mach64
Register Reference Guide

ATI-264VT and 3D RAGE™

Technical Reference Manuals

P/N: RRG-G02700 Rev. 0.10

ATI Technologies Inc.
33 Commerce Valley Drive East
Thornhill, Ontario
Canada L3T 7N6

Developer Support:	905-882-2600 ext.6000
Offices:	905-882-2600
Fax:	905-882-2620
BBS:	905-764-9404

P/N: RRG-G02700

REVISION 0.10 Preliminary Draft

Record of Revisions

Release	Date	Description of Changes
0.10	July. 96	Preliminary Release of combined VT/3D RAGE Register guide.

System Publications Index

Technical Reference Manuals

- *mach64* BIOS Kit
  (BIO-G01000)
- *mach64* Programmer's Guide
  (PRG-G01000)
- *mach64* Register Reference Guide
  ATI-264VT and 3D RAGE
  (RRG-G02700)
- *mach64* Graphics Controller
  Specifications ATI-264VT
  (GCS-C02500)
- *mach64* Graphics Controller
  Specifications 3D RAGE
  (GCS-C02700)

Table of Contents

Appendix A VGA BIOS Function Calls

Appendix B Programming PLL Registers in the ATI-264VT

Index

Chapter 1
Overview

Introduction

This manual serves as a reference document, covering the accelerator registers for the *mach64* **ATI-264VT** and **VT2** (i.e., up to revision VT-A4), and the *mach64* **3D RAGE** (GT-A2). The VT and 3D RAGE accelerators are register-compatible with other controllers in the ATI *mach64* accelerator series; therefore, they provide the OEM with a low-cost path to additional multimedia and performance features.

ATI-264VT

In addition to backward-compatibility with ATI's proven family of 2D graphics accelerators, the VT incorporates a video coprocessor to provide superior hardware video acceleration. The VT chip includes many new registers which are used to control and configure multimedia operations such as video capture and playback. New multimedia functionalities include:

- Video scaler for full-screen video playback
- Integrated video line buffer
- Superb filtering for horizontal and vertical scaling
- True Color palette DAC, supporting pixel clock rates to 135 MHz
- Color interpolation during scaling
- Color space conversion to support different image formats
- Video and graphics keying for effective overlay of video and graphics
- Internal dual clock synthesizer

3D RAGE

The 3D RAGE offers all of the 2D and video functionalities of the VT, plus a rich suite of hardware-accelerated 3D rendering and scaling functionalities. The 3D RAGE has incorporated many new registers to support the core 3D features of the chip. New features include:

- Trapezoidal trajectory
- Six texture filtering modes
- Perspectively-correct texture mapping
- Video textures
- Gouraud shading

- Both horizontal and vertical scaling
- Alpha blending
- Fog effects
- Double buffering
- Dithering of limited colors (8 and 16 bpp)
- New pixel formats: RGB332, RGB444, YUV422, YUV444, and Y8

The remainder of this chapter provides a brief overview of the registers classifications and general layout of the manual, and explains the notation conventions used.

Register Classifications

The **VT** accelerator has six major accelerator register classes, in addition to the standard VGA registers, as follows:

- Setup and Control registers
- Accelerator CRTC and DAC registers
- Draw Engine Trajectory registers
- Draw Engine Control registers
- Multimedia registers
- PCI Configuration Space registers
- VGA registers

The **3D RAGE** includes all of the above register classes (with minor modifications as noted), and also includes:

- Scaling and 3D Operations registers

Note that these are general register classes only. Instances where specific bit fields of the same register may belong to different register classes is noted in the descriptions. The following discussion provides a brief overview of each register class.

Setup and Control Registers

Setup and control registers are memory mapped and aliased at an I/O address. Most of these registers are initialized only once at boot time. Setup and control registers are:

- **General I/O Control Registers** — used to configure the General Purpose I/O pins on the accelerator chip.
- **Scratch Pad Registers —** used for general purpose storage for the adapter ROM and for communicating the adapter ROM segment location to host applications. In test modes, these registers are used for chip diagnostics.
- **Bus Control Registers** — used to configure the on-chip bus interface unit.

- **Memory Control Registers** — used to configure the memory interface unit.
- **General/Test Register** — used for chip diagnostics.
- **Configuration Registers** — used for configuring the aperture and reading the current board configuration.
- **Hardware Debug Register**— reserved for debugging hardware on engineering samples.

Accelerator CRTC and DAC Registers

Accelerator CRTC and DAC Registers are memory mapped and aliased at an I/O address. Note that accelerator CRTC registers are not the same as the VGA CRTC registers.

- **Accelerator CRTC Registers** — used to configure the video mode.
- **Clock Control Registers** — used to configure the pixel clock.
- **DAC Control Registers** — used to configure the DAC.
- **Overscan Registers** — used to configure overscan borders.
- **Hardware Cursor Registers** — used to define and move the hardware cursor.

Draw Engine Trajectory Registers

Draw Engine Trajectory Registers are memory mapped. They set up the source and destination trajectories and initiate draw operations.

In the 3D RAGE, a new trajectory is provided – the trapezoid. This new trajectory may be used for the destination, the texture map sources, the 2D source, and the Z source.

- **Destination (and Z) Trajectory Registers** — define the region in which pixels are drawn, where the region may be a rectangular area, a line, or (in the case of the 3D RAGE) a trapezoid shape.
- **Source Trajectory Registers** — define a rectangular region from which pixel data is taken; the pixel data may be used as a monochrome or color pixel source, or a polygon fill mask.

Draw Engine Control Registers

Draw Engine Control Registers are memory mapped. They set up the source pixel data, the draw engine data path, and the destination mixing logic.

- **Host Data Registers** — used for transferring data from the host to the draw engine.
- **Pattern Registers** — used to enable and define fixed patterns.
- **Scissor Registers** — used to define a draw region.
- **Data Path Registers** — used to configure the data path and ALU.
- **Color Compare Registers** — used to configure the source or destination color compare.

- **Command FIFO Status Registers** — used to report the status of the command FIFO.
- **Context Control Registers** — used to load contexts or execute context chains.
- **Draw Engine Composite Control Registers** — abbreviated composites of other draw engine control registers.
- **Draw Engine Status Registers** — used to report the current state of the draw engine.

Multimedia Registers

The VT and 3D RAGE chips have incorporated many new registers which are used for multimedia operations such as video capture and playback. See *Chapter 5* for more detailed descriptions on the following groups of multimedia registers.

- **Overlay Window Registers** — used to specify the overlayed scaling window dimensions and coordinates to be displayed
- **Overlay Scaler Registers** — used to enable scaling and set up the scaling factors
- **General Video Registers** — used to set the video configuration (e.g., video source, format)
- **Capture Registers** — used to initialize the for video capture and to trigger capturing
- **Buffer Registers** — used to define the scaling and video capture buffer requirements
- **VMC Registers** — used to initialize the VMC port for sending and receiving data either during a video capture or as streamed host data.

Scaler and 3D Operations Registers (3D RAGE only)

Scaler Pipe and 3D Operations Registers are memory mapped. For more detailed descriptions on the following groups of registers, see *Chapter 6*:

- **Scaler Pipe Registers** — used to configure the scaler source data, and control any subsequent blending, color conversion and dithering; most of the scaler registers are aliased with certain 3D and texture mapping registers.
- **Texture Mapping Registers** — used to hold the ‘S’ and ‘T’ sample address offsets to the start of the available mipmaps, and configure the associated quadratic interpolators.
- **Color, Z, and Alpha Interpolator Registers** — used to configure the Z buffering and interpolation, the RGB and alpha interpolation, alpha blending and fogging.

PCI Configuration Space Registers

Host Bus Dependent Registers are used in *mach64* accelerators to support bus-specific implementations. For the VT and 3D RAGE, the internal Host Bus interface has been optimized to support the PCI Version 2.1 bus configuration, providing full 32-bit memory and I/O operations. The PCI Configuration Space registers, which determine the host bus configuration during system reset, are summarized in *Chapter 7*.

VGA Registers

The VGA registers in all *mach64* accelerator chips are completely segregated from the accelerator registers and their functions are mutually exclusive. Instead, they provide register-level compatible with the IBM VGA display adapter. *Chapter 8* provides a programmer's overview of the VGA, its memory aperture, and VESA BIOS Extension support. *Chapter 9* describes the display modes and specifications. *Chapter 10* lists the VGA-compatible registers. VGA BIOS function calls are included in the appendix.

Note that the ATI VGA extended registers, which were available on the *mach64* GX family at 1CEh to 1CFh, are not included in the VT and 3D RAGE chips.

How To Find the Registers

Chapter 2 outlines the register memory and I/O mapping for the VT/3D RAGE accelerator chips. The tables in *Chapter 3, Cross Reference* summarize the accelerator registers by class, mnemonic (listed alphabetically by register name), and address. They also include the page number where each register is described.

A detailed listing of the accelerator registers can be found in *Chapter 4, Accelerator Register Reference*. This includes the following major accelerator register classes previously summarized:

- Setup and Control registers
- Accelerator CRTC and DAC registers
- Draw Engine Trajectory registers
- Draw Engine Control registers

In *Chapter 5*, the Multimedia registers are grouped into general register classes, and arranged alphabetically within each general class. In *Chapter 6*, the Scaler and 3D Operations registers are grouped into general register classes, and arranged alphabetically within each general class. The PCI Configuration Space registers are listed in *Chapter 7*.

For details on the standard VGA registers, refer to *VGA Registers* above.

Notation Conventions

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]**
- **Field_Name@Register_Mnemonic**

The following example is the mnemonic for the Configuration Chip ID register:

CONFIG_CHIP_ID

Continuing the above example, the Product Type Code field within the above register occupies bit positions 0 through 15. The examples below describe this field in two ways:

CONFIG_CHIP_ID[15:0]

CFG_CHIP_TYPE@CONFIG_CHIP_ID

The second convention is the preferred one. Note that in the first example, *square brackets* [ ] are used.

Hexadecimal numbers are appended with "h" (Intel assembly-style notation). All other numbers are in decimal. When several signals of identical function are described, the part of the signal name that differs may be shown in parentheses ( ). For example, the four Select signals — SEL0#, SEL1#, SEL2#, and SEL3# — may be represented as SEL(0:3)#

Chapter 2
VT/3D RAGE Register Mapping

Introduction

This chapter provides an overview of the register mapping used for the *mach64* ATI-264VT and 3D RAGE accelerators. The programming model for these accelerators is fully memory mapped. The registers are memory mapped, I/O mapped, or both. In general, the VGA registers are I/O mapped only, the *mach64* draw engine, 3D engine (3D RAGE only), and multimedia registers are memory mapped only, and the remaining registers are I/O mapped with memory mapped register aliases. All registers are 32 bits wide, except DAC_REGS, which are 4x8 bit registers.

I/O mapped registers are selected by their I/O or Memory Mapped (MM) Select, depending on their I/O type, i.e., standard I/O or block I/O (PCI only). For standard I/O type, the I/O base address will usually be 2ECh (1C8h or 1CCh are also supported). For block I/O, the I/O base address can be anywhere within the 64K I/O space.

Registers are numbered from 0xh to FFxh. Readable registers are indicated by an ‘R’, and writable registers are indicated by a ‘W’. Registers not associated with the draw engine are directly readable and writeable, and are numbered from 0xh to 3Fxh. They may also be accessed at I/O addresses aliased to registers offset from the base I/O address. All draw engine registers are memory mapped with Dword offsets between 40xh and FFxh inclusive. These registers are written through a command FIFO and read directly.

Up to 32,768 draw engine contexts may be restored from 64 Dword segments. A 15-bit context load pointer allow contexts to be positioned anywhere in memory (refer to *page 4-102* for more information on context control).

Aperture Modes

The VT and 3D RAGE operate in one of two modes – VGA aperture or linear aperture mode – which may be selected based on the system configuration. The linear aperture mode requires that the linear aperture be enabled, while the VGA aperture mode requires that the VGA portion of the chip be enabled. All registers are mapped relative to the top of the defined memory aperture. The linear aperture mode is optimized for PCI configurations, while the VGA aperture mode is suitable for backwards compatibility to ISA-based systems.

In linear aperture mode, the aperture size is fixed at 8MB. The aperture position is set by the system BIOS at configuration time through the Base Address registers in the PCI configuration space (see *Chapter 7*). In PCI systems, the aperture size and position can also be read from the *CONFIG_CNTL* register (which is read-only).

In previous *mach64* accelerators (e.g. CT), the upper 1KB of the graphics aperture (VGA or linear) was reserved for the controller register space. In the VT and 3D RAGE, this upper 1KB block is known as block 0. The VT and 3D RAGE expand this register space by adding an additional 1KB register block below the original block 0. This new block is referred to as block 1.

The following diagram shows register block 0 and 1 locations relative to a typical linear aperture configuration and system memory:

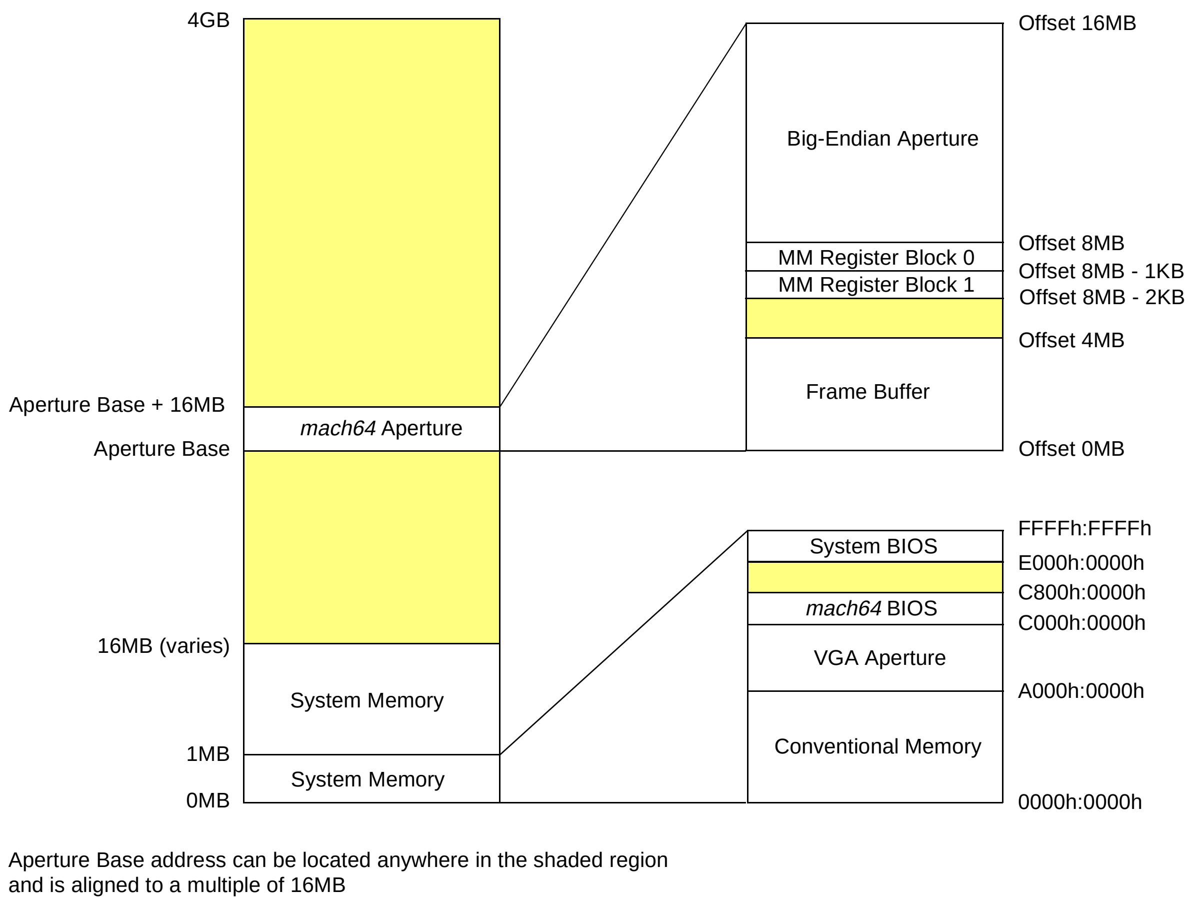


Figure 2.1. Typical Organization of *mach64* Aperture Within Host Address Space (PC-compatible)

In VGA aperture mode, the upper 1KB (or 2KB) of the 64KB aperture at B0000h is reserved for the controller register space. The following table defines the register block offset within the graphics aperture for both VGA and linear apertures:

Mode	Aperture Size	Register Block 0 Offset	Register Block 1 Offset
VGA	64KB	BFC00	BF800
Linear	8MB	7FFC00	7FF800

If the VGA mode is disabled, a 4KB initialization ROM may be enabled. The ROM position and enable is strap or POS selectable from C0000h to FE000h, with the first 2KB fixed and the second 2KB paged. Up to 16 2KB ROM pages may be accessed.

Memory Mapping

All registers not associated with the draw engine, 3D engine (3D RAGE only), and multimedia are I/O mapped (see the next section), with memory mapped register aliases. All draw engine registers are memory mapped with Dword offsets greater than or equal to 40h.

For memory mapped configurations, the reference tables give a block/Dword offset to describe the register's address, using the following notation:

MM:block#_offset

where:

block# – identifies the block that the register belongs to (0 or 1)

offset – is the register Dword offset *within* the associated block.

An underscore (_) separates the block number from the register offset.

Determining the Memory Mapped Address

As an example, using the above notation the OVERLAY_SCALE_CNTL register address is defined as **MM:1_09** (detailed on *page 5-8*). This indicates that this register is located in register block 1 (2K below the top of the aperture) at Dword offset 9h (or byte address 24h). The aperture for the VT or 3D RAGE is 8MB in size. With the base of the aperture located at 8MB in memory, then the absolute physical register address is calculated as:

aperture_base_address + reg_block_offset + reg_byte_offset

For this example (OVERLAY_SCALE_CNTL):

aperture base address	= 800000h
register block 1 offset	= 7ff800h
register byte offset	= 24h
Absolute register address	$\rightarrow$ fff824h

The relative register address (to the base of the aperture) is calculated as follows:

register block offset + register byte offset

Thus, in the above example, the relative address of OVERLAY_SCALE_CNTL $\rightarrow$ 7ff824h

Note that all register offsets that are **not** preceeded by a block number are assumed to be in block 0.

I/O Mapping

All registers not associated with the draw engine, 3D engine and multimedia extensions are I/O mapped and have memory mapped register aliases (as outlined previously). In PCI systems, there are two distinct ways to configure the I/O mapping of these registers – block I/O (also referred to as “relocatable” I/O) and sparse I/O. Both mapping methods are supported by the VT and 3D RAGE.

In block I/O mapping (only on PCI), registers not associated with the draw engine and multimedia map into a continuous block that starts at the I/O base address specified in the PCI configuration registers (which are summarized in *Chapter 6*).

In sparse I/O mapping, the I/O addresses are in the ISA I/O address style. The lower ten bits comprise the I/O base address, and the upper six bits are used for the register selects. In order for Plug and Play (PnP) to be able to configure the I/O space, it requires the ability to have three options for the I/O base addresses for configurable I/O spaces. For sparse I/O mapping, the I/O base address will usually be 2ECh, but I/O base addresses of 1CCh and 1C8h are also supported (as selected by an external strap resistor).

Determining the I/O Base Address

Since the I/O base address may be different depending on the card configuration, it cannot be assumed to be a specific value. The easiest way to obtain the I/O base address is to call the *mach64* BIOS function 12h (*see Appendix A, BIOS Services* of the *mach64 Programmer’s Guide* for more information).

This function also returns the I/O base address type – standard or block (relocatable) I/O. If it is standard, the I/O base address will typically be 2ECh. If it is relocatable, the I/O base address can be any value within a 64KB I/O space. The value is decided by the system to insure that no conflicts exist and is in accord with the “plug and play” specification of a PCI system. Refer to Chapter 2 of the *mach64 Programmer’s Guide* for more information on using this function call.

Determining the Absolute I/O Address

For registers not associated with the draw engine or multimedia, the offset is given for sparse I/O (where applicable) and block I/O selects. Not all of these registers are visible in sparse I/O; however, they are all visible in block I/O.

Note that only Block 0 of the VT or 3D RAGE contains I/O-mappable registers.

Where a **sparse I/O** select is used to describe the register’s address, the physical address can be determined by the following equation:

Absolute I/O address = (I/O select << 10) + I/O base address

To use SCRATCH_REG1 as an example, if the I/O base address = 2ECh and the I/O Select = 11h, the absolute I/O address would be 46ECh.

If **block I/O** is enabled (PCI only), the offset is given as the Dword offset (**BLK:**) from the memory mapped register base address and the block I/O base address. For this case, the equation becomes:

Absolute I/O address = (BLK select << 2) + I/O base address

Using the example above, if the I/O base address = E000h and the Dword offset = 21h (SCRATCH_REG1), the physical I/O address would be E084h.

For some I/O registers, it is necessary to access individual bytes within the 32-bit register (such as DAC_REGS). The I/O select or BLK select must be converted to a byte offset before adding the individual byte offset (0, 1, 2, or 3). For example, to access the DAC_MASK byte of DAC_REGS, the equation is:

byte offset	= I/O select << 10 = 17h << 10 = 5C00h (DAC_REGS)
individual byte offset	= 2 (DAC_MASK@DAC_REGS)
I/O base address	= 2ECh
Absolute I/O address	= byte offset + individual byte offset + I/O base address
	= 5C00h + 2 + 2ECh = 5EEEh

For block I/O, the equation is:

byte offset	= BLK select << 2 = 30h << 2 = C0h (DAC_REGS)
individual byte offset	= 2 (DAC_MASK byte)
I/O base address	= E000h
Absolute I/O address	= byte offset + individual byte offset + I/O base address
	= C0h + 2 + E000h = E0C2h

VGA Registers

The VGA registers are completely segregated from the accelerator registers. They provide compatibility with the IBM VGA Display Adapter. Standard VGA apertures are 64K or 128K for standard VGA modes. VGA I/O and memory spaces are always fixed; they cannot be moved and are not configurable. The VGA aperture has several personalities, but is fixed between A0000h and BFFFFh. VGA I/O space is also fixed at the following locations – 102h, 46E8h (and some aliases), 3C0h through 3CFh (except 3CBh and 3CDh), 3B4h and 3B5h for monochrome display or 3D4h and 3D5h for color display.

An extended VGA aperture type (2x32K) is supported in accelerator mode to guarantee that the host CPU has some method of directly accessing graphics memory.

Chapters 8 to 10 provide reference and programming information for the VGA registers.

Non-Intel Based Memory Mapping

To incorporate the VT or 3D RAGE into a design, non-Intel platforms (such as the Apple Power Macintosh) must conform to the PCI specification. For information on configuring the VT or 3D RAGE in non-Intel environments, refer to Chapter 2 of the *mach64 Programmer's Guide*.

This page intentionally left blank.

Chapter 3
Cross Reference

This section contains tables that list the registers by address, by mnemonic (listed alphabetically) and page number. Use these tables to locate specific registers in the rest of the manual.

In the **3D RAGE**, all of the Scaler Pipe registers (except SCALE_3D_CNTL and SCALE_VACC) are aliased with certain new 3D and Texture Mapping registers, as noted by the duplicate **DWORD** Offsets. Registers which are specific to a certain accelerator or revision are also noted in the following tables.

9

Registers by Address

Register Class	Mnemonic	Read/ Write	I/O Select	DWORD Offset	Page
Accelerator CRTC	CRTC_H_TOTAL_DISP	R/W	0h,1Fh	0_00h	*4-20*
	CRTC_H_SYNC_STRT_WID	R/W	1h	0_01h	*4-21*
	CRTC_V_TOTAL_DISP	R/W	2h	0_02h	*4-22*
	CRTC_V_SYNC_STRT_WID	R/W	3h	0_03h	*4-23*
	CRTC_VLINE_CRNT_VLINE	R/W	4h	0_04h	*4-24*
	CRTC_OFF_PITCH	R/W	5h	0_05h	*4-25*
	CRTC_INT_CNTL	R/W	6h	0_06h	*4-26*
	CRTC_GEN_CNTL	R/W	7h	0_07h	*4-27*
Overscan	OVR_CLR	R/W	8h	0_10h	*4-31*
	OVR_WID_LEFT_RIGHT	R/W	9h	0_11h	*4-32*
	OVR_WID_TOP_BOTTOM	R/W	Ah	0_12h	*4-32*
Hardware Cursor	CUR_CLR0	R/W	Bh	0_18h	*4-33*
	CUR_CLR1	R/W	Ch	0_19h	*4-34*
	CUR_OFFSET	R/W	Dh	0_1Ah	*4-35*
	CUR_HORZ_VERT_POSN	R/W	Eh	0_1Bh	*4-36*
	CUR_HORZ_VERT_OFF	R/W	Fh	0_1Ch	*4-37*
General I/O Control	GP_IO	R/W	-	0_1Eh	*4-4*
	GP_IO_CNTL	R/W	-	0_1Fh	*4-3*
Scratch Pad and Test	SCRATCH_REG0	R/W	10h	0_20h	*4-6*
	SCRATCH_REG1	R/W	11h	0_21h	*4-7*
Clock Control	CLOCK_CNTL	R/W	12h	0_24h	*4-38*
Bus Control	BUS_CNTL	R/W	13h	0_28h	*4-8*
Memory Control	MEM_CNTL	R/W	14h	0_2Ch	*4-10*
	MEM_VGA_WP_SEL	R/W	15h	0_2Dh	*4-12*
	MEM_VGA_RP_SEL	R/W	16h	0_2Eh	*4-13*
DAC Control and Test	DAC_REGS	R/W	17h	0_30h	*4-39*
	DAC_CNTL	R/W	18h	0_31h	*4-40*
General/Test	GEN_TEST_CNTL	R/W	19h	0_34h	*4-14*
Configuration	CONFIG_CNTL	R/W	1Ah	0_37h	*4-16*
	CONFIG_CHIP_ID	R	1Bh	0_38h	*4-17*
	CONFIG_STAT0	R/W	1Ch	0_39h	*4-18*
DAC Control and Test	CRC_SIG	R	1Dh	0_3Ah	*4-42*

Registers by Address

Register Class	Mnemonic	Read/ Write	I/O Select	DWORD Offset	Page
Draw Engine Destination (and Z) Trajectory Control	DST_OFF_PITCH	R/W	-	0_40h	*4-53*
	DST_X	R/W	-	0_41h	*4-55*
	DST_Y	R/W	-	0_42h	*4-57*
	DST_Y_X	W	-	0_43h	*4-58*
	DST_WIDTH	R/W	-	0_44h	*4-54*
	DST_HEIGHT	R/W	-	0_45h	*4-51*
	DST_HEIGHT_WIDTH	W	-	0_46h	*4-52*
	DST_X_WIDTH	W	-	0_47h	*4-56*
	DST_BRES_LNTH (in GT, aliased to 0_51h)	R/W	-	0_48h	*4-46*
	DST_BRES_ERR (LEAD_BRES_ERR, GT)	R/W	-	0_49h	*4-44*
	DST_BRES_INC (LEAD_BRES_INC, GT)	R/W	-	0_4Ah	*4-45*
	DST_BRES_DEC (LEAD_BRES_DEC, GT)	R/W	-	0_4Bh	*4-43*
	DST_CNTL	R/W	-	0_4Ch	*4-48*
	DST_Y_X (alias, in GT only, for 0_43h)	W	-	0_4Dh	*4-58*
	TRAIL_BRES_ERR (GT only)	R/W	-	0_4Eh	*4-59*
	TRAIL_BRES_INC (GT only)	R/W	-	0_4Fh	*4-59*
	TRAIL_BRES_DEC (GT only)	R/W	-	0_50h	*4-59*
	LEAD_BRES_LNTH (GT, aliased to 0_48h)	R/W	-	0_51h	*4-46*
	Z_OFF_PITCH (GT only)	R/W	-	0_52h	*4-60*
	Z_CNTL (GT only)	R/W	-	0_53h	*4-61*
Draw Engine Source Trajectory Control	SRC_OFF_PITCH	R/W	-	0_60h	*4-68*
	SRC_X	R/W	-	0_61h	*4-71*
	SRC_Y	R/W	-	0_62h	*4-73*
	SRC_Y_X	W	-	0_63h	*4-75*
	SRC_WIDTH1	R/W	-	0_64h	*4-69*
	SRC_HEIGHT1	R/W	-	0_65h	*4-64*
	SRC_HEIGHT1_WIDTH1	W	-	0_66h	*4-65*
	SRC_X_START	R/W	-	0_67h	*4-72*
	SRC_Y_START	R/W	-	0_68h	*4-74*
	SRC_Y_X_START	W	-	0_69h	*4-76*
	SRC_WIDTH2	R/W	-	0_6Ah	*4-70*
	SRC_HEIGHT2	R/W	-	0_6Bh	*4-66*
	SRC_HEIGHT2_WIDTH2	W	-	0_6Ch	*4-67*
	SRC_CNTL	R/W	-	0_6Dh	*4-62*
Scaler Pipe (GT only)	SCALE_Y_OFF	R/W	-	0_70h	*6-1*
Texture Mapping (GT only)	TEX_0_OFF	R/W	-	0_70h	*6-8*
	TEX_1_OFF	R/W	-	0_71h	*6-8*
	TEX_2_OFF	R/W	-	0_72h	*6-8*
	TEX_3_OFF	R/W	-	0_73h	*6-8*
	TEX_4_OFF	R/W	-	0_74h	*6-8*
	TEX_5_OFF	R/W	-	0_75h	*6-8*
	TEX_6_OFF	R/W	-	0_76h	*6-8*
	TEX_7_OFF	R/W	-	0_77h	*6-8*
Scaler Pipe (GT only)	SCALE_WIDTH	R/W	-	0_77h	*6-2*
	SCALE_HEIGHT	R/W	-	0_78h	*6-2*
Texture Mapping (GT only)	TEX_8_OFF	R/W	-	0_78h	*6-8*
	TEX_9_OFF	R/W	-	0_79h	*6-8*
	TEX_10_OFF	R/W	-	0_7Ah	*6-8*
	S_Y_INC (aliased to 0_D4h)	R/W	-	0_7Bh	*6-9*
Scaler Pipe (GT only)	SCALE_Y_PITCH (aliased to 0_D4h)	R/W	-	0_7Bh	*6-2*
	SCALE_X_INC (aliased to 0_F0h)	R/W	-	0_7Ch	*6-2*
Color, Z & Alpha Inter. (GT only)	RED_X_INC (aliased to 0_F0h)	R/W	-	0_7Ch	*6-12*
	GREEN_X_INC (aliased to 0_F3h)	R/W	-	0_7Dh	*6-12*
Scaler Pipe (GT only)	SCALE_Y_INC (aliased to 0_F3h)	R/W	-	0_7Dh	*6-2*
	SCALE_VACC	R/W	-	0_7Eh	*6-3*
	SCALE_3D_CNTL	R/W	-	0_7Fh	*6-4*

Registers by Address

Register Class	Mnemonic	Read/ Write	I/O Select	DWORD Offset	Page
Host Data	HOST_DATA[15:0]	W	-	0_80-8Fh	*4-77*
	HOST_CNTL	R/W	-	0_90h	*4-78*
Pattern	PAT_REG0	R/W	-	0_A0h	*4-79*
	PAT_REG1	R/W	-	0_A1h	*4-80*
	PAT_CNTL	R/W	-	0_A2h	*4-81*
Scissor	SC_LEFT	R/W	-	0_A8h	*4-82*
	SC_RIGHT	R/W	-	0_A9h	*4-83*
	SC_LEFT_RIGHT	W	-	0_AAh	*4-84*
	SC_TOP	R/W	-	0_ABh	*4-85*
	SC_BOTTOM	R/W	-	0_ACh	*4-86*
	SC_TOP_BOTTOM	W	-	0_ADh	*4-87*
Data Path	DP_BKGD_CLR	R/W	-	0_B0h	*4-88*
	DP_FRGD_CLR (also DP_FOG_CLR, GT)	R/W	-	0_B1h	*4-89*
	DP_WRITE_MSK	R/W	-	0_B2h	*4-90*
	DP_CHAIN_MSK	R/W	-	0_B3h	*4-91*
	DP_PIX_WIDTH	R/W	-	0_B4h	*4-92*
	DP_MIX	R/W	-	0_B5h	*4-95*
	DP_SRC	R/W	-	0_B6h	*4-97*
Color Compare	CLR_CMP_CLR	R/W	-	0_C0h	*4-98*
	CLR_CMP_MSK	R/W	-	0_C1h	*4-99*
	CLR_CMP_CNTL	R/W	-	0_C2h	*4-100*
FIFO Status	FIFO_STAT	R	-	0_C4h	*4-101*
Context Control	CONTEXT_MSK	R/W	-	0_C8h	*4-102*
	CONTEXT_LOAD_CNTL	R/W	-	0_CBh	*4-103*
Engine Control	GUI_TRAJ_CNTL	R/W	-	0_CCh	*4-104*
Engine Status	GUI_STAT	R	-	0_CEh	*4-106*
Texture Mapping (GT only)	S_X_INC2	R/W	-	0_D0h	*6-8*
	S_Y_INC2	R/W	-	0_D1h	*6-8*
	S_XY_INC2	R/W	-	0_D2h	*6-9*
	S_XINC_START	R/W	-	0_D3h	*6-9*
	S_Y_INC	R/W	-	0_D4h	*6-9*
Scaler Pipe (GT only)	SCALE_Y_PITCH (aliased to 0_7Bh)	R/W	-	0_D4h	*6-2*
Texture Mapping (GT only)	S_START	R/W	-	0_D5h	*6-9*
	T_X_INC2	R/W	-	0_D6h	*6-9*
	T_Y_INC2	R/W	-	0_D7h	*6-10*
	T_XY_INC2	R/W	-	0_D8h	*6-10*
	T_XINC_START	R/W	-	0_D9h	*6-10*
	T_Y_INC	R/W	-	0_DAh	*6-10*
	T_START	R/W	-	0_DBh	*6-10*
	TEX_SIZE_PITCH	R/W	-	0_DCh	*6-11*
Scaler Pipe (GT only)	SCALE_X_INC (aliased to 0_7Ch)	R/W	-	0_F0h	*6-2*
Color, Z & Alpha Inter. (GT only)	RED_X_INC (aliased to 0_7Ch)	R/W	-	0_F0h	*6-12*
	RED_Y_INC	R/W	-	0_F1h	*6-12*
	RED_START	R/W	-	0_F2h	*6-12*
Scaler Pipe (GT only)	SCALE_HACC	R/W	-	0_F2h	*6-7*
	SCALE_Y_INC (aliased to 0_7Dh)	R/W	-	0_F3h	*6-2*
Color, Z & Alpha Inter. (GT only)	GREEN_X_INC (aliased to 0_7Dh)	R/W	-	0_F3h	*6-12*
	GREEN_Y_INC	R/W	-	0_F4h	*6-12*
	GREEN_START	R/W	-	0_F5h	*6-13*
	BLUE_X_INC	R/W	-	0_F6h	*6-13*
Scaler Pipe (GT only)	SCALE_XUV_INC	R/W	-	0_F6h	*6-7*
Color, Z & Alpha Inter. (GT only)	BLUE_Y_INC	R/W	-	0_F7h	*6-13*
	BLUE_START	R/W	-	0_F8h	*6-13*
Scaler Pipe (GT only)	SCALE_UV_HACC	R/W	-	0_F8h	*6-7*

Registers by Address

Register Class	Mnemonic	Read/ Write	I/O Select	DWORD Offset	Page
Color, Z & Alpha Inter. (GT only)	Z_X_INC	R/W	-	0_F9h	6-13
	Z_Y_INC	R/W	-	0_FAh	6-14
	Z_START	R/W	-	0_FBh	6-14
Color, Z & Alpha Inter. (GT only)	ALPHA_X_INC	R/W	-	0_FCh	6-14
	FOG_X_INC	R/W	-	0_FCh	6-14
	ALPHA_Y_INC	R/W	-	0_FDh	6-14
	FOG_Y_INC	R/W	-	0_FDh	6-14
	ALPHA_START	R/W	-	0_FEh	6-14
	FOG_START	R/W	-	0_FEh	6-14
Overlay Window Control	OVERLAY_Y_X_START	R/W	-	1_00h	5-6
	OVERLAY_Y_X_END	R/W	-	1_01h	5-7
	OVERLAY_VIDEO_KEY_CLR	R/W	-	1_02h	5-4
	OVERLAY_VIDEO_KEY_MSK	R/W	-	1_03h	5-5
	OVERLAY_GRAPHICS_KEY_CLR	R/W	-	1_04h	5-1
	OVERLAY_GRAPHICS_KEY_MSK	R/W	-	1_05h	5-2
	OVERLAY_KEY_CNTL	R/W	-	1_06h	5-3
Overlay Scaler	OVERLAY_SCALE_INC	R/W	-	1_08h	5-10
	OVERLAY_SCALE_CNTL	R/W	-	1_09h	5-8
	SCALER_HEIGHT_WIDTH	R/W	-	1_0Ah	5-12
	OVERLAY_TEST	R/W	-	1_0Bh	5-11
	SCALER_THRESHOLD	R/W	-	1_0Ch	5-13
Video Capture	CAPTURE_Y_X	R/W	-	1_10h	5-21
	CAPTURE_HEIGHT_WIDTH	R/W	-	1_11h	5-20
General Video	VIDEO_FORMAT	R/W	-	1_12h	5-16
	VIDEO_CONFIG	R/W	-	1_13h	5-14
Video Capture	CAPTURE_CONFIG	R/W	-	1_14h	5-18
	TRIG_CNTL	R/W	-	1_15h	5-22
	VIDEO_SYNC_TEST	R/W	-	1_16h	5-24
CRTC	EXT_CRTC_GEN_CNTL (VT-A4 only)	R/W	-	1_17h	4-29
VMC Configuration and Status	VMC_CONFIG	R/W	-	1_18h	5-31
	VMC_STATUS	R	-	1_19h	5-32
VMC Command	VMC_CMD	R/W	-	1_1Ah	5-35
	VMC_ARG0	R/W	-	1_1Bh	5-33
	VMC_ARG1	R/W	-	1_1Ch	5-34
	VMC_SNOOP_ARG0	R	-	1_1Dh	5-37
	VMC_SNOOP_ARG1	R	-	1_1Eh	5-38
Video Buffer 0/1	BUF0_OFFSET	R/W	-	1_20h	5-26
	BUF0_PITCH	R/W	-	1_23h	5-27
	BUF1_OFFSET	R/W	-	1_26h	5-29
	BUF1_PITCH	R/W	-	1_29h	5-30
	BUF0_CAP_ODD_OFFSET	R/W	-	1_2Bh	5-25
	BUF1_CAP_ODD_OFFSET	R/W	-	1_2Ch	5-28
VMC Stream Data	VMC_STRM_DATA[i]	R/W	-	1_30-3Fh	5-39
Miscellaneous	HW_DEBUG	R/W	-	1_50h	4-19

Registers by Mnemonic

Register Class	Mnemonic	Read/ Write	I/O Select	DWORD Offset	Page
Color, Z & Alpha Interpolation (GT only)	ALPHA_START	R/W	-	0_FEh	*6-14*
	ALPHA_X_INC	R/W	-	0_FCh	*6-14*
	ALPHA_Y_INC	R/W	-	0_FDh	*6-14*
	BLUE_START	R/W	-	0_F8h	*6-13*
	BLUE_X_INC	R/W	-	0_F6h	*6-13*
	BLUE_Y_INC	R/W	-	0_F7h	*6-13*
Video Buffer 0/1	BUF0_CAP_ODD_OFFSET	R/W	-	1_2Bh	*5-25*
	BUF0_OFFSET	R/W	-	1_20h	*5-26*
	BUF0_PITCH	R/W	-	1_23h	*5-27*
	BUF1_CAP_ODD_OFFSET	R/W	-	1_2Ch	*5-28*
	BUF1_OFFSET	R/W	-	1_26h	*5-29*
	BUF1_PITCH	R/W	-	1_29h	*5-30*
Bus Control	BUS_CNTL	R/W	13h	0_28h	*4-8*
Video Capture	CAPTURE_CONFIG	R/W	-	1_14h	*5-18*
	CAPTURE_HEIGHT_WIDTH	R/W	-	1_11h	*5-20*
	CAPTURE_Y_X	R/W	-	1_10h	*5-21*
Clock Control	CLOCK_CNTL	R/W	12h	0_24h	*4-38*
Color Compare	CLR_CMP_CLR	R/W	-	0_C0h	*4-98*
	CLR_CMP_CNTL	R/W	-	0_C2h	*4-100*
	CLR_CMP_MSK	R/W	-	0_C1h	*4-99*
Configuration	CONFIG_CHIP_ID	R	1Bh	0_38h	*4-17*
	CONFIG_CNTL	R/W	1Ah	0_37h	*4-16*
	CONFIG_STAT0	R/W	1Ch	0_39h	*4-18*
Context Control	CONTEXT_LOAD_CNTL	R/W	-	0_CBh	*4-103*
	CONTEXT_MSK	R/W	-	0_C8h	*4-102*
DAC Control	CRC_SIG	R	1Dh	0_3Ah	*4-42*
Accelerator CRTC	CRTC_GEN_CNTL	R/W	7h	0_07h	*4-27*
	CRTC_H_SYNC_STRT_WID	R/W	1h	0_01h	*4-23*
	CRTC_H_TOTAL_DISP	R/W	0h,1Fh	0_00h	*4-22*
	CRTC_INT_CNTL	R/W	6h	0_06h	*4-26*
	CRTC_OFF_PITCH	R/W	5h	0_05h	*4-25*
	CRTC_V_SYNC_STRT_WID	R/W	3h	0_03h	*4-23*
	CRTC_V_TOTAL_DISP	R/W	2h	0_02h	*4-22*
	CRTC_VLINE_CRNT_VLINE	R/W	4h	0_04h	*4-24*
Hardware Cursor	CUR_CLR0	R/W	Bh	0_18h	*4-33*
	CUR_CLR1	R/W	Ch	0_19h	*4-34*
	CUR_HORZ_VERT_OFF	R/W	Fh	0_1Ch	*4-37*
	CUR_HORZ_VERT_POSN	R/W	Eh	0_1Bh	*4-36*
	CUR_OFFSET	R/W	Dh	0_1Ah	*4-35*
DAC Control	DAC_CNTL	R/W	18h	0_31h	*4-40*
	DAC_REGS	R/W	17h	0_30h	*4-39*
Data Path	DP_BKGD_CLR	R/W	-	0_B0h	*4-88*
	DP_CHAIN_MSK	R/W	-	0_B3h	*4-91*
	DP_FRGD_CLR (also DP_FOG_CLR, GT)	R/W	-	0_B1h	*4-89*
	DP_MIX	R/W	-	0_B5h	*4-95*
	DP_PIX_WIDTH	R/W	-	0_B4h	*4-92*
	DP_SRC	R/W	-	0_B6h	*4-97*
	DP_WRITE_MSK	R/W	-	0_B2h	*4-90*

Registers by Mnemonic

Register Class	Mnemonic	Read/ Write	I/O Select	DWORD Offset	Page
Draw Engine Destination (and Z) Trajectory	DST_BRES_DEC	R/W	-	0_4Bh	*4-43*
	DST_BRES_ERR	R/W	-	0_49h	*4-44*
	DST_BRES_INC	R/W	-	0_4Ah	*4-45*
	DST_BRES_LNTH	R/W	-	0_48, 51h	*4-46*
	DST_CNTL	R/W	-	0_4Ch	*4-48*
	DST_HEIGHT	R/W	-	0_45h	*4-51*
	DST_HEIGHT_WIDTH	W	-	0_46h	*4-52*
	DST_OFF_PITCH	R/W	-	0_40h	*4-53*
	DST_WIDTH	R/W	-	0_44h	*4-54*
	DST_X	R/W	-	0_41h	*4-55*
	DST_X_WIDTH	W	-	0_47h	*4-56*
	DST_Y	R/W	-	0_42h	*4-57*
	DST_Y_X (aliased, in GT, to 0_4Dh)	W	-	0_43h	*4-58*
Accelerator CRTC	EXT_CRTC_GEN_CNTL (VT-A4 only)	R/W	-	1_17h	*4-29*
FIFO Status	FIFO_STAT	R	-	0_C4h	*4-101*
Color, Z & Alpha Interpolation (GT only)	FOG_START	R/W	-	0_FEh	*6-14*
	FOG_X_INC	R/W	-	0_FCh	*6-14*
	FOG_Y_INC	R/W	-	0_FDh	*6-14*
General/Test	GEN_TEST_CNTL	R/W	19h	0_34h	*4-14*
General I/O Control	GP_IO	R/W	-	0_1Eh	*4-4*
	GP_IO_CNTL	R/W	-	0_1Fh	*4-3*
Color, Z & Alpha Interpolation (GT only)	GREEN_START	R/W	-	0_F5h	*6-13*
	GREEN_X_INC	R/W	-	0_7D,F3h	*6-12*
	GREEN_Y_INC	R/W	-	0_F4h	*6-12*
Engine Control	GUI_TRAJ_CNTL	R/W	-	0_CCh	*4-104*
Engine Status	GUI_STAT	R	-	0_CEh	*4-106*
Host Data	HOST_CNTL	R/W	-	0_90h	*4-78*
	HOST_DATA[15:0]	W	-	0_80-8Fh	*4-77*
Miscellaneous	HW_DEBUG	R/W	-	1_50h	*4-19*
Draw Engine Destination (and Z) Trajectory	LEAD_BRES_DEC (GT only)	R/W	-	0_4Bh	*4-43*
	LEAD_BRES_ERR (GT only)	R/W	-	0_49h	*4-44*
	LEAD_BRES_INC (GT only)	R/W	-	0_4Ah	*4-45*
	LEAD_BRES_LNTH (GT only)	R/W	-	0_48, 51h	*4-46*
Memory Control	MEM_CNTL	R/W	14h	0_2Ch	*4-10*
	MEM_VGA_RP_SEL	R/W	16h	0_2Eh	*4-13*
	MEM_VGA_WP_SEL	R/W	15h	0_2Dh	*4-12*
Overlay Window Control	OVERLAY_GRAPHICS_KEY_CLR	R/W	-	1_04h	*5-1*
	OVERLAY_GRAPHICS_KEY_MSK	R/W	-	1_05h	*5-2*
	OVERLAY_KEY_CNTL	R/W	-	1_06h	*5-3*
Overlay Scaler	OVERLAY_SCALE_CNTL	R/W	-	1_09h	*5-8*
	OVERLAY_SCALE_INC	R/W	-	1_08h	*5-10*
	OVERLAY_TEST	R/W	-	1_0Bh	*5-11*
Overlay Window Control	OVERLAY_VIDEO_KEY_CLR	R/W	-	1_02h	*5-4*
	OVERLAY_VIDEO_KEY_MSK	R/W	-	1_03h	*5-5*
	OVERLAY_Y_X_START	R/W	-	1_00h	*5-6*
	OVERLAY_Y_X_END	R/W	-	1_01h	*5-7*
Overscan	OVR_CLR	R/W	8h	0_10h	*4-31*
	OVR_WID_LEFT_RIGHT	R/W	9h	0_11h	*4-32*
	OVR_WID_TOP_BOTTOM	R/W	Ah	0_12h	*4-32*
Pattern	PAT_CNTL	R/W	-	0_A2h	*4-81*
	PAT_REG0	R/W	-	0_A0h	*4-79*
	PAT_REG1	R/W	-	0_A1h	*4-80*
Color, Z & Alpha Interpolation (GT only)	RED_START	R/W	-	0_F2h	*6-12*
	RED_X_INC	R/W	-	0_7C,F0h	*6-12*
	RED_Y_INC	R/W	-	0_F1h	*6-12*

