source.			
	DEST_X (IBM blits only)	Field is Destination X. It determines the starting X coordinate of the blit destination and trajectory.			
15:11	-	Field is 0 for IBM drawing operations			

Diagonal Step Register					
8EE8 (W)		SRC_X/DESTX_DIASTP			
		-	8514/A	Mach 8	Mach 32
Bit	Mnemonic	Description			
12:0	DIASTP	Diagonal Step Constant — should be set to 2*min(|dx|,|dy|) - 2*max(|dx|,|dy|) before initiating Bresenham linedraw operations. Valid range for this constant is (-4096 .... 0). In normal operation, it is always zero or negative.			
15:13	-	Reserved			

Register Description:

1. This register functions as the Source X (SRC_X) register when it specifies the ATI blit source starting coordinate.

   This register functions as the Destination X (DESTX) register when it specifies the IBM blit destination coordinate and trajectory.

   This register functions as the Diagonal Step (DIASTP) register when it specifies the diagonal step constant for Bresenham linedraw operations.

2. DESTX_DIASTP is a VESA register name.

Source Y Destination Y Register				
8AE8 (W)	SRC_Y/DESTY_AXSTP			
	-	8514/A	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
10:0	SRC_Y (ATI blits only)	When field is Source Y, it determines the starting Y coordinate (for the first row) of the blit source;		
	DEST_Y (IBM blits only)	When field is Destination Y, it determines the starting Y coordinate of the blit destination and trajectory.		
15:11	-	Reserved		

Axial Step Register				
8AE8 (W)	SRC_Y/DESTY_AXSTP			
	-	8514/A	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
12:0	AXSTP	Axial Step Constant — should be set to 2*min(|dx|,|dy|) before initiating Bresenham linedraw operations. Valid range is (0 ... 4095).		
15:13	-	Reserved		

Register Description:

1. This register functions as the Source Y (SRC_Y) register when it specifies the ATI blit source starting coordinate.

   This register functions as the Destination Y (DESTY) register when it specifies the IBM blit destination coordinate and trajectory.

   This register functions as the Axial Step (AXSTP) register when it specifies the axial step constant for Bresenham linedraw operations.

2. DESTY_AXSTP is a VESA register name.

Write Mask Register				
AAE8 (W)	WRT_MASK			
	-	8514/A	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
15:0	WRT_MASK	Write Mask selectively protects one or more video memory planes from updates. Upper 8 bits ignored in 8-bit mode. For each write mask bit: **0** = Writing to the associated plane disabled **1** = Writing to the associated plane enabled		

Register Description:

1. The WRT_MASK register specifies the bit-planes that are enabled for VRAM update. This register also participates in polygon fill modes. Planes which are masked out by the Write Mask register do not participate in any arithmetic ALU operations or destination color compares.

2. For **ATI68800-6**, see F2EE Destination Color Compare Mask register.

9

Coprocessor Register Extensions

CRT Control

The 8514/A CRT controller registers are used in conjunction with the extended CRT controller registers to provide compatibility in the extended modes.

- The shadow set control registers, SHADOW_SET (5AEE) and SHADOW_CTL (46EE), are used to load the shadow sets.
- The CLOCK_SEL (4AEE) register is used to select one of 16 pixel clock frequencies from the 1881X clock chip. See Appendix E for clock synthesizer specifications.
- The CRT_PITCH (26EE) register controls the logical width of the CRT.
- The CRT_OFFSET_HI (2EEE) and CRT_OFFSET_LO (2AEE) registers specify the physical start of screen memory.

The ***mach32*** graphics controller provides a CRT controller that is compatible with the IBM 8514/A CRT controller. CRTC extensions allow variable length lines in the display buffer in order that extended graphics modes such as 800x600 may be economically achieved without a significant waste of video buffer memory. Shadow CRT registers allow the ***mach32*** to provide standard resolutions on non-standard monitors such as CRTs that support 1024x768 non-interlaced display modes.

Registers that describe horizontal values are specified in multiples of 8 pixels. Registers that specify vertical amounts are specified in lines, although values written to the vertical control registers are not linear, and depend on the contents of the Y_CONTROL field of the Extended Display Control register.

If the DISP_CNTL[INTERLACE] field is logical one, the Y Counter will count twice for each line. Current Y is incremented once in the middle of the line, and once at the end; although horizontal syncs are still only generated at the end of each line. If the DISP_CNTL[DOUBLE_SCAN] field is logical one, the Y Counter will count each line treating bit D11 of the CRT controller Y register as if it were of a lower significance than bit 0. For example, the Y counter counts 0, 800h, 1, 801h, 2, 802h, and so on.

If the DISP_CNTL[Y_CONTROL] field is set to zero, bits 1 and 2 of the Y counter are skipped. A carry out of bit zero is fed into bit 3. Bits 1 and 2 remain zero. For example, the Y counter counts 0,1,8,9,10h,11h....

If the DISP_CNTL[Y_CONTROL] field is set to one, bit 2 of the Y counter is skipped. A carry out of bit one is fed into bit 3. Bit 2 will remain zero. For example, the Y counter counts 0,1,2,3,8,9,0Ah,0Bh,10h,11h,12h,13h....

The current Y counter is reset at the beginning of the line or half-line immediately following the line or half-line on which current Y first becomes equal to the contents of the Vertical Total register.

The vertical blank signal is de-asserted when the first horizontal sync occurs at or immediately after the beginning of the frame (for interlaced modes, the vertical blank will not be de-asserted until the end of the first half-line on odd frames).

The vertical blank signal is asserted at the end of the line in which the vertical counter becomes equal to the contents of the Vertical Displayed register.

The vertical sync starts at the end of the line or half-line on which the current Y counter becomes equal to the Vertical Sync Start register. At that time, an internal vertical sync width counter is set to zero. The vertical sync width counter is incremented at the end of each line or half line. The vertical sync drops at the end of the line or half-line in which the contents of the vertical sync counter first becomes equal to the contents of the Vertical Sync Width register.

The ***mach32*** CRT controller runs in 8514/A-compatible mode or ATI-proprietary mode. The CRT controller mode is determined by which of the Advanced Function Control or the Clock Select register was last written.

In 8514/A-compatible mode, the CRT controller supports only 1024x768 and 640x480 resolutions in either 4-bit or 8-bit variants. CRT registers may be selectively locked when running in 8514-compatible mode. Two sets of shadow CRT Controller registers are provided for use by the ***mach32***. One set supplies default values for locked CRT controller registers when running 640x480 resolutions, the other provides default values for applications running 1024x768 resolutions. The shadow CRT registers allow the ***mach32*** to provide standard 8514/A resolutions on non-standard monitors, when running applications that expect standard 8514/A compatible monitors.

CRT registers may be selectively locked. When running in IBM-compatible mode, locked registers assume default values provided in one of two sets of shadow CRT registers, regardless of what was written to those registers by 8514/A applications. The primary register set is updated even if the specified register is marked as locked — the CRT controller however, uses the value in the appropriate shadow register.

Applications that support extended ***mach32*** graphics modes may write directly to the extended CRT controller registers.

Refer to the previous chapter for descriptions of the following IBM compatible registers:

- H_TOTAL (02E8)
- H_DISP (06E8)
- H_SYNC_STRT (0AE8)
- H_SYNC_WID (0EE8)
- V_TOTAL (12E8)
- V_DISP (16E8)
- V_SYNC_STRT (1AE8)
- V_SYNC_WID (1EE8)
- DISP_CNTL (22E8)
- ADVFUNC_CNTL (4AE8)

Clock Select Register				
4AEE (R/W)	CLOCK_SEL			
	-	-	Mach 8	Mach 32

Bit	Mnemonic	Description
0	PASS_THROUGH	**0** = Enabled — video output is driven by VGA data. from the features connector or the internal VGA controller. **1** = Disabled — video output is 8514. RAMDAC is driven by data from the video buffer.
1	-	Reserved
5:2	CLK_SEL	Selects pixel clock frequencies from the 1881x clock synthesizer. See Appendix E for pixel clock details.
6	CLK_DIV	To generate pixel clocks from a selected clock as follows (bit 7 reserved): **0** = Clock-divide-by-1 **1** = Clock-divide-by-2
7	REFRESH	**0** = Enable video memory refresh counter **1** = Force video memory refresh always
11:8	VFIFO_DEPTH	When the number of video FIFO entries is less than VFIFO_DEPTH, ***mach32*** will start filling the FIFO. Although this value is used only with DRAM, you can optimize performance at high resolutions by programming it regardless of memory type.
12	COMPOSITE_SYNC	***mach8***: **0** = Composite sync used when programming CRT shadow register sets 1 or 2 **1** = Composite sync monitor supported when using the primary CRT register set. ***mach32***: **0** = Separate sync **1** = Composite sync
15:13	-	Reserved for programmable clock chip control bits

Register Description:

1. This register read only in ***mach8*** mode. All data fields are for extended functions (Mach8 or Mach32).
2. This register is an extension of ADVFUNC_CNTL (4AE8). Writing it forces the CRT controller to ATI mode. Writing ADVFUNC_CNTL (4AE8) forces the CRT controller to 8514-compatible mode.

3. Shadow register sets are always used when running in 8514-compatible mode. They are normally set during hardware reset. Normal program applications should not modify them.

CRT Offset (High) Register				
2EEE (W)	CRT_OFFSET_HI			
	-	-	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
3:0	CRT_OFFSET_HI	The upper 4 bits of a 20-bit linear address offset (19:16) which points to the start of the CRT display buffer. Specified in bytes/4. Also see CRT_OFFSET_LO (2AEE).		
15:4	-	Reserved		

CRT Offset (Low) Register				
2AEE (W)	CRT_OFFSET_LO			
	-	-	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
15:0	CRT_OFFSET_LO	The lower 16 bits of a 20-bit linear address offset (15:0) which points to the start of the CRT display buffer. Specified in bytes/4. Also see CRT_OFFSET_HI (2EEE).		

CRT Pitch Register				
26EE (W)	CRT_PITCH			
	-	-	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
7:0	CRT_PITCH	The amount of video memory pixels for one row of video data — specified in units of eight pixels. This value is 128 in 8514-compatible mode (1024/8). Compare with GE_PITCH (76EE). ***mach8***: When ADVFUNC_CNTL (4AE8) or MEM_CNTL (BEE8[5] is written, CRT_PITCH is reset to 128.		
15:8	-	Reserved		

Shadow Set Control Register				
46EE (W)	SHADOW_CTL			
	-	-	Mach 8	Mach 32

Bit	Mnemonic	Description
0	LOCK_CRT_PARAMS	Display mode CRT parameters — double scan and interlace
1	LOCK_Y_CONTROL	Y_CONTROL field of DISP_CNTL (22E8)
2	LOCK_H_PARAMS	H sync parameters: position, width, and total
3	LOCK_H_DISP	Horizontal displayed
4	LOCK_V_PARAMS	V sync parameters: width, start, and total
5	LOCK_V_DISP	Vertical displayed
6	LOCK_OVERSCAN	***mach32*** only: Selects the shadow set of overscan registers in the 1024x768 screen resolution.
15:7	-	Reserved

Register Description:

1. The Shadow Control register contains lock bits that choose between the shadow and primary registers in the 1024x768 screen resolution.
2. CLOCK_SELECT (4AEE) is not lockable.

Shadow Set Register					
5AEE (W)	SHADOW_SET				
	-	-	Mach 8	Mach 32	
Bit	**Mnemonic**	**Description**			
1:0	SHADOW_SET	CRT register sets: **0** = Primary CRT register set **1** = Shadow set 1 — generally for 640x480 **2** = Shadow set 2 — generally for 1024x768			
7:2	-	Reserved			
9:8	LOAD_SRC/DST	**ATI68800-6:** Pointer to allow destination and source offset and pitch registers to be loaded independently. Allows unsymmetrical blit operations between any areas of the memory. See 6EEEh, 72EEh, and 76EEh. **0** = Load Destination and Source registers concurrently **1** = Load Destination registers **2** = Load Source registers			
15:10	-	Reserved			

Vertical Line Counter Register					
CEEE (R)	VERT_LINE_CNTR				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
10:0	VERT_LINE_CNTR	Vertical line counter			
15:11	-	Reserved			

Engine Setup

Setup registers are initialized by the boot ROM according to the configuration of the current system and adapter. Setup registers should not be modified.

Graphics engine control provides the selection of wait states enable and disable as well as selection of the host I/O bus width of either 8, 16, or 32 bits.

FIFO Options Register				
36EE (W)	FIFO_OPT			
	-	-	Mach 8	-

Bit	Mnemonic	Description
0	W_STATE_ENA	**0** = Wait states disabled unless FIFO is full. **1** = Wait states enabled — generates WS if FIFO is at least half full. Setting is enabled on power up. Wait states are always generated if FIFO is full. This field does not affect WS generation for 8514/A compatible registers.
1	HOST_8_ENA	**0** = 16-bit host data I/O — setting may not function correctly in some ISA systems. **1** = 8-bit host data I/O
15:2	-	Reserved

Register Description:

1. For ***mach32*** description, see MISC_OPTIONS (36EE) on page 9-12.
2. This register is initialized by the POST ROM. Normal program applications should not modify it.

Maximum Wait-States Register				
6AEE (R/W)	MAX_WAITSTATES			
	-	-	Mach 8	-

Bit	Mnemonic	Description
3:0	Q_WSTATES	Maximum number of wait-states (in clocks divided by 4) that the engine is allowed to generate when I/O writes are being performed. Preset to Ch on a hardware reset, no wait-state if set to 0h.
7:4	ROM_SPEED	Number of wait-states used by the engine when the host processor is reading the POST ROM. Preset to Fh (wait forever) on a hardware reset, no wait-state if set to 0h.
8	LINE_OPT_ENA	**0** = H Linedraw optimizations enabled **1** = H Linedraw optimizations disabled Horizontal linedraw optimizations must be disabled when performing degree-mode linedraws to ensure that ERR_TERM is 8514-compatible.
9	IOR16_ENA	**0** = 16-bit I/O read enabled **1** = 16-bit I/O read disabled IOR16_ENA is enabled on power up. If disabled, I/O read from PIX_TRANS will be executed as two 8-bit bus cycles. When in DRAM based implementations, resetting of this bit to zero may allow most monochrome data reads to complete without over-running.
10	PASSTHROUGH_ OVERRIDE	**0** = Passthrough connection made **1** = Passthrough connection not made, VGA syncs not detected over the passthrough. Proper DAC operation cannot be guaranteed if the setting of this bit does not reflect the system configuration correctly.
15:11	-	Not used

Register Description:

1. This Max Waitstates register description is applicable to the ***mach8*** only. See following pages for ***mach32***.
2. This register is initialized by the POST ROM. It should not be modified by normal program applications.

Maximum Wait-States Register				
6AEE (R/W)	MAX_WAITSTATES			
	-	-	-	Mach 32

Bit	Mnemonic	Description
3:0	Q_WSTATES	Maximum number of wait-states (in clocks divided by 4) that the engine is allowed to generate when I/O writes are being performed. Preset to Ch on a hardware reset, no wait-state if set to 0h.
7:4	ROM_SPEED	Number of wait-states used by the engine when the host processor is reading the POST ROM. It is set to Fh on a hardware reset. Zero wait-state if set to 0h.
8	LINE_OPT_ENA	**0** = H Linedraw optimizations enabled **1** = H Linedraw optimizations disabled Horizontal linedraw optimizations must be disabled when performing degree-mode linedraws to ensure that ERR_TERM is 8514-compatible.
15:9	-	Reserved

Register Description:

1. This Max Waitstates register description is applicable to the ***mach32*** only. See previous page for ***mach8*** description.
2. This register description is applicable to the ATI68800-6. 6AEE-RW for the ATI68800AX is APERTURE_CNTL, a PCI control register, described on page 9-76.

Miscellaneous Register					
36EE (R/W)	MISC_OPTIONS				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
0	W_STATE_ENA	**0** = Wait states disabled unless FIFO is full. **1** = Wait states enabled — generates WS if FIFO is at least half full. Setting is enabled on power up. This flag is valid only for ATI-extended operations Wait states are always generated if FIFO is full.			
1	HOST_8_ENA	**0** = 16-bit host data I/O — setting may not function correctly in some ISA systems. **1** = 8-bit host data I/O			
3:2	MEM_SIZE_ALIAS	**0** = 512KB **1** = 1MB **2** = 2MB **3** = 4MB			
4	DISABLE_VGA	**0** = VGA controller enabled **1** = VGA controller disabled			
5	16_BIT_IO	**0** = 16-bit 8514 I/O cycles disabled **1** = 16-bit 8514 I/O cycles enabled			
6	DISABLE_DAC	Disables local DAC			
7	DLY_LATCH_ENA	VRAM - serial data delay latch enable DRAM - memory data delay latch enable for bits 63:0			
8	TEST_MODE	Extends draw engine page write cycle to 3 clocks to meet tester requirements			
9	-	Reserved			
10	BLK_WR_ENA	**ATI68800-6:** Block Write Enable			
11	64_DRAW_ENA	**ATI68800-6:** 64-bit Draw Enable			
12	MEM_DATA_SEL	**0** = Flow-through video memory read data **1** = Latch video memory read data (by the rising edge of CASB)			
13	DLY_LATCH_ENA	Memory data delay latch enable for data bits 63:0			
14	LATCH_FUL_ENA	Memory data latch full clock pulse enable			
15	-	Reserved			

Register Description:

1. For ***mach8*** description, see FIFO_OPT (36EE-W) on page 9-9.

Engine Control

The extended engine control registers are divided into four classes: draw engine setup, data path control, ALU control, base address select for the drawing buffer, and draw control. Remember that the 8514/A-compatible engine control registers still apply.

Draw Engine Setup Registers

The draw engine setup registers are used primarily for mode initialization. They are:

- GE_PITCH (76EE)
- GE_OFFSET_LO (6EEE)
- GE_OFFSET_HI (72EE)
- EXT_GE_CONFIG (7AEE)

Extended Data Path Control Registers

The extended data path control registers are:

- DP_CONFIG (CEEE)
- PATT_LENGTH (D2EE)
- PATT_INDEX (D6EE)
- PATT_DATA_INDEX (82EE)
- PATT_DATA[F:0] (8EEE) - Color
- PATT_DATA[11:10] (8EEE) - Monochrome

Extended ALU Control Registers

The extended ALU control registers perform the same function as the 8514/A compatible registers except that they are not grouped with other pieces of information. They are:

- ALU_FG_FN (BAEE)
- ALU_BG_FN (B6EE)
- DEST_CMP_FN (EEEE).

Extended Draw Control Registers

The extended draw control registers include:

- DP_CONFIG (CEEE)
- LINEDRAW_OPT (A2EE)
- SCISSOR_TOP (DEEE)
- SCISSOR_BOTTOM (E6EE)
- SCISSOR_LEFT (DAEE)
- SCISSOR_RIGHT (E2EE)

Data Path Configuration Register				
CEEE (W)	DP_CONFIG			
	-	-	Mach 8	Mach 32

Bit	Mnemonic	Description
0	READ_WRITE	**0** = Read data from drawing trajectory **1** = Write data to drawing trajectory
1	POLY_FILL_MODE	**0** = Polygon-fill blit mode disabled **1** = Polygon-fill blit mode enabled, blit source mechanism triggered, VRAM source blit data used as the polygon mask for blits Each pixel of the source data is ORed with the inverted RD_MASK register. If the result bits are all logical one, the source pixel is considered a polygon outline boundary and the flag is inverted. This flag is cleared at the start of each row of the blit.
2	READ_MODE	**0** = Read host data — color data **1** = Read host data — monochrome data
3	-	Reserved
4	DRAW	**0** = Disables Draw **1** = Enables Draw
6:5	MONO_SRC	Monochrome data source: **0** = Always "1" **1** = Mono pattern register **2** = Pixel transfer register **3** = VRAM blit source
8:7	BG_COLOR_SCR	Background color source select: **0** = Background color register **1** = Foreground color register **2** = Pixel transfer register **3** = VRAM blit Ssurce
9	DATA_WIDTH	**0** = 8 bits **1** = 16 bits The GE may be instructed to ignore either the MSB or LSB byte of the 16-bit CPU Data Transfer register. Host data transfer is normally 16-bit wide. This bit must be set to logical one in 16bpp modes.

Data Path Configuration Register				
CEEE (W)	DP_CONFIG			
	-	-	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
11:10	-	Reserved		
12	LSB_FIRST*	**0** = Most-significant-byte first (D15:8) **1** = Least-significant-byte first (D7:0) This bit is not ignored when DATA_WIDTH=0 as for 8514/A draw operations.		
15:13	FG_COLOR_SRC	Foreground color source select: **0** = Background color register **1** = Foreground color register **2** = Pixel transfer register **3** = VRAM blit source **5** = Color pattern shift register		

*This bit is ignored in ***mach8*** mode when DATA_WIDTH=0; but is not ignored in ***mach32*** mode.

Extended FIFO STatus Register				
9AEE (R)	EXT_FIFO_STATUS			
	-	-	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
15:0	-	Extended FIFO status **0** = FIFO Entry 1 **1** = FIFO Entry 2 **2** = FIFO Entry 3 **3** = FIFO Entry 4 **4** = FIFO Entry 5 **5** = FIFO Entry 6 **6** = FIFO Entry 7 **7** = FIFO Entry 8 **8** = FIFO Entry 9 **9** = FIFO Entry 10 **10** = FIFO Entry 11 **11** = FIFO Entry 12 **12** = FIFO Entry 13 **13** = FIFO Entry 14 **14** = FIFO Entry 15 **15** = FIFO Entry 16		

Extended Graphics Engine Configuration Register				
7AEE (W)	EXT_GE_CONFIG			
	-	-	Mach 8(8)	-

Bit	Mnemonic	Description
0	EE_DATA_OUT	
1	EE_CLK	
2	EE_CS	
3	ALIAS_ENA	Must be set to logincal zero for EEPROM operations
4	Z1280	
6:5	-	Reserved
7	EE_SELECT	
15:8	-	Reserved

Register Description:

1. The above register description is mapped for 8-bit operation (i.e., the card is in an 8-bit slot or the 8/16 bit jumper is set for 8-bit, and ALIAS_ENA is set to logical zero.)

Extended Graphics Engine Configuration Register					
7AEE (W)	EXT_GE_CONFIG				
	-	-	Mach 8(16)	-	
Bit	**Mnemonic**	**Description**			
2:0	MONITOR_ALIAS	A monitor ID which is not necessarily that of the attached monitor.			
3	ALIAS_ENA	**0** = Reporting of MONITOR_ALIAS disabled **1** = Reporting enabled ALIAS_ENA allows the graphics engine to mislead 8514/A applications as to the true identity of the attached monitor with the content of SUBSYS_STATUS[MONITOR_ID] (42E8-R)			
11:4	-	Reserved			
12	EE_DATA_OUT	Data output for the EEPROM.			
13	EE_CLK	Clock signal for the EEPROM.			
14	EE_CS	Chip select line for the EEPROM.			
15	EE_SELECT	1 = Enable read/write of external EEPROM EEPROM will not respond until this field is set to 1.			

Register Description:

1. This register contains control fields which deal with the external EEPROM and the current configuration of the graphics controller.
2. Reads from the controllers are from the EE_DATA_IN field of EXT_GE_STATUS (62EE-R).
3. When reading or writing the EEPROM, the CRT controller must be disabled by setting DISP_CNTL[ENA_DISPLAY] (22E8) to 2 (reset 8514 CRT Controller).
4. To access the EEPROM, applications must first obtain the monitor alias from SUBSYS_STAT (42E8-R), and restore the monitor alias after the operation.

Extended Graphics Engine Configuration Register					
7AEE (W)	EXT_GE_CONFIG				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
2:0	MONITOR_ALIAS	Alternate monitor ID, for 8514 application use.			
3	ALIAS_ENA	This field allows the controller to mis-lead 8514 applications as to the ID of the connected monitor. **0** = Reports actual monitor ID **1** = Reports MONITOR_ALIAS as Monitor ID in the Subsystem Status register			
5:4	PIXEL_WIDTH	8 bits per pixel (bpp) when video memory is less than 1MB. 16 or 24 bpp when appropriate DAC (ATI68830, AT&T, Sierra, or TI) is supported. **0** = 4 bpp **1** = 8 bpp **2** = 16 bpp **3** = 24 bpp See bits 15,11 description below.			
7:6	16_BIT_COLOR_MODE	RGB bit organization within a 16-bit color word: **0** = (5,5,5): Red-(14:10), Green-(9:5), Blue-(4:0) **1** = (5,6,5): Red-(15:11), Green-(10:5), Blue-(4:0) **2** = (6,5,5): Red-(15:10), Green-(9:5), Blue-(4:0) **3** = (6,6,4): Red-(15:10), Green-(9:4), Blue-(3:0)			
8	MULTIPLEX_PIXELS	**1** = Four pixels processed by the DAC in parallel			
9	24_BIT_ COLOR_CONFIG	Bytes per pixel (Bpp) Memory organization select: **0** = 3 Bpp **1** = 4 Bpp Bit 10 = 0: LSByte in each 32 bit word reserved Bit 10 = 1: MSByte in each 32 bit word reserved			
10	24_BIT_ COLOR_ORDER	RGB bit organization within each 24- or 32-bit pixel is selected as follows: **0** = RGB(8,8,8) 3 Bpp: Red-(23:16), Green-(15:8), Blue-(7:0) 4 Bpp: Red-(31:24), Green-(23:16), Blue-(15:8) **1** = BGR (8,8,8) 3 Bpp: Blue-(23:16), Green-(15:8), Red-(7:0) 4 Bpp: Blue-(23:16), Green-(15:8), Red-(7:0)			
11	DISPLAY_PIXEL_ SIZE[1]	**ATI68800-6:** **1** = Display pixel size to be written			
13:12	DAC_EXT_ADDR	DAC address inputs RS(3:2) control			
14	DAC_8_BIT_EN	**0** = 6-bit DAC operation **1** = 8-bit DAC operation			

Extended Graphics Engine Configuration Register					
7AEE (W)	EXT_GE_CONFIG				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
15	DRAW_PIXEL_SIZE[1]	**ATI68800-6:** **1** = Drawing pixel size to be written			

Register Description:

1. Bits 11 and 15 are pointers that allow the display and drawing pixel sizes to be set independently. For compatibility, if bits 15 and 11 are both "0", then both display and drawing pixel sizes are written.

Graphics Engine Pitch Register				
76EE (W)	GE_PITCH			
	-	-	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
7:0	GE_PITCH	The length of each line in the display buffer — specified in units of eight pixels. ***mach8*** Only: Pitch is reset to 128 (1024/8) on power up and also whenever MEM_CNTL (BEE8[5]) or ADVFUNC_CNTL (4AE8) is updated. Compare with CRT_PITCH (26EE).		
15:8	-	Reserved		

Register Description:

1. The Drawing Pitch register determines the number of pixels on each line of the display buffer when drawing.

2. For **ATI68800-6:**
   Depending on the Shadow Set pointer LOAD_SRC/DST[9:8] (5AEE-W), source and destination offset and pitch registers are loaded either concurrently or separately, as follows:
   0 = Both source and destination offset/pitch registers are loaded concurrently
   1 = Destination offset/pitch registers are loaded
   2 = Source offset/pitch registers are loaded

Graphics Engine Offset High Register				
72EE (W)	GE_OFFSET_HI			
	-	-	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
3:0	-	The 4 MSB (19:16) of a 20-bit GE video buffer address offset (start of GE video buffer)		
15:4	-	Reserved		

Graphics Engine Offset Low Register				
6EEE (W)	GE_OFFSET_LO			
	-	-	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
15:0	-	The 16 LSB (15:0) of a 20-bit GE video buffer address offset (start of GE video buffer)		

Register Description:

1. The upper Offset bits are at GE_OFFSET_HI (72EE), the lower bits at GE_OFFSET_LO (6EEE).
2. The address offset is in linear format, specified in bytes/4, depending on the setting of DISP_CNTL[Y_CONTROL] (22E8).
3. For **ATI68800-6:**
   Depending on the Shadow Set pointer LOAD_SRC/DST[9:8] (5AEE-W), source and destination offset and pitch registers are loaded either concurrently or separately, as follows:
   0 = Both source and destination offset/pitch registers are loaded concurrently
   1 = Destination offset/pitch registers are loaded
   2 = Source offset/pitch registers are loaded

Linedraw Options Register					
A2EE (R/W)	LINEDRAW_OPT				
	-	-	Mach 8	Mach 32	
Bit	**Mnemonic**	**Description**			
0	-	Reserved			
1	POLY_MODE	**0** = Normal linedraw mode **1** = Polygon linedraw mode As in edge flag polygon fill algorithm			
2	LAST_PEL_OFF	**0** = Last pixel drawn **1** = Last pixel not drawn Applies to both raw Bresenham and direct linedraws			
3	DIR_TYPE	**0** = Bresenham/Octant Lines based on Bresenham parameters or specified octant **1** = Line length/degree Bresenham count determines length and OCTANT determines line direction Used by both raw Bresenham linedraw and short stroke vector operations.			
4	-	Reserved			
7:5	OCTANT (Bit 3=0)	Bit 7 - YDIR **0** = Decrement Y coordinate **1** = Increment Y coordinate Bit 6 - YMAJOR **0** = X-Major **1** = Y-Major Bit 5 - XDIR **0** = Decrement X coordinate **1** = Increment X coordinate			
	DEGREE (Bit 3=1)	**0** = 0° **1** = 45° **2** = 90° **3** = 135° **4** = 180° **5** = 225° **6** = 270° **7** = 315°			

Linedraw Options Register				
A2EE (W)	LINEDRAW_OPT			
	-	-	Mach 8	Mach 32

Bit	Mnemonic	Description
8	BOUNDS_RESET	**0** = No reset **1** = The four bounds accumulator registers reset to: Left: 2047 Top: 2047 Right: -2048 Bottom: -2048
10:9	CLIP_MODE	**0** = Clip exception disabled **1** = Stroked line segments **2** = Polygon boundary lines **3** = Patterned lines
15:11	-	Reserved

Read Extended Graphics Engine Configuration Register					
8EEE (R)		R_EXT_GE_CONFIG			
		-	-	-	Mach 32
Bit	**Mnemonic**	**Description**			
2:0	MONITOR_ALIAS	Monitor ID, for application use.			
3	ALIAS_ENA	**0** = Reports actual monitor ID **1** = Reports MONITOR_ALIAS as Monitor ID in Subsystem Status register			
5:4	PIXEL_WIDTH	Pixel width is forced to 8 bits per pixel (bpp) when video memory size is less than 1MB. 16 and 24 bpp modes allowed only if appropriate RAMDAC is installed. **0** = 4 bpp **1** = 8 bpp **2** = 16 bpp **3** = 24 bpp			
7:6	16_BIT_ COLOR_MODE	RGB bit organization within 16-bit color word: **0** = (5,5,5): Red-(14:10), Green-(9:5), Blue-(4:0) **1** = (5,6,5): Red-(15:11), Green-(10:5), Blue-(4:0) **2** = (6,5,5): Red-(15:10), Green-(9:5), Blue-(4:0) **3** = (6,6,4): Red-(15:10), Green-(9:4), Blue-(3:0)			
8	MULTIPLEX_PIXELS	**1** = Two bits are passed to the DAC in parallel			
9	24_BIT_ COLOR_CONFIG	Bytes per pixel (Bpp) Memory organization select: **0** = 3 bpp **1** = 4 bpp If 4 Bpp & 24_BIT_COLOR_ORDER field = 0, LSByte in each 32 bit word is reserved If 4 Bpp & 24_BIT_COLOR_ORDER field = 1, MSByte in each 32 bit word is reserved			
10	24_BIT_COLOR _ORDER	RGB bit organization within each 24- or 32-bit pixel is selected as follows: **0** = RBG(8,8,8) 3 Bpp: Red-(23:16), Blue-(15:8), Green-(7:0) 4 Bpp: Red-(31:24), Blue-(23:16), Green-(15:8) **1** = BGR (8,8,8) 3 Bpp: Blue-(23:16), Green-(15:8), Red-(7:0) 4 Bpp: Blue-(23:16), Green-(15:8), Red-(7:0)			
11	-	Reserved			
13:12	DAC_EXT_ADDR	DAC address inputs RS(3:2) control			
14	DAC_8_BIT_EN	**0** = 6-bit DAC operation **1** = 8-bit DAC operation			

Read Extended Graphics Engine Configuration Register				
8EEE (R)	R_EXT_GE_CONFIG			
	-	-	-	Mach 32

Bit	Mnemonic	Description
15	-	Reserved

Register Description:

1. 8EEE-R is the read address of EXT_GE_CONFIG which is at 7AEE (W).

