3D RAGE™LT PRO

Register Reference Guide

(Restricted Version)

Technical Reference Manual

P/N: RRG-G03300 Rev. 2.01X

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

Record of Revisions

Release	Date	Description of Changes
0.01-0.03	Mar-May 97	Pre-production, Preliminary
1.00-1.05	Jun 97-Jan 98	For the production chip, new registers added, major change of format, chapters re-organized. Final
2.00	Jan 98	No longer an Addendum, full document
2.01	Apr 98	General updates

Related Manuals

3D RAGE LT PRO series

- 3D RAGE™ LT PRO Register Reference Guide (RRG-GO3300)
- 3D RAGE™ LT PRO Graphics Controller Specifications (GCS-C03300)

Table of Contents

Chapter 1: Introduction

Chapter 2: Overview and Memory Mapping

Chapter 3: Cross Reference Tables

Chapter 4: Display and Configuration

Chapter 5: GUI Draw Engine

Chapter 6: Host Interface

Chapter 7: VGA-Compatible Registers

Chapter 8: LCD Panel

Chapter 1
Introduction

1.1 Scope

This document serves as a ***restricted*** register reference guide to the RAGE LT PRO. Certain registers mentioned in Chapters 2 and 3 — registers related to 3D, Multimedia, and TV Out — have been omitted. If you require information on these, please contact your ATI representative.

1.2 Contents

This document is organized into 8 chapters.

Chapter 1 outlines the contents of this document and explains the notations and conventions used throughout.

Chapter 2 gives an overview of the registers and describes how they are memory and I/O mapped.

Chapter 3 lists the accelerator registers in two tables by register name (mnemonic) and by address. For ease of reference, both tables feature a page number column (hypertext linked in the online document) which helps to locate quickly any of the registers.

Chapter 4 describes the 2D Accelerator registers which include:

- Setup and Control registers
- Accelerator CRTC and DAC registers

Chapter 5 describes the GUI Engine registers.

Chapter 6 describes the Host Interface registers which include:

- PCI Configuration Space registers
- System Bus Mastering registers

  - Draw Engine Bus Mastering registers
  - AGP registers

- *Chapter 7* details the VGA-Compatible registers.
- *Chapter 8* describes the LCD Panel registers

1.3 Notations and Conventions

1.3.1 Mnemonics

Mnemonics are expressed in upper-case and are used throughout this document in place of hardware register names and field names. The naming conventions for registers and bit fields are as indicated below:

- REGISTER_MNEMONIC

  For example, CONFIG_CHIP_ID is the mnemonic for the Configuration Chip ID register.

- REGISTER_MNEMONIC[Bit_Numbers] or
- FIELD_NAME@REGISTER_MNEMONIC

  For example, CONFIG_CHIP_ID[15:0] refers to the bit field that occupies bit positions 0 through 15 within this register.

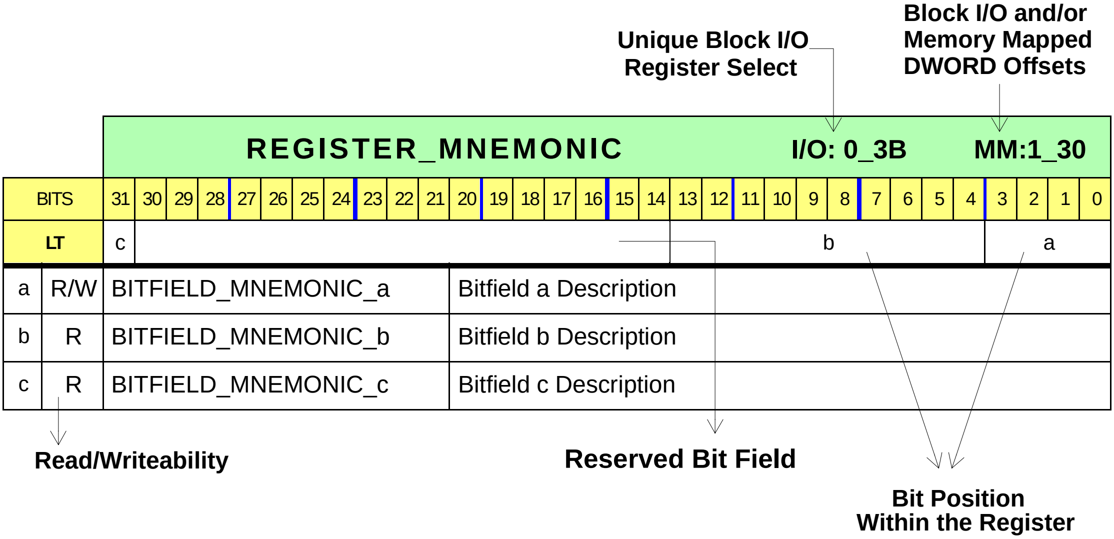
  CFG_CHIP_TYPE@CONFIG_CHIP_ID is the alternative to the above, but it gives the field name CFG_CHIP_TYPE (Product Type Code) instead of the bits position.

1.3.2 Numeric Representations

- Hexadecimal numbers are appended with "h" whenever there is a risk of ambiguity. Other numbers are assumed to be in decimal.
- Several signals of identical function are sometimes described by a single expression in which the part of the signal name that differs is shown in parentheses [ ]. For example, the four Select signals — SEL0#, SEL1#, SEL2#, and SEL3# — are represented by the single expression SEL[0-3]#.

1.3.3 Register Description Format

All registers in this document are described in the same or similar format as shown by the self-explained sample below.



REGISTER_MNEMONIC I/O: 0_3B MM:1_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
LT	c	b																											a			

a	R/W	BITFIELD_MNEMONIC_a	Bitfield a Description
b	R	BITFIELD_MNEMONIC_b	Bitfield b Description
c	R	BITFIELD_MNEMONIC_c	Bitfield c Description

1.3.4 Acronyms

Standard acronyms or abbreviations used in the literature are presumed known and therefore freely used without any explanation. When in doubt, refer to Table 1-1 below for a quick check. Less frequently used or ATI-specific acronyms will be accompanied by the full expression when appearing for the first time in the document.

Table 1-1

Acronym	Full Expression
AGP	Accelerated Graphics Port
AMC	ATI Multimedia Channel
BIOS	basic input/output system
bpp	bits per pixel
DAC	digital-to-analog converter
EDO RAM	Extended Data Output RAM
FIFO	first in first out
GUI	graphical user interface
I^2C	inter IC's communication
I/O	input/output
MPEG	Motion Picture Experts Group
MPP	Multimedia Peripheral Port
PCI	Peripheral Component Interconnect
PLL	phase-locked loop
POST	power-on self-test
RAMDAC	RAM digital-to-analog converter
RGB	red-green-blue (may refer to a color encoding scheme or a video signal)
R/W	read/write
SDRAM	Synchronous DRAM
SGRAM	Synchronous Graphics RAM
VGA	Video Graphics Array
WRAM	Window RAM
YUV	A color encoding scheme, no direct correspondence to the letters

Chapter 2
Overview and Memory Mapping

2.1 Features

The 3D RAGE LT PRO graphics controller offers all of the 2D, video, AGP-2X and hardware-accelerated 3D rendering and scaling functionality of the 3D RAGE PRO, plus an advanced LCD engine, ImpacTV2 quality TV output, dual CRTC capability and enhanced power management.

A full list of the features of the 3D RAGE LT PRO are found in the 3D RAGE LT PRO Controller Specifications, which you should also have in your possession. A subset of features that are LT specific are reproduced below.

2.1.1 LCD Related Features

The following are features related to the LCD functionality:

- Single chip solution in 256-pin and 312-pin BGA package, 0.35µm, 3.3V CMOS technology
- Full compliance with mobile PCI 1.0 and mobile AGP 1.0
- ZV (Zoom Video) port – supports video capture, video conferencing and MPEG decoder ZV-port.
- On-chip LVDS interface
    - Up to 10 LVDS channels
    - 65MHz pixel clock rate
    - 455Mbps/channel
    - Compatible with FPDI-2 and National Semiconductor receivers
- Supports optional external PanelLink transmitters
- Full-featured LCD panel support:
    - Universal Panel Interface allows manufacturing-time configuration of panel without swapping BIOS
    - Ratiometric expansion with horizontal filtering and automatic centering of

    screens to best fit panel size

    - Simulcast CRT and panel display
    - Backlight modulation for brightness control

- Panel resolution support for STN up to 1024x768 and TFT up to 1280x1024
- Panel color depth support:

Panel Type	**Non-dithered**	**Dithered**
Color STN	8 colors	256K colors
9-bit TFT	512 colors	32K colors
12-bit TFT	4096 colors	256K colors
15-bit TFT	32K colors	2.1M colors
18-bit TFT	256K colors	16.7M colors
24-bit TFT	16.7M colors	N/A

2.1.2 TV-Out Related Features

The following are features related to the TV-Out functionality:

- ImpacTV2 proven design producing scaled, flicker removed, artifact suppressed display on a PAL or NTSC TV
- Second Triple DAC for Composite and S-Video, or RGB (SCART) TV output formats
- TV Clock PLL to generate output at 50+ MSamples/sec to virtually eliminate need for external analogue filtering
- Independent horizontal and vertical scaling of the TV and CRT images
- Fully programmable TV timing, filter settings, brightness and colour saturation
- Y flicker filter with programmable (2, 3, 4, 5, 6) taps, and high quality UV filtering
- Optional Composite Dot Crawl freeze for both PAL and NTSC
- YUV direct and passthrough mode for video/MPEG playback and DVD
- Secondary display support for RGB modes and full screen YUV video/DVD playback on TV
- Line 21 Closed Caption, Extended Data Service and Line 20VBI capabilities
- Macrovision 7.01 copy protection standard (required by DVD players)
- Sample NTSC (Extended and VGA) Modes supported:

320x350; 320x400; 320x480; 360x400; 400x600; 512x384; 640x350; 640x400; 640x480; 720x350; 720x400; 720x480; 704x480; 800x600; 512x768; 640x768; 848x480; 1024x768; 1064x600

- Sample PAL (Extended and VGA) Modes supported:
  320x350; 320x400; 320x480; 352x576; 400x600; 512x384; 640x350; 640x400; 640x480; 720x350; 720x400; 720x576; 704x576; 800x600; 512x768; 640x768; 848x480; 1024x768; 1064x600

2.1.3 Dual CRTC Related Features

The following are features related to the Dual CRTC functionality:

- Two independent CRT controllers support two asynchronous simultaneous displays (LCD/CRT, LCD/TV, CRT/TV)
- All three displays may be used in any combinations with the dual CRTCs (LCD+TV/CRT, LCD+CRT/TV, CRT+TV/LCD, etc.)
- Independent resolutions, refresh rates, overscan and display data
- Secondary display supports YUV422 direct for full screen direct YUV to TV Output (ideal for full screen DVD video on TV while displaying in a window on CRT/LCD
- Duplicate secondary display fifo control for programming and flexibility
- Video genlocking support for both CRTCs
- Additional V2CLK PLL for secondary display
- Hardware Cursor and Hardware Icon (128x128) for primary display
- VGA and video overlay supported on primary display

2.1.4 ACPI Power Management Related Features

The following are features related to the Power Management functionality:

- Advanced Configuration and Power Interface (ACPI) with *On*, *Standby*, *Suspend* and *Off* modes.
- Mobile AGP 1.0 and mobile PCI 1.0 support.
- *Pin*, *Register*, *Timer* and *PCI Power Management* modes for hardware and software power management.

- Backlight control.
- Panel bias voltage and digital power control.
- Dynamic clock switching in *On* mode.
- Self-refresh and F32KHz refresh modes.

2.2 Register Classification

For ease of discussion and reference, the registers are grouped into the following main classes according to their functionality:

- Setup and Control registers
- Accelerator CRTC and DAC registers
- GUI registers
- Scaling and 3D Operations registers
- Multimedia registers
- Bus Mastering registers
- AGP registers
- LCD Panel registers
- TV Out Support registers
- PCI Configuration Space registers
- VGA registers

Note that these are general register classes only. There may be instances when specific bit fields of the same register may belong to a different class register. When this happens, it is noted in the register description.

Only registers in the first nine classes are tabulated in Chapter 3; PCI and VGA registers must be looked up from within their own chapters (8 and 9 respectively).

The following is an overview of all the registers. As can be seen, some classes are further divided into sub-groups.

2.2.1 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. They are further divided into:

- **General I/O Control register** — 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 register** — used to configure the on-chip bus interface unit.
- **Memory registers** — used to configure the memory interfaces.
- **Test and Debug registers** — used for chip diagnostics and hardware debugging.
- **Configuration registers** — used to configure the aperture and to read the current board configuration.
- **Custom Macros register** — used to set up custom macros

2.2.2 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.) They are further divided into the following groups:

- **Second CRTC registers** — used to define the secondary display parameters
- **Overscan registers** — used to configure overscan borders.
- **Hardware Cursor registers** — used to define and move the hardware cursor.
- **Genlocking registers** — used to synchronize the CRT with the video source.
- **Clock Control and PLL registers** — used to configure the pixel clock.
- **DAC Control registers** — used to configure the DAC.

2.2.3 Draw Engine Trajectory Registers

Draw Engine Trajectory Registers are memory mapped. They set up the source and destination trajectories and initiate draw operations. They are divided into two groups:

- **Destination Trajectory registers** — used to define the region in which pixels are drawn. The region may be a a line, a rectangular, or a trapezoidal area.
- **Source Trajectory registers** — used to 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.

2.2.4 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. They are divided into the following groups:

- **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 register** — used to report the status of the command FIFO.
- **Draw Engine Composite Control register** — abbreviated composites of other draw engine control registers.
- **Draw Engine Status register** — used to report the current state of the draw engine.

2.2.5 Scaler and 3D Accelerator Registers

The Scaler Pipe and 3D Accelerator registers are memory mapped and are further divided into the following groups:

- **Front-End Scaler Pipe registers** — used to configure the front-end scaler source data and to 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 to configure the associated quadratic interpolators.
- **Specular, Color, Z, and Alpha Interpolator registers** — used to configure the specular interpolation, the Z buffering and interpolation, the RGB and alpha interpolation, alpha blending, and fogging.
- **Setup Engine registers** — used to setup the draw and color/texture functions.

2.2.6 Multimedia Registers

These are registers used for multimedia operations such as video capture and playback. They are divided into the following groups:

- **Overlay Window registers** — used to specify the overlayed scaling window dimensions and coordinates to be displayed.
- **Overlay Scaler registers** — used to enable scaling and to set up the scaling factors.
- **Video Capture registers** — used to initialize, set the video configuration, define the capture buffer requirements, and trigger the capture.
- **Multimedia Peripheral Port (MPP) registers** — used to configure and access the MPP.
- **Hardware Assisted I2C registers** — used to control a 16 entry deep buffer for storing out-going or in-coming data.

2.2.7 Bus Mastering Registers

The RAGE LT PRO provides full support for bus mastering to and from system memory. In other words, it is capable of reading system memory and transferring data to the frame buffer as well as writing frame buffer memory out to system memory.

Bus mastering operations are invoked either through the bus master system table or via the GUI Draw Engine.

- **Draw Engine Bus Mastering registers** — used to specify register address/data for bus mastering operations invoked through the Draw Engine.
- **System Bus Mastering registers** — used to control and determine the status of all the bus mastering operations.

2.2.8 AGP Registers

These registers are used to configure the Accelerated Graphic Port.

2.2.9 LCD Panel Registers

These registers are used to configure the LCD panel.

2.2.10 TV Out Support Registers

These registers are used to control TV Out related functions.

2.2.11 PCI Configuration Space Registers

The PCI Configuration Space registers determine the host bus configuration during system reset. For the RAGE LT PRO, 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.

2.2.12 VGA Registers

The VGA registers provide register-level compatibility with the IBM VGA display adapter. They and the accelerator registers are completely segregated from each other, and their functions are mutually exclusive.

Note that the ATI VGA extended registers, available with the *mach64* GX family (through 1CEh to 1CFh), are no longer included in the RAGE LT PRO.

2.3 Memory Mapping

The RAGE LT PRO uses a fully memory mapped programming model. All registers (except DAC_REGS, which is made up of a group of four 8-bit registers) are 32-bit wide and numbered from 0h to FFh. They are memory mapped into two 1K-blocks — block ‘0’ and block ‘1’ (see Fig 2.1 next page).

2.3.1 Mapping Model

In terms of memory mapping, the registers can be grouped into six major categories as shown below.

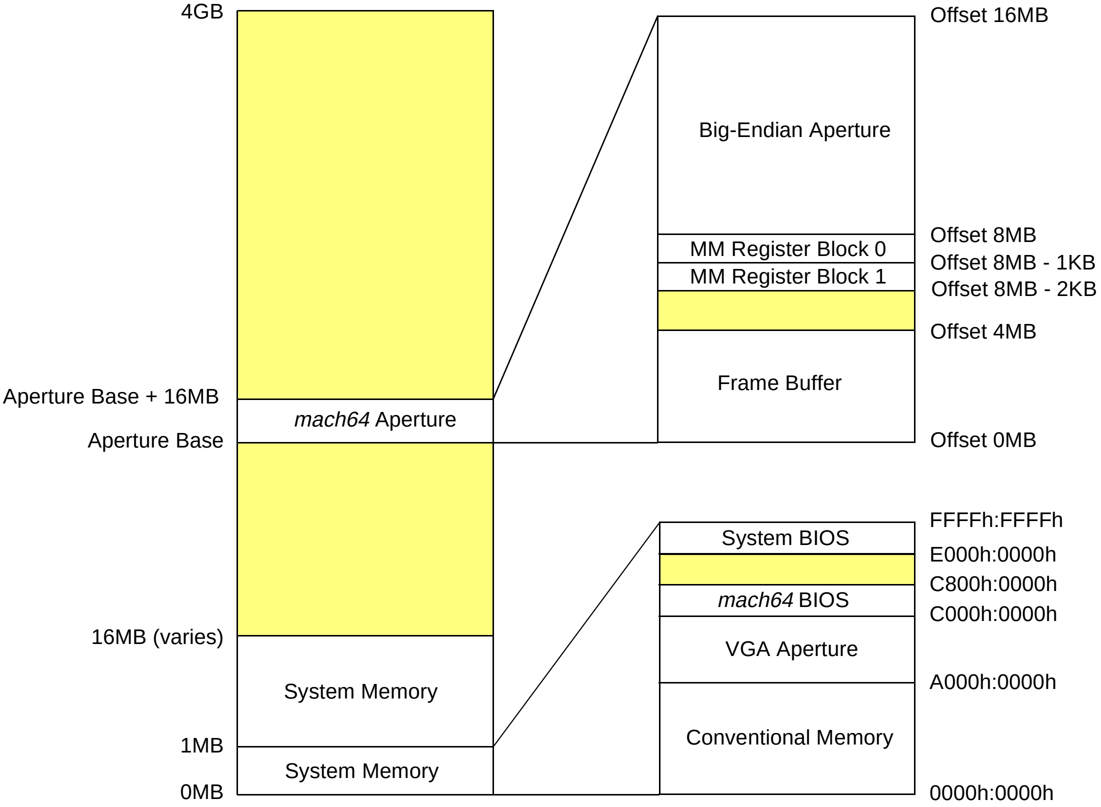
Table 2-1

Register Group	Purpose
PCI POS registers	PCI bus configuration
VGA registers	Registers for VGA compatibility
Display and Configuration	Accelerator mode registers the BIOS needs to access
GUI registers	2D and 3D draw engine registers
Multimedia registers	For video capture, overlay and multimedia port
PLL registers	Clock synthesis and clock source control

- The PCI POS registers exist only in the PCI configuration space.
- VGA registers are mapped to the standard VGA addresses in the I/0 space.
- The Accelerator registers (non-VGA) use apertures in both I/O and memory space. All accelerator registers are visible in the memory space, but only a subset is mapped to the I/O space.
- All 2D and 3D draw engine registers are memory mapped to Block ‘0’, with Dword offsets between 40h and FFh inclusive. These registers are written through a command FIFO and are read directly.
- The multimedia registers are memory mapped to Block ‘1’, with Dword offsets between 00h and FFh inclusive. Some selected multimedia registers also appear in the I/O space, but with different offsets for I/O and memory.
- The PLL registers are accessed indirectly through the CLOCK_CNTL register.

Registers not associated with the 2D and 3D draw engines and multimedia are generally used for display and configuration purposes. These registers, numbered from 00h to 3Fh, are directly readable and writeable. In addition to being memory mapped, they are also mapped into a single continuous I/O block (referred to as “block” or “relocatable” I/O) which starts at the I/O base address specified in the PCI Configuration registers. This allows the registers to be accessed at I/O addresses aliased to offsets from the base I/O address. For block I/O, the I/O base address can be located anywhere within the 64K I/O space.

Note that Sparse I/O decoding is **not** supported by the RAGE LT PRO.



Aperture Base address can be located anywhere in the shaded region and is aligned to a multiple of 16MB.

Figure 2-1 Typical Organization of Aperture Within Host Address Space

The table below summarizes the mapping of the register groups to memory and to the I/O space.

Table 2-2

Register Group	I/O Mapping	Memory Mapping	Comments
PCI configuration	No	No	Accessed with configuration cycles
VGA	Yes	No	VGA standard addresses
Display and Configuration	00h to 3Fh	0_00h to 0_3Fh	Same offset for I/O and memory
GUI	No	0_40h to 0_FFh	Memory mapped only
Multimedia registers	some in 00h to 3Fh	1_00h to 1_FFh	Some selected registers in I/O space, but different offsets for I/O and memory
PLL	24h	0_24h	Accessed indirectly through CLOCK_CNTL register

2.3.2 Accessing Bytes, Words, and Dwords

The table below indicates which register groups may be accessed as bytes, words, or Dwords.

Table 2-3

Register Group	Byte Addressing	Word Addressing	Dword Addressing
PCI POS registers	Yes	Yes	Yes
VGA registers	Yes	Note 1	Note 1
Display & Configuration	Yes	Yes, note 2	Yes, note 2
GUI registers	No	No	Yes
Multimedia registers	No, note 3	No, note 3	Yes
PLL registers	Yes	No	No

Notes:

- If two or four VGA registers are continuous in the I/O space, 16 or 32 cycles may be used. The cycle will be broken up internally into 2 or 4 sequential cycles starting with the lowest address first.

- The DAC_REGS register is actually four 8-bit registers. Word or Dword cycles will be broken up internally into 2 or 4 sequential cycles starting with the lowest address first.
- The multimedia registers that appear in I/O space are Dword-only registers. This means 32 bit IN or OUT operations must be used.
- When trying to access only a byte or word of a 32 bit register, simply add 1, 2 or 3 to the absolute address calculated as shown earlier.
- It is not recommended to perform word or Dword cycles that span a Dword boundary. This will not work correctly in all cases.

2.3.3 Non-Intel Based Memory Mapping

When incorporating the RAGE LT PRO into a non-Intel platform (such as the Apple Power Macintosh), make sure the platform conforms to the PCI specification. For information on how to configure the RAGE LT PRO in non-Intel environments, refer to Chapter 2 of the *mach64 Programmer's Guide*.

2.4 Mapping Modes

Depending on the system configuration, the RAGE LT PRO operates in either of two selectable register mapping modes – Linear Aperture mode or VGA Aperture mode. The Linear Aperture mode is optimized for PCI configurations, while the VGA Aperture mode is used for backward compatibility to ISA-based systems. 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.

2.4.1 Linear Aperture Mapping

In Linear Aperture mode, a large linear (primary) memory aperture and a small auxiliary register aperture are set up. The primary aperture size is fixed at 2x8 MB, and that of the auxillary register aperture is 4 KB. The latter is used exclusively for register access (no mapping to the frame buffer) and is always enabled.

Primary Aperture

This aperture maps the entire frame buffer into the system memory address space. The memory mapped registers are optionally visible at the top of the first 8 MB of the primary linear aperture.

The aperture position is set by the system BIOS at configuration time through the Base Address registers in the PCI configuration space (see *Chapter 6*). The aperture size and position can also be read from the read-only *CONFIG_CNTL* register.

For the primary aperture, register Block ‘0’ (CT-compatible block) is located in the top 1K of the first 8 MB, and register Block ‘1’ (multimedia extensions) is 1K below Block ‘0’. The figure below shows the positions of register Block ‘0’ and register Block ‘1’ in a typical primary aperture configuration.

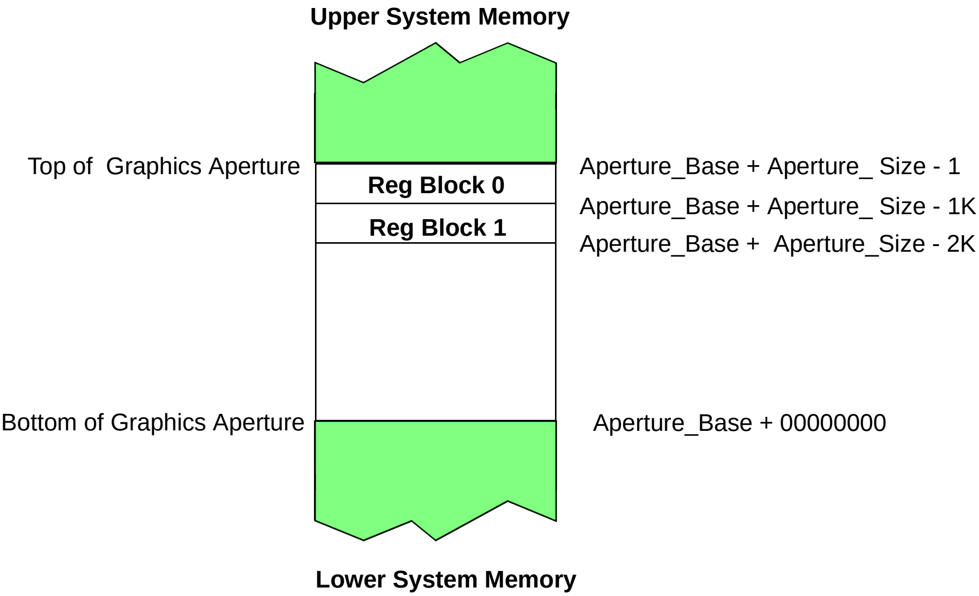

Upper System Memory
Top of Graphics Aperture
Reg Block 0
Reg Block 1
Aperture_Base + Aperture_ Size - 1
Aperture_Base + Aperture_ Size - 1K
Aperture_Base + Aperture_Size - 2K
Bottom of Graphics Aperture
Aperture_Base + 00000000
Lower System Memory

Figure 2-2 Primary Linear Aperture Register Map

The table below defines the register block offset within the linear aperture. “N/A” indicates that the registers are not visible with those settings. Reads or writes to those addresses will go to the frame buffer memory.

Table 2-4

BUS_APER_REG_DIS @BUS_CNTL	BUS_EXT_REG_EN @BUS_CNTL	Register Block 0 Offset	Register Block 1 Offset
1	X	N/A	N/A
0	0	7FFC00	N/A
0	1	7FFC00	7FF800

Auxillary Aperture

The auxillary register aperture is permanently enabled and available. All the registers in this aperture are mapped to the same offset as the primary linear register aperture (at the top of the first 8MB of the linear aperture). This aperture can be used in place of the primary aperture. The purpose of this auxillary register aperture is to allow the primary aperture to be disabled to enable access to the frame buffer memory mapped behind the memory-mapped registers. In this way, the auxiliary and the primary apertures are independent of each other.

The memory map of the auxillary aperture is shown in the figure below:

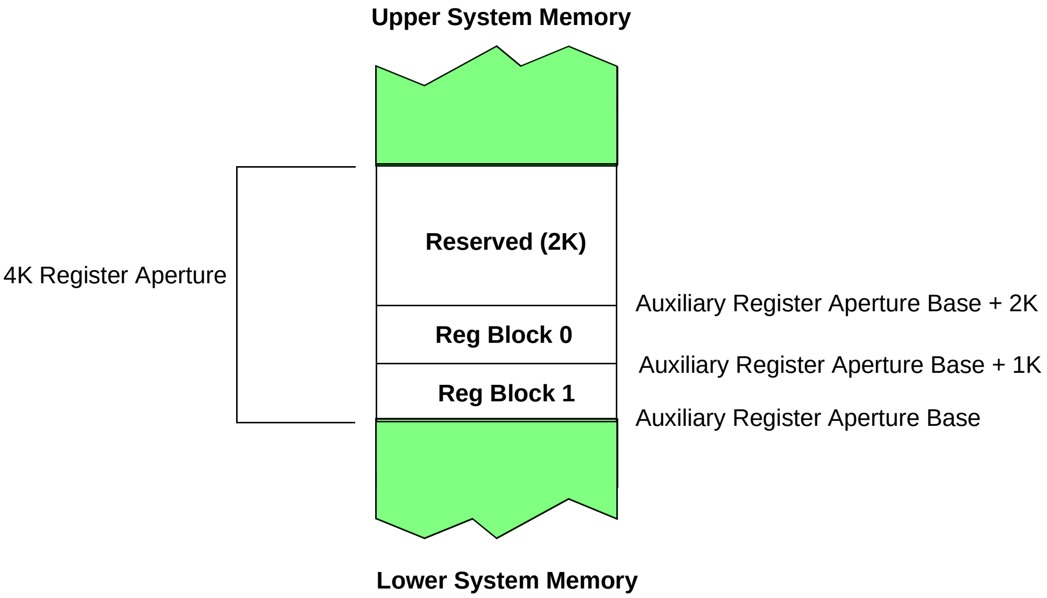

Upper System Memory
4K Register Aperture
Reserved (2K)
Reg Block 0
Reg Block 1
Auxiliary Register Aperture Base + 2K
Auxiliary Register Aperture Base + 1K
Auxiliary Register Aperture Base
Lower System Memory

Figure 2-3 Auxiliary Register Aperture Memory Map

In the auxiliary register aperture, the base of Block '1' is always at offset 0 and the base of Block '0' is always at offset 400h. The upper 2K of the aperture is reserved. This allows the register aperture to be 4K-aligned.

2.4.2 VGA Aperture Mapping

The VGA registers are completely segregated from the Accelerator registers. They provide compatibility with the IBM VGA Display Adapter. VGA apertures are 64K or 128K for standard VGA modes. VGA registers are I/O mapped only (with absolute addresses given in the descriptions in Chapter 9). They cannot be moved and are not configurable.

The VGA aperture is fixed between A0000h and BFFFFh, and the VGA I/O space is fixed at these 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.

In VGA aperture mode, the upper 1KB (or 2KB) of the 128KB aperture is reserved for the controller register space. The Graphics Miscellaneous Register (a VGA register) has two bits (GRPH_ADRSEL) that control which part of the VGA aperture is enabled. The register mapping can only occur when the entire 128KB aperture is enabled, or just the top 32KB is enabled. The other two cases do not include the area from BF800h to BFFFFh, and therefore can not map the registers.

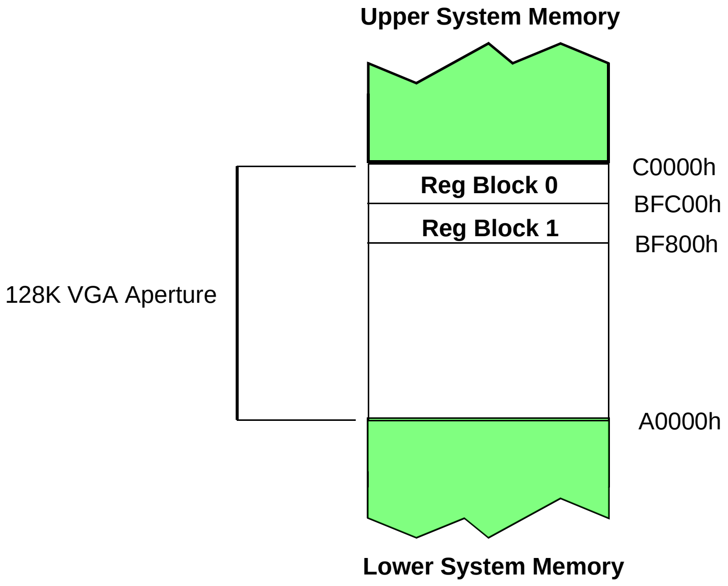


Figure 2-4 VGA Aperture Memory Map

The following table indicates the various bits that must be set to enable register mapping into the VGA aperture.

Table 2-5

CFG_MEM_VGA_AP_EN @CONFIG_CNTL	BUS_EXT_REG_EN @BUS_CNTL	GRPH_ADRSEL @GRA06	Register Block 0 Base	Register Block 1 Base
0	X	XX	N/A	N/A
X	X	01 or 10	N/A	N/A
1	0	00 or 11	BFC00	N/A
1	1	00 or 11	BFC00	BF800

2.5 Determining Mapped Addresses

2.5.1 Memory Address

The notation for the block/Dword offset (see cross-reference tables in Chapter 3) is

MM: block#_offset, where:

MM denotes memory mapped

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

offset identifies the register Dword offset *within* the associated block, in hexidecimal

For example, the OVERLAY_SCALE_CNTL register address, given by **MM: 1_09**, indicates that this register is located in register Block ‘1’ (2K below the top of the graphics aperture) at Dword offset 9h. The Dword offset is converted to a byte offset by multiplying it by 4 (i.e., Dword offset 9h is byte offset 24h).

Note: For non-VGA registers, which are visible in both I/O and memory space, **MM** is replaced by **I/O**.