Registers by Mnemonic

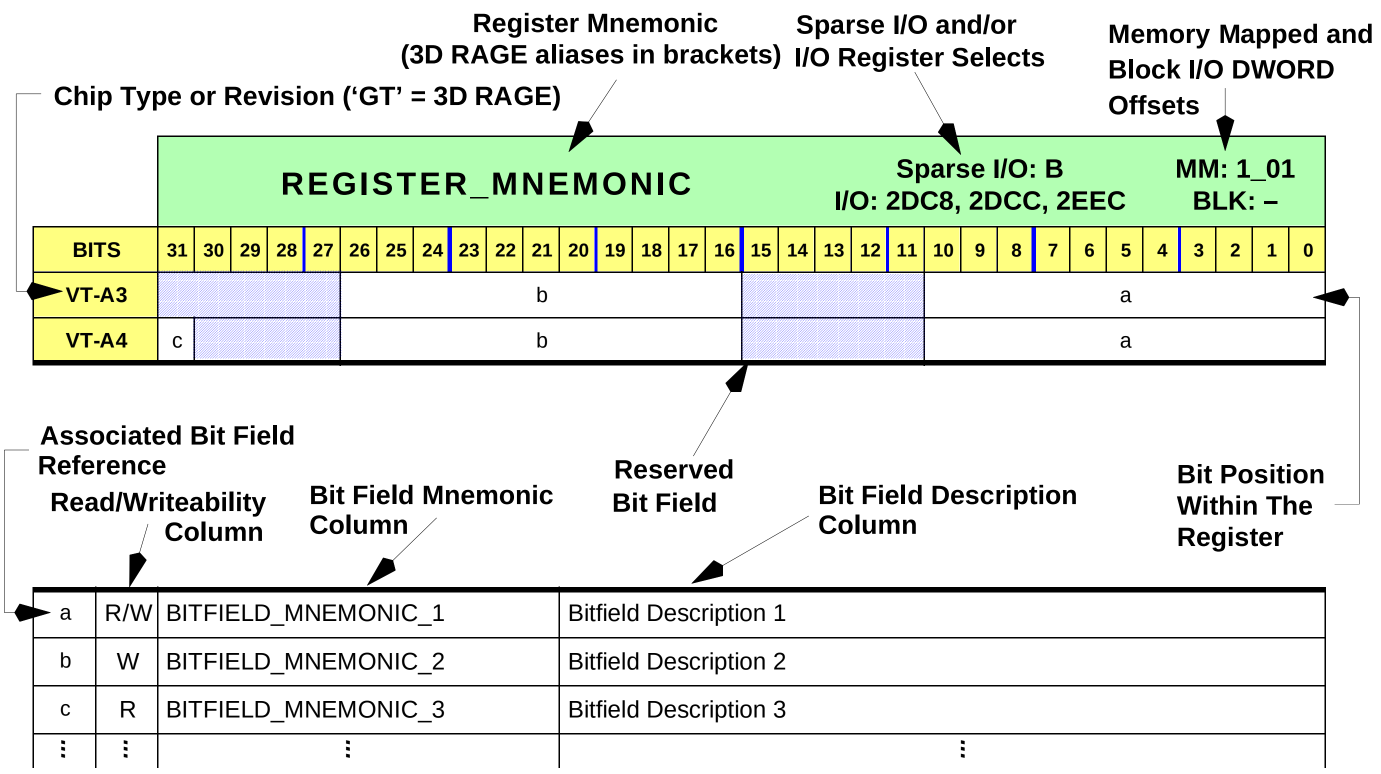
Register Class	Mnemonic	Read/ Write	I/O Select	DWORD Offset	Page
Scaler Pipe (GT only)	SCALE_3D_CNTL	R/W	-	0_7Fh	*6-4*
	SCALE_HACC	R/W	-	0_F2h	*6-7*
	SCALE_HEIGHT	R/W	-	0_78h	*6-2*
	SCALE_UV_HACC	R/W	-	0_F8h	*6-7*
	SCALE_VACC	R/W	-	0_7Eh	*6-3*
	SCALE_WIDTH	R/W	-	0_77h	*6-2*
	SCALE_X_INC	R/W	-	0_7C,F0h	*6-2*
	SCALE_XUV_INC	R/W	-	0_F6h	*6-7*
	SCALE_Y_INC	R/W	-	0_7D,F3h	*6-2*
	SCALE_Y_OFF	R/W	-	0_70h	*6-1*
	SCALE_Y_PITCH	R/W	-	0_7B,D4h	*6-2*
Overlay Scaler	SCALER_HEIGHT_WIDTH	R/W	-	1_0Ah	*5-12*
	SCALER_THRESHOLD	R/W	-	1_0Ch	*5-13*
Scratch Pad and Test	SCRATCH_REG0	R/W	10h	0_20h	*4-6*
	SCRATCH_REG1	R/W	11h	0_21h	*4-7*
Scissor	SC_BOTTOM	R/W	-	0_ACh	*4-86*
	SC_LEFT	R/W	-	0_A8h	*4-82*
	SC_LEFT_RIGHT	W	-	0_AAh	*4-84*
	SC_RIGHT	R/W	-	0_A9h	*4-83*
	SC_TOP	R/W	-	0_ABh	*4-85*
	SC_TOP_BOTTOM	W	-	0_ADh	*4-87*
Draw Engine Source Trajectory	SRC_CNTL	R/W	-	0_6Dh	*4-62*
	SRC_HEIGHT1	R/W	-	0_65h	*4-64*
	SRC_HEIGHT1_WIDTH1	W	-	0_66h	*4-65*
	SRC_HEIGHT2	R/W	-	0_6Bh	*4-66*
	SRC_HEIGHT2_WIDTH2	W	-	0_6Ch	*4-67*
	SRC_OFF_PITCH	R/W	-	0_60h	*4-68*
	SRC_WIDTH1	R/W	-	0_64h	*4-69*
	SRC_WIDTH2	R/W	-	0_6Ah	*4-70*
	SRC_X	R/W	-	0_61h	*4-71*
	SRC_X_START	R/W	-	0_67h	*4-72*
	SRC_Y	R/W	-	0_62h	*4-73*
	SRC_Y_START	R/W	-	0_68h	*4-74*
	SRC_Y_X	W	-	0_63h	*4-75*
	SRC_Y_X_START	W	-	0_69h	*4-76*
Texture Mapping (GT only)	S_START	R/W	-	0_D5h	*6-9*
	S_X_INC2	R/W	-	0_D0h	*6-8*
	S_XINC_START	R/W	-	0_D3h	*6-9*
	S_XY_INC2	R/W	-	0_D2h	*6-9*
	S_Y_INC	R/W	-	0_7B,D4h	*6-9*
	S_Y_INC2	R/W	-	0_D1h	*6-8*
	TEX_[0-10]_OFF	R/W	-	0_70-7Ah	*6-8*
	TEX_SIZE_PITCH	R/W	-	0_DCh	*6-11*
Draw Engine Destination (and Z) Trajectory (GT only)	TRAIL_BRES_ERR	R/W	-	0_4Eh	*4-59*
	TRAIL_BRES_INC	R/W	-	0_4Fh	*4-59*
	TRAIL_BRES_DEC	R/W	-	0_50h	*4-59*
Video Capture	TRIG_CNTL	R/W	-	1_15h	*5-22*
Texture Mapping (GT only)	T_START	R/W	-	0_DBh	*6-10*
	T_X_INC2	R/W	-	0_D6h	*6-9*
	T_XINC_START	R/W	-	0_D9h	*6-10*
	T_XY_INC2	R/W	-	0_D8h	*6-10*
	T_Y_INC	R/W	-	0_DAh	*6-10*
	T_Y_INC2	R/W	-	0_D7h	*6-10*
General Video	VIDEO_FORMAT	R/W	-	1_12h	*5-16*
	VIDEO_CONFIG	R/W	-	1_13h	*5-14*

Registers by Mnemonic					
Register Class	**Mnemonic**	**Read/ Write**	**I/O Select**	**DWORD Offset**	**Page**
Video Capture	VIDEO_SYNC_TEST	R/W	-	1_16h	*5-24*
VMC Command	VMC_ARG0	R/W	-	1_1Bh	*5-33*
	VMC_ARG1	R/W	-	1_1Ch	*5-34*
VMC Config / Status	VMC_CONFIG	R/W	-	1_18h	*5-31*
VMC Command	VMC_CMD	R/W	-	1_1Ah	*5-35*
	VMC_SNOOP_ARG0	R	-	1_1Dh	*5-37*
	VMC_SNOOP_ARG1	R	-	1_1Eh	*5-38*
VMC Config / Status	VMC_STATUS	R	-	1_19h	*5-32*
Draw Engine Destin. (and Z) Trajectory	Z_CNTL (GT only)	R/W	-	0_53h	*4-61*
	Z_OFF_PITCH (GT only)	R/W	-	0_52h	*4-60*
Color, Z & Alpha Interpolation (GT only)	Z_START	R/W	-	0_FBh	*6-14*
	Z_X_INC	R/W	-	0_F9h	*6-13*
	Z_Y_INC	R/W	-	0_FAh	*6-14*

Chapter 4
Accelerator Register Reference

Chapter 4 serves as a reference for the ATI-264VT and 3D RAGE accelerator registers, including the following general register classes: Setup and Control, Accelerator CRTC and DAC, Draw Engine Trajectory and Draw Engine Control registers. Multimedia, 3D, PCI Configuration Space, and VGA registers are detailed in subsequent chapters. All of the register classes are summarized in *Chapter 1*.

The table below describes the layout of all Accelerator Register Reference Tables within Chapter 4.



BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT-A3	▨	▨	▨	▨	▨	b											▨	▨	▨	▨	▨	a										
VT-A4	c	▨	▨	▨	▨	b											▨	▨	▨	▨	▨	a										

a	R/W	BITFIELD_MNEMONIC_1	Bitfield Description 1
b	W	BITFIELD_MNEMONIC_2	Bitfield Description 2
c	R	BITFIELD_MNEMONIC_3	Bitfield Description 3
⋮	⋮	⋮	⋮

All addresses are in hexadecimal, according to the following notation:

Sparse I/O: If applicable, indicates the upper six bits of the standard 16-bit I/O address when using sparse I/O ('B' hex in the above example). The upper six bits are used for the register selects, while the lower 10 bits comprise the I/O base address.

I/O: Indicates the absolute sparse I/O address (if applicable), dependent on the I/O base address used for sparse I/O mapping – 1C8h, 1CCh or 2ECh (typically, the base address will be 2ECh). In the above example, the absolute sparse I/O address is 2DC8h, 2DCCh or 2EECh.

BLK:	Represents the block I/O DWORD offset, if supported.
MM:	Represents the block/DWORD offset for memory mapped configurations, using the following notation:

MM:block#_offset

where:

block# – identifies the block that the register belongs to (0 or 1)

offset – is the register DWORD offset *within* the associated block.

An underscore (_) separates the block number from the register offset.

In the above example, the memory mapped register address is defined as 1_01. Therefore, this register would be located in register block 1 (2K below the top of the aperture) at DWORD offset 9h (or byte address 24h).

See *Chapter 2, VT/3D RAGE Register Mapping* for further information on calculating memory mapped and I/O mapped addresses.

Note that the value of all reserved bits is not guaranteed on read-back. Host applications should mask out all reserved bits in these cases and not rely on their results.

Setup and Control Registers

General I/O Control

GP_IO_CNTL	Sparse I/O:– I/O:–	MM: 0_1F BLK: 1F

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

a	R/W	GP_IO_MODE	Indicates the TARGET for the General Purpose IO 0 = VIDEO_IO 1 = VMC 2 = VFC 3 = ENG_DIAG 4 = GP_IO 5-15 = Reserved
b	R/W	GP_IO_EN	0 = Disable (Default = 0) 1 = Enable

Description

This register is used to configure the General Purpose IO bus, which can be used as a video IO interface in the VT. This bus allows different modes to be supported – VFC, VMC, and DVS (Digital Video Stream) modes.

Usage

GP_IO_MODE specifies the general mode of operation of the General Purpose IO bus, while GP_IO_EN provides the bus enable/disable function.

See Also

GP_IO on *page 4-4*

	GP_IO																															
	Sparse I/O: 1E I/O: 79C8, 79CC, 7AEC																												MM: 0_1E BLK: 1E			
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT/GT	ff	ee	dd	cc	bb	aa	z	y	x	w	v	u	t	s	r	q	p	o	n	m	l	k	j	i	h	g	f	e	d	c	b	a

a	R/W	GP_IO_0	Write/Read (DIR0 = output/input)Pin: DATA(0)
b	R/W	GP_IO_1	Write/Read (DIR1 = output/input)Pin: DATA(1)
c	R/W	GP_IO_2	Write/Read (DIR2 = output/input)Pin: DATA(2)
d	R/W	GP_IO_3	Write/Read (DIR3 = output/input)Pin: DATA(3)
e	R/W	GP_IO_4	Write/Read (DIR4 = output/input)Pin: DATA(4)
f	R/W	GP_IO_5	Write/Read (DIR5 = output/input)Pin: DATA(5)
g	R/W	GP_IO_6	Write/Read (DIR6 = output/input)Pin: DATA(6)
h	R/W	GP_IO_7	Write/Read (DIR7 = output/input)Pin: DATA(7)
i	R/W	GP_IO_8	Write/Read (DIR8 = output/input)Pin: ESYNC
j	R/W	GP_IO_9	Write/Read (DIR9 = output/input)Pin: EVIDEO
k	R/W	GP_IO_A	Write/Read (DIRA = output/input)Pin: BLANKB
l	R/W	GP_IO_B	Write/Read (DIRB = output/input)Pin: SNRDYB
m	R/W	GP_IO_C	Write/Read (DIRC = output/input)Pin: SAB
n	R/W	GP_IO_D	Write/Read (DIRD = output/input)Pin: EDCLK
o	R/W	GP_IO_E	Write/Read (DIRE = output/input)Pin: VMCMASK
p	R/W	GP_IO_F	Write/Read (DIRF = output/input)Pin: DCLK
q	R/W	GP_IO_DIR_0	GP IO Direction: 0 = Input1 = Output
r	R/W	GP_IO_DIR_1	GP IO Direction: 0 = Input1 = Output
s	R/W	GP_IO_DIR_2	GP IO Direction: 0 = Input1 = Output
t	R/W	GP_IO_DIR_3	GP IO Direction: 0 = Input1 = Output
u	R/W	GP_IO_DIR_4	GP IO Direction: 0 = Input1 = Output
v	R/W	GP_IO_DIR_5	GP IO Direction: 0 = Input1 = Output
w	R/W	GP_IO_DIR_6	GP IO Direction: 0 = Input1 = Output
x	R/W	GP_IO_DIR_7	GP IO Direction: 0 = Input1 = Output
y	R/W	GP_IO_DIR_8	GP IO Direction: 0 = Input1 = Output
z	R/W	GP_IO_DIR_9	GP IO Direction: 0 = Input1 = Output
aa	R/W	GP_IO_DIR_A	GP IO Direction: 0 = Input1 = Output
bb	R/W	GP_IO_DIR_B	GP IO Direction: 0 = Input1 = Output
cc	R/W	GP_IO_DIR_C	GP IO Direction: 0 = Input1 = Output
dd	R/W	GP_IO_DIR_D	GP IO Direction: 0 = Input1 = Output
ee	R/W	GP_IO_DIR_E	GP IO Direction: 0 = Input1 = Output
ff	R/W	GP_IO_DIR_F	GP IO Direction: 0 = Input1 = Output

Description

This register specifies the data/direction for each pin (GPIO[F:0]) of the General Purpose IO bus.

Usage

Refer to the ATI-264VT Graphics Controller Specification for details on the typical pin configurations used to support the various operational modes (VFC, DVS, VMC, etc.).

See Also

GP_IO_CNTL on *page 4-3*

Scratch Pad

	SCRATCH_REG0																							Sparse I/O: 10 I/O: 41C8, 41CC, 42EC						MM: 0_20 BLK: 20		
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	a																															

a	R/W	SCRATCH_REG0	Scratch pad 0

Description

SCRATCH_REG0 is a general purpose storage register. The scratch pad registers (0 and 1) may be used to allow two decoupled programs to exchange information. Typically they are used by the BIOS to pass configuration information to drivers or for BIOS data storage.

Usage

Only the adapter BIOS should use this register.

See Also

SCRATCH_REG1 on *page 4-7*

mach64 Programmer's Guide:

- *Advanced Topics: Boot-time Initialization*

	SCRATCH_REG1																			Sparse I/O: 11 I/O: 45C8, 45CC, 46EC								MM: 0_21 BLK: 21				
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	a																															

a	R/W	SCRATCH_REG1	Scratch pad 1

Description

SCRATCH_REG1 is a general purpose storage register. The scratch pad registers (0 and 1) may be used to allow two decoupled programs to exchange information. Typically they are used by the BIOS to pass configuration information to drivers or for BIOS data storage.

Usage

Only the adapter BIOS should write to this register. Applications must read it to determine the adapter BIOS segment location.

See Also

SCRATCH_REG0 on *page 4-6*

***mach64* Programmer's Guide:**

- *Advanced Topics: Boot-time Initialization*

Bus Control

BUS_CNTL	Sparse I/O: 13 I/O: 4DC8, 4DDC, 4EEC	MM: 0_28 BLK: 28

BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	o		n	m	l	▨	▨	▨	k	j	i	h	g				f	e	▨	d	c				b				a			

a	R/W	BUS_WS	Bus wait states (Default = F)
b	R/W	BUS_ROM_WS	ROM access wait states (Default = F)
c	R/W	BUS_ROM_PAGE	ROM page select (Default = X)
d	R/W	BUS_ROM_DIS	ROM disable (Default = 0)
e	R/W	BUS_DAC_SNOOP_EN	Enables DAC snooping
f	R/W	BUS_PCI_RETRY_EN	Allow retry for PCI (Default = 0) 0 = Normal operation 1 = enable retry cycle in PCI
g	R/W	BUS_FIFO_WS	Maximum number of wait states that the FIFO can generate when full. If the maximum is exceeded the BUS_FIFO_ERR_INT flag will be set. Fh – If FIFO is full then write will complete once a FIFO entry is available
h	R/W	BUS_FIFO_ERR_INT_EN	Command FIFO error interrupt enable * (Default = 0)
i	R	BUS_FIFO_ERR_INT	Command FIFO error interrupt * (active high)
	W	BUS_FIFO_ERR_AK	Command FIFO error acknowledge * (1 -> reset the interrupt)
j	R/W	BUS_HOST_ERR_INT_EN	Command FIFO host data error interrupt enable * (Default = 0)
k	R	BUS_HOST_ERR_INT	Command FIFO host data error interrupt * (active high)
	W	BUS_HOST_ERR_AK	Command FIFO host data error acknowledge * (1 -> reset the interrupt)
l	R/W	BUS_EXT_REG_EN	VT Extended Register Block 1 Enable (Default = 0) 0 = Disable VT extended register block 1 (memory page enabled) 1 = Enable VT extended register block 1
m	R/W	BUS_PCI_MEMW_WS	Enables 1 wait states for memory writes * (Default = 0) 0 = 0 wait state 1 = 1 wait state
n	R/W	BUS_BURST	Enable burst write transfers (Default = 0) 0 = write burst transfers disabled 1 = write bursts enabled
o	R/W	BUS_RDY_READ_DLY	Bus memory read RDY signal delay (Default = 2) 0 = (reserved) 1 = no RDY delay 2 = RDY delayed 1 memory clock 3 = (reserved)

Description

BUS_CNTL is used for configuring the on-chip bus interface, controlling error condition interrupts, and configuring portions of the DAC interface unit (when accessing DAC registers).

Usage

Error condition flags that generate hard interrupts should be used only for software debugging and not be included in the final retail software.

DAC snooping allows DAC shadowing devices to monitor accesses to DAC registers on the graphics controller card.

Other control bits in this register should be used only by the adapter ROM at boot-time.

See Also

***mach64* Programmer's Guide:**

- *Advanced Topics: Interrupts*
- *Advanced Topics: Boot-time Initialization*

Memory Control

	MEM_CNTL																								**Sparse I/O: 14** **I/O: 51C8, 51CC, 52EC**						**MM: 0_2C** **BLK: 2C**	
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT/GT	▨	▨	▨	n		m			▨	l	k	j		i	h		g		f	e		d		c		b				a		

a	R/W	MEM_SIZE	Memory size: * (Default = 0) 000 = 512 Kbyte 001 = 1 MByte 010 = 2 MByte 011 = 4 MByte 1XX = (reserved)
b	R/W	MEM_REFRESH	Refresh Cycle Length, DRAM and SDRAM 0000 = 1 clock, through to 1110 1111 = Refresh disable
c	R/W	MEM_CYC_LNTH_AUX	SDRAM Actv to Cmd cycle length: 00 = 4 clocks (Default) 01 = 3 clocks 10 = 2 clocks 11 = 1 clock
d	R/W	MEM_CYC_LNTH	DRAM non-page memory cycle length: 00 = 3 clock RAS, 3 clock CAS (Default) 01 = 3 clock RAS, 2 clock CAS 10 = 2 clock RAS, 3 clock CAS 11 = 2 clock RAS, 2 clock CAS or... SDRAM Pre to Actv cycle length: 00 = 4 clocks (Default) 01 = 3 clocks 10 = 2 clocks 11 = 1 clock
e	R/W	MEM_REFRESH_RATE	Memory refresh rate: 00 = 1 refresh per 1024 MCLK cycles 01 = 1 refresh per 768 MCLK cycles 10 = 1 refresh per 512 MCLK cycles 11 = 1 refresh per 256 MCLK cycles, (Default)
f	R/W	DLL_RESET	0 = no action 1 = Relocks DLL when this bit transitions from 0 to 1
g	R/W	MEM_ACTV_PRE	SDRAM Active to Precharge Minimum Latency 00 = 4 clocks, (Default) 01 = 3 clocks 10 = 6 clocks 11 = 5 clocks
h	R/W	DLL_GAIN_CNTL	SDRAM clock DLL charge pump gain control: 00 = Maximum gain 01 = Level 1 10 = Level 2 (*Recommended) 11 = Minimum gain
i	R/W	MEM_SDRAM_RESET	Invoke SDRAM Reset on transition from 0 to 1 0 = Normal 1 = Reset * sends sequence to SDRAM consisting of PALL, 8 refresh, MRS
j	R/W	MEM_TILE_SELECT	* Must be set to ‘00’

MEM_CNTL

Sparse I/O: 14 **MM: 0_2C**
I/O: 51C8, 51CC, 52EC **BLK: 2C**

BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	▨	▨	▨	n		m			▨	l	k	j		i	h		g		f	e		d		c		b				a		

k	R/W	LOW_LATENCY_MODE	Accept requests as late as possible (i.e. low latency mode)
l	R/W	CDE_PULLBACK	Pulls back the CDE to the memory controller up to the horizontal sync.
m	R/W	MEM_PIX_WIDTH	Big endian memory aperture pixel width. 0 = 1 bpp 1 = 4 bpp 2 = 8 bpp 3 = 15 bpp (5,5,5) 4 = 16 bpp (5,6,5) 5 = 24 bpp 6 = 32 bpp 7 = (reserved)
n	R/W	MEM_OE_SELECT	Selects the function of the OEb(0) pin:* 00 = SDRAM clk (Default) 01 = DRAM 10 = MCLK (test mode) 11 = XCLK (test mode) *Note: Any setting other than '00' will break the DLL loop lock and cause SDRAM to be nonfunctional when SDRAM

Description

MEM_CNTL is for configuring the on-chip memory interface unit.

Usage

This register is normally configured only by the adapter ROM during the power-up initialization. Applications should touch only the MEM_BNDRY and MEM_BNDRY_EN fields for relocating the memory boundary between the accelerator and VGA.

See Also

mach64 Programmer's Guide:

- *Linear Aperture: VGA Interaction*
- *Advanced Topics: Boot-time Initialization*

MEM_VGA_WP_SEL — **Sparse I/O: 15** — **I/O: 55C8, 55CC, 56EC** — **MM: 0_2D** — **BLK: 2D**

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

a	R/W	MEM_VGA_WPS0	Write page pointer for lower 32 KByte aperture into 8 MByte video memory.
b	R/W	MEM_VGA_WPS1	Write page pointer for upper 32 KByte aperture into 8 MByte video memory.

Description

MEM_VGA_WP_SEL contains the two write page pointers used for the two small 32K apertures at 0xA000 and 0xA800. Pages are selectable only on 32K boundaries. These write pages are independent of the read pages.

Apertures exist only in accelerator modes, and only if CFG_MEM_VGA_AP_EN@CONFIG_CNTL is set. VGA apertures are not supported if CFG_BUS_TYPE = PCI. A 4M or 8M linear aperture must be used for PCI bus implementation.

Usage

This register is needed only when writing to the small apertures. Small apertures are required only if the big linear aperture is not available. The big linear aperture may not be available on ISA configurations.

See Also

CONFIG_CNTL on *page 4-16*

MEM_VGA_RP_SEL on *page 4-13*

mach64 Programmer's Guide:

- *Getting Started: Linear Aperture vs. VGA Aperture*

	MEM_VGA_RP_SEL	Sparse I/O: 16 I/O:59C8, 59CC, 5AEC	MM: 0_2E BLK: 2E

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

a	R/W	MEM_VGA_RPS0	Read page pointer for lower 32 KByte aperture into 8 MByte video memory.
b	R/W	MEM_VGA_RPS1	Read page pointer for upper 32 KByte aperture into 8 MByte video memory.

Description

MEM_VGA_RP_SEL contains the two read page pointers used for the two small 32K apertures at 0xA000 and 0xA800. Pages are selectable only on 32K boundaries. These read pages are independent of the write pages.

Apertures exist only in accelerator modes, and only if CFG_MEM_VGA_AP_EN@CONFIG_CNTL is set. VGA apertures are not supported if CFG_BUS_TYPE = PCI. A 4M or 8M linear aperture must be used for PCI bus implementation.

Usage

This register is needed only when writing to the small apertures. Small apertures are required only if the big linear aperture is not available. The big linear aperture may not be available on ISA configurations.

See Also

CONFIG_CNTL on *page 4-16*

MEM_VGA_WP_SEL on *page 4-12*

***mach64* Programmer's Guide:**

- *Getting Started: Linear Aperture vs. VGA Aperture*

General/Test

GEN_TEST_CNTL	Sparse I/O: 19	MM: 0_34
	I/O: 65C8, 65CC, 66EC	BLK: 34

BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT-A3/A4N /GT	▨	▨	m						▨	▨	k	j	i				▨	▨	h		▨	▨	▨	g	f	▨	e	d	c	b	▨	a
VT-A4S	▨	▨	m						l		k	j	i				▨	▨	h		▨	▨	▨	g	f	▨	e	d	c	b	▨	a

a	R/W	GEN_GIO2_DATA_OUT	GIO2 pin data output
b	R/W	GEN_GIO3_DATA_OUT	GIO3 pin data output
c	R	GEN_GIO2_DATA_IN	GIO2 pin data input
d	R/W	GEN_GIO2_EN	0 = GIO2 pin disabled or used for alternate purpose (Default) 1 = GIO2 pin enabled, pin controlled by this register
e	R/W	GEN_GIO2_WRITE	GIO2 pin output enable (Default = 0, i.e. read enable)
f	R/W	GEN_CUR_ENABLE	Enables hardware cursor * (Default = 0)
g	R/W	GEN_GUI_RESETB	Resets GUI Engine on high to low transition (Default = 0)
h	R/W	GEN_TEST_VECT_MODE	Test Vector Mode (Default = 0) 0 = Normal 1 = Output Current Limit 2 = Connectivity Test Mode 3 = IDDQ Test Mode
i	R/W	GEN_TEST_MODE	Enable test modes 0000 = Test mode disabled 0001 = CRTC (VGACRTC) test enabled 0010 = Graphics controller (VGAGC) test enabled 0011 = Attribute controller (VGAATTR) test enabled 0100 = Display address generator (DADDRGEN) test enabled 0101 = GUI engine address gen. test 0110 = Command FIFO test (Lock the FIFO) 0111 = Reserved 1000 = Ring oscillator test 1001 = Delay path test 1010 = Register block test 1011 = PLL test 1100 = Palette test 1101 = DAC test 1110 = RAMDAC functional test 1111 = CRC full speed test (24 bit)
j	R/W	GEN_TEST_CNT_EN	Enables the Scan Counter * (Default to 0)
k	R/W	GEN_CRC_EN	Enables the CRC signature block * (Default to 0)
l	R/W	GEN_DELAY_MUX	Mux out signals from 4 delay paths 00 = delaypath_1 01 = delaypath_2 10 = delaypath_3 11 = delaypath_4
m	W	GEN_TEST_CNT_VALUE	Scan Counter value *

Description

The GEN_TEST_CNTL register is used for general control and diagnostic control. Bits 0-5 control general purpose I/O pins (normally used for EEPROM). Bit 7 enables the hardware cursor. Bit 8 resets the GUI engine. Bits 16-19 enable various test modes of the ASIC. Bit 20 is used to enable the loading of a scan count value which is stored in bits 24-29. Bit 21 enables the cyclic redundancy checker (CRC).

Usage

EEPROM access, DAC configuration, and memory configuration should be touched only by the adapter BIOS. Similarly, diagnostic fields should be touched only by diagnostic programs.

Application level programs should touch only GEN_CUR_EN and GEN_GUI_EN.

See Also

mach64 **Programmer's Guide:**

- *Engine Initialization: FIFO Queue: Resetting the FIFO*
- *Engine Operations: Miscellaneous Operations: Hardware Cursor*
- *Advanced Topics: Boot-time Initialization*
- *Advanced Topics: Accessing the EEPROM*
- *Advanced Topics: DAC Programming*

Configuration

CONFIG_CNTL — **Sparse I/O: 1A** **MM: 0_37**
I/O: 69C8, 69CC, 6AEC **BLK: 37**

BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT	▨	▨	e	e	e	e	e	e	▨	▨	▨	▨	d	▨	▨	▨	▨	▨	c	c	c	c	c	c	c	c	c	c	▨	b	a	a
GT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	d	▨	▨	▨	▨	▨	c	c	c	c	c	c	c	c	▨	▨	▨	b	a	a

a	R	CFG_MEM_AP_SIZE	Linear memory aperture size: (Default = 2) 2 = 2x8 MByte apertures others = (reserved)
b	R/W	CFG_MEM_VGA_AP_EN	Register mapping to VGA aperture (Default = 0) 0 = Memory mapped registers not in VGA aperture 1 = Memory mapped registers available in VGA aperture
c	R	CFG_MEM_AP_LOC	Linear memory aperture location on 16MB boundary (for VT, bits 5:0 = 00)
d	R/W	CFG_VGA_DIS	VGA disable: (Default = 0) 0 = enable VGA if CFG_VGA_EN@CONFIG_STAT0 = 1 1 = disable VGA
e	R/W	CDE_WINDOW	Window time (in MCLKS) to allow unrestricted memory service to non-display requests

Description

CONFIG_CNTL is used to configure the linear memory aperture and for soft configuration of multiple *mach64* systems. In PCI systems, the aperture size (CFG_MEM_AP_SIZE) is always set to 2x8 MB, and the location (CFG_MEM_AP_LOC) is fixed by the PCI configuration space (see *Chapter 7*). These two fields of the CONFIG_CNTL register are read-only for PCI systems.

Usage

Aperture configuration should be done in the adapter BIOS only, during an aperture service function call. Configuration data is stored in non-volatile memory. Both CFG_CARD_ID and CFG_VGA_DIS are touched only in the adapter ROM on power-up to configure the board for multiple *mach64* usage.

For the **3D RAGE**, all offset registers are expanded to allow 8 Meg pointers, with 64-bit granularity. Texture map pointers must have byte granularity.

See Also

mach64 Programmer’s Guide:

- *Advanced Topics: Boot-time Initialization*

CONFIG_CHIP_ID	Sparse I/O: 1B I/O: 6DC8, 6DCC, 6EEC	MM: 0_38 BLK: 38

BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	e		d			c			b								a															

a	R	CFG_CHIP_TYPE	Product type code (5654h = 'VT'; 4754h = '3D RAGE)
b	R	CFG_CHIP_CLASS	Product class code (00h) – (80h when VGA disabled)
c	R	CFG_CHIP_MAJOR	Major ASIC revision number (A=0)
d	R	CFG_CHIP_FND_ID	ASIC foundry ID (000=SGS, 001=NEC, 010=KSC)
e	R	CFG_CHIP_MINOR	Minor ASIC revision number

Description

CONFIG_CHIP_ID is a read-only register. It returns the revision details of the queried chip. CFG_CHIP_TYPE is an alphanumeric code consisting of two ASCII codes; for example, “V” “T” would be “56” and “54” respectively; the remaining registers are numeric codes. VT and 3D RAGE chip IDs are listed in the table below:

ASIC Identification			
ASIC Designation	**CFG_CHIP_TYPE**	**CFG_CHIP_CLASS**	**ASIC ID**
VT	5654h	00h	See ***Usage*** below
3D RAGE	4754h	00h	See ***Usage*** below

Usage

This register is used for chip revision identification. Bits 31:24 of CONFIG_CHIP_ID (CFG_CHIP_MAJOR, CFG_CHIP_FND_ID, and CFG_CHIP_MINOR) form the ASIC ID. The ASIC ID fields also appear in the PCI configuration space (see *Chapter 7*).

The assigned ASIC IDs for the VT and 3D RAGE to date are:

ASIC ID	ASIC description
08h	NEC VT-A3
00h	NEC VT-A4
40h	SGS VT-A4
40h	NEC GT-A2 (3D RAGE)

See Also

mach64 Programmer’s Guide:

- *Getting Started: mach64 Detection: Card Detection*

	CONFIG_STAT0																								**Sparse I/O: 1C I/O: 71C8, 71CC, 72EC**						**MM: 0_39 BLK: 39**	
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	g								f	e	d	c	b	a		

a	R/W	CFG_MEM_TYPE	Memory type: 000 = Disable memory access 001 = DRAM 010 = EDO DRAM 011 = Pseudo EDO (EDO data latching with DRAM page behaviour) 100 = SDRAM 101 = Reserved 110 = Reserved 111 = Reserved Default: Strap setting
b	R/W	CFG_DUAL_CAS_EN	Dual CAS support enable: 0 = dual CAS support disabled 1 = dual CAS support enabled Default: Strap setting*
c	R/W	CFG_VGA_EN	Strap to enable/disable VGA mode Default: Strap setting
d	R/W	CFG_CLOCK_EN	0 = GUI clock controlled by GUI activity 1 = GUI clock always on Default: 0
e	R/W	VMC_SENSE	0 = No VMC connection 1 = VMC connection detected Default: 0
f	R/W	VFC_SENSE	0 = No VFC connection 1 = VFC connection detected Default: 0
g	R/W	BOARD_ID	Straps for board ID Default: Strap setting

*Support for dualwrite has been removed

Description

This register returns the configuration of the current board. For the VT and 3D RAGE, CONFIG_STAT0 is read-write.

Usage

This register is used by the adapter BIOS for query functions and determining appropriate action for other function calls. It is also used for determining the initialization parameters and boot-times.

See Also

- *Advanced Topics: Manual Mode Switching and Custom CRT Modes: Manual Mode Switching*

	HW_DEBUG																				**Sparse I/O:–**								**MM: 1_50**			
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT/GT	▨																a															

a	R/W	HW_DEBUG	Debug register

Description

This register is used for debugging hardware on chip samples only.

Accelerator CRTC and DAC Registers

Accelerator CRTC

	CRTC_H_TOTAL_DISP	Sparse I/O: 0, 1F I/O: 01C8, 01CC, 02EC	MM: 0_00 BLK: 00

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

a	R/W	CRTC_H_TOTAL	Horizontal total (pixels*8)
b	R/W	CRTC_H_DISP	Horizontal display end (pixels*8)

Description

CRTC_H_TOTAL_DISP is used to specify horizontal total and horizontal displayed parameters for the accelerator CRTC. All horizontal parameters are specified in characters (pixels-times-8).

Usage

This register is used only for mode switching, and should be touched only by the adapter BIOS.

See Also

***mach64* Programmer's Guide:**

- *Advanced Topics: Manual Mode Switching and Custom CRT Modes: Manual Mode Switching*
- *Advanced Topics: Manual Mode Switching and Custom CRT Modes: Designing a Custom CRT Mode*
- *Appendix C, CRTC Parameters*

	CRTC_H_SYNC_STRT_WID																									**Sparse I/O: 1** **I/O: 05C8, 05CC, 06EC**					**MM: 0_01** **BLK: 01**	
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT/GT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	e	d					▨	▨	▨	c	▨	b			a							

a	R/W	CRTC_H_SYNC_STRT	Horizontal sync start (pixels*8)
b	R/W	CRTC_H_SYNC_DLY	Horizontal sync start delay in pixels
c	R/W	CRTC_H_SYNC_STRT_HI	High bit for Horizontal sync start
d	R/W	CRTC_H_SYNC_WID	Horizontal sync width (pixels*8)
e	R/W	CRTC_H_SYNC_POL	Horizontal sync polarity (1 -> active low)

Description

CRTC_H_SYNC_STRT_WID specifies the horizontal sync attributes for the accelerator CRTC. All horizontal parameters are specified in characters (pixels-times-8).

Usage

This register is used only for mode switching and should be touched only by the adapter BIOS.

See Also

***mach64* Programmer's Guide:**

- *Advanced Topics: Manual Mode Switching and Custom CRT Modes: Manual Mode Switching*
- *Advanced Topics: Manual Mode Switching and Custom CRT Modes: Designing a Custom CRT Mode*
- *Appendix C, CRTC Parameters*

	CRTC_V_TOTAL_DISP																					Sparse I/O: 2 I/O: 09C8, 09CC, 0AEC								MM: 0_02 BLK: 02		
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	▨	▨	▨	▨	▨	b											▨	▨	▨	▨	▨	a										

a	R/W	CRTC_V_TOTAL	Vertical total
b	R/W	CRTC_V_DISP	Vertical display end

Description

CRTC_V_TOTAL_DISP is used to specify vertical total and vertical displayed parameters for the accelerator CRTC. All vertical parameters are specified in lines.

Usage

This register is used only for mode switching, and should be touched only by the adapter BIOS.

See Also

***mach64* Programmer's Guide:**

- *Advanced Topics: Manual Mode Switching and Custom CRT Modes: Manual Mode Switching*
- *Advanced Topics: Manual Mode Switching and Custom CRT Modes: Designing a Custom CRT Mode*
- *Appendix C, CRTC Parameters*

	CRTC_V_SYNC_STRT_WID	**Sparse I/O: 3** **I/O: 0DC8, 0DCC, 0EEC**	**MM: 0_03** **BLK: 03**

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

a	R/W	CRTC_V_SYNC_STRT	Vertical sync start
b	R/W	CRTC_V_SYNC_WID	Vertical sync width
c	R/W	CRTC_V_SYNC_POL	Vertical sync polarity (1 -> active low)

Description

CRTC_V_SYNC_STRT_WID specifies the vertical sync attributes for the accelerator CRTC. All vertical parameters are specified in lines.

Usage

This register is used only for mode switching, and should be touched only by the adapter BIOS.

See Also

***mach64* Programmer's Guide:**

- *Advanced Topics: Manual Mode Switching and Custom CRT Modes: Manual Mode Switching*
- *Advanced Topics: Manual Mode Switching and Custom CRT Modes: Designing a Custom CRT Mode*
- *Appendix C, CRTC Parameters*

CRTC_VLINE_CRNT_VLINE	**Sparse I/O: 4** **I/O: 11C8, 11CC, 12EC**	**MM: 0_04** **BLK: 04**

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

a	R/W	CRTC_VLINE	Vertical line at which vertical line interrupt is triggered.
b	R	CRTC_CRNT_VLINE	Current vertical line.

Description

The CRTC_VLINE field determines the line at which a CRTC interrupt will be triggered if the interrupts are enabled. The CRTC_CRNT_VLINE field is read-only. It returns the current value of the accelerator CRTC vertical line counter.

Usage

This register is used only in applications that require synchronization to the CRTC, such as smooth animation.

See Also

CRTC_INT_CNTL on *page 4-26*

mach64 **Programmer's Guide:**

- *Advanced Topics: Interrupts*
- *Advanced Topics: CRT Synchronization and Animation*

	CRTC_OFF_PITCH	Sparse I/O: 5 I/O: 15C8, 15CC, 16EC	MM: 0_05 BLK: 05

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

a	R/W	CRTC_OFFSET	Display address offset in terms of 64 bit words.
b	R/W	CRTC_PITCH	Display pitch in pixels*8

Description

CRTC_OFF_PITCH is used to specify the starting memory offset and pitch of the accelerator CRTC. The pitch value must correspond exactly to the destination draw engine pitch for visible screen memory. Remember that if the memory boundary is enabled, the offset must be set to a value above or equal to the boundary offset.

Usage

The offset register may be used for scrolling and panning on a large desktop if the pitch is set to a value larger than the display resolution. This register may also be used for double buffering applications.

See Also

MEM_CNTL on *page 4-10*

SRC_OFF_PITCH on *page 4-68*

DST_OFF_PITCH on *page 4-53*

***mach64* Programmer's Guide:**

- *Linear Aperture: VGA Interaction*
- *Advanced Topics: Scrolling and Panning*
- *Advanced Topics: CRT Synchronization and Animation: Double Buffering (Memory)*

CRTC_INT_CNTL

Sparse I/O: 6 MM: 0_06
I/O: 19C8, 19CC, 1AEC BLK: 06

BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	▨	▨	▨	▨	▨	▨	▨	▨	o	n	m	l	k	j	i	h	▨	▨	▨	▨	▨	▨	▨	▨	▨	g	f	e	d	c	b	a

a	R	CRTC_VBLANK	Vertical blank
b	R/W	CRTC_VBLANK_INT_EN3	Vertical blank interrupt enable * (Default = 0, active high)
c	R	CRTC_VBLANK_INT	Vertical blank interrupt* (active high)
	W	CRTC_VBLANK_INT_AK	Vertical blank acknowledge* (1 -> clears interrupt)
d	R/W	CRTC_VLINE_INT_EN	Vertical line interrupt enable * (Default = 0, active high)
e	R	CRTC_VLINE_INT	Vertical line interrupt* (active high)
	W	CRTC_VLINE_INT_AK	Vertical line interrupt acknowledge* (1-> clears interrupt)
f	R	CRTC_VLINE_SYNC	Vertical line sync: 0 = even scan line 1 = odd scan line
g	R	CRTC_FRAME	Interface odd/even frame: 0 = even frame 1 = odd frame
h	R/W	VIDEOIN_EVEN_INT_EN	Video in end-of-even-field interrupt enable
i	R	VIDEOIN_EVEN_INT	Video in end-of-even-field interrupt* (active high)
	W	VIDEOIN_EVEN_INT_AK	Video in end-of-even-field acknowledge* (1 -> clears interrupt)
j	R/W	VIDEOIN_ODD_INT_EN	Video in end-of-odd-field interrupt enable
k	R	VIDEOIN_ODD_INT	Video in end-of-odd-field interrupt* (active high)
	W	VIDEOIN_ODD_INT_AK	Video in end-of-odd-field acknowledge* (1 -> clears interrupt)
l	R/W	OVERLAY_EOF_INT_EN	Overlay end-of-frame interrupt enable
m	R	OVERLAY_EOF_INT	Overlay end-of-frame interrupt* (active high)
	W	OVERLAY_EOF_INT_AK	Overlay end-of-frame acknowledge* (1 -> clears interrupt)
n	R/W	VMC_EC_INT_EN	VMC exception code interrupt enable
o	R	VMC_EC_INT	VMC exception code interrupt* (active high)
	W	VMC_EC_INT_AK	VMC exception code interrupt acknowledge* (1-> clears interrupt)

Description

CRTC_INT_CNTL is used for enabling and acknowledging interrupts generated by the accelerator CRTC, video capture, overlay display, and VMC port, and reading the status of the CRTC.

Usage

Applications may use this register to achieve smooth animation, or reduce flickering and tearing.