Drawing Operations

Overview

Draw extensions include:

- Line draw with extended data path
- Short stroke vector with extended data path
- Horizontal raster scan
- Point-to-point line draw
- Non-conforming bitblts

These operations are initiated by:

- BRES_COUNT (96EE)
- EXT_SHORT_STROKE (C6EE)
- SCAN_TO_X (CAEE)
- LINEDRAW[3] (FEEE)
- DEST_Y_END (AEEE)

Extended trajectory control registers are:

- LINEDRAW[0-2] (FEEE)
- LINEDRAW[4-5] (FEEE)
- SRC_X_START (B2EE)
- SRC_X_END (BEEE)
- DEST_X_START (A6EE)
- DEST_X_END (AAEE)
- SRC_Y_DIR (C2EE)

Drawing (and clipping) capabilities of the ***mach32*** graphics processor are described here. Control is provided by configuration settings in registers associated with each type of drawing operation. Sample codes on engine-assisted pre-clipping and test conditions used in the modified Cohen-Sutherland algorithm are included.

Scissor rectangles define the area where pixels are drawn. Direct linedraws consist of placing line segments on the screen. Short-stroke vectors are vectors shorter than 16 pixels, and have directions that are multiples of 45 degrees. Short-stroke vectors are generally used for drawing text characters. Blit operations do rectangle fills and polygon fills.

Drawing Control registers are as follows:

- General Drawing Control Registers
- Scissor Registers
- Direct Linedraw Registers
- Short-Stroke Vector Register
- Raw Bresenham Linedraw Registers
- Blit Registers
- Scanline Draw to X
- Pattern Registers

General Drawing Control Registers

General drawing controls include datapaths, current X/Y positions, foreground/background and color/monochrome source/destination information for use in various drawing operations.

Scissor Registers

Scissor registers are used to inhibit drawing outside a rectangular area of a screen. This function is also called "Post-Clipping."

Scissor registers do not provide full clipping of drawing operations. If the coordinates are outside the range of (-512,-512) to (1535,1535), they are interpreted incorrectly by the engine. In that case, the host application may have to perform clipping of coordinates before passing them to the engine.

Despite this limitation, Scissor registers do have many useful applications when implementing graphics functions. For example, they may be used to generate interrupts on drawing operations which operate inside the scissor rectangle. The INSIDE_SCISSOR_ENA bit of the Subsystem Control register will enable this feature. Inside interrupts are useful for implementing high-speed pick operations.

The four ATI extended scissor registers operate in the range of (-2048....2047). This range is larger when compared to the standard IBM 8514/A compatible LEFT and TOP scissor registers which operate only in the range (-1024....1023), and the RIGHT and BOTTOM scissor registers which operate in the range (0....2047).

A pixel is considered to be inside the scissor rectangle if the values of the four ATI scissor registers (each 12-bit wide) have the following relationship:

SCISSOR_LEFT ≤ **CUR_X** ≤ **SCISSOR_RIGHT**

SCISSOR_TOP ≤ **CUR_Y** ≤ **SCISSOR_BOTTOM**

Data field names and port addresses of the four scissor registers are listed on the following page.

Direct Linedraw Registers

Direct Linedraw registers provide high-speed linedrawing capabilities. These extended registers allow applications to perform direct block transfers of line end-points to the ***mach32*** graphics engine. Coordinates are provided in (X,Y) format.

Full 16-bit precision direct line-clipping allows more than 99% of typical line-segments to be clipped at the engine without intervention by the host processor.

The operations of Linedraw registers are determined by the value in the Linedraw Register Index register. Therefore, before writing to any linedraw register, the index register must be set to a known value. Then with each write to the linedraw register, the index register value increments (by 1) to keep track of the data transfer.

If CLIP_MODE (in the Linedraw Options register) is not disabled, and the host processor attempts to draw a line segment which must be clipped, the ***mach32*** graphics engine raises an "exception" condition for the host to clip the line. A simple recovery mechanism then allows lines to be clipped at speeds approaching unclipped linedraw speed for all typical applications.

Pre-Clip Modes

Pre-clipping is supported by hardware in the ***mach32*** graphics engine. Pre-clip modes are set in the Linedraw Options register. Clip_Mode 0 disables clip exceptions. Clip_Mode 1 enables clipping of stroked plain lines. Clip_Mode 2 pre-clips polygon boundary lines. Clip_Mode 3 pre-clips patterned lines.

The hardware pre-clipping mechanism allows arbitrary lines to be drawn quickly and efficiently with little intervention by the host processor. Despite the fact that the ***mach32*** is only able to draw lines in the range (-512,-512) to (1535,1535), the pre-clipping mechanism allows almost all lines in the range (-32768,-32768) to (32767,32767) to be drawn without intervention by the host processor.

Clip_Mode 1 - Line Segments

Pre-clipping of line segments is performed using a modified version of the Cohen-Sutherland algorithm. The pre-clipping hardware examines each line to determine whether the drawing engine must actually draw the line, pixel by pixel, or whether the line needs to be drawn at all. As lines are drawn, each line segment is classified into one of three categories:

- Trivially Rejected
- Trivially Accepted
- Exception

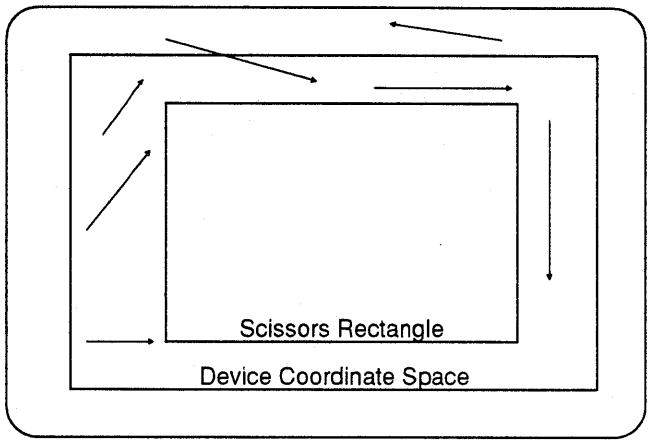


Figure 9-1. Trivially rejected lines.

1. **Trivially Rejected Lines** – are lines that definitely fall outside the scissor rectangle. Because they *do not pass through* the scissor rectangle they need not be drawn at all. When encountering a trivially rejected line, the graphics engine performs a "move" to the end coordinate of the line without attempting to draw any of the intervening pixels.

   Trivially rejected lines can be identified quickly using the following test:

   - Both endpoints of the line must be either above, below, to the left of, or to the right of the scissor rectangle (both left, both right, both above, or both below).

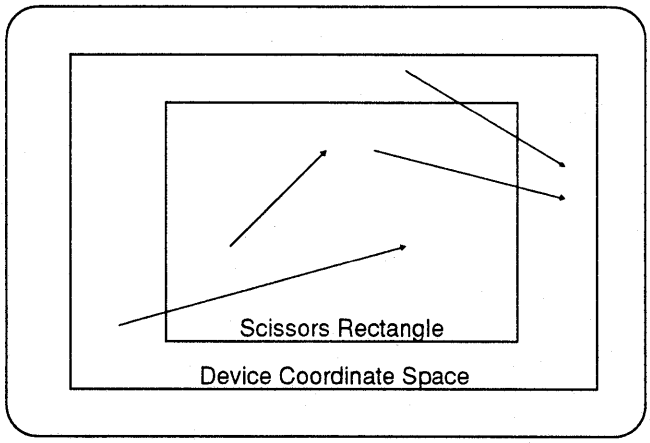


Figure 9-2. Trivially accepted lines.

2. Trivially Accepted Lines – are lines that lie entirely within the device coordinate space, and *probably* pass inside the scissor rectangle. These lines are drawn, pixel by pixel, by the graphics engine.

   A trivially accepted line must satisfy the following criteria:

   - Both endpoints of the line must lie within the device coordinate space
   - The line must not satisfy the criteria for Trivially Rejected lines (see previous section).

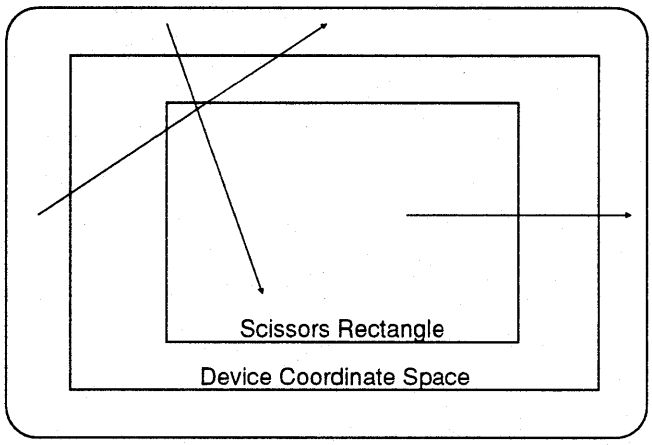


Figure 9-3. Lines generating exceptions.

3. Exception Lines – are lines that fall into neither of the two previous categories. Lines in this category may pass inside the scissor rectangle, but are not drawable by the graphics engine because one or both endpoints lie outside the device coordinate space.

   When the graphics engine encounters exception lines, two events occur: it generates a clip exception condition and stops drawing the current line as well as all subsequent lines until the clip exception condition has been cleared by the host processor. The graphics engine provides sufficient information to the host processor to allow the host to back up, identify, and process the line which generated the exception condition.

   In practice, very few lines are exception lines. In a typical CAD drawing, one or two lines out of many thousands will cause a clip exception and require further processing by the host. The majority of lines will be fully clipped and processed by the graphics engine without further intervention by the host. Since exception lines must span the scissor rectangle and the device coordinate rectangle, they must be at least 512 pixels long. For typical drawings, the average line length is significantly less than five pixels.

When a clip exception occurs, CLIP_OVERRUN field of EXT_GE_STATUS is set to logical one immediately. All subsequent attempts to write to the Line End Y through the Linedraw register are rejected by the engine, and each attempt to draw is recorded in the CLIP_OVERRUN counter. CLIP_OVERRUN is reset to zero when CLIP_MODE is set to zero, and when Y is written from LINEDRAW.

The CLIP_OVERRUN count may be used to identify the line segment which caused the exception condition. The host may then back up, endpoint-clip the offending line to device coordinate space, and continue with the rest of the transfer.

Clip_Mode 2 - Polygon Boundary Lines

The polygon boundary line pre-clipping algorithm is almost identical to the normal linedraw algorithm. It is only slightly modified to allow for correct pre-clipping of polygon boundary lines.

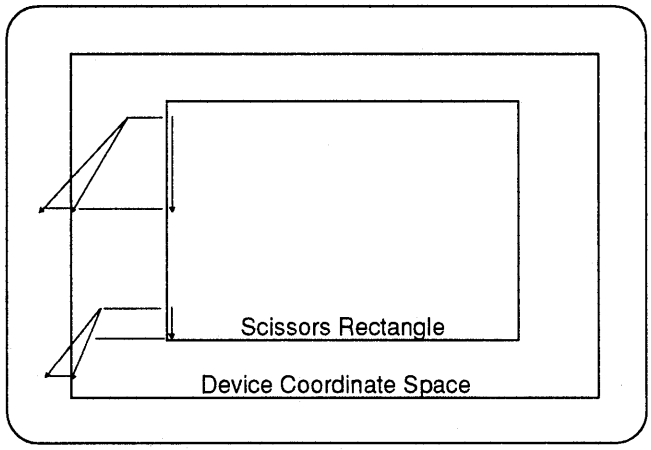


Figure 9-4. Polygon boundary lines.

1. Trivially Accepted Lines – Two polygon boundary lines are shown in figure 9-4. Both are to be drawn to the left of the scissor rectangle; but they should actually generate lines along the left edge of the scissor rectangle. A two-step procedure to draw these lines is as follows:
   1) The graphics engine clamps the endpoints to the edge of the device coordinate space;
   2) The normal linedraw procedure clamps the lines, pixel by pixel, to the left edge of the scissor rectangle.

   Lines that are outside the scissor rectangle are trivially accepted provided they are polygon boundary lines and their endpoints are neither above nor below the scissor.

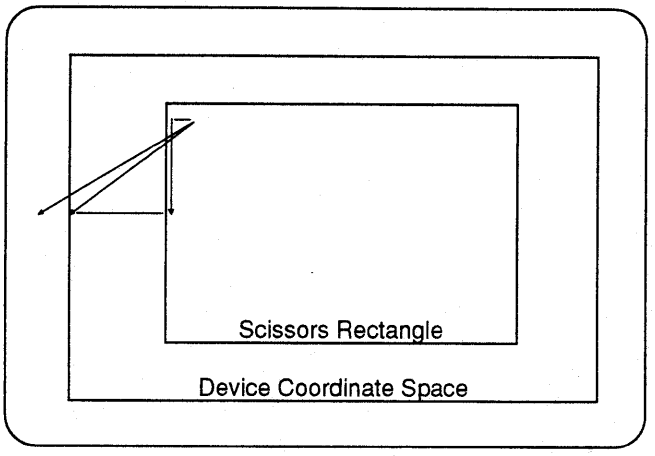


Figure 9-5. Polygon exception lines.

2. Exception Lines – Since endpoint clamping alters the angle of affected lines, lines for which one endpoint lies to the left of the device coordinate space, and the other to the right of the left edge of the scissor rectangle must generate exception conditions, for processing by the host processor.

   Although polygon line pre-clipping seems complicated, the mechanism resembles normal line pre-clipping — the engine automatically classifies each line segment as it is passed in. In both cases, the host processor writes line endpoints to the graphics processor, and checks for exception condition. If exception condition occurs, the host processor fully clips the line that generated the exception condition.

   The lines that generate exception conditions in the two clip modes discussed so far are of the same classification. The difference is only in the algorithm used. The Polygon Pre-clipping algorithm draws lines that would otherwise be trivially rejected by the Normal Linedraw algorithm.

Clip_Mode 3 - Patterned Lines

CLIP_MODE 3 supports pre-clipping of patterned lines. Patterned lines that are inside the device coordinate space are drawn. Patterned lines that are wholly or partially outside the device coordinate space generate exceptions.

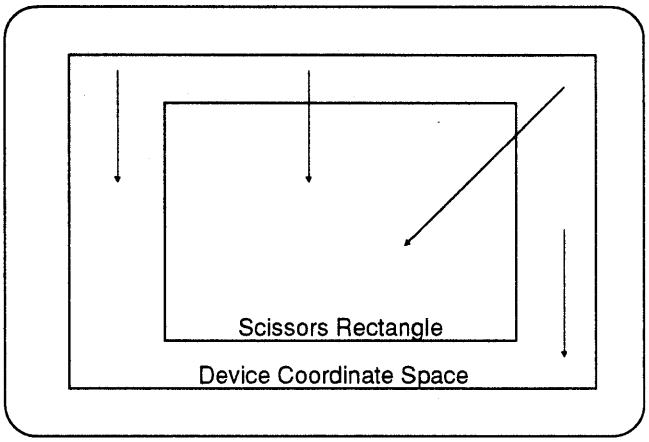


Figure 9-6. Patterned lines.

1. Patterned Lines – are lines entirely inside the device coordinate space, as shown in figure 9-6, is drawn by the graphics engine. They are classified as trivially accepted lines.

   The graphics engine attempts to draw all lines which would have been trivially rejected if the Normal Linedraw Pre-Clipping algorithm were used. It is faster to have the graphics engine draw a line and update the PATTERN_INDEX than to have the host processor reject a line and calculate a value for the PATTERN_INDEX.

2. Exception Lines– are lines wholly or partially outside the device coordinate space. All lines shown in figure 9-7 generate exception conditions. The host processor is required to fully clip these lines to the device coordinate space. The patterned linedraw algorithm then attempts to draw these lines and update the PATTERN_INDEX counter.

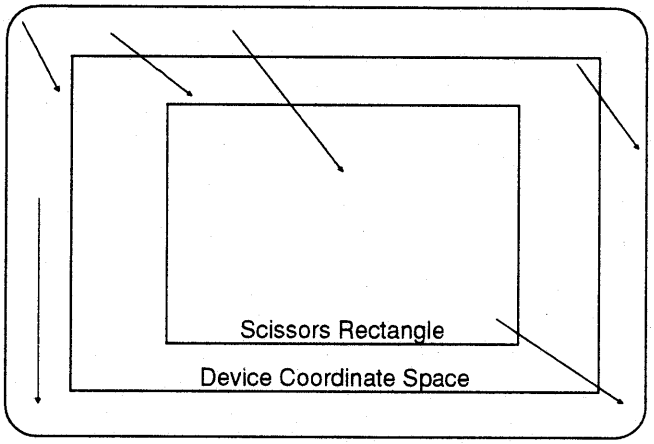


Figure 9-7. Exception generating patterned lines.

Using the Pre-Clipping Hardware

Although internal implementation of linedraw pre-clipping is quite complicated, the code required to use it is actually fairly simple. A code example for pre-clipping of lines in clip modes 1 and 2 is provided on the following page.

Once an offending line has been identified in an exception condition, it is clipped to the scissor (or device coordinate space) using the traditional Cohen-Sutherland algorithm. Since exception lines rarely occur, the time spent using the full clipping algorithm is minimal. In tests conducted by ATI, using standard industry benchmark AutoCAD drawings, the time spent executing the full clipping algorithm was not a measurable percentage of the total pan/zoom redrawing times.

```
void clipped_polyline (int npoints, int points[])
{ int n;
 outpw(linedraw_index,0);
 outpw(linedraw,points++);
 outpw(linedraw,points++);
 – npoints;

 while (npoints != 0)
 { /* block output line segments */
   n = min(8,npoints);
   npoints -= n;
   n *= 2;

   while(–npoints)    /* equivalent to rep outsw */
     outpw(linedraw,points++);

     /* check for overrun condition */
     n = inpw(overrun_count);
     npoints += (n +1)/2; /* back up to x coordinate */
     points -= (n+1)& (~1);

     /* check for clip exception */
     n = inpw(ext_ge_status) & 0x0f; /*clip_overrun count*/

     if (n != 0)
     { points -= n*2-2; /*back to segment causing exception*/
       draw_clipped_line(points[0], points[1], points[2],
         points[3]);
       points += 2;
       outpw(linedraw_index,0); /* reset current x and y */
       outpw(linedraw,points++);
       outpw(linedraw,points++);
       npoints += (n-1); /* correct number of points left */
     }
   }
}
```

Modes 1 and 2 Preclipping Code Sample

Extended Short-Stroke Vector Register

The Extended Short-Stroke Vector register accepts data in IBM 8514/A Short-Stroke Vector (SSV) format. This register operates in the same way as the IBM compatible Short-Stroke Vector register, except that monochrome data must be supplied in packed-bit format, and it supports color patterns. The IBM-compatible short-stroke vector register monochrome data is supplied in nibble format.

Short-stroke vectors are vectors of length less than 16, in pixels. Short-stroke vectors are used by the IBM AI for certain text fonts (including all the default mono-spaced fonts provided in the 8514 distribution kit.) Short-stroke text fonts are composed entirely of short-stroke vectors. Vector length is the projection of the line on the horizontal or vertical axis, whichever is greater.

This register ignores the contents of the data-width field, although the LSB_FIRST bit will be respected. All writes to this port must be 16-bit I/O operations. Two short-stroke vectors are drawn for each transfer (one may be a null vector). Short-stroke vectors also respect the current setting of LAST_PEL_OFF (Linedraw Options register). Note that short-stroke vectors are generally only useful in degree mode (DIR_TYPE=1).

Short-stroke vectors can also operate in Bresenham mode using pre-supplied Bresenham parameters. In this case, the direction field of each short-stroke vector overwrites the OCTANT field of the Linedraw Options register. Refer to the Direct Linedraw registers section for more details. Note that the DRAW bit overrides the DRAW bit previously set in DP_CONFIG.

Raw Bresenham Linedraw Registers

Raw Bresenham Linedraw Registers provide finer control over the actual linedraw operations than do the Direct Linedraw registers. Under certain circumstances it may be desirable to slightly alter the path of the linedraw.

Bresenham Error Term register is used to set up raw Bresenham linedraw operations. Valid range for the error term is (-4096..4095). If starting X is less than ending X, the Error Term register should be set to:

```
2*min(|dx|,|dy|)-max(|dx|,|dy|)-1
```

otherwise, the register is set to:

```
2*min(|dx|,|dy|)-max(|dx|,|dy|)
```

DESTY_AXSTP, DESTX_DIASTP registers determine the starting point of the Blit source pointer. Programmers should be aware that for 8514/A compatible drawing operations, the most significant five bits of Source Y, Source X registers are reserved (0).

When these registers are used to perform setup for raw Bresenham linedraw operations, before initiating the operation, both the Axial Step term and the Diagonal Step constant should be set as follows:

AXIAL STEP TERM (0 to 4095): `2*min(|dx|,|dy|)`

DIAGONAL STEP CONSTANT (-4096 to 0): `2*(min(|dx|,|dy|) - max(|dx|,|dy|))`

Under normal operation, the Diagonal Step constant is always zero or negative.

Bresenham Count register is used to initiate raw Bresenham linedraw operations. Valid range for the Bresenham count is (0...2047). The drawing control registers listed below must be correctly initialized before the Bresenham Count register is written:

CUR_X (86E8)
CUR_Y (82E8)
SRC_Y/DESTY_AXSTP (8AE8)
SRC_X/DESTX_DIASTP (8EE8)
ERR_TERM (92E8)
LINEDRAW_OPT (A2EE)

The value for the Bresenham Count register should be:

```
max(|dx|,|dy|)
```

Blit Registers

Blit is short for Bit Block Transfer. ATI Extended Blit operations support rectangle fills, polygon fills, and data transfers between the host processor and the ***mach32*** graphics engine. Unlike 8514/A-compatible blits, the exact operation of extended blits is determined by the configuration of the DP_CONFIG register.

If a blit region is selected as data source in the Datapath Configuration register, or if POLY_FILL_MODE is selected, blit source operations begin.

If the Pixel Transfer register is selected as data source, the ***mach32*** automatically waits for data to be transferred by the host processor.

Extended Blits allow more flexible specifications of the shapes of the source and destination areas — the shape and direction of the source area is independent of the shape and direction of the destination area. Extended blits also allow packed monochrome data to be passed directly to the drawing engine.

Source Data Pointer

The extent and direction of motion of the source data pointer are determined by the following blit parameter registers:

SRC_X (8EE8)
SRC_Y (8AE8)
SRC_X_START (B2EE)
SRC_X_END (BEEE)
SRC_Y_DIR (C2EE)

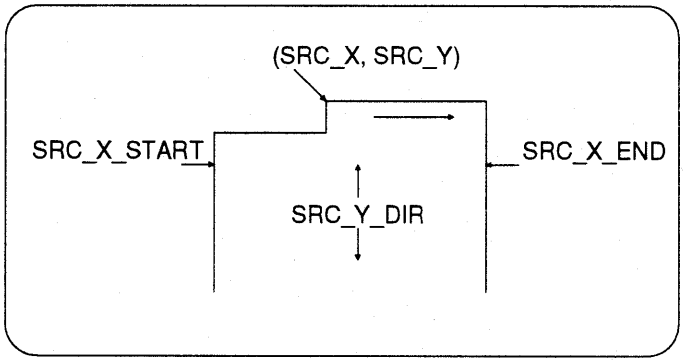


Figure 9-8. Source Data Pointer

If Blit Source X Row Start register is *equal* to Blit Source X Row End register at the time the Blit source is activated, the engine aborts the drawing routine immediately.

The graphics controller has an internal Blit-Source Data FIFO built-in. It is used to increase engine performance. It picks up 32 bytes of data at a time regardless of the actual length (size) of the data. Therefore, at the end of a blit operation, the contents of the Blit Source Current X and Blit Source Current Y registers will be *indeterminate*. However, all other Blit parameter registers will contain predictable values at that time (See Blit Source Data FIFO on page 9-41).

Destination Data Pointer

The extent and direction of motion of the destination data pointer are determined by the following blit parameter registers:

CUR_X (86E8)
CUR_Y (82E8)
DEST_X_START (A6EE)
DEST_X_END (AAEE)
DEST_Y_END (AEEE)

The Blit starts automatically as soon as the Destination Y End register is written.

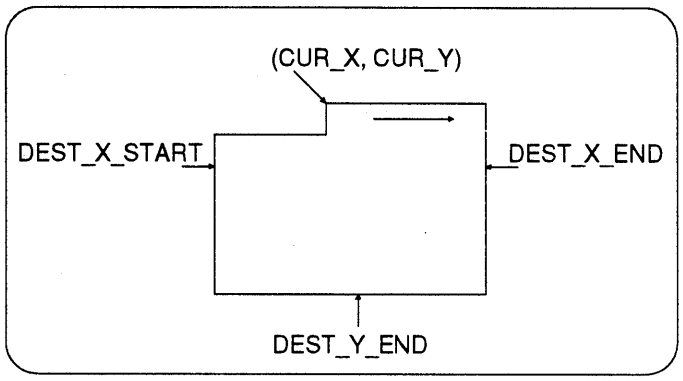


Figure 9-9. Destination Data Pointer

Blit Directions

The first row of the blit starts at Current_X, and ends at Destination_X_End. All subsequent rows start at Destination_X_Start, and end at Destination_X_End. In general, the left edge of the area is inclusive and the right edge is not.

X Direction

If Destination_X_Start is *less* than Destination_X_End, then the Blit is performed in left-to-right order (L-to-R). Current_X and Destination_X_Row_Start are inclusive. Destination_X_Row_End is exclusive.

If Destination_X_Start is *greater* than Destination_X_End, then the Blit is performed in R-to-L order. In this case, both Current_X and Destination_X_Row_Start are exclusive and Destination_X_End is inclusive.

If the starting X coordinate is *equal* to the ending X coordinate, then the drawing operation is indeterminate. Note that in the case of lines drawn with the SCAN_X register, lines for which CUR_X and SCAN_X are given the same value are indeterminate and must be tested.

The source data area behaves similarly.

Y Direction

If Destination_Y End is *less* than Current_Y, the Blit is performed in bottom-to-top order (B-to-T). Destination_Y_End is exclusive.

If Destination_Y_End is *greater* than Current_Y, the Blit is performed in T-to-B order. Destination_Y_End is exclusive.

If Destination_Y_End is *equal* to Current_Y, the drawing operation does not take place.

The source Y direction is determined by the SRC_Y_DIR register.

Blit Source

If the Blit source is named as color or monochrome source, or if Polygon Fill mode is enabled, the Blit source participates in the Blit operation. The extent and motion of the Blit source data pointer are determined by the Source Data Pointer registers.

BLIT Source Data FIFO

Advanced programmers who wish to exploit the Blit source data FIFO when performing blits to different destination areas from a single source area should be aware of the following characteristics of the source data FIFO:

- Blit Source operation commences when a graphics processor drawing operation first requests data from the source FIFO, and operates until it is halted.
- Blit Source operation halts, and the contents of the Blit-source data FIFO are flushed whenever any of the following actions occur:
  - Any of the Blit Source registers are updated.
  - The Datapath Configuration register is updated.
  - The engine is reset through SUBSYS_CNTL (42E8-W).
- The contents of the Blit-source data FIFO are destroyed whenever the color pattern registers are updated. Conversely, Color Pattern registers are destroyed when any draw operation is performed with the Blit source activated.

Host FIFO

N/A

Operations

Color Pattern Registers

In order to improve engine performance, color pattern registers are used to provide extra entries for the Blit Source Data FIFO whenever a blit operation is performed, unless these registers have been explicitly selected as a color source.

Data Transfers

Data is transferred from the host to the Pixel Transfer register if the register is designated as a color or monochrome source. During a blit operation, the engine waits for the host to supply the necessary data. After the blit is complete, any extra data written to the register overwrites the current value of the Background Color register.

When reading data from the Pixel Transfer register during blit operations, data is read from the destination trajectory. (Note that in 8514/A-compatible blit drawing operations, data is read from the source trajectory.) Data is read from each pixel in the drawing trajectory if the READ_WRITE field of the Datapath Configuration register is set to 0. If the READ_MODE field of the Datapath Configuration register is set to logical one, monochrome data is read from the drawing trajectory. Data is packed 8 bits per byte and transferred from the screen as follows:

1. During a left-to-right (L-to-R) blit operation, the MSB screen bit is transferred to the register first.

2. During a right-to-left (R-to-L) blit operation, the LSB screen bit is transferred first. In other words, the left-most screen bit will always be read into the MSB bit of the Pixel Transfer register regardless of the direction of blit transfer.

Monochrome data is expected to be in most-significant-bit-left order. In 16-bit monochrome data transfers, data is expected to be in least-significant-byte-left order. Although byte ordering is independently selected by the LSB_FIRST field of the Datapath Configuration register, generally, LSB_FIRST should be set to 1 for L-to-R Blits and set to 0 for R-to-L Blits.

8- or 16-bit transfers are determined by the DATA_WIDTH field of the Datapath Configuration register. Byte ordering is selected by the LSB_FIRST field of the same register. For example, in 8-bit transfers, the LSB_FIRST bit should be set to 0 (msb-first), the (16-bit) DATA_WIDTH field should be set to 0 (8-bit), and I/O should be performed to address 0E2E9h (PIX_TRANS+1). This behavior is supported only for ATI extended drawing operations (IBM drawing operations ignore the LSB_FIRST field when operating in 8-bit mode).