Absolute Register Address

The memory mapped registers are visible at the top of the first 8MB of the linear aperture. With the base of the aperture located at 8 MB in memory, the absolute primary aperture register address is calculated using the formula:

Absolute register address = (aperture_base_address) + (reg_block_offset) + (reg_byte_offset)

Relative Register Address

The relative register address (to the base of the primary linear aperture) is calculated using the formula:

Relative register address = register block offset + register byte offset

For example, for register (OVERLAY_SCALE_CNTL):

aperture base address	=	800000h
register block 1 offset	=	7FF800h
register byte offset	=	24h (4*Dword offset)
Absolute register address	=	FFF824h
Relative register address	=	7FF824h

Auxiliary Aperture Register Address

An auxiliary aperture register address is calculated by:

auxiliary register address = (aperture_base_address) + (reg_block_offset) + (reg_byte_offset)

For example:

aperture base address	=	C00000h
register Block 1 offset	=	0h
register byte offset	=	24h (4*Dword offset)
Absolute register address	=	C00024h

VGA Aperture Register Address

A VGA aperture register address is calculated by:

Aperture register address = (register_block_base) + (reg_byte_offset)

For example:

register Block 1 base	=	BF800h
register byte offset	=	24h (4*Dword offset)
Absolute register address	=	BF824h

Note: All register offsets that are **not** preceeded by a block number are assumed to be in Block ‘0’.

2.5.2 I/O Base Address

As mentioned earlier, all display and configuration registers not associated with the 2D/3D draw engines or multimedia are I/O mapped and have memory mapped register aliases.

To support block I/O register mapping, the RAGE LT PRO requests a 256 byte I/O aperture, thus allowing 64 I/O mapped registers. The registers are mapped into this continuous block, starting at the I/O base address specified in the PCI configuration registers (summarized in *Chapter 6*).

Since the I/O base address may be different depending on the card configuration, it cannot be assumed to be of 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).

For block I/O, the I/O base address can be of 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 (PnP) specification. Refer to Chapter 2 of the *mach64 Programmer's Guide* for more information on how to use this function call.

2.5.3 Absolute I/O Address

For display and configuration registers not associated with the 2D/3D draw engines or multimedia, the block I/O offset is given as the Dword offset (**Offset**) from the memory mapped register base address and the block I/O base address.

For block I/O, the equation for determining the Absolute I/O address is:

*Absolute I/O address = (Dword Offset *4) + I/O base address*

Using SCRATCH_REG1 as an example, the Dword offset is given by **Offset: 0_21**. This indicates that the register is located in register Block '0' at Dword offset 21h. The Dword offset is converted to a byte offset by multiplying by 4 (therefore, Dword offset 21h is byte offset 84h). If the I/O base address = E000h and the Dword offset = 21h, the physical I/O address will be E084h.

For some I/O registers, it is necessary to access individual bytes within the 32-bit register (for example, DAC_REGS, which is actually four 8-bit registers). In those cases, the Dword offset should be converted to a byte offset before adding the individual byte offset (0, 1, 2, or 3).

As an example, the procedure to access the DAC_MASK byte of DAC_REGS is shown below:

byte offset	= Dword Offset * 4 = 30h * 4 = 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

This page intentionally left blank.

Chapter 3
Cross Reference Tables

3.1 Using the Tables

This chapter comprises two main register summary tables, followed by two sub-tables listing the indexed LCD and TV Out registers.

The two main tables list all but the PCI POS and the VGA Controller registers, which are covered in Chapters 8 and 9 respectively. The first table lists the registers by address while the second by mnemonic.

The tables in this chapter offer a convenient way to locate the full description of any of the registers contained in Chapters 4, 5, 6, and 7, 10, 11, and some (excluding the PCI registers) contained in Chapter 8. If you are using the online version of this document, the page numbers are hypertext linked to the register descriptions.

Note: Since the PLL registers are accessed through the Clock Control (CLOCK_CNTL) register, they are not listed in the summary tables. To locate them, use one of the summary tables (or the Index at the end of this document) to locate CLOCK_CNTL first, then you'll find the PLL registers described in the section following it.

Table Notations:

- Registers common to both the RAGE PRO and RAGE LT PRO are in normal print, those that have been modified for the RAGE LT PRO are in **bold**, and those that are specific to the RAGE LT PRO are noted with **LT only** in addition.
- A tick (✓) in the column under the heading *Block I/O* indicates that the register is aliased at an I/O address.
- The *(h)* in the *DWORD Offset* column heading indicates that numerals are in hex.
- *R/W* means read and write, *R* means read only, and *W* means write only.

3.2 Listing by Address

Table 3-1

Registers by Address

Register Class	Mnemonic	Read/ Write	Block I/O	DWORD Offset (h)	Page
Accelerator CRTC	CRTC_H_TOTAL_DISP	R/W	✓	0_00	4-38
	CRTC2_H_TOTAL_DISP (LT only)	**R/W**	✓	**0_00**	**4-39**
	CRTC_H_SYNC_STRT_WID	R/W	✓	0_01	4-39
	CRTC2_H_SYNC_STRT_WID (LT only)	**R/W**	✓	**0_01**	**4-40**
	CRTC_V_TOTAL_DISP	R/W	✓	0_02	4-40
	CRTC2_V_TOTAL_DISP (LT only)	**R/W**	✓	**0_02**	**4-41**
	CRTC_V_SYNC_STRT_WID	R/W	✓	0_03	4-41
	CRTC2_V_SYNC_STRT_WID (LT only)	**R/W**	✓	**0_03**	**4-42**
	CRTC_VLINE_CRNT_VLINE	R/W	✓	0_04	4-42
	CRTC2_VLINE_CRNT_VLINE (LT only)	**R/W**	✓	**0_04**	**4-43**
	CRTC_OFF_PITCH	R/W	✓	0_05	4-43
	CRTC_INT_CNTL	**R/W**	✓	**0_06**	**4-44**
	CRTC_GEN_CNTL	**R/W**	✓	**0_07**	**4-47**
Memory Buffer Control	DSP_CONFIG	R/W	✓	0_08	4-9
	DSP_ON_OFF	R/W	✓	0_09	4-9
	TIMER_CONFIG	R/W	✓	0_0A	4-10
	MEM_BUF_CNTL	R/W	✓	0_0B	4-11
	MEM_ADDR_CONFIG	R/W	✓	0_0D	4-12
Accelerator CRTC	CRT_TRAP	R/W	✓	0_0E	4-50
I2C Control	I2C_CNTL_0	R/W	✓	0_0F	7-38

Table 3-1 Cont'd

Registers by Address

Register Class	Mnemonic	Read/ Write	Block I/O	DWORD Offset (h)	Page
Overscan	OVR_CLR	R/W	✓	0_10	*4-51*
	OVR2_CLR (LT only)	**R/W**	**✓**	**0_10**	***4-51***
	OVR_WID_LEFT_RIGHT	R/W	✓	0_11	*4-51*
	OVR2_WID_LEFT_RIGHT (LT only)	**R/W**	**✓**	**0_11**	***4-52***
	OVR_WID_TOP_BOTTOM	R/W	✓	0_12	*4-52*
	OVR2_WID_TOP_BOTTOM (LT only)	**R/W**	**✓**	**0_12**	***4-53***
Memory Buffer Control	VGA_DSP_CONFIG	R/W	✓	0_13	*4-11*
	VGA_DSP_ON_OFF	R/W	✓	0_14	*4-12*
	DSP2_CONFIG (LT only)	**R/W**	**✓**	**0_15**	***4-9***
	DSP2_ON_OFF (LT only)	**R/W**	**✓**	**0_16**	***4-10***
Accelerator CRTC	**CRTC2_OFF_PITCH (LT only)**	**R/W**	**✓**	**0_17**	***4-44***
Hardware Cursor	CUR_CLR0	R/W	✓	0_18	*4-54*
	CUR_CLR1	R/W	✓	0_19	*4-55*
	CUR_OFFSET	R/W	✓	0_1A	*4-56*
	CUR_HORZ_VERT_POSN	R/W	✓	0_1B	*4-56*
	CUR_HORZ_VERT_OFF	**R/W**	**✓**	**0_1C**	***4-57***
For related registers see Table 3-4	**TV_OUT_INDEX (LT only)**	**R/W**	**✓**	**0_1D**	***11-1***
General I/O Control	**GP_IO**	**R/W**	**✓**	**0_1E**	***4-1***
Test and Debug	**HW_DEBUG**	**R/W**	**✓**	**0_1F**	***4-23***
Scratch Pad and Test	SCRATCH_REG0	R/W	✓	0_20	*4-3*
	SCRATCH_REG1	R/W	✓	0_21	*4-3*
	SCRATCH_REG2	**R/W**	**✓**	**0_22**	***4-4***
	SCRATCH_REG3	**R/W**	**✓**	**0_23**	***4-4***
Clock Control	**CLOCK_CNTL**	**R/W**	**✓**	**0_24**	***4-61***
Configuration	CONFIG_STAT1	R	✓	0_25	*4-29*
	CONFIG_STAT2	**R**	**✓**	**0_26**	***4-30***
For related registers see Table 3-4	**TV_OUT_DATA (LT only)**	**R/W**	**✓**	**0_27**	***11-1***

Table 3-1 Cont'd

Registers by Address					
Register Class	**Mnemonic**	**Read/ Write**	**Block I/O**	**DWORD Offset (h)**	**Page**
Bus Control	BUS_CNTL	R/W	✓	0_28	4-5
For related registers see Table 3-3	**LCD_INDEX (LT only)**	**R/W**	✓	**0_29**	**8-1**
	LCD_DATA (LT only)	**R/W**	✓	**0_2A**	**8-1**
Memory Control	EXT_MEM_CNTL	R/W	✓	0_2B	4-13
	MEM_CNTL	R/W	✓	0_2C	4-16
	MEM_VGA_WP_SEL	R/W	✓	0_2D	4-18
	MEM_VGA_RP_SEL	R/W	✓	0_2E	4-19
I2C Control	**I2C_CNTL_1**	**R/W**	✓	**0_2F**	**7-39**
DAC Control	DAC_REGS	R/W	✓	0_30	4-82
	DAC_CNTL	**R/W**	✓	**0_31**	**4-84**
Test and Debug	**GEN_TEST_CNTL**	**R/W**	✓	**0_34**	**4-21**
Custom Macros	**CUSTOM_MACRO_CNTL**	**R/W**	✓	**0_35**	**4-31**
Configuration	CONFIG_CNTL	R/W	✓	0_37	4-26
	CONFIG_CHIP_ID	**R**	✓	**0_38**	**4-27**
	CONFIG_STAT0	**R/W**	✓	**0_39**	**4-28**
Test and Debug	CRC_SIG	R	✓	0_3A	4-23

Table 3-1 Cont'd

Registers by Address					
Register Class	**Mnemonic**	**Read/ Write**	**Block I/O**	**DWORD Offset (h)**	**Page**
Draw Engine Destination Trajectory	DST_OFF_PITCH	R/W	-	0_40	5-9
	DST_X	R/W	-	0_41	5-11
	DST_Y	R/W	-	0_42	5-14
	DST_Y_X (aliased at 0_4Dh, 3D)	W	-	0_43	5-14
	DST_WIDTH	R/W	-	0_44	5-10
	DST_HEIGHT	R/W	-	0_45	5-8
	DST_HEIGHT_WIDTH	W	-	0_46	5-8
	DST_X_WIDTH	W	-	0_47	5-12
	DST_BRES_LNTH (LEAD_BRES_LNTH) (aliased at 0_51h, 3D)	R/W	-	0_48	5-3
	DST_BRES_ERR	R/W	-	0_49	5-2
	DST_BRES_INC (LEAD_BRES_INC, 3D)	R/W	-	0_4A	5-3
	DST_BRES_DEC (LEAD_BRES_DEC, 3D)	R/W	-	0_4B	5-1
	DST_CNTL	R/W	-	0_4C	5-5
	DST_Y_X (aliased at 0_43h, 3D)	W	-	0_4D	5-14
	TRAIL_BRES_ERR	R/W	-	0_4E	5-15
	TRAIL_BRES_INC	R/W	-	0_4F	5-15
	TRAIL_BRES_DEC	R/W	-	0_50	5-16
	LEAD_BRES_LNTH (aliased at 0_48h, 3D)	R/W	-	0_51	5-3
	Z_OFF_PITCH	R/W	-	0_52	5-17
	Z_CNTL	R/W	-	0_53	5-17
	ALPHA_TST_CNTL	R/W	-	0_54	5-18

Table 3-1 Cont'd

Registers by Address					
Register Class	**Mnemonic**	**Read/ Write**	**Block I/O**	**DWORD Offset (h)**	**Page**
Texture Mapping	SECONDARY_STW_EXP	R/W	-	0_56	6-16
	SECONDARY_S_X_INC	R/W	-	0_57	6-17
	SECONDARY_S_Y_INC	R/W	-	0_58	6-17
	SECONDARY_S_START	R/W	-	0_59	6-17
	SECONDARY_W_X_INC	R/W	-	0_5A	6-17
	SECONDARY_W_Y_INC	R/W	-	0_5B	6-18
	SECONDARY_W_START	R/W	-	0_5C	6-18
	SECONDARY_T_X_INC	R/W	-	0_5D	6-18
	SECONDARY_T_Y_INC	R/W	-	0_5E	6-18
	SECONDARY_T_START	R/W	-	0_5F	6-19
Draw Engine Source Trajectory	SRC_OFF_PITCH	R/W	-	0_60	5-26
	SRC_X	R/W	-	0_61	5-28
	SRC_Y	R/W	-	0_62	5-29
	SRC_Y_X	W	-	0_63	5-31
	SRC_WIDTH1	R/W	-	0_64	5-27
	SRC_HEIGHT1	R/W	-	0_65	5-23
	SRC_HEIGHT1_WIDTH1	W	-	0_66	5-24
	SRC_X_START	R/W	-	0_67	5-29
	SRC_Y_START	R/W	-	0_68	5-30
	SRC_Y_X_START	W	-	0_69	5-31
	SRC_WIDTH2	R/W	-	0_6A	5-27
	SRC_HEIGHT2	R/W	-	0_6B	5-25
	SRC_HEIGHT2_WIDTH2	W	-	0_6C	5-25
	SRC_CNTL	R/W	-	0_6D	5-21
Scaler Pipe	SCALE_OFF	R/W	-	0_70	6-1

Table 3-1 Cont'd

Registers by Address					
Register Class	**Mnemonic**	**Read/ Write**	**Block I/O**	**DWORD Offset (h)**	**Page**
Texture Mapping	TEX_0_OFF	R/W	-	0_70	*6-13*
	TEX_1_OFF	R/W	-	0_71	*6-13*
	TEX_2_OFF	R/W	-	0_72	*6-13*
	TEX_3_OFF	R/W	-	0_73	*6-13*
	TEX_4_OFF	R/W	-	0_74	*6-13*
	TEX_5_OFF	R/W	-	0_75	*6-13*
	TEX_6_OFF	R/W	-	0_76	*6-13*
	TEX_7_OFF	R/W	-	0_77	*6-13*
Scaler Pipe	SCALE_WIDTH	R/W	-	0_77	*6-3*
	SCALE_HEIGHT	R/W	-	0_78	*6-4*
Texture Mapping	TEX_8_OFF	R/W	-	0_78	*6-13*
	TEX_9_OFF	R/W	-	0_79	*6-13*
	TEX_10_OFF	R/W	-	0_7A	*6-13*
	S_Y_INC (also at 0_D4h)	R/W	-	0_7B	*6-15*
Scaler Pipe	SCALE_PITCH (also at 0_D4h)	R/W	-	0_7B	*6-4*
	SCALE_X_INC (also at 0_F0h)	R/W	-	0_7C	*6-5*
Specular, Color, Z & Alpha Interpolation	RED_X_INC (also at 0_F0h)	R/W	-	0_7C	*6-27*
	GREEN_X_INC (also at 0_F3h)	R/W	-	0_7D	*6-28*
Scaler Pipe	SCALE_Y_INC (also at 0_F3h)	R/W	-	0_7D	*6-5*
	SCALE_VACC	R/W	-	0_7E	*6-6*
	SCALE_3D_CNTL	R/W	-	0_7F	*6-7*
Host Data	HOST_DATA[15:0]	W	-	0_80-8F	*5-33*
	HOST_CNTL	R/W	-	0_90	*5-34*
GUI Bus Mastering	BM_HOSTDATA	W	-	0_91	*6-11*
	BM_ADDR	W	-	0_92	*6-12*
	BM_DATA	W	-	0_92	*6-12*
	BM_GUI_TABLE_CMD	W	-	0_93	*6-13*

Table 3-1 Cont'd

Registers by Address					
Register Class	**Mnemonic**	**Read/ Write**	**Block I/O**	**DWORD Offset (h)**	**Page**
Pattern	PAT_REG0	R/W	-	0_A0	*5-37*
	PAT_REG1	R/W	-	0_A1	*5-37*
	PAT_CNTL	R/W	-	0_A2	*5-38*
Scissors	SC_LEFT	R/W	-	0_A8	*5-39*
	SC_RIGHT	R/W	-	0_A9	*5-40*
	SC_LEFT_RIGHT	W	-	0_AA	*5-40*
	SC_TOP	R/W	-	0_AB	*5-41*
	SC_BOTTOM	R/W	-	0_AC	*5-41*
	SC_TOP_BOTTOM	W	-	0_AD	*5-42*
Data Path	**USR1_DST_0FF_PITCH (LT only)**	**W**	**-**	**0_AE**	***5-52***
	USR2_DST_0FF_PITCH (LT only)	**W**	**-**	**0_AF**	***5-52***
	DP_BKGD_CLR	R/W	-	0_B0	*5-43*
	DP_FOG_CLR (DP_FRGD_CLR)	R/W	-	0_B1	*5-43*
	DP_WRITE_MSK	R/W	-	0_B2	*5-45*
	DP_PIX_WIDTH	R/W	-	0_B4	*5-46*
	DP_MIX	R/W	-	0_B5	*5-50*
	DP_SRC	R/W	-	0_B6	*5-58*
	DP_FRGD_CLR_MIX	W	-	0_B7	*5-45*
	DP_FRGD_BKGD_CLR	W	-	0_B8	*5-44*
Draw Engine Destination Trajectory	DST_X_Y	W	-	0_BA	*5-13*
	DST_WIDTH_HEIGHT	W	-	0_BB	*5-11*
Data Path	USR_DST_PITCH	W	-	0_BC	*5-52*
	DP_SET_GUI_ENGINE2	W	-	0_BE	*5-56*
	DP_SET_GUI_ENGINE	**W**	**-**	**0_BF**	***5-53***
Color Compare	CLR_CMP_CLR	R/W	-	0_C0	*5-59*
	CLR_CMP_MSK	R/W	-	0_C1	*5-60*
	CLR_CMP_CNTL	R/W	-	0_C2	*5-61*
Command FIFO	FIFO_STAT	R	-	0_C4	*5-62*
Engine Control	GUI_TRAJ_CNTL	R/W	-	0_CC	*5-64*

Table 3-1 Cont'd

Registers by Address					
Register Class	**Mnemonic**	**Read/ Write**	**Block I/O**	**DWORD Offset (h)**	**Page**
Engine Status	GUI_STAT	R	-	0_CE	5-66
Texture Mapping	TEX_PALETTE_INDEX	R/W	-	0_D0	6-25
	STW_EXP	R/W	-	0_D1	6-13
	LOG_MAX_INC	R/W	-	0_D2	6-14
	S_X_INC	R/W	-	0_D3	6-15
	S_Y_INC (also at 0_7Bh)	R/W	-	0_D4	6-15
Scaler Pipe	SCALE_PITCH (also at 0_7Bh)	R/W	-	0_D4	6-4
Texture Mapping	S_START	R/W	-	0_D5	6-15
	W_X_INC	R/W	-	0_D6	6-14
	W_Y_INC	R/W	-	0_D7	6-14
	W_START	R/W	-	0_D8	6-14
	T_X_INC	R/W	-	0_D9	6-15
	T_Y_INC	R/W	-	0_DA	6-16
Scaler Pipe	SECONDARY_SCALE_PITCH	R/W	-	0_DA	6-4
Texture Mapping	T_START	R/W	-	0_DB	6-16
	TEX_SIZE_PITCH	R/W	-	0_DC	6-19
	TEX_CNTL	R/W	-	0_DD	6-20
Scaler Pipe	SECONDARY_SCALE_OFF	R/W	-	0_DE	6-2
Texture Mapping	SECONDARY_TEX_OFFSET	R/W	-	0_DE	6-13
	TEX_PALETTE	R	-	0_DF	6-25
Scaler Pipe	SCALE_PITCH_BOTH	W	-	0_E0	6-4
	SECONDARY_SCALE_OFF_ACC	R/W	-	0_E1	6-3
	SCALE_OFF_ACC	R/W	-	0_E2	6-2
	SCALE_DST_Y_X	R/W	-	0_E3	6-7
Draw Engine Destination Trajectory	COMPOSITE_SHADOW_ID	R/W	-	0_E6	5-20
Scaler Pipe	SECONDARY_SCALE_X_INC	R/W	-	0_E7	6-5

Table 3-1 Cont'd

Registers by Address					
Register Class	**Mnemonic**	**Read/ Write**	**Block I/O**	**DWORD Offset (h)**	**Page**
Specular, Color, Z & Alpha Interpolation	SPECULAR_RED_X_INC	R/W	-	0_E7	6-25
	SPECULAR_RED_Y_INC	R/W	-	0_E8	6-25
	SPECULAR_RED_START	R/W	-	0_E9	6-25
Scaler Pipe	SECONDARY_SCALE_HACC	R/W	-	0_E9	6-12
Specular, Color, Z & Alpha Interpolation	SPECULAR_GREEN_X_INC	R/W	-	0_EA	6-26
	SPECULAR_GREEN_Y_INC	R/W	-	0_EB	6-26
	SPECULAR_GREEN_START	R/W	-	0_EC	6-26
	SPECULAR_BLUE_X_INC	R/W	-	0_ED	6-26
	SPECULAR_BLUE_Y_INC	R/W	-	0_EE	6-27
	SPECULAR_BLUE_START	R/W	-	0_EF	6-27
Scaler Pipe	SCALE_X_INC (also at 0_7Ch)	R/W	-	0_F0	6-5
Specular, Color, Z & Alpha Interpolation	RED_X_INC (also at 0_7Ch)	R/W	-	0_F0	6-27
	RED_Y_INC	R/W	-	0_F1	6-27
	RED_START	R/W	-	0_F2	6-28
Scaler Pipe	SCALE_HACC	R/W	-	0_F2	6-11
	SCALE_Y_INC (also at 0_7Dh)	R/W	-	0_F3	6-5
Specular, Color, Z & Alpha Interpolation	GREEN_X_INC (aliased to 0_7Dh)	R/W	-	0_F3	6-28
	GREEN_Y_INC	R/W	-	0_F4	6-28
Scaler Pipe	SECONDARY_SCALE_Y_INC	R/W	-	0_F4	6-6
	SECONDARY_SCALE_VACC	R/W	-	0_F5	6-6

Table 3-1 Cont'd

Registers by Address					
Register Class	**Mnemonic**	**Read/ Write**	**Block I/O**	**DWORD Offset (h)**	**Page**
Specular, Color, Z & Alpha Interpolation	GREEN_START	R/W	-	0_F5	6-28
	BLUE_X_INC	R/W	-	0_F6	6-29
	BLUE_Y_INC	R/W	-	0_F7	6-29
	BLUE_START	R/W	-	0_F8	6-29
	Z_X_INC	R/W	-	0_F9	6-29
	Z_Y_INC	R/W	-	0_FA	6-30
	Z_START	R/W	-	0_FB	6-30
	ALPHA_X_INC	R/W	-	0_FC	6-30
	FOG_X_INC	R/W	-	0_FC	6-30
	ALPHA_Y_INC	R/W	-	0_FD	6-30
	FOG_Y_INC	R/W	-	0_FD	6-30
	ALPHA_START	R/W	-	0_FE	6-31
	FOG_START	R/W	-	0_FE	6-31
Overlay Window Control	OVERLAY_Y_X_START	R/W	-	1_00	7-1
	OVERLAY_Y_X_END	R/W	-	1_01	7-2
	OVERLAY_VIDEO_KEY_CLR	R/W	-	1_02	7-3
	OVERLAY_VIDEO_KEY_MSK	R/W	-	1_03	7-4
	OVERLAY_GRAPHICS_KEY_CLR	R/W	-	1_04	7-4
	OVERLAY_GRAPHICS_KEY_MSK	R/W	-	1_05	7-5
	OVERLAY_KEY_CNTL	R/W	-	1_06	7-6
Overlay Scaler	OVERLAY_SCALE_INC	R/W	-	1_08	7-9
	OVERLAY_SCALE_CNTL	R/W	-	1_09	7-10
	SCALER_HEIGHT_WIDTH	R/W	-	1_0A	7-15
	SCALER_TEST	R/W	-	1_0B	7-16
	SCALER_BUF0_OFFSET	R/W	-	1_0D	7-12
	SCALER_BUF1_OFFSET	R/W	-	1_0E	7-13
	SCALER_BUF_PITCH	R/W	-	1_0F	7-14

Table 3-1 Cont'd

Registers by Address

Register Class	Mnemonic	Read/ Write	Block I/O	DWORD Offset (h)	Page
Video Capture	CAPTURE_START_END	R/W	-	1_10	*7-25*
	CAPTURE_X_WIDTH	R/W	-	1_11	*7-25*
	VIDEO_FORMAT	R/W	-	1_12	*7-20*
	VBI_START_END	R/W	-	1_13	*7-31*
	CAPTURE_CONFIG	**R/W**	**-**	**1_14**	***7-22***
	TRIG_CNTL	R/W	-	1_15	*7-26*
Overlay Window Control	OVERLAY_EXCLUSIVE_HORZ	R/W	-	1_16	*7-7*
	OVERLAY_EXCLUSIVE_VERT	R/W	-	1_17	*7-8*
Video Capture	VBI_WIDTH	R/W	-	1_18	*7-31*
	CAPTURE_DEBUG	R	-	1_19	*7-32*
	VIDEO_SYNC_TEST	R/W	-	1_1A	*7-27*
GenLocking	SNAPSHOT_VH_COUNTS	R	-	1_1C	*4-58*
	SNAPSHOT_F_COUNT	R	-	1_1D	*4-58*
	N_VIF_COUNT	R/W	-	1_1E	*4-59*
	SNAPSHOT_VIF_COUNT	R/W	-	1_1F	*4-59*
Video Capture	CAPTURE_BUF0_OFFSET	R/W	-	1_20	*7-29*
	CAPTURE_BUF1_OFFSET	R/W	-	1_21	*7-30*
	ONESHOT_BUF_OFFSET	R/W	-	1_22	*7-28*
GenLocking	**SNAPSHOT2_VH_COUNTS (LT only)**	**R**	**-**	**1_2C**	***4-58***
	SNAPSHOT2_F_COUNT (LT only)	**R**	**-**	**1_2D**	***4-58***
	N_VIF2_COUNT (LT only)	**R/W**	**-**	**1_2E**	***4-59***
	SNAPSHOT2_VIF_COUNT (LT only)	**R/W**	**-**	**1_2F**	***4-60***
Multimedia Peripheral Port	MPP_CONFIG	R/W	0_3B	1_30	*7-34*
	MPP_STROBE_SEQ	R/W	0_3C	1_31	*7-35*
	MPP_ADDR	R/W	0_3D	1_32	*7-36*
	MPP_DATA	R/W	0_3E	1_33	*7-36*
	TVO_CNTL	**R/W**	**0_3F**	**1_40**	***7-36***
Test and Debug	CRT_HORZ_VERT_LOAD	R/W	-	1_51	*4-25*

Table 3-1 Cont'd

Registers by Address					
Register Class	**Mnemonic**	**Read/ Write**	**Block I/O**	**DWORD Offset (h)**	**Page**
AGP	AGP_BASE	R/W	-	1_52	*6-14*
	AGP_CNTL	R/W	-	1_53	*6-14*
Overlay Scaler	SCALER_COLOUR_CNTL	R/W	-	1_54	*7-16*
	SCALER_H_COEFF0	R/W	-	1_55	*7-17*
	SCALER_H_COEFF1	R/W	-	1_56	*7-18*
	SCALER_H_COEFF2	R/W	-	1_57	*7-18*
	SCALER_H_COEFF3	R/W	-	1_58	*7-19*
	SCALER_H_COEFF4	R/W	-	1_59	*7-19*
Command FIFO	GUI_CMDFIFO_DEBUG	R/W	-	1_5C	*5-63*
	GUI_CMDFIFO_DATA	R/W	-	1_5D	*5-63*
	GUI_CNTL	R/W	-	1_5E	*5-63*
Bus Mastering	BM_FRAME_BUF_OFFSET	R	-	1_60	*6-10*
	BM_SYSTEM_MEM_ADDR	R	-	1_61	*6-10*
	BM_COMMAND	R	-	1_62	*6-10*
	BM_STATUS	R	-	1_63	*6-10*
	BM_GUI_TABLE	R/W	-	1_6E	*6-13*
	BM_SYSTEM_TABLE	R/W	-	1_6F	*6-11*
Overlay Scaler	SCALER_BUF0_OFFSET_U	R/W	-	1_75	*7-13*
	SCALER_BUF0_OFFSET_V	R/W	-	1_76	*7-14*
	SCALER_BUF1_OFFSET_U	R/W	-	1_77	*7-14*
	SCALER_BUF1_OFFSET_V	R/W	-	1_78	*7-14*

Table 3-1 Cont'd

Registers by Address

Register Class	Mnemonic	Read/ Write	Block I/O	DWORD Offset (h)	Page
Setup Engine	VERTEX_1_S	R/W	-	1_90, AB	6-31
	VERTEX_1_T	R/W	-	1_91,AC	6-32
	VERTEX_1_W	R/W	-	1_92,AD	6-32
	VERTEX_1_SPEC_ARGB	R/W	-	1_93,B4	6-33
	VERTEX_1_Z	R/W	-	1_94,B7	6-32
	VERTEX_1__ARGB	R/W	-	1_95, BA	6-33
	VERTEX_1_X_Y	R/W	-	1_96,BD	6-33
	ONE_OVER_AREA	R/W	-	1_97, 9F, A7	6-42
	VERTEX_2_S	R/W	-	1_98,AE	6-34
	VERTEX_2_T	R/W	-	1_99, AF	6-34
	VERTEX_2_W	R/W	-	1_9A,B0	6-34
	VERTEX_2_SPEC_ARGB	R/W	-	1_9B_B5	6-35
	VERTEX_2_Z	R/W	-	1_9C,B8	6-35
	VERTEX_2_ARGB	R/W	-	1_9D,BB	6-35
	VERTEX_2_X_Y	R/W	-	1_9E,BE	6-36
	VERTEX_3_S	R/W	-	1_A0,B1	6-36
	VERTEX_3_T	R/W	-	1_A1,B2	6-36
	VERTEX_3_W	R/W	-	1_A2,B3	6-37
	VERTEX_3_SPEC_ARGB	R/W	-	1_A3,B6	6-37
	VERTEX_3_Z	R/W	-	1_A4,B9	6-37
	VERTEX_3_ARGB	R/W	-	1_A5,BC	6-38
	VERTEX_3_X_Y	R/W	-	1_A6,BF	6-38
	VERTEX_3_SECONDARY_S	R/W	-	1_B0	6-41
	VERTEX_3_SECONDARY_T	R/W	-	1_B1	6-42
	VERTEX_3_SECONDARY_W	R/W	-	1_B2	6-42
	ONE_OVER_AREA_UC	R/W	-	1_C0	6-43
	SETUP_CNTL	R/W	-	1_C1	6-43
	VERTEX_1_SECONDARY_S	R/W	-	1_CA	6-38

Table 3-1 Cont'd

Registers by Address					
Register Class	**Mnemonic**	**Read/ Write**	**Block I/O**	**DWORD Offset (h)**	**Page**
Setup Engine	VERTEX_1_SECONDARY_T	R/W	-	1_CB	6-39
	VERTEX_1_SECONDARY_W	R/W	-	1_CC	6-39
	VERTEX_2_SECONDARY_S	R/W	-	1_CD	6-40
	VERTEX_2_SECONDARY_T	R/W	-	1_CE	6-40
	VERTEX_2_SECONDARY_W	R/W	-	1_CF	6-41

3.3 Listing by Mnemonic

Table 3-2

Registers by Mnemonic					
Register Class	**Mnemonic**	**Read/ Write**	**Block I/O**	**DWORD Offset (h)**	**Page**
AGP	AGP_BASE	R/W	-	1_52	*6-14*
	AGP_CNTL	R/W	-	1_53	*6-14*
Specular, Color, Z & Alpha Interpolation	ALPHA_START	R/W	-	0_FE	*6-31*
Draw Engine Destination Trajectory	ALPHA_TST_CNTL	R/W	-	0_54	*5-18*
Specular, Color, Z & Alpha Interpolation	ALPHA_X_INC	R/W	-	0_FC	*6-30*
	ALPHA_Y_INC	R/W	-	0_FD	*6-30*
	BLUE_START	R/W	-	0_F8	*6-29*
	BLUE_X_INC	R/W	-	0_F6	*6-29*
	BLUE_Y_INC	R/W	-	0_F7	*6-29*
GUI Bus Mastering	BM_ADDR	W	-	0_92	*6-12*
	BM_DATA	W	-	0_92	*6-12*
Bus Mastering	BM_COMMAND	R	-	1_62	*6-10*
	BM_FRAME_BUF_OFFSET	R	-	1_60	*6-10*
GUI Bus Mastering	BM_GUI_TABLE	R/W	-	1_6E	*6-13*
	BM_GUI_TABLE_CMD	W	-	0_93	*6-13*
	BM_HOSTDATA	W	-	0_91	*6-11*
Bus Mastering	BM_STATUS	R	-	1_63	*6-10*
	BM_SYSTEM_MEM_ADDR	R	-	1_61	*6-10*
	BM_SYSTEM_TABLE	R/W	-	1_6F	*6-11*
Bus Control	BUS_CNTL	R/W	✓	0_28	*4-5*