See Also

CRTC_VLINE_CRNT_VLINE on *page 4-24*
***mach64* Programmer's Guide:**

- *Advanced Topics: Interrupts*
- *Advanced Topics: CRT Synchronization and Animation*

CRTC_GEN_CNTL

Sparse I/O: 7 **MM: 0_07**
I/O: 1DC8, 1DCC, 1EEC **BLK: 07**

BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	w	v	u	t	s	r	q	p	o	n	m	l	k				j		▨	▨	i	h			g	f	▨	e	d	c	b	a

a	R/W	CRTC_DBL_SCAN_EN	Double scan enable
b	R/W	CRTC_INTERLACE_EN	Interlace enable.
c	R/W	CRTC_HSYNC_DIS	Disables horizontal sync output
d	R/W	CRTC_VSYNC_DIS	Disables vertical sync output
e	R/W	CRTC_CSYNC_EN	Enables composite sync on horizontal sync output.
f	R/W	CRTC_DISPLAY_DIS	Disables the display, forcing the blanking signal to be active.
g	R/W	CRTC_VGA_XOVERSCAN	0 = disables overscan in VGA mode 1 = enables overscan in VGA mode
h	R/W	CRTC_PIX_WIDTH	Display pixel width: 0 = (reserved) 1 = 4 bpp 2 = 8 bpp 3 = 15 bpp (5,5,5) 4 = 16 bpp (5,6,5) 5 = 24 bpp 6 = 32 bpp 7 = (reserved)
i	R/W	CRTC_BYTE_PIX_ORDER	Enables reversing the pixel order within each memory byte in 4 bpp mode. 0 = pixel order from MSNibble to LSNibble. 1 = pixel order from LSNibble to MSNibble.
j	R/W	CRTC_FIFO_OVERFILL	Number of double quadwords to overfill FIFO (0-3)
k	R/W	CRTC_FIFO_LWM	Display FIFO low water mark. Note that the display FIFO is 16 entries deep.
l	R/W	VGA_128KAP_PAGING	Enable extended aperture paging in 128K VGA aperture mode: 0 = disable (normal) 1 = enable paging through 128K aperture
m	R/W	CRTC_DISPREQ_ONLY	0 = Normal 1 = Only display requests are serviced
n	R/W	CRTC_LOCK_REGS	Lock extended CRTC registers from being written to: 0 = unlocked 1 = locked (read only)
o	R/W	CRTC_SYNC_TRISTATE	0 = Normal 1 = Tri-state Hsync & Vsync
p	R/W	CRTC_EXT_DISP_EN	Extended display mode enable: (Default = 0) 0 = VGA display 1 = Extended mode display
q	R/W	CRTC_ENABLE	Enables CRT controller: (Default = 0) 0 = resets CRTC 1 = enables CRTC

(continued on next page)

CRTC_GEN_CNTL

Sparse I/O: 7
I/O: 1DC8, 1DCC, 1EEC
MM: 0_07
BLK: 07

BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	w	v	u	t	s	r	q	p	o	n	m	l	k				j		▨	▨	i	h			g	f	▨	e	d	c	b	a

r	R/W	CRTC_DISP_REQ_ENB	0 = enable display requests 1 = disable display requests (Default = 1)
s	R/W	VGA_ATI_LINEAR	Enable linear addressing through VGA aperture 0 = disable linear addressing 1 = enable linear addressing
t	R/W	CRTC_VSYNC_FALL_EDGE	Select VSYNC edge to start frame sequence 0 = rising edge of VSYNC 1 = falling edge of VSYNC
u	R/W	VGA_TEXT_132	Extended text mode select (linear address 132 column text mode) 1 = Active 0 = Inactive
v	R/W	VGA_XCRT_CNT_EN	Extended CRTC display address counter enable. Active High
w	R/W	VGA_CUR_B_TEST	Test cursor blinking. Active High.

Description

All miscellaneous initialization bits for the accelerator CRTC are contained in CRTC_GEN_CNTL.

CRTC_HSYNC_DIS and CRTC_VSYNC_DIS are used specifically for the Display Power Management System (DPMS).

CRTC_PIX_WIDTH and CRTC_BYTE_PIX_ORDER are used to specify pixel arrangement in memory. These bits correspond exactly to their respective fields in DP_PIX_WIDTH.

CRTC_FIFO_LWM is used only in DRAM configurations. It specifies the emptiness of the display FIFO that must be reached before the CRTC should get more data from memory. There is a lower limit before the display becomes corrupted. The upper limit is 15 because the size of the display FIFO is 16 entries deep. The higher the number, the greater the number of memory page faults. This leads to a decrease in available memory bandwidth, which in turn leads to a slower draw engine.

Usage

This register is used only for mode switching and should be touched only by the adapter BIOS.

See Also

***mach64* Programmer's Guide:**

- *Advanced Topics: Manual Mode Switching and Custom CRT Modes: Manual Mode Switching*
- *Advanced Topics: Manual Mode Switching and Custom CRT Modes: Designing a Custom CRT Mode*
- *Appendix C, CRTC Parameters*

EXT_CRTC_GEN_CNTL — I/O:– — MM: 0_17 (W), BLK: 17 (W)

BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT-A4N	▨	▨	▨	▨	▨	▨	▨	i	▨	▨	▨	▨	▨	▨	h	g			f			e	d	c		b				a		

a	W	CRTC_DISP_REQ_MAX	Maximum burst length of a display request to memory
b	W	CRTC_DISP_REQ_WS	Display wait-states (in memory clocks) between consecutive requests to mem. Note, a value of ‘zero’ will disable both ‘CRTC_DISP_REQ’ options.
c	W	CRTC_FIFO_EXTSENSE	External select signal for custom FIFO
d	W	CRTC_FORCE_BS2	Force block size 2 for all requests
e	W	CRTC_DISP_REQ_MODE	0 = Always relinquish memory controller after burst for WS period (Default) 1 = **Only** relinquish memory controller after burst if another request pending
f	W	CRTC_DISP_RST_WS	Number of wait states inserted into the start of display FIFO reset
g	W	CRTC_FIFO_LWM_BIT5	Bit 5 of CRTC_FIFO_LWM@CRTC_GEN_CNTL
h	W	CRTC_FIFO_OVERFILL_BIT2	Bit 2 of CRTC_FIFO_OVERFILL@CRTC_GEN_CNTL
i	W	HALF_SIZE_DFIFO	1 = Display FIFO is half size

EXT_CRTC_GEN_CNTL — I/O:– — MM: 1_17 (R)

BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT-A4N	▨	▨	▨	h	▨	▨	▨	g	▨	▨	▨			f			e	▨	▨	▨	▨	▨	d	c		b				a		

a	R	CRTC_DISP_REQ_MAX	Maximum burst length of a display request to memory
b	R	CRTC_DISP_REQ_WS	Display wait-states (in memory clocks) between consecutive requests to mem. Note, a value of ‘zero’ will disable both ‘CRTC_DISP_REQ’ options.
c	R	CRTC_FIFO_EXTSENSE	External select signal for custom FIFO
d	R	CRTC_FORCE_BS2	Force block size 2 for all requests
e	R	CRTC_DISP_REQ_MODE	0 = Always relinquish memory controller after burst for WS period (Default) 1 = **Only** relinquish memory controller after burst if another request pending
f	R	CRTC_DISP_RST_WS	Number of wait states inserted into the start of display FIFO reset
g	R	CRTC_FIFO_LWM_BIT5	Bit 5 of CRTC_FIFO_LWM@CRTC_GEN_CNTL
h	R	CRTC_FIFO_OVERFILL_BIT2	Bit 2 of CRTC_FIFO_OVERFILL@CRTC_GEN_CNTL

	EXT_CRTC_GEN_CNTL																								I/O:–				MM: 0_17, BLK: 17			
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT-A4S	▨	▨	▨	▨	▨	▨	▨	l	▨	▨	k	j	i	▨	h	g	f					e	d	c	b				a			

a	R/W	CRTC_DISP_REQ_MAX	Maximum burst length of a display request to memory
b	R/W	CRTC_DISP_REQ_WS	Display wait-states (in memory clocks) between consecutive requests to mem. Note, a value of ‘zero’ will disable both ‘CRTC_DISP_REQ’ options.
c	R/W	CRTC_FIFO_EXTSENSE	External select signal for custom FIFO
d	R/W	CRTC_FORCE_BS2	Force block size 2 for all requests
e	R/W	CRTC_DISP_REQ_MODE	0 = Always relinquish memory controller after burst for WS period (Default) 1 = **Only** relinquish memory controller after burst if another request pending
f	R/W	CRTC_DISP_RST_WS	Number of wait states inserted into the start of display FIFO reset
g	R/W	CRTC_FIFO_LWM_BIT5	Bit 5 of CRTC_FIFO_LWM@CRTC_GEN_CNTL
h	R/W	CRTC_FIFO_OVERFILL_BIT2	Bit 2 of CRTC_FIFO_OVERFILL@CRTC_GEN_CNTL
i	R/W	CRTC_DISP_REQ_MAX_BIT5	Bit 5 of CRTC_DISP_REQ_MAX
j	R/W	CRTC_DISP_REQ_WS_BIT5	Bit 5 of CRTC_DISP_REQ_WS
k	R/W	CRTC_DISP_REQ_CNT_EN	1 = In extended mode and BS = 4, disable display request when required number of request grants has been obtained (default = 0)
l	R/W	HALF_SIZE_DFIFO	1 = Display FIFO is half size

Description

This register extends the set of miscellaneous initialization bits for the accelerator CRTC in the VT-A4.

Usage

For **VT-A4N**, in order to write this register, the offset is mapped to MM: 0_17 or BLK: 17. For reading back this register, the offset is mapped to MM: 1_17 (no block I/O access). In other words, the write is in block ‘0’ and the read is in block ‘1’.

For **VT-A4S**, the register has been moved such that to read or write this register, the offset is mapped to MM: 0_17 or BLK: 17.

See Also

CRTC_GEN_CNTL on *page 4-27*

Overscan

Display overscan is enabled if any of the overscan width values are non-zero. The left and right overscan widths are described in terms of pixels*8 and the top and bottom overscan widths are described in terms of vertical lines.

The overscan color is defined by an 8-bit index and a 24-bit color. In all display modes, the 24-bit color will be used by the internal RAMDAC and displayed on the monitor attached to the VT. Note this is always a true color that is not mapped through the palette. The 8-bit index color is used in 4 bpp and 8 bpp modes for data going out on the 8-bit feature connector. The receiving board is expected to index all 4 bpp and 8 bpp data through its own palette.

	OVR_CLR																				**Sparse I/O: 8** **I/O: 21C8, 21CC, 22EC**								**MM: 0_10** **BLK: 10**			
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT/GT	d								c								b								a							

a	R/W	OVR_CLR_8	Overscan color INDEX, output on feature connector in 4 and 8 bpp modes
b	R/W	OVR_CLR_B	Blue overscan color, to internal DAC
c	R/W	OVR_CLR_G	Green overscan color, to internal DAC
d	R/W	OVR_CLR_R	Red overscan color, to internal DAC

Description

This register specifies the overscan color.

Usage

This register should be touched only by the adapter BIOS when mode switching or by the adapter installation program for overscan configuration.

See Also

CUR_CLR0 on *page 4-33*

***mach64* Programmer's Guide:**

- *Advanced Topics: Manual Mode Switching and Custom CRT Modes: Designing a Custom CRT Mode*

OVR_WID_LEFT_RIGHT — **Sparse I/O: 9** — **I/O: 25C8, 25CC, 26EC** — **MM: 0_11** — **BLK: 11**

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

a	R/W	OVR_WID_LEFT	Left overscan width (in 8*pixels)
b	R/W	OVR_WID_RIGHT	Right overscan width (in 8*pixels)

Description

OVR_WID_LEFT_RIGHT specifies the left and right overscan widths in characters (i.e., pixels-by-8).

Usage

This register should be touched only by the adapter BIOS for mode switching or by the adapter installation program for overscan configuration. The left overscan width must not exceed the horizontal back porch timing; the right overscan width must not exceed the horizontal front porch timing.

See Also

***mach64* Programmer's Guide:**

- *Advanced Topics: Manual Mode Switching and Custom CRT Modes: Designing a Custom CRT Mode*

OVR_WID_TOP_BOTTOM — **Sparse I/O: A** — **I/O:29C8, 29CC, 2AEC** — **MM: 0_12** — **BLK: 12**

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

a	R/W	OVR_WID_TOP	Top overscan width (in scan lines)
b	R/W	OVR_WID_BOTTOM	Bottom overscan width (in scan lines)

Description

OVR_WID_TOP_BOTTOM specifies the top and bottom overscan widths in lines.

Usage

This register should be touched only by the adapter BIOS for mode switching or by the adapter installation program for overscan configuration. The top overscan width must not exceed the vertical back porch timing; the bottom overscan width must not exceed the vertical front porch timing.

See Also

***mach64* Programmer's Guide:**

- *Advanced Topics: Manual Mode Switching and Custom CRT Modes: Designing a Custom CRT Mode*

Hardware Cursor

The hardware cursor may be any size up to 64x64 pixels. The cursor pitch is always 64 pixels meaning the cursor definition is always 64 pixels wide although pixels outside of the visible cursor area are ignored. The cursor definition is in reverse pixel order within each byte. Once the cursor is defined, it is moved around on the screen simply by updating the cursor position.

The cursor is stored as a linear block of off-screen video memory, starting at address CUR_OFFSET. The upper left corner of the cursor is specified by CUR_HORZ_POSN and CUR_VERT_POSN. The cursor size may be decreased from 64x64 by setting CUR_HORZ_OFF and CUR_VERT_OFF to non-zero.

	CUR_CLR0	**Sparse I/O: B I/O: 2DC8, 2DCC, 2EEC**	**MM: 0_18 BLK: 18**
BITS	**31–24**	**23–16**	**15–8**	**7–0**

BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT/GT	d								c								b								a							

a	R/W	CUR_CLR0_8	Cursor color 0 INDEX, output on feature connector in 4 bpp and 8 bpp modes
b	R/W	CUR_CLR0_B	Blue cursor color 0, to internal DAC
c	R/W	CUR_CLR0_G	Green cursor color 0, to internal DAC
d	R/W	CUR_CLR0_R	Red cursor color 0, to internal DAC

Description

CUR_CLR0 contains color 0 for the hardware cursor. For non-integrated controllers (GX, CX), the color is specified in the lower 8 bits for pseudocolor modes or the upper 24 bits in direct color modes.

For controllers with integrated DACs (CT, ET), cursor color 0 going to the internal DAC is 24 bits (CUR_CLR0_R, CUR_CLR0_G, CUR_CLR0_B) in all display modes. In 4 bpp and 8 bpp modes when the VESA feature connector is on, then CUR_CLR0_8 is the pseudo color value for cursor color 0 on the pixel data lines.

Usage

This register is used when defining the hardware cursor attributes.

See Also

CUR_CLR1 on *page 4-34*

***mach64* Programmer's Guide:**

- *Engine Operations: Miscellaneous Operations: Hardware Cursor*

	CUR_CLR1				**Sparse I/O: C** **I/O: 31C8, 31CC, 32EC**	**MM: 0_19** **BLK: 19**
BITS	**31 30 29 28 27 26 25 24**		**23 22 21 20 19 18 17 16**	**15 14 13 12 11 10 9 8**		**7 6 5 4 3 2 1 0**
VT/GT	d		c	b		a

a	R/W	CUR_CLR1_8	Cursor color 1 INDEX, output on feature connector in 4 bpp and 8 bpp modes
b	R/W	CUR_CLR1_B	Blue cursor color 1, to internal DAC
c	R/W	CUR_CLR1_G	Green cursor color 1, to internal DAC
d	R/W	CUR_CLR1_R	Red cursor color 1, to internal DAC

Description

CUR_CLR1 contains color 1 for the hardware cursor. For non-integrated controllers (GX, CX), the color is specified in the lower 8 bits for pseudocolor modes or the upper 24 bits in direct color modes.

For controllers with integrated DACs (CT, ET), cursor color 0 going to the internal DAC is 24 bits (CUR_CLR1_R, CUR_CLR1_G, CUR_CLR1_B) in all display modes. In 4 bpp and 8 bpp modes when the VESA feature connector is on, then CUR_CLR1_8 is the pseudocolor value for cursor color 1 on the pixel data lines.

Usage

This register is used when defining the hardware cursor attributes.

See Also

CUR_CLR0 on *page 4-33*

***mach64* Programmer’s Guide:**

- *Engine Operations: Miscellaneous Operations: Hardware Cursor*

	CUR_OFFSET	Sparse I/O: D I/O: 35C8, 35CC, 36EC	MM: 0_1A BLK: 1A

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

a	R/W	CUR_OFFSET	Cursor address offset in terms of 64 bit words

Description

CUR_OFFSET points to the top left corner of the 64x64 cursor definition block.

Usage

This register is used to define the hardware cursor.

See Also

***mach64* Programmer's Guide:**

- *Engine Operations: Miscellaneous Operations: Hardware Cursor*

CUR_HORZ_VERT_POSN	Sparse I/O: E I/O:39C8, 39CC, 3AEC	MM: 0_1B BLK: 1B

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

a	R/W	CUR_HORZ_POSN	Cursor horizontal position
b	R/W	CUR_VERT_POSN	Cursor vertical position

Description

CUR_HORZ_VERT_POSN specifies the top left corner of the hardware cursor in the display area, referenced to the top left corner of the cursor definition area.

Usage

This register is used to move the hardware cursor on the screen.

See Also

***mach64* Programmer's Guide:**

- *Engine Operations: Miscellaneous Operations: Hardware Cursor*

	CUR_HORZ_VERT_OFF	**Sparse I/O: F** **I/O: 3DC8, 3DCC, 3EEC**	**MM: 0_1C** **BLK: 1C**

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

a	R/W	CUR_HORZ_OFF	Cursor horizontal offset
b	R/W	CUR_VERT_OFF	Cursor vertical offset

Description

CUR_HORZ_VERT_OFF specifies the offsets from the 64x64 cursor definition block where the cursor definition area is to begin. Each offset should be set such that offset = 64 - size.

Usage

This register is used when defining the hardware cursor attributes.

See Also

***mach64* Programmer's Guide:**

- *Engine Operations: Miscellaneous Operations: Hardware Cursor*

Clock Control

	CLOCK_CNTL																				Sparse I/O: 12 I/O: 49C8, 49CC, 4AEC									MM: 0_24 BLK: 24		
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	▨	▨	▨	▨	▨	▨	▨	▨	d								▨	▨	c				b	▨	▨	▨	▨	▨	▨	▨	a	

a	R/W	CLOCK_SEL	Non-VGA mode video clock frequency select. In VGA mode clock select is determined by GENMO(3:2).
b	R/W	PLL_WR_EN	Internal clock synthesizer (PLL) register write enable. 0 = PLL_DATA is read-only 1 = PLL_DATA is read/write
c	R/W	PLL_ADDR	Selects register in internal clock synthesizer (PLL) to read or write.
d	R/W	PLL_DATA	Internal clock synthesizer (PLL) read/write data. See PLL_WR_EN.

Description

CLOCK_CNTL is used to select a pixel clock in non-VGA modes. It is also used for programming the internal clock synthesizer (PLL). The internal clock synthesizer in the VT and 3D RAGE has only 4 programmable pixel clock settings. Therefore, only bits 0 and 1 of CLOCK_SEL are used.

Usage

This register should be touched only by the adapter BIOS when switching video modes.

See Also

***mach64* Programmer's Guide:**

- *Advanced Topics: Manual Mode Switching and Custom CRT Modes*
- *Appendix C: CRTC Parameters*
- *Appendix D: Clock Chip Reference*

Note that the PLL register fields are detailed in *Appendix B, Programming PLL Registers*.

DAC Control

The DAC_REGS are also addressed at VGA I/O addresses 3C6h to 3C9h (not in the order below). The same palette data and DAC_MASK are used whether in VGA or extended modes.

	DAC_REGS	Sparse I/O: 17 I/O:5DC8, 5DCC,5EEC	MM: 0_30 BLK: 30

BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	d								c								b						▨	▨	a							

a	R/W	DAC_W_INDEX	DAC write index register * Indexes the 256x24 entry palette RAM for write operations.
b	R/W	DAC_DATA	DAC data register * If DAC is in 6-bit mode then two MSB are ignored on write, zero on read. If DAC is in 8-bit mode, then two LSB are ignored on write, zero on read. DAC WRITE: First 8 bit write is red data, next two writes are green and blue, respectively. After blue write the 24 bit palette is updated and DAC_W_INDEX auto-increments to next index. DAC READ: After DAC_R_INDEX is written, three reads from DAC_DATA will give red, green and blue color components, respectively. After every third read the next index in the palette is read automatically and the read index is auto-incremented.
c	R/W	DAC_MASK	DAC mask register * This 8 bit mask value is ANDed with the incoming 8 bit pseudocolor pixel data. The resultant value is used for looking up the true color in the LUT.
d	W	DAC_R_INDEX	DAC read index register * Indexes the 256x24 entry palette RAM for read operations.

Description

DAC_REGS is actually a group of four 8-bit registers (not a single 32-bit register) aliased to the VGA DAC registers DAC_MASK (3C6), DAC_R_INDEX (3C7), DAC_W_INDEX (3C8) and DAC_DATA (3C9). See the *mach64 VGA Register Guide* for more details.

These registers must be accessed in 8-bit chunks. These registers may also be accessed in accelerator mode through the VGA I/O addresses if DAC_VGA_ADR_EN@DAC_CNTL is set.

Usage

These registers are used by applications to reprogram the DAC look up table (LUT).

See Also

DAC_CNTL on *page 4-40*

Chapter 9, VGA-Compatible Registers

***mach64* Programmer's Guide:**

- *Advanced Topics: CRT Synchronization and Animation: Double Buffering (Palette)*

DAC_CNTL

Sparse I/O: 18 MM: 0_31
I/O: 61C8, 61CC, 62EC BLK: 31

BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT/GT	o	▨	n	m	l	k	j	i	▨	▨	▨	▨	▨		h		g	f	e	▨	▨	▨	▨	d	c	▨	▨	▨	b	a	▨	▨

a	R/W	DAC_BLANKING	0 = 0 IRE blanking pedestal 1 = Enable 7.5 IRE blanking pedestal
b	R/W	DAC_CMP_DISABLE	Enable/Disable DAC comparators 0 = enable DAC comparators (Default) 1 = disable (powerdown) DAC comparators
c	R	DAC_CMP_OUTPUT	DAC comparator output 0 = At least 1 comparator > 0.42V 1 = All 3 comparators < 0.28V
d	R/W	DAC_8BIT_EN	Enables 8 bit DAC operation 1 = 8 bit operation 0 = 6 bit operation
e	R/W	DAC_VGA_ADR_EN	Enables addressing the DAC at the VGA IO DAC address when CRTC_EXT_DISP_EN is a '1'.
f	R/W	DAC_FEA_CON_EN	Enables feature connector signal outputs. Turns on output of 8 bit pixel data, clock and blank signals to feature connector. Feature connector should be disabled in 24 bpp and 32 bpp modes and high resolution modes when pixel clock rate is too high.
g	R/W	DAC_PDWN	Power down internal DAC (DAC macro only). Feature connector outputs can still run normally.
h	R	DAC_TYPE	DAC Type (Always 1 for VT) 0 = internal DAC, 18-bit palette, no gamma correction 1 = internal DAC, 24-bit palette, gamma correction 2-7 = (reserved)
i	R/W	DAC_GIO_STATE_1	GIO1 pin external state (R) GIO1 pin output value when DAC_GIO_DIR_1 = 1 (W)
j	R/W	DAC_GIO_STATE_0	GIO0 pin external state (R) GIO0 pin output value when DAC_GIO_DIR_0 = 1 (W)
k	R/W	DAC_GIO_STATE_4	GIO4 pin external state (R) GIO4 pin output value when DAC_GIO_DIR_4 = 1 (W)
l	R/W	DAC_GIO_DIR_1	0 = GIO1 pin input only (tri-state output) (Default) 1 = GIO1 pin output enabled. Output value = DAC_GIO_STATE_1
m	R/W	DAC_GIO_DIR_0	0 = GIO0 pin input only (tri-state output) (Default) 1 = GIO0 pin output enabled. Output value = DAC_GIO_STATE_0
n	R/W	DAC_GIO_DIR_4	0 = GIO4 pin input only (tri-state output) (Default) 1 = GIO4 pin output enabled. Output value = DAC_GIO_STATE_4
o	R/W	DAC_RW_WS	0 = Disable DAC Read/Write wait states 1 = Enable DAC Read/Write wait states

Description

DAC_CNTL configures the on-chip DAC interface unit. If the DAC has extended address bits to access extended DAC registers, then those upper address bits will be specified in DAC_EXT_SEL. DAC_8BIT_EN selects between 8-bit or 6-bit modes, and is used only if both modes are supported.

The DAC_TYPE can be overwritten for non-integrated controllers (GX, CX) to override the initial DAC type. Please consult the manufacturer's DAC specification.

Usage

This register is used only for mode switching and should be touched only by the adapter BIOS.

See Also

***mach64* Programmer's Guide:**

- *Advanced Topics: DAC Programming*

	CRC_SIG	**Sparse I/O: 1D** **I/O: 75C8, 75CC, 75EC**	**MM: 0_3A** **BLK: 3A**

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

a	R	CRC_SIG	CRC signature value

Description

On a VT and 3D RAGE, this register is used to accumulate the display CRC check.

Usage

CRC_SIG is used for diagnostics of the CRTC, DAC, hardware cursor, and overscan.

See Also

GEN_TEST_CNTL on *page 4-14*

Draw Engine Trajectory Registers

Draw Engine registers are only visible in the memory space, not in sparse or block I/O.

Destination (and Z) Trajectory

In the **3D RAGE**, a number of new 3D operations have been added. To support hardware Z buffering, a Z source FIFO has been added. New registers associated with this FIFO in the **3D RAGE** are: Z_OFF_PITCH and Z_CNTL.

A new trajectory, the trapezoid, has been added to the **3D RAGE**. In this mode, the standard line engine is used to walk the leading edge of the trapezoid, and a new line engine is used to walk the trailing edge. The pixels operated on are those lying on the scan lines between the two edges. This new trajectory may be used for the destination, the texture map sources, the 2D source, and the Z source. The new registers added to the **3D RAGE** are: TRAIL_BRES_ERR, TRAIL_BRES_INC, and TRAIL_BRES_DEC. DST_BRES_LENGTH is expanded in the **3D RAGE** to include the span length of the trapezoid and also to kick off trapezoidal operations. The DST_CNTL and SRC_CNTL registers are also affected in the **3D RAGE**.

DST_BRES_DEC (LEAD_BRES_DEC) **I/O:–** **MM: 0_4B**

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

a	R/W	DST_BRES_DEC	Bresenham decrement for line and (in the 3D RAGE) trapezoid leading edge

Description

DST_BRES_DEC is a signed 18 bit register that stores the Bresenham line decrement term. The number loaded into this register must be negative. This term is added to the DST_BRES_ERR term whenever the Bresenham error is positive.

For the **VT**, the value written to this register should be calculated:

DST_BRES_DEC = 2 *[min(|dx|,|dy|) - max(|dx|,|dy|)]

Usage

In the **VT**, this register is used only for line draw operations. In the **3D RAGE**, this register is used for line draw or trapezoid draw operations, and is aliased as LEAD_BRES_DEC.

See Also

DST_BRES_ERR on *page 4-44*

DST_BRES_INC on *page 4-45*

DST_BRES_LNTH on *page 4-46*

***mach64* Programmer's Guide:**

- *Engine Operations: Background Information: Trajectories: Destination Trajectory 2, Line*
- *Engine Operations: Draw Operations: Colour Source: Drawing Lines*

	DST_BRES_ERR (LEAD_BRES_ERR)	I/O:–	MM: 0_49

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

a	R/W	DST_BRES_ERR	Bresenham error term for line and (in the 3D RAGE) trapezoid leading edge

Description

DST_BRES_ERR is a signed, 18-bit register that stores the Bresenham line error term. If the error term is negative, an axial step is taken and DST_BRES_INC is added to this register; otherwise, a diagonal step is taken in the direction of the major axis, and DST_BRES_DEC is added.

For the **VT**, the initial value of the register field DST_BRES_ERR should be:

DST_BRES_ERR = 2 * min(|dx|,|dy|) - max(|dx|,|dy|)

Usage

In the **VT**, this register is used only for line draw operations.   In the **3D RAGE**, this register is used for line draw or trapezoid draw operations, and is aliased as LEAD_BRES_ERR.

See Also

DST_BRES_DEC on *page 4-43*

DST_BRES_INC on *page 4-45*

DST_BRES_LNTH on *page 4-46*

mach64 **Programmer's Guide:**

- *Engine Operations: Background Information: Trajectories: Destination Trajectory 2, Line*
- *Engine Operations: Draw Operations: Colour Source: Drawing Lines*

	DST_BRES_INC (LEAD_BRES_INC)																								I/O:–					MM: 0_4A		
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	a																	

a	R/W	DST_BRES_INC	Bresenham increment for line and (in the 3D RAGE) trapezoid leading edge

Description

DST_BRES_INC is a signed 18 bit register which stores the Bresenham line increment term. The number loaded into this register must be positive. This term is added to the DST_BRES_ERR term whenever the Bresenham error is negative.

In the **VT**, the value written to the field DST_BRES_INC should be:

DST_BRES_INC = 2 *min(|dx|,|dy|)

Usage

In the **VT**, this register is used only for line draw operations. In the **3D RAGE**, this register is used for line draw or trapezoid draw operations, and is aliased as LEAD_BRES_INC.

See Also

DST_BRES_DEC on *page 4-43*

DST_BRES_ERR on *page 4-44*

DST_BRES_LNTH on *page 4-46*

***mach64* Programmer's Guide:**

- *Engine Operations: Background Information: Trajectories: Destination Trajectory 2, Line*
- *Engine Operations: Draw Operations: Colour Source: Drawing Lines*

DST_BRES_LNTH (LEAD_BRES_LNTH) I/O:– MM: 0_48*

BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT	b	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	a														
GT	g	f		e													d	c														

a	R/W	DST_BRES_LNTH	Bresenham line length
b	W	DST_BRES_LNTH_LINE_DIS	Disables initiation of bresenham line draw operations: 0 = Bresenham line draw operation initiated. 1 = No bresenham line draw operation initiated. **NOTE:** This function is performed when the register is written. The bit is not stored, or read.
c	R/W	DST_BRES_LNTH	Bresenham line length and trapezoid leading edge length. This field is aliased with DST_WIDTH[14:0].
d	R/W	DRAW_TRAP	To initiate a trapezoid, set to ‘1’. This field is aliased with DST_WIDTH[15].
e	R/W	TRAIL_X	Location of trapezoid trailing edge. Note: This field is not written if bit 15 is a ‘1’ and bit 31 is a ‘0’. This field is aliased with DST_HEIGHT[12:0].
f	–	(Reserved)	Aliased to DST_HEIGHT[14:13]
g	W	DST_BRES_LNTH_LINE_DIS	Disables initiation of bresenham line draw operations: Bit 31 Bit 15 0 0 Bresenham line draw operation initiated. TRAIL_X and DST_BRES_LNTH are loaded. 0 1 Trapezoid draw operation. TRAIL_X is not updated, but DST_BRES_LNTH is loaded. 1 0 TRAIL_X and DST_BRES_LNTH are loaded. No line or trapezoid operations are done. 1 1 Trapezoid draw operation. TRAIL_X and DST_BRES_LNTH are loaded.

Description

Writing the value of line length to register DST_BRES_LNTH will initiate a line draw. The number written to this register is the number of pixels that will be drawn when DST_LAST_PEL@DST_CNTL is set.

Writing to this register also overwrites the contents of DST_WIDTH.

DST_BRES_LNTH = max(|dx|,|dy|) + 1

Usage

In the **VT**, this register is used for line draw operations.

*In the **3D RAGE**, DST_BRES_LNTH is used for line draw or trapezoid draw operations, and is aliased by LEAD_BRES_LNTH at **MM:0_51**.

See Also

DST_BRES_DEC on *page 4-43*

DST_BRES_ERR on *page 4-44*

DST_BRES_INC on *page 4-45*

***mach64* Programmer's Guide:**

- *Engine Operations: Background Information: Trajectories: Destination Trajectory 2, Line*
- *Engine Operations: Draw Operations: Colour Source: Drawing Lines*

DST_CNTL I/O:– MM: 0_4C

BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	l	j	i			h	g	f	e	d	c	b	a
GT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	p	▨	o	n	m	l	k	i			h	g	f	e	d	c	b	a

a	R/W	DST_X_DIR	Destination X direction 0 = right to left 1 = left to right
b	R/W	DST_Y_DIR	Destination Y direction 0 = bottom to top 1 = top to bottom
c	R/W	DST_Y_MAJOR	Destination Y major axis flag for bresenham lines 0 = X major line 1 = Y major line
d	R/W	DST_X_TILE	Enables rectangular tiling in the X direction
e	R/W	DST_Y_TILE	Enables rectangular tiling in the Y direction
f	R/W	DST_LAST_PEL	Destination last pel enable
g	R/W	DST_POLYGON_EN	Destination polygon outline and polygon fill enable
h	R/W	DST_24_ROT_EN	Enables 24 bpp rotation. DSTPIXWIDTH MUST be set to 8 BPP
i	R/W	DST_24_ROT	Initial foreground color, background color, write mask, and monochrome pattern rotation when drawing packed 24 bpp. The initial DST_24_ROT value is defined as follows: If DST_X_DIR = ‘0’ then DST_24_ROT = (Trunc(((DST_X * 3) + 2)/4)) Mod 6 Else DST_24_ROT = (Trunc((DST_X * 3)/4)) Mod 6 End If
j	R/W	DST_BRES_ZERO	Sign of DST_BRES_ERR when DST_BRES_ERR = 0 0 = DEST_BRES_ERR = 0 is positive number 1 = DEST_BRES_ERR = 0 is negative number
k	R/W	DST_BRES_SIGN	Bresenham sign: 0 = zero error term is positive number 1 = zero error term is negative number
l	R/W	DST_POLYGON_RTEDGE_DIS	Disables drawing of the right edge pixel of a polygon fill operation. 0 = drawing of right edge pixel is enabled 1 = drawing of right edge pixel is disabled
m	R/W	TRAIL_X_DIR	Trapezoid trailing edge direction: 0 = right to left 1 = left to right
n	R/W	TRAP_FILL_DIR	Trapezoid fill direction: 0 = right to left (trailing edge is to the left of the leading edge) 1 = left to right (trailing edge is to the right of the leading edge)
o	R/W	TRAIL_BRES_SIGN	Bresenham sign for trailing edge of trapezoids: 0 = zero error term is positive number 1 = zero error term is negative number

(continued on next page)

DST_CNTL **I/O:–** **MM: 0_4C**

BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	l	j	i			h	g	f	e	d	c	b	a
GT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	p	▨	o	n	m	l	k	i			h	g	f	e	d	c	b	a

p	R/W	BRES_SIGN_AUTO	Bresenham sign: 0 = zero error term is defined by DST_BRES_SIGN and TRAIL_BRES_SIGN bit 1 = Override DST_BRES_SIGN and TRAIL_BRES_SIGN bit. Zero error term is positive for X Major lines whose Y_DIR is ‘0’ or for Y Major lines whose X_DIR is ‘0’.

Description

Miscellaneous control bits for the destination area:

If the destination trajectory is rectangular, DST_X_DIR and DST_Y_DIR will determine the trajectory quadrant that the destination area and the source area will take. Rectangular areas are always X-major.

If the destination trajectory is a line, DST_X_DIR, DST_Y_DIR, and DST_Y_MAJOR will determine the trajectory octant that the destination line will take and the source area direction is specified in SRC_LINE_X_DIR@SRC_CNTL. Source areas are always rectangular. Source areas do not advance in the Y direction when destination trajectory is a line.

DST_X_TILE and DST_Y_TILE affect only rectangular destinations. These bits determine the side effect of the DST_X and DST_Y registers after the draw operation is completed. If DST_X_TILE is set, then DST_X will be assigned DST_X+DST_WIDTH upon draw completion for a left-to-right draw operation (DST_X_DST_WIDTH for right-to-left); otherwise DST_X is unchanged.

Similarly, if DST_Y_TILE is set, then DST_Y will be assigned DST_Y+DST_HEIGHT upon draw completion (for a top-to-bottom draw operation (DST_Y_DST_HEIGHT for bottom-to-top); otherwise DST_Y is unchanged.

DST_LAST_PEL affects only destination line trajectories. When set, the last pixel in the line is drawn, otherwise it is not. This register does *not* affect DST_X and DST_Y trajectories.

DST_POLYGON_EN affects line and rectangle destinations differently. (1) For lines, with this bit set, only one pixel will be drawn per scan line (with the exception of horizontal lines, where no pixels will be drawn). Lines exceeding the left scissor boundary will be saturated to the left scissor. (2) For rectangles, with this bit set, an implicit polygon source (specified by the source trajectory registers) is used to conduct an alternate-fill polygon fill on the destination. Blit sources cannot be used in conjunction with polygon fills. DST_X_DIR must be set to left-to-right operation for correct polygon fill behavior.

DST_24_ROT_EN and DST_24_ROT are used to set the initial rotation factor in packed 24 bpp mode.

DST_BRES_SIGN controls the behavior of the line draw engine when DST_BRES_ERR is zero. When set, a zero error term is considered negative, otherwise it is positive.

Usage

This register must be set for all draw operations. DST_Y_MAJOR and DST_LAST_PEL are applicable only for line draw operations. DST_X_TILE and DST_Y_TILE are applicable only for rectangle fills.

See Also

GUI_TRAJ_CNTL on *page 4-104*

SRC_CNTL on *page 4-62*

***mach64* Programmer’s Guide:**

- *Engine Operations: Background Information: Trajectories: Destination Trajectory 1, Rectangular*
- *Engine Operations: Background Information: Trajectories: Destination Trajectory 2, Line*
- *Engine Operations: Draw Operations: Colour Source: Drawing Lines*
- *Engine Operations: Background Information: Trajectories: Trajectory Modifier 2, DST_POLYGON_EN*
- *Engine Operations: Background Information: Side Effects of Trajectories*
- *Advanced Topics: Polygons*
- *Engine Operations: Miscellaneous Operations: Drawing in Packed 24 Bit Per Pixel Mode*

	DST_HEIGHT																							I/O:–					MM: 0_45			
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	a														

a	R/W	DST_HEIGHT	Destination height (For 3D RAGE, Bits12:0 are aliased with TRAIL_BRES_X)

Description

This register specifies the height in pixels of a rectangular destination area.

Usage

In the **VT**, this register is used only when drawing a rectangular destination area. In the **3D RAGE**, this register is used when drawing a rectangular or trapezoidal destination area.

See Also

DST_WIDTH on *page 4-54*

DST_HEIGHT_WIDTH on *page 4-52*

***mach64* Programmer's Guide:**

- *Engine Operations: Background Information: Trajectories: Destination Trajectory 1, Rectangular*
- *Engine Operations: Draw Operations: Colour Source: Drawing Rectangles*
- *Engine Operations: Draw Operations: Standard Bitblit Source*
- *Engine Operations: Draw Operations: Specialized Bitblit Source: Transparent BitBlts*
- *Advanced Topics: Polygons*

	DST_HEIGHT_WIDTH																							I/O:–					MM: 0_46			
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT	▨	▨	▨	b													▨	a														
GT	b																▨	a														

a	W	DST_HEIGHT	Destination height
b	W	DST_WIDTH	Destination width

Description

DST_HEIGHT_WIDTH is a composite of registers DST_HEIGHT and DST_WIDTH. Writing to this register will initiate a rectangle fill operation.

Usage

These registers are used only for drawing rectangular destinations.

See Also

DST_HEIGHT on *page 4-51*

DST_WIDTH on *page 4-54*

	DST_OFF_PITCH																				I/O:–						MM: 0_40				
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	b										▨	▨	a																			

a	R/W	DST_OFFSET	Destination offset address in terms of 64 bit words.
b	R/W	DST_PITCH	Destination pitch in pixels*8. Note that for monochrome modes the destination pitch must be a multiple of 64 pixels. Also in 4 bpp modes the destination pitch must be a multiple of 16 pixels.

Description

DST_OFF_PITCH is used to specify the offset (in QWORDs) and pitch (in pixels) of the destination area. If the memory boundary is enabled, ensure that the offset points to an area above or equal to the boundary. If the destination is on-screen memory, any value of pitch smaller than the display area is not meaningful.

Usage

This register should be set for all draw operations.

See Also

SRC_OFF_PITCH on *page 4-68*

mach64 **Programmer's Guide:**

- *Engine Operations: Background Information: Trajectories: Destination Trajectory 1, Rectangular*
- *Engine Operations: Background Information: Trajectories: Destination Trajectory 2, Line*
- *Advanced Topics: CRT Synchronization and Animation: Double Buffering (Memory)*
- *Linear Aperture: VGA Interaction*

	DST_WIDTH																									I/O:–				MM: 0_44		
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT	c	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	a	a	a	a	a	a	a	a	a	a	a	a	a
GT	c	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	b	b	b	b	b	b	b	b	b	b	b	b	b	b	b	b

a	R/W	DST_WIDTH	Destination width (VT)
b	R/W	DST_WIDTH	Destination width (3D RAGE). Only bits [12:0] are used for rectangle draws. Bit 15 is write ONLY and will always read back as ‘0’. Bits [15:13] are aliased to DST_BRES_LENGTH[15:13] and are used for trapezoid draw operations.
c	W	DST_WIDTH_FILL_DIS	Disables initiation of rectangular fill operations: 0 = Rectangular fill operation initiated. 1 = No rectangular fill operation initiated. **NOTE:** This function is performed when the register is written. The bit is not stored, or read.

Description

DST_WIDTH specifies the width in pixels of a rectangular destination area and initiates a draw operation. DST_WIDTH can be set without initiating a draw operation by setting the DST_WID_FILL_DIS bit.

Writing to this register also overwrites the contents of DST_BRES_LNTH.

Usage

This register is used only when drawing a rectangular destination area.

See Also

DST_HEIGHT on *page 4-51*

DST_HEIGHT_WIDTH on *page 4-52*

DST_X_WIDTH on *page 4-56*

***mach64* Programmer’s Guide:**

- *Engine Operations: Background Information: Trajectories: Destination Trajectory 1, Rectangular*
- *Engine Operations: Draw Operations: Colour Source: Drawing Rectangles*
- *Engine Operations: Draw Operations: Standard Bitblit Source*
- *Engine Operations: Draw Operations: Specialized Bitblit Source: Transparent BitBlts*
- *Advanced Topics: Polygons*

	DST_X																									I/O:–				MM: 0_41		
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	a												

a	R/W	DST_X	Destination X coordinate

Description

DST_X specifies the starting X coordinate of the destination trajectory. This is a signed 13 bit number.

Usage

This register is used for all draw operations.

See Also

DST_X_WIDTH on *page 4-56*

DST_Y on *page 4-57*

DST_Y_X on *page 4-58*

mach64 **Programmer's Guide:**

- *Engine Operations: Background Information: Trajectories: Destination Trajectory 1, Rectangular*
- *Engine Operations: Background Information: Trajectories: Destination Trajectory 2, Line*

DST_X_WIDTH **I/O:–** **MM: 0_47**

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

a	W	DST_X	Destination X coordinate
b	W	DST_WIDTH	Destination width

Description

DST_X_WIDTH is a composite of registers DST_X and DST_WIDTH.

Usage

This register can alternatively be used to initiate rectangle fill operations when drawing a rectangular destination area.

See Also

DST_X on *page 4-55*

DST_WIDTH on *page 4-54*

	DST_Y																										I/O:–				MM: 0_42	
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	a														

a	R/W	DST_Y	Destination Y coordinate

Description

DST_Y specifies the starting Y coordinate of the destination trajectory. This is a signed 15 bit number.

Usage

This register is used for all draw operations.

See Also

DST_X on *page 4-55*

DST_Y_X on *page 4-58*

mach64 **Programmer's Guide:**

- *Engine Operations: Background Information: Trajectories: Destination Trajectory 1, Rectangular*
- *Engine Operations: Background Information: Trajectories: Destination Trajectory 2, Line*

	DST_Y_X																									I/O:–				MM: 0_43*		
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	▨	▨	▨	b													▨	a														

a	W	DST_Y	Destination Y coordinate
b	W	DST_X	Destination X coordinate

Description

DST_Y_X is a composite of registers DST_X and DST_Y.

*In the **3D RAGE**, DST_Y_X is also aliased to **MM:0_4D**.

Usage

These registers are used for all draw operations.

See Also

DST_X on *page 4-55*

DST_Y on *page 4-57*

	TRAIL_BRES_ERR																						I/O:–						MM: 0_4E			
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	a																	

a	R/W	TRAIL_BRES_ERR	Bresenham error term for line and trapezoid trailing edge

Description

In the **3D RAGE**, TRAIL_BRES_ERR is a signed, 18-bit register that stores the Bresenham error term for line and trapezoid trailing edges.

	TRAIL_BRES_INC																						I/O:–						MM: 0_4F			
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	a																	

a	R/W	TRAIL_BRES_INC	Bresenham increment for line and trapezoid trailing edge

Description

In the **3D RAGE**, TRAIL_BRES_INC is a signed 18 bit register which stores the Bresenham increment for line and trapezoid trailing edges. The number loaded into this register must be positive. This term is added to the DST_BRES_ERR term whenever the Bresenham error is negative.

	TRAIL_BRES_DEC																						I/O:–						MM: 0_50			
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	a																	

a	R/W	TRAIL_BRES_DEC	Bresenham decrement for line and trapezoid trailing edge

Description

In the **3D RAGE**, TRAIL_BRES_DEC is a signed 18 bit register which stores the Bresenham decrement for line and trapezoid trailing edges. The number loaded into this register must be negative. This term is added to the DST_BRES_ERR term whenever the Bresenham error is positive.