Examples:

1. When a 16-bit monochrome read is performed in R-to-L order, pixels will be read sequentially off the screen and transferred to the host in this bit order: D8 to D15, D0 to D7.

2. If a 16-bit monochrome write is performed in L-to-R order, pixels will be transferred from the host to the video buffer in this order: D7 to D0, D15 to D8.

Monochrome host data is considered *row-packed.* This means the last bit of a row of data is followed immediately by the first bit of the next row of data, regardless of byte boundaries in the host data stream.

Data is considered row-packed in a manner consistent with the direction of transfer. Thus for R-to-L monochrome transfers, the first bit of a row of data will be followed immediately by the last bit of the next row of data. This allows several lines of data to be transferred by the following code regardless of byte alignment:

```
MOV CX, WIDTH * HEIGHT      ; Length of transfer
REP OUTSB                   ; Perform block transfer
```

Bitmap data in byte-aligned row format may be transferred to memory directly, by shifting the right edge of the destination area, and scissoring extraneous bits.

If the Blit uses either the color or monochrome pattern as a source, the Pattern Index register will be updated as follows:

- Post-incremented if the Blit is performed left-to-right;
- Pre-decremented if the Blit is performed right-to-left.

In order to facilitate pattern fills, the Pattern Index is reset to the previous value whenever Current X becomes equal to Blit X End, if the blit is dependent on either the monochrome or color pattern registers.

Sample codes for Blit Destination and Blit Source Addressing are provided on the following pages.

```
      /* If the Blit mode is Conforming
      then calculate starting parameters*/
IF PatternDependent() THEN
  StartPatternIndex := PatternIndex
WHILE (RowStart != RowEnd) AND (CurrentY != Y_End)
DO BEGIN
  IF (OCTANT.XDIR == 0) THEN BEGIN
      /* Transfer and post-increment */
    VRAM[CurrentY,CurrentX] = ALUresult();
    CurrentX += 1;
    IF (PATTERNDEPENDENT)
      ++ PatternIndex;
  END;
  ELSE BEGIN
      /* Pre-decrement and then transfer*/
    CurrentX -= 1;
    IF (PATTERNDEPENDENT)
      - PatternIndex;
    VRAM[CurrentY,CurrentX] = ALUresult();
  END;
  IF (CurrentX == RowEnd) THEN BEGIN
    IF PatternDependent() THEN
      PatternIndex := StartPatternIndex;
    CurrentX = RowStart;
    IF (OCTANT.YDIR = 0)
      CurrentY += 1;
    ELSE
      CurrentY -= 1;
  END;
END. {While}
```

Blit Destination Addressing

```
SrcXDir := Sign(SourceXRowEnd-SourceXRowStart);
WHILE (SourceDataRequired()) DO BEGIN
  IF (ScrXdir == 1) THEN BEGIN
      /* Left to right...*/
    PassPixel(GetPixel(CurrentSrcX,CurrentSrcY));
    CurrentSrcX += 1;          /* Post increment */
  END;
  ELSE BEGIN
    CurrentSrcX -= 1;          /* Pre-decrement */
    PassPixel(GetPixel(CurrentSrcX,CurrentSrcY));
  END;
  IF (CurrentSrcX == SourceXRowEnd) THEN BEGIN
    CurrentSrcX := SourceXRowStart;
    IF (SrcYDir == 1) THEN BEGIN
      CurrentSrcY += 1;
  END;
  ELSE
    CurrentSrcY -= 1;
END. {While}
```

Blit Source Addressing

Source Blit Parameter Registers are modified during Blit/Fill operations only if the Blit source is named as a monochrome source, a color source, or if the Polygon-Fill mode is enabled.

The Blit Source also operates if Blit Source is selected as a data source in the Datapath Configuration register, or if the Polygon Fill mode is selected when performing drawing operations other than Blit operations, such as linedraws, short-stroke vectors, or Scan-to-X operations. Data is read from the Blit Source area for these operations as for regular blits as soon as the drawing operation is initiated.

Miscellaneous Drawing Commands

As well as supporting linedraws and blits, the ***mach32*** graphics processor provides several additional drawing commands that are generally useful for implementing graphics library primitives.

Pattern Registers

The graphics engine supports a 32-pixel color pattern register block, a 32-bit monochrome pattern register block, and an 8x8x1 monochrome pattern extension. (The 8x8x1 mono pattern extension is only available in ATI68800-6 or higher controllers. See page 7-13 for description.) Each of these sets of registers may be selected as data source. These pattern registers are accessed via a data register and an index register.

Linear vs Rectangular

The graphic engine uses the 32 bit pixel color pattern register block as a variable length linear color pattern source, which is tiled into the destination. The monochrome pattern register block normally behaves in a similar fashion; but on an ATI68800-6 or higher controller, the monochrome pattern register block may also be configured as an 8x8 rectangular pattern which is tiled aligned to the destination.

Pixel Transfer Control Registers

The Pixel Transfer ALU is used to combine each pixel on a drawing trajectory with one or more sources of data. Pixel transfer control registers collectively define how destination pixels are modified during draw operations.

Pixel Transfer Function Registers

Two registers, Foreground ALU Function and Background ALU Function, are provided for in the engine. The Write Mask register affects all arithmetic/Boolean operations performed by the pixel transfer ALU. Planes which are masked out by the Write Mask register do not participate in any ALU operation. Effectively, all unmasked planes are treated as if they were contiguous. During add and subtract operations, carries or borrows entering into a bit position which is masked by the plane mask are passed through, unmodified, to the next most significant bit, except over R/G/B boundaries in 16bpp modes.

SUBTRACT — The exact implementation is equivalent to:

```
Result := (X AND PlaneMask)-(Y AND PlaneMask)
```

ADD — The exact implementation is equivalent to:

```
Result := ((X AND PlaneMask) + (Y OR (NOT PlaneMask))) XOR (NOT PlaneMask)
```

DIVIDE-BY-TWO — is performed by shifting the result of the previous operation (ADD or SUBTRACT, Saturated or Unsaturated) right by one bit. The carry or borrow from the preceding operation is shifted into the most significant bit of the result.

MAX and MIN — use unsigned comparisons. The plane mask affects the comparison between source and destination data. The decision on whether the source data or destination data is passed as a result is based on whether a SUBTRACT(D,S) operation would produce a borrow from the most significant bit position.

Pixel Transfer Registers

The ***mach32*** graphics controller supports many different formats of data transfers to and from the graphics engine. Each of the following described registers is used to transfer a stream of image data to or from the graphics engine. It is the responsibility the host application to ensure that the appropriate Pixel Transfer register is selected as a color or monochrome source.

Background Function Register				
B6EE (W)	ALU_BG_FN			
	-	-	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
4:0	ALU_BG_FN	Background ALU function code. Code definitions are listed on page 8-24 of this manual.		
15:5	-	Reserved		

Foreground Function Register				
BAEE (W)	ALU_FG_FN			
	-	-	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
4:0	ALU_FG_FN	Foreground ALU function code. Code definitions are listed on page 8-24 of this manual.		
15:5	-	Reserved		

Bounds Accumulator Bottom Register				
7EEE (R)	BOUNDS_BOTTTOM			
	-	-	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
15:0	BOUNDS_BOTTOM	The bottom edge of the bounding box containing the points written through the linedraw register. Reset to -2048 when LINEDRAW_OPT[8] (A2EE) is one.		

Bounds Accumulator Left Register				
72EE (R)	BOUNDS_LEFT			
	-	-	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
15:0	BOUNDS_LEFT	The left edge of the bounding box containing the points written through the linedraw register. Reset to 2047 when LINEDRAW_OPT[8] (A2EE) is one.		

Bounds Accumulator Right Register				
7AEE (R)	BOUNDS_RIGHT			
	-	-	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
15:0	BOUNDS_RIGHT	The right edge of the bounding box containing the points written through the linedraw register. Reset to -2048 when LINEDRAW_OPT[8] (A2EE) is one.		

Bounds Accumulator Top Register				
76EE (R)	BOUNDS_TOP			
	-	-	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
15:0	BOUNDS_TOP	The top edge of the bounding box containing the points written through the linedraw register. Reset to 2047 when LINEDRAW_OPT[8] (A2EE) is one.		

Bresenham Count Register				
96EE (R/W)	BRES_COUNT			
	-	-	Mach 8	Mach 32

Bit	Mnemonic	Description
10:0	COUNT	The Bresenham count is MAX (|dx|,|dy|); it should be within the range of (0 ... 2047).
15:11	-	Reserved

Register Description:

1. The Bresenham Count register is used to initiate raw Bresenham linedraw operations. Before writing this register, the following drawing control registers must be correctly initialized:

   CUR_X (086E)
   CUR_Y (082E8)
   SRC_Y/DESTY_AXSTP (8AE8)
   SRC_X/DESTX_DIASTP (8EE8)
   ERR_TERM (92E8)
   LINEDRAW_OPT (A2EE)

2. The contents of MAJ_AXIS_PCNT (96E8) can be read through this register.

Destination Color Compare Function Registers				
EEEE (W)	DEST_CMP_FN			
	-	-	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
2:0	-	Reserved		
5:3	DEST_CMP_FN_4/8	4/8-bit mode destination compare function code[1]		
8:6	DEST_CMP_FN_B	16-bit blue destination compare function code[1]		
11:9	DEST_CMP_FN_G	16-bit green destination compare function code[1]		
14:12	DEST_CMP_FN_R	16-bit red destination compare function code[1]		
15	-	Reserved		

Register Description:

1. The codes in the DEST_CMP_FN fields of this register are described as follows:

Code	Function Descriptions
0h	False
1h	True
2h	Destination Pixel >= DEST_CMP_CLR
3h	Destination Pixel < DEST_CMP_CLR
4h	Destination Pixel != DEST_CMP_CLR
5h	Destination Pixel = DEST_CMP_CLR
6h	Destination Pixel <= DEST_CMP_CLR
7h	Destination Pixel > DEST_CMP_CLR

2. The pixel data in the comparison are unsigned. The result of the comparison according to the selected relation is as follows:

   If TRUE: Destination pixel unchanged
   If FALSE: Destination pixel written

3. In 4- and 8-bit pixel modes the compare function is applied to each destination pixel.

4. Only planes that have been enabled by the Write Mask participate in the comparison. The separate Red, Green, and Blue compare functions are applied to the Red, Green, and Blue color fields of the destination pixel. If all three relations hold, the destination pixel is not over-written.

Destination Color Compare Function Registers				
F2EE (R/W)	DEST_COLOR_CMP_MASK			
	-	-	-	Mach 32

Bit	Mnemonic	Description
15:0	-	Destination Color Compare Mask

Register Description:

1. This register is applicable only to ATI68800-6 ***mach32*** controllers. (Note: ATI68800LX and ATI68800AX controllers have compatible registers as the ATI68800-6.)

Destination X End Register				
AAEE (W)	DEST_X_END			
	-	-	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
10:0	DEST_X_END	The last X coordinate of each row of the destination blit		
15:11	-	Reserved		

Destination Start X Register				
A6EE (W)	DEST_X_START			
	-	-	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
10:0	DEST_X_START	The starting X coordinate for the second and subsequent rows of the destination blit. Valid range is (-512 ... 1535)		
15:11	-	Reserved		

Destination Y End Register				
AEEE (W)	DEST_Y_END			
	-	-	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
10:0	DEST_Y_END	The last line of the destination blit for VRAM-to-VRAM, rectangle file, or polyfill operation		
15:11	-	Reserved		

Register Description:

1. The blit source is enabled if one of the following is true:

   - Color source is set to VRAM blit source
   - Monochrome source is set to VRAM blit source
   - DP_CONFIG[POLY_FILL_MODE] (CEEE) enabled

Extended Bottom Scissor Register				
E6EE (W)	EXT_SCISSOR_B			
	-	-	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
11:0	EXT_SCISSOR_B	Coordinate of extended mode bottom scissor, has a range of (-2048...2047).		
15:12	-	Reserved		

Register Description:

1. Extended mode scissor registers operate in the range of (-2048...2047). Compare with 8514/A compatible bottom scissor range of (0...2047).

Extended Left Scissor Register				
DAEE (W)	EXT_SCISSOR_L			
	-	-	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
11:0	EXT_SCISSOR_L	Coordinate of extended mode left scissor, has a range of (-2048...2047).		
15:12	-	Reserved		

Register Description:

1. Extended mode scissor registers operate in the range of (-2048...2047). Compare with 8514/A compatible left scissor range of (-1024...1023).

Extended Right Scissor Register				
E2EE (W)	EXT_SCISSOR_R			
	-	-	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
11:0	EXT_SCISSOR_R	Coordinate of extended mode right scissor, has a range of (-2048...2047).		
15:12	-	Reserved		

Register Description:

1. Extended mode scissor registers operate in the range of (-2048...2047). Compare with 8514/A compatible right scissor range of (0...2047).

Extended Top Scissor Register				
DEEE (W)	EXT_SCISSOR_T			
	-	-	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
11:0	EXT_SCISSOR_T	Coordinate of extended mode top scissor, has a range of (-2048...2047).		
15:12	-	Reserved		

Register Description:

1. Extended mode scissor registers operate in the range of (-2048...2047). Compare with 8514/A compatible top scissor range of (-1024...1023).

Extended Short Stroke Vector Transfer Register				
C6EE (W)	EXT_SHORT_STROKE			
	-	-	Mach 8	Mach 32

Bit	Mnemonic	Description
7:0	SHORT_STROKE_2	A short stroke vector (SSV).
15:8	SHORT_STROKE_1	A short stroke vector.

Register Description:

1. The Extended Short-Stroke Vector Transfer register accepts two bytes concurrently in IBM-compatible SSV format. Both bytes are always used regardless of data transfer size. The format is:

D	D	D	O	L	L	L	L

D — Direction of Vector
O — Operation Type:
0 = Move
1 = Draw
L — Length of the projection of the vector on the horizontal or vertical axis, whichever is greater, in pixels

2. Color patterns are supported; monochrome data must be in packed-bit format.
3. The operation of this register is similar to the Short-Stroke Vector Transfer register (9EE8). I/O operations are 16-bit.
4. Null vectors are valid extended SSVs.

Linedraw Register				
FEEE (W)	LINEDRAW			
	-	-	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
15:0	LINEDRAW	Value is determined by the Linedraw Index register as follows: Set Current X Set Current Y Set Line End X Set Line End Y and draw/move operation		

Register Description:

1. All references to the Y coordinates through this register are linear, regardless of the value of MEM_CNTL (B8EE, Index5).

Linedraw Index Register				
9AEE (W)	LINEDRAW_INDEX			
	-	-	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
2:0	LINEDRAW_INDEX	**0** = Set current X **1** = Set current Y **2** = Set line end X **3** = Set line end Y, draw line, reset index to 2 **4** = Set current X **5** = Set current Y, reset index to 4		
15:3	-	Reserved		

Register Description:

1. The linedraw index specifies the operation to be performed by the Direct Linedraw Data register. Continuous lines can be written in two steps as follows:

   a. Setting this register:

   ```
   mov   dx, linedraw_index
   mov   ax, 0              ;Start with a move
   out   dx, ax
   ```

   b. Block-copying moveto/lineto/lineto commands to the graphics engine:

   ```
   mov   dx, linedraw
   mov   cx, points*2
   lea   si, [pointbuffer]
   cld
   rep   outsw
   ```

2. The bounding box of a series of points may be determined by using a "moveto" command on each point in the buffer, by setting the index to 4. The GE will test each point against the current content of the Bounds Accumulator register and enlarge the bounds accumulator rectangle to enclose all points.

Color Pattern Data Registers				
8EEE (W)	PATT_DATA_*[1]			
	-	-	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
15:0	-	Color pattern data bits		

Register Description:

1. *The 16 Color Pattern Data registers are numbered 0h to Fh: PATT_DATA_0 to PATT_DATA_F

Monochrome Pattern Data Registers				
8EEE (W)	PATT_DATA_*[1]			
	-	-	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
15:0	-	Monochrome pattern data bits		

Register Description:

1. *The eight (8) Monochrome Pattern Data registers are numbered 10h to 17h, i.e., PATT_DATA_10 to PATT_DATA_17. These registers contain up to 32 bits of monochrome patterns to be used as the monochrome source during pixel transfers. PATT_INDEX and PATT_LENGTH will determine the first pixel and the number of pixels respectively to participate in the mix. The 32-bit linear monochrome pattern (which only has uses for mono pattern data registers) are not destination-aligned.
2. On **ATI68800-6** or later controllers, it is possible to configure the monochrome data contained in PATT_DATA as an 8x8 rectangular destination-aligned region, to be tiled into the destination. See PATT_LENGTH (D2EE-W) and *Linear vs Rectangular* description on pages 9-46 and 9-60.
3. The 16 Color Pattern Data registers are numbered from 0h to Fh. Together they may contain up to 32 bytes of color information for a 32-byte color pattern. When the registers participate in a pixel transfer, as color source, determined by DP_CONF[FG_COLOR_SRC] (CEEE), the two PATT_INDEX (D6EE) and PATT_LENGTH (D2EE) registers will indicate the first pixel and the number of pixels, respectively.
4. If color pattern is not selected as a source, these registers are used as scratch area for the blit-source data FIFO during draw operations.

Pattern Data Index Register				
82EE (R/W)	PATT_DATA_INDEX			
	-	-	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
7:0	PATT_DATA_INDEX	**ATI68800-3**, using bits 7:0: A pointer to one of the 18 internal registers of the color pattern register, 16 color pattern registers and 2 monochrome pattern registers. **ATI68800-6**, using bits 4:0: A pointer to one of the 24 internal registers of the color pattern register, 16 color pattern registers and 8 monochrome pattern registers. Bits 7:5 reserved.		
15:8	-	Reserved		

Register Description:

1. The Pattern Data Index register is incremented after each write.

Pattern Index Register				
D6EE (W)	PATT_INDEX			
	-	-	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
4:0	PATT_INDEX	Determines the next pixel in the color or monochrome pattern for the destination.		
15:5	-	Reserved		

Pattern Length Register				
D2EE (W)	PATT_LENGTH			
	-	-	Mach 8	Mach 32

Bit	Mnemonic	Description
4:0	PATT_LENGTH	Determines the length of color or monochrome pattern. Value is actual length less 1.
6:5		Reserved
7		8x8 Mono Pattern Enable
14:8	-	Reserved
15		8x8 Block Write Mono Pattern Enable

Register Description:

1. If bits 7 and 15 are both "0" —
   This register determines the length of the color or monochrome pattern to be repeated linearly into the destination.
2. If bit 7 is "1" —
   Bits 4:0 are ignored. The pattern data contained in PATT_DATA_[10:17] will be used as an 8x8 destination-aligned rectangular monochrome pattern to be tiled onto the destination.
3. If bit 7 and 15 are both "1" —
   Bits 4:0 are ignored. An 8x8 pattern will be used as described in step 2. This monochrome pattern will be used as a transparency mask for the block write capable memory. During a block write operation, the ALU foreground function (BAEE-W) should be set to "Paint (7)". The ALU background function (B6EE-W) should be set to "Leave Alone (3)".
4. Bit 15 must not be set unless Block Write Enable, MISC_OPTIONS[10] (bit 400h of 36EE), is also set.

Read Source X Register				
DAEE (R)	R_SRC_X			
	-	-	-	Mach 32

Bit	Mnemonic	Description
10:0	R_SRC_X	Source current X
15:11	-	Reserved

Register Description:

1. DAEE-R is the read address of SRC_X which is at 8EE8-W.

Read Source Y Register				
DEEE (R)	R_SRC_Y			
	-	-	-	Mach 32

Bit	Mnemonic	Description
10:0	R_SRC_Y	Source current Y
15:11	-	Reserved

Register Description:

1. DEEE-R is the read address of SRC_Y which is at 8AE8-W.

Scan To X Register				
CAEE (W)	SCAN_X			
	-	-	Mach 8	Mach 32

Bit	Mnemonic	Description
10:0	SCAN_TO_X	Scan line draw to X data
15:11	-	Reserved

Register Description:

1. The Scan To X register may be used by the host processor to perform fast polygon scan conversions.
   a. The host sets CUR_X and CUR_Y with the address of the first point in the active edge table and sets SCAN_TO_X with the address of the destination. At this time, the SCAN_TO_X_STATE internal fill flag is cleared.
   b. The draw engine traverses each scan line from left to right (if the address in CUR_X> SCAN_TO_X) or right to left (if the address in CUR_X <SCAN_TO_X). In either case, the left edge is inclusive and the right edge is exclusive.
   c. After each operation, the fill flag is toggled. When the flag is cleared, scan conversion takes place as specified by the mix in the ALU. When the flag is set, the operation is setting CUR_X and does not draw.
2. The Scan To X register can also be used to draw fast horizontal lines.
   a. The host sets CUR_X and CUR_Y.
   b. Any blit control must be set as the line will use the ATI blit engine.
   c. The host writes SCAN_TO_X.
   d. The left edge will always be inclusive and the right exclusive regardless of direction.
   e. CAUTION: Do not optimize out the writing of CUR_X. See step 1.

Source X End Register				
BEEE (W)	**SRC_X_END**			
	-	**-**	**Mach 8**	**Mach 32**
Bit	**Mnemonic**	**Description**		
10:0	SRC_X_END	The last pixel of the blit source data.		
15:11	-	Reserved		

Source X Start Register				
B2EE (W)	**SRC_X_START**			
	-	**-**	**Mach 8**	**Mach 32**
Bit	**Mnemonic**	**Description**		
10:0	SRC_X_START	The X coordinate of the first pixel of the second and subsequent rows of the blit source data.		
15:11	-	Reserved		

Source Y Direction Register				
C2EE (W)	**SRC_Y_DIR**			
	-	**-**	**Mach 8**	**Mach 32**
Bit	**Mnemonic**	**Description**		
0	SRC_Y_DIR	The direction of the drawing operation: **0** = Bottom-to-top **1** = Top-to-bottom.		
15:1	-	Reserved		

Status Registers

The ***mach32*** graphics controller contains a set of registers that indicate the general status of the engine as follows:

- Extended FIFO Status
- FIFO Status
- FIFO Control
- Subsystem Status
- Extended Graphics Engine Status
- Configuration Status 1
- Configuration Status 2

Configuration Status 1 Register				
12EE (R)	CONFIG_STATUS_1			
	-	-	Mach 8	-

Bit	Mnemonic	Description
0	CLK_MODE	**0** = Use crystals only **1** = Use clock chip
1	BUS_16	**0** = 8-bit bus configured **1** = 16-bit bus configured
2	MC_BUS	**0** = ISA/EISA bus configured **1** = MicroChannel bus configured
3	EEPROM_ENA	Default parameters are loaded from EEPROM after power on reset. **0** = EEPROM disabled **1** = EEPROM Enabled
4	DRAM_ENA	**0** = VRAM installed **1** = DRAM installed
6:5	MEM_INSTALLED	**0** = 512K memory **1** = 1024K memory **2,3** = Reserved
7	ROM_ENA	**0** = Disabled **1** = Enabled
8	ROM_PAGE_ENA	**0** = Disabled **1** = Enabled
15:9	ROM_LOCATION	Page 0 of the boot ROM will be enabled after power on reset, at C0000h + 800h*n, where n is the value of this field, if the following conditions are true: EEPROM_ENA = 0 MC_BUS = 0

Configuration Status 1 Register					
12EE (R)	CONFIG_STATUS_1				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
0	8514_ONLY	**0** = VGA+8514 function enabled **1** = VGA function disabled			
3:1	BUS_TYPE	**0** = 16-Bit ISA **1** = EISA **2** = 16-Bit MicroChannel (µC) **3** = 32-Bit MicroChannel (µC) **4** = LBus_SX; 386SX specific **5** = LBus_1, LBus_2; 386DX specific **6** = LBus_1, LBus_2; 486 specific **7** = PCI			
6:4	MEM_TYPE	**ATI68800-3:** **0** = 256Kx4 DRAM memory **1** = 256Kx4 VRAM memory; 512-bit serial transfer **2** = 256Kx4 VRAM memory; 256-bit serial transfer **3** = 256Kx16 DRAM memory **ATI68800-6:** **0** = 256Kx4 DRAM memory **1** = 256Kx4 VRAM memory; 512-bit serial transfer **2** = 256Kx16 VRAM memory; 256-bit serial transfer **3** = 256Kx16 DRAM memory **4** = 256Kx4 Graphics DRAM memory **5** = 256Kx4 VRAM memory; 512-bit split transfer **6** = 256Kx16 VRAM memory; 256-bit split transfer			
7	CHIP_DIS	**0** = Chip enabled **1** = Chip disabled			
8	TST_VCTR_ENA	**0** = Normal operation **1** = Delay memory write data by 1/2 MCLK for test vector generation			
11:9	DACTYPE (See Appendix F for a detailed listing of supported DACs.)	**0** = ATI68830 **1** = SC11483 **2** = ATI68875 **3** = Bt476 **4** = Bt481 **5** = ATI68860 (requires ATI68800AX or higher)			
12	MC_ADR_DECODE	**0** = Enable external µC address decode **1** = Enable internal µC address decode			
15:13	CARD_ID	Indicator of controller ID, used for multiple controller systems. **0** = Multiple controllers not supported.			

Configuration Status 2 Register					
16EE (R)	CONFIG_STATUS_2				
	-	-	Mach 8	-	
Bit	**Mnemonic**	**Description**			
0	SHARE_CLOCK	**0** = ***mach8*** does not share clock with VGA **1** = ***mach8*** shares clock with VGA			
1	HIRES_BOOT	This bit is for the benefit of the boot ROMs which must know whether to enforce the higher resolutions but may not be using the monitor ID: **0** = Not hi-res **1** = Hi-res			
2	EPROM_16_ENA	**0** = Adapter not configured for 16-bit boot ROM **1** = Adapter configured for 16-bit boot ROM			
3	WRITE_PER_BIT	For VRAM configuration supporting internal plane masks: **0** = Faster write-masked operation not supported **1** = Faster write-masked operation supported			
4	FLASH_ENA	**0** = Flash page writes not supported **1** = Flash page writes supported			
15:5	-	Reserved			

Configuration Status 2 Register					
16EE (R)	CONFIG_STATUS_2				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
0	SLOW_SEQ_EN	**ATI68800-3:** **0** = Enable 1 clock sequencer timing (Default) **1** = Enable 2 clock sequencer timing **ATI68800-6:** Reserved			
1	MEM_ADDR_DIS	**1** = Disable mem address range FE0000:FFFFFF			
2	ISA_16_ENA	ISA only (Bit ignored in other configurations): **0** = 8-Bit ISA bus enabled **1** = 16-Bit ISA bus enabled			
3	KOR_TXT_MODE_ ENA	**0** = Korean character font support disabled **1** = Korean character font support enabled			
5:4	LOCAL_BUS_ SUPPORT	**0** = Reserved **1** = LOCAL2# local bus signal selected **2** = LOCAL3# local bus signal selected **3** = LOCAL1# local bus signal selected			
6	LOCAL_BUS_ CONFIG_2	**0** = LBus_1 configuration (multiplexed) **1** = LBus_2 configuration (non-multiplexed)			
7	LOCAL_BUS_ RD_DLY_ENA	**1** = Read data to be held for 1 bus clock after RDY			
8	LOCAL_DAC_EN	**1** = Disable local decode of RAMDAC write in a local bus configuration			
9	LOCAL_RDY_EN	**0** = Enable 1 bus clock RDY delay for write **1** = Disable 1 bus clock RDY delay for write			
10	EEPROM_ADR_SEL	**0** = BIOS EEPROM decode at E000:0 - E7FF:F **1** = BIOS EEPROM decode at C000:0 - C7FF:F			
11	GE_STRAP_SEL	**EISA Bus:** **0** = Always enable chip in lieu of POS register **1** = Enable POS register function **Local Bus:** **0** = Enable local decode of 102h register **1** = Disable local decode of 102h register			
12	VESA_RDY	**0** = Enable VESA compliant RDY format			
13	Z4GBYTE	**ATI68800-3:** Reserved **ATI68800-6:** **0** = Enable 128MB memory aperture **1** = Enable 4GB memory aperture			

Configuration Status 2 Register					
16EE (R)	CONFIG_STATUS_2				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
14	LOC2_MDRAM	**ATI68800-3:** Reserved **ATI68800-6:** **0** = Support 1MB DRAM in LBus_2 configuration **1** = Support 2MB DRAM in LBus_2 configuration			
15	-	Reserved			

Register Description:

1. This register is an indicator of the controller strap configuration.

Extended Graphics Engine Status Register					
62EE (R)	EXT_GE_STATUS				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
3:0	CLIP_OVERRUN	Clip Overrun			
7:4	-	Reserved			
8	CLIP_INSIDE	Clip Inside			
12:9	CLIP_FLAGS	For each bit definition: **0** = Outside scissor **1** = Inside scissor			
13	GE_ACTIVE	**0** = Graphics engine is idle **1** = Graphics engine is busy			
14	EE_DATA_IN	EE Data In			
15	POINTS_OUTSIDE	Points Outside			
15:13	-	Reserved			

Vertical Line Counter Register					
CEEE (R)	VERT_LINE_CNTR				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
10:0	VERT_LINE_CNTR	Vertical line counter			
15:11	-	Reserved			

Overscan

The overscan registers are used for defining a color in the overscan area of the screen. Since high refresh video modes tend to have very small overscan areas, CRT parameters should be recomputed using higher pixel clocks.

The overscan feature allows a rectangular border region around the active display area to be displayed in the overscan color. Normally this region would be blanked or set to black. The width of the left and right border regions is defined in terms of characters with a range of 0 to 15 characters. The height of the top and bottom border regions is defined in terms of lines with a range of 0 to 255 lines. The horizontal sync may be adjusted on a pixel basis to improve horizontal alignment. By setting all overscan widths, heights, and alignments to zero, the overscan feature is effectively disabled.

All overscan registers are included in the 640x480 and 1024x768 resolution shadow register sets. An overscan lock bit, SHADOW_CTL[6] (46EE), chooses between one of these shadow register sets and the primary register set. This capability provides transparent overscan support in the basic 1024x768 8514/A resolutions. Overscan feature in non-standard resolutions must be supported by an application display driver.

Horizontal Overscan Register					
62EE (W)	HORZ_OVERSCAN				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
3:0	-	Overscan width - left side			
7:4	-	Overscan width - right side			
10:8	H_SYNC_DELAY	Horizontal sync delay			
11	-	Reserved			
12	OSCAN_INV	**1** = Inverts Overscan Polarity			
13	SYN_CONT_SEL	Uses bits 15:14 of this register as the VSYNC and HSYNC signal outputs for use in Monitor Power Down mode			
14	HSYN_CONT	General purpose bit. HSYNC when bit 13 = 1			
15	VSYN_CONT	General purpose bit. VSYNC when bit 13 = 1			

Register Description:

1. On the ATI68800-3 ***mach32***, the upper byte (15:8) of this register is reserved
2. The Horizontal Overscan Control register defines the left and right overscan widths in character units. It also defines a delay value in pixel units for horizontal sync to allow exact centering of the overscan screen. In 1280x1024 multiplexed pixel modes, the adjustment is in steps of two pixels.