Table 3-2 Cont'd

Registers by Mnemonic					
Register Class	**Mnemonic**	**Read/ Write**	**Block I/O**	**DWORD Offset (h)**	**Page**
Video Capture	CAPTURE_BUF0_OFFSET	R/W	-	1_20	*7-29*
	CAPTURE_BUF1_OFFSET	R/W	-	1_21	*7-30*
	CAPTURE_CONFIG	**R/W**	**-**	**1_14**	***7-22***
	CAPTURE_DEBUG	R	-	1_19	*7-32*
	CAPTURE_START_END	R/W	-	1_10	*7-25*
	CAPTURE_X_WIDTH	R/W	-	1_11	*7-25*
Clock Control	**CLOCK_CNTL**	**R/W**	✓	**0_24**	***4-61***
Color Compare	CLR_CMP_CLR	R/W	-	0_C0	*5-59*
	CLR_CMP_CNTL	R/W	-	0_C2	*5-61*
	CLR_CMP_MSK	R/W	-	0_C1	*5-60*
Draw Engine Destination Trajectory	COMPOSITE_SHADOW_ID	R/W	-	0_E6	*5-20*
Configuration	**CONFIG_CHIP_ID**	**R**	✓	**0_38**	***4-27***
	CONFIG_CNTL	R/W	✓	0_37	*4-26*
	CONFIG_STAT0	**R/W**	✓	**0_39**	***4-28***
	CONFIG_STAT1	R	✓	0_25	*4-29*
	CONFIG_STAT2	**R**	✓	**0_26**	***4-30***
Test and Debug	CRC_SIG	R	✓	0_3A	*4-23*
	CRT_HORZ_VERT_LOAD	R/W	-	1_51	*4-25*
Accelerator CRTC	**CRTC_GEN_CNTL**	**R/W**	✓	**0_07**	***4-47***
	CRTC_H_SYNC_STRT_WID	R/W	✓	0_01	*4-41*
	CRTC_H_TOTAL_DISP	R/W	✓	0_00	*4-40*
	CRTC_INT_CNTL	**R/W**	✓	**0_06**	***4-44***
	CRTC_OFF_PITCH	R/W	✓	0_05	*4-43*
	CRTC_V_SYNC_STRT_WID	R/W	✓	0_03	*4-41*
	CRTC_V_TOTAL_DISP	R/W	✓	0_02	*4-40*
	CRTC_VLINE_CRNT_VLINE	R/W	✓	0_04	*4-42*
	CRT_TRAP	R/W	✓	0_0E	*4-50*
	CRTC2_H_SYNC_STRT_WID (LT only)	**R/W**	✓	**0_01**	***4-40***

Table 3-2 Cont'd

Registers by Mnemonic					
Register Class	Mnemonic	Read/ Write	Block I/O	DWORD Offset (h)	Page
Accelerator CRTC	**CRTC2_H_TOTAL_DISP (LT only)**	**R/W**	✓	**0_00**	***4-39***
	CRTC2_OFF_PITCH (LT only)	**R/W**	✓	**0_17**	***4-44***
	CRTC2_VLINE_CRNT_VLINE (LT only)	**R/W**	✓	**0_04**	***4-43***
	CRTC2_V_SYNC_STRT_WID (LT only)	**R/W**	✓	**0_03**	***4-42***
	CRTC2_V_TOTAL_DISP (LT only)	**R/W**	✓	**0_02**	***4-41***
Hardware Cursor	CUR_CLR0	R/W	✓	0_18	*4-54*
	CUR_CLR1	R/W	✓	0_19	*4-55*
	CUR_HORZ_VERT_OFF	**R/W**	✓	**0_1C**	***4-57***
	CUR_HORZ_VERT_POSN	R/W	✓	0_1B	*4-56*
	CUR_OFFSET	R/W	✓	0_1A	*4-56*
Custom Macros	**CUSTOM_MACRO_CNTL**	**R/W**	✓	**0_35**	***4-31***
DAC Control	**DAC_CNTL**	**R/W**	✓	**0_31**	***4-84***
	DAC_REGS	R/W	✓	0_30	*4-82*
Data Path	DP_BKGD_CLR	R/W	-	0_B0	*5-43*
	DP_FRGD_BKGD_CLR	W	-	0_B8	*5-44*
	DP_FRGD_CLR (DP_FOG_CLR, 3D)	R/W	-	0_B1	*5-43*
	DP_FRGD_CLR_MIX	W	-	0_B7	*5-45*
	DP_MIX	R/W	-	0_B5	*5-50*
	DP_PIX_WIDTH	R/W	-	0_B4	*5-46*
	DP_SET_GUI_ENGINE	**W**	**-**	**0_BF**	***5-53***
	DP_SET_GUI_ENGINE2	W	-	0_BE	*5-56*
	DP_SRC	R/W	-	0_B6	*5-58*
	DP_WRITE_MSK	R/W	-	0_B2	*5-45*
Memory Buffer Control	DSP_CONFIG	R/W	✓	0_08	*4-9*
	DSP_ON_OFF	R/W	✓	0_09	*4-9*
	DSP2_CONFIG (LT only)	**R/W**	✓	**0_15**	***4-9***
	DSP2_ON_OFF (LT only)	**R/W**	✓	**0_16**	***4-10***

Table 3-2 Cont'd

Registers by Mnemonic					
Register Class	**Mnemonic**	**Read/ Write**	**Block I/O**	**DWORD Offset (h)**	**Page**
Draw Engine Destination Trajectory	DST_BRES_DEC (LEAD_BRES_DEC)	R/W	-	0_4B	5-1
	DST_BRES_ERR	R/W	-	0_49	5-2
	DST_BRES_INC (LEAD_BRES_INC)	R/W	-	0_4A	5-3
	DST_BRES_LNTH (LEAD_BRES_LNTH)	R/W	-	0_48, 51	5-3
	DST_CNTL	R/W	-	0_4C	5-5
	DST_HEIGHT	R/W	-	0_45	5-8
	DST_HEIGHT_WIDTH	W	-	0_46	5-8
	DST_OFF_PITCH	R/W	-	0_40	5-9
	DST_WIDTH	R/W	-	0_44	5-10
	DST_WIDTH_HEIGHT	W	-	0_BB	5-11
	DST_X	R/W	-	0_41	5-11
	DST_X_WIDTH	W	-	0_47	5-12
	DST_X_Y	W	-	0_BA	5-13
	DST_Y	R/W	-	0_42	5-14
	DST_Y_X	W	-	0_43, 4D	5-14
Memory Control	EXT_MEM_CNTL	R/W	✓	0_2B	4-13
Command FIFO	FIFO_STAT	R	-	0_C4	5-62
Specular, Color, Z & Alpha Interpolation	FOG_START	R/W	-	0_FE	6-31
	FOG_X_INC	R/W	-	0_FC	6-30
	FOG_Y_INC	R/W	-	0_FD	6-30
Test and Debug	**GEN_TEST_CNTL**	**R/W**	✓	**0_34**	**4-21**
General I/O Control	**GP_IO**	**R/W**	**-**	**0_1E**	**4-1**
Specular, Color, Z & Alpha Interpolation	GREEN_START	R/W	-	0_F5	6-28
	GREEN_X_INC	R/W	-	0_7D,F3	6-28
	GREEN_Y_INC	R/W	-	0_F4	6-28

Table 3-2 Cont'd

Registers by Mnemonic

Register Class	Mnemonic	Read/ Write	Block I/O	DWORD Offset (h)	Page
Command FIFO	GUI_CMDFIFO_DATA	R/W	-	1_5D	*5-63*
	GUI_CMDFIFO_DEBUG	R/W	-	1_5C	*5-63*
	GUI_CNTL	R/W	-	1_5E	*5-63*
Engine Status	GUI_STAT	R	-	0_CE	*5-66*
Engine Control	GUI_TRAJ_CNTL	R/W	-	0_CC	*5-64*
Host Data	HOST_CNTL	R/W	-	0_90	*5-34*
	HOST_DATA[15:0]	W	-	0_80-8F	*5-33*
Test and Debug	**HW_DEBUG**	**R/W**	✓	**0_1F**	***4-23***
I2C Control	I2C_CNTL_0	R/W	✓	0_0F	*7-38*
	I2C_CNTL_1	**R/W**	✓	**0_2F**	***7-39***
For related registers see Table 3-3	**LCD_DATA (LT only)**	**R/W**	✓	**0_2A**	***8-1***
	LCD_INDEX (LT only)	**R/W**	✓	**0_29**	***8-1***
Draw Engine Destination Trajectory	LEAD_BRES_DEC (DST_BRES_DEC)	R/W	-	0_4B	*5-1*
	LEAD_BRES_INC (DST_BRES_INC)	R/W	-	0_4A	*5-3*
	LEAD_BRES_LNTH (DST_BRES_LNTH)	R/W	-	0_48, 51	*5-3*
Texture Mapping	LOG_MAX_INC	R/W	-	0_D2	*6-14*
Memory Buffer Control	MEM_ADDR_CONFIG	R/W	✓	0_0D	*4-12*
	MEM_BUF_CNTL	R/W	✓	0_0B	*4-11*
Memory Control	MEM_CNTL	R/W	✓	0_2C	*4-16*
	MEM_VGA_RP_SEL	R/W	✓	0_2E	*4-19*
	MEM_VGA_WP_SEL	R/W	✓	0_2D	*4-18*
Multimedia Peripheral Port	MPP_ADDR	R/W	0_3Dh	1_32	*7-36*
	MPP_CONFIG	R/W	0_3Bh	1_30	*7-34*
	MPP_DATA	R/W	0_3Eh	1_33	*7-36*
	MPP_STROBE_SEQ	R/W	0_3Ch	1_31	*7-35*
GenLocking	N_VIF_COUNT	R/W	-	1_1E	*4-59*
	N_VIF2_COUNT (LT only)	**R/W**	**-**	**1_2E**	***4-59***

Table 3-2 Cont'd

Registers by Mnemonic					
Register Class	**Mnemonic**	**Read/ Write**	**Block I/O**	**DWORD Offset (h)**	**Page**
Setup Engine	ONE_OVER_AREA	R/W	-	1_97, 9F, A7	*6-42*
	ONE_OVER_AREA_UC	R/W	-	1_C0	*6-43*
Video Capture	ONESHOT_BUF_OFFSET	R/W	-	1_22	*7-28*
Overlay Window Control	OVERLAY_EXCLUSIVE_HORZ	R/W	-	1_16	*7-7*
	OVERLAY_EXCLUSIVE_VERT	R/W	-	1_17	*7-8*
	OVERLAY_GRAPHICS_KEY_CLR	R/W	-	1_04	*7-4*
	OVERLAY_GRAPHICS_KEY_MSK	R/W	-	1_05	*7-5*
	OVERLAY_KEY_CNTL	R/W	-	1_06	*7-6*
Overlay Scaler	OVERLAY_SCALE_CNTL	R/W	-	1_09	*7-10*
	OVERLAY_SCALE_INC	R/W	-	1_08	*7-9*
Overlay Window Control	OVERLAY_VIDEO_KEY_CLR	R/W	-	1_02	*7-3*
	OVERLAY_VIDEO_KEY_MSK	R/W	-	1_0	*7-4*
	OVERLAY_Y_X_START	R/W	-	1_00	*7-1*
	OVERLAY_Y_X_END	R/W	-	1_01	*7-2*
Overscan	OVR_CLR	R/W	✓	0_10	*4-51*
	OVR_WID_LEFT_RIGHT	**R/W**	✓	**0_11**	***4-51***
	OVR_WID_TOP_BOTTOM	**R/W**	✓	**0_12**	***4-52***
	OVR2_CLR (LT only)	**R/W**	✓	**0_10**	***4-51***
	OVR2_WID_LEFT_RIGHT (LT only)	**R/W**	✓	**0_11**	***4-52***
	OVR2_WID_TOP_BOTTOM (LT only)	**R/W**	✓	**0_12**	***4-53***
Pattern	PAT_CNTL	R/W	-	0_A2	*5-38*
	PAT_REG0	R/W	-	0_A0	*5-37*
	PAT_REG1	R/W	-	0_A1	*5-37*
Specular, Color, Z & Alpha Interpolation	RED_START	R/W	-	0_F2	*6-28*
	RED_X_INC	R/W	-	0_7C,F0	*6-27*
	RED_Y_INC	R/W	-	0_F1	*6-27*

Table 3-2 Cont'd

Registers by Mnemonic					
Register Class	**Mnemonic**	**Read/ Write**	**Block I/O**	**DWORD Offset (h)**	**Page**
Scaler Pipe	SCALE_3D_CNTL	R/W	-	0_7F	6-7
	SCALE_DST_Y_X	R/W	-	0_E3	6-7
	SCALE_HACC	R/W	-	0_F2	6-11
	SCALE_HEIGHT	R/W	-	0_78	6-4
	SCALE_OFF	R/W	-	0_70	6-1
	SCALE_OFF_ACC	R/W	-	0_E2	6-2
	SCALE_PITCH	R/W	-	0_7B,D4	6-4
	SCALE_PITCH_BOTH	W	-	0_E0	6-4
	SCALE_VACC	R/W	-	0_7E	6-6
	SCALE_WIDTH	R/W	-	0_77	6-3
	SCALE_X_INC	R/W	-	0_7C,F0	6-5
	SCALE_Y_INC	R/W	-	0_7D,F3	6-5
Overlay Scaler	SCALER_BUF0_OFFSET	R/W	-	1_0D	7-12
	SCALER_BUF0_OFFSET_U	R/W	-	1_75	7-13
	SCALER_BUF0_OFFSET_V	R/W	-	1_76	7-14
	SCALER_BUF1_OFFSET	R/W	-	1_0E	7-13
	SCALER_BUF1_OFFSET_U	R/W	-	1_77	7-14
	SCALER_BUF1_OFFSET_V	R/W	-	1_78	7-14
	SCALER_BUF_PITCH	R/W	-	1_0F	7-14
	SCALER_COLOUT_CNTL	R/W	-	1_54	7-16
	SCALER_H_COEFF0	R/W	-	1_55	7-17
	SCALER_H_COEFF1	R/W	-	1_56	7-18
	SCALER_H_COEFF2	R/W	-	1_57	7-18
	SCALER_H_COEFF3	R/W	-	1_58	7-19
	SCALER_H_COEFF4	R/W	-	1_59	7-19
	SCALER_HEIGHT_WIDTH	R/W	-	1_0A	7-15
	SCALER_TEST	R/W	-	1_0B	7-16

Table 3-2 Cont'd

Registers by Mnemonic					
Register Class	**Mnemonic**	**Read/ Write**	**Block I/O**	**DWORD Offset (h)**	**Page**
Scissors	SC_BOTTOM	R/W	-	0_AC	5-41
	SC_LEFT	R/W	-	0_A8	5-39
	SC_LEFT_RIGHT	W	-	0_AA	5-40
	SC_RIGHT	R/W	-	0_A9	5-40
	SC_TOP	R/W	-	0_AB	5-41
	SC_TOP_BOTTOM	W	-	0_AD	5-42
Scratch Pad and Test	SCRATCH_REG0	R/W	✓	0_20	4-3
	SCRATCH_REG1	R/W	✓	0_21	4-3
	SCRATCH_REG2	**R/W**	✓	**0_22**	**4-4**
	SCRATCH_REG3	**R/W**	✓	**0_23**	**4-4**
Scaler Pipe	SECONDARY_SCALE_HACC	R/W	-	0_E9	6-12
	SECONDARY_SCALE_OFF	R/W	-	0_DE	6-2
	SECONDARY_SCALE_OFF_ACC	R/W	-	0_E1	6-3
	SECONDARY_SCALE_PITCH	R/W	-	0_DA	6-4
	SECONDARY_SCALE_VACC	R/W	-	0_F5	6-6
	SECONDARY_SCALE_X_INC	R/W	-	0_E7	6-5
	SECONDARY_SCALE_Y_INC	R/W	-	0_F4	6-6
Texture Mapping	SECONDARY_STW_EXP	R/W	-	0_56	6-16
	SECONDARY_S_START	R/W	-	0_59	6-17
	SECONDARY_S_X_INC	R/W	-	0_57	6-17
	SECONDARY_S_Y_INC	R/W	-	0_58	6-17
	SECONDARY_TEX_OFFSET	R/W	-	0_DE	6-13
	SECONDARY_T_START	R/W	-	0_5F	6-19
	SECONDARY_T_X_INC	R/W	-	0_5D	6-18
	SECONDARY_T_Y_INC	R/W	-	0_5E	6-18
	SECONDARY_W_START	R/W	-	0_5C	6-18
	SECONDARY_W_X_INC	R/W	-	0_5A	6-17
	SECONDARY_W_Y_INC	R/W	-	0_5B	6-18
Setup Engine	SETUP_CNTL	R/W	-	1_C1	6-43

Table 3-2 Cont'd

Registers by Mnemonic					
Register Class	**Mnemonic**	**Read/ Write**	**Block I/O**	**DWORD Offset (h)**	**Page**
GenLocking	SNAPSHOT_F_COUNT	R	-	1_1D	*4-58*
	SNAPSHOT_VH_COUNTS	R	-	1_1C	*4-58*
	SNAPSHOT_VIF_COUNT	R/W	-	1_1F	*4-59*
	SNAPSHOT2_F_COUNT (LT only)	**R**	**-**	**1_2D**	***4-58***
	SNAPSHOT2_VH_COUNTS (LT only)	**R**	**-**	**1_2C**	***4-58***
	SNAPSHOT2_VIF_COUNT (LT only)	**R/W**	**-**	**1_2F**	***4-60***
Specular, Color, Z & Alpha Interpolation	SPECULAR_BLUE_START	R/W	-	0_EF	*6-27*
	SPECULAR_BLUE_X-INC	R/W	-	0_ED	*6-26*
	SPECULAR_BLUE-Y_INC	R/W	-	0_EE	*6-27*
	SPECULAR_GREEN_START	R/W	-	0_EC	*6-26*
	SPECULAR_GREEN_X_INC	R/W	-	0_EA	*6-26*
	SPECULAR_GREEN_Y_INC	R/W	-	0_EB	*6-26*
	SPECULAR_RED_START	R/W	-	0_E9	*6-25*
	SPECULAR_RED_X-INC	R/W	-	0_E7	*6-25*
	SPECULAR_RED_Y_INC	R/W	-	0_E8	*6-25*

Table 3-2 Cont'd

Registers by Mnemonic					
Register Class	**Mnemonic**	**Read/ Write**	**Block I/O**	**DWORD Offset (h)**	**Page**
Draw Engine Source Trajectory	SRC_CNTL	R/W	-	0_6D	5-21
	SRC_HEIGHT1	R/W	-	0_65	5-23
	SRC_HEIGHT1_WIDTH1	W	-	0_66	5-24
	SRC_HEIGHT2	R/W	-	0_6B	5-25
	SRC_HEIGHT2_WIDTH2	W	-	0_6C	5-25
	SRC_OFF_PITCH	R/W	-	0_60	5-26
	SRC_WIDTH1	R/W	-	0_64	5-27
	SRC_WIDTH2	R/W	-	0_6A	5-27
	SRC_X	R/W	-	0_61	5-28
	SRC_X_START	R/W	-	0_67	5-29
	SRC_Y	R/W	-	0_62	5-29
	SRC_Y_START	R/W	-	0_68	5-30
	SRC_Y_X	W	-	0_63	5-31
	SRC_Y_X_START	W	-	0_69	5-31
Texture Mapping	S_START	R/W	-	0_D5	6-15
	STW_EXP	R/W	-	0_D1	6-13
	S_X_INC	R/W	-	0_D3	6-15
	S_Y_INC	R/W	-	0_7B,D4	6-15
	TEX_CNTL	R/W	-	0_DD	6-20
	TEX_[0-10]_OFF	R/W	-	0_70-7A	6-13
	TEX_PALETTE	R	-	0_DF	6-25
	TEX_PALETTE_INDEX	R/W	-	0_D0	6-25
	TEX_SIZE_PITCH	R/W	-	0_DC	6-19
Memory Buffer Control	TIMER_CONFIG	R/W	✓	0_0A	4-10
Draw Engine Destination Trajectory	TRAIL_BRES_ERR	R/W	-	0_4E	5-15
	TRAIL_BRES_INC	R/W	-	0_4F	5-15
	TRAIL_BRES_DEC	R/W	-	0_50	5-16
Video Capture	TRIG_CNTL	R/W	-	1_15	7-26
Texture Mapping	T_START	R/W	-	0_DB	6-16

Table 3-2 Cont'd

Registers by Mnemonic

Register Class	Mnemonic	Read/ Write	Block I/O	DWORD Offset (h)	Page
MM Peripheral Port	**TVO_CNTL**	**R/W**	**0_3Fh**	**1_40**	***7-36***
For related registers see Table 3-4	**TV_OUT_DATA (LT only)**	**R/W**	✓	**0_27**	***11-1***
	TV_OUT_INDEX (LT only)	**R/W**	✓	**0_1D**	***11-1***
Texture Mapping	T_X_INC	R/W	-	0_D9	*6-15*
	T_Y_INC	R/W	-	0_DA	*6-16*
Data Path	USR_DST_PITCH	W	-	0_BC	*5-52*
	USR1_DST_0FF_PITCH (LT only)	**W**	**-**	**0_AE**	***5-52***
	USR2_DST_0FF_PITCH (LT only)	**W**	**-**	**0_AF**	***5-52***
Video Capture	VBI_START_END	R/W	-	1_13	*7-31*
	VBI_WIDTH	R/W	-	1_18	*7-31*
Setup Engine	VERTEX_1__ARGB	R/W	-	1_95, BA	*6-33*
	VERTEX_1_S	R/W	-	1_90, AB	*6-31*
	VERTEX_1_SECONDARY_S	R/W	-	1_CA	*6-38*
	VERTEX_1_SECONDARY_T	R/W	-	1_CB	*6-39*
	VERTEX_1_SECONDARY_W	R/W	-	1_CC	*6-39*
	VERTEX_1_SPEC_ARGB	R/W	-	1_93,B4	*6-33*
	VERTEX_1_T	R/W	-	1_91,AC	*6-32*

Table 3-2 Cont'd

Registers by Mnemonic					
Register Class	**Mnemonic**	**Read/ Write**	**Block I/O**	**DWORD Offset (h)**	**Page**
Setup Engine	VERTEX_1_W	R/W	-	1_92,AD	*6-32*
	VERTEX_1_X_Y	R/W	-	1_96,BD	*6-33*
	VERTEX_1_Z	R/W	-	1_94,B7	*6-32*
	VERTEX_2_ARGB	R/W	-	1_9D,BB	*6-35*
	VERTEX_2_S	R/W	-	1_98,AE	*6-34*
	VERTEX_2_SECONDARY_S	R/W	-	1_CD	*6-40*
	VERTEX_2_SECONDARY_T	R/W	-	1_CE	*6-40*
	VERTEX_2_SECONDARY_W	R/W	-	1_CF	*6-41*
	VERTEX_2_SPEC_ARGB	R/W	-	1_9B_B5	*6-35*
	VERTEX_2_T	R/W	-	1_99, AF	*6-34*
	VERTEX_2_W	R/W	-	1_9A,B0	*6-34*
	VERTEX_2_X_Y	R/W	-	1_9E,BE	*6-36*
	VERTEX_2_Z	R/W	-	1_9C,B8	*6-35*
	VERTEX_3_ARGB	R/W	-	1_A5,BC	*6-38*
	VERTEX_3_S	R/W	-	1_A0,B1	*6-36*
	VERTEX_3_SECONDARY_S	R/W	-	1_B0	*6-41*
	VERTEX_3_SECONDARY_T	R/W	-	1_B1	*6-42*
	VERTEX_3_SECONDARY_W	R/W	-	1_B2	*6-42*
	VERTEX_3_SPEC_ARGB	R/W	-	1_A3,B6	*6-37*
	VERTEX_3_T	R/W	-	1_A1,B2	*6-36*
	VERTEX_3_W	R/W	-	1_A2,B3	*6-37*
	VERTEX_3_X_Y	R/W	-	1_A6,BF	*6-38*
	VERTEX_3_Z	R/W	-	1_A4,B9	*6-37*
Memory Buffer Control	VGA_DSP_CONFIG	R/W	✓	0_13	*4-11*
	VGA_DSP_ON_OFF	R/W	✓	0_14	*4-12*
Video Capture	VIDEO_FORMAT	R/W	-	1_12	*7-20*
	VIDEO_SYNC_TEST	R/W	-	1_1A	*7-27*

Table 3-2 Cont'd

Registers by Mnemonic					
Register Class	**Mnemonic**	**Read/ Write**	**Block I/O**	**DWORD Offset (h)**	**Page**
Texture Mapping	W_START	R/W	-	0_D8	*6-14*
	W_X_INC	R/W	-	0_D6	*6-14*
	W_Y_INC	R/W	-	0_D7	*6-14*
Draw Engine Destination Trajectory	Z_CNTL	R/W	-	0_53	*5-17*
	Z_OFF_PITCH	R/W	-	0_52	*5-17*
Specular, Color, Z & Alpha Interpolation	Z_START	R/W	-	0_FB	*6-30*
	Z_X_INC	R/W	-	0_F9	*6-29*
	Z_Y_INC	R/W	-	0_FA	*6-30*

3.4 Sub-Listings

3.4.1 LCD Panel Related Registers

Table 3-3

LCD Registers by Index Number				
Name	**Read/Write**	**Index**	**Offset(h)**	**Page**
LCD_INDEX	R/W		0_29	8-1
CONFIG_PANEL	R/W	0	0_2A	8-2
LCD_GEN_CTRL	R/W	1	0_2A	8-4
DSTN_CONTROL	R/W	2	0_2A	8-7
HFB_PITCH_ADDR	R/W	3	0_2A	8-8
HORZ_STRETCHING	R/W	4	0_2A	8-9
VERT_STRETCHING	R/W	5	0_2A	8-10
EXT_VERT_STRETCH	R/W	6	0_2A	8-11
LT_GIO	R/W	7	0_2A	8-12
POWER_MANAGEMENT	R/W	8	0_2A	8-14
ZVGPIO	R/W	9	0_2A	8-13

3.4.2 TV Out Support Registers

Table 3-4

TV Out Support Registers by Index Number				
Name	**Read/Write**	**Index**	**Offset(h)**	**Page**
TV_OUT_INDEX	R/W		0_1D	11-1
TV_MASTER_CNTL	R/W	10	0_27	11-2
TV_RGB_CNTL	R/W	12	0_27	11-3
TV_SYNC_CNTL	R/W	14	0_27	11-4
TV_HTOTAL	R/W	20	0_27	11-5
TV_HDISP	R/W	21	0_27	11-6
TV_HSIZE	R/W	22	0_27	11-6
TV_HSTART	R/W	23	0_27	11-6
TV_HCOUNT	R	24	0_27	11-6
TV_VTOTAL	R/W	25	0_27	11-7
TV_VDISP	R/W	26	0_27	11-7
TV_VCOUNT	R	27	0_27	11-7
TV_FTOTAL	R/W	28	0_27	11-7
TV_FCOUNT	R	29	0_27	11-8
TV_FRESTART	R/W	2A	0_27	11-8
TV_HRESTART	R/W	2B	0_27	11-8
TV_VRESTART	R/W	2C	0_27	11-8
TV_HOST_READ_DATA	R/W	60	0_27	11-9
TV_HOST_WRITE_DATA	R/W	61	0_27	11-9
TV_HOST_RD_WT_CNTL	R/W	62	0_27	11-9
TV_VSCALER_CNTL	R/W	70	0_27	11-10
TV_TIMING_CNTL	R/W	71	0_27	11-12
TV_GAMMA_CNTL	R/W	72	0_27	11-13
TV_Y_FALL_CNTL	R/W	73	0_27	11-14
TV_Y_RISE_CNTL	R/W	74	0_27	11-15
TV_Y_SAW_TOOTH_CNTL	R/W	75	0_27	11-15
TV_MODULATOR_CNTL1	R/W	80	0_27	11-16

Table 3-4 Cont'd

TV Out Support Registers by Index Number

Name	Read/Write	Index	Offset(h)	Page
TV_MODULATOR_CNTL2	R/W	81	0_27	*11-17*
TV_PRE_DAC_MUX_CNTL	R/W	90	0_27	*11-18*
TV_DAC_CNTL	R/W	A0	0_27	*11-19*
TV_CRC_CNTL	R/W	B0	0_27	*11-20*
TV_VIDEO_PORT_SIG	R/W	B1	0_27	*11-21*
TV_VBI_CC_CNTL	R/W	B8	0_27	*11-21*
TV_VBI_EDS_CNTL	R/W	B9	0_27	*11-22*
TV_VBI_20BIT_CNTL	R/W	BA	0_27	*11-23*
TV_VBI_DTO_CNTL	R/W	BD	0_27	*11-23*
TV_VBI_LEVEL_CNTL	R/W	BE	0_27	*11-24*
TV_UV_ADR	R/W	C0	0_27	*11-24*
TV_FIFO_TEST_CNTL	R/W	C1	0_27	*11-25*

This page intentionally left blank.

Chapter 4
Display and Configuration

For an explanation of the notations used to describe the registers in this and the following chapters, refer to Chapter 1, section *1.3.3*.

4.1 Setup and Control Registers

4.1.1 General I/O Control

	GP_IO																								Offset: 0_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
			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: GPIO(0)**
b	**R/W**	**GP_IO_1**	**Write/Read (DIR1 = output/input) Pin: GPIO(1)**
c	**R/W**	**GP_IO_2**	**Write/Read (DIR2 = output/input) Pin: GPIO(2)**
d	**R/W**	**GP_IO_3**	**Write/Read (DIR3 = output/input) Pin: GPIO(3)**
e	**R/W**	**GP_IO_4**	**Write/Read (DIR4 = output/input) Pin: GPIO(4)**
f	**R/W**	**GP_IO_5**	**Write/Read (DIR5 = output/input) Pin: GPIO(5)**
g	**R/W**	**GP_IO_6**	**Write/Read (DIR6 = output/input) Pin: GPIO(6)**
h	**R/W**	**GP_IO_7**	**Write/Read (DIR7 = output/input) Pin: GPIO(7)**
i	**R/W**	**GP_IO_8**	**Write/Read (DIR8 = output/input) Pin: GPIO(8)**
j	**R/W**	**GP_IO_9**	**Write/Read (DIR9 = output/input) Pin: GPIO(9)**
k	**R/W**	**GP_IO_A**	**Write/Read (DIRA = output/input) Pin: GPIO(10)**
l	**R/W**	**GP_IO_B**	**Write/Read (DIRB = output/input) Pin: GPIO(11)**
m	**R/W**	**GP_IO_C**	**Write/Read (DIRC = output/input) Pin: GPIO(12)**
n	**R/W**	**GP_IO_D**	**Write/Read (DIRD = output/input) Pin: GPIO(13)**
o	R/W	GP_IO_DIR_0	GP IO Direction: 0 = Input 1 = Output (Default = 0)
p	R/W	GP_IO_DIR_1	GP IO Direction: 0 = Input 1 = Output (Default = 0)
q	R/W	GP_IO_DIR_2	GP IO Direction: 0 = Input 1 = Output (Default = 0)

Cont'd	GP_IO																							Offset: 0_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
			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

r	R/W	GP_IO_DIR_3	GP IO Direction: 0 = Input 1 = Output (Default = 0)
s	R/W	GP_IO_DIR_4	GP IO Direction: 0 = Input 1 = Output (Default = 0)
t	R/W	GP_IO_DIR_5	GP IO Direction: 0 = Input 1 = Output (Default = 0)
u	R/W	GP_IO_DIR_6	GP IO Direction: 0 = Input 1 = Output (Default = 0)
v	R/W	GP_IO_DIR_7	GP IO Direction: 0 = Input 1 = Output (Default = 0)
w	R/W	GP_IO_DIR_8	GP IO Direction: 0 = Input 1 = Output (Default = 0)
x	R/W	GP_IO_DIR_9	GP IO Direction: 0 = Input 1 = Output (Default = 0)
y	R/W	GP_IO_DIR_A	GP IO Direction: 0 = Input 1 = Output (Default = 0)
z	R/W	GP_IO_DIR_B	GP IO Direction: 0 = Input 1 = Output (Default = 0)
aa	R/W	GP_IO_DIR_C	GP IO Direction: 0 = Input 1 = Output (Default = 0)
bb	R/W	GP_IO_DIR_D	GP IO Direction: 0 = Input 1 = Output (Default = 0)

Description

This register specifies the data/direction for each pin (GPIO[13:0]) of the General Purpose IO bus. For data/direction for GPIO[46:44], refer to the register *LT_GIO on page 8-12*.