	Z_OFF_PITCH																								I/O:–				MM: 0_52			
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GT	b										▨	▨	a																			

a	R/W	Z_OFFSET	Z offset address in terms of 64 bit words.
b	R/W	Z_PITCH	Z pitch in pixels*8.

Description

Z_OFF_PITCH is used in the **3D RAGE** to specify the offset (in QWORDs) and pitch (in pixels) of the Z-buffer area. The Z-buffer destination will always track the normal destination in X and Y, but with its own pitch and offset.

Usage

This register should be set for all 3D draw operations.

See Also

Z_CNTL on *page 4-61*

	Z_CNTL																							**I/O:–**						**MM: 0_53**		
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
GT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	d	▨		c		▨	▨	b	a

a	R/W	Z_EN	Enables use of Z functions: 0 = Z testing is disabled 1 = Z testing is enabled
b	R/W	Z_SRC	Enables use of Z functions: 0 = Z testing is disabled 1 = Z testing is enabled
c	R/W	Z_TEST	Specific Z test to be enabled: Z test code — Test 000 — Z test never passes 001 — Z < current Z 010 — Z <= current Z 011 — Z == current Z 100 — Z >= current Z 101 — Z > current Z 110 — Z != current Z 111 — Z test always passes A passing Z test will overwrite the existing value with the new source value.
d	R/W	Z_MASK	Enables writing to the Z planes: 0 = writing to the Z planes is disabled 1 = writing to the Z planes is enabled

Description

In **3D RAGE**, Z_CNTL controls the new Z source FIFO which supports the hardware Z buffering.

Usage

Z_EN turns Z on/off. Z_TEST specifies which Z test is to be done. Z_MASK allows Z to apply to colors, even if Z itself is never written.

See Also

Z_OFF_PITCH on *page 4-60*

Source Trajectory

SRC_CNTL **I/O:–** **MM: 0_6D**

BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT																												e	d	c	b	a
GT																		g	g						f			e	d	c	b	a

a	R/W	SRC_PATT_EN	Enables pattern source. SRC_Y_END is only used if this bit is enabled.
b	R/W	SRC_PATT_ROT_EN	Enables pattern source rotation. SRC_X_START, SRC_Y_START is only used if this bit is enabled.
c	R/W	SRC_LINEAR_EN	Enables the source to be advanced linearly in memory. The source starts at SRC_OFFSET and advances in the left-to-right direction. **Note:** DST_X_DIR should also be set to the left-to-right to operate properly. Note that all other source registers and control bits with the exception of SRC_BYTE_ALIGN are ignored.
d	R/W	SRC_BYTE_ALIGN	Allows the source to skip to the next data byte boundary when the destination advances in the Y direction. **Note:** SRC_LINEAR_EN MUST be set. This is only for 1 bpp and 4 bpp source pixel width definitions.
e	R/W	SRC_LINE_X_DIR	Source X direction when drawing operation is a bresenham line.
f	R/W	SRC_TRACK_DST	Source will track the trajectory which the Dst FIFO is using. In this case, the X, Y values used for the source and destination will be identical. The SRC_X and SRC_Y values will be ignored.
g	R/W	SRC_BLOCK_FILL_FCN	Denotes that a block fill will be done. This is an optimized way of filling large regions in memory. When performing this operation, the destination must be aligned on a 64 byte boundary. Only multiples of 64 bytes are written. No scissoring is performed. In this mode, the source comes from the FGND CLR register. Note that an SGRAM color register write cycle must be performed prior to the actual block fill. When doing a block fill,the DST_WIDTH register should be programmed with the number of 64 byte regions in the width. Bit 14 Bit 13 0 0 Normal Mode 0 1 SGRAM color register write using FGND_CLR (a NO-OP for other memory types) 1 0 Block Write using FGND_CLR 1 1 Reserved If the memory type used is not SGRAM, the block fill is performed correctly, but no performance increase results.

Description

SRC_CNTL contains various enable bits for blit source trajectory control.

SRC_PATT_EN, SRC_PATT_ROT_EN, and SRC_LINEAR_EN are set as shown in the table below to select the source trajectories as follows:

SRC_LINEAR_EN	SRC_PATT_ROT_EN	SRC_PATT_EN	Source Trajectory
1	0	0	Strictly Linear
0	0	0	Unbounded Y
0	0	1	General Pattern
0	1	1	General Pattern with Rotation

SRC_BYTE_ALIGN is applicable only when the destination is rectangular. In 1 bpp and 4 bpp modes, if this field is set, the source pointer will advance to the nearest byte boundary when the destination advances in the Y direction.

SRC_LINE_X_DIR is applicable only when the destination is a line. It is used to specify the source direction.

Source and destination trajectory directions are de-coupled for line draws. The source is always rectangular, but never advances in the Y direction for lines.

Usage

Use this register only if a blit source is selected in the pixel data path.

See Also

DST_CNTL on *page 4-48*

GUI_TRAJ_CNTL on *page 4-104*

***mach64* Programmer's Guide:**

- *Engine Operations: Background Information: Trajectories*
- *Engine Operations: Background Information: Source and Destination Alignment*
- *Engine Operations: Draw Operations: Standard Bitblit Source*

	SRC_HEIGHT1																							**I/O:–**					**MM: 0_65**			
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	a														

a	R/W	SRC_HEIGHT1	Source height 1

Description

This register is used to specify the height of the source area for general-pattern sources or the vertical distance (in lines) from DST_Y to the bottom of a pattern block for general-pattern-with-rotation sources.

Usage

Set this register only if a general-pattern blit source or general-pattern-with-rotation blit source is selected in the pixel data path.

See Also

SRC_HEIGHT1_WIDTH1 on *page 4-65*

SRC_WIDTH1 on *page 4-69*

***mach64* Programmer's Guide:**

- *Engine Operations: Background Information: Trajectories: Source Trajectory 3, General Pattern*
- *Engine Operations: Background Information: Trajectories: Source Trajectory 4, General Pattern with Rotation*
- *Engine Operations: Draw Operations: Standard Bitblit Source: General Pattern*
- *Engine Operations: Draw Operations: Standard Bitblit Source: General Pattern with Rotation*

	SRC_HEIGHT1_WIDTH1																									I/O:–				MM: 0_66		
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT/GT	▨	▨	▨	b													▨	a														

a	W	SRC_HEIGHT1	Source height 1
b	W	SRC_WIDTH1	Source width 1

Description

This register is a composite of SRC_HEIGHT1 and SRC_WIDTH1.

Usage

Set this register only if a general-pattern blit source or general-pattern-with-rotation blit source is selected in the pixel data path.

See Also

SRC_HEIGHT1 on *page 4-64*

SRC_WIDTH1 on *page 4-69*

	SRC_HEIGHT2																								I/O:–				MM: 0_6B			
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨								a							

a	R/W	SRC_HEIGHT2	Source height 2

Description

This register is used to specify the height of the general pattern for general-pattern-with-rotation sources.

Usage

Set this register only if a general-pattern-with-rotation blit source is selected.

See Also

SRC_HEIGHT2_WIDTH2 on *page 4-67*

SRC_WIDTH2 on *page 4-70*

***mach64* Programmer's Guide:**

- *Engine Operations: Background Information: Trajectories: Source Trajectory 4, General Pattern with Rotation*
- *Engine Operations: Draw Operations: Standard Bitblit Source: General Pattern with Rotation*

	SRC_HEIGHT2_WIDTH2																								I/O:–				MM: 0_6C			
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT/GT	▨	▨	▨	b													▨	a														

a	W	SRC_HEIGHT2	Source height 2
b	W	SRC_WIDTH2	Source width 2

Description

This register is a composite of SRC_HEIGHT2 and SRC_WIDTH2.

Usage

Set these registers only if a general-pattern-with-rotation blit source is selected.

See Also

SRC_HEIGHT2 on *page 4-66*

SRC_WIDTH2 on *page 4-70*

SRC_OFF_PITCH **I/O:–** **MM: 0_60**

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

a	R/W	SRC_OFFSET	Source offset address in terms of 64 bit words.
b	R/W	SRC_PITCH	Source pitch in pixels x 8. **Note:** In monochrome mode the source pitch must be a multiple of 64 pixels. Also, in 4 bpp mode the source pitch must be a multiple of 16 pixels.

Description

This register is used to specify the offset (in QWORDs) and pitch (in pixels) of the blit source area.

Usage

This register should be set for any draw operations that select a blit source in the pixel data path.

See Also

DST_OFF_PITCH on *page 4-53*

mach64 **Programmer's Guide:**

- *Engine Operations: Background Information: Trajectories: Source Trajectory 1, Strictly Linear*
- *Engine Operations: Background Information: Trajectories: Source Trajectory 2, Unbounded Y*
- *Engine Operations: Background Information: Trajectories: Source Trajectory 3, General Pattern*
- *Engine Operations: Background Information: Source Trajectory 4, General Pattern with Rotation*

	SRC_WIDTH1																									I/O:–				MM: 0_64		
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT/GT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	a												

a	R/W	SRC_WIDTH1	Source width 1

Description

This register is used to specify the width of the source area for general pattern sources or the horizontal distance (in pixels) from DST_X to the right edge of a pattern block for general pattern sources with rotation.

Usage

Set this register only if a general-pattern blit source, a general-pattern-with-rotation blit source, or an unbounded Y source is selected in the pixel data path.

See Also

SRC_HEIGHT1 on *page 4-64*

SRC_HEIGHT1_WIDTH1 on *page 4-65*

***mach64* Programmer's Guide:**

- *Engine Operations: Background Information: Trajectories: Source Trajectory 2, Unbounded Y*
- *Engine Operations: Background Information: Trajectories: Source Trajectory 3, General Pattern*
- *Engine Operations: Background Information: Trajectories: Source Trajectory 4, General Pattern with Rotation*
- *Engine Operations: Draw Operations: Standard Bitblit Source: General Pattern*
- *Engine Operations: Draw Operations: Standard Bitblit Source: General Pattern with Rotation*

SRC_WIDTH2 **I/O:–** **MM: 0_6A**

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

a	R/W	SRC_WIDTH2	Source width 2

Description

This register is used to specify the width of the pattern for general-pattern-with-rotation sources.

Usage

Set this register only if a general-pattern-with-rotation blit source is selected.

See Also

SRC_HEIGHT2 on *page 4-66*

SRC_HEIGHT2_WIDTH2 on *page 4-67*

mach64 **Programmer's Guide:**

- *Engine Operations: Background Information: Trajectories: Source Trajectory 4, General Pattern with Rotation*
- *Engine Operations: Draw Operations: Standard Bitblit Source: General Pattern with Rotation*

	SRC_X																								I/O:–				MM: 0_61			
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT/GT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	a												

a	R/W	SRC_X	Source X coordinate

Description

This register specifies the starting X coordinate of the blit source trajectory. This is a signed 13 bit number.

Usage

This register is used for any draw operation which selects a blit source in the pixel data path.

See Also

SRC_Y on *page 4-73*

SRC_Y_X on *page 4-75*

***mach64* Programmer's Guide:**

- *Engine Operations: Background Information: Trajectories: Source Trajectory 1, Strictly Linear*
- *Engine Operations: Background Information: Trajectories: Source Trajectory 2, Unbounded Y*
- *Engine Operations: Background Information: Trajectories: Source Trajectory 4, General Pattern*
- *Engine Operations: Background Information: Trajectories: Source Trajectory 4, General Pattern with Rotation*

SRC_X_START **I/O:–** **MM: 0_67**

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

a	R/W	SRC_X_START	Pattern source X start for pattern rotation in the X direction

Description

This register specifies the starting horizontal edge of a general-pattern-with-rotation blit source. This is a signed 13 bit number.

Usage

Set this register only if a draw operation selects a general-pattern-with-rotation in the pixel data path.

See Also

SRC_Y_START on *page 4-74*

SRC_Y_X_START on *page 4-76*

mach64 **Programmer's Guide:**

- *Engine Operations: Background Information: Trajectories: Source Trajectory 4, General Pattern with Rotation*
- *Engine Operations: Draw Operations: Standard Bitblit Source: General Pattern with Rotation*

	SRC_Y																								**I/O:–**				**MM: 0_62**			
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	a														

a	R/W	SRC_Y	Source Y coordinate

Description

This register specifies the starting Y coordinate of the blit source trajectory. This is a signed 15 bit number.

Usage

This register is used for any draw operation that selects a blit source in the pixel data path.

See Also

SRC_X on *page 4-71*

SRC_Y_X on *page 4-75*

***mach64* Programmer's Guide:**

- *Engine Operations: Background Information: Trajectories: Source Trajectory 1, Strictly Linear*
- *Engine Operations: Background Information: Trajectories: Source Trajectory 2, Unbounded Y*
- *Engine Operations: Background Information: Trajectories: Source Trajectory 4, General Pattern*
- *Engine Operations: Background Information: Trajectories: Source Trajectory 4, General Pattern with Rotation*

	SRC_Y_START																								I/O:–				MM: 0_68			
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT/GT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	a														

a	R/W	SRC_Y_START	Pattern source Y start for pattern rotation in the Y direction

Description

This register specifies the starting vertical edge of a general-pattern-with-rotation blit source. This is a signed 15 bit number.

Usage

Set this register only if a draw operation selects a general-pattern-with-rotation in the pixel data path.

See Also

SRC_X_START on *page 4-72*

SRC_Y_X_START on *page 4-76*

mach64 **Programmer's Guide:**

- *Engine Operations: Background Information: Trajectories: Source Trajectory 4, General Pattern with Rotation*
- *Engine Operations: Draw Operations: Standard Bitblit Source: General Pattern with Rotation*

	SRC_Y_X																								I/O:–				MM: 0_63			
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	▨	▨	▨	b													▨	a														

a	W	SRC_Y	Source Y coordinate
b	W	SRC_X	Source X coordinate

Description

This register is a composite of SRC_Y and SRC_X.

Usage

Set these registers only if a blit source is selected in the pixel data path.

See Also

SRC_Y on *page 4-73*

SRC_X on *page 4-71*

	SRC_Y_X_START																								I/O:–				MM: 0_69			
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT/GT	▨	▨	▨	b													▨	a														

a	W	SRC_Y_START	Pattern source Y start for pattern rotation in the Y direction
b	W	SRC_X_START	Pattern source X start for pattern rotation in the X direction

Description

This register is a composite of SRC_X_START and SRC_Y_START.

Usage

Set these registers only if a general pattern with rotation blit source is selected in the pixel data path.

See Also

SRC_X_START on *page 4-72*

SRC_Y_START on *page 4-74*

Draw Engine Control Registers

Host Data

	HOST_DATA[15:0]																									I/O:–					MM:0_80-0_8F	
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	a																															

a	W	HOST_DATA[i]	Host data register – pixel data taken from the least significant bit, nibble, byte, or word for left-to-right rectangular drawing operations; and taken from the most significant bit, nibble, byte, or word for right-to-left rectangular drawing operations. Data for line drawing operations are always taken from the least significant bit, nibble, byte, or word. (See DP_BYTE_PIX_ORDER@DP_PIX_WIDTH for more details on monochrome and 4 bpp modes.)

Description

HOST_DATA is actually a single register mapped to 16 consecutive addresses, thus HOST_DATA[15:0]. This scheme enables applications to conduct high speed host transfers using REP MOVSD. The register corresponds directly to the host data source in the pixel data path.

If a draw operation expects host data and any other draw engine register is written, the draw operation will *panic* and complete the draw operation with a garbage color. This condition is interruptible through BUS_CNTL.

If HOST_DATA is written and host data is not expected, the data is discarded.

Full FIFO discipline must be applied to this register; that is, check the FIFO before doing a REP MOVSD.

Usage

Data is fed to the draw engine through a host source by repeatedly writing pixel data to this register. Under certain conditions, it may be more desirable to write directly to the big linear aperture instead of using the host data port.

In the 3D RAGE, when using HOST_DATA for 3D operations (either shading or texture mapping), the data is not allowed to be packed. That is, only a single pixel at a time is sent to the host data register. The pixel will be assumed to be aligned to bit 0. The DP_SCALE_PIX_WIDTH rather than the DP_HOST_PIX_WIDTH field will determine the size of the data.

See Also

BUS_CNTL on *page 4-8*

HOST_CNTL on *page 4-78*

***mach64* Programmer's Guide:**

- *Engine Operations: Background Information: Logical Pixel Data Path: Host Data Consumption*
- *Advanced Topics: Performance Issues*

	HOST_CNTL																											**I/O:–**				**MM: 0_90**
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	b	a

a	R/W	HOST_BYTE_ALIGN	Enables byte aligning the host data.
b	R/W	HOST_BIG_ENDIAN_EN	Enables big endian data translation for 15 bpp, 16 bpp, and 32 bpp pixel width. In 15 bpp and 16 bpp modes the bytes within each word are swapped. In 32 bpp mode the order of the four bytes within each dword is reversed. 0 = big endian data translation disabled 1 = big endian data translation enabled

Description

HOST_BYTE_ALIGN controls the host data consumption for 1 bpp and 4 bpp data. When host data byte align is enabled and the destination trajectory advances in the Y direction, pixels are consumed from the host data port until the nearest byte boundary is reached. When host data byte align is not enabled, pixel data is packed.

Usage

HOST_BIT_ENDIAN_EN controls the endians of the HOST_DATA register.This register is used only if a data path source is set to host data, and host data pixel width is 1 bpp or 4 bpp.

See Also

GUI_TRAJ_CNTL on *page 4-104*

HOST_DATA on *page 4-77*

***mach64* Programmer's Guide:**

- *Engine Operations: Background Information: Logical Pixel Data Path: Host Data Consumption*
- *Engine Operations: Draw Operations: Colour Source: Drawing Rectangles*

Pattern

	PAT_REG0																									I/O:–				MM: 0_A0		
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	a																															

a	R/W	PAT_REG0	Pattern register 0

Description

PAT_REG0 defines one half of a fixed pattern. PAT_REG1 defines the other half.

Usage

Set this register only when a fixed monochrome or fixed color pattern is selected as a data path source.

See Also

PAT_CNTL on *page 4-81*

PAT_REG1 on *page 4-80*

***mach64* Programmer's Guide:**

- *Engine Operations: Background Information: Logical Pixel Data Path*
- *Engine Operations: Background Information: Logical Pixel Data Path: Pattern Consumption*
- *Engine Operations: Draw Operations: Pattern Source: Fixed Patterns*

	PAT_REG1																								I/O:–				MM: 0_A1			
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	a																															

a	R/W	PAT_REG1	Pattern register 1

Description

PAT_REG1 defines one half of a fixed pattern. PAT_REG0 defines the other half.

Usage

Set this register only when a fixed monochrome or fixed color pattern is selected as a data path source.

See Also

PAT_CNTL on *page 4-81*

PAT_REG0 on *page 4-79*

mach64 **Programmer's Guide:**

- *Engine Operations: Background Information: Logical Pixel Data Path*
- *Engine Operations: Background Information: Logical Pixel Data Path: Pattern Consumption*
- *Engine Operations: Draw Operations: Pattern Source: Fixed Patterns*

	PAT_CNTL																														**I/O:–**	**MM: 0_A2**
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	c	b	a

a	R/W	PAT_MONO_EN	Monochrome 8x8 pattern enable
b	R/W	PAT_CLR_4x2_EN	Color 4x2 pattern enable
c	R/W	PAT_CLR_8x1_EN	Color 8x1 pattern enable

Description

PAT_CNTL is used for fixed pattern control. All enable bits are mutually exclusive – do not set more than one for any draw operation.

Usage

This register need only be used when the monochrome source is set for fixed mono patterns or when either of the two color sources is set for fixed color patterns. When a fixed pattern is selected, one and only one pattern type can be selected (i.e., set one, and only one bit in this register).

Only 8 bpp color pattern source is supported. Use generalized source pattern for 16 bpp and 32 bpp color patterns.

See Also

GUI_TRAJ_CNTL on *page 4-104*

PAT_REG0 on *page 4-79*

PAT_REG1 on *page 4-80*

***mach64* Programmer's Guide:**

- *Engine Operations: Background Information: Logical Pixel Data Path*
- *Engine Operations: Background Information: Logical Pixel Data Path: Pattern Consumption*
- *Engine Operations: Draw Operations: Pattern Source: Fixed Patterns*

Scissors

	SC_LEFT																							I/O:–					MM: 0_A8			
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	a												

a	R/W	SC_LEFT	Left scissor

Description

SC_LEFT defines the left edge of a scissor rectangle. Drawing is inhibited for any pixel that is outside this scissor rectangle. Scissors are inclusive. This is a signed, 13-bit number.

Usage

This register must be set for all draw operations.

See Also

SC_TOP on *page 4-85*

SC_BOTTOM on *page 4-86*

SC_RIGHT on *page 4-83*

SC_LEFT_RIGHT on *page 4-84*

mach64 **Programmer's Guide:**

- *Engine Operations: Miscellaneous Operations: Scissoring and Masking*

	SC_RIGHT																								I/O:–					MM: 0_A9		
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	a	a	a	a	a	a	a	a	a	a	a	a	a

a	R/W	SC_RIGHT	Right scissor

Description

SC_RIGHT defines the right edge of a scissor rectangle. Drawing is inhibited for any pixel which is outside of this scissor rectangle. Scissors are inclusive. This is a signed 13-bit number.

Usage

This register must be set for all draw operations.

See Also

SC_TOP on *page 4-85*

SC_LEFT on *page 4-82*

SC_LEFT_RIGHT on *page 4-84*

SC_BOTTOM on *page 4-86*

***mach64* Programmer's Guide:**

- *Engine Operations: Miscellaneous Operations: Scissoring and Masking*

	SC_LEFT_RIGHT																								I/O:–				MM: 0_AA			
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	▨	▨	▨	▨													▨	▨	▨	▨						a						

a	W	SC_LEFT	Left scissor
b	W	SC_RIGHT	Right scissor

Description

SC_LEFT_RIGHT is a composite of registers SC_LEFT and SC_RIGHT.

Usage

This register must be set for all draw operations.

See Also

SC_LEFT on *page 4-82*

SC_RIGHT on *page 4-83*

	SC_TOP																								I/O:–				MM: 0_AB			
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	a														

a	R/W	SC_TOP	Top scissor

Description

SC_TOP defines the top edge of a scissor rectangle. Drawing is inhibited for any pixel which is outside of this scissor rectangle. Scissors are inclusive. This is a signed 15-bit number.

Usage

This register must be set for all draw operations.

See Also

SC_BOTTOM on *page 4-86*

SC_LEFT on *page 4-82*

SC_RIGHT on *page 4-83*

SC_TOP_BOTTOM on *page 4-87*

***mach64* Programmer's Guide:**

- *Engine Operations: Miscellaneous Operations: Scissoring and Masking*

	SC_BOTTOM																								I/O:–					MM: 0_AC		
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	a														

a	R/W	SC_BOTTOM	Bottom scissor

Description

SC_BOTTOM defines the bottom edge of a scissor rectangle. Drawing is inhibited for any pixel which is outside of this scissor rectangle. Scissors are inclusive. This is a signed 15-bit number.

Usage

This register must be set for all draw operations.

See Also

SC_TOP on *page 4-85*

SC_TOP_BOTTOM on *page 4-87*

SC_LEFT on *page 4-82*

SC_RIGHT on *page 4-83*

***mach64* Programmer's Guide:**

- *Engine Operations: Miscellaneous Operations: Scissoring and Masking*

	SC_TOP_BOTTOM I/O:– MM: 0_AD																															
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT/GT	▨	b															▨	a														

a	W	SC_TOP	Top scissor
b	W	SC_BOTTOM	Bottom scissor

Description

SC_TOP_BOTTOM is a composite of registers SC_TOP and SC_BOTTOM.

Usage

This register must be set for all draw operations.

See Also

SC_TOP on *page 4-85*

SC_BOTTOM on *page 4-86*

Data Path

In the **3D RAGE**, six new pixel formats are supported in the Scaler/3D pipeline. The others are packed. DP_SCALE_PIX_WIDTH@DP_PIX_WIDTH is a new field to support these pixel types. Three of these types (RGB332, YUV422, and Y8) are valid destination types and must be supported by the 2D pipeline. DP_DST_PIX_WIDTH@DP_PIX_WIDTH is modified to support these types.

The 3D/Scaler engine does not support packed 24 bpp mode.

		DP_BKGD_CLR	**I/O:–**	**MM: 0_B0**
BITS		31 … 0		
VT/GT		a		
a	R/W	DP_BKGD_CLR	Background color	

Description

DP_BKGD_CLR is used to hold a solid color source. The number of bits used varies depending on graphics modes, as follows:

Video Mode	Bits Used
1 bpp	the least significant bit
4 bpp	the least significant 4 bits
8 bpp	the least significant 8 bits
15 bpp/16 bpp	the least significant 16 bits
packed 24 bpp	the least significant 24 bits
32 bpp	all 32 bits

Usage

Generally, this register is used for the background source in a color expansion of monochrome data.

See Also

mach64 **Programmer's Guide:**

- *Engine Operations: Background Information: Logical Pixel Data Path*

DP_FRGD_CLR (DP_FOG_CLR) **I/O:–** **MM: 0_B1**

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

a	R/W	DP_FRGD_CLR	Foreground color

Description

DP_FRGD_CLR is used to hold a solid color source. The number of bits used varies depending on graphics modes, as follows:

Video Mode	Bits Used
1 bpp	the least significant bit
4 bpp	the least significant 4 bits
8 bpp	the least significant 8 bits
15 bpp/16 bpp	the least significant 16 bits
packed 24 bpp	the least significant 24 bits
32 bpp	all 32 bits

Usage

Generally this register is used for solid color fill or for the foreground source in a color expansion of monochrome data.

In the **3D RAGE**, this register (DP_FOG_CLR) is used to source the solid **Fog** color.

See Also

***mach64* Programmer's Guide:**

- *Engine Operations: Background Information: Logical Pixel Data Path*

		DP_WRITE_MSK																								I/O:–				MM: 0_B2		
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	a																															

a	R/W	DP_WRITE_MSK	Write mask

Description

DP_WRITE_MSK is used to inhibit destination writing of selected bits within a pixel. Each occurrence of a zero in the mask will preserve the content of the destination pixel at that bit position in the pixel. The bits used vary depending on the video modes, as follows:

Video Mode	Bits Used
1 bpp	the least significant bit
4 bpp	the least significant 4 bits
8 bpp	the least significant 8 bits
15 bpp/16 bpp	the least significant 16 bits
packed 24 bpp	the least significant 24 bits
32 bpp	all 32 bits

Usage

All draw operations require this register to be set.

In the **3D RAGE**, when Alpha Blending is enabled, the Destination Read FIFO is unavailable to the 2D engine. This register **must** be set to 0xFFFFFFFFh.

See Also

mach64 **Programmer's Guide:**

- *Engine Operations: Miscellaneous Operations: Scissoring and Masking*

	DP_CHAIN_MSK																								I/O:–					MM: 0_B3		
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	a															

a	R/W	DP_CHAIN_MSK	Chain mask

Description

DP_CHAIN_MSK is the carry chain mask register. Each incidence of a '1' in the mask will inhibit the carry bit in that bit position from adding to the next bit in the pixel ALU. This register is 15 bits wide. There is an implicit carry break in the most significant bit position. This register only affects the (S+D)>>1 mix function.

The normal value for this register should be set according to the table below:

Pixel Depth	DP_CHAIN_MASK
1	N/A
4 bpp pseudocolor	0x8888
8 bpp pseudocolor	0x8080
8 bpp, RGB 332	0x9292
15 bpp, aRGB 1555	0x4210
16 bpp, RGB 565	0x8410
24 bpp, RGB 888	0x8080
32 bpp, RGBa 8888	0x8080

Usage

Set this register only when the foreground mix or background mix is set to function 0x17.

See Also

DP_MIX on *page 4-95*

***mach64* Programmer's Guide:**

- *Engine Operations: Background Information: Source and Destination Mixing Logic*

	DP_PIX_WIDTH																							I/O:–					MM: 0_B4			
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT	▨	▨	▨	▨	▨	▨	▨	f	▨	▨	▨	▨	▨	d			▨	▨	▨	▨	▨	c			▨	▨	▨	▨	▨	a		
GT	j				i	h	g	f	e				d				▨	▨	▨	▨	c				▨	▨	▨	▨	b			

a	R/W	DP_DST_PIX_WIDTH[2:0]	Destination data path pixel width: 0 = monochrome 1 = 4 bpp 2 = 8 bpp 3 = 15 bpp (5,5,5) 4 = 16 bpp (5,6,5) 5 = (reserved) 6 = 32 bpp 7 = (reserved)
b	R/W	DP_DST_PIX_WIDTH[3:0]	Destination data path pixel width: 0 = monochrome 1 = 4 bpp pseudocolor 2 = 8 bpp pseudocolor 3 = 16 bpp aRGB 1555 4 = 16 bpp RGB 565 5 = (reserved) 6 = 32 bpp aRGB 8888 7 = 8 bpp RGB 332 8 = Y8 greyscale 9–10 = (reserved) 11 = YUV 422 packed 12–13 = (reserved) 14 = aYUV 444 (8:8:8:8) 15 = (reserved)
c	R/W	DP_SRC_PIX_WIDTH	Source data path pixel width – bit description same as those for DP_DST_PIX_WIDTH[2:0] ([3:0] for 3D RAGE), shown above.
d	R/W	DP_HOST_PIX_WIDTH	Host data path pixel width – bit description same as those for DP_DST_PIX_WIDTH[2:0] ([3:0] for 3D RAGE), shown above.
e	R/W	DP_CI4_RGB_INDEX	These bits will be used as the upper 4 bits of the C14 color value to select 1 of 16 different pallettes.
f	R/W	DP_BYTE_PIX_ORDER	Reverses the pixel order within each byte in monochrome and 4 bpp modes: 0 = pixel order from MSBit (nibble) to LSBit (nibble). 1 = pixel order from LSBit (nibble) to MSBit (nibble)
g	R/W	DP_CONVERSION_TEMP	YUV to RGB conversion temperature: 0 = red@6500 K, green and blue @9300 K 1 = RGB@9300 K
h	R/W	DP_CI4_RGB_LOW_NIBBLE	Denotes that when in C18 –> RGB texture lookup mode, the texture should be interpreted as 4 bits/pixel, aligned in bits 3-0 of the byte.
i	R/W	DP_CI4_RGB_HIGH_NIBBLE	Denotes that when in C18 –> RGB texture lookup mode, the texture should be interpreted as 4 bits/pixel, aligned in bits 7-4 of the byte.

(continued on next page)

DP_PIX_WIDTH																										I/O:–				MM: 0_B4		
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT	▨	▨	▨	▨	▨	▨	▨	f	▨	▨	▨	▨	▨	d			▨	▨	▨	▨	▨	c			▨	▨	▨	▨	▨	a		
GT	j				i	h	g	f	e				d				▨	▨	▨	▨	c				▨	▨	▨	▨	b			

j	R/W	DP_SCALE_PIX_WIDTH	Scaler source and 3D (texture and shading) data path pixel width: 0–1 = (reserved) 2 = 8 bpp pseudocolor 3 = 15 bpp aRGB 1555 4 = 16 bpp RGB 565 5 = (reserved) 6 = 32 bpp aRGB 8888 7 = 8 bpp RGB8 332 8 = Y8 greyscale 9–10 = (reserved) 11 = YUV 422 packed (YUYV) 12–13 = (reserved) 14 = aYUV 444 (8:8:8:8) 15 = 16 bpp aRGB 4444

Description

DP_PIX_WIDTH specifies the pixel format of the destination area, blit source area, and host data register. Although each may be specified independently, the only pixel format conversions supported by *mach64* are 1 bpp to any pixel size when doing color expansion of monochrome data.

DP_BYTE_PIX_ORDER affects pixel ordering within a byte of data for 1 bpp and 4 bpp modes. This bit affects the pixel order when writing to destination memory or reading from blit source memory. It also affects the interpretation of the HOST_DATA register.

If the display mode is 4 bpp, this field should be set to the same value as CRTC_BYTE_PIX_ORDER@CRTC_GEN_CNTL. These bits should be set only once upon mode initialization.

Usage

This register is used for setting draw engine pixel width and pixel ordering within a byte. The source, host, and destination pixel widths may be specified separately, although only the following combinations are supported for simple colour sources:

Supported Pixel Widths	
Host or Source Pixel Width	**Destination Pixel Width**
1	1, 4, 8, 15, 16, 32
4	4
8	8
15	15
16	16
32	32

In the **3D RAGE**, 8 bpp pseudocolor, Y8, and 8 bpp RGB332 are treated as raw 8 bpp data by the standard draw engine, and are differentiated from one another by the Scaler/3D block, which needs to pack expanded 24 bpp pixels into their respective destination pixel formats.

YUV422 is treated as raw 32 bpp data by the standard draw engine, and is differentiated by the Scaler/3D block

When using the Scaler/3D pipeline, the following combination of scaler source and destination pixel formats may be selected:

Scaler Pipe Pixel Conversions

Scaler Source Pixel Width	Destination Pixel Widths
Pseudo 8	Pseudo 8
Y8	RGB8, 15, 16, 32, Y8 YUV422, YUV444
Pseudo 8 or Y8	RGB 8, 15, 32*
RGB 8	RGB 8, 15, 16, 32
RGB 12	RGB 8, 15, 16, 32
RGB 15	RGB 8, 15, 16, 32
RGB 16	RGB 8, 15, 16, 32
RGB 32	RGB 8, 15, 16, 32
YUV422	RGB8, 15, 16, 32, Y8 YUV422, YUV444
YUV444	RGB8, 15, 16, 32, Y8 YUV422, YUV444

*This combination is only available during Texture Mapping or Scaling. The Pseudocolour-to-RGB conversion is done via a read of the RAMDAC palette.

See Also

mach64 Programmer's Guide:

- *Engine Operations: Draw Operations: Specialized BitBlt Source: Monochrome Expansion*

	DP_MIX																						I/O:–						MM: 0_B5			
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	b					▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	a				

a	R/W	DP_BKGD_MIX	Background Mix. (See table below)
b	R/W	DP_FRGD_MIX	Foreground Mix. (See table below)

Description

DP_MIX specifies the ALU mix function for both foreground and background expansions. If the result of the monochrome pixel consumption is zero, then the ALU uses DP_BKGD_MIX for that pixel; otherwise, DP_FRGD_MIX is used.

Mix Function	Description
0h	(not DST)
1h	“0”
2h	“1”
3h	DST
4h	(not SRC)
5h	DST xor SRC
6h	(not DST) xor SRC
7h	SRC
8h	(not DST) or (not SRC)
9h	DST or (not SRC)
Ah	(not DST) or SRC
Bh	DST or SRC
Ch	DST and SRC
Dh	(not DST) and SRC
Eh	DST and (not SRC)
Fh	(not DST) and (not SRC)
10h-16h	Reserved
17h	(DST+SRC)/2 (Reserved in 3D RAGE)
18h-1Fh	Reserved

Usage

DP_FRGD_MIX must always be set. DP_BKGD_MIX is *don’t_care* for non-trivial color expansion of monochrome data. A non-trivial monochrome source is anything but *Always_'1'*.

In the **3D RAGE**, when Alpha Blending is enabled, the Destination Read FIFO is unavailable to the 2D engine. In this case, DP_MIX **must not** use the Destination.

See Also

DP_MONO_SRC@DP_SRC on *page 4-97*

***mach64* Programmer’s Guide:**

- *Engine Operations: Background Information: Logical Pixel Data Path*
- *Engine Operations: Background Information: Source and Destination Mixing Logic*

	DP_SRC																									I/O:–				MM: 0_B6		
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	c		▨	▨	▨	▨	▨	b			▨	▨	▨	▨	▨	a		

a	R/W	DP_BKGD_SRC	Background source: 0 = Background color 1 = Foreground color 2 = Host data 3 = Blit source 4 = Pattern registers 5 = Scaler/3D data (3D RAGE); (reserved in VT) 6-7 = (reserved)
b	R/W	DP_FRGD_SRC	Foreground source – bit descriptions same as those for DP_BKGD_SRC[2:0], shown above.
c	R/W	DP_MONO_SRC	Monochrome source: 0 = ‘1’ 1 = Pattern registers 2 = Host data 3 = Blit source

Description

DP_SRC controls the mono mux and the two color muxes in the pixel data path.

Usage

DP_FRGD_SRC and DP_MONO_SRC are required to be set for all draw operations. DP_BKGD_SRC is *don’t_care* for non-trivial color expansion of monochrome data. A non-trivial monochrome source is anything but *Always_'1'*.

See Also

***mach64* Programmer’s Guide:**

- *Engine Operations: Background Information: Logical Pixel Data Path*

Color Compare

The color compare function allows color keying on destination or source color values. Note that the color comparison function is not supported in 1 bpp mode.

In the **3D RAGE**, when color keying on the Texel source, the key is compared against the **expanded** (24 bit) source. When color keying 8 bit pseudo color sources, the source data is located on the low order 8 bits.

	CLR_CMP_CLR																							**I/O:–**					**MM: 0_C0**			
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT/GT	a																															

a	R/W	CLR_CMP_CLR	Color comparison color

Description

CLR_CMP_CLR is compared against the source or destination data to determine whether source data will overwrite the destination data.

Usage

Use this register only when CLR_CMP_FN@CLR_CMP_CNTL is set to a non-trivial compare function.

See Also

CLR_CMP_CNTL on *page 4-100*

CLR_CMP_MSK on *page 4-99*

***mach64* Programmer's Guide:**

- *Engine Operations: Draw Operations: Specialized BitBlt Source: Transparent BitBlts*

	CLR_CMP_MSK																						I/O:–					MM: 0_C1			
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	a																															

a	R/W	CLR_CMP_MSK	Color comparison mask

Description

The CLR_CMP_MSK register is used in conjunction with CLR_CMP_FN. Both CLR_CMP_CLR and the source/destination data are masked by the color comparison mask.

Usage

Use this register only when CLR_CMP_FN@CLR_CMP_CNTL is set to a non-trivial compare function.

See Also

CLR_CMP_CLR on *page 4-98*

CLR_CMP_CNTL on *page 4-100*

***mach64* Programmer's Guide:**

- *Engine Operations: Draw Operations: Specialized BitBlt Source: Transparent BitBlts*

	CLR_CMP_CNTL																								I/O:–				MM: 0_C2			
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	▨	▨	▨	▨	▨	▨	b	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	a		
GT	▨	▨	▨	▨	▨	▨	c		▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	a		

a	R/W	CLR_CMP_FCN	Color comparison function: 0 = False 1 = True 2-3 = (reserved) 4 = DST_CLR != CLR_CMP_CLR 5 = DST_CLR = CLR_CMP_CLR 6-7 = (reserved)
b	R/W	CLR_CMP_SRC	Defines source for color keying: 0 = Destination 1 = Source
c	R/W	CLR_CMP_SRC	Defines source for color keying: 0 = Destination 1 = 2D Source 2 = Texel Source/Scaler Source 3 = Reserved

Description

CLR_CMP_CNTL is used to configure the source or destination compare logic.

CLR_CMP_SRC determines whether the CLR_CMP_CLR register is to be compared against the source or the destination data.

CLR_CMP_FN determines the compare function. If the result of the comparison is false, then color source data is written to the destination; otherwise destination data is written to the destination.

Setting CLR_CMP_FN to any function other than FALSE or TRUE when CLR_CMP_SRC is set for destination keying will automatically cause the destination operation to be read-modify-write.

Usage

This register is used to selectively inhibit the drawing of certain pixels which key on the source data or destination data.

See Also

CLR_CMP_CLR on *page 4-98*

CLR_CMP_MSK on *page 4-99*

***mach64* Programmer's Guide:**

- *Engine Operations: Background Information: Logical Pixel Data Path*
- *Engine Operations: Draw Operations: Specialized BitBlt Source: Transparent BitBlts*

Command FIFO

FIFO_STAT **I/O:–** **MM: 0_C4**

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

a	R	FIFO_STAT	Register indicates the number of full command FIFO entries (For 3D RAGE, register indicates the number of full command FIFO entries in the top half of the FIFO entries. The number is represented as a single bit per entry. Thus if there are 23 full entries, then bits 6:0 will be set. The number of free entries in an encoded form is available in the GUI_STAT register.
b	R	FIFO_ERR	FIFO overun error

Description

Reading FIFO_STAT returns the status of the command FIFO. Any occurrence of a ‘1’ in the FIFO_STAT field indicates that the corresponding FIFO entry is filled. Writing to the command FIFO when insufficient entries are available will cause the FIFO_ERR bit to go high and lock the draw engine. This circumstance should never occur. An interrupt may be wired to the FIFO_ERR bit for debugging purposes through BUS_CNTL. The draw engine may reset the error condition through GEN_TEST_CNTL.

Only registers with DWORD indices greater than or equal to 0x40 go through the command FIFO. All other registers bypass the FIFO.

Usage

Each grouping of register writes through the command FIFO must be preceded by a FIFO check to ensure that sufficient entries are available.

See Also

BUS_CNTL on *page 4-8*

GEN_TEST_CNTL on *page 4-14*

***mach64* Programmer’s Guide:**

- *Engine Initialization:Background Information on the mach64 Engine: FIFO Queue*
- *Engine Operations: Draw Operations*

Context Control

	CONTEXT_MASK		I/O:–	MM: 0_C8
BITS	31–0			
VT/GT	a			

a	R/W	CONTEXT_MASK	Context mask. Each bit in the mask which is set to ‘1’ enables the corresponding register to be loaded from the context buffer. The mapping of mask bits to registers is indicated the *mach64 Programmer’s Guide*, under *Engine Operations: Draw Engine Contexts*.

Description

CONTEXT_MASK masks the loading of registers for context load operations. Each bit in this register corresponds to a DWORD entry in the context load structure. For instance, bit 2 corresponds to DWORD entry 2, the DST_OFF_PITCH entry.

In context load operations, both the CONTEXT_MASK *entry* and CONTEXT_LOAD_CNTL *entry* are always loaded.

Usage

Applications do not need to touch this register. Context load operations use the CONTEXT_MASK entry in the context save structure.

See Also

CONTEXT_LOAD_CNTL on *page 4-103*

***mach64* Programmer’s Guide:**

- *Engine Operations: Draw Operations*
- *Engine Operations: Draw Engine Contexts*
- *Engine Operations: Draw Engine Contexts: Saving and Restoring a Context*

	CONTEXT_LOAD_CNTL																I/O:–								MM: 0_CB							
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT/GT	c	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	b	b	▨	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a

a	R/W	CONTEXT_LOAD_PNTR	Context load pointer
b	R/W	CONTEXT_LOAD_CMD	Context load cmd 0 = no context load 1 = load context from CONTEXT_LOAD_PNTR 2 = load context from CONTEXT_LOAD_PNTR and initiate rectangular fill 3 = load context from CONTEXT_LOAD_PNTR and initiate bresenham line (For 3D RAGE, 3 = load context from CONTEXT_LOAD_PNTR and initiate bresenham line or trapezoid draw. If bit 15 of DST_BRES_LNTH or DST_WIDTH is set, a trapezoid draw will be done.)
c	R/W	CONTEXT_LOAD_DIS	Disables context command from executing. Note that this bit is ignored when this register is loaded within a context. The context command will always execute in this case. 0 = execute context command 1 = don't execute context command

Description

Writing to register CONTEXT_LOAD_CNTL will initiate a context load and optionally perform a draw operation.

On a context load, the CONTEXT_MASK *entry* specified in the context load area determines which register will be loaded. The CONTEXT_MASK *register* is ignored for this operation.

The CONTEXT_LOAD_CNTL *entry* in the context save structure must specify a no-op to halt the chain; otherwise the context will load and execute the next context in the chain.

Context pointers are specified in 64 DWORD chunks in reverse order from top of memory.

CONTEXT_LOAD_DIS in the CONTEXT_LOAD_CNTL *entry* is ignored during a context load and cannot be used to prevent chaining of contexts.

Usage

This register is used to load a default context into the draw engine or to execute a context chain.

See Also

CONTEXT_MASK on *page 4-102*

***mach64* Programmer's Guide:**

- *Engine Operations: Draw Operations*
- *Engine Operations: Draw Engine Contexts*
- *Engine Operations: Draw Engine Contexts: Saving and Restoring a Context*

Draw Engine Composite Control

	GUI_TRAJ_CNTL																										I/O:–				MM: 0_CC	
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	▨	y	x	▨	w	v	u	▨	▨	▨	s	r	q	p	o	▨	▨	▨	k	j		i		h	g	f	e	d	c	b	a
GT	▨	▨	y	x	▨	w	v	u	t	▨	▨	s	r	q	p	o	n	m	l	k	j		i		h	g	f	e	d	c	b	a

a	R/W	DST_X_DIR	Destination X direction 0 = right to left 1 = left to right
b	R/W	DST_Y_DIR	Destination Y direction 0 = bottom to top 1 = top to bottom
c	R/W	DST_Y_MAJOR	Destination Y major axis flag for bresenham lines 0 = X major line 1 = Y major line
d	R/W	DST_X_TILE	Enables rectangular tiling in the X direction
e	R/W	DST_Y_TILE	Enables rectangular tiling in the Y direction
f	R/W	DST_LAST_PEL	Destination last pel enable
g	R/W	DST_POLYGON_EN	Destination polygon outline and polygon fill enable
h	R/W	DST_24_ROT_EN	Enables 24 bpp rotation. DSTPIXWIDTH **must** be set to 8 bpp.
i	R/W	DST_24_ROT	Initial foreground color, background color, write mask, and monochrome pattern rotation when drawing packed 24 bpp.
j	R/W	DST_BRES_SIGN	Sign of DEST_BRES_ERR when DEST_BRES_ERR = 0 0 = DEST_BRES_ERR = 0 is defined as a positive number 1 = DEST_BRES_ERR = 0 is defined as negative number
k	R/W	DST_POLYGON_RTEDGE_DIS	Disables drawing of the right edge pixel of a polygon fill operation. 0 = drawing of right edge pixel is enabled 1 = drawing of right edge pixel is disabled
l	R/W	TRAIL_X_DIR	Trapezoid trailing edge direction. 0 = right to left 1 = left to right
m	R/W	TRAP_FILL_DIR	Trapezoid fill direction 0 = right to left (trailing edge is to the left of the leading edge) 1 = left to right (trailing edge is to the right of the leading edge)
n	R/W	TRAIL_BRES_SIGN	Sign of TRAIL_BRES_ERR when TRAIL_BRES_ERR = 0 0 = TRAIL_BRES_ERR = 0 is defined as a positive number 1 = TRAIL_BRES_ERR = 0 is defined as negative number
o	R/W	SRC_PATT_EN	Enables patten source. SRC_Y_END will only be used if this bit is enabled.
p	R/W	SRC_PATT_ROT_EN	Enables pattern source rotation. SRC_X_START, SRC_Y_START will only be used if this bit is enabled
q	R/W	SRC_LINEAR_EN	Enables the source to be advanced linearly in memory. The source starts at SRC_OFFSET and advances in the left-to-right direction. DST_X_DIR should also be set to the left-to-right to operate properly. Note that all other source registers and control bits with the exception of SRC_BYTE_ALIGN are ignored.