Vertical Overscan Register					
66EE (W)	VERT_OVERSCAN				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
7:0	-	Overscan height - top			
15:8	-	Overscan height - bottom			

Register Description:

1. The Vertical Overscan Control register defines the top and bottom overscan heights in line units.

Overscan Color Register					
02EE (W)	OVERSCAN_COLOR_8				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
7:0	OVERSCAN_ COLOR_8	The 4- and 8-bpp overscan color			

24-Bit Blue Overscan Component					
02EF (W)	OVERSCAN_BLUE_24				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
7:0	OVERSCAN_ BLUE_24	The 16- and 24-bpp blue overscan color			

Register Description:

1. 16-bpp video modes do not use the lower color bits.
2. The other two color components are defined in registers OVERSCAN_GREEN (06EE) and OVERSCAN_RED (06EF).

24-Bit Green Overscan Component					
06EE (W)	OVERSCAN_GREEN_24				
	-	-	-	Mach 32	
Bit	Mnemonic	Description			
7:0	OVERSCAN_ GREEN_24	The 16- and 24-bpp green overscan color			

Register Description:

1. 16-bpp video modes do not use the lower color bits.
2. The other two color components are defined in registers OVERSCAN_BLUE (02EF) and OVERSCAN_RED (06EF).

24-Bit Red Overscan Component					
06EF (W)	OVERSCAN_RED_24				
	-	-	-	Mach 32	
Bit	Mnemonic	Description			
7:0	OVERSCAN_RED_24	The 16- and 24-bpp red overscan color			

Register Description:

1. 16-bpp video modes do not use the lower color bits.
2. The other two color components are defined in registers OVERSCAN_BLUE (02EF) and OVERSCAN_GREEN (06EE).

Memory Boundary

The memory boundary register controls a logical memory divider that separates available video memory for use by the VGA and the accelerator. Without memory boundary, VGA and accelerator will be permitted to write anywhere in the video memory.

Memory boundary is specified in units of 256 KByte pages, in the MEM_PAGE_BNDRY field of this register. It is enabled by setting MEM_BDRY_ENA to logical one.

With memory boundary enabled, all memory writes through the 64K aperture are handled as follows: VGA drawing circuitry is prohibited from writing to any memory page above or at the specified boundary. At the same time, the accelerator drawing circuitry is prohibited from writing to any memory page below the boundary.

The memory boundary setting has no effect on memory aperture. See *Memory Interface* for details.

Memory Boundary Register					
42EE (R/W)	MEM_BNDRY				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
3:0	MEM_PAGE_BNDRY	Video memory boundary specified in units of 256KB-pages: VGA cannot access memory page >= 8514 cannot access memory page <			
4	MEM_BNDRY_ENA	Video memory partition enable flag: **0** = VGA and 8514 drawing circuitry can write anywhere in the video memory **1** = VGA and 8514 drawing circuitry must write within their memory partition			
15:5	-	Reserved			

Memory Interface

The direct memory interface (DMA) allows fast image data transfer by mapping the video memory to the system memory address space. The size of this memory aperture is normally either 1MB or 4MB as specified by MEM_APERTURE_SEL. This aperture must be aligned to a 1M byte page boundary which is specified by MEM_APERTURE_LOC. Page selection of a 1M byte aperture is controlled by MEM_APERTURE_PAGE. The DMA is fully software programmable.

A 64K aperture is also available if the VGA circuitry is enabled. This aperture is accessible at the VGA graphic page (B800). To set it up, the VGA must have initial access to the full address range; therefore, the DMA should be disabled during setup, i.e., MEM_APERTURE_SEL = 0.

Memory Aperture Configuration Registers					
5EEE (R/W)		MEM_CFG			
		-	-	-	Mach 32
Bit	**Mnemonic**	**Description**			
1:0	MEM_APERT_SEL	The direct memory interface maps the video memory to the system memory address space. Size of memory aperture is as follows: **0** = Direct memory interface disabled **1** = 1MB memory aperture **2** = 4MB memory aperture **3** = Reserved Note: The PCI configuration only supports **0** and **2** i.e., disabled or 4MB aperture			
3:2	MEM_APERT_PAGE	Memory aperture page select of video memory configurations: **0** = Page 0 **1** = Page 1 **2** = Page 2 **3** = Page 3			
7:4	-	**ATI68800-6:** Multiplexed local bus configurations: Bits 7:4 are used together with bits 15:8 to allow mapping of the aperture between 0 and 4GB. It is done through input straps, 16EE[13] = 1 **PCI configurations:** Memory aperture base is always bits 15:4.			

Memory Aperture Configuration Registers				
5EEE (R/W)	MEM_CFG			
	-	-	-	Mach 32

Bit	Mnemonic	Description
15:8	MEM_APERT_LOC	Memory aperture location aligned to any 1MB boundary, between 0 and 127MB.

Register Description:

1. PCI configuration — uses only bits 1:0 (value is always "2") and bits 15:4 which is always the Memory Aperture Base. Bits 3:2 are reserved.

Memory Aperture Configuration Registers				
6AEE (R/W)	APERTURE_CNTL (PCI)			
	-	-	-	Mach 32

Bit	Mnemonic	Description	Power Up Default
9:0	-	Reserved	0
10	F_APERT_ENA	Enables zero waitstate aperture write, i.e., a single write to the aperture can be done in as few as 2 bus cycles.	0 (Disabled)
11	FIFO_RD_AHEAD	Enables Read Ahead for aperture read operation — speeds up host memory reads if reads are sequential.	0 (Disabled)
12	SCLK_DLY	Pixel Stream 1 SCLK Delay	0 (No Delay)
13	DEC_BURST_ENA	**0** = Disables Decrement Burst **1** = Enables Decrement Burst	0
14	INC_BURST	Direction of burst: **0** = Decrements Burst **1** = Increments Burst	0
15	PCI_TIMEOUT_DIS	Bus Timeout on burst read/writes **0** = Enables Timeout **1** = Disables Timeout	0

Register Description:

1. This register description is applicable to the PCI configuration only. 6AEE-RW for the ATI68800-6 is the MAX_WAITSTATES register, described on page 9-11.

Hardware Cursor

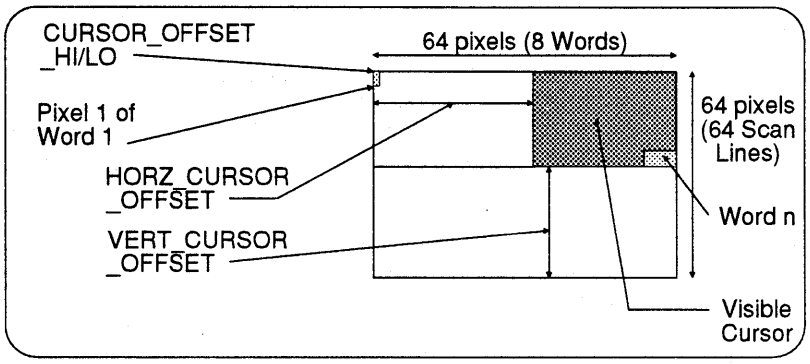
The hardware cursor is programmable to be any size up to 64x64 pixels. The cursor shape is defined in off-screen memory and offers 2 colors plus transparent and complement. Cursor pitch is always 64 pixels, meaning the cursor definition is always 64 pixels (eight words) wide although pixels outside the visible cursor area are ignored. The cursor definition is in reverse pixel order within each word. See Cursor Definition Mapping on the following page. Once the cursor is defined, it is moved around on screen simply by updating the cursor position. This cursor is available in any accelerator display mode. The color of each cursor pixel is defined by two bits, as follows:

2-Bit Pixel Value	Pixel Color
00	Cursor Color 0
01	Cursor Color 1
10	Transparent
11	Complement

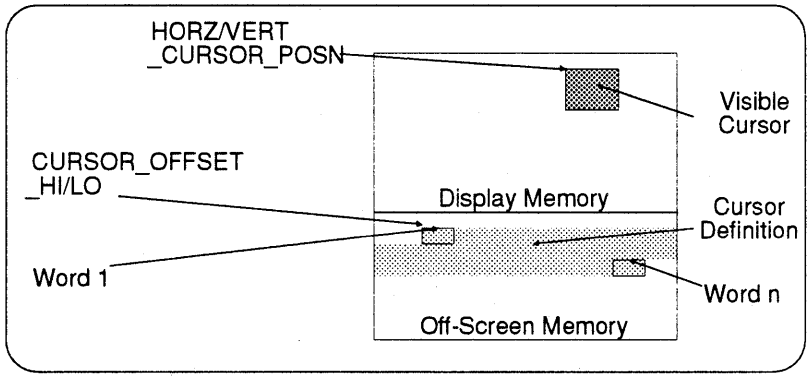
The cursor is stored as a linear block of off-screen video memory, starting at an address specified by registers CURSOR_OFFSET_HI and CURSOR_OFFSET_LO. The upper left corner of the cursor is specified by HORZ_CURSOR_POSN and VERT_CURSOR_POSN. The cursor size may be as large as 64x64 pixels, as specified by HORZ_CURSOR_OFFSET and VERT_CURSOR_OFFSET. The cursor is enabled if the CURSOR_OFFSET_HI[15] bit to set to logical one. Once defined, the cursor is moved by simply updating the cursor position.

Cursor color in 4 and 8 bpp modes is defined by registers CURSOR_COLOR_0 and CURSOR_COLOR_1. Cursor color in 16 and 24 bpp modes requires two additional registers — EXT_CURSOR_COLOR_0 and EXT_CURSOR_COLOR_1.

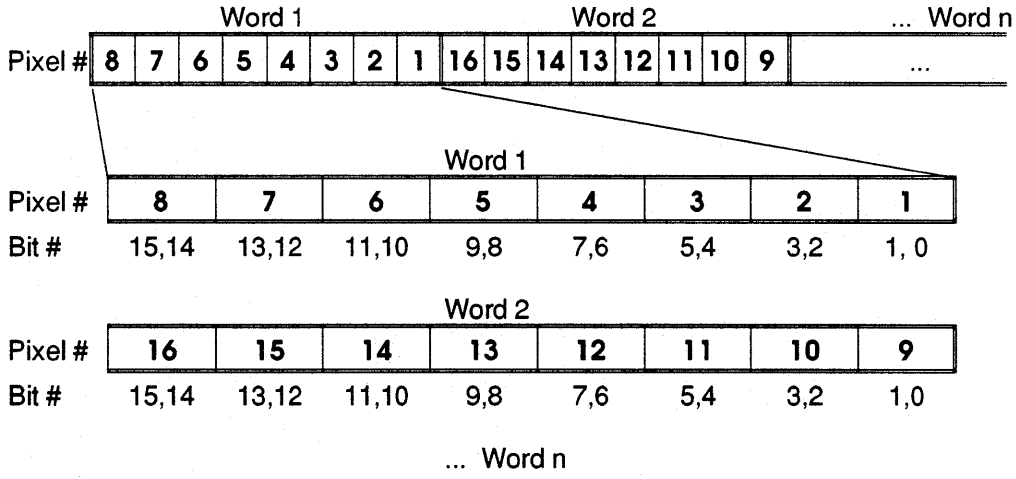
Cursor Definition Mapping



Cursor Definition



Physical Location of Cursor and Cursor Definition



Cursor Offset Low Register					
0AEE (W)	CURSOR_OFFSET_LO				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
15:0	CURSOR_OFFSET_ LO	The lower bits of the cursor offset, a 2 bits/pixel count from the start of display memory. Also see CURSOR_ OFFSET_HI (0EEE).			

Cursor Offset High Register					
0EEE (W)	CURSOR_OFFSET_HI				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
3:0	CURSOR_OFFSET_ HI	The upper bits of the cursor offset, a 2 bits/pixel count from the start of display memory. Also see CURSOR_OFFSET_LO (0AEE).			
14:4	-	Reserved			
15	CURSOR_ENA	Cursor enable			

Horizontal Cursor Position Registers					
12EE (W)	HORZ_CURSOR_POSN				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
10:0	H_CUR_POSN	Screen position of the cursor specified in units of eight pixels — measured horizontally from the left edge of the display area.			
15:11	-	Reserved			

Vertical Cursor Position Register					
16EE (W)	VERT_CURSOR_POSN				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
11:0	V_CUR_POSN	Screen position of the cursor specified in lines — measured vertically from the top edge of the display area.			
15:12	-	Reserved			

Cursor Color 0 Register					
1AEE (W)	CURSOR_COLOR_0				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
7:0	CUR_COLOR_0	**4- and 8-bpp modes**: Cursor color 0			
	EXT_CURSOR_ COLOR_0_B	**16- and 24-bpp modes**: Blue extended cursor color 0. Also see EXT_CURSOR_COLOR_0 (3AEE).			

Cursor Color 1 Register					
1AEF (W)	CURSOR_COLOR_1				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
7:0	CUR_COLOR_1	**4- and 8-bpp modes**: Cursor color 1			
	EXT_CURSOR_ COLOR_1_B	**16- and 24-bpp modes**: Extended blue cursor color 1. Also see EXT_CURSOR_COLOR_1 (3EEE).			

Horizontal Cursor Offset Register					
1EEE (W)	HORZ_CURSOR_OFFSET				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
5:0	CUR_H_OFFSET	A horizontal offset at which to begin the display of the 64x64 pixel cursor. Also see CUR_V_OFFSET (1EEF).			
7:6	-	Reserved			
13:8	-	Reserved			
15:14	-	Reserved			

Vertical Cursor Offset Register					
1EEF (W)	VERT_CURSOR_OFFSET				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
5:0	CUR_V_OFFSET	A vertical offset at which to begin the display of the 64x64 pixel cursor. Also see CUR_H_OFFSET (1EEE).			
7:6	-	Reserved			
13:8	-	Reserved			
15:14	-	Reserved			

Extended Cursor Color 0 Register					
3AEE (W)	EXT_CURSOR_COLOR_0				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
7:0	EXT_CUR_COL_0_G	16- and 24-bpp green extended cursor color 0			
15:8	EXT_CUR_COL_0_R	16- and 24-bpp red extended cursor color 0			

Register Description:

1. Blue extended cursor color 0 is at CURSOR_COLOR_0 (1AEE).

Extended Cursor Color 1 Register					
3EEE (W)	EXT_CURSOR_COLOR_1				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
7:0	EXT_CUR_COL_1_G	16- and 24-bpp green extended cursor color 1			
15:8	EXT_CUR_COL_1_R	16- and 24-bpp red extended cursor color 1			

Register Description:

1. Blue extended cursor color 1 is at CURSOR_COLOR_1 (1AEF).

Miscellaneous Registers

Local Control Register					
32EE (R/W)	LOCAL_CNTL				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
0	MED_ NON-PAGE-CYC	Enables 6 clock non-page cycle.			
1	LONG_ NON-PAGE-CYC	Enables 7 clock non-page cycle.			
2	SHORT_CAS_ PULSE_EN	Enables 1/2 memory clock CAS precharge time			
3	DAC_BLANK_ADJ	Enables DAC to be clocked on positive clock edge			
4	FIFO_TEST	Enables testing of FIFO			
6:5	Bits 6:5: FIFO_TIMING_ADJ	**ATI68800-3:** Enables filtering of 1 clock IOW low or high pulse			
	Bit 5: MEM_MAP_ENA	**ATI68800-6:** Memory Mapped Register Enable			
	Bit 6: LOC_BIOS_ENA	**ATI68800-6:** Local Bus BIOS ROM Decode Enable, provided that Strap bit P15 is also enabled. (Local decode is disabled when either this bit or P15 is set for disable.)			
9:7	ROM_WAIT	Number of ROM wait states. Default=7.			
11:10	MEM_R_DELAY	Additional wait states inserted for memory reads. Default = 3			
15:12	LOCAL_BUS_WAIT	**ATI68800-x:** Minimum local bus wait states. Default=15			
	8514IO_WAIT	**ATI68800AX** PCI Control only: Modifies the number of wait states on I/O read/write operations e.g., 3 = increases the wait states by 3 bus clocks.			

ROM/EEPROM/DAC Control Register					
7EEE (W)	MISC_CNTL				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
0	EE_DATA_OUT	Data output for the EEPROM			
1	EE_CLK	Clock signal for the EEPROM			
2	EE_CS	Chip select for the EEPROM			
3	EE_SELECT	Enables the external EEPROM for reading or writing. The EEPROM will not respond until this bit is set to logical one.			
7:4	ROM_PAGE_SEL	Selects a 2K page within the 32K ROM			
9:8	BLANK_ADJUST	Type 2 DAC only (ATI68875): Delays BLANK by 1 or 2 PCLK			
11:10	PIXEL_DELAY	Adjusts pixel data skew from PCLK (adjusts setup and hold times to suit different DACs)			
12	PASSTHRU_ OVERIDE	8514/A mode: Allows the pixel clock to remain active even when PASSTHROUGH = 0.			
15:13	CARD_SELECT	For selecting a card in a multi-card system. The card is selected if its card select strap value is equal to CARD_SELECT or zero.			

Register Description:

1. The ROM Page Select and EEPROM Control register allows software-controlled reading and writing of user configuration data stored in the EEPROM.
2. This register is read back at 92EE.

ROM Page Select & EEPROM Control Register					
92EE (R)	R_MISC_CNTL				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
3:0	-	Reserved			
7:4	ROM_PAGE_SEL	Selects a 2K page within the 32K ROM			
9:8	BLANK_ADJUST	Type 2 DAC only (ATI68875): Delays BLANK_1_PCLK			
11:10	PIXEL_DELAY	Adjusts pixel data skew from PCLK (adjusts setup and hold times to suit different DACs)			
15:12	-	Reserved			

FIFO Test Data Register					
1AEE (R)	FIFO_TEST_DATA				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
15:0	-	FIFO test data			

FIFO Test Tag Register					
3AEE (R)	FIFO_TEST_TAG				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
4:0	-	FIFO test data			
7:5	-	Reserved			

R Horizontal Displayed & Total Register					
B2EE (R)	R_H_TOTAL&DISP				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
7:0	H_DISP	Displayed portion of the screen — measured horizontally in units of eight pixels.			
15:8	H_TOTAL	Total horizontal area of the screen in units if eight pixels — pixel count measured from the beginning of one row to the beginning of the next row.			

Register Description:

1. This is the read register of both H_TOTAL (02E8) and H_DISP (06E8).

Horizontal Sync Start Register					
B6EE (R)	R_H_SYNC_STRT				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
7:0	H_SYNC_STRT	Screen position of the H sync specified in units of eight pixels — measured horizontally from the start of the display area.			

Register Description:

1. This is the read register of H_SYNC_STRT (0AE8).

Horizontal Sync Width Register					
BAEE (R)	R_H_SYNC_WID				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
4:0	H_WIDTH	Width of the sync width measured horizontally in units of eight pixels.			
5	H_POLARITY	**0** = Positive H sync polarity **1** = Negative H sync polarity			
7:6	-	Reserved			

Register Description:

1. This is the read register of H_SYNC_WID (0EE8).

Vertical Total Register					
C2EE (R)	R_V_TOTAL				
	-	8514/A	Mach 8	Mach 32	
Bit	**Mnemonic**	**Description**			
11:0	V_TOTAL	Size of the screen specified in lines — line count measured from the first line of one frame to the first line of the next frame.			
15:12	-	Reserved			

Register Description:

1. This is the read register of V_TOTAL (12E8).

Vertical Displayed Register					
C6EE (R)	R_V_DISP				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
11:0	V_DISP	Displayed portion of the screen — measured vertically in lines.			
15:12	-	Reserved			

Register Description:

1. This is the read register of V_DISP (16E8).

Vertical Sync Start Register					
CAEE (R)	R_V_SYNC_STRT				
	-	-	-	Mach 32	
Bit	**Mnemonic**	**Description**			
11:0	V_SYNC_STRT	Screen position of the V sync specified in lines — measured from the top edge of the display.			
15:12	-	Reserved			

Register Description:

1. This is the read register of V_SYNC_STRT (1AE8).

Vertical Sync Width Register				
D2EE (R)	R_V_SYNC_WID			
	-	8514/A	Mach 8	Mach 32
Bit	**Mnemonic**	**Description**		
4:0	V_WIDTH	Width of the sync pulse measured vertically in lines		
5	V_POLARITY	**0** = Positive V sync polarity **1** = Negative V sync polarity		
15:6	-	Reserved		

Register Description:

1. This is the read register of V_SYNC_WID (1EE8).

Chip ID Register					
FAEE (R)	CHIP_ID				
	-				Mach 32
Bit	**Mnemonic**	**Description**			
4:0	CHIP_CODE_0	The chip ID is a coded (abbreviated) ASCII number which is decoded by adding 41h to it. For example, if bits 4:0 = 17h, adding 41h to it yields 58h, which is ASCII for **X** Current chip ID: **XX** = ATI68800-6 **LX** = ATI68800LX **AX** = ATI68800AX			
9:5	CHIP_CODE_1				
11:10	CHIP_CLASS	The class number of the controller			
15:12	CHIP_REV	The revision number of the controller			

Register Description:

1. This register is only applicable to ATI68800-6 and higher ***mach32*** controllers.

Scratch Pad 0 Register					
52EE (R/W)	SCRATCH_PAD_0				
	-	-	-		Mach 32
Bit	**Mnemonic**	**Description**			
15:0	-	Scratch pad 0 — for use by the boot ROM. Do not overwrite.			

Scratch Pad 1 Register					
56EE (R/W)	SCRATCH_PAD_1				
	-	-	-		Mach 32
Bit	**Mnemonic**	**Description**			
15:0	-	Scratch pad 1 — for use by the boot ROM. Do not overwrite.			

PCI Control Register					
22EE (R/W)	**PCI_CNTL**				
	-	-	-	**Mach 32**	
Bit	**Mnemonic**	**Description**			**Power Up Default**
2:0	DAC_RW_WS	RAMDAC Read/Write Wait States — the length of active-low DAC R/W pulse			0
3	TARGET_ABORT_EN	Enable Target Abort Cycle **0** = Disable Target Abort **1** = Enable Target Abort			0
4	PCI_DAC_DLY	PCI DAC Delay **0** = No delay **1** = Increases the hold time of the register select signals w.r.t. the falling edge of the DAC Write signal.			0
5	DAC_SNOOP_EN	**0** = No snooping on DAC read **1** = Snooping on DAC read			
6	FAST_BURST	**0** = Fast Burst disabled **1** = 0WS on aperture Burst Write			
7	FAST_MEM_IO	**0** = No Fast Mem. I/O R/W **1** = Fast memory-mapped I/O read/write operation enabled			
15:8	-	Reserved			-

DAC Registers

The ***mach32*** supports a wide variety of RAMDACs. Special consideration should be made for 16bpp and 24/32bpp (bits per pixel) modes when using the ATI 68860 or any other HiColor RAMDAC — The recommended method of initializing a DAC is through a BIOS call. See Appendix F for specifications, DAC register descriptions, and programming examples.

A

Coprocessor Registers

Register Names and Addresses

This appendix provides two register listings showing the register attributes and page references to their descriptions — the first one is sorted by register names, the second by I/O addresses. Both lists contain 8514/A-compatible registers and ATI-extended registers.

One way to differentiate between 8514/A-compatible registers and ATI-extended registers is by looking at their I/O addresses. Compatible registers usually end in "8", for example, EE8h. Extended registers usually end in "EE", as in 36EEh.

VGA register listings are provided at the beginning of each VGA chapter — IBM-compatible registers are in Chapter 5, ATI-extended VGA registers in Chapter 6.

Register	Port	Attributes	Page
ADVFUNC_CNTL	4AE8h	W	8-6
ALU_BG_FN	B6EEh	W	9-47
ALU_FG_FN	BAEEh	W	9-47
BKGD_COLOR	A2E8h	W	8-27
BKGD_MIX	B6E8h	W	8-27
BOUNDS_BOTTOM	7EEEh	R	9-48
BOUNDS_LEFT	72EEh	R	9-48
BOUNDS_RIGHT	7AEEh	R	9-48
BOUNDS_TOP	76EEh	R	9-48
BRES_COUNT	96EEh	R/W	9-49
CHIP_ID	FAEEh	R	9-88
CLOCK_SEL	4AEEh	R/W	9-4
CMD	9AE8h	W	8-28
CMP_COLOR	B2E8h	W	8-36
CONFIG_STATUS_1 (mach8)	12EEh	R	9-64
CONFIG_STATUS_1 (mach32)	12EEh	R	9-65
CONFIG_STATUS_2 (mach8)	16EEh	R	9-66
CONFIG_STATUS_2 (mach32)	16EEh	R	9-67
CRT_OFFSET_HI	2EEEh	W	9-5
CRT_OFFSET_LO	2AEEh	W	9-5
CRT_PITCH	26EEh	W	9-5
CUR_X	86E8h	R/W	8-37
CUR_Y	82E8h	R	8-38
CURSOR_COLOR_0	1AEEh	W	9-80
CURSOR_COLOR_1	1AEFh	W	9-80
CURSOR_OFFSET_LO	0AEEh	W	9-79
CURSOR_OFFSET_HI	0EEEh	W	9-79
DAC_CONT (PCI)	22EEh	R/W	9-89
DAC_DATA	02EDh	R/W	8-2
DAC_MASK	02EAh	R/W	8-2
DAC_R_INDEX	02EBh	R/W	8-2
DAC_W_INDEX	02ECh	R/W	8-3
DEST_CMP_FN	EEEEh	W	9-50
DEST_COLOR_CMP_MASK	F2EEh	R/W	9-51
DEST_X_END	AAEEh	W	9-52
DEST_X_START	A6EEh	W	9-52
DEST_Y_END	AEEEh	W	9-52
DISP_CNTL	22E8h	W	8-7
DISP_STATUS	02E8h	R	8-8
DP_CONFIG	CEEEh	W	9-15
ERR_TERM	92E8h	R/W	8-39
EXT_CURSOR_COLOR_0	3AEEh	W	9-82
EXT_CURSOR_COLOR_1	3EEEh	W	9-82
EXT_FIFO_STATUS	9AEEh	R	9-16
EXT_GE_CONFIG (mach8 8-bit)	7AEEh	W	9-17
EXT_GE_CONFIG (mach8 16-bit)	7AEEh	W	9-18

Register	Port	Attributes	Page
EXT_GE_CONFIG (mach32)	7AEEh	W	9-19
EXT_GE_STATUS	62EEh	R	9-68
EXT_SCISSOR_B	E6EEh	W	9-53
EXT_SCISSOR_L	DAEEh	W	9-53
EXT_SCISSOR_R	E2EEh	W	9-54
EXT_SCISSOR_T	DEEEh	W	9-54
EXT_SHORT_STROKE	C6EEh	W	9-55
FIFO_OPT	36EEh	W	9-9
FIFO_TEST_DATA	1AEEh	R	9-85
FIFO_TEST_TAG	3AEEh	R	9-85
FRGD_COLOR	A6E8h	W	8-40
FRGD_MIX	BAE8h	W	8-40
GE_OFFSET_HI	72EEh	W	9-21
GE_OFFSET_LO	6EEEh	W	9-21
GE_PITCH	76EEh	W	9-20
GE_STAT	9AE8h	R	8-41
GENENA (Add-On)	46E8h	W	5-8
H_DISP	06E8h	W	8-8
H_SYNC_STRT	0AE8h	W	8-9
H_SYNC_WID	0EE8h	W	8-9
H_TOTAL	02E8h	W	8-9
HORZ_CURSOR_OFFSET	1EEEh	W	9-81
HORZ_CURSOR_POSN	12EEh	W	9-79
HORZ_OVERSCAN	62EEh	W	9-71
LINEDRAW	FEEEh	W	9-56
LINEDRAW_INDEX	9AEEh	W	9-57
LINEDRAW_OPT	A2EEh	R/W	9-22
LOCAL_CNTL	32EEh	R/W	9-83
MAJ_AXIS_PCNT	96E8h	W	8-42
MAX_WAITSTATES (mach8)	6AEEh	R/W	9-10
MAX_WAITSTATES (mach32)	6AEEh	R/W	9-11
MEM_BNDRY	42EEh	R/W	9-74
MEM_CFG	5EEEh	R/W	9-75
MEM_CNTL	BEE8*5h	W	8-17
MIN_AXIS_PCNT	BEE8*0h	W	8-43
MISC_CNTL	7EEEh	W	9-84
MISC_CONT (PCI)	6AEEh	R/W	9-76
MISC_OPTIONS	36EEh	R/W	9-12
OVERSCAN_BLUE_24	02EFh	W	9-72
OVERSCAN_COLOR_8	02EEh	W	9-72
OVERSCAN_GREEN_24	06EEh	W	9-73
OVERSCAN_RED_24	06EFh	W	9-73
PATT_DATA	8EEEh	W	9-58
PATT_DATA_INDEX	82EEh	R/W	9-59
PATT_INDEX	D6EEh	W	9-59
PATT_LENGTH	D2EEh	W	9-60

Register	Port	Attributes	Page
PATTERN_H	BEE8*9h	W	8-44
PATTERN_L	BEE8*8h	W	8-45
PIX_TRANS	E2E8h	R/W	8-47
PIXEL_CNTL	BEE8*Ah	W	8-46
R_EXT_GE_CONFIG	8EEEh	R	9-24
R_H_SYNC_STRT	B6EEh	R	9-86
R_H_SYNC_WID	BAEEh	R	9-86
R_H_TOTAL&DISP	B2EEh	R	9-85
R_MISC_CNTL	92EEh	R	9-84
R_SRC_X	DAEEh	R	9-60
R_SRC_Y	DEEEh	R	9-61
R_V_DISP	C6EEh	R	9-87
R_V_SYNC_STRT	CAEEh	R	9-87
R_V_SYNC_WID	D2EEh	R	9-87
R_V_TOTAL	C2EEh	R	9-86
RD_MASK	AEE8h	W	8-48
ROM_SETUP, EISA	zC85h	W	2-6
ROM_SETUP, uC	0103h	W	2-4
SCAN_X	CAEEh	W	9-61
SCISSOR_B	BEE8*3h	W	8-49
SCISSOR_L	BEE8*2h	W	8-49
SCISSOR_R	BEE8*4h	W	8-50
SCISSOR_T	BEE8*1h	W	8-50
SCRATCH_PAD_0	52EEh	R/W	9-88
SCRATCH_PAD_1	56EEh	R/W	9-88
SETUP_1, EISA	zC86h	W	2-7
SETUP_1, uC	0104h	W	2-4
SETUP_2, EISA	zC87h	W	2-7
SETUP_2, uC	0105h	W	2-4
SETUP_ID1, EISA	zC80h	R	2-5
SETUP_ID1, uC	0100h	R	2-3
SETUP_ID2, EISA	zC81h	R	2-5
SETUP_ID2, uC	0101h	R	2-3
SETUP_ID3, EISA	zC82h	R	2-5
SETUP_ID4, EISA	zC83h	R	2-6
SETUP_OPT, EISA	zC84h	R/W	2-6
SETUP_OPT, uC	0102h	R/W	2-3
SHADOW_CTL	46EEh	W	9-6
SHADOW_SET	5AEEh	W	9-7
SHORT_STROKE	9EE8h	W	8-51
SRC_X/DEST_X/DIASTP	8EE8h	W	8-52
SRC_X_END	BEEEh	W	9-62
SRC_X_START	B2EEh	W	9-62
SRC_Y/DEST_Y/AXSTP	8AE8h	W	8-53
SRC_Y_DIR	C2EEh	W	9-62
SUBSYS_CNTL	42E8h	W	8-18