Usage

Refer to the RAGE LT PRO Graphics Controller Specifications for details on the typical pin configurations used to support the various operational modes (VFC, DVS, etc.).

4.1.2 Scratch Pad

	SCRATCH_REG0																			Offset: 0_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
	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 used for BIOS data storage.

Usage

Only the adapter BIOS should use this register.

See Also

SCRATCH_REG1 on *page 4-3*, SCRATCH_REG2 on *page 4-4*, and SCRATCH_REG3 on *page 4-4*.

mach64 Programmer’s Guide:

- *Advanced Topics: Boot-time Initialization*

	SCRATCH_REG1																			Offset: 0_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
	a																															

a	R/W	SCRATCH_REG1	Scratch pad 1

Description

Same as for SCRATCH_REG0 on *page 4-3*.

Usage

Same as for SCRATCH_REG0 on *page 4-3*.

See Also

SCRATCH_REG0 on *page 4-3*, SCRATCH_REG2 on *page 4-4*, and SCRATCH_REG3 on *page 4-4*.

mach64 Programmer's Guide:

- *Advanced Topics: Boot-time Initialization*

	SCRATCH_REG2																								Offset: 0_22							
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
LT	a																															

a	R/W	SCRATCH_REG2	Scratch pad 2

Description

Same as for SCRATCH_REG0 on *page 4-3*.

Usage

Same as for SCRATCH_REG0 on *page 4-3*.

See Also

SCRATCH_REG0 on *page 4-3*, SCRATCH_REG1 on *page 4-3* and SCRATCH_REG3 on *page 4-4*

mach64 Programmer's Guide:

- *Advanced Topics: Boot-time Initialization*

	SCRATCH_REG3																								Offset: 0_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
LT	a																															

a	R/W	SCRATCH_REG3	Scratch pad 3

Description

Same as for SCRATCH_REG0 on *page 4-3*.

Usage

Same as for SCRATCH_REG0 on *page 4-3*.

See Also

SCRATCH_REG0 on *page 4-3*, SCRATCH_REG1 on *page 4-3*, SCRATCH_REG2 on *page 4-4*.

mach64 Programmer’s Guide:

- *Advanced Topics: Boot-time Initialization*

4.1.3 Bus Control

	BUS_CNTL																									**Offset: 0_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
	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	BUS_DBL_RESYNC	Add 1 clock settling time to BCLK and MCLK resynchronizers (default =1)
b	W	BUS_MSTR_RESET	Writing a ‘1’ to this bit resets the bus master. One shot, no need to write 0
c	W	BUS_FLUSH_BUF	Writing a ‘1’ to this bit flushes data from buffer. One shot, no need to write 0.
d	R/W	BUS_STOP_REQ_DIS	Disable burst read requests once stop has been asserted: (default = 0) 0 = Normal 1 = Disable
e	R/W	BUS_APER_REG_DIS	Disable memory mapped register decoding in the linear aperture: 0 = Enable register decoding in linear aperture (default = 0) 1 = Disable register decoding in linear aperture
f	R/W	BUS_EXTRA_PIPE_DIS	0 = Enable extra pipeline stage (default = 0) 1 = Disable extra pipeline stage
g	R/W	BUS_MASTER_DIS	0 =Enable bus master operation 1= Disable bus master operation
h	R/W	ROM_WRT_EN	0 = Disable write to Flash memory BIOS (default = 0) 1 = Enable write to Flash memory BIOS (ROMWRTEN strap must also be enabled)
i	R	MINOR_REV_ID	Additional Minor revision ID bits (read only) from A31 version on. For more info, see *Table 4-1* below under Usage.
j	R/W	BUS_PCI_READ_RETRY_EN	Allow retry for PCI read transfers (default = 0) 0 = normal operation (reads will hold bus until complete) 1 = enable retry cycle in PCI (reads will retry when timeout counter expired)

Cont'd	BUS_CNTL																								Offset: 0_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
	x	w	v	u	t	s	r	q	p	o	n	m	l				k		j			i			h	g	f	e	d	c	b	a

k	R/W	BUS_PCI_WRT_RETRY_EN	Allow retry for PCI write transfers (default = 0) 0 = normal operation (writes will hold bus until complete) 1 = enable retry cycle in PCI (writes will retry when timeout counter expired)
l	R/W	BUS_RETRY_WS	Control value for timeout counter. See *Table 4-2* below under usage for translation to actual wait states.
m	R/W	BUS_MSTR_RD_MULT	Enable 'read multiple' command for bus master (default = 0) 0 = When transfer length > cache line size reg., use 'read line' 1 = When transfer length > cache line size reg., use 'read multiple'
n	R/W	BUS_MSTR_RD_LINE	Enable 'read line' command for bus master (default = 0) 0 = Use 'read command' exclusively 1 = When transfer length > 1, use 'read line' command
o	R/W	BUS_SUSPEND	1 = Suspend the current bus master transfer. This transfer will resume when the bit is cleared. 0 = Resume bus master transfer
p	R	LAT16X	1 = Multiply the latency timer value by 16 0 = Use the latency timer value as is
q	R/W	BUS_RD_DISCARD_EN	1 = Enable PCI slave read data discard (default = 0)
r	R/W	BUS_RD_ABORT_EN	1 = Enable aborting slave's delayed read transaction (for BM conflicts)
s	R/W	BUS_MSTR_WS	Number of wait states to allow until bus master transaction is terminated: 0 = 8 wait states 1 = 32 wait states **Note:** This is valid only when BUS_MSTR_DISCONNECT_EN is enabled
t	R/W	BUS_EXT_REG_EN	Extended Register Block 1 Enable (default = 0): 0 = Disable extended register block 1 1 = Enable extended register block 1
u	R/W	BUS_MSTR_DISCONNECT_EN	1 = Enable bus master disconnect after allowed wait states has elapsed (default = 0)
v	R/W	BUS_WRT_BURST	Enable burst write transfers (default = 0) 0 = write burst transfers disabled 1 = write bursts enabled
w	R/W	BUS_READ_BURST	Enable burst read transfers: 0 = Disabled 1 = Enabled

Cont'd	BUS_CNTL																								Offset: 0_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
	x	w	v	u	t	s	r	q	p	o	n	m	l				k		j			i			h	g	f	e	d	c	b	a

x	R/W	BUS_RDY_READ_DLY	Bus memory read RDY signal delay (default = 1) 0 = no RDY delay 1 = RDY delayed 1 memory clock

Description

BUS_CNTL is used for configuring the on-chip bus interface, controlling bus mastering, and controlling error condition interrupts.

Usage

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

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

Table 4.1 below lists the minor revision id's and

Table 4.2 specifies conversion of the BUS_RETRY_WS field into actual wait states on BCLK when BUS_PCI_WRT_RETRY_EN = 1 or BUS_PCI_READ_RETRY_EN = 1.

Table 4-1 Minor Revision ID's

Minor Revision ID MINOR_REV_ID(4:2) & CFG_CHIP_REV	Description
0 (hex)	first silicon - prototype (A11)
1 (hex)	metal mask spin (A12 & A13)
2 (hex)	all layer spin (A21)
3 (hex)	fast metal spin (A22) - **production part**
7 (hex) - CFG_CHIP_REV is still 3 and we added one more bit from MINOR_REV_ID field	all layer spin (A31)

Table 4-2

BUS_RETRY_WS	Actual Number of Clocks to TRDY or STOP	Time (@33MHz)
0	3	90 ns
1	5	150 ns
2	7	210 ns
3	8	240 ns
4	9	270 ns
5	Bh	330 ns
6	Eh	420 ns
7	Fh	470 ns
8	13h	570 ns
9	35h	1.59 ms
Ah	57h	2.61 µs
Bh	78h	3.6 µs
Ch	99h	4.59 µs
Dh	BBh	5.61 µs
Eh	FFh	7.65 µs
Fh	Infinite	Infinite

Note: First dataphase response = 8 clks + actual clocks from the above table.

See Also

mach64 Programmer’s Guide:

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

4.1.4 Memory Buffer Control

	DSP_CONFIG																							Offset: 0_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
										d			c				b		a													

a	R/W	DSP_XCLKS_PER_QW	Amount of time in XCLKs that one QWORD in the display FIFO occupies
b	R/W	DSP_FLUSH_WB	Flush the write buffer at VSYNC: 0 = Flush at VSYNC, at threshold or on read (default) 1 = Flush at threshold or on read
c	R/W	DSP_LOOP_LATENCY	Display FIFO control parameter
d	R/W	DSP_PRECISION	Integer.fraction precision point for: DSP_XCLKS_PER_QW DSP_ON DSP_OFF

	DSP_ON_OFF																							Offset: 0_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
						b																a										

a	R/W	DSP_OFF	The display memory request off threshold time in terms of XCLKs
b	R/W	DSP_ON	The display memory request on threshold time in terms of XCLKs

The next two registers are used to set DSP parameters for second display.

	DSP2_CONFIG																							Offset: 0_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
LT										c			b						a													

a	R/W	DSP2_XCLKS_PER_QW	Amount of time in XCLKs that one QWORD in the second display FIFO occupies
b	R/W	DSP2_LOOP_LATENCY	Display FIFO control parameter

Cont'd

DSP2_CONFIG Offset: 0_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
LT										c			b						a													

c	R/W	DSP2_PRECISION	Integer. fraction precision point for: DSP2_XCLKS_PER_QW DSP2_ON DSP2_OFF

DSP2_ON_OFF Offset: 0_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
LT						b																a										

a	R/W	DSP2_OFF	The display memory request off threshold time in terms of XCLKs
b	R/W	DSP2_ON	The display memory request on threshold time in terms of XCLKs

TIMER_CONFIG Offset: 0_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
				c													b								a							

a	R/W	VID_INTRA_ACCESS_TIMER	Video intra-access time for memory accesses (in XCLKs)
b	R/W	SCL_INTRA_ACCESS_TIMER	Scaler intra-access time for memory accesses (in XCLKs)
c	R/W	VID_TIMER_MODE	Video timer mode: 0 = Free running (independent) 1 = Relative to end of display burst

<table>
<tr><td></td><td colspan="32"><b>MEM_BUF_CNTL</b> <b>Offset: 0_0B</b></td></tr>
<tr><td>BITS</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td></td><td colspan="8"></td><td>f</td><td></td><td colspan="6">e</td><td colspan="3">d</td><td colspan="5">c</td><td colspan="4">b</td><td colspan="4">a</td></tr>
</table>

a	R/W	Z_WB_FLUSH	Z write buffer flush: 0 = Flush when buffer is full 1-11 = Flush when [1-11] QWORDs are present in write buffer
b	R/W	VID_WB_FLUSH	Video write buffer flush. See also VID_WB_FLUSH_MSB above. 0 = Flush when buffer is full 1-11 = Flush when [1-11] QWORDs are present in write buffer
c	R/W	GUI_WB_FLUSH	GUI write buffer flush: 0 = Flush when buffer is full 1-23 = Flush when [1-23] QWORDs are present in write buffer
d	R/W	HST_WB_FLUSH	Host write buffer flush: 0 = Flush when buffer is full 1-7 = Flush when [1-7] QWORDs are present in write buffer
e	R/W	SCL_THRESH	Scaler threshold: Maximum difference in QWORDs between Y, U and V components before switching
f	W	INVALIDATE_RB_CACHE	Write a ‘1’ to invalidate (clear) the readback cache

<table>
<tr><td></td><td colspan="32"><b>VGA_DSP_CONFIG</b> <b>Offset: 0_13</b></td></tr>
<tr><td>BITS</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td></td><td colspan="5"></td><td colspan="3">c</td><td></td><td colspan="3">b</td><td colspan="6"></td><td colspan="14">a</td></tr>
</table>

a	R/W	VGA_DSP_XCLKS_PER_QW	Amount of time in XCLKs that one QWORD in the display FIFO occupies
b	R/W	VGA_DSP_PREC_PCLKBY2	Integer.fraction precision point for: VGA_DSP_PREC_PCLK+1
c	R/W	VGA_DSP_PREC_PCLK	Integer.fraction precision point for: DSP_XCLKS_PER_QW DSP_ON DSP_OFF

VGA_DSP_ON_OFF **Offset: 0_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
						b																a										

a	R/W	VGA_DSP_OFF	The display memory request off threshold time in terms of XCLKs
b	R/W	VGA_DSP_ON	The display memory request on threshold time in terms of XCLKs

MEM_ADDR_CONFIG **Offset: 0_0D**

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
																							c			b				a		

a	R/W	MEM_ROW_MAPPING	Row Address Mapping: (default = 0) MA: 11 10 9 8 7 6 5 4 3 2 1 0 0 = A22 A21 A11 A20 A19 A18 A17 A16 A15 A14 A13 A12 1 = A22 A11 A21 A20 A19 A18 A17 A16 A15 A14 A13 A12 2 = A11 A22 A21 A20 A19 A18 A17 A16 A15 A14 A13 A22 3 -7= reserved
b	R/W	MEM_COL_MAPPING	Column Address Mapping: (default = 0) MA: 11 10 9 8 7 6 5 4 3 2 1 0 0 = A11 A11 A11 A11 A10 A9 A8 A7 A6 A5 A4 A3 1-7 = reserved
c	R/W	MEM_GROUP_SIZE	Memory group size: 00 - 2MB, 01 - 4MB, 10 - 8MB, 11 - Reserved This register field will be used in 11x8 and 12x8 SD/SGRAM configurations. Otherwise it is 00. For 11x8 SGRAM, the SGRAM's address pins will connect to our chip as follows: SGRAM's A0-A9 <=> RAGE LT PRO's MA0-MA9 SGRAM's A10 <=> RAGE LT PRO’s CS2 For 4 MB with 11x8 SGRAM, SGRAM's CS tied low. For 8 MB with 11x8 SGRAM, SGRAM's CS will be tied to RAGE LT PRO's CS0 and CS1. For 12x8 SDRAM, the SDRAM's address pins will connect to our chip as follows: SGRAM's A0-A9 <=> RAGE LT PRO's MA0-MA9 SGRAM's A10 <=> RAGE LT PRO's CS2 SGRAM's A11 <=> RAGE LT PRO's CS3

Table 4-3 Memory Combinations

Memory Configuration	MEM_ROW_MAPPING	MEM_COL_MAPPING	MEM_GROUP_SIZE
9x9 DRAM	0	0	0

Table 4-3 Memory Combinations Cont'd

Memory Configuration	MEM_ROW_MAPPING	MEM_COL_MAPPING	MEM_GROUP_SIZE
10x8 SGRAM	0	0	0
11x8 SGRAM	1	0	1
12x8 SDRAM	2	0	2

4.1.5 Memory Control

EXT_MEM_CNTL Offset: 0_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
	q	p	o		n						m	l	k	j	i	h	g				f		e		d				c		b	a

a	R/W	MEM_CS	SDRAM command behaviour: 0 = every other clock, commands CS driven 1 = every clock,CS is always active (low).
b	R/W	MEM_SDRAM_RESET	Invokes SDRAM Reset on transition from 0 to 1 0 = Normal 1 = Reset (See note below for resetting SDRAM) * sends sequence to SDRAM consisting of PALL, 8 refresh, MRS After writing a '1', ensure to write a '0' before next reset. This bit has no effect in shared memory configurations.
c	R/W	MEM_CYC_TEST	Invokes memory cycle test mode. **Note that the chip WILL NOT FUNCTION NORMALLY in this mode.** The settings are: 00 = normal operation (default) 01 = Reserved 10 = test mode initiate 11 = test mode sequence run. After test sequence finished, to run another cycle test, this field must be first reset to '10'.

Cont'd

EXT_MEM_CNTL Offset: 0_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
	q	p	o		n						m	l	k	j	i	h	g				f		e		d				c		b	a

d	R/W	MEM_TILE_SELECT	SDRAM Memory Tile Boundary: 0000 = No tiling 0001 = 256 bytes @ 1024 byte pitch 0010 = 512 bytes @ 1024 byte pitch 0011 = reserved 0100 = 128 bytes @ 1kB pitch 0101 = 128 bytes @ 2kB pitch 0110 = 128 bytes @ 4kB pitch 0111 = 128 bytes @ 8kB pitch 1000 = 128 bytes @ 640 byte pitch 1001 = 128 bytes @ 1280 byte pitch 1010 = 128 bytes @ 2560 byte pitch 1011 - 1111 = reserved
e	R/W	MEM_CLK_SELECT	Selects the function of HCLK pin: 00 = SDRAM clock from DLL 01 = (reserved) 10 = XCLK 11 = inverted XCLK
f	R/W	MEM_CAS_LATENCY	SGRAM CAS latency (typically same setting as MEM_LATENCY@MEM_CNTL) For SGRAM only, has no effect for DRAM 00 = 1 clock 01 = 2 clocks (DRAM setting or SDRAM CL=1) 10 = 3 clocks (SDRAM CL = 2) 11 = 4 clocks (SDRAM CL = 3)
g	R/W	MEM_TILE_BOUNDARY	Memory Tile Boundary: Indicates addresses to be tiled according to MEM_TILE_SELECT. 0000 = all memory tiled 0001 - 1111 = addresses below 1/2 MB multiple of this value are tiled
h	R/W	MEM_MDA_DRIVE	Boost drive strength of MD pins not connected to BIOS(0-31, 49-55): 0 = no boost, 1 = boost
i	R/W	MEM_MDB_DRIVE	Boost drive strength of MD pins connected to BIOS(32-48, 56-63): 0 = no boost, 1 = boost
j	R/W	MEM_MDE_DELAY	Delay output of even MD pins: 0 = no delay, 1 = delay
k	R/W	MEM_MDO_DELAY	Delay output of odd MD pins: 0 = no delay, 1 = delay
l	R/W	MEM_MA_DRIVE	Boost drive strength of MA pins: 0 = no boost, 1 = boost

Cont'd **EXT_MEM_CNTL** **Offset: 0_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
	q	p	o		n						m	l	k	j	i	h	g				f		e		d				c		b	a

m	R/W	MEM_MA_DELAY	Delay output of MA pins: 0 = no delay, 1 = delay
n	R/W	MEM_GCMRS	Mode setting for graphics controller (SDRAM): Bit (1:0)00 = Burst length of 1 (not always valid) 01 = Burst length of 2 10 = Burst length of 4 11 = Burst length of 8 Bit (2)0 = Sequential 1 = Interleave Bit (3)0 = Burst read and burst write 1 = Burst read and single write
o	R/W	SDRAM_MEM_CFG	Select configuration of RAS, CAS & CS pins in SDRAM and SGRAM 0 = 2 RAS, 2 CAS, 2 CS 1 = 1 RAS, 1 CAS, 4 CS
p	R/W	MEM_ALL_PAGE_DIS	Controls all page memory cycles 0 = enables 1 = disables
q	R/W	MEM_GROUP_FAULT_EN	Controls page faulting between 2 Meg groups 0 = enables 1 = disables

Usage

Changes in settings to this register will not take affect until MEM_SDRAM_RESET is pulsed (from 0 –>1). The sequence should be as follows:

1. Write EXT_MEM_CNTL with the desired settings, setting MEM_SDRAM_RESET = 0 (use read/modify/write).

2. Rewrite EXT_MEM_CNTL, setting MEM_SDRAM_RESET = 1.

3. Rewrite EXT_MEM_CNTL, setting MEM_SDRAM_RESET = 0 (clear reset bit).

In order to reset SDRAM, the following steps must be performed:

1. Set MEM_SDRAM_RESET to ‘1’

2. Set MEM_CYC_TEST to ‘10’

3. Set MEM_CYC_TEST to ‘11’. Wait at least 3ns.

4. Set MEM_CYC_TEST to ‘00’

5. Set MEM_SDRAM_RESET to ‘0’

MEM_CNTL Offset: 0_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
			o		n		m			l			k	j			i	h	g	f	e		d		c		b		a			

a	R/W	MEM_SIZE	Memory size: 0: Reserved 1: 1 Mbyte 2: Reserved 3: 2 Mbyte 4-6: Reserved 7: 4 Mbyte 9: 6 Mbyte 11: 8 Mbyte 12-14: Reserved 15: 16 Mbyte **Note:** Only above sizes are implemented. Reserved settings are for future use according to the following: (a) Memory sizes from 0-7 increment by 1/2 MB increments (b) Memory sizes from 8-11 increment by 1 MB increments (c) Memory sizes from 12-15 increment by 2 MB increments
b	R/W	MEM_LATENCY	Memory read data latching delay from CAS: (Typically same setting as MEM_CAS_LATENCY) 00 = 1 clock 01 = 2 clocks (DRAM setting or SDRAM CL=1) 10 = 3 clocks (SDRAM CL = 2) 11 = 4 clocks (SDRAM CL = 3)
c	R/W	MEM_LATCH	Memory data latching mechanism: 00 = CAS feedback (DUAL CAS/Fast page mode DRAM) 01 = HCLK feedback 10 = positive edge of XCLK 11 = negative edge of XCLK
d	R/W	MEM_TRP	RAS precharge time, or PRE to ACTV time: 00 = 1 clock 01 = 2 clock 10 = 3 clock 11 = 4 clock
e	R/W	MEM_TRCD	RAS to CAS delay, or ACTV to CMD time: 00 = 1 clock 01 = 2 clock 10 = 3 clock 11 = 4 clock

Cont’d	**MEM_CNTL**																				**Offset: 0_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
			o		n		m			l			k	j			i	h	g	f	e		d		c		b		a			

f	R/W	MEM_TCRD	CAS to RAS delay 0 = no clock delay between CAS high and RAS high 1 = one clock delay between CAS high and RAS high
g	R/W	MEM_TR2W	Read to write delay 0 = 1 clock dead cycle between read to writes 1 = 2 clock dead cycles between read to writes
h	R/W	MEM_CAS_PHASE	For single clock CAS dram modes: 0 = CAS low when clock low 1 = CAS low when clock high
i	R/W	MEM_OE_PULLBACK	Pull back OE after read for DRAM modes 0 = OE goes high 1 clock after CAS high for last read 1 = OE goes high the same time as CAS high for last read
j	R/W	MEM_TRAS	RAS low minimum pulse width, or ACTV to PRE same bank: 000 = 1 clock 001 = 2 clock 010 = 3 clock 011 = 4 clock 100 = 5 clock 101 = 6 clock 110 = 7 clock 111 = 8 clock
k	R/W	MEM_REFRESH_DIS	0 = Enable 1 = Disable refresh Note**:** **MUST** be turned off in shared configurations!
l	R/W	MEM_REFRESH_RATE	Set depending on XCLK frequency: 000 = 10 Mhz - 43 Mhz (1 refresh every 156 XCLK’s) 001 = 44 Mhz - 49 Mhz (1 refresh every 687 XCLK’s) 010 = 50 Mhz - 54 Mhz (1 refresh every 781 XCLK’s) 011 = 55 Mhz - 65 Mhz (1 refresh every 859 XCLK’s) 100 = 66 Mhz - 74 Mhz (1 refresh every 1031 XCLK’s) 101 = 75 Mhz - 79 Mhz (1 refresh every 1171 XCLK’s) 110 = 80 Mhz - 100 Mhz (1 refresh every 1250 XCLK’s) 111 = 100 Mhz and above (1 refresh every 1562 XCLK’s) Note**:** No effect in shared configurations
m	R/W	LOWER_APER_ENDIAN	Lower aperture ‘byte endian’ sense (0-8MB): (default = 0) 00 = Little endian: (no swapping) 01 = Big endian: 16 bpp swapping 10 = Big endian: 32 bpp swapping 11 = (reserved)

Cont'd

MEM_CNTL Offset: 0_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
			o		n		m			l			k	j			i	h	g	f	e		d		c		b		a			

n	R/W	UPPER_APER_ENDIAN	Upper aperture 'byte endian' sense (8MB-16MB): (default = 0) 00 = Little endian: (no swapping) 01 = Big endian: 16 bpp swapping 10 = Big endian: 32 bpp swapping 11 = (reserved)
o	R/W	MEM_PAGE_SIZE	Memory Page Size: (default = 1) 0 = 2K 1 = 4K 2 = 8K 3 = 16K

Description

MEM_CNTL is used 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 Offset: 0_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
									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.

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.

> 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.

See Also

CONFIG_CNTL on *page 4-26*

MEM_VGA_RP_SEL on *page 4-19*

mach64 Programmer's Guide:

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

MEM_VGA_RP_SEL **Offset: 0_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
									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.

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.

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.

See Also

CONFIG_CNTL on *page 4-26*

MEM_VGA_WP_SEL on *page 4-18*

mach64 Programmer's Guide:

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

4.1.6 Test and Debug

	GEN_TEST_CNTL																			Offset: 0_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
	h										g		f						e			d	c	b	a							

a	R/W	GEN_CUR_ENABLE	Enables hardware cursor * (default = 0)
b	R/W	GEN_GUI_RESETB	Resets GUI Engine on high to low transition (default = 0)
c	R/W	GEN_SOFT_RESET	Reset the memory controller: (default = 0) 0 = Normal 1 = Memory controller reset
d	**R/W**	**GEN_MEM_TRISTATE**	**Enables testing of memory interface signals** **0 = normal operation** **1 = tristate memory interface signals**
e	R/W	GEN_TEST_VECT_MODE	Test Vector Mode (default = 0) 0 = Normal 1 = (reserved) 2 = (reserved) 3 = IDDQ Test Mode (I/O pull-ups and pull-downs disabled)
f	R/W	GEN_TEST_MODE	Enable test modes: 0000 = Test mode disabled 0001 = (reserved) 0010 = (reserved) 0011 = (reserved) 0100 = (reserved) 0101 = Video port window test 0110 = Command FIFO test (Lock the FIFO) 0111 = DEBUG mode select (see: GEN_DEBUG_MODE) 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 = (reserved)
g	R/W	GEN_CRC_EN	Enables the CRC signature block (default to 0)

Cont'd

GEN_TEST_CNTL **Offset: 0_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
	h										g		f						e			d	c	b	a							

h	**W**	**GEN_DEBUG_MODE**	**Debug Modes:** **00 = Memory cycle debug** **01 = Display debug** **02 = DRAM state machine debug** **03 = SDRAM state machine debug** **04-0F = (reserved)** **10 = HBIU host request signals** **11 = HBIU slave state machine signals** **12 = HBIU buffer control signals** **13 = HBIU bus master state machine signals** **14 = HBIU bus master status flags** **15-1F = (reserved)** **2x =Power Management state machine debug (from A21 version)** **30 = GUI write buffer** **31 = Host write buffer** **32 = Video write buffer** **33 = GUI read request** **34 = Read buffer bus master** **35-3F = (reserved)** **4x = LCD engine debug** **60-6F = AGP debug** **61-FF = (reserved)** **8x = Palette read/write state machine debug (from A21 version)**

Description

The GEN_TEST_CNTL register is used for general control and diagnostic control. Most of the test modes are only for use during ASIC testing or for debugging purposes. Bit 7 enables the hardware cursor. Bit 8 resets the Draw engine. Bits 16-19 enable various test modes of the ASIC. Bit 21 enables the cyclic redundancy checker (CRC). Bits 24-31 enable various debug modes of the ASIC.

Usage

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_ENABLE.

For GEN_DEBUG_MODE, it is required to set GPIO_DIR1 through GPIO_DIRA to 1's in order to see the selected debug mode.

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*

	CRC_SIG																								Offset: 0_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
									a																							

a	R	CRC_SIG	CRC signature value (Default = 0)

Description

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 4-21*

	HW_DEBUG																								Offset: 0_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
		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	DISP_QW_FIX_DIS	0 = enables extra display qword fix. 1 = disables extra display qword fix.
b	R/W	GUI_BEATS_HOST	0 = normal arbitration between host and gui request channels. 1 = locks out arbitration to host when gui is active
c	R/W	INTER_BLIT_FIX_DIS	0 = enables inter-blit performance improvement 1 = disables inter-blit performance improvement
d	R/W	INTER_PRIM_DIS	0 = enables fast-fill/block-write scissoring 1 = disables fast-fill/block-write scissoring

Cont'd	**HW_DEBUG**																								**Offset: 0_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
		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		

e	R/W	SRC_TRACK_DST_DIS	0 = enables SRC_TRACK_DIS fix 1 = disables SRC_TRACK_DIS fix
f	R/W	AUTO_BLKWRT_COLOR_DIS	0 = enables auto color register updates for block writes 1 = disables auto color register updates for block writes
g	R/W	INTER_LINE_OVERLAP_DIS	0 = enables inter-line overlapping 1 = disables inter-line overlapping
h	R/W	DBL_BUFFER_EN	0 = Double Buffering feature disabled 1 = Double Buffering feature enabled
i	R/W	CMDFIFO_SIZE_MODE_EN	0 = CMDFIFO size change disabled 1 = CMDFIFO sixe change enabled
j	R/W	AUTO_FF_DIS	0 = enables auto-fast-fills 1 = disables auto-fast-fills
k	R/W	AUTO_BLKWRT_DIS	0 = enables auto-block-writes 1 = disables auto-block-writes
l	R/W	ORed_INVLD_RB_CACHE	0 = invalidate the readback cache 1 = invalidate the readback cache (ORed with INVALIDATE_RB_CACHE pulse)
m	R/W	BLOCK_DBL_BUF	Blocks double-buffering of CRTC_OFFSET value if set.
n	R/W	HCLK_FB_SKEW	HCLK feedback skew adjustment for memory read data latching. This register only has an effect when MEM_LATCH@MEM_CNTL = "01" (HCLK feedback). 000 = earliest 111 = latest
o	R/W	DISABLE_SWITCH_FIX	0 = allows switch_en gate level fix (default) 1 = reverts back to the original gate level
p	R/W	SEL_VBLANK_DBL_BUF	0 = VBLANK_BIT2 will output the state of VBLANK 1 = VBLANK_BIT2 will output the state of the CRTC_OFFSET double buffering status.
q	R/W	CMDFIFO_64EN	0 = 64-bits GUI FIFO read disabled 1 = 64-bits GUI FIFO read enabled
r	R/W	BM_FIX_DIS	0 = Enables Bus Master mode 1 = Disables Bus Master mode
s	**R/W**	**Z_SWITCH_EN**	**0 = Do not flush the z-buffer.** **1 = Flush the z-buffer before accepting new data.**
t	R/W	FLUSH_HOST_WB	0 = Not flushing the host write buffer 1 = Flushing the host write buffer

Cont'd

HW_DEBUG Offset: 0_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
		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		

u	R/W	HW_DEBUG_WRITE_MSK_FIX_DIS	0 = Enables write mask fix 1 = Disables write mask fix
v	R/W	Z_NO_WRITE_EN	0 = Do not write to the memory when Z compare fails 1 = Write to memory when Z compare fails
w	R/W	DISABLE_PCLK_RESET (from A21 version)	0 = Generate PCLK reset when memory cntl. reset is generated 1 = Disable PCLK reset when memory cntl. reset is generated
x	R/W	PM_D3_SUPPORT_ENABLE (from A31 version)	0 = Do not generate reset when going from D3 to D0 mode 1 = Generate internal reset when going from D3 to D0 mode
y	R/W	STARTCYCLE_FIX_ENABLE (from A31 version)	0 = Disable STARTCYCLE fix 1 = Enable STARTCYCLE fix in host bus interface
z	R/W	C3_FIX_ENABLE	0 = Disable C3 fix 1 = Enable C3 fix (required for AGP 2X mobile systems)

Description

This register is used for debugging hardware. The BIOS will set this register correctly and no other drivers or applications should use it.

CRT_HORZ_VERT_LOAD MM: 1_51

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
	f	e	d	c				b														a										

a	R/W	VCNTR_VALUE	Vertical count
b	R/W	HCNTR_VALUE	Horziontal count
c	R/W	HCNTR_LOAD	Horizontal count load
d	R/W	VCNTR_LOAD	Vertical count load
e	R/W	EOL_STOP	End of line stop
f	R/W	EOF_STOP	End of field stop

Description

This register affects the CRT controller, but was not put in Block 0 for backward compatibility (there is no block I/O mapping for this register).

Usage

It is only used for ASIC test purposes.

4.1.7 Configuration

	CONFIG_CNTL																								Offset: 0_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
													d						c											b	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 (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

Description

CONFIG_CNTL is used to configure the linear memory aperture and for soft configuration of multiple *mach64* 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 6*). 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.

All offset registers are expanded to allow 8 Mb pointers, with 64-bit granularity. Texture map pointers must have byte granularity.

See Also

mach64 Programmer’s Guide:

- *Advanced Topics: Boot-time Initialization*

<table>
<tr><th></th><th colspan="16">CONFIG_CHIP_ID</th><th colspan="16">Offset: 0_38</th></tr>
<tr><td>BITS</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td></td><td colspan="2">e</td><td colspan="3">d</td><td colspan="3">c</td><td colspan="8">b</td><td colspan="16">a</td></tr>
</table>

a	R	CFG_CHIP_TYPE	Product type code, see DEVICE ID table below
b	R	CFG_CHIP_CLASS	Product class code (0x00)
c	R	CFG_CHIP_VERSION	ASIC Major revision number (0x4)
d	R	CFG_CHIP_FOUNDARY	ASIC Foundary (0x3) (UMC)
e	R	CFG_CHIP_REV	ASIC Minor revision number (0x3 - for A22 part)

Description

CONFIG_CHIP_ID is a read-only register. It returns the revision details of the queried chip. CFG_CHIP_TYPE (Device ID) is an alphanumeric code consisting of two ASCII codes, for example, 4C42h denotes LB (see table below).