(continued on next page)

GUI_TRAJ_CNTL **I/O:–** **MM: 0_CC**

BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT	▨	▨	y	x	▨	w	v	u	▨	▨	▨	s	r	q	p	o	▨	▨	▨	k	j		i		h	g	f	e	d	c	b	a
GT	▨	▨	y	x	▨	w	v	u	t	▨	▨	s	r	q	p	o	n	m	l	k	j		i		h	g	f	e	d	c	b	a

r	R/W	SRC_BYTE_ALIGN	Allows the source to skip to the next data byte boundary when the destination advances in the Y direction. SRC_LINEAR_EN **must** be set.
s	R/W	SRC_LINE_X_DIR	Source X direction when drawing operation is a bresenham line.
t	R/W	SRC_TRACK_DST	Source will track the trajectory which the DST FIFO is using
u	R/W	PAT_MONO_EN	Monochrome 8x8 pattern enable
v	R/W	PAT_CLR_4x2_EN	Color 4x2 pattern enable
w	R/W	PAT_CLR_8x1_EN	Color 8x1 pattern enable
x	R/W	HOST_BYTE_ALIGN	Enables byte alignment of the host data
y	R/W	HOST_BIG_ENDIAN_EN	Enables big endian data translation for 15 bpp, 16 bpp, and 32 bpp pixel widths. In 15 bpp and 16 bpp modes, the bytes within each word are swapped. In 32 bpp mode, the order of the four bytes within each DWORD is reversed. 0 = big endian data translation disabled 1 = big endian data translation enabled

Description

GUI_TRAJ_CNTL is a composite of registers DST_CNTL, SRC_CNTL, PAT_CNTL, and HOST_CNTL.

Usage

This register is used for general draw operations.

See Also

DST_CNTL on *page 4-48*

SRC_CNTL on *page 4-62*

PAT_CNTL on *page 4-81*

HOST_CNTL on *page 4-78*

Draw Engine Status

	GUI_STAT																									I/O:–				MM: 0_CE		
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	e	d	c	b	▨	▨	▨	▨	▨	▨	▨	a
GT	▨	▨	▨	▨	▨	▨	▨	▨	f								▨	▨	▨	▨	e	d	c	b	▨	▨	▨	▨	▨	▨	▨	a

a	R	GUI_ACTIVE	Indicates that GUI engine is busy For 3D RAGE: Indicates that the GUI engine is busy OR the 3D engine is busy OR the command FIFO is not empty OR context loading is occurring
b	R	DSTX_LT_SCISSOR_LEFT	Indicates DSTX is left of left scissor
c	R	DSTX_GT_SICISSOR_RIGHT	Indicates DSTX is right of right scissor
d	R	DSTY_LT_SCISSOR_TOP	Indicates DSTY is above top scissor
e	R	DSTY_GT_SCISSOR_BOTTOM	Indicates DSTY is below bottom scissor
f	R	FIFO_CNT	Indicates the number of available (empty) entries in the command FIFO. In the current version of the 3D RAGE, this number is less than or equal to 32.

Description

GUI_STAT reports the status of the draw engine.

Usage

The GUI_ACTIVE bit is used to determine whether the draw engine is busy or idle. All status bits in this register should be read-only when the draw engine is idle.

For the **3D RAGE**, the parameter FIFO is expanded to 48 entries. A new field (FIFO_CNT) is added to GUI_STAT to contain the number of available (empty) FIFO entries.

See Also

FIFO_STAT on *page 4-101*

***mach64* Programmer's Guide:**

- *Engine Initialization: Background Information on the mach64 Engine: FIFO Queue*

Chapter 5
Multimedia Register Reference

Overlay Registers

The overlay registers define the window region in which the scaled data will be placed. Within the overlay area, data is not displayed until the overlay key colour is matched in the destination (as defined by the mask/key colour). Coordinates (0,0) are defined as the top-left corner start of the **active** display.

Overlay Window Control

The following registers define the overlay window size and coordinates relative to the start of the active display. In addition, the overlay colour keying registers are defined to provide control for selecting between video and graphics sources.

	OVERLAY_GRAPHICS_KEY_CLR																								I/O:–					MM: 1_04		
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	▨	▨	▨	▨	▨	▨	▨	a																							

a	R/W	GRAPHICS_KEY_CLR	Overlay Graphics Key Colour

Description

This register is used as the key colour in the graphics keyer. This register is masked with OVERLAY_ GRAPHICS_KEY_MSK and compared (equality) with the current graphics pixel masked with OVERLAY_ GRAPHICS_KEY_MSK. The result of this comparison is used with the result of the video keyer according to the OVERLAY_KEY_CNTL to determine whether graphics or scaler video should be displayed on the screen.

Usage

The conventional 'RGB' order for fields within a pixel is followed, where 'R' is the highest byte (23:20) and 'B' is the lowest byte (7:0). For lower pixel depths, (e.g., 16 bit 565), the colour and mask registers represent the compact format of the colour. For example, in 565, the BLUE resides in the lower 5 bits (4:0) of the mask and colour, GREEN is in the subsequent 6 bits (10:5), and RED is in the upper 5 bits (15:11). Bits (24:16) must be masked in this mode.

The graphics colour and mask registers must take into account the current graphics pixel depth. In other words, the unused bits of the 24 bit fields should be masked.

See Also

OVERLAY_ GRAPHICS_KEY_MSK on *page 5-2*

OVERLAY_KEY_CNTL on *page 5-3*

OVERLAY_GRAPHICS_KEY_MSK I/O:– MM: 1_05

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

a	R/W	GRAPHICS_KEY_MSK	Overlay Graphics Key Colour Mask

Description

This register is used in the graphics keyer as a mask for the graphics pixel and the key colour (OVERLAY_GRAPHICS_KEY_CLR).

Usage

The conventional ‘RGB’ order for fields within a pixel is followed.

See Also

OVERLAY_ GRAPHICS_KEY_CLR on *page 5-4*

OVERLAY_KEY_CNTL on *page 5-3*

	OVERLAY_KEY_CNTL																								I/O:–					MM: 1_06		
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	d	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	c				▨	b			▨	a		

a	R/W	VIDEO_KEY_FN	Overlay Video Comparison Fn: (default = 0) 0 = False 1 = True 2-3 (reserved) 4 =(VID_CLR&VID_KEY_MSK)!=(VID_KEY_CLR&VID_KEY_MSK) 5 =(VID_CLR&VID_KEY_MSK) =(VID_KEY_CLR&VID_KEY_MSK)
b	R/W	GRAPHICS_KEY_FN	Overlay Graphics Comparison Fn: (default = 0) 0 = False 1 = True 2-3 (reserved) 4 =(GR_CLR & GR_KEY_MSK)!=(GR_KEY_CLR&GR_KEY_MSK) 5 =(GR_CLR& GR_KEY_MSK) =(GR_KEY_CLR&GR_KEY_MSK)
c	R/W	OVERLAY_CMP_MIX	0 = GR_CMP (default = 0) 1 = Always Graphics 2 = Always Video 3 = not GR_CMP 4 = not VID_CMP 5 = VID_CMP xor GR_CMP 6 = (not GR_CMP) xor VID_CMP 7 = VID_CMP 8 = (not GR_CMP) or (not VID_CMP) 9 = GR_CMP or (not VID_CMP) A = (not GR_CMP) or VID_CMP B = GR_CMP or VID_CMP C = GR_CMP and VID_CMP D = (not GR_CMP) and VID_CMP E = GR_CMP and (not VID_CMP) F = (not GR_CMP) and (not VID_CMP)
d	R/W	OVERLAY_EXCLUSIVE_EN	Enable suppression of graphics stream reading during overlay window: 0 = Normal 1 = Video stream only (suppress graphics)

Description

This register determines how the video and graphics keyer results are generated and then combined to select either video or graphics. A result of ‘1’ means that video will be displayed within the region defined by the overlay coordinates. Otherwise, graphics is displayed.

Usage

VIDEO_KEY_FN and GRAPHICS_KEY_FN determine which video and graphics keyer function should be used in the respective keyers. The ‘true’ and ‘false’ settings force a constant result, regardless of video or graphics data. The OVERLAY_CMP_MIX determines how the results from the video and graphics keyers are combined.

See Also

OVERLAY_ GRAPHICS_KEY_CLR on *page 5-1*

OVERLAY_ GRAPHICS_KEY_MSK on *page 5-2*

OVERLAY_ VIDEO_KEY_CLR on *page 5-4*

OVERLAY_ VIDEO_KEY_MSK on *page 5-5*

	OVERLAY_VIDEO_KEY_CLR																								I/O:–				MM: 1_02			
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT	▨	▨	▨	▨	▨	▨	▨	▨	a																							

a	R/W	VIDEO_KEY_CLR	Overlay Video Key Colour

Description

This register is used as the key colour in the video keyer. This register is masked with OVERLAY_ VIDEO_KEY_MSK and compared (equality) with the current video pixel masked with OVERLAY_ VIDEO_KEY_MSK. The result of this comparison is used with the result of the graphics keyer (according to OVERLAY_KEY_CNTL) to determine whether graphics or scaler video should be displayed on the screen.

Usage

The video from the scaler is always 24 bits in the ordered ‘RGB’, so the video colour and mask registers should be applied accordingly using all 24 bits.

See Also

OVERLAY_ VIDEO_KEY_MSK on *page 5-5*

OVERLAY_KEY_CNTL on *page 5-3*

	OVERLAY_VIDEO_KEY_MSK																						I/O:–						MM: 1_03		
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	▨	▨	▨	▨	▨	▨	▨	a																							

a	R/W	VIDEO_KEY_MSK	Overlay Video Key Colour Mask

Description

This register is used in the video keyer as a mask for the video pixel key and the video key colour (OVERLAY_VIDEO_KEY_CLR).

Usage

The video from the scaler is always 24 bits in the ordered ‘RGB’, so the video colour and mask registers should be applied accordingly using all 24 bits.

See Also

OVERLAY_ VIDEO_KEY_CLR on *page 5-4*

OVERLAY_KEY_CNTL on *page 5-3*

OVERLAY_Y_X_START I/O:– MM: 1_00

BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT-A3/GT	▨					b											▨					a										
VT-A4	c	▨				b											▨					a										

a	R/W	OVERLAY_Y	Overlay Y coordinate relative to ACTIVE screen (0,0)
b	R/W	OVERLAY_X	Overlay X coordinate relative to ACTIVE screen (0,0)
c	R/W	OVERLAY_Y_X_LOCK	Prevents latching overlay coordinates until unlocked: 0 = Unlock both OVERLAY_Y_X_START and OVERLAY_Y_X_END 1 = Lock both OVERLAY_Y_X_START and OVERLAY_Y_X_END

Description

This register specfies the vertical (OVERLAY_Y) and horizontal (OVERLAY_X) start of the overlay window relative to the start of active display.

Usage

The start and end coordinates are inclusive. The values programmed must be in the active display. Refer to the Programmer's Guide for an explanation of some programming restrictions for interlaced modes.

For VT-A4, the OVERLAY_Y_X_LOCK bit is used to prevent changing the overlay coordinates until both the start and end registers are written. Writing either the start or end register with this bit set to 1 will lock both registers (i.e., prevent the overlay controller hardware from seeing the new values). When either register is written with this bit set to a zero, the pair are unlocked and the overlay controller hardware will see the new values on the next VSYNC.

See Also

OVERLAY_Y_X_END on *page 5-7*

OVERLAY_Y_X_END I/O:– MM: 1_01

BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT-A3/GT	▨	▨	▨	▨	▨	b											▨	▨	▨	▨	▨	a										
VT-A4	c	▨	▨	▨	▨	b											▨	▨	▨	▨	▨	a										

a	R/W	OVERLAY_Y_END	Overlay Y ending coordinate relative to ACTIVE screen (0,0)
b	R/W	OVERLAY_X_END	Overlay X ending coordinate relative to ACTIVE screen (0,0)
c	R/W	OVERLAY_Y_X_LOCK	Prevents latching overlay coordinates until unlocked: 0 = Unlock both OVERLAY_Y_X and OVERLAY_Y_X_END 1 = Lock both OVERLAY_Y_X and OVERLAY_Y_X_END

Description

This register specfies the vertical (OVERLAY_Y_END) and horizontal (OVERLAY_X_END) end coordinates of the overlay window relative to the start of active display.

Usage

The start and end coordinates are inclusive. The values programmed must be in the active display. Refer to the Programmer's Guide for an explanation of some programming restrictions for interlaced modes.

For VT-A4, the OVERLAY_Y_X_LOCK bit is used to prevent changing the overlay coordinates until both the start and end registers are written. Writing either the start or end register with this bit set to 1 will lock both registers (i.e., prevent the overlay controller hardware from seeing the new values). When either register is written with this bit set to a zero, the pair are unlocked and the overlay controller hardware will see the new values on the next VSYNC.

The overlay will not clip the end coordinates past the active display region. OVERLAY_Y_X_END must not be programmed beyond the end of the active display.

See Also

OVERLAY_Y_X_START on *page 5-6*

Overlay Scaler

The following registers define the overlay scaling control registers.

OVERLAY_SCALE_CNTL   I/O:–   MM: 1_09

BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT-A3/GT	k	j	i	▨	▨	h	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	e	d	c	b	a
VT-A4N	k	j	i	▨	▨	h	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	f	d	c	b	a
VT-A4S	k	j	i	▨	▨	h	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	g		f	d	c	b	a

a	R/W	SCALE_PIX_EXPAND	Pixel expansion algorithm: 0 = zero extend 1 = dynamic range correct
b	R/W	SCALE_Y2R_TEMP	0 = Red Temp: 6500K 1 = Red Temp: 9800K
c	R/W	SCALE_HORZ_MODE	0 = Horizontal blend (when available) 1 = Horizontal pixel replication: Fix H_ALPHA = 0
d	R/W	SCALE_VERT_MODE	0 = Vertical blend (when available) 1 = Vertical line replication: Fix V_ALPHA = 0
e	R/W	LAST_DISPREQ_DROP_EARLY	0 = Disable 1 = Enable (1Mb double scan fix)
f	R/W	SCALE_SIGNED_UV	0 = Incoming UV is unsigned (default) 1 = Incoming UV is signed
g	R/W	SCALE_GAMMA_SEL	Select gamma correction factor: 0 = Gamma off (default) 1 = Gamma 2.2 2 = Gamma 1.8 3 = Gamma 1.4
h	R/W	SCALE_BANDWIDTH	Reset Bandwidth Status: (W) (default = 0) 1 = Reset Bandwidth to 0 Status of Scaler: (R) 0 = Normal 1 = Bandwidth exceeded
i	R/W	SCALE_CLK_FORCE_ON	Dynamic Scaler Clock Control: (default = 0) 0 = scaler clock controlled by scaler activity 1 = scaler clock runs continuously
j	R/W	OVERLAY_EN	0 = Overlay Disable (default=0) 1 = Overlay Enable
k	R/W	SCALE_EN	0 = Reset scaler internal states (default=0) 1 = Enable scaler operation

Description

This register contains miscellaneous control bits for the scaler.

Usage

When scaling an RGB source, pixel data is converted internally to 16-bit (565) prior to scaling and pixel replication being used. Thus, for an RGB source, SCALE_HORZ_MODE and SCALE_VERT_MODE are ignored and both horizontal and vertical alphas are fixed to zero.

For VT-A4N/S, SCALE_SIGNED_UV is added to enable the conversion of signed UV to unsigned UV for the scaler.

For VT-A4S, SCALE_GAMMA_SEL provides selection of the gamma correction factor (In VT-A4S, gamma correction was added in the scaler YUV to RGB converter).

See Also

SCALER_IN@VIDEO_FORMAT on *page 5-16*

	OVERLAY_SCALE_INC																								**I/O:–**					**MM: 1_08**		
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT	b																a															

a	R/W	VERT_INC	Vertical accumulator increment, 4.12 (integer.fraction) format.
b	R/W	HORZ_INC	Horizontal accumulator increment, 4.12 (integer.fraction) format.

Description

This register specifies the vertical (VERT_INC) and horizontal (HORZ_INC) scale factors (or increments).

Usage

Typically, the scale factors are programmed according to the following equation.

VERT_INC = (src_height << 12)/dst_height

HORZ_INC = (src_width << 12)/dst_width

This takes into account the fact that the format of both increments is 4.12 (integer.fraction).

See Also

SCALER_HEIGHT_WIDTH on *page 5-12*

OVERLAY_Y_X_START on *page 5-6*

OVERLAY_Y_X_END on *page 5-7*

	OVERLAY_TEST																						**I/O:–**						**MM: 1_0B**			
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	c																▨	▨	▨	▨	▨	▨	▨	▨	b				▨	▨	a	▨

a	R/W	SCALE_Y2R_DIS	0 = Normal operation (default = 0) 1 = Disable YUV to RGB conversion on output
b	R/W	SCALE_REG_READ_MODE	Set SCALE_REG_READ field for readback: 0 = Horz_Acc 1 = Vert_Acc 2 = Scale Read/Write address (concatenated) 3-8 = reserved 9 = Scaler source width count 10-15 = reserved
c	R	SCALE_REG_READ	Read back internal scaler value

Description

This register is used for test purposes only.

	SCALER_HEIGHT_WIDTH																								I/O:–				MM: 1_0A			
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT	▨	▨	▨	▨	▨	▨	b										▨	▨	▨	▨	▨	▨	a									

a	R/W	SCALER_HEIGHT	Scaler source height
b	R/W	SCALER_WIDTH	Scaler source width in terms of pixels

Description

This register is used to specify the height and width of the scaler source data in the frame buffer memory.

Usage

The SCALER_HEIGHT is the number of lines in the source. If the VERT_INC and the overlay coordinates are programmed such that dst_height * VERT_INC >> 12 is greater than SCALER_HEIGHT, then the last line in the source will be used until the end of the overlay.

The SCALER_WIDTH is the number of pixels for the width of the source. If the HORZ_INC and the overlay coordinates are programmed such that dst_width * HORZ_INC >> 12 is greater than SCALER_WIDTH, then the last pixel in the source will be used until the end of the overlay.

See Also

OVERLAY_SCALE_INC on *page 5-10*

OVERLAY_Y_X_START on *page 5-6*

OVERLAY_Y_X_END on *page 5-7*

SCALER_THRESHOLD **I/O:–** **MM: 1_0C**

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

a	R	SCALER_SOURCE_LINE	Current scaler source line (counts down from height-1 to 0)
b	R/W	SCALER_THRESHOLD_LINE	Scaler threshold line (Default = 0)

Description

This register specifies the scaler source threshold.

Usage

SCALER_SOURCE_LINE (read-only) is initially loaded with SCALER_HEIGHT – 1, and counts down to zero. SCALER_THRESHOLD_LINE holds off writing to the same buffer as the scaler is reading from until the threshold is crossed.

Depending on the scale ratio, the scaler may skip lines and, therefore, not count down by 1. Thus, the last line that can be reached in the scaler source is zero.

See Also

SCALER_HEIGHT@SCALER_HEIGHT_WIDTH on *page 5-12*

General Video Configuration/Control Registers

The following set of registers is used for general control and configuration of the VT or 3D RAGE video port used for video capture. It should be noted that for **video input**, the offset and pitch must be even-pixel aligned to **quadword** boundaries.

General Video

VIDEO_CONFIG **I/O:–** **MM: 1_13**

BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	i		h		g				▨	f	▨	e	d	c	b	a

a	R/W	VIDEO_MODE	Set VT/3D RAGE to video input/output mode: 0 = Video In 1 = (Reserved)
b	R/W	VIDEO_DEVICE	Determines which device will input video: 0 = VMC 1 = Video (based on VIDEO_INPUT_TYPE)
c	R/W	VIDEO_HSYNC_POL	Phillips compatible mode Hsync Polarity: 0 = -ve polarity 1 = +ve polarity
d	R/W	VIDEO_VSYNC_POL	Philllips compatible mode Vsync Polarity: 0 = -ve polarity 1 = +ve polarity
e	R/W	VIDEO_IO_SIZE	0 = 8 Bit 1 = (Reserved)
f	R/W	VIDEO_FIELD_FLIP	Flips the sense of the Video Field: 0 = Normal 1 = Flip
g	R/W	VIDEO_INPUT_TYPE	0 = Parallel input control CCIR 556 (Phillips compatible pin mode) 1 = (reserved) 2 = (reserved) 3 = Embedded synchronization (Brooktree compatible) 15-4 = (reserved)
h	R/W	VIDEO_HORZ_DOWN	Horizontal Reduction: (decimation) 0 = 1:1 unity 1 = 2:1 pixel skipping 2 = 4:1 pixel skipping 3 = (reserved)
i	R/W	VIDEO_VERT_DOWN	Vertical Reduction: (decimation) 0 = 1:1 unity 1 = 2:1 line skipping 2 = 4:1 line skipping 3 = (reserved)

Description

This register determines the capability and settings for the video input device attached to the VT, and the down-scaling factor on the input video stream.

The bit settings reflect the device configuration such as the horizontal/vertical polatrities, supported input/output format, and other device-specific capabilites.

Usage

To set the VT/3D RAGE for connection to a Brooktree-type video decoder, specify VIDEO_DEVICE = Video and VIDEO_INPUT_TYPE = Embedded synchronization.

If the video decoder has **no** down-scaling capability, VIDEO_HORZ_DOWN should be set to 1 (2:1 pixel skipping); otherwise, the down-scaling should be done in the video decoder, and VIDEO_HORZ_DOWN and VIDEO_VERT_DOWN should be set to zero (1:1).

	VIDEO_FORMAT																							I/O:–					MM: 1_12			
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT-A3/A4N	e		d	▨	▨	▨	▨	▨	▨	▨	▨	▨	b				▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	a			
VT-A4S	e		d	c	▨	▨	▨	▨	▨	▨	▨	▨	b				▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	a			

a	R/W	VIDEO_IN	Video Input Formats: 0-1 = reserved 2 = 8 bpp 3 = (reserved) 4 = 16 bpp 5 = (reserved) 6 = 32 bpp 7-10 = (reserved) 11 = VYUY422 (16 Bpp) 12 = YVYU422 (16Bpp) 13-15 = (reserved)
b	R/W	SCALER_IN	Scale source pixel format from memory: 0-1 = reserved 2 = reserved 3 = 15 bpp aRGB 1555 4 = 16 bpp RGB 565 5 = (reserved) 6 = 32 bpp aRGB 8888 7 = (reserved) 8 = reserved 9 = YUV 9 10 = YUV 12 11 = VYUY422 12 = YVYU422 13-15 = (reserved)
c	R/W	HOST_BYTE_SHIFT_EN	Shifts each byte written to the aperture left one bit and discards the upper bit (the lower bit of each byte is set to zero) 0 = Normal 1 = Byte shift *This only applies to the current aperture set by HOST_YUV_APER, and only applies to aperture writes.
d	R/W	HOST_YUV_APER	Select aperture section for Y/U/V HOST_MEM_MODE and HOST_BYTE_SHIFT_EN (VT-A4S): 0 = Upper half of enabled aperture 1 = Lower half of enabled aperture
e	R/W	HOST_MEM_MODE	Host aperture memory mode: 0 = normal 1 = Y mode 2 = U mode 3 = V mode

Description

This register specifies the input video format and scaler video format. HOST_YUV_APER and HOST_MEM_MODE are used for planar-to-packed data writes from the host to the video buffer.

Usage

To set the scaler to scale the input video stream, VIDEO_IN and SCALER_IN should have the same format. SCALER_IN can be specified independently of VIDEO_IN when the scaler is used to scale software MPEG or AVI video data written from the host.

The HOST_YUV_APER bit enables the upper or lower half of the aperture to be used for converting YUV planar data into YUV packed data, as the data is written from the host to frame buffer memory.

HOST_MEM_MODE is used to tell the hardware which type of data is being transferred from the host, and is programmed for every planar Y/U/V plane before the data transfer starts.

HOST_BYTE_SHIFT_EN (VT-A4S only) adds a mechanism to shift each Y/U/V byte up by one bit as the data is written to the frame buffer memory.

The scaler interprets its current input buffer format according to the SCALER_IN setting. For scaler output, when the scaler is configured to output directly to the DAC, it will **always** output 24 bpp.

SCALER_IN must be set to the appropriate **YUV** mode when scaling, even though the data is placed in packed mode through the aperture (using HOST_MEM_MODE@VIDEO_FORMAT).

VIDEO_IN must be set to the same format as the video data coming in. The VT/3D RAGE does not format or convert the data on video input. This field is only used to determine the size of the incoming pixel to associate with the current buffer used for capture (whose size is defined in terms of pixels).

YUV422 is interpreted as a pixel width of 16 bpp. As such, all associated buffer widths and pitches must be programmed to EVEN quantities. CAPTURE_X@ CAPTURE_Y_X must be even as well.

See Also

CAPTURE_Y_X on *page 5-21*

Video Capture

	CAPTURE_CONFIG																								**I/O:–**				**MM: 1_14**			
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT												j	i	h	g										f	e	d	c	b		a	

a	R/W	VIDEO_IN_CAP_EVEN	Video Input Even Capture Mode: 0 = Disable 1 = Auto - Continuous capture 2 = Host - Capture when triggered by host 3 = (reserved)
b	R/W	VIDEO_IN_CAP_ODD	Video Input Odd Capture Mode: 0 = Disable 1 = Auto - Continuous capture 2 = Host - Capture when triggered by host 3 = (reserved)
c	R/W	VIDEO_IN_FRAME_MODE	Video Frame Input Mode: 0 = Field 1 = Deinterlaced
d	R/W	VIDEO_IN_BUF_MODE	0 = Single Buffering 1 = Double Buffering
e	R/W	VIDEO_IN_BUF	Initial video_in buffer (W): 0 = Buffer 0 1 = Buffer 1 Current buffer being written to by video_in device (R): 0 = Buffer 0 1 = Buffer 1
f	R/W	VIDEO_IN_SYNC_CNTL	Flow controlled synchronous capture for VMC 0 = VMC always send field 1 = VMC wait for overlay to complete
g	R/W	SCALER_FRAME_MODE	Scaler Frame Read Mode: 0 = Full Frame 1-3 = (reserved)
h	R/W	SCALER_BUF_MODE	Scaler Buffer Management 0 = Single Buffering 1 = Double Buffering
i	R/W	SCALER_BUF_NEXT	Next scaler read buffer, effective at **end** of current scaler operation: 0 = Buffer 0 1 = Buffer 1 *Only valid in single buffer mode
j	R	SCALER_BUF_STATUS	Current buffer being read by scaler (R only): 0 = Buffer 0 1 = Buffer 1

Description

This register is used to specify the video capture configuration, including capture fields or frames, video capture using a single or double buffer, and scaler using a single or double buffer. It is also used to specify whether the input video stream can be held off (VMC only).

Usage

To capture a single field of video, set VIDEO_IN_CAP_EVEN or VIDEO_IN_CAP_ODD = Auto, and VIDEO_IN_FRAME_MODE = Field. To capture both fields and show them as deinterlaced, set both VIDEO_IN_CAP_EVEN and VIDEO_IN_CAP_ODD = Auto, and VIDEO_IN_FRAME_MODE = Deinterlaced.

For video capture in a window type of application, set either VIDEO_IN_CAP_EVEN or VIDEO_IN_CAP_ODD = Auto (continuous capture). For video capture to a disk type of application, set either VIDEO_IN_CAP_EVEN or VIDEO_IN_CAP_ODD = Host (capture is triggered by the host).

VIDEO_IN_BUF_MODE and SCALER_BUF_MODE are used to define the video capture and scaler using a single or double buffer. The double buffer mode is used to reduce the tiering effect in video in a window type of application.

When SCALER_BUF_MODE is set to ‘double buffering’, SCALER_BUF_NEXT is **automatically** initially set to the opposite buffer as VIDEO_IN_BUF.

In order for the scaler to flip its buffer in double buffer mode, VIDEO_CAPTURE must also be enabled and be in double buffer mode. The scaler reader will then switch buffers when the writing to the opposite capture buffer is complete.

See Also

CAPTURE_HEIGHT_WIDTH **I/O:–** **MM: 1_11**

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

a	R/W	CAP_HEIGHT	Capture height for input capture stream
b	R/W	CAP_WIDTH	Capture width in terms of pixels for input capture stream

Description

This register specifies the video input capture size, where the capture height is the number of lines and capture width is the number of pixels.

Usage

To capture a single field NTSC signal, set CAP_HEIGHT = 240, and CAP_WIDTH = 320. To capture a full video frame (both even and odd fields in a deinterlaced mode), set CAP_HEIGHT = 480, and CAP_WIDTH = 320

For field mode, the capture height reflects the height of each field being captured. For deinterlaced mode, the capture height reflects the combined height of the interlace even and odd fields. The video input is not guaranteed to capture 'CAP_HEIGHT' lines of data, but cannot exceed the height specified.

See Also

CAPTURE_Y_X on *page 5-21*

	CAPTURE_Y_X																								I/O:–				MM: 1_10			
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	▨	▨	▨	▨	▨	b										▨	▨	▨	▨	▨	▨	a									

a	R/W	CAP_Y	Capture Y line start
b	R/W	CAP_X	Capture X pixel start

Description

This register determines which starting line and pixels in the input video stream to begin writing to the current video input buffer. The coordinates are specified relative to the **active video** received. The size of the video is determined by the CAPTURE_HEIGHT_WIDTH register.

Usage

To capture a full-size NTSC video image, specify CAP_Y = 0, and CAP_X = 0.

See Also

CAPTURE_HEIGHT_WIDTH on *page 5-20*

TRIG_CNTL	I/O:–	MM: 1_15

BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT	g	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	f	e	d	c	b		a	

a	R/W	CAPTURE_TRIG_EVEN	Trigger even video in capture initiate (even host mode): (W) 0 = no action 1 = capture next even frame 2-3 = reserved Even Capture Status: (R) 0 = even capture complete 1 = even capture pending 2 = even capture in progress 3 = reserved
b	R/W	CAPTURE_TRIG_ODD	Trigger odd video in capture initiate (odd host mode): (W) 0 = no action 1 = capture next odd frame 2-3 = reserved Odd Capture Status: (R) 0 = odd capture complete 1 = odd capture pending 2 = odd capture in progress 3 = reserved
c	R	TRIG_EVEN_BUF	Buffer written to by last even host trigger (R)
d	R	TRIG_ODD_BUF	Buffer written to by last odd host trigger (R)
e	R	BUF_RD_STATUS	Overlay Buffer Read Status: (R) 0 = Current scaler buffer not being read 1 = Current scaler buffer being read by overlay
f	R	BUF_WR_STATUS	Overlay Buffer Write Status: (R) 0 = Current capture buffer not being written 1 = Current capture buffer being written
g	R/W	CAPTURE_EN	0 = Disable/Reset Capture (default = 0) 1 = Enable Capture

Description

This register specifies the **Host**-triggered video capture mode and corresponding capture status. This register also contains a CAPTURE_EN bit for enabling/resetting video capture.

Video input can occur simultaneously with the scaler scaling directly to the DAC.

Usage

When host-triggered capture is used to capture video to disk, set CAPTURE_TRIG_EVEN or CAPTURE_TRIG_ODD = 1 to initiate the capture of a single video field or frame. The software should wait for the interrupt to signal the end of capture, then poll the CRTC_INT_CNTL register to determine if the interrupt was generated by video capture. The TRIG_CNTL register is then used to determine which video field has been captured. CAPTURE_EN is a general set/reset bit for video capture, which should be set to 1 to perform any kind of video capture (in a window or host-triggered capture).

When both even and odd fields are being captured, situations may arise where only odd frames or only even frames arrive (e.g., during fast forwards or stills). To compensate for these situations, the Video Frame Synchronizer will "fake" the field information. The following tables indicate the sequence of actions taken.:

Input field:	Even	Odd	Even	Even	Even	Even	Even	Even	Even	Even	Even	Even
Output field:	Even	Odd	Even	**Drop**	Even	Odd	Even	Odd	Even	Odd	Even	Odd

Input field:	Even	Odd	Even	Odd	Odd	Odd	Odd	Odd	Odd	Odd	Odd	Odd
Output field:	Even	Odd	Even	Odd	**Drop**	Even	Odd	Even	Odd	Even	Odd	Even

See Also

CRTC_INT_CNTL on *page 4-26*

	VIDEO_SYNC_TEST																					I/O:–							MM: 1_16			
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	h	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	g	▨	▨	▨	▨	f	e	d	c	▨	▨	▨	▨	▨	▨	b	a

a	R/W	CRTC_OVLSOF	CRTC Overlay Start-of-Field
b	R/W	CRTC_VOVLEN	CRTC Vertical Overlay Enable
c	R/W	VID_SOF	Video Port Start-of-Field
d	R/W	VID_EOF	Video Port End-of-Field
e	R/W	VID_EOL	Video Port End-of-Line
f	R/W	VID_FIELD	Video Port Odd Field Indicator 0 = Even Field 1 = Odd Field
g	R/W	WRRDY	Write Buffer Ready Indicator 0 = FIFO Full 1 = Empty (able to receive data)
h	R/W	SYNC_TEST_EN	0 = Disable Test Mode (default = 0) 1 = Enable Test Mode

Description

This register holds debugging information for the video synchronization control block.

Usage

This register is used for debugging purposes.

Buffer Registers

Buffer registers specify the video data buffer parameters used by video capture and scaling. In the VT, two sets of video buffers can be specified. In single buffer mode, video capture writes to the same video buffer that the scaler reads from. In double buffer mode, it is possible for video capture to write to one buffer while the scaler reads from the alternate buffer.

The video data buffer should be defined outside the display area (off-screen area).

Buffer 0

	BUF0_CAP_ODD_OFFSET																									I/O:–				MM: 1_2B		
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	▨	▨	▨	▨	▨	▨	▨	b																					a		

a	–	(Reserved)	fixed to '0'
b	R/W	BUF0_CAP_ODD_OFFSET	Buffer 0 **QWORD** capture offset for ODD fields (FIELD MODE ONLY)

Description

This register specifies the memory address offset of an odd video field in capture buffer 0.

Usage

This register is used for **field capture mode** only, in which the user wants to capture both EVEN and ODD fields into a single capture buffer 0.

Although the buffer registers are byte-defined, the video capture buffers must be **quadword** aligned.

See Also

CAPTURE_CONFIG on *page 5-18*

	BUF0_OFFSET																						I/O:–						MM: 1_20			
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	▨	▨	▨	▨	▨	▨	▨	b																					a		

a	–	(Reserved)	fixed to ‘0’
b	R/W	BUF0_OFFSET	Buffer 0 **QWORD** offset to planar Y data and RGB modes

Description

This register specifies the memory address offset of video capture buffer 0.

Usage

The buffer 0 offset should specify an area of memory outside of the display memory area.

Although the buffer registers are byte-defined, the base BUF0_OFFSET must be **quadword** aligned.

See Also

BUF0_PITCH on *page 5-27*

BUF0_PITCH **I/O:–** **MM: 1_23**

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

a	R/W	BUF0_PITCH	Buffer 0 pitch in pixels of planar Y data and RGB modes

Description

This register specifies the pitch (the memory address increment between two video lines) of capture buffer 0.

Usage

For packed mode (RGB or YUV422), the buffer pitch is in units of pixels, and in the planar mode, the buffer pitch is in units of planar Y data. The buffer pitch should be equal to or greater than the capture width. For example, if the capture width is set to 320, BUF0_PITCH should be set to 320 or greater.

BUF0_PITCH must cause the next line to begin on a **quadword** boundary (taking into account the pixel depth).

For RGB8888, set the lowest bit to '0' to ensure a quadword-aligned pitch. For all other modes, set the lowest two bits to '0' to ensure a quadword-aligned pitch

See Also

BUF0_OFFSET on *page 5-26*

CAPTURE_HEIGHT_WIDTH on *page 5-20*

Buffer 1

	BUF1_CAP_ODD_OFFSET																							I/O:–					MM: 1_2C			
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT	▨	▨	▨	▨	▨	▨	▨	▨	b																					a		

a	–	(Reserved)	fixed to ‘0’
b	R/W	BUF1_CAP_ODD_OFFSET	Buffer 1 **QWORD** capture offset for ODD fields (FIELD MODE ONLY)

Description

This register specifies the memory address offset of an odd video field in capture buffer 1.

Usage

This register is used for **field capture mode** only, in which the user wants to capture both EVEN and ODD fields into a single capture buffer 1.

Although the buffer registers are byte-defined, the video capture buffers must be **quadword** aligned.

See Also

CAPTURE_CONFIG on *page 5-18*

	BUF1_OFFSET																									**I/O:–**				**MM: 1_26**	
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	▨	▨	▨	▨	▨	▨	▨	b																					a		

a	–	(Reserved)	fixed to ‘0’
b	R/W	BUF1_OFFSET	Buffer 1 **QWORD** offset to planar Y and RGB modes

Description

This register specifies the memory address offset of video capture buffer 1.

Usage

The buffer 1 offset should specify an area of memory outside of the display memory area.

Although the buffer registers are byte-defined, the base BUF1_OFFSET must be **quadword** aligned.

See Also

BUF1_PITCH on *page 5-30*

	BUF1_PITCH																						I/O:–						MM: 1_29		
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	a											

a	R/W	BUF1_PITCH	Buffer 1 pitch in pixels of planar Y data and RGB modes

Description

This register specifies the pitch (the memory address increment between two video lines) of capture buffer 1.

Usage

For packed mode (RGB or YUV422), the buffer pitch is in units of pixels, and in the planar mode, the buffer pitch is in units of planar Y data. The buffer pitch should be equal to or greater than the capture width. For example, if the capture width is set to 320, BUF0_PITCH should be set to 320 or greater.

BUF1_PITCH must cause the next line to begin on a **quadword** boundary (taking into account the pixel depth).

For RGB8888, set the lowest bit to ‘0’ to ensure a quadword-aligned pitch. For all other modes, set the lowest two bits to ‘0’ to ensure a quadword-aligned pitch

See Also

BUF1_OFFSET on *page 5-29*

CAPTURE_HEIGHT_WIDTH on *page 5-20*

VMC Registers

The VMC registers are used to configure the VMC controller in the VT, and to hold the sending/receiving of VMC commands and data on the VMC bus.

Configuration and Status

VMC_CONFIG **I/O:–** **MM: 1_18**

BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT	i	▨	▨	▨	▨	▨	h		▨	▨	g	f	▨	e	d	▨	▨	▨	▨	▨	▨	c	▨	▨	b				a			

a	R/W	STRM_INPUT_ID	Input Stream ID
b	R/W	STRM_OUTPUT_ID	Output Stream ID
c	R/W	HWCUR_LEFT	0 = Disable Fix 1 = Enable HW Cursor Left Fix
d	R/W	VMC_CLK_EN	0 = Disable internal clock source (default = 0) 1 = Enable internal clock source
e	R/W	VMC_CLK_SRC	0 = External Clk (default = 0) 1 = Internal Clk
f	R/W	VMC_BYPASS_CONFIG	0 = Configuration cycle enabled 1 = Bypass configuration cycle
g	R/W	VMC_CLK_OUT_EN	0 = Disable VMC clock out DCLK pin(default=0) 1 = Enable VMC clock out DCLK pin
h	R/W	VMC_STATUS_MODE	VMC Status register readback mode: 0 = Normal 1 = Schedule 2 = Debug mode 1 3 = Debug mode 2
i	R/W	VMC_EN	0 = Reset VMC (default = 0) 1 = Enable VMC

Description

This is a general configuration register for the VMC port.

Usage

The VMC_EN bit in this register is the general set/reset control for the VMC port. To enable the VMC function, set VMC_EN = 1. To use the VMC port for video capture, also set CAPTURE_EN@TRIG_CNTL = 1.

See Also

TRIG_CNTL on *page 5-22*

VIDEO_DEVICE@VIDEO_CONFIG on *page 5-14*

VMC_STATUS (DEBUG1/2 MODES) **I/O:–** **MM: 1_19**

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

a	R/W	VMC_DEBUG_STATUS	Internal status of VMC for debugging

VMC_STATUS (DEFAULT MODE) **I/O:–** **MM: 1_19**

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

a	R	EX_DID	Exception code Device ID (ID of device that generated exception)
b	R	EX_EC	Exception code of last exception
c	R/W	HOST_TIMEOUT	Host write status indicator: 0 = Normal (write accepted) 1 = Timeout (write not sent over VMC bus)

VMC_STATUS (SCHEDULE MODE) **I/O:–** **MM: 1_19**

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

a	R	VMC_DID	Assigned device ID of VT/3D RAGE VMC
b	R	VMC_NDID	Next DID scheduled (internal - default = 0)
c	R	VMC_GTR	Grant time register (internal - default = 16)

Description

This register holds debugging information for the VMC controller and the video port.

Usage

This register is used to read back debugging information from the VMC controller and the video port, in one of four modes. Which information is read back is specified in VMC_STATUS_MODE@VMC_CONFIG.

See Also

VMC_CMD on *page 5-35*

VMC_CONFIG on *page 5-31*

VMC Command

	VMC_ARG0																								I/O:–							MM: 1_1B
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT	a																															

a	R/W	ARG0_DATA	VMC command argument 0

Description

This register specifies VMC command argument 0.

Usage

Most VMC commands can have one or more arguments associated with the command. Write the argument to this register if the VMC command needs one argument. The VMC command is written to register VMC_CMD. If a VMC ‘snoop’ command is specified in the VMC_CMD register, argument 0 of the snoop command is written to this register. The user software can read this register to get the value of the argument.

See Also

VMC_CMD on *page 5-35*

VMC_ARG1 on *page 5-34*

	VMC_ARG1																								I/O:–				MM: 1_1C			
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT	a																															

a	R/W	ARG1_DATA	VMC command argument 1

Description

This register specifies VMC command argument 1.

Usage

Most VMC commands can have one or more arguments associated with the command. If the VMC command has two arguments, the second argument is written to this register.

See Also

VMC_CMD on *page 5-35*

VMC_ARG0 on *page 5-33*

VMC_CMD I/O:– MM: 1_1A

BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT	g	▨	▨	▨	f		e		d								c	▨	▨	▨	▨	▨	b		a							

a	R/W	VMC_CMD	VMC command
b	R/W	VMC_CMD_NARGS	Number of command arguments: 0 = 0 arguments 1 = 1 argument 2 = 2 arguments 3-4 = (reserved)
c	R/W	VMC_CMD_TRIG	Command Trigger: (W) (default = 0) 0 = reset 1 = trigger current command to VMC bus Command Status: (R) 0 = ready 1 = command busy
d	R/W	VMC_SNOOP_CMD	VMC snoop command
e	R	VMC_SNOOP_NARGS	Number of snoop arguments retrieved: (Read Only) 0 = 0 arguments 1 = 1 argument 2 = 2 arguments 3-4 = (reserved)
f	R/W	VMC_SNOOP_MASK	Snoop command compare mask: ‘00’ = capture only VMC bus cmds that match all 8 snoop cmd bits ‘01’ = capture only VMC bus cmds that match upper 4 snoop cmd bits ‘10’ = capture only VMC bus cmds that match lower 4 snoop cmd bits ‘11’ = accept and capture any VMC bus command
g	R/W	VMC_SNOOP_TRIG	Snoop Trigger: (W) (default = 0) 0 = reset snoop status 1 = trigger for snoop command on VMC bus Snoop Status: (R) 0 = snoop command captured 1 = snoop pending

Description

This register specifies the VMC command that will be sent on the VMC bus, or the ‘snoop’ command that will be placed on the VMC bus.

Usage

Use this register to write out the command to be sent on the VMC bus and to specify the number of arguments associated with the command. If the number of arguments is greater than zero, write the specific arguments to VMC_ARG0 (first argument) and, if required, VMC_ARG1 (second argument). To actually send the command out on the VMC bus, set VMC_CMD_TRIG = 1. To verify that the command was sent, poll VMC_CMD_TRIG.

The VMC_SNOOP_CMD is used primarily for debugging purposes. To snoop a command on the VMC bus, write the command to VMC_SNOOP_CMD, and write the number of arguments to VMC_SNOOP_NARGS. To trigger the snooping of a command, set VMC_SNOOP_TRIG = 1.

See Also

VMC_ARG0 on *page 5-33*

VMC_ARG1 on *page 5-34*

VMC_SNOOP_ARG0 on *page 5-37*

VMC_SNOOP_ARG1 on *page 5-38*

		VMC_SNOOP_ARG0																								I/O:–				MM: 1_1D	
BITS		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT		a																															

a	R	SNOOP_ARG0_DATA	VMC snoop command argument 0 read

Description

This read-only register is used to hold argument 0 of the snoop command.

Usage

If a snoop command is specified in the VMC_CMD register, argument 0 of the snoop command is written to this register by the VMC controller. Poll this register to get the value of the argument.

See Also

VMC_CMD on *page 5-35*

		VMC_SNOOP_ARG1	I/O:– MM: 1_1E
BITS		31–0	
VT		a	
a	R	SNOOP_ARG1_DATA	VMC snoop command argument 1 read

Description

This read-only register is used to hold argument 1 of the snoop command.