Register	Port	Attributes	Page
SUBSYS_STATUS	42E8h	R	8-20
V_DISP	16E8h	W	8-10
V_SYNC_STRT	1AE8h	W	8-11
V_SYNC_WID	1EE8h	W	8-12
V_TOTAL	12E8h	W	8-12
VERT_CURSOR_OFFSET	1EEFh	W	9-81
VERT_CURSOR_POSN	16EEh	W	9-80
VERT_LINE_CNTR	CEEEh	R	9-7
VERT_OVERSCAN	66EEh	W	9-72
WRT_MASK	AAE8h	W	8-54

Port	Register	Attributes	Page
0100h	SETUP_ID1, uC	R	2-3
0101h	SETUP_ID2, uC	R	2-3
0102h	SETUP_OPT, uC	R/W	2-3
0103h	ROM_SETUP, uC	W	2-4
0104h	SETUP_1, uC	W	2-4
0105h	SETUP_2, uC	W	2-4
02E8h	DISP_STATUS	R	8-8
02E8h	H_TOTAL	W	8-9
02EAh	DAC_MASK	R/W	8-2
02EBh	DAC_R_INDEX	R/W	8-2
02ECh	DAC_W_INDEX	R/W	8-3
02EDh	DAC_DATA	R/W	8-2
02EEh	OVERSCAN_COLOR_8	W	9-72
02EFh	OVERSCAN_BLUE_24	W	9-72
06E8h	H_DISP	W	8-8
06EEh	OVERSCAN_GREEN_24	W	9-73
06EFh	OVERSCAN_RED_24	W	9-73
0AE8h	H_SYNC_STRT	W	8-9
0AEEh	CURSOR_OFFSET_LO	W	9-79
0EE8h	H_SYNC_WID	W	8-9
0EEEh	CURSOR_OFFSET_HI	W	9-79
12E8h	V_TOTAL	W	8-12
12EEh	CONFIG_STATUS_1 (mach8)	R	9-64
12EEh	CONFIG_STATUS_1 (mach32)	R	9-65
12EEh	HORZ_CURSOR_POSN	W	9-79
16E8h	V_DISP	W	8-10
16EEh	CONFIG_STATUS_2 (mach8)	R	9-66
16EEh	CONFIG_STATUS_2 (mach32)	R	9-67
16EEh	VERT_CURSOR_POSN	W	9-80
1AE8h	V_SYNC_STRT	W	8-11
1AEEh	FIFO_TEST_DATA	R	9-85
1AEEh	CURSOR_COLOR_0	W	9-80
1AEFh	CURSOR_COLOR_1	W	9-80
1EE8h	V_SYNC_WID	W	8-12
1EEEh	HORZ_CURSOR_OFFSET	W	9-81
1EEFh	VERT_CURSOR_OFFSET	W	9-81
22E8h	DISP_CNTL	W	8-7
22EEh	DAC_CONT (PCI)	R/W	9-89
26EEh	CRT_PITCH	W	9-5
2AEEh	CRT_OFFSET_LO	W	9-5
2EEEh	CRT_OFFSET_HI	W	9-5
32EEh	LOCAL_CNTL	R/W	9-83

Port	Register	Attributes	Page
36EEh	MISC_OPTIONS	R/W	9-12
36EEh	FIFO_OPT	W	9-9
3AEEh	FIFO_TEST_TAG	R	9-85
3AEEh	EXT_CURSOR_COLOR_0	W	9-82
3EEEh	EXT_CURSOR_COLOR_1	W	9-82
42E8h	SUBSYS_STATUS	R	8-20
42E8h	SUBSYS_CNTL	W	8-18
42EEh	MEM_BNDRY	R/W	9-74
46E8h	GENENA (Add-On)	W	5-8
46EEh	SHADOW_CTL	W	9-6
4AE8h	ADVFUNC_CNTL	W	8-6
4AEEh	CLOCK_SEL	R/W	9-4
52EEh	SCRATCH_PAD_0	R/W	9-88
56EEh	SCRATCH_PAD_1	R/W	9-88
5AEEh	SHADOW_SET	W	9-7
5EEEh	MEM_CFG	R/W	9-75
62EEh	EXT_GE_STATUS	R	9-68
62EEh	HORZ_OVERSCAN	W	9-71
66EEh	VERT_OVERSCAN	W	9-72
6AEEh	MAX_WAITSTATES (mach8)	R/W	9-10
6AEEh	MAX_WAITSTATES (mach32)	R/W	9-11
6AEEh	MISC_CONT (PCI)	R/W	9-76
6EEEh	GE_OFFSET_LO	W	9-21
72EEh	BOUNDS_LEFT	R	9-48
72EEh	GE_OFFSET_HI	W	9-21
76EEh	BOUNDS_TOP	R	9-48
76EEh	GE_PITCH	W	9-20
7AEEh	BOUNDS_RIGHT	R	9-48
7AEEh	EXT_GE_CONFIG (mach8 8-bit)	W	9-17
7AEEh	EXT_GE_CONFIG (mach8 16-bit)	W	9-18
7AEEh	EXT_GE_CONFIG (mach32)	W	9-19
7EEEh	BOUNDS_BOTTOM	R	9-48
7EEEh	MISC_CNTL	W	9-84
82E8h	CUR_Y	R/W	8-38
82EEh	PATT_DATA_INDEX	R/W	9-59
86E8h	CUR_X	R/W	8-37
8AE8h	SRC_Y/DEST_Y/AXSTP	W	8-53
8EE8h	SRC_X/DEST_X/DIASTP	W	8-52
8EEEh	R_EXT_GE_CONFIG	R	9-24
8EEEh	PATT_DATA	W	9-58
92E8h	ERR_TERM	R/W	8-39
92EEh	R_MISC_CNTL	R	9-84
96E8h	MAJ_AXIS_PCNT	W	8-42

Port	Register	Attributes	Page
96EEh	BRES_COUNT	R/W	9-49
9AE8h	GE_STAT	R	8-41
9AE8h	CMD	W	8-28
9AEEh	EXT_FIFO_STATUS	R	9-16
9AEEh	LINEDRAW_INDEX	W	9-57
9EE8h	SHORT_STROKE	W	8-51
A2E8h	BKGD_COLOR	W	8-27
A2EEh	LINEDRAW_OPT	R/W	9-22
A6E8h	FRGD_COLOR	W	8-40
A6EEh	DEST_X_START	W	9-52
AAE8h	WRT_MASK	W	8-54
AAEEh	DEST_X_END	W	9-52
AEE8h	RD_MASK	W	8-48
AEEEh	DEST_Y_END	W	9-52
B2E8h	CMP_COLOR	W	8-36
B2EEh	R_H_TOTAL&DISP	R	9-85
B2EEh	SRC_X_START	W	9-62
B6E8h	BKGD_MIX	W	8-27
B6EEh	R_H_SYNC_STRT	R	9-86
B6EEh	ALU_BG_FN	W	9-47
BAE8h	FRGD_MIX	W	8-40
BAEEh	R_H_SYNC_WID	R	9-86
BAEEh	ALU_FG_FN	W	9-47
BEE8*0h	MIN_AXIS_PCNT	W	8-43
BEE8*1h	SCISSOR_T	W	8-50
BEE8*2h	SCISSOR_L	W	8-49
BEE8*3h	SCISSOR_B	W	8-49
BEE8*4h	SCISSOR_R	W	8-50
BEE8*5h	MEM_CNTL	W	8-17
BEE8*8h	PATTERN_L	W	8-45
BEE8*9h	PATTERN_H	W	8-44
BEE8*Ah	PIXEL_CNTL	W	8-46
BEEEh	SRC_X_END	W	9-62
C2EEh	R_V_TOTAL	R	9-86
C2EEh	SRC_Y_DIR	W	9-62
C6EEh	R_V_DISP	R	9-87
C6EEh	EXT_SHORT_STROKE	W	9-55
CAEEh	R_V_SYNC_STRT	R	9-87
CAEEh	SCAN_X	W	9-61
CEEEh	VERT_LINE_CNTR	R	9-7
CEEEh	DP_CONFIG	W	9-15
D2EEh	R_V_SYNC_WID	R	9-87
D2EEh	PATT_LENGTH	W	9-60

Port	Register	Attributes	Page
D6EEh	PATT_INDEX	W	9-59
DAEEh	R_SRC_X	R	9-60
DAEEh	EXT_SCISSOR_L	W	9-53
DEEEh	R_SRC_Y	R	9-61
DEEEh	EXT_SCISSOR_T	W	9-54
E2E8h	PIX_TRANS	R/W	8-47
E2EEh	EXT_SCISSOR_R	W	9-54
E6EEh	EXT_SCISSOR_B	W	9-53
EEEEh	DEST_CMP_FN	W	9-50
F2EEh	DEST_COLOR_CMP_MASK	W	9-51
FAEEh	CHIP_ID	R	9-88
FEEEh	LINEDRAW	W	9-56
zC80h	SETUP_ID1, EISA	R	2-5
zC81h	SETUP_ID2, EISA	R	2-5
zC82h	SETUP_ID3, EISA	R	2-5
zC83h	SETUP_ID4, EISA	R	2-6
zC84h	SETUP_OPT, EISA	R/W	2-6
zC85h	ROM_SETUP, EISA	W	2-6
zC86h	SETUP_1, EISA	W	2-7
zC87h	SETUP_2, EISA	W	2-7

Memory Mapped Registers

Memory mapping is available only in the following ***mach32*** accelerators: ATI68800LX, ATI68800-6, and ATI68800AX.

The register space of the accelerator is mapped onto the 128 dwords at the end of the linear aperture. IBM compatible registers are mapped onto the lower 64 dwords; ATI-extended registers follow directly above them in the top 64 dwords. The exact mapping places register 2E8 first, at offset FFE00 from the base of the aperture. Register 6E8 follows at FFE04 and so on up to register FEEE at offset FFFFC. See illustration and tables below. Each 16-bit register maps into the lower 16 bits of a 32-bit dword. The upper 16 bits, i.e., the shaded region in the illustration are reserved for future use.

For configurations with 4MB on-board video memory, note that the memory at the top of the aperture is otherwise not accessible when memory mapped registers are enabled.

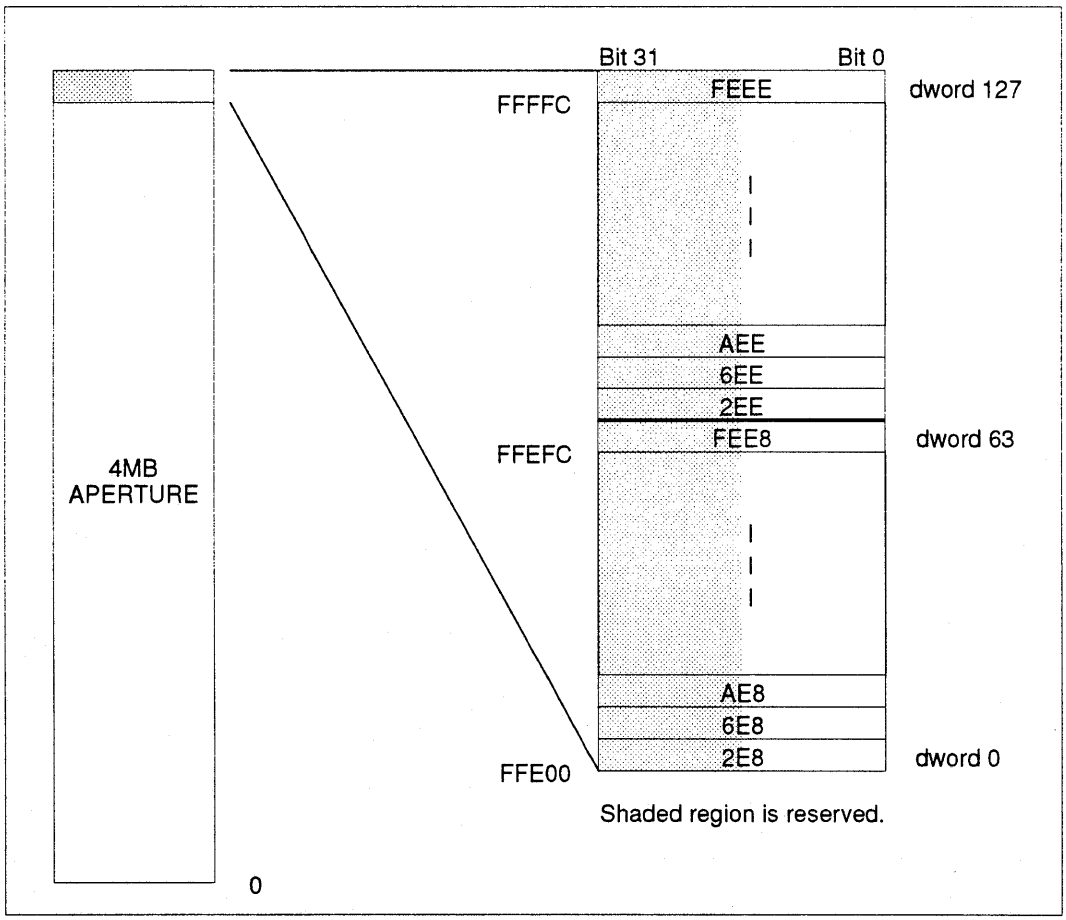


Figure A-1. Memory Mapped Registers

Memory Mapping of IBM-Compatible Registers

I/O Address	Memory Offset	Register Name
2E8	FFE00	DISP_STATUS; H_TOTAL
6E8	FFE04	H_DISP
AE8	FFE08	H_SYNC_STRT
EE8	FFE0C	H_SYNC_WID
12E8	FFE10	V_TOTAL
16E8	FFE14	V_DISP
1AE8	FFE18	V_SYNC_STRT
1EE8	FFE1C	V_SYNC_WID
22E8	FFE20	DISP_CNTL
26E8	FFE24	
2AE8	FFE28	
2EE8	FFE2C	
32E8	FFE30	
36E8	FFE34	
3AE8	FFE38	
3EE8	FFE3C	
42E8	FFE40	SUBSYS_STATUS; SUBSYS_CNTL
46E8	FFE44	ROM_PAGE_SEL
4AE8	FFE48	ADVFUNC_CNTL
4EE8	FFE4C	
52E8	FFE50	
56E8	FFE54	
5AE8	FFE58	
5EE8	FFE5C	
62E8	FFE60	
66E8	FFE64	
6AE8	FFE68	
6EE8	FFE6C	
72E8	FFE70	
76E8	FFE74	
7AE8	FFE78	
7EE8	FFE7C	

I/O Address	Memory Offset	Register Name
82E8	FFE80	CUR_Y
86E8	FFE84	CUR_X
8AE8	FFE88	SRC_Y/DEST_Y/AXSTP
8EE8	FFE8C	SRC_X/DEST_X/DOASTP
92E8	FFE90	ERR_TERM
96E8	FFE94	MAJ_AXIS_PCNT
9AE8	FFE98	GE_STAT; CMD
9EE8	FFE9C	SHORT_STROKE
A2E8	FFEA0	BKGD_COLOR
A6E8	FFEA4	FRGD_COLOR
AAE8	FFEA8	WRT_MASK
AEE8	FFEAC	RD_MASK
B2E8	FFEB0	CMP_COLOR
B6E8	FFEB4	BKGD_MIX
BAE8	FFEB8	FRGD_MIX
BEE8	FFEBC	Multi-Func Control
C2E8	FFEC0	
C6E8	FFEC4	
CAE8	FFEC8	
CEE8	FFECC	
D2E8	FFED0	
D6E8	FFED4	
DAE8	FFED8	
DEE8	FFEDC	
E2E8	FFEE0	PIX_TRANS
E6E8	FFEE4	
EAE8	FFEE8	
EEE8	FFEEC	
F2E8	FFEF0	
F6E8	FFEF4	
FAE8	FFEF8	
FEE8	FFEFC	

Memory Mapping of ATI-Extended Registers

I/O Address	Memory Offset	Register Name
2EE	FFF00	OVERSCAN_COLOR_[24,8]
6EE	FFF04	OVERSCAN_[RED,GREEN]_24
AEE	FFF08	CURSOR_OFFSET_LO
EEE	FFF0C	CURSOR_OFFSET_HI
12EE	FFF10	CONFIG_STATUS_1
16EE	FFF14	CONFIG_STATUS_2; VERT_CURSOR_POSN
1AEE	FFF18	FIFO_TEST_DATA; CURSOR_COLOR_0
1EEE	FFF1C	[VERT/HORZ]_CURSOR_OFFSET
22EE	FFF20	DAC_CONT (PCI)
26EE	FFF24	CRT_PITCH
2AEE	FFF28	CRT_OFFSET_LO
2EEE	FFF2C	CRT_OFFSET_HI
32EE	FFF30	LOCAL_CONTROL
36EE	FFF34	MISC_OPTIONS
3AEE	FFF38	FIFO_TEST_TAG; EXT_CURSOR_COLOR_0
3EEE	FFF3C	EXT_CURSOR_COLOR_1
42EE	FFF40	MEM_BNDRY
46EE	FFF44	SHADOW_CTL
4AEE	FFF48	CLOCK_SEL
4EEE	FFF4C	
52EE	FFF50	SCRATCH_PAD_0
56EE	FFF54	SCRATCH_PAD_1
5AEE	FFF58	SHADOW_SET
5EEE	FFF5C	MEM_CFG
62EE	FFF60	EXT_GE_STATUS; HORZ_OVERSCAN; SYNC (PCI)
66EE	FFF64	VERT_OVERSCAN
6AEE	FFF68	MAX_WAITSTATES; MISC_CONT
6EEE	FFF6C	GE_OFFSET_LO
72EE	FFF70	BOUNDS_LEFT; GE_OFFSET_HI
76EE	FFF74	BOUNDS_TOP; GE_PITCH
7AEE	FFF78	BOUNDS_RIGHT; EXT_GE_CONFIG
7EEE	FFF7C	BOUNDS_BOTTOM; MISC_CNTL

I/O Address	Memory Offset	Register Name
82EE	FFF80	PATT_DATA_INDEX
86EE	FFF84	
8AEE	FFF88	
8EEE	FFF8C	R_EXT_GE_CONFIG; PATT_DATA
92EE	FFF90	R_MISC_CNTL
96EE	FFF94	BRES_COUNT
9AEE	FFF98	EXT_FIFO_STATUS; LINEDRAW_INDEX
9EEE	FFF9C	
A2EE	FFFA0	LINEDRAW_OPT
A6EE	FFFA4	DEST_X_START
AAEE	FFFA8	DEST_X_END
AEEE	FFFAC	DEST_Y_END
B2EE	FFFB0	R_H_TOTAL&DISP; SRC_X_START
B6EE	FFFB4	R_H_SYNC_STRT; ALU_BG_FN
BAEE	FFFB8	R_H_SYNC_WID; ALU_FG_FN
BEEE	FFFBC	SRC_X_END
C2EE	FFFC0	R_V_TOTAL; SRC_Y_DIR
C6EE	FFFC4	R_V_DISP; EXT_SHORT_STROKE
CAEE	FFFC8	R_V_SYNC_STRT; SCAN_X
CEEE	FFFCC	VERT_LINE_CNTR; DP_CONFIG
D2EE	FFFD0	R_V_SYNC_WID; PATT_LENGTH
D6EE	FFFD4	PATT_INDEX
DAEE	FFFD8	R_SRC_X; EXT_SCISSOR_L
DEEE	FFFDC	R_SRC_Y; EXT_SCISSOR_T
E2EE	FFFE0	EXT_SCISSOR_R
E6EE	FFFE4	EXT_SCISSOR_B
EAEE	FFFE8	
EEEE	FFFEC	DEST_CMP_FN
F2EE	FFFF0	DEST_COLOR_CMP_MASK
F6EE	FFFF4	
FAEE	FFFF8	CHIP_ID
FEEE	FFFFC	LINEDRAW

A

Coprocessor Registers

Register Names and Addresses

This appendix provides two register listings showing the register attributes and page references to their descriptions — the first one is sorted by register names, the second by I/O addresses. Both lists contain 8514/A-compatible registers and ATI-extended registers.

One way to differentiate between 8514/A-compatible registers and ATI-extended registers is by looking at their I/O addresses. Compatible registers usually end in "8", for example, EE8h. Extended registers usually end in "EE", as in 36EEh.

VGA register listings are provided at the beginning of each VGA chapter — IBM-compatible registers are in Chapter 5, ATI-extended VGA registers in Chapter 6.

Register	Port	Attributes	Page
ADVFUNC_CNTL	4AE8h	W	8-6
ALU_BG_FN	B6EEh	W	9-47
ALU_FG_FN	BAEEh	W	9-47
BKGD_COLOR	A2E8h	W	8-27
BKGD_MIX	B6E8h	W	8-27
BOUNDS_BOTTOM	7EEEh	R	9-48
BOUNDS_LEFT	72EEh	R	9-48
BOUNDS_RIGHT	7AEEh	R	9-48
BOUNDS_TOP	76EEh	R	9-48
BRES_COUNT	96EEh	R/W	9-49
CHIP_ID	FAEEh	R	9-88
CLOCK_SEL	4AEEh	R/W	9-4
CMD	9AE8h	W	8-28
CMP_COLOR	B2E8h	W	8-36
CONFIG_STATUS_1 (mach8)	12EEh	R	9-64
CONFIG_STATUS_1 (mach32)	12EEh	R	9-65
CONFIG_STATUS_2 (mach8)	16EEh	R	9-66
CONFIG_STATUS_2 (mach32)	16EEh	R	9-67
CRT_OFFSET_HI	2EEEh	W	9-5
CRT_OFFSET_LO	2AEEh	W	9-5
CRT_PITCH	26EEh	W	9-5
CUR_X	86E8h	R/W	8-37
CUR_Y	82E8h	R	8-38
CURSOR_COLOR_0	1AEEh	W	9-80
CURSOR_COLOR_1	1AEFh	W	9-80
CURSOR_OFFSET_LO	0AEEh	W	9-79
CURSOR_OFFSET_HI	0EEEh	W	9-79
DAC_CONT (PCI)	22EEh	R/W	9-89
DAC_DATA	02EDh	R/W	8-2
DAC_MASK	02EAh	R/W	8-2
DAC_R_INDEX	02EBh	R/W	8-2
DAC_W_INDEX	02ECh	R/W	8-3
DEST_CMP_FN	EEEEh	W	9-50
DEST_COLOR_CMP_MASK	F2EEh	R/W	9-51
DEST_X_END	AAEEh	W	9-52
DEST_X_START	A6EEh	W	9-52
DEST_Y_END	AEEEh	W	9-52
DISP_CNTL	22E8h	W	8-7
DISP_STATUS	02E8h	R	8-8
DP_CONFIG	CEEEh	W	9-15
ERR_TERM	92E8h	R/W	8-39
EXT_CURSOR_COLOR_0	3AEEh	W	9-82
EXT_CURSOR_COLOR_1	3EEEh	W	9-82
EXT_FIFO_STATUS	9AEEh	R	9-16
EXT_GE_CONFIG (mach8 8-bit)	7AEEh	W	9-17
EXT_GE_CONFIG (mach8 16-bit)	7AEEh	W	9-18

Register	Port	Attributes	Page
EXT_GE_CONFIG (mach32)	7AEEh	W	9-19
EXT_GE_STATUS	62EEh	R	9-68
EXT_SCISSOR_B	E6EEh	W	9-53
EXT_SCISSOR_L	DAEEh	W	9-53
EXT_SCISSOR_R	E2EEh	W	9-54
EXT_SCISSOR_T	DEEEh	W	9-54
EXT_SHORT_STROKE	C6EEh	W	9-55
FIFO_OPT	36EEh	W	9-9
FIFO_TEST_DATA	1AEEh	R	9-85
FIFO_TEST_TAG	3AEEh	R	9-85
FRGD_COLOR	A6E8h	W	8-40
FRGD_MIX	BAE8h	W	8-40
GE_OFFSET_HI	72EEh	W	9-21
GE_OFFSET_LO	6EEEh	W	9-21
GE_PITCH	76EEh	W	9-20
GE_STAT	9AE8h	R	8-41
GENENA (Add-On)	46E8h	W	5-8
H_DISP	06E8h	W	8-8
H_SYNC_STRT	0AE8h	W	8-9
H_SYNC_WID	0EE8h	W	8-9
H_TOTAL	02E8h	W	8-9
HORZ_CURSOR_OFFSET	1EEEh	W	9-81
HORZ_CURSOR_POSN	12EEh	W	9-79
HORZ_OVERSCAN	62EEh	W	9-71
LINEDRAW	FEEEh	W	9-56
LINEDRAW_INDEX	9AEEh	W	9-57
LINEDRAW_OPT	A2EEh	R/W	9-22
LOCAL_CNTL	32EEh	R/W	9-83
MAJ_AXIS_PCNT	96E8h	W	8-42
MAX_WAITSTATES (mach8)	6AEEh	R/W	9-10
MAX_WAITSTATES (mach32)	6AEEh	R/W	9-11
MEM_BNDRY	42EEh	R/W	9-74
MEM_CFG	5EEEh	R/W	9-75
MEM_CNTL	BEE8*5h	W	8-17
MIN_AXIS_PCNT	BEE8*0h	W	8-43
MISC_CNTL	7EEEh	W	9-84
MISC_CONT (PCI)	6AEEh	R/W	9-76
MISC_OPTIONS	36EEh	R/W	9-12
OVERSCAN_BLUE_24	02EFh	W	9-72
OVERSCAN_COLOR_8	02EEh	W	9-72
OVERSCAN_GREEN_24	06EEh	W	9-73
OVERSCAN_RED_24	06EFh	W	9-73
PATT_DATA	8EEEh	W	9-58
PATT_DATA_INDEX	82EEh	R/W	9-59
PATT_INDEX	D6EEh	W	9-59
PATT_LENGTH	D2EEh	W	9-60

Register	Port	Attributes	Page
PATTERN_H	BEE8*9h	W	8-44
PATTERN_L	BEE8*8h	W	8-45
PIX_TRANS	E2E8h	R/W	8-47
PIXEL_CNTL	BEE8*Ah	W	8-46
R_EXT_GE_CONFIG	8EEEh	R	9-24
R_H_SYNC_STRT	B6EEh	R	9-86
R_H_SYNC_WID	BAEEh	R	9-86
R_H_TOTAL&DISP	B2EEh	R	9-85
R_MISC_CNTL	92EEh	R	9-84
R_SRC_X	DAEEh	R	9-60
R_SRC_Y	DEEEh	R	9-61
R_V_DISP	C6EEh	R	9-87
R_V_SYNC_STRT	CAEEh	R	9-87
R_V_SYNC_WID	D2EEh	R	9-87
R_V_TOTAL	C2EEh	R	9-86
RD_MASK	AEE8h	W	8-48
ROM_SETUP, EISA	zC85h	W	2-6
ROM_SETUP, uC	0103h	W	2-4
SCAN_X	CAEEh	W	9-61
SCISSOR_B	BEE8*3h	W	8-49
SCISSOR_L	BEE8*2h	W	8-49
SCISSOR_R	BEE8*4h	W	8-50
SCISSOR_T	BEE8*1h	W	8-50
SCRATCH_PAD_0	52EEh	R/W	9-88
SCRATCH_PAD_1	56EEh	R/W	9-88
SETUP_1, EISA	zC86h	W	2-7
SETUP_1, uC	0104h	W	2-4
SETUP_2, EISA	zC87h	W	2-7
SETUP_2, uC	0105h	W	2-4
SETUP_ID1, EISA	zC80h	R	2-5
SETUP_ID1, uC	0100h	R	2-3
SETUP_ID2, EISA	zC81h	R	2-5
SETUP_ID2, uC	0101h	R	2-3
SETUP_ID3, EISA	zC82h	R	2-5
SETUP_ID4, EISA	zC83h	R	2-6
SETUP_OPT, EISA	zC84h	R/W	2-6
SETUP_OPT, uC	0102h	R/W	2-3
SHADOW_CTL	46EEh	W	9-6
SHADOW_SET	5AEEh	W	9-7
SHORT_STROKE	9EE8h	W	8-51
SRC_X/DEST_X/DIASTP	8EE8h	W	8-52
SRC_X_END	BEEEh	W	9-62
SRC_X_START	B2EEh	W	9-62
SRC_Y/DEST_Y/AXSTP	8AE8h	W	8-53
SRC_Y_DIR	C2EEh	W	9-62
SUBSYS_CNTL	42E8h	W	8-18

Register	Port	Attributes	Page
SUBSYS_STATUS	42E8h	R	8-20
V_DISP	16E8h	W	8-10
V_SYNC_STRT	1AE8h	W	8-11
V_SYNC_WID	1EE8h	W	8-12
V_TOTAL	12E8h	W	8-12
VERT_CURSOR_OFFSET	1EEFh	W	9-81
VERT_CURSOR_POSN	16EEh	W	9-80
VERT_LINE_CNTR	CEEEh	R	9-7
VERT_OVERSCAN	66EEh	W	9-72
WRT_MASK	AAE8h	W	8-54