The Device ID field also appears in the PCI configuration space (see *Chapter 6*).

Usage

The 16 bits Device ID for the RAGE LT PRO in the PCI addess 2 and CONFIG_CHIP_ID non-GUI register are:

Table 4-4

DEVICE ID	Description
4C42h (LB)	AGP-133/BGA-352 (AGP bus)
4C49h (LI)	PCI-33/BGA-352 (PCI bus)
4C50h (LP)	PCI-33/BGA-256 (PCI bus)

The 8 bits at PCI address 8h are also known as the ASIC ID. The ASIC ID also appears in the CONFIG_CHIP_ID non-GUI register. The following is a list of ASIC IDs used to date:

Table 4-5

ASIC ID	Description	ASIC ID	Description
08h	NEC VT-A3	5Ah	UMC GT-B2U2
48h	NEC VT-A4	9Ah	UMC VT-B2U3
40h	SGS VT-A4	9Ah	UMC GT-B2U3
01h	SGS VT-B1S1	1Bh	UMC R3B/D/P-A1
01h	SGS GT-B1S1	5Bh	UMC R3B/D/P-A2
41h	SGS GT-B1S2	1Ch	UMC R3B/D/P-A3
1Ah	UMC GT-B2U1	5Ch	UMC R3B/D/P-A4

See Also

mach64 Programmer’s Guide:

- *Getting Started: mach64 Detection: Card Detection*

CONFIG_STAT0 **Offset: 0_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
									g		f	e															d	c	b	a		

a	**R/W**	**CFG_MEM_TYPE**	**Memory type:** **000 = Disable memory access** **001 = basic DRAM** **010 = EDO** **011 = hyper page DRAM or EDO** **100 = SDRAM** **101 = SGRAM (default)** **other = reserved**
b	R/W	ROM_REMAP	0 = No remapping (VGA ROM at bottom of ROM) (default) 1 = When VGA disabled, accelerator ROM mapped to first 8K
c	**R/W**	**CFG_VGA_EN**	**0 = Disable VGA** **1 = Enable VGA** **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	PANEL_ID(4:0)	Strap Setting

Cont'd

CONFIG_STAT0 Offset: 0_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
									g		f	e															d	c	b	a		

f	**R**	**MACROVISION_ENABLE**	**Macrovision enabled** **0 - Macrivision disabled** **1 - Macrovision enabled** **Default = Bonding option**
g	**R**	**FULLAGP_STRAP**	**Full AGP enabled** **0 = 256 BGA package** **1 = 312 BGA package** **Default = Bonding option**

Description

This register returns the configuration of the current board.

Usage

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

See Also

mach64 Programmer's Guide:

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

The registers CONFIG_STAT1 and CONFIG_STAT2 below are Read Only and contain the latched value of the MD pins [31:0] and MD pins [63:32] respectively.

CONFIG_STAT1 Offset: 0_25

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
	c								b																a							

a	R	SUBSYS_DEV_ID	PCI subsystem Device ID (See also PCI register 2E)
b	R	SUBSYS_VEN_ID (15:0)	PCI subsystem Vendor ID (See also PCI register 2C-2D)
c	R	DIMM_TYPE	See Intel DIMM spec

CONFIG_STAT2 Offset: 0_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
		o	n		m	l	k		j	i	h		g					f			e			d	c		b					a

a	**R/W**	**DIM_TYPE(3)**	**See Intel DIMM Specifications**
b	R/W	ROMWRTEN	Video BIOS installed in writeable flash RAM
c	R/W	AGPVCOGAIN(1:0)	VCO Gain Default = 00
d	R/W	BUS_TYPE	0 = normal bus type. AGP w/352 BGA and PCI w/256 BGA 1 = reverse bus type. PCI w/352 BGA
e	R/W	AGPSKEW (2:0)	X1 clock phase adjustment with respect to X2, straps from MD[43:41] 000 = 0 taps (default) 001 = 1 taps 010 = 2 taps 011 = 3 taps 100 = 4 taps 101 = 5 taps 110 = 6 taps 111 = 7 taps each tap is worth 0.5 ns roughly
f	R/W	X1CLKSKEW(2:0)	X1 feedback phase adjustment with respect to refclk (cpuclk), straps from MD[46:44] 000 = refclk 1 tap earlier than X1 (feedback) -- default 001 = refclk 2 taps earlier than X1 (feedback) 010 = refclk 3 taps earlier than X1 (feedback) 011 = agp pll testmode, X2 is used as feedback 100 = feedback (X1) 3 taps earlier than refclk 101 = feedback (X1) 2 taps earlier than refclk 110 = feedback (X1) 1 tap earlier than refclk 111 = feedback (X1) and refclk are aligned each tap is worth 0.5 ns roughly
g	**R/W**	**PANEL_ID**	**Panel ID**
h	R/W	PREFETCH_EN	0 = pre-fetch enable 1 = pre-fetch disable
i	R/W	ID_DISABLE	0 = normal operation 1 = IDSEL not connected
j	R/W	PRE_TESTEN	0 = normal operation 1 = test mode condition
k	**R/W**	**PCI5VEN**	**Determines signaling on the PCI bus.** **0 = PCI 3.3V signaling (default)** **1 = PCI 5V signaling**

Cont'd

CONFIG_STAT2 **Offset: 0_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
		o	n		m	l	k		j	i	h		g					f			e			d	c		b					a

l	R/W	VGA_DISABLE	0 = VGA enable 1 = VGA disable
m	R/W	ENINTB	0 = interrupt enable 1 = interrupt disable
n	R/W	ROM_REMAP	0 = No remapping (VGA ROM at bottom of ROM) 1 = When VGA disabled, accelerator ROM mapped to first 8K
o	R/W	IDSEL	0 = connect IDSEL to AD16 1 = connect IDSEL to AD17

4.1.8 Custom Macros

CUSTOM_MACRO_CNTL **Offset: 0_35**

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
																							j	i	h	g	f	e	d	c	b	a

a	R/W	CMD_FIFO_EXTSENSE	Command FIFO mode: 0 = normal 1 = enable extended sense
b	R/W	DSP_FIFO_EXTSENSE	Display FIFO mode: 0 = normal 1 = enable extended sense
c	R/W	RDBUF_FIFO_EXTSENSE	Read buffer FIFO mode: 0 = normal 1 = enable extended sense
d	R/W	WRBUF_FIFO_EXTSENSE	Write buffer FIFO mode: 0 = normal 1 = enable extended sense
e	R/W	GWBUF_FIFO_EXTSENSE	GUI Write buffer FIFO mode: 0 = normal 1 = enable extended sense
f	R/W	CACHE_A_EXTSENSE	Cache A mode: 0 = normal 1 = enable extended sense

Cont'd

CUSTOM_MACRO_CNTL Offset: 0_35

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
																							j	i	h	g	f	e	d	c	b	a

g	R/W	CACHE_B_EXTSENSE	Cache B mode: 0 = normal 1 = enable extended sense
h	R/W	TAG_RAM_EXTSENSE	Tag RAM mode: 0 = normal 1 = enable extended sense
i	R/W	RDRET_FIFO_EXTSENSE	Read Return FIFO mode: 0 = normal 1 = enable extended sense
j	**R/W**	**DSP2_FIFO_EXTENSE**	**Second display FIFO mode:** **0 = normal** **1 = enable extended sense**

4.2 Accelerator CRTC and DAC Registers

4.2.1 Accelerator CRTC

The registers in this group generate the horizontal sync, vertical sync, and blank signals used to position the pixel data on the display monitor. All horizontal parameters are in terms of characters (pixels*8). All vertical parameters are in terms of lines. Accurate display centering is possible by adjusting CRTC_HORZ_SYNC_DLY. A vertical blank and vertical line interrupt allows video synchronization without motion tearing artifacts. Monitor power management is controlled through CRTC_HSYNC_DIS and CRTC_VSYNC_DIS.

Shadowed CRTC Registers

In order to allow applications (i.e., mainly VGA DOS) that were originally written for CRTs to run on LCD panels, certain VGA and accelerator CRTC registers are shadowed, under the control of bit fields in LCD_GEN_CTRL register (see page *8-4*). In some cases, only part of the CRTC register is shadowed, as summarized below:

Shadowed VGA CRTC Registers

Name	I/O	Index	Bits Shadowed
Horizontal Total	03?5*	0	7:0
Horizontal Display Enable End	03?5*	1	7:0
Start Horizontal Blanking	03?5*	2	7:0
End Horizontal Blanking	03?5*	3	6:0
Start Horizontal Retrace	03?5*	4	7:0
End Horizontal Retrace	03?5*	5	7:0
Vertical Total	03?5*	6	7:0
CRTC Overflow	03?5*	7	7:5, 3:0
Maximum Scan Line	03?5*	9	5:0
Cursor Start	03?5*	A	4:0
Cursor End	03?5*	B	4:0
Start Vertical Retrace	03?5*	10	7:0

Shadowed VGA CRTC Registers Cont'd			
Name	**I/O**	**Index**	**Bits Shadowed**
End Vertical Retrace	03?5*	11	3:0
Vertical Display Enable End	03?5*	12	7:0
Underline Location	03?5*	14	4:0
Start Vertical Blanking	03?5*	15	7:0
End Vertical Blanking	03?5*	16	7:0
CRT Mode	03?5*	17	2 (forced to 0) Still read back as 1 if written

Note that the CRTC base address will be 3B4 or 3D4 depending on the I/O Address Select field in the GENMO (Miscellaneous Output Register, 3C2)

Shadowed Accelerator CRTC Registers			
Name	**Offset**	**Bits Shadowed**	**Description**
CRTC_H_TOTAL_DISP	0	24:16, 8:0	Horizontal display total and display end
CRTC_H_SYNC_STRT_WID	1	20:16, 12, 10:0	Horizontal sync start and width
CRTC_V_TOTAL_DISP	2	26:16, 10:0	Vertical display total and display end
CRTC_V_SYNC_STRT_WID	3	20:16, 10:0	Vertical sync start and width
CRTC_VLINE_CRNT_VLINE	4	26:16, 10:0	Current vertical line and vertical line interrupt

The access and the usage of the shadowed CRTC registers are controlled by the following bits inside the LCD_GEN_CTRL register: CRTC_RW_SELECT, SHADOW_RW_EN, USE_SHADOW, USE_SHADOWED_ROWCUR, USE_SHADOWED_VEND and DONT_SHADOW_VPAR.

The following tables show how the shadowed registers are used.

CRTC_RW_SELECT	SHADOW_RW_EN	HOST READ/WRITE
0	0	Host read/writes are directed to the **non-shadowed** CRTC registers
0	1	Host read/writes are directed to the **shadowed** CRTC registers of the primary CRT
1	don't care	Host read/writes are directed to the secondary CRT's registers

USE_SHADOW	DONT_SHADOW_ VPAR	CRTC registers used
1	0	Shadowed CRTC registers are used: VGA CRTC index 15 bits (7:0) (VBLANK start) VGA CRTC index 7 bit 3 (VBLANK start) VGA CRTC index 9 bit 5 (VBLANK start) VGA CRTC index 16 bits (7:0) (VBLANK end) CRTC_V_TOTAL_DISP bits (10:0) (VTOTAL)
	1	Normal CRTC registers are used
don't care	0	Shadowed CRTC registers are used: CRTC_V_SYNC_STRT_WID bits (10:0), (20:16) (VTOTAL)
	1	Normal CRTC registers are used

USE_SHADOW	CRTC registers used	
0	Normal CRTC registers are used instead of the shadowed registers.	
1	The following shadowed registers will be used:	
	Register:	Bits:
	3?5 index 0	7:0
	3?5 index 1	7:0
	3?5 index 2	7:0
	3?5 index 3	6:0
	3?5 index 4	7:0
	3?5 index 5	7:0
	3?5 index 6	7:0
	3?5 index 7	3
	3?5 index 9	5
	3?5 index 10	7:0
	3?5 index 11	3:0
	3?5 index 12	7:0
	3?5 index 15	7:0
	3?5 index 16	7:0
	3?5 index 17	2 (forced to 0) (still reads back a 1 if writen)
	CRTC_H_TOTAL_DISP	24:16, 8:0
	CRTC_H_SYNC_STRT_WID	20:16, 12,10:0
	CRTC_V_TOTAL_DISP **	10:0
	CRTC_V_SYNC_STRT_WID	20:16, 10:0

USE_SHADOWED_ROWCUR	CRTC registers used	
0	Normal CRTC registers: 3?5 index 9, A, B, 14	
1	The following shadowed registers will be used:	
	Register:	Bits:
	3?5 index 9	4:0
	3?5 index A	4:0
	3?5 index B	4:0
	3?5 index 14	4:0

USE_SHADOWED_VEND	CRTC registers used
0	Normal CRTC registers: 3?5 index 12 or CRTC_V_TOTAL_DISP bits 26:16
1	Shadowed register CRTC_V_TOTAL_DISP bits 26:16 are used as vertical display end

Notes:

- When the shadowed registers are used to generate the horizontal timing, the bit SEQ_PCLKBY2 (bit 3 of VGA sequencer data register 1) has the following effect on the timing:

SEQ_PCLKBY2	CRTC registers used
0	The horizontal timing parameters from the shadowed registers are used as programmed.
1	The contents of the following shadow registers are divided by 2 (by the hardware) before they are used: VGA CRTC index 0 bits 7:0 (HTOTAL) VGA CRTC index 1 bits 7:0 (HDISP end) VGA CRTC index 2 bits 7:0 (HBLANK start) VGA CRTC index 3 bits 6:5, 4:0 (HBLANK end) VGA CRTC index 4 bits 7:0 (HSYNC start) VGA CRTC index 5 bits 7, 6:5, 4:0 (HSYNC end) The contents of the following normal registers are divided by 2: VGA CRTC index 5 bits 4:0 (HSYNC end) VGA CRTC index 4 bits 7:0 (HSYNC start)

- When vertical expansion is enabled, the vertical CRT counter will be stalled if a line duplication happens. The vertical display end register should not be shadowed in non-vertical expansion modes to allow maximum VGA compatibility.

  When vertical expansion is enabled, the vertical display end register can either be shadowed or non-shadowed.

  If the vertical end register is non-shadowed, part of the screen will be missing due to the application programming the vertical end to be greater than the vertical end originally written by the video BIOS.

If the vertical end register is shadowed, applications programming a vertical end value other than 350, 400 and 480 will not work properly. To minimize problems with VGA compatibility, there is an option of not to shadow vertical parameters when expansion is turned on. Based the vertical display end value as well as blank start and end values, hardware will select proper ratio for vertical expansion.

- During simultaneous CRT/LCD display, the shadowed H_SYNC/V_SYNC registers should be used to generate H_SYNC/V_SYNC for the LCD panel while the CRT H_SYNC/V_SYNC should be generated by the non-shadowed CRT H_SYNC/V_SYNC registers. This way, both CRT and LCD timing can be met. If vertical parameters are not shadowed, vertical SYNC for the panel should be generated from non-shadow CRT registers.

Second CRTC Registers

Second CRT only supports accelerator mode (no VGA mode support). Ratiometric expansion is not supported. If second CRT is driving LCD panel whose resolution is bigger than current graphics mode, display will be centered.

For the second CRT controller, same address space will be used to access accelerator CRT registers. Only registers necessary to define timing for second CRT controller are duplicated, i.e.:

CRTC2_H_TOTAL_DISP
CRTC2_V_TOTAL_DISP
CRTC2_VLINE_CRNT_VLINE
OVR2_WID_LEFT_RIGHT
CRTC2_H_SYNC_STRT_WID
CRTC2_V_SYNC_STRT_WID
CRTC2_OFF_PITCH
OVR2_WID_TOP_BOTTOM.

Note: Bit CRTC_RW_SELECT in LCD_GEN_CRTL (see page *8-4*) is used to define the active CRT register set.

	CRTC_H_TOTAL_DISP																									Offset: 0_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
									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*

CRTC2_H_TOTAL_DISP **Offset: 0_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
LT								b																a								

a	R/W	CRTC2_H_TOTAL	Horizontal display total (x 8 pixels) for the second display
b	R/W	CRTC2_H_DISP	Horizonyal display end (x 8 pixels) for the second display

Note: Total set for first display when CRTC_RW_SELECT@LCD_GEN_CRTL = 0
Total set for 2nd display when CRTC_RW_SELECT@LCD_GEN_CRTL = 1

CRTC_H_SYNC_STRT_WID **Offset: 0_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
											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*

CRTC2_H_SYNC_STRT_WID Offset: 0_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
LT											e	d								c		b			a							

a	R/W	CRTC2_H_SYNC_STRT	Horizontal sync start (x 8 pixels) for the second display
b	R/W	CRTC2_H_SYNC_DLY	Horizontal sync start delay in pixels for the second display
c	R/W	CRTC2_H_SYNC_STRT_HI	High bit for Horizontal sync start for the second display
d	R/W	CRTC2_H_SYNC_WID	Horizontal sync width (x 8 pixels) for the second display
e	R/W	CRTC2_H_SYNC_POL	Horizontal sync polarity (1-> active low) for the second display

CRTC_V_TOTAL_DISP Offset: 0_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
						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*

CRTC2_V_TOTAL_DISP

Offset: 0_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
LT						b																a										

a	R/W	CRTC2_V_TOTAL	Vertical total for the second display
b	R/W	CRTC2_V_DISP	Vertical display end for the second display

CRTC_V_SYNC_STRT_WID

Offset: 0_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
											c	b										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*

<table>
<tr><th></th><th colspan="26">CRTC2_V_SYNC_STRT_WID</th><th colspan="6">Offset: 0_03</th></tr>
<tr><td>BITS</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td><b>LT</b></td><td colspan="10"></td><td>c</td><td colspan="5">b</td><td colspan="5"></td><td colspan="11">a</td></tr>
</table>

a	R/W	CRTC2_V_SYNC_STRT	Vertical sync start for second display
b	R/W	CRTC2_V_SYNC_WID	Vertical sync width for second display
a	R/W	CRTC2_V_SYNC_POL	Vertical sync polarity (1 -> active low) for second display

<table>
<tr><th></th><th colspan="26">CRTC_VLINE_CRNT_VLINE</th><th colspan="6">Offset: 0_04</th></tr>
<tr><td>BITS</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td></td><td colspan="5"></td><td colspan="11">b</td><td colspan="5"></td><td colspan="11">a</td></tr>
</table>

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 4-44*

mach64 Programmer’s Guide:

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

<table>
<tr><td></td><td colspan="24"><b>CRTC2_VLINE_CRNT_VLINE</b></td><td colspan="8"><b>Offset: 0_04</b></td></tr>
<tr><td>BITS</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td><b>LT</b></td><td colspan="5"></td><td colspan="11">b</td><td colspan="5"></td><td colspan="11">a</td></tr>
</table>

a	R/W	CRTC2_VLINE	Vertical line of the second CRT at which vertical line interrupt is triggerd.
b	**R**	CRTC2_CRNT_VLINE	Current vertical line of the second CRT controller.

<table>
<tr><td></td><td colspan="24"><b>CRTC_OFF_PITCH</b></td><td colspan="8"><b>Offset: 0_05</b></td></tr>
<tr><td>BITS</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td></td><td colspan="10">b</td><td colspan="2"></td><td colspan="20">a</td></tr>
</table>

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 4-16*

SRC_OFF_PITCH on *page 5-26*

DST_OFF_PITCH on *page 5-9*

mach64 Programmer’s Guide:

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

	CRTC2_OFF_PITCH																								Offset: 0_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
LT	b												a																			

a	R/W	CRTC2_OFFSET	Display address offset in terms of 64 bit words for second display path
b	R/W	CRTC2_PITCH	Display pitch in pixels x8 for second display path

Description:

This register is used to define the start address for the second display.

	CRTC_INT_CNTL																								Offset: 0_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
	B	A	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	CRTC_VBLANK	Vertical blank
b	R/W	CRTC_VBLANK_INT_EN *	Vertical blank interrupt enable (active high) Default = 0
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 (active high) Default = 0
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

Cont'd

CRTC_INT_CNTL **Offset: 0_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
	B	A	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

g	R	CRTC_FRAME	Interlace odd/even frame: 0 = even frame 1 = odd frame
h	**R**	**CRTC2_VBLANK**	**Vertical blank of the second CRT**
i	**R/W**	**CRTC2_VBLANK_INT_EN ***	**Vertical blank interrupt enable for second CRT, active high** **Default = 0**
j	**R**	**CRTC2_VBLANK_INT ***	**Vertical blank interrupt of the second CRT, active high**
	W	**CRTC2_VBLANK_INT_AK ***	**Vertical blank acknowledge for the second CRT,** **1 -> clears interrupt**
k	**R/W**	**CRTC2_VLINE_INT_EN ***	**Vertical line interrupt enable for the second CRT**
l	**R**	**CRTC2_VLINE_INT ***	**Vertical line interrupt of the second CRT, active high**
	W	**CRTC2_VLINE_INT_AK ***	**Vertical line acknowledge for the second CRT,** **1 -> clears interrupt**
m	R/W	CUPBUF0_INT_EN	Continuous capture buffer 0 interupt enable
n	R	CUPBUF0_INT *	Continuous capture buffer 0 interrupt, active high
	W	CUPBUF0_INT_AK *	Continuous capture buffer 0, 1 -> cleares interrupt
o	R/W	CUPBUF1_INT_EN	Continuous capture buffer 1 interrupt enable
p	R	CUPBUF1_INT *	Continuous capture buffer 1 interrupt, active high
	W	CUPBUF1_INT_AK *	Continuous capture buffer 1 acknowledge, 1 -> clears interupt
q	R/W	OVERLAY_EOF_INT_EN	Overlay end-of-frame interrupt enable
r	R	OVERLAY_EOF_INT *	Overlay end-of-frame interrupt, active high
	W	OVERLAY_EOF_INT_AK *	Overlay end-of-frame acknowledge, 1 -> clears interrupt
s	R/W	ONESHOT_CAP_INT_EN	One-shot host capture copmlete interrupt enable
t	R	ONESHOT_CAP_INT *	One-shot host capture complete interrupt, active high
	W	ONESHOT_CAP_INT_AK *	One-shot host capture complete acknowledge, 1 -> clears interrupt
u	R/W	BUSMASTER_EOL_INT_EN	Bus master end of system list interrupt enable

Cont'd

CRTC_INT_CNTL **Offset: 0_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
	B	A	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

v	R	BUSMASTER_EOL_INT *	Bus master end of system list interrupt, active high
	W	BUSMASTER_EOL_INT_AK *	Bus master end of system list acknowledge, 1 -> clears interrupt
w	R/W	GP_INT_EN	General Purpose I/O interrupt enable
x	R	GP_INT *	General Purpose I/O interrupt
	W	GP_INT_AK *	General Purpose I/O acknowledge, 1 -> clears interrupt
y	**R**	**CRTC2_VLINE_SYNC**	**Vertical line sync of the second CRT** **0 = even scan line** **1 = odd scan line**
z	**R/W**	**SNAPSHOT2_INT_EN ***	**Snapshot interrupt enable. active high** **Default = 0**
A	**R**	**SNAPSHOT2_INT ***	**Snapshot interrupt, active high**
	W	**SNAPSHOT2_INT_AK ***	**Snapshot interrupt acknowledge,** **1 -> clears interrupt**
B	R	VBLANK_BIT2_INT *	Secondary VBLZNK bit interrrupt
	W	VBLANK_BIT2_INT_AK *	Secondary VBLZNK bit acknowledge

*Two separate writes are required to program this register correctly. The first write should be used to clear the appropriate interrupts (prior to enabling), the second write should then be used to enable the interrupts. Clearing and enabling ofinterrupts should **not** be attempted in a single write.

Description

CRTC_INT_CNTL is used for enabling and acknowledging interrupts generated by the accelerator CRTC, video capture and overlay display, and for 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 4-42*

mach64 Programmer’s Guide:

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

	CRTC_GEN_CNTL																										**Offset: 0_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
	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**	**CRTC2_DBL_SCAN_EN**	**Enables double scan on the secondary display**
g	R/W	CRTC_DISPLAY_DIS	Disables the display, forcing the blanking signal to be active
h	R/W	CRTC_VGA_XOVERSCAN	0 = disables overscan in VGA mode 1 = enables overscan in VGA mode
i	R/W	CRTC_PIX_WIDTH	Display pixel width: 0 = (reserved) 1 = 4 bpp pseudo (DAC_DIRECT must be 0) 2 = 8 bpp pseudo when DAC_DIRECT = 0, 8 bpp (3,3,2) when DAC_DIRECT = 1 3 = 15 bpp (5,5,5) 4 = 16 bpp (5,6,5) 5 = 24 bpp (8,8,8) 6 = 32 bpp (a,8,8,8) 7 = (reserved)
j	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.

Cont'd

CRTC_GEN_CNTL Offset: 0_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
	w	v	u	t	s	r	q	p	o	n	m	l	k								j	i			h	g	f	e	d	c	b	a

k	**R/W**	**CRTC2_PIX_WIDTH**	**Second display pixel width:** **0 = disable second CRT** **1 = 8 bpp** **2 = Reserved** **3 = 15 bpp (5,5,5)** **4 = 16 bpp (5,6,5)** **5 = 24 bpp (8,8,8)** **6 = 32 bpp (a,8,8,8)** **7 = YUV422 (for YUV direct mode to TV Out)**
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**	**CRTC2_ENABLE**	**Enables second CRT controller: (default = 0)** **0 = resets CRTC2** **1 = enables CRTC2**
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
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

Cont'd

CRTC_GEN_CNTL Offset: 0_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
	w	v	u	t	s	r	q	p	o	n	m	l	k								j	i			h	g	f	e	d	c	b	a

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*

CRT_TRAP **Offset: 0_0E**

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
	c		b							a																						

a	R/W	CRT_TRAP_BASE_ADDR	Base address bits [22:12] for writing CRT trapped addresses **Note:** This implies the address is 4K aligned
b	R	DAC_RGB_STATE	Status of RGB DAC index reads/writes: 00 = all RGB colours written 01 = Red colour index written (await green/blue) 10 = Green colour index written (await blue) 11 = reserved
c	R/W	CRT_TRAP_EN	0 = Disable VGA CRT register trapping 1 = Enable VGA CRT register trapping

Usage

This register is used to trap all accesses through VGA I/O space to VGA CRTC registers or DAC pallete when trapping is enabled.

4.2.2 Overscan

In order to do centering on the high resolution LCD panels, the width for Left, Right, Top and Bottom Overscan is increased (the field names have not changed but their bit sizes have).

Similar to the case of Second CRTC registers, the two Overscan registers, OVER_WID_LEFT_RIGHT and OVER_WID_TOP_BOTTOM, and OVR_CLR are duplicated and accessed using CRTC_RW_SELECT@LCD_GEN_CRTL = 1

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 RAGE LT PRO. 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																								**Offset: 0_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
	c								b								a															

a	R/W	OVR_CLR_B	Blue overscan color, to internal DAC
b	R/W	OVR_CLR_G	Green overscan color, to internal DAC
c	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 4-54*

mach64 Programmer's Guide:

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

	OVR2_CLR																								**Offset: 0_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
LT	c								b								a															

a	R/W	OVR2_CLR_B	Blue overscan color for second display
b	R/W	OVR2_CLR_G	Green overscan color for second display
c	R/W	OVR2_CLR_R	Red overscan color for second display

	OVR_WID_LEFT_RIGHT																								**Offset: 0_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
											b																a					

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

OVR_WID_LEFT_RIGHT Offset: 0_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
											b																a					

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

OVR_WID_TOP_BOTTOM on *page 4-52*

mach64 Programmer’s Guide:

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

OVR2_WID_LEFT_RIGHT Offset: 0_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
LT											b																a					

a	R/W	OVR2_WID_LEFT	Left overscan width (in 8*pixels) for second display path
b	R/W	OVR2_WID_RIGHT	Right overscan width (in 8*pixels) for second display path

OVR_WID_TOP_BOTTOM Offset: 0_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
								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

OVR_WID_LEFT_RIGHT on *page 4-51*

mach64 Programmer’s Guide:

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

OVR2_WID_TOP_BOTTOM **Offset: 0_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
LT								b																a								

a	R/W	OVR2_WID_TOP	Top overscan width (in scan lines) for second display path
b	R/W	OVR2_WID_BOTTOM	Bottom overscan width (in scan lines) for second display path

4.2.3 Hardware Cursor

The hardware cursor and the hardware icon are only supported on the primary display and **not** on the secondary display.

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 hardware icon is very similar to the hardware cursor except that it can be up to 128x128 pixels.The icon definition is in reverse pixel order within each byte. We can have either hardware icon or hardware cursor. When the ICON_ENABLE bit is set, the hardware cursor becomes a hardware icon and the hardware cursor registers should be interpreted as hardware icon registers.

The hardware icon and the hardware cursor cannot be used at the same time. It is expected that the hardware icon being used in VGA compatible modes as a power meter type of applications. Although the hardware icon can be moved around after it is setup, it is expected that it will not be moved around frequently like the hardware cursor. In Windows modes, the hardware cursor will be active and a software icon will be used instead.

Software icon is used in all extended modes.

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.

The two hardware cursor colors are 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 Bedrock. Note this is always a true color that is not mapped through the palette. The 8 bit index color is used in 4 & 8bpp modes for data going out on the 8 bit feature connector. The receiving board is expected to index all 4 & 8bpp data through its own palette.

CUR_CLR0 Offset: 0_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
	c								b								a															

a	R/W	CUR_CLR0_B	Blue cursor color 0, to internal DAC
b	R/W	CUR_CLR0_G	Green cursor color 0, to internal DAC
c	R/W	CUR_CLR0_R	Red cursor color 0, to internal DAC

Description

The two hardware cursor colors are defined by an 8-bit index and a 24-bit color. CUR_CLR0 contains color 0 for the hardware cursor. Cursor color 0 going to the internal DAC is 24 bits (CUR_CLR0_R, CUR_CLR0_G, CUR_CLR0_B) in all display modes.

The receiving board is expected to index all 4 bpp and 8 bpp data through its own palette.

The color of the cursor pixel is defined by two bits as follows:

Table 4-6

2-Bit Pixel Value	Pixel Color
00	Cursor Color 0
01	Cursor Color 1
10	Transparent (current display pixel)
11	Complement (1's complement of current display pixel)

Usage

This register is used when defining the hardware cursor attributes.

See Also

CUR_CLR1 below.

mach64 Programmer's Guide:

- *Engine Operations: Miscellaneous Operations: Hardware Cursor*

CUR_CLR1 **Offset: 0_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
	c								b								a															

a	R/W	CUR_CLR1_B	Blue cursor color 1, to internal DAC
b	R/W	CUR_CLR1_G	Green cursor color 1, to internal DAC
c	R/W	CUR_CLR1_R	Red cursor color 1, to internal DAC

Description

See CUR_CLR0 above

Usage

See CUR_CLR0 above

See Also

CUR_CLR0 above.

	CUR_OFFSET																							Offset: 0_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
													a																			

a	R/W	CUR_OFFSET	Cursor/icon 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																							Offset: 0_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
						b																a										

a	R/W	CUR_HORZ_POSN	Cursor/icon horizontal position
b	R/W	CUR_VERT_POSN	Cursor/icon 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																									**Offset: 0_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
										d	c															b	a					

a	R/W	CUR_HORZ_OFF (5:0)	Cursor horizontal offset / Icon horizontal offset (5:0)
b	**R/W**	**CUR_HORZ_OFF (6)**	**Icon horizontal offset (6)**
c	R/W	CUR_VERT_OFF (5:0)	Cursor vertical offset / Icon vertical offset (5:0)
d	**R/W**	**CUR_VERT_OFF (6)**	**Icon vertical offset (6)**

Description

CUR_HORZ_VERT_OFF specifies the offsets from the 64x64 cursor definition block where the cursor definition area is to begin.

Each cursor offset should be set such that offset = 64 - size, and each icon offset should be set such that offset = 128 - size

Usage

This register is used when defining the hardware cursor attributes.

See Also

mach64 Programmer’s Guide:

- *Engine Operations: Miscellaneous Operations: Hardware Cursor*

4.2.4 GenLocking (CRT-Sync to Video)

The GenLocking registers are used to indicate the *snapshot* values of the CRTC horizontal/vertical count, video frame and CRT frame count, captured automatically or manually. These values are used for calculating the frame rate drift between CRTC and Video frames, which in turn is used to reprogram PLL M and N (CLKBLK) to synchronize the frame rates.

SNAPSHOT_VH_COUNTS Offset: 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
								b														a										

a	R	SNAPSHOT_VCOUNT	Snapshot of CRTC horizontal count value.
b	R	SNAPSHOT_HCOUNT	Snapshot of CRTC vertical count value.

SNAPSHOT2_VH_COUNTS 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
LT								b														a										

a	R	SNAPSHOT2_VCOUNT	Snapshot of second CRTC horizontal count value
b	R	SNAPSHOT2_HCOUNT	Snapshot of second CRTC vertical count value

SNAPSHOT_F_COUNT Offset: 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
												a																				

a	R	SNAPSHOT_F_COUNT	Snapshot of CRTC frame count value.

SNAPSHOT2_F_COUNT MM: 1_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
LT												a																				

a	R	SNAPSHOT2_F_COUNT	Snapshot of second CRTC frame count value.