Usage

If a snoop command is specified in the VMC_CMD register, argument 1 of the snoop command is written to this register by the VMC controller. Poll this register to get the value of the argument.

See Also

VMC_CMD on *page 5-35*

Stream Data

VMC_STRM_DATA[i] **I/O:–** **MM: 1_30 – 1_3F**

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

a	R/W	VMC_DATA[i]	VMC stream mode write data

Description

When the VMC port is configured as a video output device, VMC_STRM_DATA0 to VMC_STRM_DATAF are the VMC write data registers used to send data through the VMC port to other chips.

Usage

This register set can be used to send unformatted stream data out of the VMC port, in streams of up to 16 DWORDs. The data registers are in 16 consecutively mapped addresses to allow for block data moves, 16 DWORDs deep. The data will typically be MPEG compressed data.

See Also

VIDEO_CONFIG on *page 5-14*

Chapter 6
3D RAGE Scaler and 3D Operations Registers

This chapter contains descriptions of registers that are used to configure scaler and 3D pipeline functions in the 3D RAGE.

Scaler Pipe Registers

Both horizontal and vertical scaling are supported in the **3D RAGE**. All of the Scaler registers,except SCALE_3D_CNTL and SCALE_VACC, are **aliased** with certain 3D and Texture Mapping registers (aliases are shown in brackets in the table headers). The data to be scaled is deposited into two new FIFOs: the Line0 and Line1 FIFOs. For Texture Mapping, these FIFOs are reused to hold texel data.

For both horizontal and vertical scaling, the color interpolator DDAs are reused to function as a counter/accumulator for the scaler source memory address generator. Scaled pixel data is processed through a 2-tap 5-bit coefficient fixed linear filter blender.

Horizontal and vertical scaling are done in a single pass. For each destination line, two lines of source data are read, and then color expanded to 24 bpp for vertical blending. The resultant data is then blended horizontally, converted to RGB if necessary, and then packed to the destination pixel type and dithered. Additionally a second pass through the blender is possible if Alpha blending is desired.

The DDA X and Y accumulators have a precision of 8.12 unsigned. The blend coefficients are determined by the high 5 fractional bits of the DDA coordinate registers.

Separate scaling parameters for the U,V parameters of Y,U,V subsampled data are available to allow freedom in selecting the sampling alignment.

	SCALE_Y_OFF (TEX_0_OFF)																								I/O:–				MM: 0_70			
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
3D RAGE	▨	▨	▨	▨	▨	▨	▨	▨	▨	a																						

a	R/W	SCALE_Y_OFFSET	Offset **in bytes** to RGB, or packed YUV source data for the scaler pipe. Bits 2-0 are required to be programmed as '0'.

SCALE_WIDTH (TEX_7_OFF) I/O:– MM: 0_77

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

a	R/W	SCALE_WIDTH	Width in pixels of the scaler source data. For YUV pixel types, the Y value should be programmed. The U and V values will be derived from the pixel type.

SCALE_HEIGHT (TEX_8_OFF) I/O:– MM: 0_78

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

a	R/W	SCALE_HEIGHT	Height in pixels of the scaler source data. For YUV pixel types, the Y value should be programmed. The U and V values will be derived from the pixel type.

SCALE_Y_PITCH (S_Y_INC) I/O:– MM: 0_7B

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

a	R/W	SCALE_Y_PITCH	Pitch in pixels of the scaler source data for RGB and packed modes. The pitch is required to be programmed so that all source lines start on QUAD word boundaries.

SCALE_X_INC (RED_X_INC) I/O:– MM: 0_7C

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

a	R/W	SCALE_X_INC	X accumulator increment, 12 bits fractional, 4 bits unsigned integer. For packed or planar YUV pixels, this applies only to the Y values.

SCALE_Y_INC (GREEN_X_INC) I/O:– MM: 0_7D

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

a	R/W	SCALE_Y_INC	Y accumulator increment, 12 bits fractional, 4 bits unsigned integer.

SCALE_VACC **I/O:–** **MM: 0_7E**

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

a	R/W	SCALE_VACC	Starting value for vertical accumulator. May be non-zero for fine control. Sign, 12 bits fractional, 4 bits integer. Non-zero values have the effect of shifting the input pixel relative to the output pixel. Note that this register is incremented during the scale operation, and so must be reloaded for every scale.

	SCALE_3D_CNTL																										**I/O:–**				**MM: 0_7F**	
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
3D RAGE	x	w	v	u	t		s	r	q		p			o			n	m	l	k		j	i	h	g		f	e	d	c	b	a

a	R/W	SCALE_PIX_EXPAND	Pixel expansion algorithm 0 = zero extend 1 = dynamic range correct
b	R/W	SCALE_DITHER	Dither algorithm 0 = X error diffusion 1 = Two dimensional dither with table lookup.
c	R/W	DITHER_EN	Enable dithering of Scaled / 3D image 0 = disable dithering 1 = enable dithering
d	R/W	DITHER_INIT	Enable resetting Dither Error register (used for X error diffusion dither) back to 0 at the start of each line. 0 = use current contents of error register at the start of the line 1 = reset to 0 at the start of the line
e	R/W	ROUND_EN	Enable rounding instead of Dithering to reduce from 24 bit pixels. If both Dithering and Rounding are disabled, truncation will be used. 0 = Rounding disabled 1 = Rounding enabled
f	R/W	TEX_CACHE_DIS	If set, the Texel Cache is disabled. This speeds up Texture Mapping for those modes that are limited by memory latency, rather than bandwidth. These situations are likely to be NO BLEND modes with low resolution screens.
g	R/W	SCALE_3D_FCN	The SCALE_3D_FCN encodes the operation(s) to be performed by the 3D / Scaling pipe. 0 = No operation 1 = Scaling 2 = Texture Mapping 3 = Shading
h	R/W	SCALE_PIX_REP	Replicate pixels rather than linear blend 0 = Blend pixels during scale 1 = Replicate pixels during scale (stretchblt)
i	R/W	NEAREST_TEX_VIS	During Texture Map filtering, use the nearest texel for the Texture Visibility Test. Otherwise use test all texels involved in the blend. 0 = Test all texels involved in the blend, if Texel Visibility is enabled. 1 = Test nearest texel only, if Texel Visibility is enabled. During scaling, this bit selects whether Chroma Keying is done on all 4 source pixels.
j	R/W	APPLE_YUV_MODE	Denotes that YUV data in the Scale/3D pipe will be interpreted as signed (UV will have an offset of 0, rather than 128)
k	R/W	ALPHA_FOG_EN	Allow Alpha blending or Fog 0 = Alpha blending and Fog disabled 1 = Alpha blending enabled; Fog disabled 2 = Fog enabled (use DP_FRGD_CLR register in place of Destination FIFO); Alpha blending disabled 3 = (reserved)

(continued on next page)

	SCALE_3D_CNTL																								I/O:–				MM: 0_7F		
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
3D RAGE	x	w	v	u	t		s	r	q		p			o			n	m	l	k		j	i	h	g		f	e	d	c	b	a

l	R/W	COLOR_OVERRIDE	During texture mapping, when TEX_MAP_AEN is set, this bit will force the color interpolators to be used in place of the texel color. This operation is useful in multipass operations. Even if this bit is set, the texel will still be used for Alpha information. 0 = Used Texel Color for operation 1 = Use Interpolator Color for operation
m	R/W	RED_DITHER_MAX	Allows the range of the RED to be limited so as to avoid running into reserved areas of the DAC Lookup Table. This range is only implemented when dithering is occuring. 0 = Allow RED to occupy its full range and use all of the LUT. 1 = Used for RGB8 mode. Scales RED to 2.75 bits so that the overall color has a range of 0 – 223.
n	R/W	SIGNED_DST_CLAMP	When alpha blending, assume that the destination is a signed value and clamp the results accordingly. This mode is used for MPEG motion compensation.
o	R/W	ALPHA_BLND_SRC	Alpha blending functions (SRC) 0 = Blend factor is (0, 0, 0) 1 = Blend factor is (1, 1, 1) 2 = Blend factor is (RD, GD, BD) 3 = Blend factor is (1-RD, 1-GD, 1-BD). 4 = Blend factor is (As, As, As). 5 = Blend factor is (1-As, 1-As, 1-As).
p	R/W	ALPHA_BLND_DST	Alpha blending functions (DST) 0 = Blend factor is (0, 0, 0) 1 = Blend factor is (1, 1, 1) 2 = Blend factor is (Rs, Gs, Bs 3 = Blend factor is (1-Rs, 1-Gs, 1-Bs). 4 = Blend factor is (As, As, As). 5 = Blend factor is (1-As, 1-As, 1-As).
q	R/W	TEX_LIGHT_FCN	Specifies the function used to light texels: 0 = No Lighting. Output color is: texel color 1 = Modulate. Output color is: texel color * interpolator color 2 = Alpha Decal. This mode is only valid for texture maps with alpha (TEX_MAP_AEN = 1). Output color is : (texel color * texel alpha) + (interpolator color * (1 – texel alpha)). The texel alpha used will reflect the result of the TEX_AMASK_AEN bit. If alpha blending (with the destination) is enabled, the alpha interpolator (rather than the texel alpha) will be used as the alpha source. Note that this mode may be used even when alpha blending (with the destination) is turned off. 3= (reserved)
r	R/W	MIP_MAP_DISABLE	Disables transitioning between texture maps during texture mapping 0 = Mip mapping is enabled 1 = Mip mapping is disabled; only one (the largest present) texture map will be used.

(continued on next page)

	SCALE_3D_CNTL																										I/O:–				MM: 0_7F	
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
3D RAGE	x	w	v	u	t		s	r	q		p			o			n	m	l	k		j	i	h	g		f	e	d	c	b	a

s	R/W	BILINEAR_TEX_EN	When a larger texture map than the largest available is desired (magnification), enable a 2x2 blend of a the largest available texture map to create a source pixel. This mode is valid whether or not MIP MAPPING is enabled. If MIP MAPPING is disabled a 2x2 blend will be done using the single map that is present when a larger map is desired. 0 = bilinear interpolation is disabled; during magnification use PICK NEAREST within the largest map. Note that if the minification mode is either of the 2x2 blending modes, then NO pixels will be drawn during magnification. This allows a multipass algorithm to be used for this case. 1 = bilinear interpolation is enabled; during magnification use 2x2 blend within the largest map.
t	R/W	TEX_BLEND_FCN	Specifies the texel blending between texels during minification. 0 = No blending (select nearest map). If MIP MAPPING is disabled, the single map that is present will always be chosen. 1= Blend between two closest maps. This mode only makes sense if MIP MAPPING is enabled 2 = 2x2 blend within the nearest map. This mode may be used regardless of whether MIP MAPPING is disabled. If MIP MAPPING is disabled, a 2x2 blend will be done using the single map that is present. 3 = 2x2 blend within next to nearest map. If alpha blending is enabled, the alpha value used will be based on distance to the nearest map. Used for multi-pass trilinear blending. This mode only makes sense if MIP MAPPING is enabled.
u	R/W	TEX_AMASK_AEN	Enables use of the low order bit of texture map alpha as a mask value. Other bits of texture map alpha will be ignored. This bit is only used when TEX_MAP_AEN is set. 0 = Use complete alpha value from texture map 1 = Use only the LSB of the alpha value from the texture map
v	R/W	TEX_AMASK_MODE	When TEX_AMASK_AEN and TEX_MAP AEN are both 1, allows the alpha mask to take on different meanings. 0 = Conventional 3D-DDI mode. An alpha mask value of 0 indicates that the texel will not be drawn. An alpha mask value of 1 indicates the texel will be drawn using the alpha interpolator value. 1 = Blended Edge Mode. An alpha mask value of 0 indicates that the texel will be drawn using the alpha interpolator value. An alpha mask value of 1 indicates the texel will be drawn using an alpha value of 0xff.
w	R/W	TEX_MAP_AEN	When asserted, indicates that the texture maps (or scale sources) include alpha values. For scale sources, this mode is supported only in conjunction with TEX_AMASK_AEN. These values are then used as the source alphas. This mode is valid for 32 bit (8 bit alpha + 24 bit RGB) texels, 16 bit (1 bit alpha + 15 bit RGB or 4 bit alpha and 12 bit RGB) texels. 0 = No alpha in the texture map 1 = Alphas are in the texture map
x	R/W	SRC_3D_SEL	The source pixel for the 3D pipeline may come either from the interpolators or the host FIFO. No blending will be done on texels that are sourced from the Host. 0 = Interpolators or Texture/Line FIFOs 1 = Host FIFO

SCALE_HACC (RED_START) I/O:– MM: 0_F2

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

a	R/W	SCALE_HACC	Starting value for horizontal accumulator. May be non-zero for fine control. Sign,12 bits fractional, 4 bits integer. Non-zero values have the effect of shifting the input pixel relative to the output pixel. For sub sampled pixels, this shift occurs only on the Y values.

SCALE_XUV_INC (BLUE_X_INC) I/O:– MM: 0_F6

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

a	R/W	SCALE_XUV_INC	X accumulator increment, 12 bits fractional, 4 bits unsigned integer. This only applies to UV values in sub sampled YUV pixels.

SCALE_UV_HACC (BLUE_START) I/O:– MM: 0_F8

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

a	R/W	SCALE_UV_HACC	Starting value for horizontal accumulator for UV values sub sampled YUV pixels. May be non-zero for fine control. Sign,12 bits fractional, 4 bits integer. Non-zero values have the effect of shifting the UV values relative to the Y values.

Note that the Accumulator registers should only be written to when the SCALE_3D_FCN@ SCALE_3D_CNTL bits are set to a non-zero value.

3D Operations

Texture Mapping

TEX_[0-10]_OFFSET **I/O:–** **MM: 0_70 – 7A**

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

a	R/W	TEX_0_OFFSET TEX_1_OFFSET TEX_2_OFFSET TEX_3_OFFSET TEX_4_OFFSET TEX_5_OFFSET TEX_6_OFFSET TEX_7_OFFSET TEX_8_OFFSET TEX_9_OFFSET TEX_10_OFFSET	Byte pointer to 1x1 texture map Byte pointer to 2x2 texture map Byte pointer to 4x4 texture map Byte pointer to 8x8 texture map Byte pointer to 16x16 texture map Byte pointer to 32x32 texture map Byte pointer to 64x64 texture map Byte pointer to 128x128 texture map Byte pointer to 256x256 texture map Byte pointer to 512x512 texture map Byte pointer to 1024x1024 texture map

S_X_INC2 **I/O:–** **MM: 0_D0**

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

a	R/W	S_X_INC2	Second derivative register for texture mapping. Change of S_X_INC when stepping in X along the leading edge of the trapezoid. In S.10.16 format

S_Y_INC2 **I/O:–** **MM: 0_D1**

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

a	R/W	S_Y_INC2	Second derivative register for texture mapping. Change of S_Y_INC when stepping in Y along the leading edge of the trapezoid. In S.10.16 format

S_XY_INC2 I/O:– MM: 0_D2

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

a	R/W	S_XY_INC2	Second derivative register for texture mapping. Change of S_X_INC when stepping in Y along the leading edge of the trapezoid. In S.10.16 format

S_XINC_START I/O:– MM: 0_D3

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

a	R/W	S_XINC_START	Value of S_X_INC at beginning of trapezoid span. In S.11.16 format

S_Y_INC (SCALE_Y_PITCH) I/O:– MM: 0_D4

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

a	R/W	S_Y_INC	Change of S address when stepping in Y along the leading edge of the trapezoid. In S.11.16 format

S_START I/O:– MM: 0_D5

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

a	R/W	S_START	Initial value of S coordinate address. In 10.11 format.

T_X_INC2 I/O:– MM: 0_D6

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

a	R/W	T_X_INC2	Second derivative register for texture mapping. Change of T_X_INC when stepping in X along the leading edge of the trapezoid. In S.10.16 format

T_Y_INC2 I/O:– MM: 0_D7

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

a	R/W	T_Y_INC2	Second derivative register for texture mapping. Change of T_Y_INC when stepping in Y along the leading edge of the trapezoid. In S.10.16 format

T_XY_INC2 I/O:– MM: 0_D8

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

a	R/W	T_XY_INC2	Second derivative register for texture mapping. Change of T_X_INC when stepping in Y along the leading edge of the trapezoid. In S.10.16 format

T_XINC_START I/O:– MM: 0_D9

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

a	R/W	T_XINC_START	Value of T_X_INC at beginning of trapezoid span. In S.11.16 format

T_Y_INC I/O:– MM: 0_DA

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

a	R/W	T_Y_INC	Change of T address when stepping in Y along the leading edge of the trapezoid. In S.11.16 format

T_START I/O:– MM: 0_DB

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

a	R/W	T_START	Initial value of T coordinate address. In 10.11 format.

	TEX_SIZE_PITCH																								**I/O:–**				**MM: 0_DC**			
BITS	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
3D RAGE	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	c				b				a			

a	R/W	TEX_PITCH	The log (base 2) pitch (in pixels) of the largest texture map (i.e. 0xA for 1K, 0 for 1). If a non-square map is present, this field should contain the pitch of the actual map that is present.
b	R/W	TEX_SIZE	Size of the largest texture map (in pixels), from 0xA for 1Kx1K, to 0 for 1x1. If a non-square map is present, this field should be programed as if the map was the size of its largest dimension.
c	R/W	TEX_HEIGHT	The log (base 2) height (in pixels) of the largest texture map (i.e. 0xA for 1K, 0 for 1). If only a non-square map is present, this field should contain the height of the actual map that is present.

Color, Z and Alpha Interpolation

RED_X_INC (SCALE_X_INC) I/O:– MM: 0_F0

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

a	R/W	RED_X_INC	Interpolation value for steps in X in the DST_X _DIR direction along the leading edge of the trapezoid. In S.8.12 format

RED_Y_INC I/O:– MM: 0_F1

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

a	R/W	RED_Y_INC	Interpolation value for steps in Y in the DST_Y_DIR direction along the leading edge of the trapezoid. In S.8.12 format

RED_START (SCALE_HACC) I/O:– MM: 0_F2

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

a	R/W	RED_START	Initial value of red, at the beginning of trapezoid. In S.8.12 format

GREEN_X_INC (SCALE_Y_INC) I/O:– MM: 0_F3

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

a	R/W	GREEN_X_INC	Interpolation value for steps in X in the DST_X_DIR direction along the leading edge of the trapezoid. In S.8.12 format

GREEN_Y_INC I/O:– MM: 0_F4

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

a	R/W	GREEN_Y_INC	Interpolation value for steps in Y in the DST_Y_DIR direction along the leading edge of the trapezoid. In S.8.12 format

GREEN_START I/O:– MM: 0_F5

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

a	R/W	GREEN_START	Initial value of green, at the beginning of trapezoid. In S.8.12 format

BLUE_X_INC (SCALE_XUV_INC) I/O:– MM: 0_F6

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

a	R/W	BLUE_X_INC	Interpolation value for steps in X in the DST_X_DIR direction along the leading edge of the trapezoid. In S.8.12 format

BLUE_Y_INC I/O:– MM: 0_F7

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

a	R/W	BLUE_Y_INC	Interpolation value for steps in Y in the DST_Y_DIR direction along the leading edge of the trapezoid. In S.8.12 format

BLUE_START (SCALE_UV_HACC) I/O:– MM: 0_F8

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

a	R/W	BLUE_START	Initial value of blue, at the beginning of trapezoid. In S.8.12 format

Z_X_INC I/O:– MM: 0_F9

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

a	R/W	Z_X_INC	Interpolation value for steps in X in the DST_X_DIR direction along the leading edge of the trapezoid. In S.16.12 format

Z_Y_INC I/O:– MM: 0_FA

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

a	R/W	Z_Y_INC	Interpolation value for steps in Y in the DST_Y_DIR direction along the leading edge of the trapezoid. In S.16.12 format

Z_START I/O:– MM: 0_FB

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

a	R/W	Z_START	Initial value of z, at the beginning of trapezoid. In S.16.12 format

ALPHA_X_INC (FOG_X_INC) I/O:– MM: 0_FC

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

a	R/W	ALPHA_X_INC	Interpolation value for steps in X in the DST_X_DIR direction along the leading edge of the trapezoid. In S. 8.12 format

ALPHA_Y_INC (FOG_Y_INC) I/O:– MM: 0_FD

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

a	R/W	ALPHA_Y_INC	Interpolation value for steps in Y in the DST_Y_DIR direction along the leading edge of the trapezoid. In S.8.12 format

ALPHA_START (FOG_START) I/O:– MM: 0_FE

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

a	R/W	ALPHA_START	Initial value of alpha, at the beginning of trapezoid. In S.8.12 format

Chapter 7
PCI Configuration Registers

This chapter contains descriptions of registers that are used for the PCI host bus implementation. For *mach64s* which support multiple host bus types, the host bus definition during system reset will cause only those registers for the current bus type to appear and be used.

PCI Configuration Space

The *mach64*VT/3D RAGE have been optimized to support the Intel Peripheral Component Interconnect (PCI) local bus, and implement the PCI Configuration Space registers. Please refer to the *PCI Local Bus Specification* for detailed descriptions of these registers. Brief descriptions of the *mach64*VT/3D RAGE implementations follow.

	VENDOR ID													PCI Con: 1:0h		
BITS	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	a															

a	R	Power-up default=1002h. This is ATI's assigned PCI vendor ID.

	DEVICE ID													PCI Con: 3:2h		
BITS	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	a															

a	R	Power-up default=5654h (ASCII characters 'VT'), or 4754h (ASCII characters 'GT')

	COMMAND													PCI Con: 5:4h		
BITS	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	▨	e	d	▨	▨	▨	▨	▨	▨	▨	▨	▨	c	b	a

a	R/W	I/O Access Enable. Default to 0, disabled.
b	R/W	Memory Access Enable. Default to 0, disabled.
c	R	Bus Master Enable. Always 0, disabled.
d	R	Received Target Abort. Always 0, inactive.
e	R	Received Master Abort. Always 0, inactive.

	ASIC ID							PCI Con: 08h
BITS	7	6	5	4	3	2	1	0
VT	c		b			a		

a	R	Major ASIC version number (A=0)
b	R	ASIC foundry ID (000=SGS, 001=NEC, 010=KSC)
c	R	Minor ASIC revision number

The ASIC ID also appears in the CONFIG_CHIP_ID register (see *page 4-17*). The following are the ASIC IDs used to date:

ASIC ID	ASIC description
08h	NEC VT-A3
48h	NEC VT-A4
40h	SGS VT-A4

	CACHE LINE SIZE							PCI Con: 0Ch
BITS	7	6	5	4	3	2	1	0
VT	a							

a	R	Power-up default=00h.

	LATENCY TIMER							PCI Con: 0Dh
BITS	7	6	5	4	3	2	1	0
VT	a							

a	R	Power-up default=00h.

	HEADER TYPE							PCI Con: 0Eh
BITS	7	6	5	4	3	2	1	0
VT	a							

a	R	Power-up default=00h.

BIST PCI Con: 0Fh

BITS	7	6	5	4	3	2	1	0
VT	a							

a	R	Power-up default=00h, not used.

MEMORY APERTURE BASE ADDRESS PCI Con: 13:10h

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

a	R	Default to 000000h.
b	R/W	Default to 00h.

BLOCK DECODED I/O BASE ADDRESS PCI Con: 17:14h

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

a	R	Always 01h.
b	R/W	Default to 000000h.

ADAPTER ID PCI Con: 2F:2Ch

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

a	R/W	Default to 00000000h.

BIOS ROM PCI Con: 33:30h

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

a	R/W	BIOS ROM Enable. Default to 0h.
b	R	Reserved. Always 00h.
c	R	Reserved. Always 00h.
d	R/W	BIOS ROM Base Address. Default to 0000h.

	INTERRUPT LINE						PCI Con: 3Ch	
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	a							

a	R/W	Power-up default=00h.

	INTERRUPT PIN						PCI Con: 3Dh	
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	a							

a	R	Power-up default=01h (00h - no default - if interrupt is disabled by strap).

	USER-DEFINED CONFIGURATION						PCI Con: 40h	
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	▨	▨	▨	c	b	a	

a	R/W	Sparsely decoded I/O base address. 00 - I/O base = 2ECh 01 - I/O base = 1CCh 10 - I/O base = 1C8h
b	R/W	Select I/O decoding method. If I/O decode selection strap is set to sparse, then this bit has no effect. If I/O decode strap is set to block, then this bit defaults to 1 (block I/O) and can be changed to 0 (sparse I/O).
c	R/W	Disable decoding of GENENA VGA register at I/O address 46E8h. 0 = decode 46E8h 1 = disable decode of 46E8h

Chapter 8
VGA Programming Overview

Introduction

The VGA portion of the *mach64* controller accelerator provides the following features. Programmers who wish to program directly to the hardware registers can optimize VGA graphics performance.

- Low cost, high performance, single chip graphics solution
- Suitable for board- or system-level implementations
- 100% register level hardware compatible with the IBM VGA display adapter
- I/O Bus Type: ISA (8- and 16-bit)
- Local Buses: 486, 386DX, 386SX, and PCI
- Memory Types: *VRAM* 256Kx4, 256Kx16; *DRAM* 256Kx4, 256Kx16
- Memory Sizes: 1M, 2M, 4M, and 8M
- Colors/Resolutions:
  4 and 8 bpp (Bits Per Pixel) to 1280x1024
  16 bpp to 1280x1024
  24 bpp to 1280x1024
  16 bpp to 1600x1200
- High speed, point-to-point line draw; coprocessor supports up to 32 bpp modes
- Supports memory-mapped registers
- Supports overscan
- Hardware cursor: 64x64x2
- 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
- 0.7 micron CMOS VLSI technology

The *mach64* is a VLSI graphics controller chip consisting of a 64-bit GUI 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.

VGA Controller

The built-in VGA controller can be disabled to allow the accelerator to co-exist with an alternate external VGA.

Configurable Memory Aperture

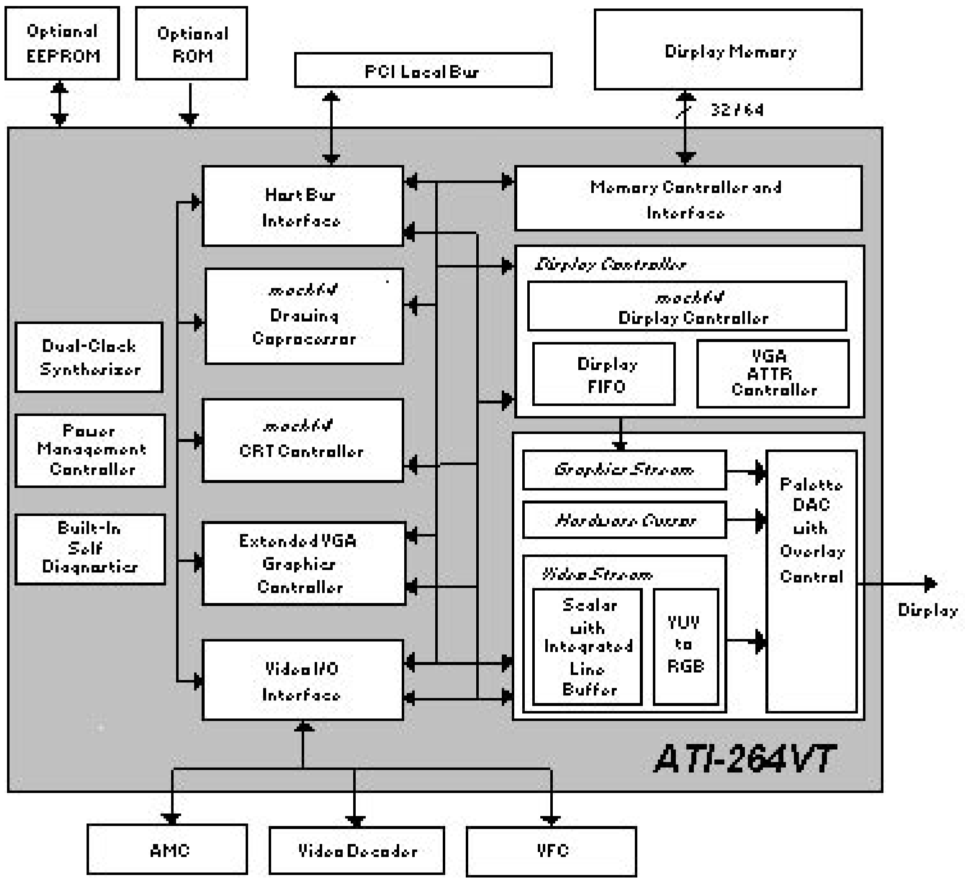
A configurable linear memory aperture is available for all modes, including VGA. In real mode, it may be configured as either a 64K aperture at A000h or two 32K apertures at A000h and A800h, depending on the graphics mode (as will be described later). In protected mode, it may be configured to 1M-paged or 8M-linear on any 8M boundry. The aperture is primarily used for increasing throughput on host-to-screen and screen-to-host data transfers.

In *mach64GX* and *mach64CX* chips, screen memory can be shared between the VGA and the GUI engine. *mach64*-aware applications may reconfigure the memory boundary (register) dynamically to give more or less memory to the VGA or the accelerator. In *mach64CT* and *mach64ET* chips, there is no memory boundary.

VBE BIOS

The VESA BIOS Extension (VBE) specification is intended to standardize the software access to graphics display controllers that support resolutions, color depths and frame buffer organizations beyond the VGA hardware standard. For detail VBE specification, please contact VESA office.

In *mach64* product, the VBE services are implemented in video BIOS. It works on all 80x86 platforms, in real and protected modes. Application programmers and system developers are encouraged to use VBE services to set up and configure the hardware. To further improve performance of an extended mode, one can use GUI functions to perform drawing. For detail programming technique in *mach64*, please refer to the *mach64 Programmer's Guide*.



mach64VT Drawing Coprocessor

This page intentionally left blank.

Chapter 9
VGA Controller

Overview

The VGA Controller registers are completely hardware compatible with the registers of the IBM Video Graphics Array (VGA) adapter. In addition, the VGA Controller provides a 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).

The following VGA functional blocks are integrated within the VGA portion of the *mach64*:

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

IBM Compatible Modes

- **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

- **Super VGA Graphics Modes**
  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)

VGA Memory Organization

The segment base address of video display buffers in the *mach64* 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.

Standard	Size	Memory 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)

Memory organization affects how the video data of each supported display mode is written or read:

- A/N mode is alphanumeric (text).
- APA mode is All-Planes-Addressable (graphics).
- Modes 0 to 13 are 100% compatible with the modes available in the various IBM display adapters.

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 that correspond to the maps. Mode 13 uses a type of page memory access called **packed pixel format**.

Memory Maps

In display mode 0, the buffer size is 2K 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. 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 fourth byte is for attribute, and so on.

When a display mode requires more than 64K of data to paint a full screen, as in display mode 62, the display buffer will map to multiple pages.

Functional Blocks

The VGA Controller has five major functional blocks:

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

These blocks are illustrated in the diagram below, including the control and data paths. Also shown are the connections to the video memory and video DAC. Data and addresses 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 in the following sections.

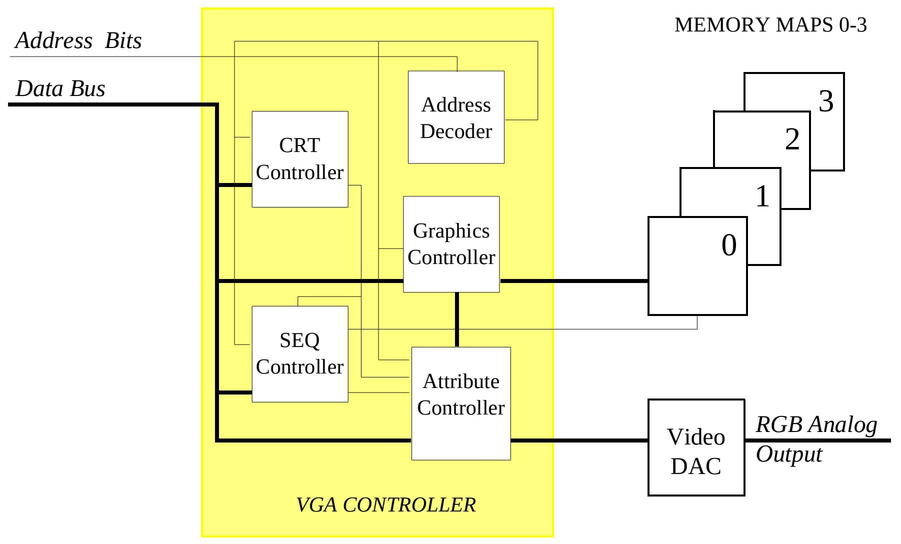


Figure 9-1. VGA Functional Block Diagram

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 starting address of video memory is programmable as VGA 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 (CRTC) 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. CRT controller registers are user programmable for controlling 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-bit 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 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 palette 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 logic for blinking, underline, and horizontal pixel panning.

VGA Display Modes

The *mach64* chip supports alphanumeric or graphics display modes.

- Modes 0 to 6 are emulations of VGA 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 ATI68800 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^+.

The standard and extended modes are fully described in the next section. 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, the 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 the color, intensity, etc. The four most significant bits (MSB) of the attribute byte define the background and 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

*Note: The mode's default value is set by the BIOS. SEQ03[x,x] refers to bits xx of the SEQ03 register in the VGA graphics controller.

Bit 7 **Bit 0**

B	R	G	B	I	R	G	B

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 2000 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 the following tables for description of each supported mode.

Modes 0⁺ and 1⁺			
Standard	**Box Size**	**Char x Row**	**Colors**
VGA	9x16	40x25	16/256K

Modes 0 and 1			
Standard	**Box Size**	**Char x Row**	**Colors**
CGA	8x8	40x25	16

Modes 2⁺ and 3⁺			
Standard	**Box Size**	**Char x Row**	**Colors**
VGA	9x16	80x25	16/256K

Modes 2⁺ and 3⁺			
Standard	**Box Size**	**Char x Row**	**Colors**
CGA	8x8	80x25	16

Mode 7⁺			
Standard	**Box Size**	**Char x Row**	**Colors**
VGA	9x16	80x25	2/256K

Mode 7			
Standard	**Box Size**	**Char x Row**	**Colors**
MDA	9x14	80x25	Mono

Modes 23, 27, 33 & 37			
Standard	**Box Size**	**Char x Row**	**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 and ATI extended modes 54, 55, 62, 63, 64. Mode numbers are expressed in hexadecimal numbers.

Modes 4 and 5		
Standard	**Pels**	**Colors**
VGA	320x200	4/256K
CGA	320x200	4 (two sets)

Mode 6		
Standard	**Pels**	**Colors**
VGA	640x200	2/256K
CGA	640x200	2

Mode D		
Standard	**Pels**	**Colors**
VGA	320x200	16/256K

Mode E		
Standard	**Pels**	**Colors**
VGA	640x200	16/256K

Mode F		
Standard	**Pels**	**Colors**
VGA	640x350	2/256K

Mode 10		
Standard	**Pels**	**Colors**
VGA	640x350	16/256K

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 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

Host Address Space/Host Window

VGA drawing operations are performed by the system processor, which reads data from and writes data to the on-screen display memory. To accomplish this, the display memory is mapped to a specific segment (or segments) of the host processor memory address space. This is sometimes referred to as the **host window** to display memory.

Standard memory organization of the 80x86 processor with 1MB of addressable memory is illustrated in table 9-1.

Table 9-1.   PC Memory Map

Address	Content
F000:FFFF	
F000:0000	BIOS ROM
E000:0000	
	LAN, Tape Backup, EMS,...
CC00:0000	
C800:0000	XT Disk BIOS
C000:0000	VGA/EGA BIOS ROM
B800:0000	CGA Display Memory
B000:0000	MDA Display Memory
A000:0000	VGA/EGA Display Memory
9000:0000	Transient Program Area (User memory)
	Resident part of COMMAND.COM
	Disk buffers, Installable Drivers, ...
	DOS Kernel
0000:0400	BIOS Data Area
0000:0000	Interrupt Vectors

The host window used by the VGA varies depending on the mode of operation. Table 9-2 contains the standard host windows and sample modes using each window.

Table 9-2. VGA Host Windows

Content	Host Address Window
BIOS ROM	C000:0000h - C000:7FFFh
Display RAM Color Text (Mode 3)	B800:0000h - B800:7FFFh
Display RAM Monochrome Text (Mode 7)	B000:0000h - B000:7FFFh
Display RAM VGA Graphics (Modes F, 10,...)	A000:0000h - A000:FFFFh
Display RAM Extended Graphics	A000:0000h - A000:FFFFh
Display RAM Extended Graphics (128K addressable)	A000:0000h - B000:FFFFh

In text modes, which require relatively little data to be moved, a 32K space is used. In graphics modes, which require much more data, a 64K space is used. When all four VGA color planes are used, the processor can access 256K of display memory (4 x 64K).

As the screen resolution and number of colors increase, the amount of display memory that must be accessed by the processor also increases. In some high resolution modes, more than 256K of display memory must be accessed.

A simple way to gain access to more display memory is to increase the size of the host memory space used by the VGA from 64K to 128K, using the memory address space from A000:0 to B000:FFFF. This standard VGA option, which is selected via the Miscellaneous Register of the Graphics Controller is both convenient and efficient. However, it has its limitation in that it interferes with other co-resident display adapters such as MDA, CGA, or Hercules.

None of the standard modes on the *mach64* uses a 128K host window; 64K windows are used instead. An alternative way to access more than 64K is to use a display **memory mapping** mechanism, as discussed later in this chapter.

Packed vs. Planar Pixels

The color of each picture element (PEL or pixel) is determined by several bits in the display memory (2, 4, or 8 bits, depending on the display mode). Two basic types of display memory organization are used to define pixels:

- **Planar organization (color planes)**. Most VGA graphics modes (Dh, Eh, Fh, 10h, 11h and 12h) use color planes.
- **Packed pixel organization**. CGA modes 4,5,6 and VGA mode 13 use packed pixels.

Planar Modes

In planar modes, each pixel on the screen corresponds to a set of four bits (one bit in each plane), as shown in figure 9-2.

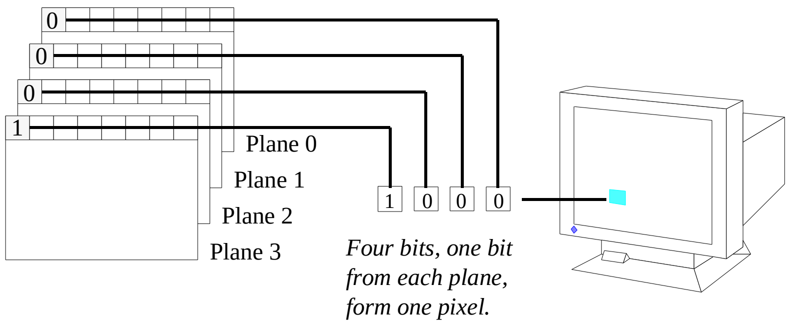


Figure 9-2. Planar Pixels

Packed Pixel Modes

For packed pixel modes, all bits of a pixel are packed into one or more bytes of display memory. In some modes, a byte holds more than one pixel. In 32K color modes, two bytes contain one pixel. In 256 color modes, a byte contains only one pixel, as shown in figure 9-3.

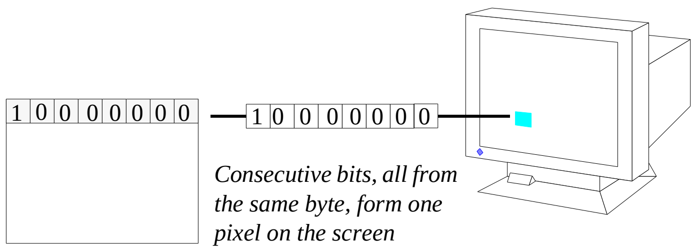

1 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0
Consecutive bits, all from the same byte, form one pixel on the screen

Figure 9-3. Packed Pixels

Display Memory Plane Selection

This display adapter can be configured with 256K, 512K, or 1M of display memory. Application programs access display memory via a 64K host window. To permit host access to all of the available display memories, the *mach64* contains two different memory mapping mechanisms that map the 64K host window to a selected 64K portion of display memory. One memory mapping mechanism, which is standard on all VGA products, utilizes the planar organization of VGA memory. Planar memory mapping, illustrated in figure 9-4, allows access to 256K of display memory (four planes of 64K).

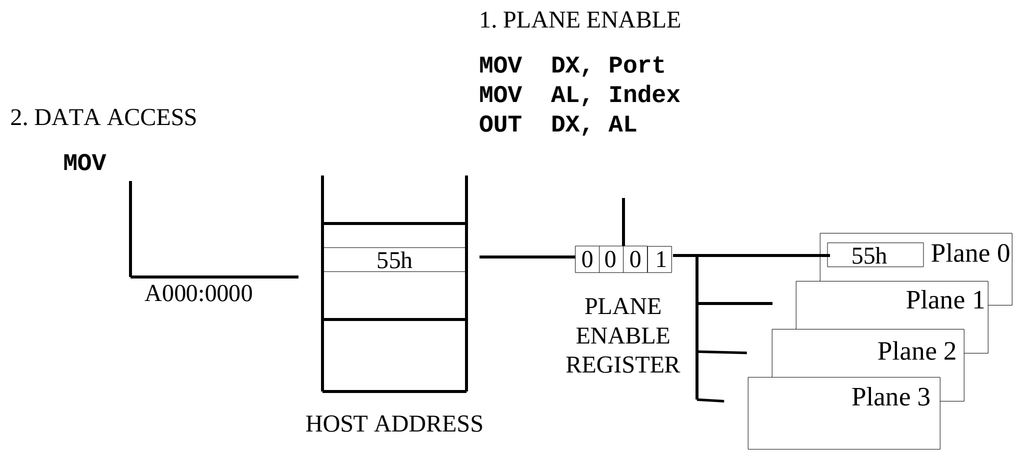


Figure 9-4.Plane Selection

Display Memory Paging

To give the processor access to a greater amount of display memory, the *mach64* also contains a number of display memory paging mechanisms that map selected portions of the display memory to the host window. **Page** is defined as a section of display memory which is addressed by the CPU through A000:0000. Typically, the page size is 64K, but in certain cases it may be 32K. It is similar in operation to EMS/LIM memory expansion boards.

The method for changing the page mapped to the host window is different for packed pixel and planar modes on the *mach64*. In packed pixel modes, data is read from and written to the screen at a selected page through small dual paged apertures. These apertures are 32K in size and are located at segment base addresses A000h and A800h. The read and write page of each aperture may be set independently. For a more detailed description of the small dual paged apertures and how to set their read and write pages, refer to the *mach64 Programmer's Guide*.

The method for changing the read or write page in planar modes varies depending on the type of *mach64* controller. For the *mach64GX* and *mach64CX*, an application must first select the desired page by loading a page select register. Data within that memory page may then be accessed in the host window. See figure 9-5 for an illustration of this process. Each mode computes the page number and screen offset differently. Currently, the *mach64* supports up to 16 pages.

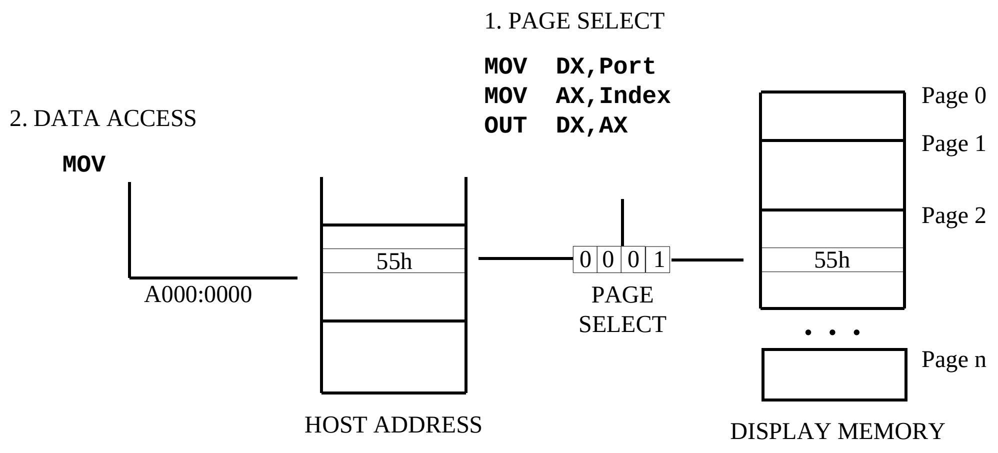


Figure 9-5. Page Selection

The page select register is part of the VGA extended register set contained within the *mach64GX* and *mach64CX*. These registers are fully described in Chapter 5. To select a display memory page, set the appropriate bit in the extended register ATI32.

Since the *mach64CT* and *mach64ET* do not contain the set of VGA extended registers contained in the other series of *mach64* controllers, extended register ATI32 cannot be used to select the read or write page. For the *mach64CT* and *mach64ET*, the small dual paged apertures described earlier must be used to access the display memory in planar modes. See the *mach64 Programmer's Guide* for an explanation of how to modify the read or write page while using the small apertures in a planar mode.

Emulations

The *mach64GX* and *mach64CX* are compatible with software written for the Color Graphics Adapter (CGA).   This compatibility is achieved by special registers that allow the *mach64* to emulate the CGA.

Extended Display Modes

In addition to total VGA compatibility, the *mach64* also provides new enhanced display modes with higher resolutions and more colors than standard VGA display modes. The number of extended modes supported by the BIOS depends on the amount of memory available and the hardware revision.

- Modes that display 132 columns of text are useful for spreadsheets and similar applications.
- High resolution graphics modes with 256 simultaneous color capability can be used to present full color photographic images with impressive fidelity.

- Modes that offer 16 colors at higher resolutions than the IBM VGA are popular for applications that involve fine visual details, such as CAD/CAM and desktop publishing.