Port	Register	Attributes	Page
0100h	SETUP_ID1, uC	R	2-3
0101h	SETUP_ID2, uC	R	2-3
0102h	SETUP_OPT, uC	R/W	2-3
0103h	ROM_SETUP, uC	W	2-4
0104h	SETUP_1, uC	W	2-4
0105h	SETUP_2, uC	W	2-4
02E8h	DISP_STATUS	R	8-8
02E8h	H_TOTAL	W	8-9
02EAh	DAC_MASK	R/W	8-2
02EBh	DAC_R_INDEX	R/W	8-2
02ECh	DAC_W_INDEX	R/W	8-3
02EDh	DAC_DATA	R/W	8-2
02EEh	OVERSCAN_COLOR_8	W	9-72
02EFh	OVERSCAN_BLUE_24	W	9-72
06E8h	H_DISP	W	8-8
06EEh	OVERSCAN_GREEN_24	W	9-73
06EFh	OVERSCAN_RED_24	W	9-73
0AE8h	H_SYNC_STRT	W	8-9
0AEEh	CURSOR_OFFSET_LO	W	9-79
0EE8h	H_SYNC_WID	W	8-9
0EEEh	CURSOR_OFFSET_HI	W	9-79
12E8h	V_TOTAL	W	8-12
12EEh	CONFIG_STATUS_1 (mach8)	R	9-64
12EEh	CONFIG_STATUS_1 (mach32)	R	9-65
12EEh	HORZ_CURSOR_POSN	W	9-79
16E8h	V_DISP	W	8-10
16EEh	CONFIG_STATUS_2 (mach8)	R	9-66
16EEh	CONFIG_STATUS_2 (mach32)	R	9-67
16EEh	VERT_CURSOR_POSN	W	9-80
1AE8h	V_SYNC_STRT	W	8-11
1AEEh	FIFO_TEST_DATA	R	9-85
1AEEh	CURSOR_COLOR_0	W	9-80
1AEFh	CURSOR_COLOR_1	W	9-80
1EE8h	V_SYNC_WID	W	8-12
1EEEh	HORZ_CURSOR_OFFSET	W	9-81
1EEFh	VERT_CURSOR_OFFSET	W	9-81
22E8h	DISP_CNTL	W	8-7
22EEh	DAC_CONT (PCI)	R/W	9-89
26EEh	CRT_PITCH	W	9-5
2AEEh	CRT_OFFSET_LO	W	9-5
2EEEh	CRT_OFFSET_HI	W	9-5
32EEh	LOCAL_CNTL	R/W	9-83

Port	Register	Attributes	Page
36EEh	MISC_OPTIONS	R/W	9-12
36EEh	FIFO_OPT	W	9-9
3AEEh	FIFO_TEST_TAG	R	9-85
3AEEh	EXT_CURSOR_COLOR_0	W	9-82
3EEEh	EXT_CURSOR_COLOR_1	W	9-82
42E8h	SUBSYS_STATUS	R	8-20
42E8h	SUBSYS_CNTL	W	8-18
42EEh	MEM_BNDRY	R/W	9-74
46E8h	GENENA (Add-On)	W	5-8
46EEh	SHADOW_CTL	W	9-6
4AE8h	ADVFUNC_CNTL	W	8-6
4AEEh	CLOCK_SEL	R/W	9-4
52EEh	SCRATCH_PAD_0	R/W	9-88
56EEh	SCRATCH_PAD_1	R/W	9-88
5AEEh	SHADOW_SET	W	9-7
5EEEh	MEM_CFG	R/W	9-75
62EEh	EXT_GE_STATUS	R	9-68
62EEh	HORZ_OVERSCAN	W	9-71
66EEh	VERT_OVERSCAN	W	9-72
6AEEh	MAX_WAITSTATES (mach8)	R/W	9-10
6AEEh	MAX_WAITSTATES (mach32)	R/W	9-11
6AEEh	MISC_CONT (PCI)	R/W	9-76
6EEEh	GE_OFFSET_LO	W	9-21
72EEh	BOUNDS_LEFT	R	9-48
72EEh	GE_OFFSET_HI	W	9-21
76EEh	BOUNDS_TOP	R	9-48
76EEh	GE_PITCH	W	9-20
7AEEh	BOUNDS_RIGHT	R	9-48
7AEEh	EXT_GE_CONFIG (mach8 8-bit)	W	9-17
7AEEh	EXT_GE_CONFIG (mach8 16-bit)	W	9-18
7AEEh	EXT_GE_CONFIG (mach32)	W	9-19
7EEEh	BOUNDS_BOTTOM	R	9-48
7EEEh	MISC_CNTL	W	9-84
82E8h	CUR_Y	R/W	8-38
82EEh	PATT_DATA_INDEX	R/W	9-59
86E8h	CUR_X	R/W	8-37
8AE8h	SRC_Y/DEST_Y/AXSTP	W	8-53
8EE8h	SRC_X/DEST_X/DIASTP	W	8-52
8EEEh	R_EXT_GE_CONFIG	R	9-24
8EEEh	PATT_DATA	W	9-58
92E8h	ERR_TERM	R/W	8-39
92EEh	R_MISC_CNTL	R	9-84
96E8h	MAJ_AXIS_PCNT	W	8-42

Port	Register	Attributes	Page
96EEh	BRES_COUNT	R/W	9-49
9AE8h	GE_STAT	R	8-41
9AE8h	CMD	W	8-28
9AEEh	EXT_FIFO_STATUS	R	9-16
9AEEh	LINEDRAW_INDEX	W	9-57
9EE8h	SHORT_STROKE	W	8-51
A2E8h	BKGD_COLOR	W	8-27
A2EEh	LINEDRAW_OPT	R/W	9-22
A6E8h	FRGD_COLOR	W	8-40
A6EEh	DEST_X_START	W	9-52
AAE8h	WRT_MASK	W	8-54
AAEEh	DEST_X_END	W	9-52
AEE8h	RD_MASK	W	8-48
AEEEh	DEST_Y_END	W	9-52
B2E8h	CMP_COLOR	W	8-36
B2EEh	R_H_TOTAL&DISP	R	9-85
B2EEh	SRC_X_START	W	9-62
B6E8h	BKGD_MIX	W	8-27
B6EEh	R_H_SYNC_STRT	R	9-86
B6EEh	ALU_BG_FN	W	9-47
BAE8h	FRGD_MIX	W	8-40
BAEEh	R_H_SYNC_WID	R	9-86
BAEEh	ALU_FG_FN	W	9-47
BEE8*0h	MIN_AXIS_PCNT	W	8-43
BEE8*1h	SCISSOR_T	W	8-50
BEE8*2h	SCISSOR_L	W	8-49
BEE8*3h	SCISSOR_B	W	8-49
BEE8*4h	SCISSOR_R	W	8-50
BEE8*5h	MEM_CNTL	W	8-17
BEE8*8h	PATTERN_L	W	8-45
BEE8*9h	PATTERN_H	W	8-44
BEE8*Ah	PIXEL_CNTL	W	8-46
BEEEh	SRC_X_END	W	9-62
C2EEh	R_V_TOTAL	R	9-86
C2EEh	SRC_Y_DIR	W	9-62
C6EEh	R_V_DISP	R	9-87
C6EEh	EXT_SHORT_STROKE	W	9-55
CAEEh	R_V_SYNC_STRT	R	9-87
CAEEh	SCAN_X	W	9-61
CEEEh	VERT_LINE_CNTR	R	9-7
CEEEh	DP_CONFIG	W	9-15
D2EEh	R_V_SYNC_WID	R	9-87
D2EEh	PATT_LENGTH	W	9-60

Port	Register	Attributes	Page
D6EEh	PATT_INDEX	W	9-59
DAEEh	R_SRC_X	R	9-60
DAEEh	EXT_SCISSOR_L	W	9-53
DEEEh	R_SRC_Y	R	9-61
DEEEh	EXT_SCISSOR_T	W	9-54
E2E8h	PIX_TRANS	R/W	8-47
E2EEh	EXT_SCISSOR_R	W	9-54
E6EEh	EXT_SCISSOR_B	W	9-53
EEEEh	DEST_CMP_FN	W	9-50
F2EEh	DEST_COLOR_CMP_MASK	W	9-51
FAEEh	CHIP_ID	R	9-88
FEEEh	LINEDRAW	W	9-56
zC80h	SETUP_ID1, EISA	R	2-5
zC81h	SETUP_ID2, EISA	R	2-5
zC82h	SETUP_ID3, EISA	R	2-5
zC83h	SETUP_ID4, EISA	R	2-6
zC84h	SETUP_OPT, EISA	R/W	2-6
zC85h	ROM_SETUP, EISA	W	2-6
zC86h	SETUP_1, EISA	W	2-7
zC87h	SETUP_2, EISA	W	2-7

Memory Mapped Registers

Memory mapping is available only in the following ***mach32*** accelerators: ATI68800LX, ATI68800-6, and ATI68800AX.

The register space of the accelerator is mapped onto the 128 dwords at the end of the linear aperture. IBM compatible registers are mapped onto the lower 64 dwords; ATI-extended registers follow directly above them in the top 64 dwords. The exact mapping places register 2E8 first, at offset FFE00 from the base of the aperture. Register 6E8 follows at FFE04 and so on up to register FEEE at offset FFFFC. See illustration and tables below. Each 16-bit register maps into the lower 16 bits of a 32-bit dword. The upper 16 bits, i.e., the shaded region in the illustration are reserved for future use.

For configurations with 4MB on-board video memory, note that the memory at the top of the aperture is otherwise not accessible when memory mapped registers are enabled.

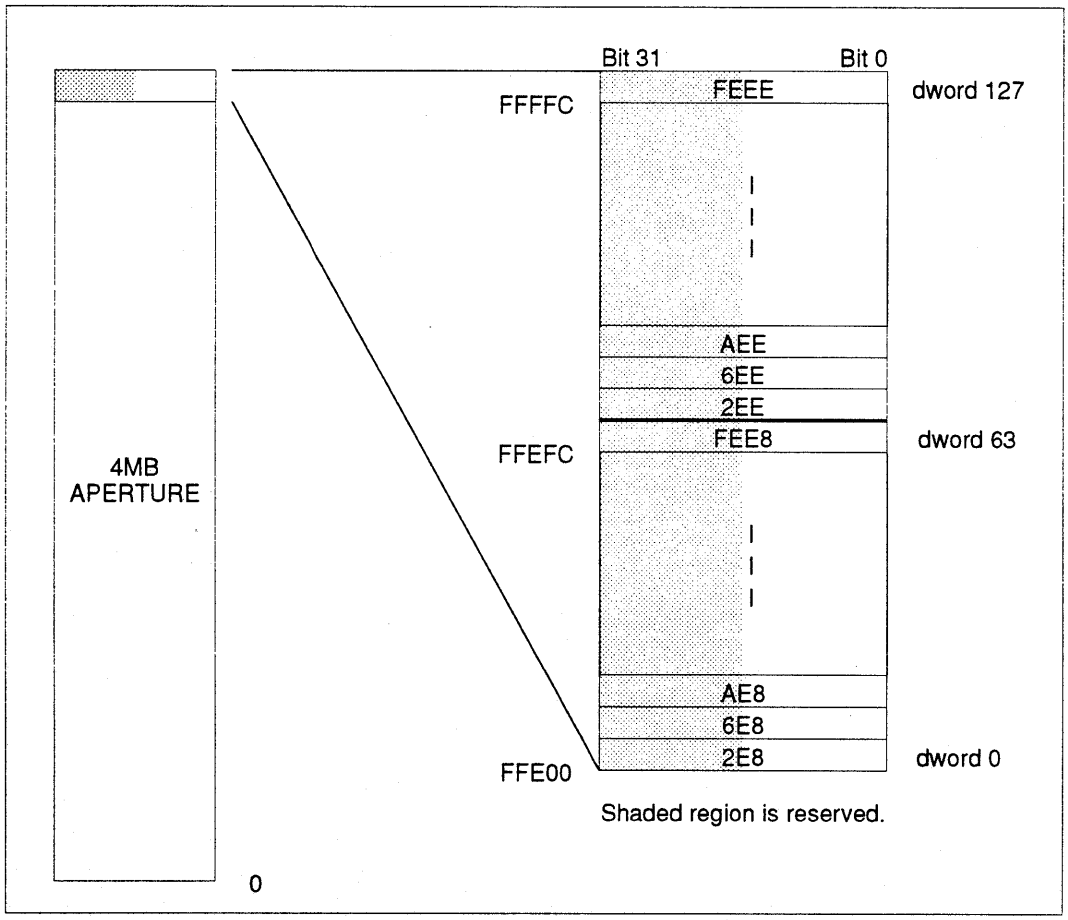


Figure A-1. Memory Mapped Registers

Memory Mapping of IBM-Compatible Registers

I/O Address	Memory Offset	Register Name
2E8	FFE00	DISP_STATUS; H_TOTAL
6E8	FFE04	H_DISP
AE8	FFE08	H_SYNC_STRT
EE8	FFE0C	H_SYNC_WID
12E8	FFE10	V_TOTAL
16E8	FFE14	V_DISP
1AE8	FFE18	V_SYNC_STRT
1EE8	FFE1C	V_SYNC_WID
22E8	FFE20	DISP_CNTL
26E8	FFE24	
2AE8	FFE28	
2EE8	FFE2C	
32E8	FFE30	
36E8	FFE34	
3AE8	FFE38	
3EE8	FFE3C	
42E8	FFE40	SUBSYS_STATUS; SUBSYS_CNTL
46E8	FFE44	ROM_PAGE_SEL
4AE8	FFE48	ADVFUNC_CNTL
4EE8	FFE4C	
52E8	FFE50	
56E8	FFE54	
5AE8	FFE58	
5EE8	FFE5C	
62E8	FFE60	
66E8	FFE64	
6AE8	FFE68	
6EE8	FFE6C	
72E8	FFE70	
76E8	FFE74	
7AE8	FFE78	
7EE8	FFE7C	

I/O Address	Memory Offset	Register Name
82E8	FFE80	CUR_Y
86E8	FFE84	CUR_X
8AE8	FFE88	SRC_Y/DEST_Y/AXSTP
8EE8	FFE8C	SRC_X/DEST_X/DOASTP
92E8	FFE90	ERR_TERM
96E8	FFE94	MAJ_AXIS_PCNT
9AE8	FFE98	GE_STAT; CMD
9EE8	FFE9C	SHORT_STROKE
A2E8	FFEA0	BKGD_COLOR
A6E8	FFEA4	FRGD_COLOR
AAE8	FFEA8	WRT_MASK
AEE8	FFEAC	RD_MASK
B2E8	FFEB0	CMP_COLOR
B6E8	FFEB4	BKGD_MIX
BAE8	FFEB8	FRGD_MIX
BEE8	FFEBC	Multi-Func Control
C2E8	FFEC0	
C6E8	FFEC4	
CAE8	FFEC8	
CEE8	FFECC	
D2E8	FFED0	
D6E8	FFED4	
DAE8	FFED8	
DEE8	FFEDC	
E2E8	FFEE0	PIX_TRANS
E6E8	FFEE4	
EAE8	FFEE8	
EEE8	FFEEC	
F2E8	FFEF0	
F6E8	FFEF4	
FAE8	FFEF8	
FEE8	FFEFC	

Memory Mapping of ATI-Extended Registers

I/O Address	Memory Offset	Register Name
2EE	FFF00	OVERSCAN_COLOR_[24,8]
6EE	FFF04	OVERSCAN_[RED,GREEN]_24
AEE	FFF08	CURSOR_OFFSET_LO
EEE	FFF0C	CURSOR_OFFSET_HI
12EE	FFF10	CONFIG_STATUS_1
16EE	FFF14	CONFIG_STATUS_2; VERT_CURSOR_POSN
1AEE	FFF18	FIFO_TEST_DATA; CURSOR_COLOR_0
1EEE	FFF1C	[VERT/HORZ]_CURSOR_OFFSET
22EE	FFF20	DAC_CONT (PCI)
26EE	FFF24	CRT_PITCH
2AEE	FFF28	CRT_OFFSET_LO
2EEE	FFF2C	CRT_OFFSET_HI
32EE	FFF30	LOCAL_CONTROL
36EE	FFF34	MISC_OPTIONS
3AEE	FFF38	FIFO_TEST_TAG; EXT_CURSOR_COLOR_0
3EEE	FFF3C	EXT_CURSOR_COLOR_1
42EE	FFF40	MEM_BNDRY
46EE	FFF44	SHADOW_CTL
4AEE	FFF48	CLOCK_SEL
4EEE	FFF4C	
52EE	FFF50	SCRATCH_PAD_0
56EE	FFF54	SCRATCH_PAD_1
5AEE	FFF58	SHADOW_SET
5EEE	FFF5C	MEM_CFG
62EE	FFF60	EXT_GE_STATUS; HORZ_OVERSCAN; SYNC (PCI)
66EE	FFF64	VERT_OVERSCAN
6AEE	FFF68	MAX_WAITSTATES; MISC_CONT
6EEE	FFF6C	GE_OFFSET_LO
72EE	FFF70	BOUNDS_LEFT; GE_OFFSET_HI
76EE	FFF74	BOUNDS_TOP; GE_PITCH
7AEE	FFF78	BOUNDS_RIGHT; EXT_GE_CONFIG
7EEE	FFF7C	BOUNDS_BOTTOM; MISC_CNTL

I/O Address	Memory Offset	Register Name
82EE	FFF80	PATT_DATA_INDEX
86EE	FFF84	
8AEE	FFF88	
8EEE	FFF8C	R_EXT_GE_CONFIG; PATT_DATA
92EE	FFF90	R_MISC_CNTL
96EE	FFF94	BRES_COUNT
9AEE	FFF98	EXT_FIFO_STATUS; LINEDRAW_INDEX
9EEE	FFF9C	
A2EE	FFFA0	LINEDRAW_OPT
A6EE	FFFA4	DEST_X_START
AAEE	FFFA8	DEST_X_END
AEEE	FFFAC	DEST_Y_END
B2EE	FFFB0	R_H_TOTAL&DISP; SRC_X_START
B6EE	FFFB4	R_H_SYNC_STRT; ALU_BG_FN
BAEE	FFFB8	R_H_SYNC_WID; ALU_FG_FN
BEEE	FFFBC	SRC_X_END
C2EE	FFFC0	R_V_TOTAL; SRC_Y_DIR
C6EE	FFFC4	R_V_DISP; EXT_SHORT_STROKE
CAEE	FFFC8	R_V_SYNC_STRT; SCAN_X
CEEE	FFFCC	VERT_LINE_CNTR; DP_CONFIG
D2EE	FFFD0	R_V_SYNC_WID; PATT_LENGTH
D6EE	FFFD4	PATT_INDEX
DAEE	FFFD8	R_SRC_X; EXT_SCISSOR_L
DEEE	FFFDC	R_SRC_Y; EXT_SCISSOR_T
E2EE	FFFE0	EXT_SCISSOR_R
E6EE	FFFE4	EXT_SCISSOR_B
EAEE	FFFE8	
EEEE	FFFEC	DEST_CMP_FN
F2EE	FFFF0	DEST_COLOR_CMP_MASK
F6EE	FFFF4	
FAEE	FFFF8	CHIP_ID
FEEE	FFFFC	LINEDRAW

B

BIOS Interface

The base ROM address is determined by bits 0-6 of register 52EEh. This address in the ***mach32*** is a scratch pad register. However, the boot ROM will set it up to contain the proper address, which is calculated as follows:

```
ROM_BASE_SEGMENT = (BYTE_VALUE_IN_52EE & 0x7F) * 0x80 + 0xC000
```

ROM services are accessible by absolute calls at these addresses:

```
xxxx:0064 Load Shadow Set
        ah = 0h, 8 bit per pixel (existing drivers must load ah=0)
        ah = DEEP_COLOR_OPTIONS
           (0)    : '1'
           (2-1)  : PIXEL_WIDTH, 0=4bpp, 1=8bpp, 2=16bpp, 3=24bpp
           (3-4)  : 16_BIT_COLOR_MODE, 0=555, 1=565, 2=655, 3=664
           (5)    : unused
           (6)    : 24_BIT_COLOR_CONFIG, 0=3 bytes/pixel, 1=4 bytes/pixel
           (7)    : 24_BIT_COLOR_ORDER, 0=RGB, 1=BGR
        al = 0h, default (shadow set 1=640x480x8, 2=1024x768x8)
        al = 1h, load 800x600 into shadow set 1 (lores)
        al = 2h, load 800x600 into shadow set 2 (hires)
        al = 11h, load 1280x1024 into shadow set 1 (lores)
        al = 12h, load 1280x1024 into shadow set 2 (hires)
        al = 21h, load 640x480 into shadow set 1 (lores)
        al = 22h, load 640x480 into shadow set 2 (hires)
        al = 41h, load 1024x768 into shadow set 1 (lores)
        al = 42h, load 1024x768 into shadow set 2 (hires)
        al = 81h, load alternate mode into shadow set 1 (lores)
        al = 82h, load alternate mode into shadow set 1 (hires)
        bx = PITCH, 0=1024, 1=pitch of mode rounded up to next 128,
                 2=1536, 3=do not modify pitch

xxxx:0068 Set Mode
        al = 0h, VGA passthrough mode
        al = 1h, lores mode
        al = 2h, hires mode
```

```
xxxx:006C Query
        al = 0,  Query information structure size in bytes
                 Returns size in ax
        al = 1,  Query device long
                 es:bx points to structure to fill in
                 (see Query structure on next page)
        al = 2,  Query device short
                 Returns:
                 al = ASIC revision
                 ah = aperture configuration
                 bx = aperture address
                 cl = mouse configuration
                 ch = DAC type
xxxx:0070 Accelerator services
        ah = 0, Reset accelerator
        ah = 1, Set DAC to default colors
        ah = 2, EEPROM services
                 al = 0, Read from EEPROM (returns result in ax)
                 al = 1, Write to EEPROM
                 bx = EEPROM word address
                 dx = Data to write
```

Query structure

DEVICE STATUS TABLE

Byte	Description
0:1h	Size of Structure, in bytes
2h	Revision of Structure
3h	Number of Mode Tables
4:5h	Offset to Mode Table, in bytes
6h	Size of Mode Table, in bytes
7h	ASIC Revision
8h	Status Flags Bit 0 = host data transfers forced to 8-bit
9h	VGA Type: 0 = Disabled 1 = Enabled
Ah	VGA Boundary: 0 = 0k 1 = 256k 2 = 512k 3 = 768k 4 = 1M 0FFh = Full access
Bh	Memory Size: 0 = 256k 1 = 512k 2 = 1M 3 = 2M 4 = 4M
Ch	DAC Type: 0 = ATI68830 1 = SC11483/6/8 2 = ATI34075 3 = Bt476/8 4 = Bt481/2 5 = ATI68860 (ATI68800AX only) See Appendix F for further details.
Dh	Memory Type: 0 = 256Kx4 DRAM 1 = 256Kx4 VRAM 2 = 256Kx4 VRAM 3 = 256Kx16 DRAM See 12EE-R[6:4] on page 9-65 for further details.

Byte	Description
Eh	Bus Type: 0 = 16-Bit ISA 1 = EISA 2 = 16-Bit Micro Channel 3 = 32-Bit Micro Channel 4 = 386SX Local Bus 5 = 386DX Local Bus 6 = 486DX Local Bus 7 = Reserved See 12EE-R[3:1] on page 9-65 for further details.
Fh	Monitor Alias
10:11h	Shadow 1 Status
12:13h	Shadow 2 Status
14:15h	Aperture Address (0MB to 4095MB)
16h	Aperture Configuration: Bits 1:0: 0 = Disabled 1 = 1MB Aperture 2 = 4MB Aperture
17h	Mouse Configuration: Bits 1:0: MOUSE_ADDR_SEL 0 = Disabled 1 = Secondary Address Selected 2 = Primary Address Selected Bits 3:2: INT_HANDLER_SEL 0 = IRQ5 1 = IRQ4 2 = IRQ3 3 = IRQ2
18h	DAC Support **Bit Description** 0 664 1 655 2 565 3 555 4 Reserved 5 32bpp 6 BGR 7 RGB

Mode tables are located immediately following the Device Status Table, starting at byte 19h. A forward pointer should be used to reference the mode tables since these tables may be expanded in the future. A query typically returns two to seven mode tables; but it is also possible none is returned.

MODE TABLE

Offset	Description
0:1	Horizontal display resolution in pixels
2:3	Vertical display resolution in pixels
4	Maximum pixel depth
5	Status Flags: Bit 0: **1** = Non-linear Y addressing mode Bit 6: **1** = MUX mode Bit 7: **1** = PCLK is divided by 2
6:9	Reserved
A:B	CLOCK_SELECT
C	H_TOTAL
D	H_DISP
E	H_SYNC_STRT
F	DISP_CNTL
10	H_SYNC_WID
11	V_SYNC_WID
12:13	V_TOTAL
14:15	V_DISP
16:17	V_SYNC_STRT
18:19	HORIZONTAL_OVERSCAN
1A:1B	VERTICAL_OVERSCAN
1C	OVERSCAN_COLOR_8
1D	OVERSCAN_COLOR_BLUE
1E	OVERSCAN_COLOR_GREEN
1F	OVERSCAN_COLOR_RED

This page intentionally left blank.

C

EEPROM Map

EEPROM Map		
WORD	**BITS**	**PARAMETERS**
00h	15:0	EEPROM_WRITE_COUNTER
01h	15:8	MOUSE_ADDR_SEL 00h = Mouse disabled 08h = Secondary address selected 18h = Primary address selected
	7:0	INT_HANDLER_SEL 20h = IRQ5 28h = IRQ4 30h = IRQ3 38h = IRQ2
02h	15:8	POWER_UP_VIDEO_MODE 03h = VGA color - secondary 05h = VGA monochrome - secondary 07h = VGA lores color - primary 09h = VGA hires color - primary 0bh = VGA monochrome - primary 12h = EGA lores color - secondary 13h = EGA hires color - secondary 15h = EGA monochrome - secondary 17h = EGA lores color - primary 19h = EGA hires color - primary 1bh = EGA monochrome - primary 20h = CGA 30h = Hercules 720x348 40h = Hercules 640x400
	7:6	MONO_COLOR_SEL 0 = White 1 = Green 2 = Amber

EEPROM Map		
WORD	**BITS**	**PARAMETERS**
02h	5 4 3 2 1 0	DUAL_MONITOR_ENA POWER_UP_FONT 0 = 8x14 or 9x14 1 = 8x16 or 9x16 VGA Bus I/O 0 = 8 bits 1 = 16 bits ZERO_WSTATE_RAM_ENA 0 = disable 1 = enable ZERO_WSTATE_ROM_ENA 0 = disable 1 = enable ROM_16_ENA 0 = disable 1 = enable
03h	15 14 13:2 3:0	Scrolling Fix enable Korean BIOS support Reserved EEPROM table revision number
04h	15:0	CUSTOM_MONITOR_INDICES
05h	15:14 13:8 7 6:4 3 2:0	HOST_DATA_TRANSFER_WIDTH 0 = Auto select 1 = 16 bit 2 = 8 bit 3 = 8 bit host data/16 bit other MONITOR_CODE Reserved VGA Boundary 0 = Shared 1 = 256K 2 = 512K 4 = 1MB ALIAS_ENA MONITOR_ALIAS
06h	15:4 3:0	APERTURE_LOCATION in Mbytes APERTURE_SIZE 0 = disabled 1 = 1M 2 = 4M

EEPROM Map		
WORD	**BITS**	**PARAMETERS**
07h	15:8 7:2 1 0	Offset to 640x480 table in words Reserved USE_STORED_PARMS_FOR_640_480 640_480_72HZ_ENA
08h	15:8 7 6 5 4 3 2 1 0	Offset to 800x600 table in words USE_STORED_PARMS_FOR_800x600 Reserved 800_600_72Hz_ENA 800_600_70Hz_ENA 800_600_60Hz_ENA 800_600_56Hz_ENA 800_600_89Hz_ENA 800_600_95Hz_ENA
09h	15:8 7 6:5 4 3 2 1 0	Offset to 1024x768 table in words USE_STORED_PARMS_FOR_1024x768 Reserved 1024_768_66Hz_ENA (not active) 1024_768_72Hz_ENA 1024_768_70Hz_ENA 1024_768_60Hz_ENA 1024_768_87Hz_ENA
0Ah	15:8 7 6:2 1 0	Offset to 1280x1024 table in words USE_STORED_PARMS_FOR_1280x1024 Reserved 1280_1024_95Hz_ENA 1280_1024_87Hz_ENA
0Bh	15:8 7 6:2 1 0	Offset to alternate mode table in words USE_STORED_PARMS_FOR_ALTERNATE Reserved 1152x900 1120x750
0Ch	15:0	Reserved
0D:1Bh	-	CRT Parameter Table 1, description starts on page C-5
1C:2Ah	-	CRT Parameter Table 2, description starts on page C-5
2B:39h	-	CRT Parameter Table 3, description starts on page C-5
3A:48h	-	CRT Parameter Table 4, description starts on page C-5
49:57h	-	CRT Parameter Table 5, description starts on page C-5

EEPROM Map		
WORD	**BITS**	**PARAMETERS**
58:66h	-	CRT Parameter Table 6, description starts on page C-5
67:75h	-	CRT Parameter Table 7, description starts on page C-5
76:7Dh	-	Reserved
7Eh	15 14 13:11 10:8 7:0	Reserved VGA ENABLE MEMORY SIZE - see Query Structure, byte B, page B-3 DAC TYPE - see Query Structure, byte C, page B-3 Reserved
7Fh	15:0	EEPROM Checksum

CRT Parameter Table x*			
OFFSET	**BITS**	**PARAMETERS**	
0	15 14 13 12 11:9 8 7 6 5:4 3:0	VERTICAL_SYNC_POLARITY HORIZONTAL_SYNC_POLARITY Interlace MUX_MODE MAXIMUM_PIXEL_DEPTH 0 = 8 bits/pixel 1 = 16 bits/pixel 2 = 24 bits/pixel PARM_TYPE 0 = VGA parameters to follow 1 = 8514 parameters to follow DOT_CLOCK_SEL 0 = Use default dot clock 1 = Use user supplied dot clock (not active) CRTC_USAGE 0 = Use sync polarities only 1 = Use all CRTC parms in EEPROM CLOCK_DIV CLOCK_CHIP_SEL	
		VGA Parameters	8514 Parameters
1	15:8 7: 0	VIDEO_MODE_SEL_1 VIDEO_MODE_SEL_2	Reserved Reserved
2	15:8 7: 0	VIDEO_MODE_SEL_3 VIDEO_MODE_SEL_4	VFIFO_24 VFIFO_16
3	15:8 7: 0	H_TOTAL (CRT00) V_TOTAL (CRT06)	H_TOTAL H_DISP
4	15:8 7: 0	H_RETRACE_STRT (CRT04) H_RETRACE_END (CRT05)	H_SYNC_STRT H_SYNC_WID
5	15:8 7: 0	V_RETRACE_STRT (CRT10) V_RETRACE_END (CRT11)	V_TOTAL (15:8) V_TOTAL (7:0)
6	15:8 7: 0	H_BLANK_STRT (CRT02) H_BLANK_END (CRT03)	V_DISP (15:8) V_DISP (7:0)
7	15:8 7: 0	V_BLANK_STRT (CRT15) V_BLANK_END (CRT16)	V_SYNC_STRT (15:8) V_SYNC_STRT (7:0)
8	15:8 7: 0	CRT_OVERFLOW (CRT07) MAX_SCANLINE (CRT09)	V_SYNC_WID DISP_CNTL

CRT Parameter Table x*		
OFFSET	**BITS**	**PARAMETERS**
9	15:8 7: 0	V_DISPLAYED (CRT12) / CLOCK_SEL (15:8) CRT_MODE (CRT17) / CLOCK_SEL (7:0)
10	15 14 13:8 7:0	PCLK/2 flag MUX flag SIZEOF_MODE_TABLE in words Offset to alternate table
11	15:0	HORIZONTAL_OVERSCAN
12	15:0	VERTICAL_OVERSCAN
13	15:8 7:0	OVERSCAN_COLOR_BLUE OVERSCAN_COLOR_8
14	15:8 7:0	OVERSCAN_COLOR_RED OVERSCAN_COLOR_GREEN

*The ***mach32*** EEPROM has seven CRT parameter tables, starting at words 0Dh, 1Ch, 2Bh, 3Ah, 49h, 58h, and 67h.