	N_VIF_COUNT																						Offset: 1_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
																							a									

a	R/W	N_VIF_COUNT_VAL	Programmable N-Video-in-Field count value used to generate a snapshot interrupt when this N-count value is equal to the count value of the lower 10 bit SNAPSHOT_VIF_COUNT (Refer to CRTC_INT_CNTL [8:7] for the snapshot interrupt specification)

Usage

N_VIF_COUNT is pre-programmed to non-zero 'N' of the Video-in-Field to capture count values automatically. When this non-zero N value is expired, an interrupt is triggered to indicate an auto-snapshot has been taken.

	N_VIF2_COUNT																						MM: 1_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
LT																							a									

a	R/W	N_VIF2_COUNT_VAL	Programmable N-video-in-field count value which is used to generate a snapshot interrupt when this N-count value is equal to the count value of the lower 10-bit SNAPSHOT_VIF_COUNT

See also:

CRTC_INT_CNTL register (*page 4-44*) , bits [8:7] - 0_06 for the snapshot interrupt specification.

	SNAPSHOT_VIF_COUNT																						Offset: 1_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
							d	c				b											a									

a	R	LSNAPSHOT_VIF_COUNT	Lower snapshot of Video_in_Field count value. (Lower 10 bits [9:0] indicate the current number of frames accumulated.)

	SNAPSHOT_VIF_COUNT																								Offset: 1_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
							d	c				b											a									

b	R	USNAPSHOT_VIF_COUNT	Upper snapshot of Video_in_Field count value. (Upper 11 bits [20:10] indicate the current number of N-frames.)
c	R/W	AUTO_SNAPSHOT_TAKEN	0 = writing ‘0’ enables both auto and manual snapshot taking and clears internal SNAPSHOT_F_COUNT and SNAPSHOT_VIF_COUNT counters (**W**). 1 = indicates that a snapshot has been taken (**R**)
d	W	MANUAL_SNAPSHOT_NOW	1 = snapshot taken immediately (writing ‘1’ to this bit prevents all auto-snapshot taking until a write of ‘0’ to the AUTO_SNAPSHOT_TAKEN bit which will re-enable the auto-snapshot taking.)

Usage

To calculate the total number of Video-in-Field frames, use the equation:

$$\textit{Total VIF frame count} = (\textit{N\_VIF\_COUNT\_VAL} * \textit{USNAPSHOT\_VIF\_COUNT}) + \textit{LSNAPSHOT\_VIF\_COUNT}$$

	SNAPSHOT2_VIF_COUNT																								MM: 1_2F							
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
LT							c	b				a																				

a	R	SNAPSHOT2_VIF_COUNT	Snapshot of Video-in-field count value: Lower 10-bit [[9:0] indicates the current number of frames accumulated. Upper 11-bit [20:10] indicates the number of N-frames. Total frame count = (N_VIF2_COUNT_VAL x Upper SNAPSHOT2_VIF_COUNT) + Lower SNAPSHOT2_VIF_COUNT
b	W	AUTO_SNAPSHOT2_TAKEN	0 = writing ‘0’ enables both auto and manual snapshot taking, and clears SNAPSHOT2_F_COUNT, SNAPSHOT2_VIF_COUNT counters
	R		1 = indicates that a snapshot for second CRT has been taken
c	W	MANUAL_SNAPSHOT2_NOW	1 = Snapshot taken immediately (writing ‘1’ to this bit prevents all auto-snapshot taking until a writing of ‘0’ to the AUTO_SNAPSHOT2_TAKEN bit that will re-enable the auto-snapshot taking)

4.2.5 Clock Control

	CLOCK_CNTL																							Offset: 0_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
									e								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_ADDR(5)**	**PLL register select bit 5**
e	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 (or PLL registers which are described next). The internal clock synthesizer 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*