Since the new modes were developed as extensions to the basic VGA, the programming models for these modes resemble the supported VESA SVGA modes. This means any application that supports VESA SVGA modes will work with the *mach64* without code changes. Table 9-3 lists the extended display modes supported by the *mach64*.

Table 9-3. Extended Display Modes

Mode	Type	Resolution	Colors	Memory Required
23h	Text	132 cols x 25 rows, 8x14 cell	16	256K
27h	Text	132 cols x 25 rows, 8x14 cell	mono	256K
33h	Text	132 cols x 44 rows, 8x8 cell	16	256K
37h	Text	132 cols x 44 rows, 8x8 cell	mono	256K
54h	Graphics	800 horizontal x 600 vertical	16	256K
55h	Graphics	1024 horizontal x 768 vertical	16 (planar)	512K
101h (VESA)/ 62h (ATI)	Graphics	640 horizontal x 480 vertical	256	512K
103h (VESA)/ 63h (ATI)	Graphics	800 horizontal x 600 vertical	256	512K
105h (VESA)/ 64h (ATI)	Graphics	1024 horizontal x 768 vertical	256	1M
107h	Graphics	1280x1024	256	2M

Mode Setting by BIOS Call

Setting Standard Modes and ATI Extended Modes

BIOS Function 0, mode select, can be used to initialize VGA and ATI extended mode of the *mach64*. For example, to invoke mode 54h, the following code can be used:

```
mov     al,54h
mov     ah,0
int     10h
```

Setting VESA Modes

VESA VBE Function 2, sets super VGA video mode. This function is used to initialize a video mode in super VGA mode. For example, to invoke mode 107h, the following code can be used:

```
mov     ax,4f02h
mov     bx,107
int     10h
cmp     ah,1
je      modefail
```

Mode Query Function

It is important to verify that the display being used is capable of supporting the colors and resolution of the selected display mode, and that the *mach64* has been properly configured. The mode table of your product can be found at the Appendix of the User's Guide.

To determine what modes are available, VBE function 00 can be used. This function provides information to the calling program about the general capabilities of the hardware. For example:

```
mov     ax,4f00h
mov     di,data_seg0
mov     es,di
xor     di,di
int     10h
```

where es:di points to information block structure define in the VESA super specification.

Summary of Display Modes

Extended text and graphics display modes of the *mach64* are summarized below.

Mode 23h - 132 x 25 Color Text

132-column text modes are useful for wide text displays such as spreadsheets. This mode uses the standard VGA 8x14 character set, and drives the display at a resolution of 1056 x 350 pixels. To support this mode, the display must be capable of displaying a horizontal resolution of 1056 pixels. While this exceeds the published specifications for most popular displays, acceptable results can still be achieved with many displays.

Mode 27h - 132 x 25 Monochrome Text

Mode 27h is the monochrome equivalent to mode 23h. It uses the same character set and operates at the same resolution.

Mode 33h - 132 x 44 Color Text

A smaller character set is used in this text mode in order to display more information on the screen; the 8x8 character set used is not as readable as the 8x14 character set used in modes 23h and 27h. As with other 132-column modes, the screen operates at a resolution of 1056 x 350 pixels. Most displays will show acceptable results at this resolution.

Mode 37h - 132 x 44 Monochrome Text

Mode 37h is the monochrome equivalent to mode 33h.

Mode 54h - 800 x 600 16 Color Graphics

800x600 is the highest 16-color resolution that can be supported using only 256K bytes of display memory. It is also the highest 16-color resolution that can be supported without utilizing display memory paging. This resolution is also the upper limit of resolution on many popular displays.

Mode 55h - 1024 x 768 16 Color Graphics

Only the highest frequency VGA displays can operate in this mode, which is the highest resolution mode of the *mach64* VGA. 512K bytes of display memory are required. Display memory paging is used to make the full display memory available to the processor.

This resolution is becoming a standard for medium cost displays (640x480 being the previous standard) and has been added to many popular monitors to support the SVGA adapter. The *mach64* VGA is capable of operating in this mode with interlaced as well as non-interlaced displays, and will automatically select the best one based on the attached display (indicated by configuration settings). Programming in this mode can be accomplished through a 128K aperture located at address A000h (on ATI28800-5 chips and above).

Mode 101h - 640 x 480 256 Color Graphics

This resolution is popular because it equals the highest standard VGA resolution, but has added 256 color capability and is supported by all low cost analog monitors. This mode requires 512K of display memory. Display memory paging is needed to make the larger display memory available to the processor.

Mode 103h - 800 x 600 256 Color Graphics

Mode 103h operates at the highest resolution that is available on the majority of multi-frequency (multi-scanning) monitors. Full color photographic images can be displayed with remarkable fidelity at this resolution. This mode requires 512K of display memory. Display memory paging is required to make the full display memory available to the processor.

Mode 105h - 1024 x 768 256 Color Graphics

Mode 105h operates at the highest resolution that is available on the majority of multi-frequency (multi-scanning) monitors. Full color photographic images can be displayed with remarkable fidelity at this resolution. This mode requires 1MB of display memory. Display memory paging is required to make the full display memory available to the processor.

Mode 107h - 1280 x 1024 256 Color Graphics

Mode 107h becomes a popular screen size for serious desktop and publishing user. It is recommended to set the refresh rate to 60Hz or higher to reduce flickering. The refresh rate can be set by running the INSTALL program found in Disk#1 of the product diskettes. This mode requires 2MB of video memory.

Display Memory Organization

Drawing algorithms for a given display mode are largely dictated by the organization of display memory (the organization of a pixel in memory and the mapping of display memory to the screen) for that mode. This section provides detailed descriptions of display memory organization for all extended modes.

Extended Text Modes

Modes 23h, 27h, 33h, and 37h

These modes utilize memory maps similar to those used in standard text modes (VGA modes 2, 3 and 7), except that the number of characters per line is increased from 80 to 132. Thus, the number of bytes used per text line increases from 160 to 264 (each character requires one ASCII character byte and one attribute byte). ASCII code is stored at even memory addresses and attribute data is stored at odd memory addresses.

Table 9-4 shows the standard IBM color text attributes. In 16 color text mode, D7 is defined as foreground blinking and D3 is the intensity bit. It allows one of sixteen colors for foreground and one of eight colors for the background. In addition, the character can be blinked.

Table 9-5 shows the standard IBM monochrome text attributes. D7 and D3 control the blinking and intensity respectively. Reverse video occurs only when bits 4-6 are one and bits 2-0 are zero.   The organization of display memory for all extended text modes can be seen in figure 9-6. The character is stored in plane 0 and the attribute in plane 1. The ROM character patterns are transferred in plane 2 and addressed by the CRT Controller.

Table 9-4.   Standard Color Text Attributes

Attribute	Standard Color	Attribute	Intensified Color
000	Black	1000	Gray
001	Blue	1001	Light Blue
010	Green	1010	Light Green
011	Cyan	1011	Light Cyan
100	Red	1100	Light Red
101	Magenta	1101	Light Magenta
110	Brown	1110	Yellow
111	Gray	1111	White

Table 9-5. Standard Monochrome Text Attributes

Attribute	Color
00000000	Blank
00000111	Normal
10000111	Blinking
00001111	Intensified
10001111	Blinking and Intensified
00000001	Underlined
10000001	Blinking and Underlined
00001001	Intensified and Underlined
10001001	Blinking, Intensified and Underlined
01110000	Reverse Video
11110000	Reverse Video and Blinking

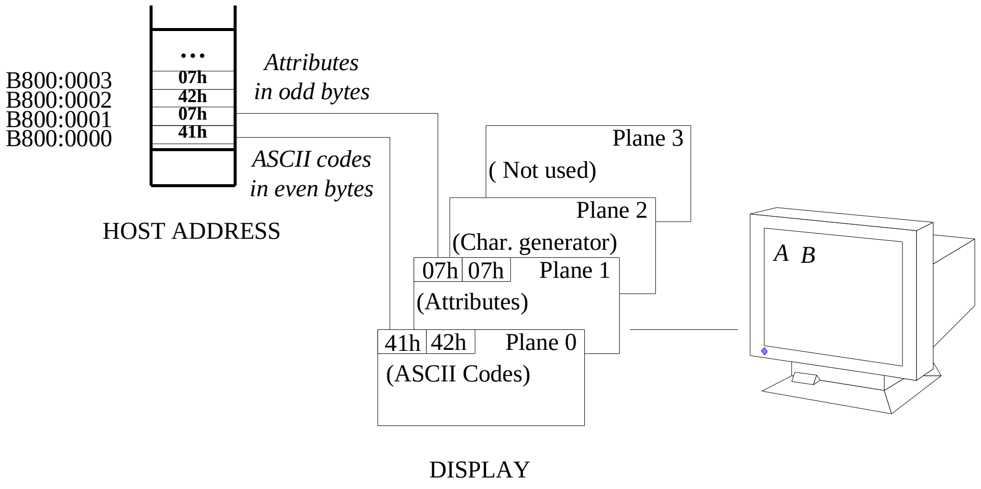


Figure 9-6. Display Memory Organization in Extended Text Modes

Extended Graphics Modes

Modes 54h - 800 x 600 (16 colors)

Memory organization for this mode resembles VGA mode 12h (640 x 480 16 color planar graphics), except that both the number of pixels per scan line and the number of scan lines increase. Display memory is organized as four color planes of 64K each. Each pixel consumes one bit position in each color plane (Plane Selection). The most significant bit in each byte (bit D7) corresponds to the leftmost pixel for that byte. One hundred consecutive bytes are used for each raster line.

For 600 lines, 60,000 bytes are required in each plane, which is less than the 64K that is addressable within the host window. So no display memory paging is required. Only 256K of display memory is required to support this mode.

Default colors are the same as the standard VGA colors (see table 9-4). To change colors in this mode, use the palette registers of the Attribute Controller.

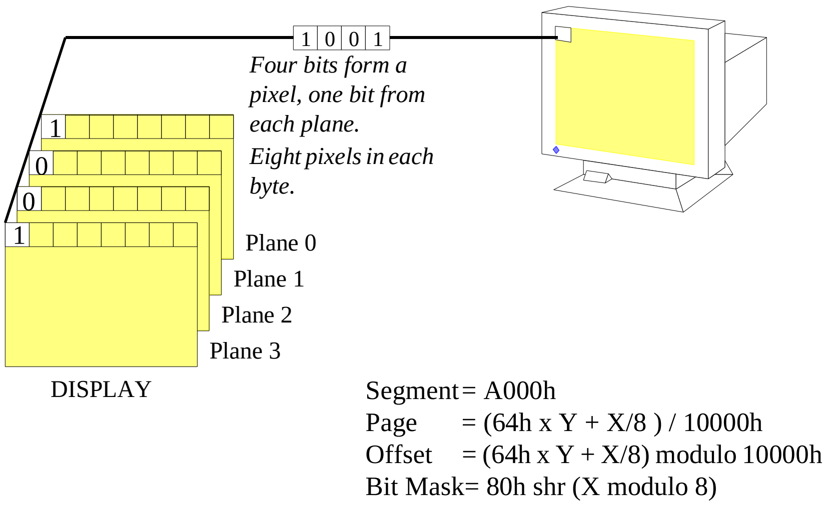

1 0 0 1
Four bits form a pixel, one bit from each plane.
Eight pixels in each byte.
1
0
0
1
Plane 0
Plane 1
Plane 2
Plane 3
DISPLAY
Segment = A000h
Page = (64h x Y + X/8) / 10000h
Offset = (64h x Y + X/8) modulo 10000h
Bit Mask= 80h shr (X modulo 8)

Figure 9-7. Memory Organization - Mode 54h

Table 9-6. Standard VGA Color Palette - 16 Color Graphics

Index	Color
0000	Black
0001	Blue
0010	Green
0011	Cyan
0100	Red
0101	Magenta
0110	Brown
0111	White
1000	Dark Gray
1001	Light Blue
1010	Light Green
1011	Light Cyan
1100	Light Red
1101	Light Magenta
1110	Yellow
1111	Intensified White

Mode 55h - 1024 x 768 (16 colors)

Memory organization for this mode resembles VGA mode 12h (640 x 480 16 color planar graphics), except that both the number of pixels per scan line and the number of scan lines increase. Display memory is organized as two pages, with four color planes in each page and 64K in each plane. Each pixel consumes one bit position in each color plane (see figure 9-2).

The most significant bit in each byte (bit D7) corresponds to the leftmost pixel for that byte. Each raster line uses 128 consecutive bytes. For 768 lines, this requires 98,304 bytes in each plane, which is addressable in two, 64K pages in the host window. 512K of display memory is required to support this mode.

Default colors are the same as the standard VGA colors (see table 9-4). To change colors in this mode, use the DAC registers.

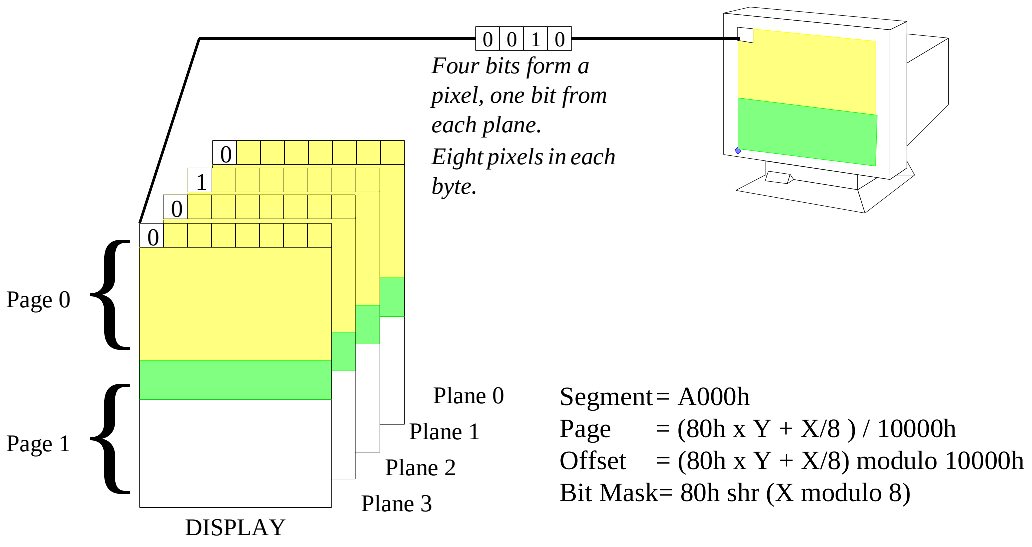

0 0 1 0
Four bits form a pixel, one bit from each plane.
Eight pixels in each byte.
0
1
0
0
Page 0
Page 1
Plane 0
Plane 1
Plane 2
Plane 3
DISPLAY
Segment = A000h
Page = (80h x Y + X/8) / 10000h
Offset = (80h x Y + X/8) modulo 10000h
Bit Mask= 80h shr (X modulo 8)

Figure 9-8. Display Memory Organization - Mode 55h

Mode 101h - 640 x 480 (256 colors)

Display memory organization for this mode resembles VGA mode 13h (320 x 200 256 color packed pixel graphics), except that the number of pixels per scan line is doubled and the number of scan lines is increased. Each pixel requires one byte of display memory. Each raster scan line uses 640 consecutive bytes. For 480 lines, this requires 307.2K (five 64K pages) of display memory. The adapter requires 512K bytes of memory. The memory map for this mode can be seen in figure 9-9.

Default colors are the same as for mode 13h. To change the default colors, use the DAC registers; the palette registers in the Attribute Controller should not be modified.

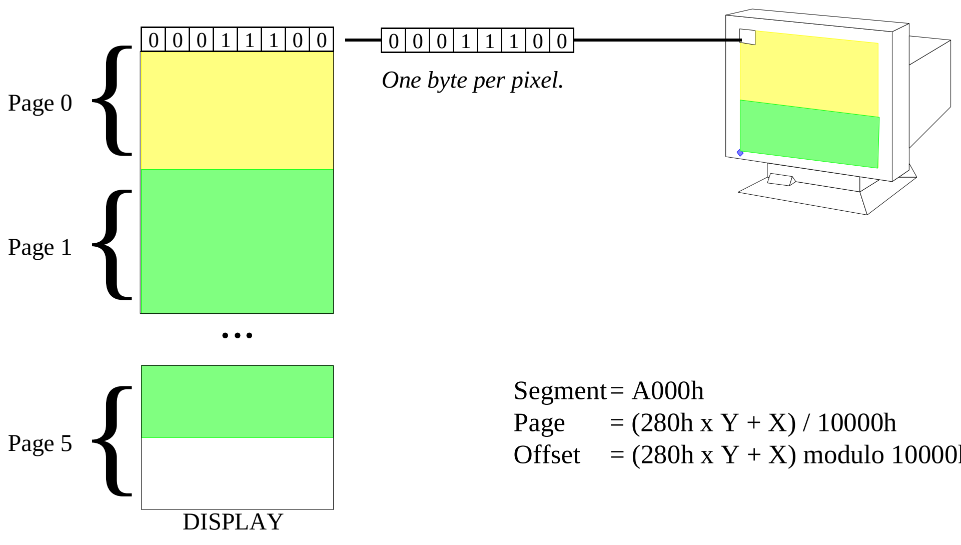


Figure 9-9. Display Memory Organization - Mode 62h

Mode 103h - 800 x 600 (256 colors)

Display memory organization for this mode resembles VGA mode 13h (320 x 200 256 color packed pixel graphics), except that the number of pixels per scan line and number of scan lines are both increased. Each pixel requires one byte of display memory. Each raster scan line uses 800 consecutive bytes. For 600 lines, this requires 480,000 bytes (eight 64K pages) of display memory. The adapter requires 512K bytes of memory. The memory map for this mode can be seen in figure 9-10.

Default colors are the same as for mode 13h. To change the default colors, use the DAC registers; the palette registers in the Attribute Controller should not be modified.

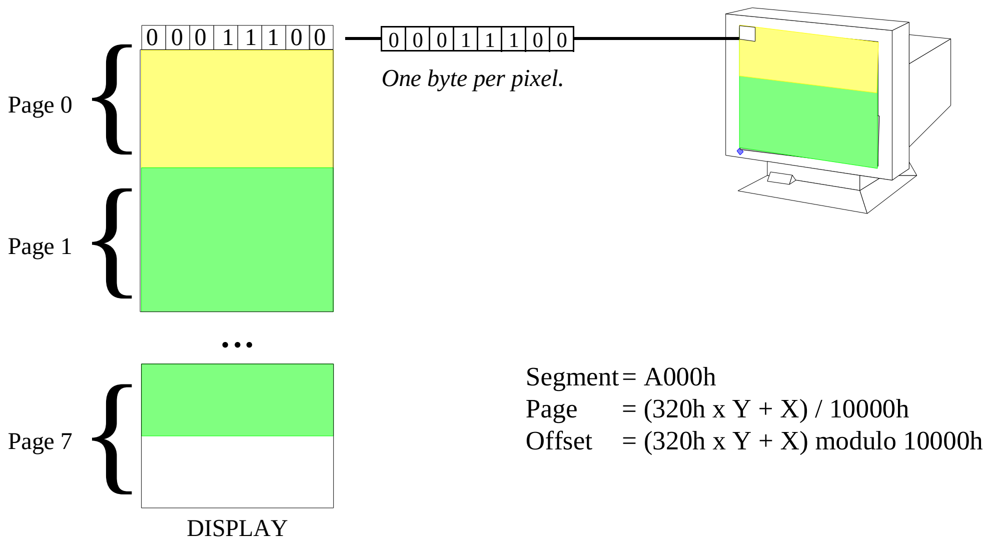
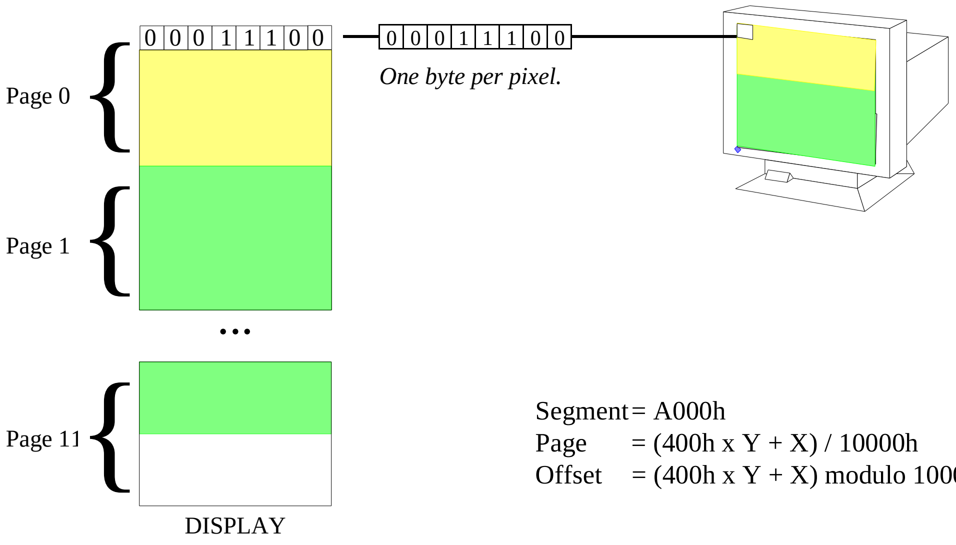


Figure 9-10. Display Memory Organization - Mode 63h

Mode 105h - 1024 x 768 (256 colors)

Display memory organization for this mode resembles VGA mode 13h. Each pixel requires one byte of display memory and 1024 bytes per scanline. The total display memory required for 768 lines is 786,432 bytes (12 64K pages). This mode works with an adapter of 1M memory only. The memory map for this mode is drawn in figure 9-11.

Default colors are the same as for mode 13h. To change the default colors, use the DAC registers; the palette registers in the Attribute Controller should not be modified.



Segment = A000h
Page = (400h x Y + X) / 10000h
Offset = (400h x Y + X) modulo 10000h

Figure 9-11. Display Memory Organization - Mode 64h

Table 9-7.   Display Mode Specifications - Multisync Monitors

Mode	Box Size	PELs	Colors	Horiz. Sync (KHz)	Vert. Sync (Hz)	H/V Polarities	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/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/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/CGA	8x8	320x200	4 (2 sets)	31.47	70	-/+	25.2
5/VGA	8x8	320x200	4/256K	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/CGA	8x8	640x200	2	31.47	70	-/+	25.2
7+/VGA	9x16	720x400	2/256K	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
E/VGA	8x8	640x200	16/256K	31.47	70	-/+	25.2
F/VGA	8x14	640x350	2/256K	31.47	70	+/-	25.2
10/VGA	8x14	640x350	16/256K	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
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
54^{V}	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
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

(continued on next page)

Table 9-7. Display Mode Specifications - Multisync Monitors

Mode	Box Size	PELs	Colors	Horiz. Sync (KHz)	Vert. Sync (Hz)	H/V Polarities	Dot Clock (MHz)
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
63^{V}	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
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

Notes:
I = Interlaced Mode
V = Applicable to VESA compatible monitors

This page intentionally left blank.

Chapter 10
VGA-Compatible Registers

Overview

VGA registers in *mach64* controllers are fully hardware-compatible with registers in the IBM VGA video adapter. (See *Chapter 8, VGA Programming Overview*.)

VGA-compatible registers are listed in the *Index*. For convenience, they are also arranged by I/O Port address (and index) on the following page. This chapter contains detailed descriptions of the compatible registers, arranged by function, as follows:

Register Functional Classes	**Page**
CRT Controller Registers (GRAxx)	*10-5*
Attribute Controller Registers (ATTRxx)	*10-18*
General Status and Configuration Registers (GENxx)	*10-22*
Sequencer Registers (SEQxx)	*10-25*
DAC Registers (DACxx)	*10-29*
Graphics Controller Registers (GRAxx)	*10-30*

VGA Compatible Registers – By I/O Port

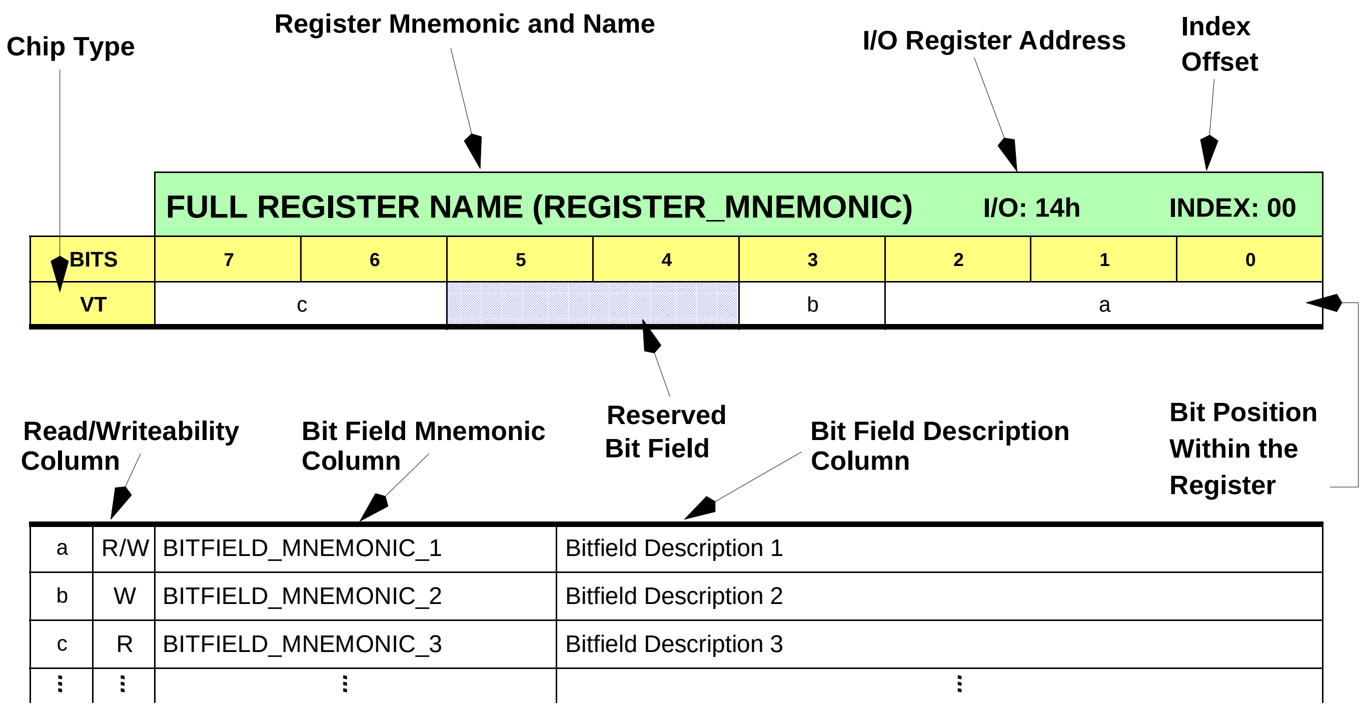
Port	Index	Function	Type	Mnemonic	Register Name	Page
0102	–	General	W	GENVS	VGA Sleep	10-22
03?4	–	CRT Controller	R/W	CRTX	CRTC Index	10-5
03?5	0		R/W	CRT00	Horizontal Total	10-5
03?5	1		R/W	CRT01	Horizontal Display Enable End	10-5
03?5	2		R/W	CRT02	Start Horizontal Blanking	10-6
03?5	3		R/W	CRT03	End Horizontal Blanking	10-6
03?5	4		R/W	CRT04	Start Horizontal Retrace	10-6
03?5	5		R/W	CRT05	End Horizontal Retrace	10-7
03?5	6		R/W	CRT06	Vertical Total	10-7
03?5	7		R/W	CRT07	CRTC Overflow	10-8
03?5	8		R/W	CRT08	Preset Row Scan	10-9
03?5	9		R/W	CRT09	Maximum Scan Line	10-9
03?5	A		R/W	CRT0A	Cursor Start	10-10
03?5	B		R/W	CRT0B	Cursor End	10-10
03?5	C		R/W	CRT0C	Start Address (High Byte)	10-11
03?5	D		R/W	CRT0D	Start Address (Low Byte)	10-11
03?5	E		R/W	CRT0E	Cursor Location (High Byte)	10-11
03?5	F		R/W	CRT0F	Cursor Location (Low Byte)	10-12
03?5	10		R/W	CRT10	Start Vertical Retrace	10-12
03?5	11		R/W	CRT11	End Vertical Retrace	10-13
03?5	12		R/W	CRT12	Vertical Display Enable End	10-13
03?5	13		R/W	CRT13	Offset	10-14
03?5	14		R/W	CRT14	Underline Location	10-14
03?5	15		R/W	CRT15	Start Vertical Blanking	10-15
03?5	16		R/W	CRT16	End Vertical Blanking	10-15
03?5	17		R/W	CRT17	CRT Mode	10-16
03?5	18		R/W	CRT18	Line Compare	10-17
03?5	1E,1F		R	CRT1E, 1F	Graphic Controller Index Decode	10-17
03?5	22		R	CRT22	RAM Data Latch Readback	10-17
03?A	–	General	W	GENFC	Feature Control	10-22
03?A	–		R	GENS1	Input Status 1	10-22

(continued on next page)

Port	Index	Function	Type	Mnemonic	Register Name	Page
03C0	–	Attribute Controller	R/W	ATTRX	Attribute Controller Index	*10-18*
03C0	00-0F		W	ATTR(00:0F)	Palette (00 to 0F)	*10-18*
03C0	10		W	ATTR10	Mode Control	*10-19*
03C0	11		W	ATTR11	Overscan Color	*10-19*
03C0	12		W	ATTR12	Color Map Enable	*10-20*
03C0	13		W	ATTR13	Horizontal PEL Panning	*10-20*
03C0	14		W	ATTR14	Color Select	*10-21*
03C1	00-0F		R	ATTR(00:0F)	Palette (00 to 0F)	*10-18*
03C1	10		R	ATTR10	Mode Control	*10-19*
03C1	11		R	ATTR11	Overscan Color	*10-19*
03C1	12		R	ATTR12	Color Map Enable	*10-20*
03C1	13		R	ATTR13	Horizontal PEL Panning	*10-20*
03C1	14		R	ATTR14	Color Select	*10-21*
03C2	–	General	W	GENMO	Miscellaneous Output	*10-23*
03C2	–		R	GENSO	Input Status 0	*10-24*
03C3	–		R	GENENB	Video Subsystem Enable (Board)	*10-24*
03C4	–	Sequencer	R/W	SEQX	Sequencer Index	*10-25*
03C5	0		R/W	SEQ00	Reset	*10-25*
03C5	1		R/W	SEQ01	Clock Mode	*10-26*
03C5	2		R/W	SEQ02	Map Mask	*10-27*
03C5	3		R/W	SEQ03	Character Map Select	*10-27*
03C5	4		R/W	SEQ04	Memory Mode	*10-28*
03C6	–	DAC	R/W	DAC_MASK	DAC Mask	*10-29*
03C7	–		R/W	DAC_R_INDEX	DAC Read Current Color Index	*10-29*
03C8	–		R/W	DAC_W_INDEX	DAC Write Current Color Index	*10-29*
03C9	–		R/W	DAC_DATA	DAC Data	*10-29*
03CA	–	General	R	GENFC	Feature Control	*10-22*
03CC	–		R	GENMO	Miscellaneous Output	*10-23*
03CE	–	Graphics Controller	R/W	GRAX	Graphics Controller Index	*10-30*
03CF	0		R/W	GRA00	Set/Reset	*10-30*
03CF	1		R/W	GRA01	Enable Set/Reset	*10-31*
03CF	2		R/W	GRA02	Color Compare	*10-31*
03CF	3		R/W	GRA03	Data Rotate	*10-32*
03CF	4		R/W	GRA04	Read Map Select	*10-32*
03CF	5		R/W	GRA05	Graphics Mode	*10-33*
03CF	6		R/W	GRA06	Graphics Miscellaneous	*10-34*
03CF	7		R/W	GRA07	Color Don't Care	*10-35*
03CF	8		R/W	GRA08	Bit Mask	*10-35*
46E8	–	General	W	GENENA	Video Subsystem Enable (Add-On)	*10-24*

Register Legend

The table below describes the layout of all the Register Description Tables in Chapter 10.


Chip Type
Register Mnemonic and Name
I/O Register Address
Index Offset
FULL REGISTER NAME (REGISTER_MNEMONIC)
I/O: 14h
INDEX: 00
BITS
7
6
5
4
3
2
1
0
VT
c
b
a
Reserved Bit Field
Read/Writeability Column
Bit Field Mnemonic Column
Bit Field Description Column
Bit Position Within the Register

a	R/W	BITFIELD_MNEMONIC_1	Bitfield Description 1
b	W	BITFIELD_MNEMONIC_2	Bitfield Description 2
c	R	BITFIELD_MNEMONIC_3	Bitfield Description 3
⋮	⋮	⋮	⋮

In the following tables, the registers are grouped according to one of the six **Register Classes** listed on *page 10-1*. **Chip Type** refers to the VT chip, which supports VGA-compatible registers. The **I/O Address** indicates the read and write address of the register. The **Index** is also shown if the register is accessible indirectly.

VGA CRT Controller Registers

CRTC INDEX (CRTX)						I/O: 3?4h		INDEX:–
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	▨	a					

a	R/W	VCRTC_IDX[5:0]	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.

Note:

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

HORIZONTAL TOTAL (CRT00)						I/O: 3?5h		INDEX: 00h
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	a							

a	R/W	H_TOTAL[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).

HORIZONTAL DISPLAY ENABLE END (CRT01)						I/O: 3?5h		INDEX: 01h
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	a							

a	R/W	H_DISP_END[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).

	START HORIZONTAL BLANKING (CRT02)					**I/O: 3?5**		**INDEX: 02h**
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	a							

a	R/W	H_BLANK_START[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).

	END HORIZONTAL BLANKING (CRT03)					**I/O: 3?5h**		**INDEX: 03h**
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	c	b		a				

a	R/W	H_BLANK_END[4:0]	H Blanking End bits 4-0, respectively. These are the five low-order bits (of six bits in total) 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”.
b	R/W	H_DE_SKEW[1:0]	Display Enable Skew: 00 = Zero-character-clock skew. 01 = One-character-clock skew. 10 = Two-character-clock skew. 11 = Three-character-clock skew.
c	R/W	CR10CR11_R_DISB	Compatibility Read: 0 = Enables write-only to CRT10 and CRT11 1 = Enables read/write access to both vertical retrace start/end register CRT10 and CRT11

Note:

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

	START HORIZONTAL RETRACE (CRT04)					**I/O: 3?5h**		**INDEX: 04h**
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	a							

a	R/W	H_SYNC_START[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).

	END HORIZONTAL RETRACE (CRT05)					I/O: 3?5H		INDEX: 05h
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	c	b		a				

a	R/W	H_SYNC_END[4:0]	H Retrace Ends bits: These are the 5-bit result from the sum of CRT04 plus the width of the hoizontal retrace pulse in character clock units.
b	R/W	H_SYNC_SKEW[1:0]	H Retrace Delay bits: 00 = Zero character clocks 01 = One character clock 10 = Two character clocks 11 = Three character clocks These two bits skew the Horizontal Retrace pulse
c	R/W	H_BLANK_START[5]	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).

	VERTICAL TOTAL (CRT06)					I/O: 3?5h		INDEX: 06h
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	a							

a	R/W	V_TOTAL[7:0]	These are the eight low-order bits of the 10-bit vertical total register. The two high-order bits are CRT07[5:0] 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).

	CRTC OVERFLOW (CRT07)				**I/O: 3?5h**		**INDEX: 07h**	
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	h	g	f	e	d	c	b	a

a	R/W	V_TOTAL[8]	V Total Bit 8 (CRT06) Bit 8 of 10-bit vertical count for V Total (for functional description, see CRT06 register)
b	R/W	V_DISP_END[8]	End V Display Bit 8 (CRT12) Bit 8 of 10-bit vertical count for V Display enable end (for functional description, see CRT12 register).
c	R/W	V_SYNC_START[8]	Start V Retrace Bit 8 (CRT10) Bit 8 of 10-bit vertical count for V Retrace start (for functional description, see CRT10 register)
d	R/W	V_BLANK_START[8]	Start V Blanking Bit 8 (CRT15) Bit 8 of 10-bit vertical count for V Blanking start (for functional description, see CRT15 register)
e	R/W	LINE_CMP[8]	Line Compare Bit 8 (CRT18) Bit 8 of 10-bit vertical count for Line Compare (for functional description, see CRT18 register)
f	R/W	V_TOTAL[9]	V Total Bit 9 (CRT06) Bit 9 or 10-bit vertical count for V Total (for functional description, see CRT06 register)
g	R/W	V_DISP_END[9]	End V Display Bit 9 (CRT12) Bit 9 of 10-bit vertical count for V Display Enable End (for functional description, see CRT12 register)
h	R/W	V_SYNC_START[9]	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.

	PRESET ROW SCAN (CRT08)				**I/O: 3?5h**		**INDEX: 08h**	
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	b		a				

a	R/W	ROW_SCAN_START[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.
b	R/W	BYTE_PAN[1:0]	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).

Note:

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

	MAXIMUM SCAN LINE (CRT09)				**I/O: 3?5h**		**INDEX: 09h**	
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	d	c	b	a				

a	R/W	MAX_ROW_SCAN[4:0]	Maximum Scan Line bits. These bits define a value that is the actual number of scan line per character minus one.
b	R/W	V_BLANK_START[9]	Start V Blanking bit 9 (CRT15) Bit 9 of 10-bit vertical count for V Blanking start (for functional description, see CRT15 register).
c	R/W	LINE_CMP[9]	Line Compare bit 9 (CRT18) Bit 9 of 10-bit vertical count for line compare (for functional description, see CRT18 register)
d	R/W	DOUBLE_CHAR_HEIGHT	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 line 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.

Note:

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

	CURSOR START (CRT0A)				**I/O: 3?5h**		**INDEX: 0Ah**	
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	▨	b	a				

a	R/W	CURSOR_START[4:0]	Cursor Starts 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 which is the same height as the character cell.
b	R/W	CURSOR_DISABLE	Cursor On/Off 0 = Cursor on. 1 = Cursor off.

Note:

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

	CURSOR END (CRT0B)				**I/O: 3?5h**		**INDEX: 0Bh**	
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	b		a				

a	R/W	CURSOR_END[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 which is the same height as the character cell.
b	R/W	CURSOR_SKEW[1:0]	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.

Note:

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

	START ADDRESS (HIGH BYTE) (CRT0C)					I/O: 3?5h		INDEX: 0Ch
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	a							

a	R/W	DISP_START[15:8]	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 0.

Note:

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

	START ADDRESS (LOW BYTE) (CRT0D)					I/O: 3?5h		INDEX: 0Dh
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	a							

a	R/W	DISP_START[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 0.

Note:

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

	CURSOR LOCATION (HIGH BYTE) (CRT0E)					I/O: 3?5h		INDEX: 0Eh
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	a							

a	R/W	CURSOR_LOC[15:8]	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.

Note:

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

	CURSOR LOCATION (LOW BYTE) (CRT0F)			I/O: 3?5h			INDEX: 0Fh	
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	a							

a	R/W	CURSOR_LOC[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.

Note:

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

	START VERTICAL RETRACE (CRT10)			I/O: 3?5h			INDEX: 10h	
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	a							

a	R/W	V_SYNC_START[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.

		END VERTICAL RETRACE (CRT11)				I/O: 3?5h		INDEX: 11h
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	e	d	c	b	a			

a	R/W	V_SYNC_END[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.
b	R/W	V_INTR_CLR	V Retrace Interrupt Set: 0 = V Retrace interrupt cleared.
c	R/W	V_INTR_EN	V Retrace Interrupt Disabled: 0 = Enable V Retrace interrupt
d	R/W	SEL_5RFRSH	Reserved
e	R/W	C0T7_WR_ONLY	Write Protect (CRT00-CRT06): 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 CRTO7[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 CRT03[7] is set to one. It is write-only enabled if CRT03[7] is set to zero.

		VERTICAL DISPLAY ENABLE END (CRT12)				I/O: 3?5h		INDEX: 12h
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	a							

a	R/W	V_DISP_END[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).

	OFFSET (CRT13)					I/O: 3?5h		INDEX: 13h
BITS	7	6	5	4	3	2	1	0
VT	a							

a	R/W	DISP_PITCH[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).

	UNDERLINE LOCATION (CRT14)					I/O: 3?5h		INDEX: 14h
BITS	7	6	5	4	3	2	1	0
VT	▨	c	b	a				

a	R/W	UNDRLN_LOC[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.
b	R/W	ADDR_CNT_BY4	Count-by-4: 0 = Character clock is used unmodified at the memory address counter for byte addressing. 1 = Character clock is divided-by-4 at the clock input to the Memory address counter. Count-by-4 clocks are used for double-word addressing. This bit overrides Addr_Cnt_By2 bit CRT17[3].
c	R/W	DOUBLE_WORD	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]

Note:

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

	START VERTICAL BLANKING (CRT15)					I/O: 3?5h		INDEX: 15h
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	a							

a	R/W	V_BLANK_START[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 number of displayed lines minus one.

Note:

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

	END VERTICAL BLANKING (CRT16)					I/O: 3?5h		INDEX: 16h
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT								

a	R/W	V_BLANK_END[7:0]	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.

Note:

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

	CRT MODE (CRT17)				I/O: 3?5h		INDEX: 17h	
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	g	f	e	▨	d	c	b	a

a	R/W	RAO_AS_A13b	Compatibility Mode: 0 = Substitutes row scan counter bit 0 as bit 13 of CRTC output during active display time. For example, this allows for compatibility with the 6845 controller or CGA APA modes. 1 = Enables row scan counter bit 13 as bit 13 of CRTC output.
b	R/W	RA1_AS_A146	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 = Elects row scan counter bit 14 (RA14) as bit 14 at the CRTC output
c	R/W	VCOUNT_BY2	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).
d	R/W	ADDR_CNT_BY2	Count_by_2: 0 = Increments the memory address counter for every character clock. 1 = Increments the memory address counter for every two character clocks.
e	R/W	WRAP_A15toA0	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.
f	R/W	BYTE_MODE	Byte/Word Mode: 0 = Selects word mode memory addressing. The memory address is rotated left by one. 1 = Selects byte mode memory addressing.
g	R/W	CRTC_SYNC_EN	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.

		LINE COMPARE (CRT18)					I/O: 3?5h		INDEX: 18h
BITS		**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT		a							
a	R/W	LINE_CMP[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 the 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 panned 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 INDEX DECODE (CRT1E,1F)					I/O: 3?5h		INDEX: 1Eh, 1Fh
BITS		**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT		a							
a	R	GRPH_DEC[8:0]	This register is used to read back the graphics controller index decode.						

Note:

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

		RAM DATA LATCH READBACK (CRT22)					I/O: 3?5h		INDEX: 22h
BITS		**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT		a							
a	R	GRPH_LATCH_DATA[7:0]	This register is used to read the data in the Graphics Controller CPU data latches. The Graphics Controller Read Map Select register bits 0 and 1 determines which byte is read back.						

Note:

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

VGA Attribute Controller Registers

	ATTR INDEX (ATTRX)					I/O: 3C0h		INDEX:–
BITS	7	6	5	4	3	2	1	0
VT	▨	▨	b	a				

a	R/W	ATTR_IDX[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.
b	R/W	ATTR_PALRW_ENB	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.

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.

	PALETTE 00-0F (ATTR00_0F)					I/O: 3C1h(R) 3C0h(W)		INDEX: 00 to 0Fh
BITS	7	6	5	4	3	2	1	0
VT	▨	▨	a					

a	R/W	ATTR_PAL[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.

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.

	MODE CONTROL (ATTR10)					**I/O: 3C1h(R) 3C0h(W)**		**INDEX: 10h**
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	g	f	e	▨	d	c	b	a

a	R/W	ATTR_GRPH_MODE	Graphics/*Alphanumeric Mode: 0 = Selects A/N: Alphanumeric mode 1 = Selects APA: graphics mode
b	R/W	ATTR_MONO_EN	Monochrome/*Color Attributes Select 0 = Selects color display 1 = Selects monochrome display
c	R/W	ATTR_LGRPH_EN	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.
d	R/W	ATTR_BLINK_EN	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
e	R/W	ATTR_PANTOPONLY	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”.
f	R/W	ATTR_PCLKBY2	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.
g	R/W	ATTR_CSEL_EN	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.

	OVERSCAN COLOR (ATTR11)					**I/O: 3C1h(R) 3C0h(W)**		**INDEX: 11h**
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	a							

a	R/W	ATTR_OVSC[7:0]	Overscan color

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].

	COLOR MAP ENABLE (ATTR12)					I/O: 3C1h(R) 3C0h(W)		INDEX: 12h
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	▨	b		a			

a	R/W	ATTR_MAP_EN[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.
b	R/W	ATTR_VSMUX[1:0]	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

	HORIZONTAL PEL PANNING (ATTR13)					I/O: 3C1h(R) 3C0h(W)		INDEX: 13h
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	▨	▨	▨	a			

a	R/W	ATTR_PPAN[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	-	-

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.

	COLOR SELECT (ATTR14)					I/O: 3C1h(R) 3C0h(W)		INDEX: 14h
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	▨	▨	▨	b		a	

a	R/W	ATTR_CSEL[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
b	R/W	ATTR_CSEL[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-F.