D

CRT Parameters

640x480 60Hz Non-interlaced

H_TOTAL = 0x63
H_DISP = 0x4F
H_SYNC_STRT = 0x52
H_SYNC_WID = 0x2C
DISP_CNTL = 0x23
CLOCK_SEL = 0x50
DOT_CLOCK = 25.125MHz

V_TOTAL = 0x418
V_DISP = 0x3BF
V_SYNC_STRT = 0x3D6
V_SYNC_WID = 0x22

	Horizontal		Vertical	
Resolution	640		480	
Scan Frequency	31.406KHz		59.82Hz	
Polarity	(-)		(-)	
Sync Width	3.821us	12 chars	0.064ms	2 lines
Front Porch	0.637us	2 chars	0.350ms	11 lines
Back Porch	1.910us	6 chars	1.019ms	32 lines
Active Time	25.473us	80 chars	15.284ms	480 lines
Blank Time	6.368us	20 chars	1.433ms	45 lines

640x480 72Hz Non-interlaced

H_TOTAL = 0x69
H_DISP = 0x4F
H_SYNC_STRT = 0x52
H_SYNC_WID = 0x25
DISP_CNTL = 0x23
CLOCK_SEL = 0x24
DOT_CLOCK = 32.00MHz

V_TOTAL = 0x40B
V_DISP = 0x3BF
V_SYNC_STRT = 0x3D0
V_SYNC_WID = 0x23

	Horizontal		Vertical	
Resolution	640		480	
Scan Frequency	37.736KHz		72.57Hz	
Polarity	(-)		(-)	
Sync Width	1.250us	5 chars	0.080ms	3 lines
Front Porch	0.750us	3 chars	0.239ms	9 lines
Back Porch	4.500us	18 chars	0.742ms	28 lines
Active Time	20.000us	80 chars	12.720ms	480 lines
Blank Time	6.500us	26 chars	1.060ms	40 lines

640x480 72Hz Non-interlaced (Alternate)

H_TOTAL = 0x70
H_DISP = 0x4F
H_SYNC_STRT = 0x57
H_SYNC_WID = 0x30
DISP_CNTL = 0x23
CLOCK_SEL = 0x6C
DOT_CLOCK = 40.00MHz

V_TOTAL = 0x4CA
V_DISP = 0x3BF
V_SYNC_STRT = 0x421
V_SYNC_WID = 0x2C

	Horizontal		Vertical	
Resolution	640		480	
Scan Frequency	44.148KHz		71.79Hz	
Polarity	(-)		(-)	
Sync Width	3.207us	16 chars	0.272ms	12 lines
Front Porch	1.604us	8 chars	1.133ms	50 lines
Back Porch	1.804us	9 chars	1.654ms	73 lines
Active Time	16.036us	80 chars	10.872ms	480 lines
Blank Time	6.615us	33 chars	3.058ms	135 lines

800x600 89Hz Interlaced

H_TOTAL = 0x84
H_DISP = 0x63
H_SYNC_STRT = 0x6E
H_SYNC_WID = 0x10
DISP_CNTL = 0x33
CLOCK_SEL = 0x7C
DOT_CLOCK = 32.50MHz

V_TOTAL = 0x580
V_DISP = 0x4AB
V_SYNC_STRT = 0x4C2
V_SYNC_WID = 0x2C

	Horizontal		Vertical	
Resolution	800		600	
Scan Frequency	31.492KHz		89.72Hz	
Polarity	(+)		(-)	
Sync Width	0.985us	4 chars	0.191ms	12 lines
Front Porch	0.492us	2 chars	0.175ms	11 lines
Back Porch	5.662us	23 chars	1.254ms	79 lines
Active Time	24.615us	100 chars	9.526ms	600 lines
Blank Time	7.138us	29 chars	1.619ms	102 lines

800x600 95Hz Interlaced

H_TOTAL = 0x84
H_DISP = 0x63
H_SYNC_STRT = 0x6D
H_SYNC_WID = 0x10
DISP_CNTL = 0x33
CLOCK_SEL = 0xC
DOT_CLOCK = 36.00MHz

V_TOTAL = 0x580
V_DISP = 0x4AB
V_SYNC_STRT = 0x4C2
V_SYNC_WID = 0xC

	Horizontal		Vertical	
Resolution	800		600	
Scan Frequency	33.835KHz		96.39Hz	
Polarity	(+)		(+)	
Sync Width	3.556us	16 chars	0.177ms	12 lines
Front Porch	2.222us	10 chars	0.163ms	11 lines
Back Porch	1.556us	7 chars	1.167ms	79 lines
Active Time	22.222us	100 chars	8.867ms	600 lines
Blank Time	7.333us	33 chars	1.507ms	102 lines

800x600 56Hz Non-interlaced

H_TOTAL = 0x7F
H_DISP = 0x63
H_SYNC_STRT = 0x66
H_SYNC_WID = 0x9
DISP_CNTL = 0x23
CLOCK_SEL = 0xC
DOT_CLOCK = 36.00MHz

V_TOTAL = 0x4E0
V_DISP = 0x4AB
V_SYNC_STRT = 0x4B0
V_SYNC_WID = 0x2

	Horizontal		Vertical	
Resolution	800		600	
Scan Frequency	35.156KHz		56.25Hz	
Polarity	(+)		(+)	
Sync Width	2.000us	9 chars	0.057ms	2 lines
Front Porch	0.667us	3 chars	0.028ms	1 lines
Back Porch	3.556us	16 chars	0.626ms	22 lines
Active Time	22.222us	100 chars	17.067ms	600 lines
Blank Time	6.222us	28 chars	0.711ms	25 lines

800x600 60Hz Non-interlaced

H_TOTAL = 0x83
H_DISP = 0x63
H_SYNC_STRT = 0x68
H_SYNC_WID = 0x10
DISP_CNTL = 0x23
CLOCK_SEL = 0x30
DOT_CLOCK = 40.00MHz

V_TOTAL = 0x4E3
V_DISP = 0x4AB
V_SYNC_STRT = 0x4B3
V_SYNC_WID = 0x4

	Horizontal		Vertical	
Resolution	800		600	
Scan Frequency	37.879KHz		60.32Hz	
Polarity	(+)		(+)	
Sync Width	3.200us	16 chars	0.106ms	4 lines
Front Porch	1.000us	5 chars	0.026ms	1 lines
Back Porch	2.200us	11 chars	0.607ms	23 lines
Active Time	20.000us	100 chars	15.840ms	600 lines
Blank Time	6.400us	32 chars	0.739ms	28 lines

800x600 70Hz Non-interlaced

H_TOTAL = 0x7d
H_SYNC_STRT = 0x64
V_TOTAL = 0x4f3
V_SYNC_STRT = 0x4c0
DISP_CNTL = 0x23
DOT_CLOCK = 44.90MHz

H_DISP = 0x63
H_SYNC_WID = 0x12
V_DISP = 0x4ab
V_SYNC_WID = 0x2c
CLOCK_SEL = 0x1c

	Horizontal		Vertical	
Resolution	800		600	
Scan Frequency	44.544KHz		70.04Hz	
Polarity	(+)		(+)	
Sync Width	3.207us	18 chars	0.269ms	12 lines
Front Porch	0.535us	3 chars	0.202ms	9 lines
Back Porch	0.891us	5 chars	0.337ms	15 lines
Active Time	17.817us	100 chars	13.470ms	600 lines
Blank Time	4.633us	26 chars	0.808ms	36 lines

800x600 72Hz Non-interlaced

H_TOTAL = 0x82
H_DISP = 0x63
H_SYNC_STRT = 0x6A
H_SYNC_WID = 0xF
DISP_CNTL = 0x23
CLOCK_SEL = 0x10
DOT_CLOCK = 50.35MHz

V_TOTAL = 0x531
V_DISP = 0x4AB
V_SYNC_STRT = 0x4F8
V_SYNC_WID = 0x6

	Horizontal		Vertical	
Resolution	800		600	
Scan Frequency	48.044KHz		72.14Hz	
Polarity	(+)		(+)	
Sync Width	2.383us	15 chars	0.125ms	6 lines
Front Porch	1.112us	7 chars	0.770ms	37 lines
Back Porch	1.430us	9 chars	0.479ms	23 lines
Active Time	15.889us	100 chars	12.489ms	600 lines
Blank Time	4.926us	31 chars	1.374ms	66 lines

800x600 76Hz Non-interlaced

H_TOTAL = 0x86
H_SYNC_STRT = 0x6D
V_TOTAL = 0x565
V_SYNC_STRT = 0x4FA
DISP_CNTL = 0x23
DOT_CLOCK = 56.64MHz

H_DISP = 0x63
H_SYNC_WID = 0x28
V_DISP = 0x4AB
V_SYNC_WID = 0x24
CLOCK_SEL = 0x14

	Horizontal		Vertical	
Resolution	800		600	
Scan Frequency	52.444KHz		76.01Hz	
Polarity	(-)		(-)	
Sync Width	1.130us	8 chars	0.076ms	4 lines
Front Porch	1.412us	10 chars	0.744ms	39 lines
Back Porch	2.401us	17 chars	0.896ms	47 lines
Active Time	14.124us	100 chars	11.441ms	600 lines
Blank Time	4.944us	35 chars	1.716ms	90 lines

1024x768 87Hz Interlaced

H_TOTAL = 0x9D
H_DISP = 0x7F
H_SYNC_STRT = 0x81
H_SYNC_WID = 0x16
DISP_CNTL = 0x33
CLOCK_SEL = 0x1C
DOT_CLOCK = 44.90MHz

V_TOTAL = 0x668
V_DISP = 0x5FF
V_SYNC_STRT = 0x600
V_SYNC_WID = 0x8

	Horizontal		Vertical	
Resolution	1024		768	
Scan Frequency	35.522KHz		86.64Hz	
Polarity	(+)		(+)	
Sync Width	3.742us	21 chars	0.113ms	4 lines
Front Porch	0.535us	3 chars	0.028ms	1 lines
Back Porch	1.069us	6 chars	0.563ms	20 lines
Active Time	22.806us	128 chars	10.838ms	385 lines
Blank Time	5.345us	30 chars	0.704ms	25 lines

1024x768 60Hz Non-interlaced

H_TOTAL = 0xA7
H_DISP = 0x7F
H_SYNC_STRT = 0x82
H_SYNC_WID = 0x31
DISP_CNTL = 0x23
CLOCK_SEL = 0x3C
DOT_CLOCK = 65.00MHz

V_TOTAL = 0x649
V_DISP = 0x5FF
V_SYNC_STRT = 0x602
V_SYNC_WID = 0x26

	Horizontal		Vertical	
Resolution	1024		768	
Scan Frequency	48.363KHz		60.00Hz	
Polarity	(+)		(+)	
Sync Width	2.092us	17 chars	0.124ms	6 lines
Front Porch	0.369us	3 chars	0.062ms	3 lines
Back Porch	2.462us	20 chars	0.600ms	29 lines
Active Time	15.754us	128 chars	15.880ms	768 lines
Blank Time	4.923us	40 chars	0.786ms	38 lines

1024x768 66Hz Non-interlaced

H_TOTAL = 0xAD
H_DISP = 0x7F
H_SYNC_STRT = 0x85
H_SYNC_WID = 0x16
DISP_CNTL = 0x23
CLOCK_SEL = 0x38
DOT_CLOCK = 75.00MHz

V_TOTAL = 0x65B
V_DISP = 0x5FF
V_SYNC_STRT = 0x60B
V_SYNC_WID = 0x4

	Horizontal		Vertical	
Resolution	1024		768	
Scan Frequency	53.879KHz		66.03Hz	
Polarity	(+)		(+)	
Sync Width	2.347us	22 chars	0.074ms	4 lines
Front Porch	0.640us	6 chars	0.148ms	8 lines
Back Porch	1.920us	18 chars	0.668ms	36 lines
Active Time	13.653us	128 chars	14.254ms	768 lines
Blank Time	4.907us	46 chars	0.891ms	48 lines

1024x768 70Hz Non-interlaced

H_TOTAL = 0xA5
H_DISP = 0x7F
H_SYNC_STRT = 0x83
H_SYNC_WID = 0x31
CLOCK_SEL = 0x38
DOT_CLOCK = 75.00MHz

V_TOTAL = 0x649
V_DISP = 0x5FF
V_SYNC_STRT = 0x602
V_SYNC_WID = 0x26

	Horizontal		Vertical	
Resolution	1024		768	
Scan Frequency	56.476KHz		70.07Hz	
Polarity	(-)		(-)	
Sync Width	1.813us	17 chars	0.106ms	6 lines
Front Porch	0.320us	3 chars	0.053ms	3 lines
Back Porch	1.920us	18 chars	0.513ms	29 lines
Active Time	13.653us	128 chars	13.599ms	768 lines
Blank Time	4.053us	38 chars	0.673ms	38 lines

1024x768 72Hz Non-interlaced

H_TOTAL = 0xA0
H_DISP = 0x7F
H_SYNC_STRT = 0x82
H_SYNC_WID = 0x31
DISP_CNTL = 0x23
CLOCK_SEL = 0x38
DOT_CLOCK = 75.00MHz

V_TOTAL = 0x649
V_DISP = 0x5FF
V_SYNC_STRT = 0x602
V_SYNC_WID = 0x26

	Horizontal		Vertical	
Resolution	1024		768	
Scan Frequency	58.230KHz		72.245Hz	
Polarity	(-)		(-)	
Sync Width	1.813us	17 chars	0.103ms	6 lines
Front Porch	0.320us	3 chars	0.052ms	3 lines
Back Porch	1.387us	13 chars	0.498ms	29 lines
Active Time	13.653us	128 chars	13.189ms	768 lines
Blank Time	3.520us	33 chars	0.653ms	38 lines

1024x768 76Hz Non-interlaced

H_TOTAL = 0xA2
H_DISP = 0x7F
H_SYNC_STRT = 0x87
H_SYNC_WID = 0xB
DISP_CNTL = 0x23
CLOCK_SEL = 0x2C
DOT_CLOCK = 80.00MHz

V_TOTAL = 0x64A
V_DISP = 0x5FF
V_SYNC_STRT = 0x60B
V_SYNC_WID = 0x4

	Horizontal		Vertical	
Resolution	1024		768	
Scan Frequency	61.350KHz		76.02Hz	
Polarity	(+)		(+)	
Sync Width	1.100us	11 chars	0.065ms	4 lines
Front Porch	0.800us	8 chars	0.130ms	8 lines
Back Porch	1.600us	16 chars	0.440ms	27 lines
Active Time	12.800us	128 chars	12.518ms	768 lines
Blank Time	3.500us	35 chars	0.636ms	39 lines

1280x1024 87Hz Interlaced

H_TOTAL = 0xC7
H_DISP = 0x9F
H_SYNC_STRT = 0xA9
H_SYNC_WID = 0xA
DISP_CNTL = 0x33
CLOCK_SEL = 0x2C
DOT_CLOCK = 80.00MHz

V_TOTAL = 0x8F8
V_DISP = 0x7FF
V_SYNC_STRT = 0x861
V_SYNC_WID = 0xA

	Horizontal		Vertical	
Resolution	1280		1024	
Scan Frequency	50.000KHz		87.03Hz	
Polarity	(+)		(+)	
Sync Width	1.000us	10 chars	0.100ms	10 lines
Front Porch	1.000us	10 chars	0.500ms	50 lines
Back Porch	2.000us	20 chars	0.650ms	65 lines
Active Time	16.00us	160 chars	10.240ms	1024 lines
Blank Time	4.000us	40 chars	1.250ms	125 lines

1280x1024 95Hz Interlaced

H_TOTAL = 0xC7
H_DISP = 0x9F
H_SYNC_STRT = 0xA9
H_SYNC_WID = 0xA
DISP_CNTL = 0x33
CLOCK_SEL = 0x2C
DOT_CLOCK = 80.00MHz

V_TOTAL = 0x838
V_DISP = 0x7FF
V_SYNC_STRT = 0x811
V_SYNC_WID = 0xA

	Horizontal		Vertical	
Resolution	1280		1024	
Scan Frequency	50.000KHz		94.97Hz	
Polarity	(+)		(+)	
Sync Width	1.000us	10 chars	0.100ms	10 lines
Front Porch	1.000us	10 chars	0.100ms	10 lines
Back Porch	2.000us	20 chars	0.090ms	9 lines
Active Time	16.00us	160 chars	10.240ms	1024 lines
Blank Time	4.000us	40 chars	0.290ms	29 lines

1280x1024 60Hz Non-interlaced

H_TOTAL = 0xD6
H_DISP = 0x9F
H_SYNC_STRT = 0xA9
H_SYNC_WID = 0x2E
DISP_CNTL = 0x23
CLOCK_SEL = 0x28
DOT_CLOCK = 110.00MHz

V_TOTAL = 0x852
V_DISP = 0x7FF
V_SYNC_STRT = 0x800
V_SYNC_WID = 0x25

	Horizontal		Vertical	
Resolution	1280		1024	
Scan Frequency	63.953KHz		59.94Hz	
Polarity	(-)		(-)	
Sync Width	1.018us	14 chars	0.078ms	5 lines
Front Porch	0.727us	10 chars	0.016ms	1 lines
Back Porch	2.255us	31 chars	0.579ms	37 lines
Active Time	11.636us	160 chars	16.012ms	1024 lines
Blank Time	4.000us	55 chars	0.672ms	43 lines

1280x1024 70Hz Non-interlaced

H_TOTAL = 0xD2
H_DISP = 0x9F
H_SYNC_STRT = 0xA9
H_SYNC_WID = 0xE
DISP_CNTL = 0x23
CLOCK_SEL = 0x4
DOT_CLOCK = 126.00MHz

V_TOTAL = 0x851
V_DISP = 0x7FF
V_SYNC_STRT = 0x800
V_SYNC_WID = 0x5

	Horizontal		Vertical	
Resolution	1280		1024	
Scan Frequency	74.645KHz		70.02Hz	
Polarity	(+)		(+)	
Sync Width	0.889us	14 chars	0.067ms	5 lines
Front Porch	0.635us	10 chars	0.013ms	1 lines
Back Porch	1.714us	27 chars	0.482ms	36 lines
Active Time	10.159us	160 chars	13.718ms	1024 lines
Blank Time	3.238us	51 chars	0.563ms	42 lines

1280x1024 74Hz Non-interlaced

H_TOTAL = 0xCF
H_DISP = 0x9F
H_SYNC_STRT = 0xAE
H_SYNC_WID = 0x11
DISP_CNTL = 0x23
CLOCK_SEL = 0x20
DOT_CLOCK = 135.00MHz

V_TOTAL = 0x851
V_DISP = 0x7FF
V_SYNC_STRT = 0x818
V_SYNC_WID = 0x10

	Horizontal		Vertical	
Resolution	1280		1024	
Scan Frequency	78.855KHz		74.11Hz	
Polarity	(+)		(+)	
Sync Width	1.067us	18 chars	0.380ms	30 lines
Front Porch	0.237us	4 chars	0.000ms	0 lines
Back Porch	1.896us	32 chars	0.127ms	10 lines
Active Time	9.481us	160 chars	12.986ms	1024 lines
Blank Time	3.200us	54 chars	0.507ms	40 lines

E

Pixel Clock Specifications

18811-0 Clock Chip Pixel Clocks (MHz)

Select (MHz)	0	1	2	3	4	5	6	7
	42.95	48.77	92.40	36.00	50.35	56.64	External	44.90
Select (MHz)	8	9	10	11	12	13	14	15
	30.24	32.00	110.00	80.00	39.91	44.90	75.00	65.00

18811-1 Clock Chip Pixel Clocks (MHz)

Select (MHz)	0	1	2	3	4	5	6	7
	100.00	126.00	92.40	36.00	50.35	56.64	External	44.90
Select (MHz)	8	9	10	11	12	13	14	15
	135.00	32.00	110.00	80.00	39.91	44.90	75.00	65.00

This page intentionally left blank.

F

RAMDAC Programming

mach32 graphics accelerators support different RAMDACs for various display resolution and color depth requirements. The RAMDAC type is readable in register CONFIG_STATUS_1[11:9] (12EE-R). All RAMDACs listed below and their compatibles will provide the performance as indicated:

RAMDAC	Type	Interface	Color Depth	Resolution
ATI68830	0	8-bit	8bpp	640x480 800x600 1024x768
			16bpp	640x480 800x600 1024x768
IMS-G173 MU9C4870 SC11483 SC11486 SC11488	1	8-bit	8bpp	640x480 800x600 1024x768
			16bpp	640x480 800x600
ATI68875 Bt885 TLC34075	2	24-/32-bit	8bpp	640x480 800x600 1024x768 1280x1024
			16bpp	640x480 800x600 1024x768
			24bpp	640x480 800x600
Bt476 Bt478 INMOS176 INMOS178	3	8-bit	8bpp	640x480 800x600 1024x768

RAMDAC	Type	Interface	Color Depth	Resolution
AT&T20C490 AT&T20C491 Bt481 Bt482 IMS-G174 MU9C1880 MU9C4910 SC15025 SC15026	4	8-bit	8bpp	640x480 800x600 1024x768
			16bpp	640x480 800x600
			24bpp	640x480
ATI68860	5	24/32/64-bit	8bpp	640x480 800x600 1024x768 1280x1024
			16bpp	640x480 800x600 1024x768 1280x1024
			24bpp	640x480 800x600 1024x768 1280x1024

RAMDAC DESCRIPTIONS

Type 0 RAMDAC — ATI68830

This DAC type is rated at 80MHz with a maximum pixel width of 16bpp (555, 565, 655, and 664) for screen resolutions up to 1024x768 and 4bpp/8bpp up to 1280x1024 interlaced. No special programming is required for initializing the DAC to a specific video mode. Sample source code is provided on page F-7. In general, the procedure for programming a video mode is as follows:

1. Set up the CRT controller by programming it directly or by using one of the shadow register sets configured at ADVFUNC_CNTL[2] (4AE8).
2. Set EXT_GE_CONFIG[5:4] (7AEE) to the desired pixel width.
3. Set mode-specific values for pixel delay MISC_CNTL[11:10] (7EEE-W).

Type 1 RAMDAC — IMS-G173/SC11486

This DAC type is rated at 80MHz with a maximum pixel width of 8bpp for screen resolutions up to 1024x768 and 16bpp up to 800x600. A sample source code is provided on page F-7. In general, the procedure for programming a video mode is as follows:

1. Set up the CRT controller by programming it directly or by using one of the shadow register sets configured at ADVFUNC_CNTL[2] (4AE8).
2. Select a clock rate from CLOCK_SEL[6] (4AEE).
3. Set the high DAC address bits RS(3:2) on EXT_GE_CONFIG[13:12] (7AEE-W).
4. Program the Command register of the RAMDAC to the desired mode.
5. Reset EXT_GE_CONFIG[5:4] to the desired pixel width.
6. Reset PASSTHRU.

Type 2 RAMDAC — ATI68875/TLC34075

The ATI68875/TLC34075 DAC type is a high performance palette DAC with a pixel clock rating of 135MHz. These DACs are 100% compatible with industry standard triple-6 RAMDACs. They also feature triple-8 support, wide pixel data path, and a variety of multiplexing options — support for four pixel arrangements in 24bpp mode (RGB, BGR, RGBa, and aBGR) as well as four weightings in 16bpp mode (555, 565, 655, and 664). The 24bpp mode supports up to a screen resolution of 800x600; the 16bpp mode supports up to 1024x768, and 4bpp/8bpp modes support up to 1280x1024. Select pins RS(3:0) are used to select RAMDAC registers as follows:

Register	Name
0h	Palette address register - write mode
1h	Color palette holding register
2h	Pixel read mask
3h	Palette address register - read mode
4:7h	Reserved
8h	General control register
9h	Input clock selection register
Ah	Output clock selection register
Bh	Mux control register
Ch	Palette page register
Dh	Reserved
Eh	Test register
Fh	Reset state

Registers 0, 1, 2, and 3 are mapped to ports 2EC, 2ED, 2EA, and 2EB respectively. The other registers are accessed by mapping the two high address bits RS(3:2) to

EXT_GE_CONFIG[13:12] (7AEE), setting the register code for RS(3:2), and writing the values to ports 2EC, 2ED, 2EA, and 2EB. The procedure to initialize this DAC to a given mode is as follows:

1. Set up the CRT controller by programming it directly or by using one of the shadow register sets configured at ADVFUNC_CNTL[2] (4AE8).
2. Ensure that the DAC is receiving clean pixel clocks by disabling VGA passthrough mode and selecting a sufficiently low clock rate from CLOCK_SEL (4AEE).
3. Configure the high DAC address RS(3:2) by setting EXT_GE_CONFIG[13:12] (7AEE) and ensure that EXT_GE_CONFIG[5:4] is set to 8bpp pixel width.
4. Program the following RAMDAC registers: INPUT_CLK_SEL, OUTPUT_CLK_SEL, and MUX_CNTL.
5. If the mode requires a 32 bit data path (e.g., in 16bpp and 24bpp modes) ensure that CLOCK_SEL[5:2] (4AEE) is set for clock-divide-by-1; and the DAC is set for VCLK/2.
6. Reset EXT_GE_CONFIG[5:4] (7AEE) to the desired pixel width and configure EXT_GE_CONFIG[8] for multiplex pixels accordingly. All modes with a pixel clock above 80MHz should have multiplex set to Mux mode.
7. Configure EXT_GE_CONFIG[14] appropriately. It is enabled for all 16bpp and 24bpp modes; but disabled for all other modes.
8. Set DAC_MASK to 0x0F for 4bpp modes, 0xFF for 8bpp modes, and zero for all other pixel widths.

Type 3 RAMDAC — INMOS176/INMOS178

No initialization is required for Type 3 VGA-compatible DACs.

Type 4 RAMDAC — Bt481/Bt482

The Bt481/Bt482 are rated at 80MHz. and are capable of supporting a maximum pixel width of 24bpp (RGB only) at a 640x480 screen resolution; 16bpp (555 and 565 only) up to 800x600, and 4bpp/8bpp up to 1280x1024 interlaced. The DAC registers are as follows:

Register	Name
0h	Palette address register - RAM write mode
1h	Color palette holding register
2h	Pixel read mask
3h	Palette address register - RAM read mode
4h	Palette address register - overlay write mode
5h	Overlay register
6h	Mode Control register
7h	Palette address register - overlay read mode

The procedure to initialize this DAC to a given mode is as follows:

1. Set up the CRT controller by programming it directly or by using one of the shadow register sets configured at ADVFUNC_CNTL[2] (4AE8).
2. Ensure that the DAC is receiving clean pixel clocks by disabling VGA passthrough mode and selecting a sufficiently low clock rate from CLOCK_SEL (4AEE).
3. Configure the high DAC address bits RS(3:2) by setting EXT_GE_CONFIG[13:12] (7AEE) and ensure that EXT_GE_CONFIG[5:4] is set to 8bpp pixel width.
4. Program the MODE_CONTROL register (6h) on the DAC. It should be enabled for 16bpp and 24bpp modes. Note that "8-bit Enable" is not configured in EXT_GE_CONFIG[14].
5. Program mode-specific values for PIXEL_DELAY and HORIZONTAL_SKEW.
6. Reset EXT_GE_CONFIG[5:4] (7AEE) to the desired pixel width remembering to re-set the high DAC address bits RS(3:2).
7. Set DAC_MASK to 0x0F for 4bpp modes, 0xFF for 8bpp modes, and zero for all other pixel widths.

Type 5 RAMDAC — ATI68860

This RAMDAC type is supported by the ATI68800AX or higher ***mach32*** controllers only. The ATI68860 is a high performance palette DAC that can operate at 135MHz and support display modes up to 1280x1024 at 24bpp. It supports different multiplexed modes in 16bpp, 24bpp and 32bpp configurations. This DAC is downward compatible with the IMS-G176 and IMS-G178. See sample code segments on page F-7 for details. The procedure to initialize this DAC to a given mode is as follows:

1. Set EXT_GE_CONFIG[13:12] to configure the high DAC address RS(3:2).
2. Program the CLOCK_SELECT.
3. Program the Graphic Mode Control register GMR[6:0] to set a graphics mode.
4. Reset EXT_GE_CONFIG[5:4] to the desired pixel width.
5. Reset PASSTHRU.