4.2.6 PLL Control

The PLL registers are accessed indirectly through the CLOCK_CNTL register described above. 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
```

Notes:

- PLL registers 0 and 1 control gain and duty cycle of analog PLL's. Gain bits affect lock and jitter of PLL's. These registers should only be adjusted by the BIOS.
- Oscillator is always turned on regardless of whether 14.3 or 29.5 MHz crsytal is being used.
- 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.
- PLL_TEST_CTRL and PLL_TEST_COUNT are used only during manufacturing tests of analog PLL's.

	MPLL_CNTL					Addr: 0		
BITS	7	6	5	4	3	2	1	0
	d	c		b		a		

a	R/W	MPLL_PC_GAIN	MPLL Charge-pump gain setting
b	R/W	MPLL_VC_GAIN	MPLL VCGEN gain setting
c	R/W	MPLL_D_CYC	Duty cycle control for MPLL
d	R/W	MPLL_RANGE	MPLL range control

Description

Controls to MPLL analog macro.

Usage

Do not change the default set by BIOS.

	VPLL_CNTL					Addr: 1		
BITS	7	6	5	4	3	2	1	0
	d	c		b		a		

a	R/W	VPLL_PC_GAIN	VPLL Charge-pump gain setting
b	R/W	VPLL_VC_GAIN	VPLL VCGEN gain setting
c	R/W	VPLL_D_CYC	Duty cycle control for VPLL
d	R/W	VPLL_RANGE	VPLL range control

Description

Controls to VPLL analog macro.

Usage

Do not change the default set by BIOS.

	PLL_REF_DIV						Addr: 2	
BITS	7	6	5	4	3	2	1	0
	a							

a	R/W	PLL_REV_DIV	Common reference setting for MPLL and VPLL (default = 24h)

	PLL_GEN_CNTL						Addr: 3	
BITS	7	6	5	4	3	2	1	0
	f	e			d	c	b	a

a	R/W	PLL_SLEEP	1 : Power-down MPLL and VPLL
b	R/W	PLL_MRESET	1 : Reset MPLL
c	R/W	OSC_EN	1 : Oscillator enable
d	R/W	EXT_CLK_EN	1 : Force DCLK pin output to tri-state
e	R/W	MCLK_SRC_SEL	MCLK is GUI and 3D Engine clock 000 : MCLK = PLLMCLK (MPLL primary output) 001 : MCLK = PLLMCLK/2 010 : MCLK = PLLMCLK/4 011 : MCLK = PLLMCLK/8 100 : MCLK = PLLMCLK/3 101 : MCLK = CPUCLK 110 : MCLK = SCLK HCLK (direct, no DLL) 111 : MCLK = XTALIN
f	R/W	DLL_PWDN	0 : DLL Enabled 1 : DLL Powerdown. For full powerdown DLL_GAIN=10 must also be set

Description

PLL and DLL general controls and MCLK source and/or post divider selection. (Default = CFh)

	MCLK_FB_DIV					Addr: 4		
BITS	7	6	5	4	3	2	1	0
	a							

a	R/W	MCLK_FB_DIV	MPLL feedback divider (default = F6h, 50 MHz)

Description

MPLL feedback divider.

Usage

MPLL output (PLLMCLK) will run at:

*PLLMCLK = (XTALIN * x * MCLK_FB_DIV)/(PLL_REF_DIV)*

where *x* is either 2 or 4 as set by MFB_TIMES_4_2b@PLL_EXT_CNTL

	PLL_VCLK_CNTL					Addr: 5		
BITS	7	6	5	4	3	2	1	0
	e		d		c	b	a	

a	R/W	VCLK_SRC_SEL	VLCK is the pixel clock 00 : VCLK_SRC = CPUCLK 01 : VCLK_SRC = DCLK 10 : VCLK_SRC = GIO(1) 11 : VCLK_SRC = PLLVCLK (VPLL primary output) VCLK = VCLK_SRC / VCLKx_POST
b	R/W	PLL_PRESET	1 : Reset VCLK PLL
c	R/W	VCLK_INVERT	1 : Invert VCLK to get opposite duty cycle
d	R/W	ECP_DIV	ECP is the Scaler/Ovelay clock 00 : ECP = VCLK 01 : ECP = VCLK/2 10 : ECP = VCLK/4 11 : reserved
e		(scratch)	Reserved for future use

Description

Pixel clock control (default = 04h)

Usage

Display will stop and the system will hang if anything is done to stop VLCK. Switch VCLK_SRC_SEL to another source before resetting or stopping PLLVCLK or XTALIN.

VCLK_POST_DIV Addr: 6

BITS	7	6	5	4	3	2	1	0
	d		c		b		a	

a	R/W	VLCK0_POST	Lower bits of post divider for VCLK0
b	R/W	VLCK1_POST	Lower bits of post divider for VCLK1
c	R/W	VLCK2_POST	Lower bits of post divider for VCLK2
d	R/W	VLCK3_POST	Lower bits of post divider for VCLK3

Description

Lower bits of post dividers for VCLK 0-3 (default = 00h)

Usage

Post divider selection made by VGA_CKSEL@GENM0 in VGA mode (see page *7-23*) or by CLOCK_SEL@ CLOCK_CNTL (see page *4-61*) in extended display modes.

VCLK0_FB_DIV Addr: 7

BITS	7	6	5	4	3	2	1	0
	a							

a	R/W	VCLK0_FB_DIV	Feedback divider for VCLK0 (default = FDh)

VCLK1_FB_DIV Addr: 8

BITS	7	6	5	4	3	2	1	0
	a							

a	R/W	VCLK1_FB_DIV	Feedback divider for VCLK1 (default = 8Eh)

	VCLK2_FB_DIV						Addr: 9	
BITS	7	6	5	4	3	2	1	0
	a							

a	R/W	VCLK2_FB_DIV	Feedback divider for VCLK2 (default = 9Eh)

	VCLK3_FB_DIV						Addr: 10	
BITS	7	6	5	4	3	2	1	0
	a							

a	R/W	VCLK3_FB_DIV	Feedback divider for VCLK3 (default =65h)

Description

VPLL feedback divider. Selection made by VGA_CKSEL@GENM0 in VGA mode (see page 7-23) or by CLOCK_SEL@ CLOCK_CNTL (see page 4-61) in extended display modes.

Usage

VPLL output (PLLVCLK) will run at:

*PLLVCLK = XTALIN * 2 * VCLKx_FB_DIV)/PLL_REF_DIV*

where VCLKx can be VCLK1, VCLK2 or VCLK3.

	PLL_EXT_CNTL						Addr: 11	
BITS	7	6	5	4	3	2	1	0
	f	e	d	c	b	a		

a	R/W	XCLK_SRC_SEL	XCLK is the memory interface clock 000 : XCLK = PLLMCLK (MPLL primary output) 001 : XCLK = PLLMCLK/2 010 : XCLK = PLLMCLK/4 011 : XCLK = PLLMCLK/8 100 : XCLK = PLLMCLK/3 101 : XCLK = CPUCLK 110 : XCLK = HCLK (direct, no DLL) 111 : XCLK = DLL_CLK

Cont'd	PLL_EXT_CNTL						Addr: 11	
BITS	7	6	5	4	3	2	1	0
	f	e	d	c	b	a		

b	R/W	MFB_TIMES_4_2b	Selects ratio of MCLK_FB_DIV to effective feedback value: 0 : PLLMCLK feedback = 2 * MCLK_FB_DIV 1 : PLLMCLK feedback = 4 * MCLK_FB_DIV
c	R/W	ALT_VCLK0_POST	Select alternate post dividers for VCLK0
d	R/W	ALT_VCLK1_POST	Select alternate post dividers for VCLK1
e	R/W	ALT_VCLK2_POST	Select alternate post dividers for VCLK2
f	R/W	ALT_VCLK3_POST	Select alternate post dividers for VCLK3

Description

Extended control of XCLK & VCLK (default = 05h)

	DLL_CNTL						Addr: 12	
BITS	7	6	5	4	3	2	1	0
	c	b					a	

a	R/W	DLL_REF_SRC	Selection of source for DLL_REF_CLK 00 : DLL_REF_CLK stopped 01 : DLL_REF_CLK = HCLK input 10 : DLL_REF_CLK = XCLK 11 : reserved
b	R/W	DLL_RESET	DLL reset control. DLL resets on rising edge of this signal if DLL_REF_CLK is running.
c	R/W	HCLK_OUT_EN	0 : HCLK forced to tri-state 1 : HCLK output enabled

Description

Controls XCLK DLL (default = 40h)

Usage

When using SDRAM/SGRAM, the DLL phase locks the external version of the memory clock to the internal XCLK.

	VFC_CNTL					**Addr: 13**		
BITS	7	6	5	4	3	2	1	0
	f		e	d		c	b	a

a	R/W	DCLK_INVERTb	0: PIXEL data and BLANKB off DCLK falling edge 1: PIXEL data and BLANKB off DCLK rising edge
b	R/W	DCLKBY2_EN	DCLK selection for VGA Modes: 0,1,4,5,D,13 (no affect in other display modes) 0: DCLK = 2 x VGA default 1: DCLK = VGA default
c	R/W	VFC_MULT_EN	True colour mode selection (no affect in VGA, 4bpp or 8bpp) 0: Single clock VFC (DCLK = VCLK) 1: Multi clock VFC (DCLK set by pixel depth and VCLK post divider)
d	R/W	VFC_DELAY	PIXEL and BLANKB hold adjustment 00: least delay to 11: most delay
e	R/W	DCLKBY2_SHIFT	Shift DCLK by 1/4 period in VGA modes 0,1,4,5,D,13 and DCLKBY2_EN active. Depends on setting of DCLK_INVERTb and DCLKBY2_SHIFT: PIXEL and BLANKB timing 00 : Off DCLK falling edge 01 : 20 ns after DCLK rising edge 10 : Off DCLK rising edge 11 : 20 ns after DCLK falling edge
f	**R/W**	**TST_SRC_SEL_BIT5**	**Bit 5 of select source of PLL test clock. See VT/GT-B CLKBLK test plan for details. Bits from (4:0) are in PLL_TEST_CTRL register.**

Description

Controls VESA Feature Connector (default = 00h)

	PLL_TEST_CTRL					**Addr: 14**		
BITS	7	6	5	4	3	2	1	0
	d	c	b	a				

a	R/W	TST_SRC_SEL	Selects source of PLL test clock. See VT/GT-B CLKBLK test plan for details.
b	R/W	TST_DIVIDERS	1 : Open reference and feedback dividers for test
c	R/W	PLL_MASK_READb	0 : Mask PLL_TEST_COUNT(2:0) and disable test output pin

Cont'd	PLL_TEST_CTRL					Addr: 14		
BITS	7	6	5	4	3	2	1	0
	d	c	b	a				

d	R/W	ANALOG_MON_EN	Control PLL & Bandgap analog test mode 0 : disable 1 : enable

Description

PLL test mode control, used for ASIC production testing

	PLL_TEST_COUNT					Addr: 15		
BITS	7	6	5	4	3	2	1	0
	a							

a	R/W	PLL_TEST_COUNT	PLL test mode counter (read only, no default). Writing any value will reset to 00h.

Description

Used for ASIC production testing only.

	LVDSPLL_CNTL0					Addr: 16		
BITS	7	6	5	4	3	2	1	0
LT	c				b			a

a	R/W	FPDI_NS_TIMING	0: Timing compatible National (DS90CR562/582) 1: Timing compatible with FPDI proposal
b	R/W	CURR_LEVEL	Output current level control
c	R/W	LVDS_TEST_MODE	Define testmode for LVDS

	LVDSPLL_CNTL1					**Addr: 17**		
BITS	7	6	5	4	3	2	1	0
LT	d			c		b		a

a	R/W	LPLL_RANGE	Range control for LPLL
b	R/W	LPLL_DUTY	Duty cycle control for LPLL
c	R/W	LPLL_VC_GAIN	VCGEN gain control for LPLL (default 1)
d	R/W	LPLL_CP_GAIN	CP gain control for LPLL (default 5)

	AGP1_CNTL					**Addr: 18**		
BITS	7	6	5	4	3	2	1	0
	c		b			a		

a	R/W	X1_CLOCK_SKEW	AGP X1 phase adjustment (0 = default)
b	R/W	X2_CLOCK_SKEW	AGP X2 phase adjustment (0 = default)
c	R/W	PUMP_GAIN	AGP PLL charge-pump gain setting

Description

X1 and X2_CLOCK_SKEW are used to phase shift X1 clock (to roughly 0.5 ns step) w.r.t. X2 clock, and vice-versa.

Usage

They should not be changed from their BIOS settings.

X1 and X2 come from AGP PLL

	AGP2_CNTL					**Addr: 19**		
BITS	7	6	5	4	3	2	1	0
	e			d	c	b		a

a	R/W	AP_TST_EN	1 = Set up AGP PLL test mode 0 = default

Cont'd	AGP2_CNTL					Addr: 19		
BITS	7	6	5	4	3	2	1	0
	e			d	c	b		a

b	R/W	AP_X_SEL	00 = PCI reference clock is selected 01 = x1 clock is selected 10 = x2 clock is selected This field is only used for AGP PLL test mode
c	R/W	AP_SLEEP	Only for AGP PLL test, default = 0 (not sleep)
d	R/W	AP_RESET	Only for AGP PLL test, default = 0 (not reset)
e	R/W	ANALOG_MON	Controls select mux for PLL & Bandgap analog test.

Description

This register is only used to test AGP PLL in ASIC production.

	DLL2_CNTL					Addr: 20		
BITS	7	6	5	4	3	2	1	0
	c			b		a		

a	R/W	DLL_SKEW	DLL skew control (0 = default)
b	R/W	DLL_RANGE	DLL range control (2 = default)
c	R/W	DLL_FB_SKEW	DLL feedback skew control (0 = default)

Description

Extended control for XCLK DLL

	SCLK_FB_DIV					Addr: 21		
BITS	7	6	5	4	3	2	1	0
	a							

a	R/W	SCLK_FB_DIV	Feedback divider value for the secondary PLL

	SPLL_CNTL1						Addr: 22	
BITS	7	6	5	4	3	2	1	0
	d	c		b		a		

a	R/W	SPLL_PC_GAIN	SPLL Charge-pump gain setting
b	R/W	SPLL_VC_GAIN	SPLL VCGEN gain setting
c	R/W	SPLL_D_CYC	Duty cycle control for SPLL
d	R/W	SPLL_RANGE	SPLL range control

Description

Control lines for the Secondary PLL

	SPLL_CNTL2						Addr: 23	
BITS	7	6	5	4	3	2	1	0
		c					b	a

a	R/W	S_SLEEP	Secondary PLL sleep
b	R/W	S_RESET	Secondary PLL reset
c	R/W	SCLK_SRC_SEL	Clock select for different divide downs of SCLK from secondary PLL 000 : SCLK = PLLSCLK 001 : SCLK = PLLSCLK/2 010 : SCLK = PLLSCLK/4 011 : SCLK = PLLSCLK/8 100 -111 : SCLK = CPUCLK

Description

Secondary PLL serves as another clock source for the GUI Engine, in addition to MPLL. Using this PLL allows The GUI clock to run at any frequency (within the operating range) regardless of what frequency the memory clock (XCLK) is running at. (Default = 53h)

	APLL_STRAPS					Addr: 24		
BITS	7	6	5	4	3	2	1	0
	c		b			a		

a	R	APLL_STRAPS1	X1 feedback phase adjustment with respect to refclk (cpuclk), straps from MD[46:44] 000 = refclk 1 tap earlier than X1 (feedback) -- default 001 = refclk 2 taps earlier than X1 (feedback) 010 = refclk 3 taps earlier than X1 (feedback) 011 = agp pll testmode, X2 is used as feedback 100 = feedback (X1) 3 taps earlier than refclk 101 = feedback (X1) 2 taps earlier than refclk 110 = feedback (X1) 1 tap earlier than refclk 111 = feedback (X1) and refclk are aligned Each tap is worth 0.5 ns roughly
b	R	APLL_STRAPS2	X1 clock phase adjustment with respect to X2, straps from MD[43:41] 000 = 0 taps (default) 001 = 1 taps 010 = 2 taps 011 = 3 taps 100 = 4 taps 101 = 5 taps 110 = 6 taps 111 = 7 taps Each tap is worth 0.5 ns roughly
c	R	FILL_GAIN	VCO filter gain control, default = straps from MD[39:38] 00 = 10 to VCO gain (default) 01 = 11 to VCO gain 10 = 00 to VCO gain 11 = 01 to VCO gain

Description

Read only straps to control APLL at reset to ensure proper PCI operation at first cycle.

	EXT_VPLL_CNTL						**Addr: 25**	
BITS	7	6	5	4	3	2	1	0
	e			d	c	b	a	

a	R/W	EXT_VPLL_REF_SRC	00 = VPLL_REF = XTALIN 01 = VPLL_REF = PLLMCLK/2 10 = VPLL_REF = PLLSCLK/2 11 = reserved
b	R/W	EXT_VPLL_EN	Control of VPLL reference and feedback. 0 = Use feedback divider settings in VCLKx_FB_DIV, reference divider setting in PLL_REF_DIV and XTALIN reference clock. 1 = Use EXT_VPLL_FB_DIV, EXT_VPLL_REF_DIV and EXT_VPLL_REF_SRC settings for extended display modes. Use for VGA only if EXT_VPLL_VGA_EN = 1.
c	R/W	EXT_VPLL_VGA_EN	0 = Do not use any EXT_VPLL settings in VGA modes. 1 = Use EXT_VPLL settings in VGA modes.
d	R/W	EXT_VPLL_INSYNC	Controls when EXT_VPLL_UPDATE action occurs: 0 = do atomic update ASAP 1 = wait for vertical sync before atomic update
e	**R/W**	**EXT_V2PLL_EN**	**0 = Use reference divider value programmed in PLL_REF_DIV** **1 = Use reference divider value programmed in EXT_VPLL_REF_DIV** **Default = 1**

Description

Controls use of extended precision reference and feedback for VCLK (default = 00h)

	EXT_VPLL_REF_DIV						**Addr: 26**	
BITS	7	6	5	4	3	2	1	0
	a							

a	R/W	EXT_VPLL_REF_DIV	Lower 8 bits of 10 bit extended VPLL reference divider. New value not used until EXT_VPLL_UPDATE is written. Readback is of pending value. (default = 00h)

	EXT_VPLL_FB_DIV					Addr: 27		
BITS	7	6	5	4	3	2	1	0
	a							

a	R/W	EXT_VPLL_FB_DIV	Lower 8 bits of 11 bit extended VPLL feedback divider. Effective feedback divider is the same as register setting, not twice register setting as for VCLKx_FB_DIV registers. New value not used until EXT_VPLL_UPDATE is written. Readback is of pending value. (default = 00h)

	EXT_VPLL_MSB					Addr: 28		
BITS	7	6	5	4	3	2	1	0
		c			b		a	

a	R/W	EXT_VPLL_REF_DIV	EXT_VPLL_REF_DIV bits 9:8
b	R/W	EXT_VPLL_UPDATE	Controls transfer of EXT_VPLL_REF_DIV and EXT_VPLL_FB_DIV from registers to functional units. Write: 0 = no update 1 = atomically update reference and feedback divider Read: 0 = atomic update complete, ready for next writes 1 = atomic update still pending, do not write EXT_VPLL_REF_DIV or EXT_VPLL_FB_DIV
c	R/W	EXT_VPLL_FB_DIV	EXT_VPLL_FB_DIV bits 10:8

Description

Most significant bits of EXT_VPLL_REF_DIV and EXT_VPLL_FB_DIV (default = 00h)

	HTOTAL_CNTL						Addr: 29	
BITS	7	6	5	4	3	2	1	0
	c	b				a		

a	R/W	PLLVCLK_SLIP	**Reserved, not yet implemented** Number of 1/5th of PLLVCLK periods to extend HTOTAL by. Valid range is 0 to 4, for 5 or more increment VCLK_POST_SLIP.
b	R/W	VCLK_POST_SLIP	**Reserved, not yet implemented** Number of periods of PLLVCLK to extend each HTOTAL by stopping VCLK post divider. For corrections equal to or greater than the current post divider use the TVO_H_TOT_PIX register.
c	R/W	HTOTAL_CNTL_VGA_EN	**Reserved, not yet implemented** 0 = No HTOTAL control through this register in VGA modes. 1 = Enable HTOTAL control through this register in VGA modes.

Description

Controls sub-pixel adjustment of HTOTAL timing. TVO_CNTL register controls active edge of HSYNC and HTOTAL adjustment down to pixel level. Update of this register does not take effect until opposite edge of HSYNC than specified in TVO_H_TOT_EDGE register to ensure atomic change of each slip field outside active area. (default = 00h)

	BYTE_CLK_CNTL						Addr: 30	
BITS	7	6	5	4	3	2	1	0
			c	b		a		

a	R/W	BYTE_CLK_SKEW	Selects BYTE_CLK phase in 1/2 PLLVCLK increments. Valid range depends on BYTE_CLK_POST_DIV, and may not exceed (2 * byte clock post divider)-1. e.g. for byte clock post divider of 3 (BYTE_CLK_POST_DIV = 10), then (2*3)-1=5, so BYTE_CLK_SKEW has range 0 to 5.
b	R/W	BYTE_CLK_POST_DIV	TV out byte clock post divider. If VCLK for TV out clock sourced in ImpacTV then set to 00, otherwise VCLK for TV out sourced from internal PLL and post dividers are changed as indicated. 00 = BYTE_CLK = GIO(1) input, no change in VCLK_POST_DIV 01 = BYTE_CLK = PLLVCLK/2, multiply VCLK_POST_DIV by 2 10 = BYTE_CLK = PLLVCLK/3, multiply VCLK_POST_DIV by 3 11 = BYTE_CLK = PLLVCLK/4, multiply VCLK_POST_DIV by 4

	BYTE_CLK_CNTL						**Addr: 30**	
BITS	7	6	5	4	3	2	1	0
			c	b		a		

c	R/W	BYTE_CLK_OUT_EN	0 = BYTE_CLK from ImpacTV to graphics controller 1 = forces BYTE_CLK to be muxed out GIO(1) to ImpacTV

Description

Control of byte clock sent to (or received from) ImpacTV and extra VCLK post divider. VCLK_SRC_SEL must be set to 10 for BYTE_CLK generation. (default = 00h)

	TV_PLL_CNTL1						**Addr: 31**	
BITS	7	6	5	4	3	2	1	0
LT	a							

a	R/W	TV_M	Reference Divider setting for TV clock (default = 2)

	TV_PLL_CNTL2						**Addr: 32**	
BITS	7	6	5	4	3	2	1	0
LT	a							

a	R/W	TV_N	Feedback Divider setting for TV clock (default = 6)

	TVPLL_CNTL						**Addr: 33**	
BITS	7	6	5	4	3	2	1	0
LT	d	c		b		a		

a	R/W	TVPLL_PC_GAIN	TVPLL Charge-pump gain setting
b	R/W	TVPLL_VC_GAIN	TVPLL VCGEN gain setting
c	R/W	TVPLL_D_CYC	Duty cycle control for TVPLL
d	R/W	TVPLL_RANGE	TVPLL range control

Description:

Controls to TVPLL analog macro (default = AD)

	EXT_TV_PLL					Addr: 34		
BITS	7	6	5	4	3	2	1	0
LT	e			d		c	b	a

a	R/W	TV_N_BIT9	Bit 9 of TV PLL feedback divider TV_N (default = 0)
b	R/W	TV_PLL_RST	1 - TV_PLL reset (default = 1)
c	R/W	TV_PLL_SLEEP	1 - TV_PLL power-down (default = 1)
d	R/W	TVCLK_SRC_SEL	00 - CPUCLK (default) 01 - XTALIN 10 - reserved 11 - TVPLLCLK
e	R/W	VCLKTV_SRC_SEL	000 - CPUCLK (inverted) (default) 001 - DCLK 010 - VCLK 011 - VCLK2 100 - VCLK/2 101 - V2CLK/2 others - reserved

	V2PLL_CNTL					Addr: 35		
BITS	7	6	5	4	3	2	1	0
LT	d	c		b		a		

a	R/W	V2PLL_PC_GAIN	TVPLL Charge-pump gain setting
b	R/W	V2PLL_VC_GAIN	V2PLL VCGEN gain setting
c	R/W	V2PLL_D_CYC	Duty cycle control for V2PLL
d	R/W	V2PLL_RANGE	V2PLL range control

Description:

Controls to V2PLL analog macro (default = CC)

	PLL_V2CLK_CNTL					Addr: 36		
BITS	7	6	5	4	3	2	1	0
LT	e			d	c	b	a	

a	R/W	V2CLK_SRC_SEL	00 : V2CLK_SRC = CPUCLK (default) 01 : V2CLK_SRC = DCLK 10 : V2CLK_SRC = GIO(1) 11 : V2CLK_SRC = PLLV2CLK V2CLK = V2CLK_SRC/V2CLK_POST_DIV
b	R/W	PLL_PRESET2	1 : Reset V2PLL (default = 1)
c	R/W	VCLK2_INVERT	1 : Invert V2CLK to get opposite duty cycle (default = 0)
d	R/W	V2PLL_SLEEP	1 : Power-down V2PLL (default = 1)
e	R/W	V2CLK_POST_DIV	V2CLK post divider select (default = 0) 000 - PLLV2CLK 001 - PLLV2CLK/2 010 - PLLV2CLK/4 011 - PLLV2CLK/8 100 - PLLV2CLK/3 101 - PLLV2CLK/5 110 - PLLV2CLK/6 111 - PLLV2CLK/12

Description:

Second pixel clock control (default = 14h)

	EXT_V2PLL_REF_DIV					Addr: 37		
BITS	7	6	5	4	3	2	1	0
LT	a							

a	R/W	EXT_V2PLL_REF_DIV	Lower 8 bits of 10 bit extended V2PLL reference divider. New value not used until EXT_V2PLL_UPDATE is written. Readback is of pending value. (default = 24h)

EXT_V2PLL_FB_DIV Addr: 38

BITS	7	6	5	4	3	2	1	0
LT	a							

a	R/W	EXT_V2PLL_FB_DIV	Lower 8 bits of 11 bit extended V2PLL feedback divider. New value not used until EXT_V2PLL_UPDATE is written. Readback is of pending value. (default = FDh)

EXT_V2PLL_MSB Addr: 39

BITS	7	6	5	4	3	2	1	0
LT		c			b		a	

a	R/W	EXT_V2PLL_REF_DIV	EXT_V2PLL_REF_DIV bits 9:8 (default = 0)
b	R/W	EXT_V2PLL_UPDATE	Controls transfer of EXT_V2PLL_REF_DIV and EXT_V2PLL_FB_DIV from registers to functional units (default = 0) Write: 0 = no update 1 = atomically update reference and feedback divider Read: 0 = atomic update complete, ready for next writes 1 = atomic update still pending, do not write EXT_V2PLL_REF_DIV or EXT_V2PLL_FB_DIV
c	R/W	EXT_V2PLL_FB_DIV	EXT_V2PLL_FB_DIV bits 10:8 (default = 0)

Description:

Most significant bits of EXT_V2PLL_REF_DIV and EXT_V2PLL_FB_DIV (default = 00h)

HTOTAL2_CNTL Addr: 40

BITS	7	6	5	4	3	2	1	0
LT	c	b				a		

a	R/W	PLLV2CLK_SLIP	Number of 1/5th of PLLV2CLK periods to extend HTOTAL by. Valid range is 0 to 4, for 5 or more increment V2CLK_POST_SLIP. (default = 0)

	HTOTAL2_CNTL					Addr: 40		
BITS	7	6	5	4	3	2	1	0
LT	c	b				a		

b	R/W	V2CLK_POST_SLIP	Number of periods of PLLV2CLK to extend each HTOTAL by stopping V2CLK post divider. For corrections equal to or greater than the current post divider use the TVO_H_TOT2_PIX register. (default = 0)
c	R/W	EXT_V2PLL_INSYNC	Controls when EXT_V2PLL_UPDATE action occurs: 0 = do atomic update ASAP (default) 1 = wait for vertical sync before atomic update

Description:

Controls sub-pixel adjustment of HTOTAL timing for second display. TVO_CNTL register controls active edge of HSYNC and HTOTAL adjustment down to pixel level. Update of this register does not take effect until opposite edge of HSYNC than specified in TVO_H_TOT2_EDGE register to ensure atomic change of each slip field outside active area. (default = 00h)

4.2.7 DAC Control

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

	DAC_REGS																								Offset: 0_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
	d								c								b								a							

a	R/W	DAC_W_INDEX	DAC write index register * Indexes the 256x24 entry palette RAM for write operations.

Cont'd

	DAC_REGS																								Offset: 0_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
	d								c								b								a							

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.

Byte accesses are recommended over word or Dword accesses. 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 4-84*

Chapter 8, VGA-Compatible Registers

mach64 Programmer's Guide:

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

DAC_CNTL **Offset: 0_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
															l		k	j	i					h	g	f	e	d	c	b	a	

a	R/W	DAC_RANGE_CNTL	DAC output standard 0 = PAL 1 = NTSC 2 = PS2 3 = Reserved (PS2)
b	R/W	DAC_BLANKING	0 = 0 IRE blanking pedestal 1 = Enable 7.5 IRE blanking pedestal
c	R/W	DAC_CMP_DISABLE	Enable/Disable DAC comparators 0 = enable DAC comparators (Default) 1 = disable (powerdown) DAC comparators
d	**R/W**	**DAC1_CLK_SEL**	**Selects data source for the monitor** **0 = CRTC1 (primary display)** **1 = CRTC2 (secondary display)**
e	**R/W**	**PALETTE_ACCESS_CNTL**	**Selects access to primary or secondary palette** **0 = access primary palette through DAC_REGS register** **1 = access secondary palette through DAC_REGS register**
f	**R/W**	**PALETTE2_SNOOP_EN**	**Enables snooping of primary palette writes** **0 = disabled** **1 = everything written to primary palette will be written to secondary palette as well**
g	R	DAC_CMP_OUTPUT	DAC comparator output 0 = At least 1 comparator > 0.42V 1 = All 3 comparators < 0.28V
h	R/W	DAC_8BIT_EN	Enables 8 bit DAC operation 1 = 8 bit operation 0 = 6 bit operation
i	R/W	DAC_VGA_ADR_EN	Enables addressing the DAC at the VGA IO DAC address when CRTC_EXT_DISP_EN is a '1'
j	R/W	DAC_FEA_CON_EN	Enables feature connector signal outputs
k	R/W	DAC_PDWN	Power down internal DAC (DAC macro only)
l	R	DAC_TYPE	DAC Type (always 1 for LT PRO) 0 = Internal DAC, 18-bit palette, no gamma correction 1 = Internal DAC, 24-bit palette, gamma correction 2-7 = (Reserved)

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.

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*

This page intentionally left blank.

Chapter 5
GUI Draw Engine

5.1 Draw Engine Trajectory Registers

This chapter describes the GUI engine registers. Operations are initiated implictly by writing to DST_WIDTH or DST_BRES_LNTH. X and Y coordinates are in the range -8192 to +8191 and -16384 to +16383 respectively. All drawing operations are orthogonal with respect to pixel size. Packed 24 bpp is partially supported by setting DST_PIX_WIDTH to 8 bpp and adjusting the x coordinates and scissor values. The DST_24_ROT_EN and DST_24_ROT control bits allow for a full 24 bpp foreground color, background color, and write mask, which will be rotated properly by the GUI engine. In addition, the 8x8x1 monchrome pattern source will be expanded and rotated properly. Note that Bresenham line drawing operations are not generally supported in the partial 24 bpp mode.

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

5.1.1 Destination Trajectory

This register has two names—DST_BRES_DEC or LEAD_BRES_DEC

<table>
<tr><td></td><td colspan="32">DST_BRES_DEC (LEAD_BRES_DEC) MM: 0_4B</td></tr>
<tr><td>BITS</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td></td><td colspan="14"></td><td colspan="18">a</td></tr>
</table>

a	R/W	DST_BRES_DEC	Bresenham decrement for line and 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.

Usage

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 5-2*

DST_BRES_INC on *page 5-3*

DST_BRES_LNTH on *page 5-3*

mach64 Programmer's Guide:

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

	DST_BRES_ERR																								**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
															a																	

a	R/W	DST_BRES_ERR	Bresenham error term for line and 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.

Usage

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 5-1*

DST_BRES_INC on *page 5-3*

DST_BRES_LNTH on *page 5-3*

mach64 Programmer's Guide:

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

This register has two names—DST_BRES_INC or LEAD_BRES_INC

<table>
<tr><td></td><td colspan="28">DST_BRES_INC (LEAD_BRES_INC)</td><td colspan="4">MM: 0_4A</td></tr>
<tr><td>BITS</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td></td><td colspan="14"></td><td colspan="18">a</td></tr>
</table>

a	R/W	DST_BRES_INC	Bresenham increment for line and 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.

Usage

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 5-1*

DST_BRES_ERR on *page 5-2*

DST_BRES_LNTH on *page 5-3*

mach64 Programmer's Guide:

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

The register below has two names—DST_BRES_LNTH or LEAD_BRES_LNTH

<table>
<tr><td></td><td colspan="20">DST_BRES_LNTH (LEAD_BRES_LNTH)</td><td colspan="12">MM: 0_48 and MM: 0_51</td></tr>
<tr><td>BITS</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td></td><td>d</td><td colspan="15">c</td><td>b</td><td colspan="15">a</td></tr>
</table>

a	R/W	DST_BRES_LNTH	Bresenham line and trapezoid leading edge length. This field is aliased to DST_WIDTH[14:0].
b	R/W	DRAW_TRAP	To initiate a trapezoid, set to '1'. This field is aliased to DST_WIDTH[15].
c	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 to DST_HEIGHT[14:0]. If SUB_PIX_ON@DST_CNTL is set, this field is interpreted as a S.12.2, otherwise a S.14.0 bit integer.

Cont'd **DST_BRES_LNTH (LEAD_BRES_LNTH)** **MM: 0_48 and MM: 0_51**

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
	d	c															b	a														

d	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

DST_BRES_LNTH is used for line draw or trapezoid draw operations

See Also

DST_BRES_DEC on *page 5-1*

DST_BRES_ERR on *page 5-2*

DST_BRES_INC on *page 5-3*

mach64 Programmer's Guide:

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

	DST_CNTL																											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
												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. 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_SIGN	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_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)

Cont'd	DST_CNTL																								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
												q	p		o		n	m	l	k	j	i			h	g	f	e	d	c	b	a

n	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
o	R/W	BRES_SIGN_AUTO	Bresenham sign: 0 = zero error term is defined by DST_BRES_SIGN and TRAIL_BRES_SIGN bit 1 = Overrides 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. For Trapezoids with sub-pixel addressing, this bit is changed to include pixels on the top/left of the triangle.
p	R/W	ALPHA_OVERLAP_ENB	Allow a pipeline optimization to fetch data for the current triangle before the previous triangle has been written to memory. If the triangles do in fact overlap in the frame buffer, this runs the risk of blending with the wrong data. 0 = disallow optimization. Guarantees correct blend. 1 = allow optimization. Risks incorrect data during triangle overlap.
q	R/W	SUB_PIX_ON	This forces the interpretation of BRES_LNTH, DST_X, DST_Y and TRAIL_X as having 2 bits of sub-pixel precision. This bit is valid only for Trapezoid trajectories. It also changed the interpretation of BRES_SIGN_AUTO.

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 5-64*

SRC_CNTL on *page 5-21*

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																								**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
																		a														

a	R/W	DST_HEIGHT	Destination height. (Bits14:0 aliased to TRAIL_X @ DST_BRES_LNTH)

Description

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

Usage

This register is used when drawing a rectangular or trapezoidal destination area.

See Also

DST_WIDTH on *page 5-10*

DST_HEIGHT_WIDTH on *page 5-8*

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																								**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
	b																a															

a	W	DST_HEIGHT	Destination height (see register DST_HEIGHT)
b	W	DST_WIDTH	Destination width (see register DST_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 5-8*

DST_WIDTH on *page 5-10*

DST_WIDTH_HEIGHT on *page 5-11*

	DST_OFF_PITCH																								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
	b												a																			

a	R/W	DST_OFFSET	Destination offset address in terms of 64 bit words. For SGRAM configuration, DST_OFFSET must be aligned on a 64 byte boundary.
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. For SGRAM configuration, DST_PITCH must be a multiple of 64 bytes.

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 5-26*

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																				**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
	b																a															

a	R/W	DST_WIDTH	Destination width. Only bits 13:0 are used for rectangle draws. Bit 15 is write ONLY and will always read back as '0'. Bits [15:14] are aliased to DST_BRES_LENGTH[15:14] and are used for trapezoid draw operations.
b	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 5-8*

DST_HEIGHT_WIDTH on *page 5-8*

DST_X_WIDTH on *page 5-12*

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_WIDTH_HEIGHT																								**MM: 0_BB**							
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
	b																a															

a	W	DST_WIDTH	Destination width (see register DST_WIDTH)
b	W	DST_HEIGHT	Destination height (see register DST_HEIGHT)

Description

DST_WIDTH_HEIGHT is a composite of registers DST_WIDTH and DST_HEIGHT (as is DST_HEIGHT_WIDTH, but in the inverse order). 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 5-8*

DST_WIDTH on *page 5-10*

DST_HEIGHT_WIDTH on *page 5-8*

	DST_X																								**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
																	b	a														

a	R/W	DST_X	Destination X coordinate. If SUB_PIX_ON is set, this field is interpreted as a S.12.2 number; otherwise it is a S.13.0 bit integer with bit 14 reserved,
b	R/W	SCALE_Y_SECONDARY_LSB	The LSB of the Y index of the secondary source. This bit is used as a “hint” to give better performance. It is ignored for non-scaling operations.

Description

DST_X specifies the starting X coordinate of the destination trajectory. This is a signed 14 bit number.

Usage

This register is used for all draw operations.

See Also

DST_X_WIDTH on *page 5-12*

DST_Y on *page 5-14*

DST_Y_X on *page 5-14*

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																										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
	c																b	a														

a	W	DST_X	Destination X coordinate
b	W	SCALE_Y_SECONDARY_LSB	The LSB of the Y index of the secondary source (see register DST_X)
c	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 5-11*

DST_WIDTH on *page 5-10*

	DST_X_Y																										MM: 0_BA					
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
	d	c															b	a														

a	W	DST_X	Destination X coordinate
b	W	SCALE_Y_SECONDARY_LSB	The LSB of the Y index of the secondary source (see register DST_X)
c	W	DST_Y	Destination Y coordinate
d	W	SCALE_Y_LSB	The LSB of the Y index of the primary source (see register DST_Y)

Description

DST_X_Y is a composite of registers DST_X and DST_Y, providing destination coordinates in the inverse order to DST_Y_X.

Usage

These registers are used for all draw operations.

See Also

DST_X on *page 5-11*

DST_Y on *page 5-14*

DST_Y_X on *page 5-14*

	DST_Y																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
																	b	a														

a	R/W	DST_Y	Destination Y coordinate. If SUB_PIX_ON is set, this field is interpreted as a S.12.2 number. Otherwise it is a S.14.0 integer.
b	R/W	SCALE_Y_LSB	The LSB of the Y index of the secondary source. This bit is used as a “hint” to give better performance. It is ignored for non-scaling ops.

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 5-11*

DST_Y_X on *page 5-14*

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																MM: 0_43 and MM: 0_4D															
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
	d	c															b	a														

a	W	DST_Y	Destination Y coordinate
b	W	SCALE_Y_LSB	The LSB of the Y index of the primary source (see register DST_Y)
c	W	DST_X	Destination X coordinate
d	W	SCALE_Y_SECONDARY_LSB	The LSB of the Y index of the secondary source (see register DST_X)

Description

DST_Y_X is a composite of registers DST_X and DST_Y.

Usage

These registers are used for all draw operations.

See Also

DST_X on *page 5-11*

DST_Y on *page 5-14*

TRAIL_BRES_ERR

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
															a																	

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

Description

TRAIL_BRES_ERR is a signed 18-bit register that stores the Bresenham error term for line and trapezoid trailing edges.

TRAIL_BRES_INC

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
															a																	

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

Description

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																								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
															a																	

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

Description

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																										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
	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 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.

All other Z functions directly track the DST registers.

See Also

Z_CNTL on *page 5-17*

	Z_CNTL																										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
																								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	Denotes that the 2D source, added to the Z interpolator should be used as the source for new Z values. 0 = Z source (new Z) from Z interpolator 1 = Z source (new Z) from 2D source + Z interpolator

Cont'd

Z_CNTL 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
																								d		c					b	a

c	R/W	Z_TEST	Specific Z test to be enabled: Z test codeTest 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

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 5-17*

ALPHA_TST_CNTL MM: 0_54

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
	k	j	i	h	g				f											e		d			c	b						a

a	R/W	ALPHA_TST_EN	Enables use of Alpha testing functions 0 = Alpha testing is disabled 1 = Alpha testing is enabled

Cont'd **ALPHA_TST_CNTL** **MM: 0_54**

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
	k	j	i	h	g				f											e		d			c	b						a

b	R/W	ALPHA_TEST	Specific Alpha test to be enabled. Alpha test code — Test 000 Alpha test never passes 001 Src_Alpha< RefDst Alpha 010 Src Alpha<= RefDst Alpha 011 Src Alpha == RefDst Alpha 100 Src Alpha >= RefDst Alpha 101 Src Alpha> Ref Dst Alpha 110 Src Alpha != RefDst Alpha 111 Src Alpha test always passes Tst_Src_Alpha is either the expanded alpha from the texture map OR the source alpha, depending on the setting of the ALPHA_TST_SRC_SEL bit. A passing Alpha test will overwrite the entire destination pixel with the new source pixel. Note that the Alpha used is the expanded alpha from the texture map. If TEX_MAP_AEN in SCALE_3D_CNTL is OFF, the alpha value is assumed to be 0xFF.
c	R/W	ALPHA_MOD_MSB	If this bit is set, and TEX_MAP_AEN is set, the msb of the TEXEL alpha is used to modify the source alpha value used to blend with the destination. MSB of Tex Alpha 0 = Use 0x00 as the source alpha value 1 = Use the alpha interpolator as the source alpha value This bit does not imply that any "alpha masking" is being done and in fact it is invalid to set both this bit and TEX_MASK_AEN.
d	R/W	ALPHA_DST_SEL	The Alpha value written to the destination is: 0 = destination alpha is 0x00 1 = destination alpha is 0xff 2 = (reserved) 3 = (reserved) 4 = destination alpha is As 5 = destination alpha is 1-As 6 = destination alpha Ad 7 = destination alpha is 1-Ad
e	R/W	ALPHA_TST_SRC_SEL	Selects the value that is compared to REF_ALPHA during alpha testing. 0 = use Texel Alpha 1 = use Source Alpha
f	R/W	REF_ALPHA	The reference alpha used for Alpha testing
g	R/W	COMPOSITE_CI4_RGB_ INDEX	These bits will be used as the upper 4 bits of the CI4 color value to select 1 of 16 different palettes for the secondary map.

Cont'd

ALPHA_TST_CNTL MM: 0_54

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
	k	j	i	h	g				f											e		d			c	b						a

h	R/W	COMPOSITE_CI4_RGB_LOW_NIBBLE	Denotes that when in CI8 -> RGB texture lookup mode for the secondary map, the texture should be interpreted as 4 bits/pixel, aligned in bits 3-0 of the byte.
i	R/W	COMPOSITE_CI4_RGB_HIGH_NIBBLE	Denotes that when in CI8 -> RGB texture lookup mode for the secondary map, the texture should be interpreted as 4 bits/pixel, aligned in bits 7-4 of the byte.
j	R/W	COMPOSITE_SHADOW	If this bit is set and compositing is being carried out, with the COMPOSITE_COMBINE_FCN set to **10,** the COMPOSITE_SHADOW_ID field is compared to the **unexpanded** composite texels. If they match, the SPECULAR color is substituted for the composite texel, otherwise, 0 is substituted for the composite texel.
k	R/W	SPECULAR_LIGHT_EN	Denotes whether a specular value should be added to texture mapped pixels as part of the lighting function. This bit should NOT be set if COMPOSITE_SHADOW is ON. 0 = Specular Lighting Off 1 = Specular Lighting On

COMPOSITE_SHADOW_ID MM: 0_E6

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
									a																							

a	R/W	COMPOSITE_SHADOW_ID	This field is a count of executed 3D primitives. It is used as part of the shadow ID algorithm, but may also be used as a general counter for performance purposes.

5.1.2 Source Trajectory

	SRC_CNTL																										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
																		n	m	l	k		j	i	h	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.
e	R/W	SRC_LINE_X_DIR	Source X direction when drawing operation is a bresenham line.
f	R/W	SRC_8x8x8_BRUSH	Treats source as an 8x8x8 linear brush (SRC must be QWORD aligned)
g	R/W	FAST_FILL_EN	Fast filling for transparent DST. Not needed if auto-fast-fills are enabled (see HW_DEBUG). Write as ‘0’
h	R/W	SRC_TRACK_DST	Source will track the trajectory which the Dst FIFO is using.
i	R/W	BUS_MASTER_EN	Enable bus mastering for any subsequent GUI operations
j	R/W	BUS_MASTER_SYNC	Synchronize GUI operations to bus master. No operations permitted until both GUI and bus master are complete
k	R/W	BUS_MASTER_OP	GUI operation performed by the bus master: 0 = Frame buffer to system memory operation 1 = System memory to frame buffer operation 2 = Foreground register to system memory operation 3 = System memory to host data register operation
l	R/W	SRC_8x8x8_BRUSH_LOADED	Holds current 8x8x8 brush for next brush operation: 0 = Load new brush 1 = Hold 8x8x8 brush from previous operation **Note:** SRC_8x8x8_BRUSH must be set to ‘1’ for this bit to have effect

Cont'd	SRC_CNTL																									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
																		n	m	l	k		j	i	h	g	f	e	d	c	b	a

m	R/W	COLOR_REG_WRITE_EN	Enables color register blit for SGRAM. Not needed when auto-color register updates are enabled (see HW_DEBUG). Write as 0
n	R/W	BLOCK_WRITE_EN	Enables block write blit using SGRAM colour register. Not needed if auto-block-writes are enabled (see HW_DEBUG). Write as '0').

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:

Table 5-1

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 mode, 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.

For SRC_8x8x8_BRUSH, the SRC_LINEAR_EN must be set as well. The source pixel depth should be set to 8Bpp and the source in DP_SRC should be set the 'Blit Source' in order for the 8x8x8 brush to be used.

For BUS_MASTER_OP = 2 (Foreground register to system memory), all 32-bits of the foreground register are always transferred to system memory. Thus, prior to setting the bus master to this mode, the desired colour to be transferred must be replicated in all 32-bits of the foreground colour register if the current GUI pixel depth is < 32Bpp.

For block write operations, the following apply:

- The color register must be written qword-aligned.
- There are no restrictions on the alignment of operations (i.e., DST X), except that DST_OFF_PITCH must be 64 byte aligned and DST_PITCH must be a multiple of 64 bytes in pixel depth.
- Pixel depths of 8/16/32 bpp are fully supported (not 24 bpp).
- Fastfill bit must be set.

Usage

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

See Also

DST_CNTL on *page 5-5*

GUI_TRAJ_CNTL on *page 5-64*

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																								**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
																	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 5-24*

SRC_WIDTH1 on *page 5-27*

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																								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
			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 5-23*

SRC_WIDTH1 on *page 5-27*

SRC_HEIGHT2

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
																		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 5-25*

SRC_WIDTH2 on *page 5-27*

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

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
			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 5-25*

SRC_WIDTH2 on *page 5-27*

	SRC_OFF_PITCH																								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
	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 5-9*

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 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
																			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 5-23*

SRC_HEIGHT1_WIDTH1 on *page 5-24*

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 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
																			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 5-25*

SRC_HEIGHT2_WIDTH2 on *page 5-25*

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*

<table>
<tr><td></td><td colspan="22">SRC_X</td><td colspan="10">MM: 0_61</td></tr>
<tr><td>BITS</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td></td><td colspan="18"></td><td colspan="14">a</td></tr>
</table>

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 14 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 5-29*

SRC_Y_X on *page 5-31*

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																								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
																			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 14 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 5-30*

SRC_Y_X_START on *page 5-31*

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																								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