General VGA Status and Configuration Registers

	VGA SLEEP (GENVS)					I/O:102h		INDEX:–
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	▨	▨	▨	▨	▨	▨	a

a	W	VGA_ENABLE2	VGA Sleep: 0 = Disables VGA video subsystem (controller) The VGA video sybsystem 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

Notes:

1. Writes to this register are controlled by GENENA[4].
2. Example of enabling 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
```

	FEATURE CONTROL (GENFC)					I/O: 3CAh(R) 3?Ah(W)		INDEX:–
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	▨	▨	▨	a	▨	▨	▨

a	R/W	VSYNC_SEL	Vertical Sync Select: 0 = Normal vertical sync 1 = Sync is ‘vertical sync’ ORed’ vertical display enable’

Notes:

? = B when GENMO[0]=0 (Monochrome emulation).

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

	INPUT STATUS 1 (GENS1)					I/O: 3?Ah		INDEX:–
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	▨	c	c	b	▨	▨	a

a	R	NO_DISPLAY	Display Enable: 0 = Enables display of video data 1 = Disables display of video data
b	R	VGA_VSTATUS	Vertical Retrace Status

(continued on next page)

	INPUT STATUS 1 (GENS1)					**I/O: 3?Ah**		**INDEX:–**
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	▨	c		b	▨	▨	a

c	R	PIXEL_READ_BACK[1:0]	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

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.

	MISCELLANEOUS OUTPUT (GENMO)					**I/O: 3CCh(R) 3C2h(W)**		**INDEX:–**
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	e		d	▨	c		b	a

a	R/W	GENMO_MONO_ADDRESS	0 = Addressing for monochrome emulation (0) 1 = Addressing for color/graphic emulation
b	R/W	VGA_RAM_ENABLE	0 = Disables CPU access to video RAM (0) (default value) 1 = Enables CPU access to video RAM
c	R/W	VGA_CKSEL[1:0]	00 = 25.1744 MHz (640PELs) 01 = 28.3212 MHz (720PELs)
d	R/W	ODD_EVEN_PGSEL	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 odd (high) video memory locations 1 = Selects even (low) video memory locations
e	R/W	VGA_VSYNC_POL VGA_HSYNC_POL	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.

	INPUT STATUS 0 (GENS0)					I/O: 3C2h		INDEX:–
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	b	▨	▨	a	▨	▨	▨	▨

a	R	SENSE_SWITCH	Switch Sense: 0 = Output state of the DAC lookup table. Comparators are inactive 1 = Output state of the DAC lookup table. Comparators are active
b	R	CRT_INTR	CRT Interrupt: 0 = Vertical retrace interrupt is cleared 1 = Vertical retrace interrupt is pending

	VIDEO SUBSYSTEM ENABLE (BOARD) (GENENB)					I/O: 3C3h		INDEX:–
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	▨	▨	▨	▨	▨	▨	a

a	R	VGA_ENABLE1	VGA Enable: Read back status of GENVS[0](0102)

	VIDEO SUBSYSTEM ENABLE (ADD ON) (GENENA)					I/O: 46E8h		INDEX:–
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	▨	▨	b	a	▨	▨	▨

a	W	VGA_ENABLE0	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.
b	W	GENVS ENABLE	GENVS[0] Enable: 0 = Disables I/O write to GENVS(0102) 1 = Enables I/O write to GENVS(0102)

Note:

In *mach64*VT, the decode of this register is optionally controlled by the PCI configuration space. Refer to *Chapter 6, PCI Configuration Registers*.

VGA Sequencer Registers

	SEQUENCER INDEX (SEQX)					I/O: 3C4h		INDEX:–
BITS	7	6	5	4	3	2	1	0
VT	▨	▨	▨	▨	▨	a		

a	R/W	SEQ_IDX[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.

	RESET (SEQ00)					I/O: 3C5h		INDEX: 00h
BITS	7	6	5	4	3	2	1	0
VT	▨	▨	▨	▨	▨	▨	b	a

a	R/W	SEQ_RST0b	Synchronous Reset Bit 0: 0 = Follows SEQ00[1] 1 = Allows the sequencer to run unless SEQ00[1] is zero
b	R/W	SEQ_RST1b	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

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.

	CLOCK MODE (SEQ01)					I/O: 3C5h		INDEX: 01h
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	▨	e	b	d	c	▨	a

a	R/W	SEQ_DOT8	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.
b,c	R/W	SEQ_SHIFT4 SEQ_SHIFT2	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
d	R/W	SEQ_PCLKBY2	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
e	R/W	SEQ_MAXBW	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.

Note:

To change this register, SEQ00[1 or 0] must first be logical zero.

	MAP MASK (SEQ02)			I/O: 3C5h			INDEX: 02h
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	▨	▨	▨	d	c	b	a

a	R/W	SEQ_MAP0_EN	Enable Map 0: 0 = Disables write access to memory map 0 1 = Enables write access to memory map 0
b	R/W	SEQ_MAP1_EN	Enable Map 1: 0 = Disables write access to memory map 1 1 = Enables write access to memory map 1
c	R/W	SEQ_MAP2_EN	Enable Map 2: 0 = Disables write access to memory map 2 1 = Enables write access to memory map 2
d	R/W	SEQ_MAP3_EN	Enable Map 3: 0 = Disables write access to memory map 3 1 = Enables write access to memory map 3

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, the CPU only needs to access maps 0 and 1; therefore, this register should have a value of 03h.
3. When in odd/even modes, the 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.

	CHARACTER MAP SELECT (SEQ03)			I/O: 3C5h			INDEX: 03h
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	▨	f	e	d	c	b	a

b,a,e	R/W	SEQ_FONTB[2:0]	Character Map Select B Bits 2:0
d,c,f	R/W	SEQ_FONTA[2:0]	Character Map Select A Bits 2:0

Notes:

1. The above register may seem unusual in the way that bits 1,0,4 and 3,2,5 are grouped. This is correct and the above notation is valid.
2. Extended memory SEQ04[1] must be logical in order to enable this select function; otherwise, the first 8K of map 2 is always selected.
3. Any changes made to this register take effect at the start of the next character line on the display.
4.

*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

	MEMORY MODE (SEQ04) **I/O: 3C5h** **INDEX: 04h**			
BITS	**3**	**2**	**1**	**0**
VT	c	b	a	▨

a	R/W	SEQ_256K	Extended Memory: Indicates 256K of video memory is present. Also enables character map selection in SEQ03.
b	R/W	SEQ_ODDEVEN	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 accessed for each CPU address.
c	R/W	SEQ_CHAIN	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 AO 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]

VGA DAC Registers

		DAC MASK (DAC_MASK)					I/O: 03C6h		INDEX:–
BITS		**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT		a							
a	R/W	DAC_MASK	Participating bit positions in the mask for DAC lookup are set to one.						

		DAC READ CURRENT COLOR INDEX (DAC_R_INDEX)					I/O: 03C7h		INDEX:–
BITS		**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT		a							
a	R/W	DAC_R_INDEX	The current read index for a DAC operation - increments after every third read of DAC_Data (03C9). 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 WRITE CURRENT COLOR INDEX (DAC_W_INDEX)					I/O: 03C8h		INDEX:–
BITS		**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT									
a	R/W	DAC_W_INDEX	The current write index for a DAC operation. Also see DAC_R_INDEX (03C7h)						

		DAC DATA (DAC_DATA)					I/O: 03C9h		INDEX:–
BITS		**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT		a							
a	R/W	DAC_DATA	DAC Data						

VGA Graphics Controller Registers

	GRAPHICS CONTROLLER INDEX (GRAX)					I/O: 3CEh		INDEX:–
BITS	7	6	5	4	3	2	1	0
VT	▨	▨	▨	▨	a			

a	R/W	GRPH_IDX[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.

	SET/RESET (GRA00)					I/O: 3CFh		INDEX: 00h
BITS	7	6	5	4	3	2	1	0
VT	▨	▨	▨	▨	d	c	b	a

a	R/W	GRPH_SET_RESET[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 one 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.
b	R/W	GRPH_SET_RESET[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.
c	R/W	GRPH_SET_RESET[2]	Set/Reset Map 2: 0 = All eight bits of buffer map 2 are to 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.
d	R/W	GRPH-SET-RESET[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.

ENABLE SET/RESET (GRA01) I/O: 3CFh INDEX: 01h

BITS	7	6	5	4	3	2	1	0
VT	▨	▨	▨	▨	d	c	b	a

a	R/W	GRPH_SET_RESET_ENA[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.
b	R/W	GRPH_SET_RESET_ENA[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.
c	R/W	GRPH_SET_RESET_ENA[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.
d	R/W	GRPH_SET_RESET_ENA[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), GRA003[3] is written to all eight bits of memory map 3.

Note:
This register has no effect on data source select when the video memory map write mode is 1, 2, or 3.

COLOR COMPARE (GRA02) I/O: 3CFh INDEX: 02h

BITS	7	6	5	4	3	2	1	0
VT	▨	▨	▨	▨	a			

a	R/W	GRPH_CCOMP[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 the CPU reads a one for a match in that bit position. If the Color Don’t Care bit for one map is logical zero, the latched data from the map is excluded from the compare, and only the remaining three bits are compared to generate the bus data

	DATA ROTATE (GRA03)					I/O: 3CFh		INDEX: 03h
BITS	7	6	5	4	3	2	1	0
VT	▨	▨	▨	b		a		

a	R/W	GRPH_ROTATE[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.
b	R/W	GRPH_FN_SEL[1:0]	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.

	READ MAP SELECT (GRA04)					I/O: 3CFh		INDEX: 04h
BITS	7	6	5	4	3	2	1	0
VT	▨	▨	▨	▨	▨	▨	a	

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

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.

	GRAPHIC MODE (GRA05)					I/O: 3CFh		INDEX: 05h
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	d		c	b	▨	a	

a	R/W	GRPH_WRITE_MODE[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 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].
b	R/W	GRPH_READ1	Read Mode: 0 = Byte-oriented: The CPU reads the memory map specified by the Read Map Select Register GRA04 unless SEQ04[3] is logical one (Chain). In the case where 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. The 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 the CPU as a one. This process is repeated for all eight pixels.
c	R/W	CGA_ODDEVEN	Odd/Even Addressing Enable Used to enable CGA emulation, this bit enables the 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.

(continued on next page)

	GRAPHIC MODE (GRA05) I/O: 3CFh INDEX: 05h							
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	d		c	b	▨	a	

d	R/W	GRPH_PACK GRPH_OES	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 → C3 M3D0 M3D1 M3D2 M3D3 M3D4 M3D5 M3D6 M3D7 → C4 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.

	GRAPHICS MISCELLANEOUS (GRA06) I/O: 3CFh INDEX: 06h							
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	▨	▨	▨	c		b	a

a	R/W	GRPH_GRAPHICS	Graphics/Alphanumeric Mode: 0 = Selects A/N (alphnumeric 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.
b	R/W	GRPH_ODDEVEN	Chain Odd Maps to Even: 1= CPU address bit AO is replaced by a higher order address bit. Even maps (0 and 2) are select when A0 = zero; odd maps are selected when A0 = one.
c	R/W	GRPH_ADRSEL[1:0]	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.

		COLOR DON'T CARE (GRA07)				I/O: 3CFh		INDEX: 07h
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	▨	▨	▨	▨	d	c	b	a

a	R/W	GRPH_XCARE[0]	Ignore Map 0.
b	R/W	GRPH_XCARE[1]	Ignore Map 1.
c	R/W	GRPH_XCARE[2]	Ignore Map 2.
d	R/W	GRPH_XCARE[3]	Ignore Map 3.

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 the CPU reads the data.

		BIT MASK (GRA08)				I/O: 3CFh		INDEX: 08h
BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VT	a							

a	R/W	GRPH_BMSK [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

This page intentionally left blank.

Appendix A
VGA BIOS Function Calls

VGA Controller

AH = 0 **; set video mode** (AL = video mode)

AL	Mode/Type	Resolution	Dim/Colour	Start Address
IBM Compatible Modes:				
00h	color/alpha	640x200	40x25/BW	B800h:0
01h	color/alpha	640x200	40x25/16	B800h:0
02h	color/alpha	640x200	80x25/BW	B800h:0
03h	color/alpha	640x200	80x25/16	B800h:0
04h	color/graphics	320x200	40x25/4	B800h:0
05h	color/graphics	320x200	40x25/BW	B800h:0
06h	color/graphics	320x200	80x25/BW	B800h:0
07h	mono/alpha	720x350	80x25/BW	B000h:0
0Dh	color/graphics	320x200	40x25/16	A000h:0
0Eh	color/graphics	640x200	80x25/16	A000h:0
0Fh	mono/graphics	640x350	80x25/BW	A000h:0
10h	color/graphics	640x350	80x25/16	A000h:0
11h	color/graphics	640x480	80x30/BW	A000h:0
12h	color/graphics	640x480	80x30/16	A000h:0
13h	color/graphics	320x200	80x25/256	A000h:0

AL	MODE/TYPE	RESOLUTION	DIM/COLOR	START ADDRESS
ATI Enhanced Modes:				
21h	color/alpha	800x400	100x25	B800h:0
22h	color/alpha	800x480	100x30	B800h:0
23h	color/alpha	1056x200	132x25/16	B800h:0
33h	color/alpha	1056x352	132x44/16	B800h:0
55h	color/graphics	1024x768	128x48/16	A000h:0
61h	color/graphics	640x400	80x25/256	A000h:0
62h	color/graphics	640x480	80x30/256	A000h:0
63h	color/graphics	800x600	100x42/256	A000h:0
64h	color/graphics	1024x768	128x48/256	A000h:0
6Ah	color/graphics	800x600	100x42/16	A000h:0

AH = 1 **; set cursor type**

CH = start line of cursor
CL = end line of cursor
CX = 1F00h to turn off cursor

AH = 2 **; set current cursor position**

BH = page number of the desired page
DH, DL = row and column of cursor

AH = 3 **; read current cursor position at the specified page**

BH = page number of the desired page
On Exit:
CH, CL = cursor type
DH, DL = row, column of cursor at the specified page

AH = 4 **; read current light pen position** (VGA does not support light pen)

AH = 5 **; select active display page**

AL = page number to be active

AH = 6 **; scroll active page up**

AL = number of lines to be scrolled
= 0 ; blanks the whole window
BH = attribute of blanked line
CH, CL = row, column of upper left hand corner of scrolling window
DH, DL = row, column of lower right hand corner of scrolling window

AH = 7 **; scroll active page down**

AL = number of lines to be scrolled
= 0 ; blanks the whole window
BH = attribute of blanked line
CH, CL = row, column of upper left hand corner of scrolling window
DH, DL = row, column of lower right hand corner of scrolling window

AH = 8 **; read character/attribute at current active cursor position**

BH = page number of the desired page
On Exit:
AL = character
AH = attribute (for text mode only)

AH = 9 **; write character/attribute at current cursor position of a specified page**

AL = character to be written
BL = attribute of character
BH = page number
CX = count of character to write

AH = 0Ah **; write character at current cursor position of a specified page**

AL = character to be written
BH = page number
CX = count of character to write

AH = 0Bh **; set color palette, valid for modes 4 and 5 only**

BH = 0 ; selects the background color
BL = color value used with that color id
BH = 1 ; selects the palette to be used
BL = 0 ; palette value is GREEN(1)/RED(2)/BROWN(3)
= 1 ; palette value is CYAN(1)/MAGENTA(2)/WHITE(3)

AH = 0Ch **; write dot (graphics mode)**

BH = page number
DX, CX = row, column of dot position
AL = color value of dot (if bit 7 of AL is ON, the color value will XOR with the current value of the dot)

AH = 0Dh **; read dot (graphics mode)**

BH = page number
DX, CX = row, column of dot position
On Exit
AL = color value of dot

AH = 0Eh **; write teletype to active page**

AL = character to write
BL = foreground color in graphics mode

AH = 0Fh **; return current video setting**

On Exit:
AL = current mode
AH = number of column (in characters) on screen
BH = current active display page

AH = 10h **; set palette registers**

```
AL  =  0    ; set individual palette register
            BL  =  palette register
            BH  =  palette value
AL  =  1    ; set overscan register
            BH  =  palette value
AL  =  2    ; set all palette and overscan registers
            ES:DX  =  pointer to palette value table (17 bytes long), bytes 0 - 15 are
                      palette values for 16 palette registers, byte 16 is palette value
                      for the overscan register
AL  =  3    ; toggle between intensity/blinking bit
            BL  =  0   ; set intensity on
            BL  =  1   ; set blinking on
AL  =  7    ; read individual palette register
            BL  =  palette register
            On Exit:
            BH  =  palette value
AL  =  8    ; read overscan register
            On Exit:
            BH  =  overscan value
AL  =  9    ; read all palette and overscan registers
            ES:DX  =  pointer to 17-byte buffer
            On Exit:
            ES:DX  =  pointer to palette value table (17 bytes long), bytes 0 - 15 are
                      palette values for 16 palette registers, byte 16 is palette value
                      for the overscan register
AL  = 10h   ; set a color register
            BX  =  color register
            DH  =  red value
            CH  =  green value
            CL  =  blue value
AL  = 12h   ; set a block of color registers
            BX  =  first color register to be set
```

```
               CX  =   total number of color registers to be set
               ES:DX  =  pointer to table of color register values in red, green, blue, red,
                         green, blue ,... format
 AL  = 13h     ; set color pages (only valid for 16 color modes)
               BL   = 0    ; select color page mode
               BH   = 0    ; select 4 pages of 64 color registers each
               BH   = 1    ; select 16 pages of 16 color registers each
               BL   = 1    ; select color page
               BH   =   color page number
 AL  = 15h     ; read a color register
               BX  =   color register
               On Exit:
               DH  =   red value
               CH  =   green value
               CL  =   blue value
 AL  =17h      ; read a block of color registers
               BX  =   first color register to be set
               CX  =   total number of color registers to be set
               ES:DX  =  pointer to buffer to store the color register values
               On Exit:
               ES:DX  =  pointer to table of color register values in red, green, blue, red,
                         green, blue, ..., format
AL   =18h      ;update DAC mask register
               BL  =   new mask value
AL   =19h      ;read DAC mask register
               BL  =   value read from DAC mask register
AL   = 1Ah     ; read current color page information
               BL  =   current color page mode
               BH  =   current color page
 AL  = 1Bh     ; change color values to gray shades
               BX  =   first color register to be changed
               CX  =   total number of color registers to be changed
```

AH=11h **; character generator routines**

```
AL  =  00      ; load user specified character set
               ES:BP  =  pointer to character table
               CX  =   number of characters to be stored
               DX  =   character of offset into current table
               BL  =   block to load
               BH  =   bytes per character

AL  =  01      ; load 8x14 character set
               BL  =   block to load
AL  =  02      ; load 8x8 character set
               BL  =   block to load
AL  =  03      ; set block specifier
               BL  =   character generator block specifier
AL  =  04      ; load 8x16 character set
               BL  =   block to load
```

Note that the following functions, AL = 1?h, are similar to the functions AL = 0?h, except that with AL=1?h, the number of rows on the screen is recalculated.

```
AL  =  10h   ; load user specified character set
             ES:BP  =   pointer to character table
             CX  =   number of characters to be stored
             DX  =   character of offset into current table
             BL  =   block to load
             BH  =   bytes per character
AL  =  11h   ; load 8x14 character set
             BL  =   block to load
AL = 12h     ; load 8x8 character set
             BL  =   block to load
AL = 14h     ; load 8x16 character set
             BL  =   block to load
AL = 20h     ; update alternative character generator pointer (INT 1F)
             ES:BP  =   pointer to table
AL = 21h     ; update alternative character generator pointer (INT 43)
             ES:BP  =   pointer to table
             CX  =   bytes per character
             BL  =   row specifier
                 =   0   ; DL = rows
                 =   1   ; rows = 14
                 =   2   ; rows = 25
                 =   3   ; rows = 43
AL = 22h     ; update alternative character generator pointer (INT 43) with the 8x14
             ; character generator in ROM
AL = 23h     ; update alternative character generator pointer (INT 43) with the 8x8
             ; character generator in ROM
AL = 24h     ; update alternative character generator pointer (INT 43) with the 8x16
             ; character generator in ROM
AL = 30h     ; return EGA character generator information
             BH  =   0   ; return current INT 1F pointer
                 =   1   ; return current INT 43 pointer
                 =   2   ; return pointer to 8x14 character generator
                 =   3   ; return pointer to 8x8 character generator (lower)
                 =   4   ; return pointer to 8x8 character generator (upper)
                 =   5   ; return pointer to alternate 9x14 alpha
                 =   6   ; return pointer to 8x16 character generator
                 =   7   ; return pointer to alternate 9x16 alpha
             On Exit:
             ES:BP  =   pointer to table as requested
             CX  =   points (pixel column per char)
             DL  =   rows (scan line per char)
```

AH = 12h ; return current EGA settings/print screen routine selection

```
BL = 10h    ; return EGA information
            On Exit:
            BH  =  0   ; color mode in effect
                =  1   ; monochrome mode in effect
            BL  =  3   ; 256k video memory installed (always return 3)
            CH  =  simulated value of feature bits
            CL  =  simulated EGA/VGA dip switch setting
BL = 20h    ; select alternate print screen routine for EGA graphics mode
BL = 30h    ; select number of scan lines for alpha modes
            AL  =  0   ; 200 scan lines
                =  1   ; 350 scan lines
                =  2   ; 400 scan lines
            On Exit:
            AL = 12h   ; function supported
BL = 31h    ; default palette loading during mode set
            AH  =  0
            AL  =  0   ; enable
                =  1   ; disable
            On Exit:
            AL = 12h   ; function supported
BL = 32h    ; video controller
            AL  =  0   ; enable video controller
                =  1   ; disable video controller
            On Exit:
            AL = 12h   ; function supported
BL = 33h    ; summing of color registers to gray shades
            AL  =  0   ; enable summing
                =  1   ; disable summing
            On Exit:
            AL = 12h   ; function supported
BL = 34h    ; cursor emulation
            AL  =  0   ; enable cursor emulation
                =  1   ; disable cursor emulation
            On Exit:
            AL = 12h   ; function supported
BL = 36h    ; video screen on/off
            AL  =  0   ; video screen on
                =  1   ; video screen off
            On Exit:
            AL = 12h   ; function supported
 BX=5506h   ; VGAWONDER BIOS extension
            AL  =  video mode
            BP   =  0FFFFh
            DI  =  0
            SI  =  0
            On Exit:
            if BP is not equal to 0FFFFh
                then ES:BP = pointer to parameter table
            if SI is not equal to 0
                then ES:SI = pointer to parameter table supplement
```

```
AH = 13h      ; write string to specified page
              ES:BP = pointer to string
              CX  =  length of string
              BH  =  page number
              DH,DL =    starting row and column of cursor in which the string is placed
              AL  =  0   ; cursor is not moved
                         BL  =  attribute
                         string = (char, char, char, char, ...)
              AL  =  1   ; cursor is moved
                         BL  =  attribute
                         string = (char, char, char, char, ...)
              AL  =  2   ; cursor is not moved
                         string = (char, attr, char, attr, ...)
              AL  =  3   ; cursor is moved
                         string = (char, attr, char, attr, ...)

AH=1Ah        ; read display combination code
              AL  =  0   ; read current display combination information
                         On Exit
                         AL  =  1Ah
                         BL  =  current active display code
                         BH  =  alternate display code
                         Display codes
                         00 - No display
                         01 - MDA mode
                         02 - CGA mode
                         04 - EGA in color mode
                         05 - EGA in monochrome mode
                         07 - VGA with analog monochrome monitor
                         08 - VGA with analog color monitor
              AL  =  1   ;set display combination information
                         BL  =  active display
                         BH  =  inactive display
                         On Exit
                         AL  =  1Ah

AH=1Bh        ; return VGA functionality and state information
              BX  =  0   ;
                         ES:DI  =   pointer to buffer used to store the functionality and state
                                    information (minimum 64 bytes)
                         On Exit:
                         AL   =  1Bh
                         ES:DI  =   pointer to buffer with functionality and state information
                         [DI+00h] word = offset to static functionality information
                         [DI+02h] word = segment to static functionality information
                         [DI+04h] byte = current video mode
                         [DI+05h] word = character columns on screen
                         [DI+07h] word = page size in number of bytes
                         [DI+09h] word = starting address of current page
```

```
[DI+0Bh] word = cursor position for eight display pages
[DI+1Bh] word = current cursor type
[DI+1Dh] byte = current active page
[DI+1Eh] word = current CRTC address
[DI+20h] byte = current 3x8 register setting
[DI+21h] byte = current 3x9 register setting
[DI+22h] byte = number of character rows on screen
[DI+23h] word = number of scan lines per character
[DI+25h] byte = active display combination code
[DI+26h] byte = alternate display combination code
[DI+27h] word = number of colors supported in current mode
[DI+29h] byte = number of pages supported in current mode
[DI+2Ah] byte = 0   ; 200 scan lines in current mode
              = 1   ; 350 scan lines in current mode
              = 2   ; 400 scan lines in current mode
              = 3   ; 480 scan lines in current mode
[DI+2Bh] byte = Reserved
[DI+2Ch] byte = Reserved
[DI+2Dh] byte = miscellaneous state information
          bits 7, 6 = Reserved
          bit 5 = 0   ; background intensity
                = 1   ; blinking
          bit 4 = 1   ; cursor emulation active
          bit 3 = 1   ; mode set default palette loading disabled
          bit 2 = 1   ; monochrome display attached
          bit 1 = 1   ; summing active
          bit 0 = 1   ; all modes on all display active
[DI+2Eh] byte = Reserved
[DI+2Fh] byte = Reserved
[DI+30h] byte = Reserved
[DI+31h] byte = 3; 256Kb of video memory available
[DI+32h] byte = save pointer information
          bits 7, 6 = Reserved
          bit 5 = 1   ; DCC extension active
          bit 4 = 1   ; palette override active
          bit 3 = 1   ; graphics font override active
          bit 2 = 1   ; alpha font override active
          bit 1 = 1   ; dynamic save area active
          bit 0 = 1   ; 512 character set active
[DI+33h] 13 bytes = Reserved
```

static functionality table format

```
          0 - function not supported
          1 - supported function

[00h] byte = supported video mode
          bit 7 = mode 07h
          bit 6 = mode 06h
          bit 5 = mode 05h
          bit 4 = mode 04h
          bit 3 = mode 03h
          bit 2 = mode 02h
```

bit 1 = mode 01h
bit 0 = mode 00h
[01h] byte = supported video mode
bit 7 = mode 0Fh
bit 6 = mode 0Eh
bit 5 = mode 0Dh
bit 4 = mode 0Ch
bit 3 = mode 0Bh
bit 2 = mode 0Ah
bit 1 = mode 09h
bit 0 = mode 08h
[02h] byte = supported video mode
bits 7 to 4= Reserved
bit 3 = mode 13h
bit 2 = mode 12h
bit 1 = mode 11h
bit 0 = mode 10h
[03h] to [06h] = Reserved
[07h] = scan lines available in text modes
bits 7 to 3 = Reserved
bit 2 = 400 scan lines
bit 1 = 350 scan lines
bit 0 = 200 scan lines
[08h] = number of character fonts available in text modes
[09h] = maximum number of character fonts that can be active in text modes
[0Ah] byte = miscellaneous functions
bit 7 = color paging
bit 6 = color palette (color register)
bit 5 = EGA palette
bit 4 = cursor emulation
bit 3 = default palette loading when mode set
bit 2 = character font loading
bit 1 = color palette summing
bit 0 = all modes supported on all displays
[0Bh] = scan lines available in text modes
bits 7 to 4 = Reserved
bit 3 = DCC supported
bit 2 = background intensity/blinking control
bit 1 = save/restore supported
bit 0 = light pen supported
[0Ch] to [0Dh] = Reserved
[0Eh] = save pointer functions
bits 7 to 6 = Reserved
bit 5 = DCC extension supported
bit 4 = palette override
bit 3 = graphics font override
bit 2 = alpha font override
bit 1 = dynamic save area
bit 0 = 512-character set
[0Fh] = Reserved

AH=1Ch ; save and restore video state

```
AL  =  0    ; return video save state buffer size requirement
            CX  =  requested states
                   bit 0 = video hardware state
                   bit 1 = video BIOS data area
                   bit 2 = video DAC state and color registers
            On Exit:
            AL  =  1Ch
            BX  =  number of 64 bytes block required for the states requested in CX
AL  =  1    ; save video state
            CX  =  requested states (see AL=0)
            ES:BX=pointer to buffer to store the video states information
            On Exit:
            AL = 1Ch
AL  =  2    ; restore video state
            CX  =  requested states (see AL=0)
            ES:BX = pointer to buffer with previous saved video states information
            On Exit:
            AL = 1Ch
```

This page intentionally left blank.

Appendix B
Programming PLL Registers

Introduction

The PLL registers on the next page are accessed indirectly through the CLOCK_CNTL register on *page 4-38*. Example reads and writes of the PLL registers are given below. The address CLOCK_CNTL0 represents bits 7:0, CLOCK_CNTL1 bits 15:8, and CLOCK_CNTL2 bits 23:16.

PLL Register Read

```
iow8 CLOCK_CNTL1 PLL_ADDR    ; PLL address to read (PLL_WR_EN = 0)
ior8 CLOCK_CNTL2 PLL_DATA    ; data is put into variable PLL_DATA
```

PLL Register Write

```
iow8 CLOCK_CNTL1 PLL_ADDR | PLL_WR_EN; PLL address to write and PLL_WR_EN = 1
iow8 CLOCK_CNTL2 PLL_DATA            ; PLL data to write
```

32 bit I/O write:

```
iow32 CLOCK_CNTL CLOCK_SEL | PLL_ADDR | PLL_WR_EN | PLL_DATA
```

Clock Sources

All clock signals in *mach64*VT/3D RAGE are derived from three master clocks — Bus Clock (CPUCLK), MCLK and VCLK. MCLK and VCLK each has four different source choices. These include internal PLLs (PLLMCLK and PLLVCLK), external clock pins (CPUCLK and EXTFREQ0 or EXTFREQ1), XTALIN pin and the PLL reference (PLLREFCLK), which XTALIN/reference divider setting. When RESETb goes active, all clocks switch to using CPUCLK as their source. After reset, either the test vectors will select external sources or the BIOS will select internal sources.

PLL Registers

PLL Registers				
Addr	**Register Name**	**Field**	**Bits**	**Function**
0	Reserved			
1	PLL_MACRO_CNTL			Controls to analog PLL macro (default = D4h)
		PLL_PC_GAIN	2:0	Charge-pump gain setting
		PLL_VC_GAIN	4:3	VCGEN gain setting
		PLL_DUTY_CYC	7:5	Duty cycle control for pixel clock PLL
2	PLL_REF_DIV		7:0	Reference divider setting (default = 36h)
3	PLL_GEN_CNTL			MCLK and general control (default = 4Fh)
		PLL_OVERRIDE	0	1 : Power-down PLL
		PLL_MRESET	1	1 : Reset MCLK PLL
		OSC_EN	2	1 : Oscillator enable
		EXT_CLK_EN	3	1 : Force DCLK pin output to tri-state
		MCLK_SRC_SEL	6:4	000 : MCLK_SRC = PLLMCLK 001 : MCLK_SRC = PLLMCLK / 2 010 : MCLK_SRC = PLLMCLK / 4 011 : MCLK_SRC = PLLMCLK / 8 100 : MCLK_SRC = CPUCLK 101 : MCLK_SRC = DCLK 110 : MCLK_SRC = PLLREFCLK 111 : MCLK_SRC = XTALIN
		(reserved)	7	Used in CT and LT
4	MCLK_FB_DIV		7:0	MCLK feedback divider (default = 97h, 40 MHz)
5	PLL_VCLK_CNTL			Pixel clock control (default = 04h)
		VCLK_SRC_SEL	1:0	00 : VCLK = CPUCLK 01 : VCLK = DCLK 10 : VCLK = XTALIN 11 : VCLK = PLLVCLK/(VCLK Post Divider)
		PLL_PRESET	2	Reset VCLK PLL
		VCLK_INVERT	3	Invert VCLK to get opposite duty cycle
		ECP_DIV	5:4	00 : ECP = VCLK 01 : ECP = VCLK / 2 10 : ECP = VCLK / 4 11 : reserved
		ERATE_GT_XRATE	6	Set when ERATE greater than XRATE 0 = Normal 1 = Always request
		SCALER_LOCK_EN	7	Set when scaler requires exclusive memory access 0 = Normal 1 = Lock out memory requesters of lower priority than scaler
6	VCLK_POST_DIV			Post dividers for VCLK 0–3 (default = 6Ah)
		VCLK0_POST	1:0	Post divider for VCLK setting 0
		VCLK1_POST	3:2	Post divider for VCLK setting 1
		VCLK2_POST	5:4	Post divider for VCLK setting 2
		VCLK3_POST	7:6	Post divider for VCLK setting 3
7	VCLK0_FB_DIV		7:0	Feedback divider for VCLK 0 (default = BEh)

(continued on next page)

PLL Registers				
Addr	**Register Name**	**Field**	**Bits**	**Function**
8	VCLK1_FB_DIV		7:0	Feedback divider for VCLK 1 (default = D6h)
9	VCLK2_FB_DIV		7:0	Feedback divider for VCLK 2 (default = EEh)
10	VCLK3_FB_DIV		7:0	Feedback divider for VCLK 3 (default = 88h)
11	PLL_XCLK_CNTL			Extended control of MCLK (default = 00h)
		XCLK_MCLK_RATIO	1:0	Division of XCLK and MCLK from MCLK_SRC 00 : XCLK = MCLK_SRC, MCLK = MCLK_SRC 01 : XCLK = MCLK_SRC/2, MCLK = MCLK_SRC/4 10 : XCLK = MCLK_SRC/2, MCLK = MCLK_SRC/3 11 : XCLK = MCLK_SRC/3, MCLK = MCLK_SRC/4
		MFB_TIMES_4_2b	2	Selects ratio of MCLK_FB_DIV to effective feedback value 0 = PLLMCLK feedback = 2 * MCLK_FB_DIV 1 = PLLMCLK feedback = 4 * MCLK_FB_DIV
		XCLK_MCLK_TST	3	Sets XCLK and MCLK to their respective tester clocks 0 = Normal 1 = Tester mode (override XCLK/ MCLK ratio)
		(reserved)	7:4	
12	PLL_FCP_CNTL			Controls FCP clock to video port (default = 41h)
		FCP_POST_DIV	3:0	FCP post divider setting 0 : FCP_SRC is off other N : FCP_SRC = FCP_REF/N
		FCP_SRC_SEL	6:4	000 : FCP_REF = PLLMCLK 001 : FCP_REF = PLLMCLK / 2 010 : FCP_REF = PLLMCLK / 4 011 : FCP_REF = PLLMCLK / 8 100 : FCP_REF = CPUCLK 101 : FCP_REF = DCLK 110 : FCP_REF = PLLREFCLK 111 : FCP_REF = XTALIN
		DCLK_BY2_EN	7	0 : DCLK = VCLK in VGA Mode 13 1 : DCLK = VCLK/2 in VGA Mode 13
13	VFC_CNTL (VT-A4) (VT-A3 = reserved)			Control of VESA feature connector (default = 00h)
		DCLK_INVb	0	0 : Invert DCLK output, DCLK = not VCLK 1 : DCLK = VCLK
		DCLKBY2_EN	1	DCLK frequency select for VGA Mode 13. No effect in other display modes. 0 : DCLK 25 MHz in VGA Mode 13 1 : DCLK 12.5 MHz in VGA Mode 13
		VFC_2PHASE	2	Selects VFC operation mode in 15 and 16 bpp. No effect in other display modes. 0 : Single phase operation in 15 and 16 bpp 1 : Two phase operation in 15 and 16 bpp
		VFC_DELAY	4:3	PIXEL and BLANKB hold adjustment 00 : Least delay to 11 : most delay
		(reserved)	7:5	

(continued on next page)

PLL Registers				
Addr	**Register Name**	**Field**	**Bits**	**Function**
14	PLL_TEST_CRTL			PLL test mode control (forced to 00h when not in PLL test mode from GEN_TEST_CTRL register).
		TST_SRC_SEL	4:0	Selects source of PLL test clock.
		TST_DIVIDERS	5	1 : Open reference and feedback dividers for test
		PLL_MASK_READb	6	0 : Mask PLL_TEST_COUNT(2:0) and disable test output pin
		(reserved)	7	
15	PLL_TEST_COUNT		7:0	PLL test mode counter (read only, no default). Writing any value will reset to 00h.

Notes:

1. PLL_MACRO_CNTL settings control gain and duty cycle of analog PLL's. Gain bits affect lock and jitter of PLL's. This register should only be adjusted by the BIOS.
2. The reference divider setting must be in the range of 2h to FFh.
3. Oscillator enable is only supported in NEC foundry due to limitations in oscillator macro cells. Oscillator will always run in other foundries, no matter how this bit is set.
4. The effective feedback divider for PLLMCLK is controlled by MFB_TIMES_4_2b and is either 4*MCLK_FB_DIV or 2*MCLK_FB_DIV. The effective feedback divider for PLLMCLK should not be set below 100h (i.e., 80h or 40h). The effective feedback divider for PLLVCLK is always twice the current VCLKx_FB_DIV value. The suggested range for VCLK feedback dividers is 80h to FFh. Setting the effective feedback divider below the suggested limits results in coarser control of output frequency and possibility of clock jitter. Feedback dividers below 02h will not function.
5. Pixel clock (VCLK) post-divider values are: 00=divide-by-1; 01=divide-by-2; 10=divide-by-4; 11=divide-by-8.
6. All clock sources can be programmed to exceed the frequency limitations of the hardware. Do not attempt to program the PLL registers without a good understanding of the frequency limitations of all clock nets.
7. PLL_TEST_CTRL and PLL_TEST_COUNT are used only during manufacturing tests of analog PLL's.

External Clock Support

The external clock sources are supported by *mach64*VT, primarily for testing but also on a board if required. The control signals for the external clock chip are multiplexed on the feature connector pins. The feature connector may not be used when the external clock sources are active.

Switching to external clocks is done as follows:

1. Disable the feature connector (DAC_FEA_CON_EN@DAC_CNTL, defaults to disabled).
2. Set EXT_CLK_EN@PLL_GEN_CNTL = 1 to enable external clock support pins (defaults to high).
3. Make sure the external clock signals are being driven into the chip.
4. Set MCLK_SRC_SEL@PLL_GEN_CNTL = 101 for EXTFREQ1 as MCLK. Also set VCLK_SRC_SEL@PLL_VCLK_CNTL = 01 for EXTFREQ0 as VCLK.

Switching to internal clocks at boot time is done as follows:

1. Program reference, feedback and VCLK post dividers to the desired settings.
2. Write to PLL_GEN_CNTL, setting PLL_OVERRIDE = 0, PLL_MCLK_RST = 0 and OSC_EN = 1.
3. Write to PLL_VCLK_CNTL, setting PLL_VCLK_RST = 0.
4. Allow 5 ms for internal PLL to lock frequencies.
5. Set MCLK_SRC_SEL@PLL_GEN_CNTL = 001.
6. Set VCLK_SRC_SEL@PLL_VCLK_CNTL = 11.

Frequency Limits

The design of *mach64*VT imposes the following limits on the clock source frequencies:

- MCLK may not exceed 68 MHz or the limit imposed by memory type.
- VCLK is limited by the current display mode:
    – In VGA, it may not exceed 80 MHz.
    – In 4 bpp and 8 bpp, it may be up to 135 MHz.
    – In 15 to 32 bpp, it may not exceed 80 MHz.
- CPUCLK may not exceed 33 MHz.

The clock going out the feature connector (DCLK) may not exceed 40 MHz according to the VESA specification. In practice, a higher limit (possibly 80 MHz) will be attempted. When VCLK is set to exceed the limit, then DAC_FEA_CON_EN@DAC_CNTL must be set low to turn off the feature connector.

Frequency Synthesis Description

To generate a specific output frequency, the reference (M), feedback (N), and post dividers (P) must be loaded with the appropriate divide-down ratios. The internal PLLs for CT and ET are optimized to lock to output frequencies in the range from 135 MHz to 68 MHz. The PLLs for other members of the *mach64*VT family are optimized to lock with output frequencies from 100 MHz to 200 MHz. Setting the PLLs to lock outside these ranges can result in increased jitter or total malfunction (no lock).

The PLLREFCLK lower limit is found based on the upper limit of the PLL lock range (e.g., 135 MHz) and the maximum feedback divider (255) as follows:

$$\text{Minimum PLLREFCLK} = 135 \text{ MHz} / (2 * 255) = 265 \text{ kHz}$$

This is then used to find the reference divider based on the XTALIN frequency.

XTALIN is normally 14.318 MHz and the maximum reference divider M is found by:

M = Floor[14.318 MHz / 265 kHz] = 54 (the Floor function means round down)

Using the maximum reference divider allowed (in this case is 54) ensures the best clock step resolution. However, lower reference dividers might be used to improve clock jitter.

Feedback dividers (N) should kept in the range 80h to FFh. The effective feedback divider is twice the register setting due to the structure of the internal PLL. The post divider (P) may be either 1, 2, 4, or 8.

To determine the N and P values to program for a target frequency, follow the procedure below (where R is the frequency of XTALIN and T is the target frequency):

1. Calculate the value of P. Find the value of Q from the equation below and use it to find P in the following table:

$$Q = (T * M) / (2 * R)$$

Q Range	Result
more than 255	M too big
127.5 to 255	P = 1
63.5 to 127.5	P = 2
31.5 to 63.5	P = 4
16 to 31.5	P = 8
less than 16	M too small

2. Calculate the value of N by using the value of P obtained in step 1. N is given by:

$$N = Q * P$$

   The result N is rounded to the nearest whole number.

3. Determine the actual frequency. Given P and the rounded-off N, the actual output frequency is found by:

$$Output\_Frequency = (2 * R * N) / (M * P)$$

For example:

If R = 14.318 MHz and M = 54, then Q = 75.43 (if the desired frequency is 40MHz). The table indicates P = 2 for this Q value. The calculation of N = Q*P gives 150.85 and rounding up gives N = 151. The final output frequency is therefore 40.04 MHz.

The maximum frequency that can be synthesized is the upper limit of PLL lock range for the specific version of *mach64*VT. It may be 135, 160, 200, or 240 MHz. The minimum frequency that can be synthesized depends on the largest post divider available. For VCLK, P = 8 is always available and minimum VCLK = (2*R*128)/(M*8). For MCLK, post divider settings of 4 and 8 are not available on some versions of the controller. The minimum frequency setting for MCLK is limited to the correspondingly higher values for these controllers.

Sample divider settings for typical Pixel and Memory clock frequencies when R = 14.318 MHz and M = 54:

Target Freq. (MHz)	Post Divider P	Feedback Register N	Feedback Register N	Actual Freq. (MHz)	Percent Error (%)
135	1	255	FFh	135.23	0.17
126	1	238	EEh	126.21	0.17
110	1	207	CFh	109.77	0.21

Target Freq. (MHz)	Post Divider P	Feedback Register N	Feedback Register N	Actual Freq. (MHz)	Percent Error (%)
100	1	189	BDh	100.23	0.23
92.4	1	174	AEh	92.27	0.14
80	1	151	97h	80.08	0.1
75	1	141	8Dh	74.77	0.31
65	2	245	F5h	64.96	0.06
56.6	2	213	D5h	56.48	0.21
50.2	2	189	BDh	50.11	0.19
49.95	2	188	BCh	49.85	0.2
45	2	170	AAh	45.08	0.18
44.95	2	170	AAh	45.08	0.29
40	2	151	97h	40.04	0.1
36	2	136	88h	36.06	0.17
32.97	4	249	F9h	33.01	0.12
32	4	241	F1h	31.95	0.16
31.5	4	238	EEh	31.55	0.16
28.322	4	214	D6h	28.37	0.17
25.175	4	190	BEh	25.19	0.06

Duty Cycle Control

The DAC clock (VCLK) is the fastest clock on a *mach64*VT chip. When displayed in 1280x1024 or higher resolutions, VCLK will exceed 100 MHz. The DAC circuitry is sensitive to the duty cycle of VCLK in this range. Duty cycle adjustment for VCLK is available through PLL_DUTY_CYC@PLL_MACRO_CNTL and VCLK_INVERT@PLL_VCLK_CNTL.

The optimal settings for the duty cycle control bits have been determined by ATI during testing under extreme conditions of temperature and voltage. The BIOS sets the proper values for each version of *mach64*VT. There should be no need to change these settings.

PLL Gain Settings

The internal PLLs have two settings that affect their gain characteristics. These are set by PLL_PC_GAIN and PLL_VC_GAIN in the PLL_MACRO_CNTL register. They will affect optimal lock ranges and jitter characteristics. ATI has determined the optimal settings for these bits under extreme operating conditions. The BIOS sets these bits to optimal values for each version of *mach64*VT. There should not be any need to modify these values.

This page intentionally left blank.

Index

User Response Form

In our continuing effort to improve our products and documentation, ATI Technologies Inc. is anxious to obtain your feedback on this manual. Using a scale of 1 to 5, with 5 representing the most favorable, and 1 the least favorable, please circle the number that best reflects your opinion. Everyone who completes and returns this questionnaire will receive an ATI souvenir (e.g. golf shirt, mug etc.) depending on availability.

Name and Organization ______________________________

Title ______________________________ **Years in position** ________

Manual _*mach64* Register Reference Guide – ATI-264VT & 3D RAGE_ **Release No.** ______

How do you rate the manual, generally?	*1*	*2*	*3*	*4*	*5*
Is the technical level of the manual appropriate?	*1*	*2*	*3*	*4*	*5*
Are operating instructions clear and complete?	*1*	*2*	*3*	*4*	*5*
Are terms and concepts explained clearly?	*1*	*2*	*3*	*4*	*5*
Is the manual well organized?	*1*	*2*	*3*	*4*	*5*
Are the tables easy to understand?	*1*	*2*	*3*	*4*	*5*
Is the artwork clear and easy to understand?	*1*	*2*	*3*	*4*	*5*
How helpful is the index?	*1*	*2*	*3*	*4*	*5*
How does this manual compare with other, similar manuals you have used?	*1*	*2*	*3*	*4*	*5*

Comments:

What did you like best about this manual ?______________________________

____

What did you like least about this manual ? ______________________________

____

Are there any errors/omissions in the manual?______________________________

____

THANK YOU FOR YOUR COMMENTS! Please mail to: ATI Technologies Inc., 33 Commerce Valley Drive East, Thornhill, Ontario, Canada L3T 7N6. You may also fax this to (905) 882-2620 (Attention: Technical Publications)

ATI TECHNOLOGIES INC.

Perfecting the PC

Printed in Canada