Sample Code Segments

```
; ============================================================================
; INITDAC.ASM - Copyright (c) 1992 ATI Technologies Inc. All rights reserved
;
; Utility 68800 DAC functions.
;
; Compiling:
;   masm /Ml /D<memory model> initac.asm;
;       <memory model> = mem_S for SMALL model,
;                        mem_M for MEDIUM model,
;                        mem_L for LARGE model
; ============================================================================

include ati8514.inc

IFDEF mem_S
PARM        equ     4   ; passed parameters start at bp+4 for small model
ELSE
PARM        equ     6   ; passed parameters start at bp+6 for other models
ENDIF

IFDEF mem_S
.MODEL  SMALL, C
ELSEIFDEF mem_M
.MODEL  MEDIUM, C
ELSE
.MODEL  LARGE, C
ENDIF

.DATA

.CODE
.286

ATI68830_DAC    equ     0
ATT20C491_DAC   equ     2
TI_DAC          equ     4
BROOKTREE_DAC   equ     6
BT481_DAC       equ     8
ATI_68860       equ     5*2

ATT_MODE_CNTL   equ     DAC_MASK
INPUT_CLK_SRC   equ     1

; Macro for 'call' model handling
```

```
Mcall       macro   routine
IFDEF mem_S
            call    NEAR PTR routine
ELSE
            call    FAR PTR routine
ENDIF
              endm

;-------------------------------------
; INIT_DAC - Sets up RAMDAC on 68800 for specified configuration.
; This is a main routine to be invoked to initialize a RAMDAC.
; For example, if you want to set RAMDAC to 8 bit mode, call
;       init_dac ( 1 );
;
; Inputs : al = RAMDAC configuration
;                 bits 1:0 - Pixel width
;                             0 = 4 bpp
;                             1 = 8 bpp
;                             2 = 16 bpp
;                             3 = 24 bpp
;                 bits 3:2 - 16 bit color mode
;                             0 = 555
;                             1 = 565
;                             2 = 655
;                             3 = 664
;                 bit 4 - mux bit for non-interlaced 1280 mode on ti dac
;                             0 = no mux mode
;                             1 = turn on mux mode
;                 bit 5 - 24 bit color config
;                             0 = 3 bytes per pixel
;                             1 = 4 bytes per pixel
;                 bit 6 - 24 bit color order
;                             0 = RGB
;                             1 = BGR
;                 bit 7 - DAC reset
;                             0 = use bits 6:0 to config (no reset)
;                             1 = ignore bits 6:0 and reset DAC
;
; Outputs: none
;-------------------------------------
            public  init_dac

IFDEF mem_S
init_dac    proc    near
ELSE
init_dac    proc    far
ENDIF
            ; save used registers
            push    bx
            push    cx
```

```
            push    dx

            ; check for reset bit
            test    al, 80h
            jz      no_reset

            ; reset dac and leave
            Mcall   uninit_ti_dac
            mov     ax, 0fh             ; set VGA DAC MASK to 4 bpp
            mov     dx, VGA_DAC_MASK
            out     dx, al

            jmp     init_dac_end

no_reset:
            ; use bits 6:0 to initialize dac
            and     al, 7fh             ; zero bit 7
            mov     cl, al              ; save input in cl
            and     cl, 3               ; zero upper six bits
            xor     bh, bh
            mov     bl, al
            shl     bx, 4
            mov     dx, R_EXT_GE_CONFIG ; preserve monitor alias
            in      ax, dx
            and     ax, 000eh           ; zero all but monitor alias
            or      ax, 0008h           ; enable monitor alias bit
            or      ax, bx              ; map input bits on ax

            ; call dac initialization routine depending on color depth
            cmp     cl, 0               ; check for 4 bpp
            jne     check_8bpp
            push    ax
            and     ah, 0feh            ; zero mux bit
            Mcall   init_ti_8
            pop     ax
            test    ah, 1               ; check if mux bit is set
            jz      no_mux_1
            Mcall   init_ti_mux

no_mux_1:
            jmp     init_dac_end

check_8bpp:
            cmp     cl, 1               ; check for 8 bpp
            jne     check_16bpp
            push    ax
            and     ah, 0feh            ; zero mux bit
            Mcall   init_ti_8
            pop     ax
            test    ah, 1               ; check if mux bit is set
```

```
            jz      no_mux_2
            Mcall   init_ti_mux

no_mux_2:
            jmp     init_dac_end

check_16bpp:
            and     ah, 0feh            ; zero mux bit for rest of checks
            cmp     cl, 2               ; check for 16 bpp
            jne     check_24bpp
            Mcall   init_ti_16
            jmp     init_dac_end

check_24bpp:
            cmp     cl, 3               ; check for 24 bpp
            jne     init_dac_end
            Mcall   init_ti_24

init_dac_end:

            ; restore saved registers
            pop     dx
            pop     cx
            pop     bx

            ret

init_dac    endp


; ---------------------------------------
; PASSTH_8514 - Disable VGA passthrough
;
; Inputs : none
;
; Outputs: none
; ---------------------------------------
            public  passth_8514

IFDEF mem_S
passth_8514 proc    near
ELSE
passth_8514 proc    far
ENDIF
            ; save registers used
            push    ax
            push    dx

            ; disable VGA passthrough
            mov     dx, CLOCK_SELECT
```

```
            in      ax, dx
            or      ax, 1
            out     dx, ax

            ; restore registers used
            pop     dx
            pop     ax

            ret

passth_8514 endp

; ---------------------------------------
; PASSTH_VGA - Enable VGA passthrough
;
; Inputs : none
;
; Outputs: none
; ---------------------------------------
            public  passth_vga

IFDEF mem_S
passth_vga  proc    near
ELSE
passth_vga  proc    far
ENDIF
            ; save registers used
            push    ax
            push    dx

            ; disable VGA passthrough
            mov     dx, CLOCK_SELECT
            in      ax, dx
            and     ax, 0fffeh
            out     dx, ax

            ; restore registers used
            pop     dx
            pop     ax

            ret

passth_vga  endp
```

```
; ---------------------------------------
; SET_BLANK_ADJ - Sets the blank adjust and pixel delay values
;
; Inputs : al = blank adjust and pixel delay
;
; Outputs: none
; ---------------------------------------
            public  set_blank_adj

IFDEF mem_S
set_blank_adj proc  near
ELSE
set_blank_adj proc  far
ENDIF
            ; save registers used
            push    dx

            ; set blank adjust
            mov     ah, al
            mov     dx, R_ROM_CNTL+1
            in      al, dx
            and     al, 0f0h
            or      al, ah
            mov     dx, ROM_EEPROM_CNTL+1
            out     dx, al

            ; restore registers used
            pop     dx

            ret

set_blank_adj endp

;---------------------------------------
; INIT_TI_8 - Initialize DAC for standard 8 bit per pixel mode
;
; Inputs : ax = default EXT_GE_CONFIG
;                 (should be 0Ah for 4bpp,
;                            1Ah for 8bpp)
;
; Outputs: none
;---------------------------------------
            public  init_ti_8

IFDEF mem_S
init_ti_8   proc    near
ELSE
init_ti_8   proc    far
ENDIF
            ; save used registers
```

```
        push    bx
        push    cx
        push    dx

        ; place input value in cx
        mov     cx, ax

        ; set PIXEL_DELAY to 3
        mov     bh, 0ch
        mov     dx, CONFIG_STATUS_1
        in      ax, dx
        and     ah, 0eh
        cmp     ah, ATI68830_DAC
        jne     ti8_3
        mov     bh, 04h
ti8_3:
        push    ax
        mov     al, bh
        Mcall   set_blank_adj
        pop     ax
ti8_4:
        cmp     ah, ATT20C491_DAC
        je      ti8_2
        cmp     ah, BT481_DAC
        jne     ti8_1
ti8_2:
        ; Set EXT_DAC_ADDR
        mov     dx, EXT_GE_CONFIG
        mov     ax, 101ah
        out     dx, ax

        mov     dx, ATT_MODE_CNTL
        mov     al, 0
        out     dx, al
ti8_1:
        mov     dx, CONFIG_STATUS_1
        in      ax, dx
        and     ah, 0eh
        cmp     ah, TI_DAC
        jne     ti8_end

        ; Set EXT_DAC_ADDR field
        mov     dx, EXT_GE_CONFIG
        mov     ax, 201ah
        out     dx, ax

        ; input clock source is CLK0
        mov     dx, INPUT_CLK_SEL
        mov     al, 0
        out     dx, al
```

```
            ; output clock is SCLK/1 and VCLK disabled
            mov     dx, OUTPUT_CLK_SEL
            mov     al, 30h
            out     dx, al

            ; set MUX control to 8/16
            mov     dx, MUX_CNTL
            mov     al, 2dh
            out     dx, al
ti8_end:
            ; reset EXT_DAC_ADDR, put DAC in 6 bit mode
            mov     dx, EXT_GE_CONFIG
            mov     ax, cx
            out     dx, ax

            ; enable DAC_MASK
            mov     dx, DAC_MASK
            mov     al, 0ffh
            out     dx, al

            ; restore saved registers
            pop     dx
            pop     cx
            pop     bx

            ret

init_ti_8   endp


;-----------------------------------
;
; Inputs : cx = value to program EXT_GE_CONFIG
; Outputs: ax = value to program Graphics Mode Control register
;          Applicable to ATI68860 only
;
;-----------------------------------
IFDEF mem_S
find_GMR_data proc  near
ELSE
find_GMR_data proc  far
ENDIF
        xor     ax,ax
        cmp     cx,2ah          ;16bpp 555
        jnz     @F
        mov     ax,20h
@@:     cmp     cx,6ah          ;16bpp 565
        jnz     @F
        mov     ax,21h
```

```
@@:     cmp     cx,0aah         ;16bpp 655
        jnz     @F
        mov     ax,22h
@@:     cmp     cx,0eah         ;16bpp 664
        jnz     @F
        mov     ax,23h

@@:     cmp     cx,3ah          ;24bpp RGB
        jnz     @F
        mov     ax,40h
@@:     cmp     cx,43ah         ;24bpp BGR
        jnz     @F
        mov     ax,41h
@@:     cmp     cx,23ah         ;24bpp RGBa
        jnz     @F
        mov     ax,60h
@@:     cmp     cx,63ah         ;24bpp aBGR
        jnz     @F
        mov     ax,61h
@@:     ret

find_GMR_data   endp

IFDEF mem_S
init_68860  proc  near
ELSE
init_68860 proc  far
ENDIF
            mov     dx,EXT_GE_CONFIG
            mov     ax,201ah
            out     dx,ax
            mov     dx,OUTPUT_CLK_SEL
            mov     ax,0
            out     dx,ax
            call    find_GMR_data
            mov     dx,EXT_GE_CONFIG+1  ;device setup
            mov     al,30h
            out     dx,al
            mov     dx,2ech
            out     dx,al
            mov     dx,EXT_GE_CONFIG
            mov     ax,cx
            out     dx,ax
            ret
init_68860  endp


IFDEF mem_S
init_34075 proc  near
ELSE
```

```
init_34075 proc  far
ENDIF
            ; set BLANK_ADJUST=1, PIXEL_DELAY=0
            push    ax
            mov     al, 1
            Mcall   set_blank_adj
            pop     ax

            ; Set EXT_DAC_ADDR field
            mov     dx, EXT_GE_CONFIG
            mov     ax, 201ah
            out     dx, ax

            ; input clock source is CLK3
            mov     dx, INPUT_CLK_SEL
            mov     al, INPUT_CLK_SRC
            out     dx, al

            ; output clock source is SCLK/1 and VCLK/1
            ;   - for modes which require PCLK/2, set VCLK/2
            mov     dx, CLOCK_SELECT
            in      ax, dx
            test    ax, 0c0h
            jz      ti16_2
            and     ax, 0ff3fh
            out     dx, ax

            mov     dx, SRC_X_START
            in      al, dx          ; get H_DISP
            cmp     al, 4fh         ; 640?
            jne     ti16_4

            ; exception case: 640x480 60 Hz needs longer blank adjust (2)
            push    ax
            mov     al, 2
            Mcall   set_blank_adj
            pop     ax
ti16_4:
            mov     al, 8
            jmp     short ti16_3
ti16_2:
            mov     al, 0
ti16_3:
            mov     dx, OUTPUT_CLK_SEL
            out     dx, al

            ; set MUX control to 24/32
            mov     dx, MUX_CNTL
            mov     al, 0dh
            out     dx, al
```

```
            ; reset EXT_DAC_ADDR, put DAC in 8 bit mode, engine in 555 mode
            mov     dx, EXT_GE_CONFIG
            mov     ax, cx
            or      ax, 4000h
            out     dx, ax
            mov     dx,DAC_MASK
            mov     al,0
            out     dx,al
            ret
init_34075      endp
;-------------------------------------
; INIT_TI_16 - Initialize DAC for 16 bit per pixel mode
;
; Inputs : ax = default value for EXT_GE_CONFIG
;                 (should be 2Ah for 555 16 bpp,
;                            6Ah for 565 16 bpp,
;                            AAh for 655 16 bpp,
;                            EAh for 664 16 bpp)
;
; Outputs: none
;-------------------------------------
            public  init_ti_16

IFDEF mem_S
init_ti_16  proc    near
ELSE
init_ti_16  proc    far
ENDIF
            ; save used registers
            push    bx
            push    cx
            push    dx

            ; place input value in cx
            mov     cx, ax

            ; set pixel delay=1 for 68830 and 3 for all others
            mov     bh, 0ch
            mov     dx, CONFIG_STATUS_1
            in      ax, dx
            and     ah, 0eh
            cmp     ah, ATI68830_DAC
            jne     ti16_5
            mov     bh, 04h
ti16_5:
            push    ax
            mov     al, bh
            Mcall   set_blank_adj
            pop     ax
```

```
init16_1:
            cmp     ah, BT481_DAC
            je      att16_1
            cmp     ah, ATT20C491_DAC
            je      att16_1
            cmp     ah, TI_DAC
            je      ti16_1
            cmp     ah, ATI_68860
            jnz     set_ext_ge_16
            call    init_68860
            jmp     ti_16_end
set_ext_ge_16:
            mov     ax, cx
            mov     dx, EXT_GE_CONFIG
            out     dx, ax
            jmp     ti_16_end

            ; ATT20C491 initialization
att16_1:
            mov     dx, DAC_MASK
            mov     al, 0
            out     dx, al

            ; set EXT_DAC_ADDR
            mov     dx, EXT_GE_CONFIG
            mov     ax, 101ah
            out     dx, ax

            ; determine 555 or 565
            test    cx, 00c0h
            jz      att16_2
            mov     dx, CONFIG_STATUS_1
            in      ax, dx
            and     ah, 0eh
            cmp     ah, BT481_DAC
            je      bt16_1
            mov     al, 0c2h        ; 565, 8 bit
            jmp     short att16_5
bt16_1:
            mov     al, 0e8h        ; 565, 8 bit
            jmp     short att16_5
att16_2:
            mov     dx, CONFIG_STATUS_1
            in      ax, dx
            and     ah, 0eh
            cmp     ah, BT481_DAC
            je      bt16_2
            mov     al, 0a2h        ; 555, 8 bit
            jmp     short att16_5
```

```
bt16_2:
            mov     al, 0a8h        ; 555, 8 bit
att16_5:
            mov     dx, ATT_MODE_CNTL
            out     dx, al

            mov     dx, CLOCK_SELECT
            in      ax, dx
            test    ax, 00c0h       ; test for divide by 2
            jz      att16_3
            and     ax, 0ff3fh      ; divide by 1 instead
            jmp     short att16_4

att16_3:
            mov     bx, ax
            and     bx, 003ch
            cmp     bl, 30h         ; 40 MHz?
            jne     att16_4
            and     ax, 0ffc3h
            or      ax, 2ch         ; 80 MHz
att16_4:
            out     dx, ax
            jmp     set_ext_ge_16

            ; TLC34075 initialization
ti16_1:
            ; disable overlay feature
            mov     dx, DAC_MASK
            mov     al, 0
            out     dx, al

            call    init_34075

ti_16_end:
            ; make sure VGA DAC MASK is set to 4 bpp
            mov     ax, 0fh
            mov     dx, VGA_DAC_MASK
            out     dx, al

            ; restore saved registers
            pop     dx
            pop     cx
            pop     bx

            ret

init_ti_16  endp

;-----------------------------

; INIT_TI_24 - Initialize DAC for 24 bit per pixel mode, 3 bytes, RGB
```

```
;
; Inputs : ax = default EXT_GE_CONFIG
;                (03Ah for RGB,
;                 43Ah for BGR,
;                 23Ah for RGBa,
;                 63Ah for aBGR)
;
; Outputs: none
;-------------------------------
          public  init_ti_24

IFDEF mem_S
init_ti_24  proc    near
ELSE
init_ti_24  proc    far
ENDIF
          ; save used registers
          push    bx
          push    cx
          push    dx

          ; place input value in cx
          mov     cx, ax

          mov     bh, 0ch
          mov     dx, CONFIG_STATUS_1
          in      ax, dx
          and     ah, 0eh
          cmp     ah, ATI68830_DAC
          jne     ti24_5
          mov     bh, 04h
ti24_5:
          push    ax
          mov     al, bh
          Mcall   set_blank_adj
          pop     ax
init24_1:
          cmp     ah, TI_DAC
          je      ti24_1
          cmp     ah, BT481_DAC
          je      att24_1
          cmp     ah, ATT20C491_DAC
          je      att24_1
          cmp     ah, ATI_68860
          jnz     set_ext_ge_24
          call    init_68860
          jmp     ti_24_end


set_ext_ge_24:
```

```
                mov     ax, cx
                mov     dx, EXT_GE_CONFIG
                out     dx, ax
                jmp     ti_24_end

                ; ATT20C491 initialization
att24_1:
                mov     dx, DAC_MASK
                mov     al, 0
                out     dx, al

                ; set EXT_DAC_ADDR field
                mov     dx, EXT_GE_CONFIG
                mov     ax, 101ah
                out     dx, ax

                ; set 24bpp bypass mode
                mov     dx, CONFIG_STATUS_1
                in      ax, dx
                and     ah, 0eh
                cmp     ah, BT481_DAC
                je      bt24_1
                mov     al, 0e2h
                jmp     short att24_2
bt24_1:
                mov     al, 0f8h
att24_2:
                mov     dx, ATT_MODE_CNTL
                out     dx, al

                ; set pixel clock to 75 MHz (640x480/60 is only valid mode)
                mov     dx, CLOCK_SELECT
                in      ax, dx
                and     ax, 0ff03h
                or      ax, 38h
                out     dx, ax
                jmp     set_ext_ge_24

                ; TLC34075 initialization
ti24_1:
                call    init_34075

ti_24_end:
                ; make sure VGA DAC MASK is set to 4 bpp
                mov     ax, 0fh
                mov     dx, VGA_DAC_MASK
                out     dx, al

                ; restore saved registers
                pop     dx
```

```
            pop     cx
            pop     bx

            ret

init_ti_24  endp

;-----------------------------------
; INIT_TI_MUX - Put TI DAC into MUX mode for 1280 non-interlaced displays
;
; Inputs : ax = default EXT_GE_CONFIG
;                 (should be 10Ah for 4bpp,
;                            11Ah for 8bpp)
;
; Outputs: none
;-----------------------------------
            public  init_ti_mux

IFDEF mem_S
init_ti_mux proc    near
ELSE
init_ti_mux proc    far
ENDIF
            ; save used registers
            push    cx
            push    dx

            mov     cx, ax

            ; check if DAC is TI
            mov     dx, CONFIG_STATUS_1
            in      ax, dx
            and     ah, 0eh
            cmp     ah, TI_DAC
            je      is_ti_dac
            jmp     ti_mux_end
is_ti_dac:
            mov     dx, CLOCK_SELECT
            in      ax, dx
            push    ax              ; save clock select
            mov     al, 11h         ; guarantee a low pixel clock (50 MHz)
            out     dx, ax

            ; Set EXT_DAC_ADDR field
            mov     dx, EXT_GE_CONFIG
            mov     ax, 201ah
            out     dx, ax

            ; output clock is SCLK/2 and VCLK/2
            mov     dx, OUTPUT_CLK_SEL
```

```
            mov     al, 9h
            out     dx, al

            ; set MUX control to 8/16
            mov     dx, MUX_CNTL
            mov     al, 1dh
            out     dx, al

            ; input clock source is CLK3 (must be last)
            mov     dx, INPUT_CLK_SEL
            mov     al, INPUT_CLK_SRC
            out     dx, al

            ; reset EXT_DAC_ADDR, put DAC in 6 bit mode, engine in 8 bit mode, enable MUX mode
            mov     dx, EXT_GE_CONFIG
            mov     ax, cx
            out     dx, ax

            ; set BLANK_ADJUST=1, PIXEL_DELAY=0
            push    ax
            mov     al, 1
            Mcall   set_blank_adj
            pop     ax

            pop     ax
            mov     dx, CLOCK_SELECT
            out     dx, ax

ti_mux_end:
            ; restore saved registers
            pop     dx
            pop     cx

            ret

init_ti_mux endp

;-------------------------------
; UNINIT_TI_DAC - Prepare DAC for 8514/A compatible operation
;
; Inputs : none
;
; Outputs: none
;-------------------------------
            public  uninit_ti_dac

IFDEF mem_S
uninit_ti_dac proc  near
ELSE
uninit_ti_dac proc  far
```

```
ENDIF
            ; save used registers
            push    ax
            push    dx

            Mcall   passth_8514  ; can only program DAC in 8514 mode
utd6:
            mov     dx, CONFIG_STATUS_1
            in      ax, dx
            and     ah, 0eh
            cmp     ah, BT481_DAC
            je      utd5
            cmp     ah, ATT20C491_DAC
            je      utd5
            cmp     ah, TI_DAC
            jne     utd2

            ; set default 8bpp pixel delay and blank adjust
            mov     dx, LOCAL_CNTL
            in      ax, dx
            or      ax, 8
            out     dx, ax          ; TI_DAC_BLANK_ADJUST is always on

utd5:
            mov     ax, 1ah
            Mcall   init_ti_8
            mov     dx, EXT_GE_CONFIG
            mov     ax, 201ah
            out     dx, ax

            ; output clock is SCLK/1 and VCLK/1
            mov     dx, OUTPUT_CLK_SEL
            mov     al, 0h
            out     dx, al
utd2:
           ; reset EXT_DAC_ADDR, put DAC in 6 bit mode, engine in 8 bit mode
            mov     dx, EXT_GE_CONFIG
            mov     ax, 1ah
            out     dx, ax
            Mcall   passth_vga

            ; restore saved registers
            pop     dx
            pop     ax

            ret

uninit_ti_dac endp

            end
```

G

ATI68800-6 & ATI68800LX Features

A range of enhancements has been addded to the ATI68800-3 ***mach32*** accelerator to create two new chips, the ATI68800-6 and ATI68800LX. They are identical to each other except for video memory device support — the ATI68800-6 supports both DRAM and VRAM; the ATI68800LX supports only DRAM. With all "-3" functionalities retained, these new chips are therefore backward-compatible to the ATI68800-3.

Descriptions on the enhancements are incorporated into Chapters 7, 8, and 9. However, for the convenience of comparing with the "-3" chip, they are also included in this appendix. New or updated registers are as follows:

- CONFIG_STATUS_1 (12EE-R)
- CONFIG_STATUS_2 (16EE-R)
- LOCAL_CONTROL (32EE-W)
- MISC_OPTIONS (36EE-RW)
- SHADOW_SET (5AEE-W)
- MEM_CFG (5EEE-RW)
- EXT_GE_STATUS (62EE-R)
- EXT_GE_CONFIG (7AEE-W)
- PATT_DATA_INDEX (82EE-RW)
- PATT_LENGTH (D2EE-W)
- DEST_COLOR_CMP_MASK (F2EE-W)
- CHIP_ID (FAEE-R)

Mono Pattern

An 8x8x1 destination-aligned mono pattern has been implemented. It is enabled by setting D2EE[7]. This new mono pattern uses the four existing mono pattern registers and four additional mono pattern registers. The relationship between the register bits, the mono pattern, and the destination is as follows:

Register Index	DST_Y Coordinate	DST_X Coordinate 0 1 2 3 4 5 6 7
10h	0	7 6 5 4 3 2 1 0
11h	1	7 6 5 4 3 2 1 0
12h	2	7 6 5 4 3 2 1 0
13h	3	7 6 5 4 3 2 1 0
14h	4	7 6 5 4 3 2 1 0
15h	5	7 6 5 4 3 2 1 0
16h	6	7 6 5 4 3 2 1 0
17h	7	7 6 5 4 3 2 1 0

When the 8x8x1 mono pattern is used, the bits of the pattern will be drawn aligned to the destination coordinates. This behaviour differs from that of the 32x1 linear mono pattern, which has no absolute relationship between the pattern and the destination coordinates. The 32x1 linear mono pattern therefore is not considered destination-aligned. See page 9-58 for details.

(A+B)/2 16-Bit ROP

The (A+B)/2 function in 16bpp drawing modes has been implemented. This feature is used for enhancing image quality in motion video applications.

Destination Color Compare Mask

A write mask has been split into separate write mask (AAE8) and destination color compare mask (F2EE). This feature allows any plane or group of planes to be used as an alpha channel which is particularly useful in motion video applications.

For compatibility, both registers are loaded when writing to the write mask.

Far-Blit

The graphics engine offset and pitch registers have been split into separate destination and source offset and pitch registers. The new registers are accessed through a 2-bit pointer in SHADOW_SET[9:8] 5AEE-W. A pointer value of "1" will load the destination registers. A pointer value of "2" will load the source registers.

This feature allows unsymetrical blit operations between any areas of the memory and is particularly useful for efficient linearized cache management, rendering images directly into linearized cache and 24-bit drawing operations.

For compatibility, if the pointer is set to "0" both the destination and source registers will be loaded concurrently.

Block Write and 64-Bit Fill Draw

Block Write

Most VRAM devices have a block write mode in which they can perform a pseudo mono data expansion using an internal color register — Block Write Mono Pattern mode. Data bits which are set to "1" enable an internal 4-bit color register to be drawn into a nibble of memory. Data bits which are "0" do not alter the addressed nibble of memory. Since there is only one internal color register, this mode can only be used for foreground color paint operations. A mono pattern may be applied if the background ALU function is transparent. The more common mono pattern which uses the foreground and background color registers may be performed in two passes by first painting the background and then applying a mono pattern with the foreground color.

This feature is only supported with memory type 5 or 6. See CONFIG_STATUS[6:4] (12EE). Block Write mode is enabled by setting MISC_OPTIONS[10] (36EE) to 1. Block Write Mono Pattern mode is enabled by setting PATT_LENGTH[15] (D2EE) to "1".

The foreground color register is written into the VRAM internal color register whenever a block write operation is initiated after the foreground color register has been written. Block write enable is set by the BIOS and should not be changed. A driver must read the state of this bit to determine if this feature is enabled. The block write mode will only be activated under the following conditions:

- The write mask is all "1"s.
- The destination compare function is false (Mode = 0).
- The foreground select is set to the foreground color register (Mode = 0).
- The mono select is true (Mode = 0).

- The foreground ALU function is Paint (Function = 7).
- The polygon file mode is disabled.
- The drawing operation is horizontal degree line, horizontal degree short stroke vector (IBM or ATI compatible), scanline, or fill (H1H4 IBM or ATI compatible).

64-Bit Fill Draw - 2MB DRAM

Fill drawing operations under certain conditions will fully utilize the 64-bit data bus. MISC_OPTIONS[11] must be set to "1" by the BIOS to enable this mode. A driver must read the state of this bit to determine if the feature is enabled. For 2MB DRAM video memory configurations, this mode may be enabled all the time.

Accelerated painting will be enabled when all of the following conditions are true:

- Foreground color select is the foreground color register or the background color register.
- If the mono select is pattern then the background color select must be either foreground color register or background color register.
- The mono select is "always 1" or pattern. If the type of operation is ATI then the 8x8 mono pattern mode must be enabled — for this mode only supports destination-aligned mono patterns.
- The CPUDATA bit must be "false" if the type of operation is IBM.
- The FGALU function must be one of: 0, 1, 4, or 7.
- If the mono select is pattern then the BGALU function must be one of: 0, 1, 4, or 7.
- All color compare functions must be set to zero.
- Polygon fill must be disabled.
- The write mask must be all "1"s.
- The pixel size must be 8- or 16-bit.
- The block write function must be disabled.

512KB Support

With 512KB of video memory, the coprocessor will now support the packed 4bpp and 8bpp drawing modes. The 8514/A-compatible min mode is no longer supported.

Split Transfer Cycle

Split transfer cycle support has been added to the ATI68800-6 to allow decoupling of the memory clock and serial clock. This allows higher memory clocks in VRAM

implementations with a corresponding increase in performance. The ATI68800LX does not support this feature.

This feature is enabled if the memory type is set to 5 or 6. See CONFIG_STATUS[6:4] (12EE).

Separate Display/Drawing Pixel Sizes

Both the display and drawing pixel sizes can be written independently. The display pixel size is written when EXT_GE_CONFIG[11] is set to "1". The drawing pixel size is written when EXT_GE_CONFIG[15] is set to "1".

For compatibility, both pixel sizes are written if bits 11 and 15 are both set to "0".

Memory Mapped Registers

Memory mapping of coprocessor registers is supported by the ATI68800LX, ATI68800-6, and ATI68800AX ***mach32*** accelerators. This feature is enabled when bit 32EE[5] is "1". Since the feature is enabled by the BIOS, 32EE[5] should not be changed. A driver must read the state of this bit to determine if it can use the feature.

All of the coprocessor registers may be mapped into the upper 128 dwords of a 4MB linear aperture. The address offset from the start of register space relative to the base of linear aperture is FFF80. Each 16-bit register will map into the lower 16 bits of a 32-bit dword. The upper 16 bits are currently unused and therefore should be ignored.

The 64 dwords from FFE00 to FFEFF are mapped into 64 IBM-compatible registers which are aliased to I/O address 2E8h. The 64 dwords from FFF00 to FFFFF are mapped into 64 ATI-extended registers which are aliased to I/O address 2EEh.

To the extent that I/O registers are sparse for either reads or writes the memory mapped registers will also be sparse. Note that the 128 dwords reserved for registers are not available for memory operations. Examples of both IBM and ATI register mappings are described for your reference in Appendix A.

This page intentionally left blank.

H

ATI68800AX Features

The "AX" is at the high-end of all ***mach32*** graphics accelerators. Like all other ***mach32***, it is compatible with the IBM 8514/A, and with one another. The "AX" is an enhanced ATI68800-6, with support for an additional local bus configuration — Intel's Peripheral Component Interconnect (PCI). Also introduced in the "AX" are five new display modes that provide high-resolution Hi-Color Support using the new ATI68860 palette DAC and 32-bit overlay. The new modes are as follows:

- 800x600x32
- 1024x768x24
- 1024x768x32
- 1280x1024x16
- 1280x1024x24

All ***mach32*** PCI-exclusive configuration registers are described in Chapter 2. Other PCI-related control registers are described in chapters 7, 8, and 9 as follows:

- PCI_CNTL (22EE-RW)
- LOCAL_CNTL (32EE-RW)
- MISC_OPTIONS (36EE-RW)
- APERTURE_BASE (5EEE-RW)
- APERTURE_CNTL (PCI) (6AEE-RW)

This page intentionally left blank.

Index

A

B

C

D

E

F

G

H

I

L

M

O

P

Q

R

S

U

V

W

This page intentionally left blank.

User Response Form

Thank you for taking the time to complete this response form. Because ATI products serve customers with a wide range of experience, we want our documentation to meet your needs. Your response will help us achieve this goal. Using a scale of 1 to 6, with 6 representing the most favorable, and 1 the least favorable, please circle the number that best reflects your opinion.

Name and Organization: _____

Title: _____ **Years in position:** ____

Name of Manual _____ **Release No.** _____

How do you rate the manual, generally?	1	2	3	4	5	6
Is the technical level of the manual appropriate?	1	2	3	4	5	6
Are operating instructions clear and complete?	1	2	3	4	5	6
Are terms and concepts explained clearly?	1	2	3	4	5	6
Is the manual well organized?	1	2	3	4	5	6
Are the tables easy to understand?	1	2	3	4	5	6
Is the artwork clear and easy to understand?	1	2	3	4	5	6
How helpful is the index?	1	2	3	4	5	6
How does this manual compare with other similar manuals you have used?	1	2	3	4	5	6

Comments

What do you suggest to improve this document? _____

_____

What features or techniques that you've seen in other manuals would you like to see in ours?

_____

_____

What type of information would you like to see included or expanded upon?

_____

_____

Are there any errors or omissions in the manual? (please specify):

_____

_____

Place
Postage
Here

ATI Technologies Inc.
3761 Victoria Park Avenue
Scarborough, Ontario
Canada
M1W 3S2

Attn: Technical Publications

Printed in Canada