																		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 5-28*

SRC_Y_X on *page 5-31*

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																								**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
																		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 5-29*

SRC_Y_X_START on *page 5-31*

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																								**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
			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 5-29*

SRC_X on *page 5-28*

	SRC_Y_X_START																								**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
			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 5-29*

SRC_Y_START on *page 5-30*

5.2 Draw Engine Control Registers

5.2.1 Host Data

The host data registers provide pixel data which are utilized in the current drawing operation. The pixel data may be used as a monochrome pixel source or color pixel source. For rectangular drawing operations the pixel data may be either packed from one horizontal line to the next or unpacked. All registers are treated identically and data is fed to the engine in the order in which it is written to any of the host data registers. Up to sixteen host data registers are provided to allow block data moves of variable length up to the depth of the parameter FIFO.

HOST_DATA[15:0] 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
	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 mode.)

Description

HOST_DATA is actually a single register mapped to 16 consecutive addresses, thus the notation 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.

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 4-5*

HOST_CNTL on *page 5-34*

mach64 Programmer's Guide:

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

	HOST_CNTL																									**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
																															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 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.

See Also

GUI_TRAJ_CNTL on *page 5-64*

HOST_DATA on *page 5-33*

mach64 Programmer's Guide:

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

5.2.2 Pattern

Two pattern registers support three fixed destination aligned pattern modes; monochrome 8x8, 8bpp color 4x2, and 8bpp color 8x1. For the VT/GT-B, 8x8x8 patterns or brushes can be using a linear source in conjuction with the SRC_8x8x8_BRUSH@SRC_CNTL. For all patterns, the alignment of register data to the least significant bits of DST_X and DST_Y is as follows:

Table 5-2

Monochrome 8x8x1, DP_BYTE_PIX_ORDER = 0								
	DST_X							
DST_Y	0	1	2	3	4	5	6	7
0	P0(7)	P0(6)	P0(5)	P0(4)	P0(3)	P0(2)	P0(1)	P0(0)
1	P0(15)	P0(14)	P0(13)	P0(12)	P0(11)	P0(10)	P0(9)	P0(8)
2	P0(23)	P0(22)	P0(21)	P0(20)	P0(19)	P0(18)	P0(17)	P0(16)
3	P0(31)	P0(30)	P0(29)	P0(28)	P0(27)	P0(26)	P0(25)	P0(24)
4	P1(7)	P1(6)	P1(5)	P1(4)	P1(3)	P1(2)	P1(1)	P1(0)
5	P1(15)	P1(14)	P1(13)	P1(12)	P1(11)	P1(10)	P1(9)	P1(8)
6	P1(23)	P1(22)	P1(21)	P1(20)	P1(19)	P1(18)	P1(17)	P1(16)
7	P1(31)	P1(30)	P1(29)	P1(28)	P1(27)	P1(26)	P1(25)	P1(24)

Table 5-3

Monochrome 8x8x1, DP_BYTE_PIX_ORDER = 1								
	DST_X							
DST_Y	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	P0(0)	P0(1)	P0(2)	P0(3)	P0(4)	P0(5)	P0(6)	P0(7)
1	P0(8)	P0(9)	P0(10)	P0(11)	P0(12)	P0(13)	P0(14)	P0(15)
2	P0(16)	P0(17)	P0(18)	P0(19)	P0(20)	P0(21)	P0(22)	P0(23)
3	P0(24)	P0(25)	P0(26)	P0(27)	P0(28)	P0(29)	P0(30)	P0(31)
4	P1(0)	P1(1)	P1(2)	P1(3)	P1(4)	P1(5)	P1(6)	P1(7)
5	P1(8)	P1(9)	P1(10)	P1(11)	P1(12)	P1(13)	P1(14)	P1(15)
6	P1(16)	P1(17)	P1(18)	P1(19)	P1(20)	P1(21)	P1(22)	P1(23)
7	P1(24)	P1(25)	P1(26)	P1(27)	P1(28)	P1(29)	P1(30)	P1(31)

Table 5-4

Color 4x2x8				
	DST_X			
DST_Y	**0**	**1**	**2**	**3**
0	P0(7:0)	P0(15:8)	P0(23:16)	P0(31:24)
1	P1(7:0)	P1(15:8)	P1(23:16)	P1(31:24)

Table 5-5

Color 8x1x8							
DST_X							
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
P0(7:0)	P0(15:8)	P0(23:16)	P0(31:17)	P1(7:0)	P1(15:8)	P1(23:16)	P1(31:24)

	PAT_REG0	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
	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 5-38*

PAT_REG1 on *page 5-37*

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	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
	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 5-38*

PAT_REG0 on *page 5-37*

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																											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
																														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 5-64*

PAT_REG0 on *page 5-37*

PAT_REG1 on *page 5-37*

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*

5.2.3 Scissors

The scissor registers define the rectangular region within which data is drawn. Left and right scissor registers are within the range -8192 to +8191. Top and bottom scissor registers are within the range -16384 to +16383. Polylines which follow a trajectory to the left of the left scissor register will result in a line drawn along the left scissor coordinate.

	SC_LEFT																								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
																			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, 14-bit number.

Usage

This register must be set for all draw operations.

See Also

SC_TOP on *page 5-41*

SC_BOTTOM on *page 5-41*

SC_RIGHT on *page 5-40*

SC_LEFT_RIGHT on *page 5-40*

mach64 Programmer’s Guide:

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

	SC_RIGHT																									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
																			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 14-bit number.

Usage

This register must be set for all draw operations.

See Also

SC_TOP on *page 5-41*

SC_LEFT on *page 5-39*

SC_LEFT_RIGHT on *page 5-40*

SC_BOTTOM on *page 5-41*

mach64 Programmer's Guide:

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

	SC_LEFT_RIGHT																									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
			b																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 5-39*

SC_RIGHT on *page 5-40*

	SC_TOP																								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
																		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 5-41*

SC_LEFT on *page 5-39*

SC_RIGHT on *page 5-40*

SC_TOP_BOTTOM on *page 5-42*

mach64 Programmer’s Guide:

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

	SC_BOTTOM																								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
																		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 5-41*

SC_TOP_BOTTOM on *page 5-42*

SC_LEFT on *page 5-39*

SC_RIGHT on *page 5-40*

mach64 Programmer’s Guide:

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

<table>
<tr><td></td><td colspan="32">SC_TOP_BOTTOM MM: 0_AD</td></tr>
<tr><td>BITS</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td></td><td></td><td colspan="15">b</td><td></td><td colspan="15">a</td></tr>
</table>

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 5-41*

SC_BOTTOM on *page 5-41*

5.2.4 Data Path

DP_BKGD_CLR **MM: 0_B0**

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
	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:

Table 5-6

Video Mode	Bits Used
1 bpp	the least significant bit
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 (ALSO DP_FOG_CLR) **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
	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:

Table 5-7

Video Mode	Bits Used
1 bpp	the least significant bit
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.

The 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*

<table>
<tr><td></td><td colspan="24">DP_FRGD_BKGD_CLR</td><td colspan="8">MM: 0_B8</td></tr>
<tr><td>BITS</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td></td><td colspan="16">b</td><td colspan="16">a</td></tr>
</table>

a	W	DP_FRGD_CLR	Foreground color [0..15]
b	W	DP_BKGD_CLR	Background color [0..15]

Description

DP_FRGD_BKGD_CLR is used to set pixel depth.

Usage

Set for 16 bpp pixel depths and below.

See Also

mach64 Programmer's Guide:

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

DP_FRGD_CLR_MIX **MM: 0_B7**

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
				c								b					a															

a	W	DP_FRGD_CLR	Foreground color [0..15]
b	W	DP_FRGD_MIX	Foreground mix
c	W	DP_FRGD_MIX	Background mix

See Also

mach64 Programmer’s Guide:

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

DP_WRITE_MSK **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
	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 according to the video mode used as shown in *Table 5-7 on page 5-44*

Usage

All draw operations require this register to be set.

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_PIX_WIDTH **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
	k				j	i	h	g	f				e						d		c				b				a			

a	R/W	DP_DST_PIX_WIDTH	Destination datapath pixel width : 0 = monochrome 1 = (reserved) 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 = RGB8 greyscale (8 bit intensity, duplicated for all 3 channels. Green channel is used on writes) 10 = (reserved) 11 = YUV 422 packed (VYUY) 12 = YUV 422 packed (YVYU) 13 = (reserved) 14 = aYUV 444 (8:8:8:8) 15 = aRGB4444 (intermediate format only, not understood by the Display Controller)
b	R/W	COMPOSITE_PIX_WIDTH	Datapath pixel width for secondary texture. Note that if the primary texture and the secondary texture are required to be the same width (in terms of bpp), but they may vary in format within that restriction.. 0 = (reserved) 1 = (reserved) 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 = RGB8 greyscale (8 bit intensity, duplicated for all 3 channels. Green channel is used on writes) 10 = (reserved) 11 = YUV 422 packed (VYUY) 12 = YUV 422 packed (YVYU) 13 = (reserved) 14 = aYUV 444 (8:8:8:8) 15 = aRGB4444

Cont'd

DP_PIX_WIDTH 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
	k				j	i	h	g	f				e						d		c				b				a			

c	R/W	DP_SRC_PIX_WIDTH	Source datapath pixel width: 0 = monochrome 1 = (reserced) 2 = 8 bpp pseudocolor 3 = 16 bpp aRGB 1555 4 = 16 bpp RGB 565 5 = (reserved) 6 = 32 bpp aRGB 8888 7 = 8 bpp RGB332 8 = Y8 greyscale 9-10 = (reserved) 11 = YUV 422 packed (VYUY) 12 = YUV 422 packed (VYUY) 13 = (reserved) 14 = aYUV 444 (8:8:8:8) 15 = aRGB4444
d	R/W	DP_HOST_TRIPLE_EN	0 = disable host data triplication 1 = Enable host data triplication
e	R/W	DP_HOST_PIX_WIDTH	Host datapath pixel width: 0 = monochrome 1 = (reserved) 2 = 8 bpp pseudocolor 3 = 16 bpp aRGB 1555 4 = 16 bpp RGB 565 5 = (reserved) 6 = 32 bpp aRGB 8888 7 = 8 bpp RGB8 332 8 = Y8 greyscale 9 = RGB8 greyscale 10 = (reserved) 11 = YUV 422 packed (VYUY) 12 = YUV 422 packed (YVYU) 13 = (reserved) 14 = aYUV444 (8:8:8:8) 15 = aRGB4444
f	R/W	DP_CI4_RGB_INDEX	These bits will be used as the upper 4 bits of the CI4 color value to select 1 of 16 different pallettes.
g	R/W	DP_BYTE_PIX_ORDER	Reverses the pixel order within each byte in monochrome modes: 0 = pixel order from MSBit to LSBit. 1 = pixel order from LSBit to MSBit
h	R/W	DP_CONVERSION_TEMP	YUV to RGB conversion temperature 0 = red@6500 K, GB@9300 K 1 = RGB@9300K

Cont'd	DP_PIX_WIDTH																								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
	k				j	i	h	g	f				e						d		c				b				a			

i	R/W	DP_CI4_RGB_LOW_NIBBLE	Denotes that when in CI8 -> RGB texture lookup mode, the texture should be interpreted as 4 bits/pixel, aligned in bits 3-0 of the byte.
j	R/W	DP_CI4_RGB_HIGH_NIBBLE	Denotes that when in CI8 -> RGB texture lookup mode, the texture should be interpreted as 4 bits/pixel, aligned in bits 7-4 of the byte.
k	R/W	DP_SCALE_PIX_WIDTH	Scaler source and 3D (texture and shading) datapath pixel width: 0 = (reserved) 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 = RGB8 greyscale (8 bit intensity, duplicated for all three channels, Green channel is used on writes) 10 = (reserved) 11 = YUV 422 packed (VYUY) 12 = YUV 422 packed (YVYU) 13 = (reserved) 14 = aYUV444 (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 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 mode. 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:

Table 5-8

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

Note that 8 bpp pseudo-color, 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.
Also, 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:

Table 5-9

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 texture palette.

See Also

mach64 Programmer's Guide:

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

DP_MIX **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
												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.

.

Table 5-10

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-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'*.

Note that when Alpha Blending or Anti-Aliasing 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 5-58*

mach64 Programmer’s Guide:

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

	USR_DST_PITCH																						MM: 0_BC									
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
																							a									

a	W	USR_DST_PITCH	DST_PITCH preset value for DP_SET_GUI_ENGINE/DP_SET_GUI_ENGINE2

Usage

Used with DP_SET_GUI_ENGINE and DP_SET_GUI_ENGINE2

The two LT sepecific registers below are used to support 2D/3D operations on the secondary display

	USR1_DST_OFF_PITCH																						MM: 0_AE									
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
LT	b												a																			

a	W	USR1_DST_OFFSET	DST1_OFFSET value
b	W	USR1_DST_PITCH	DST1_PITCH value

	USR2_DST_OFF_PITCH																						MM: 0_AF									
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
LT	b												a																			

a	W	USR2_DST_OFFSET	DST2_OFFSET value
b	W	USR2_DST_PITCH	DST2_PITCH value

The register DP_SET_GUI_ENGINE below is modified (from the RAGE PRO version) to include support for 2D/3D operations on the secondary display.

DP_SET_GUI_ENGINE MM: 0_BF

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
		n	m	l	k	j	i		h						g		f	e	d				c			b	a					

a	W	SET_DP_DST_PIX_WIDTH	DP_DST_PIX_WIDTH: 0 = Mono 1 = Reserved 2 = 8 Bpp 3 = 15 Bpp 4 = 16 Bpp 5 = Reserved 6 = 32 Bpp 7 = Reserved
b	W	SET_DP_SRC_PIX_WIDTH	DP_SRC_PIX_WIDTH: 0 = Mono 1 = Set same as SET_DP_DST_PIX_WIDTH setting
c	**W**	**SET_DST_OFFSET**	DST_OFFSET: 0 = 0 1 = 256K 2 = 512K 3 = 768K 4 = 1MB 5 = Reserved **6 = USR1_DST_OFFSET** **7 = USR2_DST_OFFSET**
d	**W**	**SET_DST_PITCH**	**DST_PITCH:** **0 = USR1_DST_PITCH (if SET_DST_PITCH_BY_2 = 0)** **USR2_DST_PITCH (if SET_DST_PITCH_BY_2 = 1)** 1 = 320 2 = 352 3 = 384 4 = 640 5 = 800 6 = 896 7 = 512 8 = 1024 9 = 1152 10 = 1280 11 = 400 12 = 832 13 = 1600 14 = 448 15 = 2048
e	W	SET_DST_PITCH_BY_2	Modify SET_DST_PITCH setting accordingly: 0 = leave alone 1 = DstPitch*2 (Ignored when SET_DST_PITCH = 0)

Cont'd

DP_SET_GUI_ENGINE **MM: 0_BF**

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
		n	m	l	k	j	i		h						g		f	e	d				c			b	a					

f	W	SET_SRC_OFFPITCH_COPY	SRC_OFF_PITCH: 0 = SRC_OFF_PITCH set to 0 1 = SRC_OFF_PITCH set to DST_OFF_PITCH
g	W	SET_SRC_HGTWID1_2	SRC_HEIGHT_WIDTH_1, SRC_HEIGHT_WIDTH_2: 0: Height = 8, Width = 8 1: Height = 1, Width = 32 2: Height = 8, Width = 24 3: Reserved
h	W	SET_DRAWING_COMBO	(See DRAWING_COMBO table below)
i	W	SET_BUS_MASTER_OP	GUI operation performed by the bus master: 0 = Frame buffer to system memory operation 1 = System memory to frame buffer operation 2 = Foreground register to system memory operation 3 = System memory to host data register operation
j	W	SET_BUS_MASTER_EN	Enables bus mastering for any subsequent GUI operations
k	W	SET_BUS_MASTER_SYNC	Synchronizes GUI operations to bus master. No operations permitted until both GUI and bus master are complete
l	W	DP_HOST_TRIPLE_EN	Enables triplication of monochrome host data
m	W	FAST_FILL_EN	Fast filling for transparent DST. Not needed if auto-fast-fills are enabled (see HW_DEBUG). Write as '0'
n	W	BLOCK_WRITE_EN	Enables block write blit using SGRAM colour register. Not needed if auto-block-writes are enabled (see HW_DEBUG). Write as '0').

Usage

Writing this register will set the following registers to the known values indicated, in addition to the registers set by the bit fields:

Table 5-11

Register	Value
DST_Y_X	0
DST_HEIGHT_WIDTH	0
SRC_Y_X	0
SC_TOP_BOTTOM	3FFF0000h = OPEN completely
SC_LEFT_RIGHT	1FFF0000h = OPEN completely
DP_WRITE_MSK	FFFFFFFFh = enable all destination writes

Table 5-11 Cont'd

Register	Value
DP_HOST_PIX_WIDTH@DP_PIX_WIDTH	0
DP_BYTE_PIX_ORDER@DP_PIX_WIDTH	0
SRC_8x8x8_BRUSH@SRC_CNTL	0
SRC_8x8x8_BRUSH_LOADED@SRC_CNTL	0
CLR_CMP_CNTL	0
SRC_X_START	0
DP_SRC_AUTONA_FIX_DIS@DP_PIX_WIDTH	0
DP_FAST_SRCCOPY_DIS@DP_PIX_WIDTH	0
DP_CI4_RGB_INDEX@DP_PIX_WIDTH	0
DP_CONVERSION_TEMP@DP_PIX_WIDTH	0
DP_CI4_RGB_LOW_NIBBLE@DP_PIX_WIDTH	0
DP_CI4_RGB_HIGH_NIBBLE@DP_PIX_WIDTH	0
DP_SCALE_PIX_WIDTH@DP_PIX_WIDTH	0
DP_COMPOSITE_PIX_WIDTH@DP_PIX_WIDTH	0
SRC_Y_START	0
COLOR_REG_WRITE_EN@SRC_CNTL	0
BLOCK_WRITE_EN@SRC_CNTL	0
TRAIL_X_DIR@DST_CNTL	0
TRAIL_FILL_DIR@DST_CNTL	0
TRAIL_BRES_SIGN@DST_CNTL	0

Table 5-12 DRAWING_COMBO Table

	DP_SRC	DP_MIX	GUI_TRAJ_CNTL	Used in
0000	XXXXXXXX	XXXXXX XX	XXXXXXXX	state is unknown and undefined
0001	0000100h (FgFrgdClr)	070003h	00000023h DstXDir+DstYDir+ DstLastPel	DefaultContext (Default Screen Blit)
0010	0000200h (FgHost)	070007h	00000003h DstXDir+DstYDir	BltSCol2DScr via HOST_DATA (GWM3).
0011	0020100h (MonoHost+FgFrgdClr)	070007h	00000003h DstXDir + DstYDir	BltSMonl2DScr via HOST_DATA (GWM4).
0100	0000100h (FgFrgdClr)	070007h	00000023h DstXDir+DstYDir+ DstLastPel	SolPat2DstScr PATCOPY (GWM1)

Table 5-12 DRAWING_COMBO Table Cont'd

	DP_SRC	DP_MIX	GUI_TRAJ_CNTL	Used in
0101	0010100h (MonoPattRegs+ FgFrgdClr)	070007h	01000003h DstXDir+DstYDir+ PatMonoEnable	BltHatPat2DstScr MonoPATCOPY
0110	0000100h (FgFrgdClr)	070007h	00000003 DstXDir+DstYDir	patCopy and srcCopy for solid ROPs for Apple
0111	0000300h (FgBlit)	070007h	00030003 DstXDir+DstYDir+Src PattEn+SrcPattRotEn	patCopy and srcCopy for pattern ROPs for Apple
1000	0000300h (FgBlit)	070007h	00000000h	DefaultContext + S to S SRCCOPY
1001	0000300h (FgBlit)	070007h	00000001h	DefaultContext + S to S SRCCOPY
1010	0000300h (FgBlit)	070007h	00000002h	DefaultContext + S to S SRCCOPY
1011	0000300h (FgBlit)	070007h	00000003h	DefaultContext + S to S SRCCOPY + CacToS SRCCOPY (GWM3,5)
1100	0020100h (FgFrgdClr+MonoHst)	070003h	1004001Bh (HostByteAlign+ SrcLinearEnable+ DstXDir + DstYDir+ DstXTile+DstYTile	MonoContext : (Default MonoExpand Blit)
1101	0020100h (FgFrgdClr+MonoHst)	070003h	0004001Bh (SrcLinearEnable+ DstXDir + DstYDir + DstXTile + DstYTile)	MonoContext : (Text Blit)
1111	0000300h (FgBlit)	070007h	0004001Bh (SrcLinearEnable + DstXDir + DstYDir + DstXTile + DstYTile)	Src_8x8x8_Brush, polygon fill, pat-copy

DP_SET_GUI_ENGINE2 **MM: 0_BE**

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
	p	o				n	m			l	k	j	i	h	g	f	e		d			c			b				a			

a	R/W	DP_BKGD_MIX	as in DP_MIX
b	R/W	DP_FRGD_MIX	as in DP_MIX
c	R/W	DP_BKGD_SRC	as in DP_SRC

Cont'd	DP_SET_GUI_ENGINE2																								MM: 0_BE							
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
	p	o				n	m			l	k	j	i	h	g	f	e		d			c			b				a			

d	R/W	DP_FRGD_SRC	as in DP_SRC
e	R/W	DP_MONO_SRC	as in DP_SRC
f	R/W	DST_X_DIR	as in GUI_TRAJ_CNTL
g	R/W	DST_Y_DIR	as in GUI_TRAJ_CNTL
h	R/W	PAT_MONO_EN	as in GUI_TRAJ_CNTL
i	R/W	SRC_PATT_ROT_EN	as in GUI_TRAJ_CNTL
j	R/W	FAST_FILL_EN	as in SRC_CNTL
k	R/W	BLOCK_WRITE_EN	as in SRC_CNTL
l	R/W	SET_DP_WRITE_MASK	Set DP_WRITE_MASK 0 = leave alone 1 = set to 0xFFFFFFFF
m	R/W	DST_PIX_WIDTH	as in DP_PIX_WIDTH/DP_SET_GUI_ENGINE
n	R/W	SET_SRC_PIX_WIDTH	as in DP_SET_GUI_ENGINE
o	R/W	SET_DST_PITCH	as in DP_SET_GUI_ENGINE
p	R/W	SRC_OFFPITCH_COPY	as in DP_SET_GUI_ENGINE

Usage

Writing this register will set the following registers to the following known values in addition to the registers set by the bit fields:

Table 5-13

Register	Value
DST_Y_X	0
DST_HEIGHT_WIDTH	0
SRC_Y_X	0
DP_HOST_PIX_WIDTH@DP_PIX_WIDTH	0
DP_BYTE_PIX_ORDER@DP_PIX_WIDTH	0
CLR_CMP_CNTL	0

	DP_SRC																								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
															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 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*

5.2.5 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.

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																									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
	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 5-61*

CLR_CMP_MSK on *page 5-60*

mach64 Programmer's Guide:

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

	CLR_CMP_MSK																								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
	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 5-59*

CLR_CMP_CNTL on *page 5-61*

mach64 Programmer's Guide:

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

	CLR_CMP_CNTL																												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
							b																							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 = 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. When CLR_CMP_SRC is ‘1’, Auto Fastfill must be disabled.

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 5-59*

CLR_CMP_MSK on *page 5-60*

mach64 Programmer’s Guide:

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

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

5.2.6 Command FIFO

The command FIFO is ‘n’ words deep by 32 bits, where n > 16.
For the RAGE LT PRO, n = 64,128 or 192 as determined by
CMD_FIFO_SIZE_MODE@GUI_CNTL.

	FIFO_STAT																								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
																	a															

a	R	FIFO_STAT	Register represents the occupancy of the last 16 entries in the FIFO, regardless of the actual total FIFO depth.

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 4-5*

GEN_TEST_CNTL on *page 4-21*

mach64 Programmer’s Guide:

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

GUI_CMDFIFO_DEBUG MM: 1_5C

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
	f						e	d	c								b								a							

a	R/W	REG_INDEX	Read: target register for register write
b	R/W	RADR	Read: read pointer value driven by hardware Write (SNOOP mode): read pointer to FIFO
c	R/W	WADR	Read: write pointer value driven by hardware
d	R/W	REN	Read: read enable driven by hardware Write (SNOOP mode): read enable to FIFO
e	R/W	WEN	Read: write enable driven by hardware
f	R/W	SNOOP	SNOOP mode: 0 = disable snoop mode, normal operation 1 = enable snoop mode

GUI_CMDFIFO_DATA MM: 1_5D

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
	a																															

a	R	GUI_CMDFIFO_DATA	Value read from FIFO pointed to by the read pointer

GUI_CNTL MM: 1_5E

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
																															a	

a	R/W	CMDFIFO_SIZE_MODE	Sets CMDFIFO size: 00 = 192 01 = 128 (default) 10 = 64 11 = reserved FIFO must be empty before writing this register Writing to this register should be follwed by a reading from GUI_STAT for proper synchronization

5.2.7 Draw Engine Composite Control

	GUI_TRAJ_CNTL																								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
			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	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_ZREO	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

Cont'd	GUI_TRAJ_CNTL																										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
			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

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.
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_8x8x8_BRUSH	Treats source as an 8x8x8 linear brush (SRC must be QWORD aligned)
u	R/W	FAST_FILL_EN	Fast filling for transparent DST. Not needed if auto-fast-fills are enabled (see HW_DEBUG). Write as '0'
v	R/W	SRC_TRACK_DST	Source will track the trajectory that the DST FIFO is using
w	R/W	PAT_MONO_EN	Monochrome 8x8 pattern enable
x	R/W	PAT_CLR_4x2_EN	Color 4x2 pattern enable
y	R/W	PAT_CLR_8x1_EN	Color 8x1 pattern enable
z	R/W	HOST_BYTE_ALIGN	Enables byte alignment of the host data
aa	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 5-5*

SRC_CNTL on *page 5-21*

PAT_CNTL on *page 5-38*

HOST_CNTL on *page 5-34*

5.2.8 Draw Engine Status

	GUI_STAT																								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
											f										e	d	c	b								a

a	R	GUI_ACTIVE	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_SICISSOR_ BOTTOM	Indicates DSTY is below bottom scissor
f	R	GUI_FIFO	Indicates the number of free DWORDS remaining in the FIFO (See CMDFIFO_SIZE_MODE@GUI_CNTL)

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.

See Also

FIFO_STAT on *page 5-62*

mach64 Programmer’s Guide:

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

Chapter 6
Host Interface

6.1 PCI Configuration Space Registers

The RAGE LT PRO is optimized to support the PCI local bus and to implement the PCI Configuration Space registers. A brief description of the registers with their byte addresses is given below (for detailed descriptions please refer to the *PCI Local Bus Specification*).

PCI configuration reads and writes may be in 8, 16, or 32 bits. A 32-bit read of byte address 0, for example, would read the four bytes 0 to 3.

	VENDOR ID												Byte Addr: 1:0			
BITS	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	a															

a	R	Power-up default = 1002h. This is ATI’s assigned PCI vendor ID.

	DEVICE ID												Byte Addr: 3:2			
BITS	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	a															

a	R	Power-up default = 5654h (ASCII characters ‘VT’), 4754h (ASCII characters ‘GT’), or 4C47h (ASCII characters ‘LG’)

Usage

The 16 bits Device ID (or Chip ID) in the PCI register 2h and CONFIG_CHIP_ID non-GUI register are:

Table 6-1

DEVICE ID	Description
4C42h (LB)	AGP-133/BGA-312 (AGP bus)
4C49h (LI)	PCI-33/BGA-312 (PCI bus)
4C51h(LQ)	PCI-33/BGA-256 (PCI bus)

	COMMAND												Byte Addr: 5:4			
BITS	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								i	h	g	f	e	d	c	b	a

a	R/W	I/O Access Enable. Defaults to 0, disabled.
b	R/W	Memory Access Enable. Defaults to 0, disabled.
c	R/W	Bus Master Enable. Defaults to 0, disabled.
d	R	Special Cycles Enable. Always 0, disabled.
e	R	Memory Write and Invalidate Enable. Always 0, disabled.
f	R/W	VGA Palette Snooping Enable. Defaults to 0, disabled.
g	R	Parity Error Response. Always 0, disabled.
h	R	Read Wait Cycle Control. Always 1.
i	R	SERR# Enable. Always 0, disabled.

	STATUS												Byte Addr: 7:6			
BITS	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	h	g	f	e	d	c			b			a				

a	R	Capability list. Always 1(indicates device implements a list of capabilities)
b	-	Fast Back-to-Back Capable. Always 1.
c	R	DEVSEL Timing. Defaults to 1, Medium.
d	R	Signaled Target Abort. Defaults to 0, no Target Abort.
e	R/W	Received Target Abort. Defaults to 0, inactive.
f	R/W	Received Master Abort. Defaults to 0, inactive.
g	R	Signaled System Error. Always 0, inactive.
h	R	Detected Parity Error. Always 0, inactive.

	ASIC ID						**Byte Addr: 08**	
BITS	7	6	5	4	3	2	1	0
	c		b			a		

a	R	Major ASIC version number (A=0)*
b	R	ASIC foundry ID (000=SGS, 001=NEC, 011=UMC)*
c	R	Minor ASIC revision number*

*Refer to the ASIC ID table below for default values.

Usage

The 8 bits at PCI address 8h are also known as the ASIC ID. The ASIC ID also appears in the CONFIG_CHIP_ID non-GUI register. The following are the ASIC ID's used to date:

Table 6-2

ASIC ID	Description	ASIC ID	Description
08h	NEC VT-A3	5Ah	UMC GT-B2U2
48h	NEC VT-A4	9Ah	UMC VT-B2U3
40h	SGS VT-A4	9Ah	UMC GT-B2U3
01h	SGS VT-B1S1	1Bh	UMC R3B/D/P-A1
01h	SGS GT-B1S1	5Bh	UMC R3B/D/P-A2
41h	SGS GT-B1S2	1Ch	UMC R3B/D/P-A3
1Ah	UMC GT-B2U1	5Ch	UMC R3B/D/P-A4

	REGISTER-LEVEL PROGRAMMING INTERFACE						**Byte Addr: 09**	
BITS	7	6	5	4	3	2	1	0
	a							

a	R	Power-up default = 00h.

	SUB-CLASS CODE							Byte Addr: 0A
BITS	7	6	5	4	3	2	1	0
	a							

a	R	Strap setting. 00h = VGA-compatible, 80h = non-VGA device

	BASE-CLASS CODE							Byte Addr: 0B
BITS	7	6	5	4	3	2	1	0
	a							

a	R	Power-up default = 03h, Display Controller.

	CACHE LINE SIZE							Byte Addr: 0C
BITS	7	6	5	4	3	2	1	0
	a							

a	R/W	Power-up default = 00h.

	LATENCY TIMER							Byte Addr: 0D
BITS	7	6	5	4	3	2	1	0
	a							

a	R/W	Power-up default = 00h.

	HEADER TYPE							Byte Addr: 0E
BITS	7	6	5	4	3	2	1	0
	a							

a	R	Power-up default = 00h.

			BIST				Byte Addr: 0F	
BITS	7	6	5	4	3	2	1	0
	a							

a	R	Power-up default = 00h, not used.

	MEMORY APERTURE BASE ADDRESS																					Byte Addr: 13:10										
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
	d								c																				b	a		

a	R	Reserved. Power-up default = 0h
b	R/W	Memory Prefetch Enable. Default to 0h, (strap setting).
c	R	Reserved. Power-up default = 00000h
d	R/W	Memory aperture base address.

	BLOCK DECODED I/O BASE ADDRESS																							Byte Addr: 17:14									
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
	b																								a							

a	R	Always 01h.
b	R/W	Block Decoded I/O Address.

	REGISTER APERTURE BASE ADDRESS																							Byte Addr: 1B: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
	b																				a											

a	R	Always 000h.
b	R/W	Register Aperture Base Address.

ADAPTER ID — Byte Addr: 2F: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
	a																															

a	R	Power-up default = 69871002h.

BIOS ROM ENABLE — Byte Addr: 33:30

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
	d															c									b							a

a	R/W	BIOS ROM Enable. Defaults to 0
b	R	Reserved. Always 00h.
c	R	Reserved. Always 00h.
d	R/W	BIOS ROM Base Address. Defaults to 0000h

POINTER TO CAPABILITY — Byte Addr: 37:34

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
	▨																								a							

a	R	Pointer to capability. Always 50h.

INTERRUPT LINE — Byte Addr: 3C

BITS	7	6	5	4	3	2	1	0
	a							

a	R/W	Power-up default = 00h.

INTERRUPT PIN — Byte Addr: 3D

BITS	7	6	5	4	3	2	1	0
	a							

a	R	Power-up default = 01h (00h - no default - if interrupt is disabled by strap).

MINIMUM GRANT

Byte Addr: 3E

BITS	7	6	5	4	3	2	1	0
	a							

a	R	Power-up default = 08h.

MAXIMUM LATENCY

Byte Addr: 3F

BITS	7	6	5	4	3	2	1	0
	a							

a	R	Power-up default = 00h.

USER-DEFINED CONFIGURATION

Byte Addr: 40

BITS	7	6	5	4	3	2	1	0
					.a			

a	R/W	Disable decoding of GENENA VGA register at I/O address 46E8h. 0 = decode 46E8h 1 = disable decode of 46E8h Power-up default = 1; **Note:** These bits are set by straps on non-shared configurations.

ADAPTER ID W

Byte Addr: 4F: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
	a																															

a	W	Power-up default = 69871002h.

AGP CAPABILITY

Byte Addr: 53:50

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
									d				c				b								a							

a	R	Cap ID. Always 02h (indicating that this list item is for AGP registers)
b	R	Next Ptr (pointer to next capability). Always 5Ch - Power Management Capability
c	R	AGP MINOR. Always 0
d	R	AGP MAJOR. Always 1 (this chip conforms to AGP spec 1.0)

RATE SBA   Byte Addr: 57:54

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
	c																						b								a	

a	R	Rate. Always 3h (indicates data transfer rate supported <Bit 0:1X, Bit 1:2X>)
b	R	SBA (Side Band Address) support. Always 1
c	R	The maximum number of AGP commands this device can manage. Always FFh.

DATA RATE   Byte Addr: 5B:58

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
	f								e														d	c	b						a	

a	R/W	Data rate: indicates enabled transfer rate <Bit 0:1X, Bit 1:2X>. Only one bit may be on (defaults to 0 after reset).
b	R	Reserved. Always 0
c	R/W	AGP_ENABLE. Enables AGP (defaults to zero after reset).
d	R/W	SBA_ENABLE. Enables Side Band Addressing (defaults to zero after reset).
e	R	Reserved. Always 0.
f	R/W	RQ_DEPTH. The number of requests the device is allowed to enqueue (defaults to zero after reset).

	POWER MANAGEMENT CAPABILITY ID							Byte Addr: 5C
BITS	7	6	5	4	3	2	1	0
	a							

a	R	Power-up default = 1h.

	NEXT POINTER							Byte Addr: 5D
BITS	7	6	5	4	3	2	1	0
	a							

a	R	Power-up default = 0h.

	POWER MANAGEMENT CAPABILITY															Byte Addr: 5F:5E
BITS	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	g					f	e	d			c	b	a			

a	R	PME Clock. Power-up default = 0h
b	R	Auxiliary Power Source. Power-up default = 0h
c	R	DSI. Power-up default = 0h
d	R	(Reserved) Power-up default = 0h
e	R	D1 Support. Power-up default = 1h
f	R	D2 Support. Power-up default = 1h
g	R	PME Support. Power-up default = 0h

	POWER MANAGEMENT CONTROL/STATUS																															Byte Addr: 63:60
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
																																a

a	R/W	Power State. Power-up default = 0h 00 = D0 01 = D1 10 = D2 11 = D3

6.2 Bus Mastering Registers

6.2.1 System Bus Mastering

The following registers are used for controlling and determining the status of the bus mastering operations.

BM_FRAME_BUF_OFFSET **MM: 1_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
									a																							

a	R	FRAME_BUF_OFFSET	Frame buffer byte offset for current bus master operation

BM_SYSTEM_MEM_ADDR **MM: 1_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
	a																															

a	R	FRAME_MEM_ADDRESS	System memory byte address (physical) for current bus master operation

BM_COMMAND **MM: 1_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
	c	b																		a												

a	R	BYTE_COUNT	Number of bytes for transferring (up to 4096)
b	R	FRAME_OFFSET_HOLD	0 = Increment frame buffer offset 1 = Hold frame buffer offset
c	R	END_OF_LIST_STATUS	1 = End of bus master list

BM_STATUS **MM: 1_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
	a																															

a	R	BUS_MASTER_STATUS	Status of current bus master operation

										BM_SYSTEM_TABLE																MM: 1_6F					
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
	b																													a		

a	R/W	SYSTEM_TRIGGER	0 = Transfer system memory data to frame buffer immediately 1 = Transfer frame buffer data to system memory immediately 2 = Transfer frame buffer to system memory on capture Buf0 completion 3 = Transfer frame buffer to system memory on capture Buf1 completion 4 = Transfer frame buffer to system memory on host one-shot completion 5 = Transfer system memory to MPP data port 6-7 = Reserved
b	R/W	SYSTEM_TABLE_ADDR	Physical address [31:4] in system memory for start of SYSTEM Descriptor Table

6.2.2 Draw Engine Bus Mastering

											BM_HOSTDATA															MM: 0_91					
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
	a																															

a	W	BM_HOSTDATA	Bus Master host data register

See Also

BM_ADDR on *page 6-12*

BM_DATA on *page 6-12*

The register below has two purposes. See *Usage.*

BM_ADDR **MM: 0_92**

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
	▨											b													a							

a	W	GUIREG_ADDR	Index offset to desired GUI register
b	W	GUIREG_COUNTER	The number of GUI registers to be consecutively written: register offset will be automatically incremented by 1 DWORD after each write. The value programmed represents the (number of registers + 1). Thus '0' means write a single register.

and

BM_DATA **MM: 0_92**

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
	a																															

a	W	GUIREG_DATA	Data for register at offset specified by BM_ADDR

Description

BM_ADDR determines the 8-bit memory mapped offset for the desired GUI register. BM_DATA is used to write the 32-bit data.

Usage

The BM_ADDR/DATA register is dual-purpose. Note that the BM_ADDR/DATA register is mapped to the same address intentionally to support this type of bus mastering operation. Upon the first write, the address of the desired GUI register is loaded into BM_ADDR. This is an 8-bit address defined by the memory-mapped (MM) offset designated for the register. Only GUI registers in block zero can be loaded in this manner. The second write (to BM_DATA) permits the 32-bit data written to be loaded into the specified register. After this write, the BM returns to address mode to await an the next register address.

Writing any register except BM_ADDR will reset BM_ADDR/DATA to address mode.

This register is extremely useful for transferring a list of register setup information from system memory using the bus master. Simply set up memory as a series of alternating register offset/data pairs (2 DWORDS per pair) and initiate a bus master operation (system –> frame) that tranfers the data in this list to the BM_ADDR register. Ensure that the frame buffer offset points to the BM_ADDR/DATA register and the FRAME_OFFSET_HOLD bit is set in the BM_COMMAND DWORD entry for the current descriptor.

See Also

BM_HOSTDATA on *page 6-11*

BM_GUI_TABLE MM: 1_6E

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
	b																														a	

a	R/W	CIRCULAR_BUF_SIZE	0 = 16KB (1K entries) 1 = 32KB (2K entries) 2 = 64KB (4K entries) 3 = 128KB (8K entries)
b	R/W	GUI_TABLE_ADDR	Physical address [31:4] in system memory for start of GUI Descriptor Table

Usage

The GUI_TABLE_ADDR will wrap when it internally reaches a 16KB, 32KB, 64KB, or 128KB boundary (according to its circular buffer size).

BM_GUI_TABLE_CMD MM: 0_93

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
	b																														a	

a	R/W	CIRCULAR_BUF_SIZE	0 = 16KB (1K entries) 1 = 32KB (2K entries) 2 = 64KB (4K entries) 3 = 128KB (8K entries)
b	R/W	GUI_TABLE_ADDR	Physical address [31:4] in system memory for start of GUI Descriptor Table

Usage

BM_GUI_TABLE_CMD is an alias for BM_GUI_TABLE. The distinction is that BM_GUI_TABLE_CMD goes through the GUI command FIFO and is thus synchronized to the command stream.

6.3 AGP Registers

	AGP_BASE																								**MM: 1_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
	a																															

a	R/W	AGP_BASE_ADDR	AGP Base address: AGP base address = bits 31:22, when aperture size = 4MB AGP base address = bits 31:23, when aperture size = 8MB AGP base address = bits 31:24, when aperture size = 16MB AGP base address = bits 31:25, when aperture size = 32MB AGP base address = bits 31:26, when aperture size = 64MB AGP base address = bits 31:27, when aperture size = 128MB AGP base address = bits 31:28, when aperture size = 256MB

	AGP_CNTL																								**MM: 1_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
															d	c	b										a					

a	R/W	AGP_APER_SIZE	AGP Aperture size: 111111 = 4MB (default) 111110 = 8MB 111100 = 16MB 111000 = 32MB 110000 = 64MB 100000 = 128MB 000000 = 256MB (default)
b	R/W	MAX_IDLE_CLK	This is the number of clocks (MAX_IDLE_CLK times 32) that the AGP block will wait before stopping the generation of the 2X sideband strobe after it no longer has a request to service. default = 0 (disabled)
c	R/W	HIGH_PRIORITY_READ_EN	1 = generate all read as high priority reads 0 = generate all read at their default priority (default)
d	R/W	AGP_TRDY_MODE	0 = DATA comes in clock after TRDY sampled 1 = DATA comes in clock in which TRDY sampled (default)

Chapter 7
VGA-Compatible Registers

7.1 VGA Compatible Registers Summary – By I/O Port

VGA registers in the RAGE LT PRO are fully hardware-compatible with registers in the IBM VGA video adapter. They are grouped and described in details on the pages as indicated in the table below.

Table 7-1

Register Functional Classes	Page
CRT Controller Registers (GRAxx)	*7-5*
Attribute Controller Registers (ATTRxx)	*7-17*
General Status and Configuration Registers (GENxx)	*7-22*
Sequencer Registers (SEQxx)	*7-26*
DAC Registers (DACxx)	*7-30*
Graphics Controller Registers (GRAxx)	*7-31*

Also, for convenience, they are listed by I/O port address (and by index) in table 9-2 on the next three pages.

Note:
under the Port column:
? = B when GENMO[0]=0 (Monochrome emulation).
? = D when GENMO[0]=1 (Color/Graphics emulation).

Table 7-2 VGA Compatible Registers Reference List

Port	Index	Function	Type	Mnemonic	Register Name	Page
0102	–	General	W	GENVS	VGA Sleep	*7-22*
03?4	–	CRT Controller	R/W	CRTX	CRTC Index	*7-5*
03?5	0		R/W	CRT00	Horizontal Total	*7-5*
03?5	1		R/W	CRT01	Horizontal Display Enable End	*7-5*
03?5	2		R/W	CRT02	Start Horizontal Blanking	*7-6*
03?5	3		R/W	CRT03	End Horizontal Blanking	*7-6*
03?5	4		R/W	CRT04	Start Horizontal Retrace	*7-6*
03?5	5		R/W	CRT05	End Horizontal Retrace	*7-7*
03?5	6		R/W	CRT06	Vertical Total	*7-7*
03?5	7		R/W	CRT07	CRTC Overflow	*7-7*
03?5	8		R/W	CRT08	Preset Row Scan	*7-8*
03?5	9		R/W	CRT09	Maximum Scan Line	*7-9*
03?5	A		R/W	CRT0A	Cursor Start	*7-9*
03?5	B		R/W	CRT0B	Cursor End	*7-10*
03?5	C		R/W	CRT0C	Start Address (High Byte)	*7-10*
03?5	D		R/W	CRT0D	Start Address (Low Byte)	*7-11*
03?5	E		R/W	CRT0E	Cursor Location (High Byte)	*7-11*
03?5	F		R/W	CRT0F	Cursor Location (Low Byte)	*7-11*
03?5	10		R/W	CRT10	Start Vertical Retrace	*7-12*
03?5	11		R/W	CRT11	End Vertical Retrace	*7-12*
03?5	12		R/W	CRT12	Vertical Display Enable End	*7-13*
03?5	13		R/W	CRT13	Offset	*7-13*
03?5	14		R/W	CRT14	Underline Location	*7-13*
03?5	15		R/W	CRT15	Start Vertical Blanking	*7-14*
03?5	16		R/W	CRT16	End Vertical Blanking	*7-14*
03?5	17		R/W	CRT17	CRT Mode	*7-15*
03?5	18		R/W	CRT18	Line Compare	*7-16*
03?5	1E,1F		R	CRT1E, 1F	Graphic Controller Index Decode	*7-16*
03?5	22		R	CRT22	RAM Data Latch Readback	*7-17*
03?A	–	General	W	GENFC	Feature Control	*7-22*
03?A	–		R	GENS1	Input Status 1	*7-23*

Table 7-2 VGA Compatible Registers Reference List Cont'd

Port	Index	Function	Type	Mnemonic	Register Name	Page
03C0	–	Attribute Controller	R/W	ATTRX	Attribute Controller Index	7-17
03C0	00-0F		W	ATTR(00:0F)	Palette (00 to 0F)	7-18
03C0	10		W	ATTR10	Mode Control	7-18
03C0	11		W	ATTR11	Overscan Color	7-19
03C0	12		W	ATTR12	Color Map Enable	7-20
03C0	13		W	ATTR13	Horizontal PEL Panning	7-20
03C0	14		W	ATTR14	Color Select	7-21
03C1	00-0F		R	ATTR(00:0F)	Palette (00 to 0F)	7-18
03C1	10		R	ATTR10	Mode Control	7-18
03C1	11		R	ATTR11	Overscan Color	7-19
03C1	12		R	ATTR12	Color Map Enable	7-20
03C1	13		R	ATTR13	Horizontal PEL Panning	7-20
03C1	14		R	ATTR14	Color Select	7-21
03C2	–	General	W	GENMO	Miscellaneous Output	7-23
03C2	–		R	GENSO	Input Status 0	7-24
03C3	–		R	GENENB	Video Subsystem Enable (Board)	7-24
03C4	–	Sequencer	R/W	SEQX	Sequencer Index	7-26
03C5	0		R/W	SEQ00	Reset	7-26
03C5	1		R/W	SEQ01	Clock Mode	7-26
03C5	2		R/W	SEQ02	Map Mask	7-27
03C5	3		R/W	SEQ03	Character Map Select	7-28
03C5	4		R/W	SEQ04	Memory Mode	7-29
03C6	–	DAC	R/W	DAC_MASK	DAC Mask	7-30
03C7	–		R/W	DAC_R_INDEX	DAC Read Current Color Index	7-30
03C8	–		R/W	DAC_W_INDEX	DAC Write Current Color Index	7-30
03C9	–		R/W	DAC_DATA	DAC Data	7-31
03CA	–	General	R	GENFC	Feature Control	7-22
03CC	–		R	GENMO	Miscellaneous Output	7-23

Table 7-2 VGA Compatible Registers Reference List Cont'd

Port	Index	Function	Type	Mnemonic	Register Name	Page
03CE	–	Graphics Controller	R/W	GRAX	Graphics Controller Index	7-31
03CF	0		R/W	GRA00	Set/Reset	7-31
03CF	1		R/W	GRA01	Enable Set/Reset	7-32
03CF	2		R/W	GRA02	Color Compare	7-33
03CF	3		R/W	GRA03	Data Rotate	7-34
03CF	4		R/W	GRA04	Read Map Select	7-34
03CF	5		R/W	GRA05	Graphics Mode	7-35
03CF	6		R/W	GRA06	Graphics Miscellaneous	7-36
03CF	7		R/W	GRA07	Color Don't Care	7-37
03CF	8		R/W	GRA08	Bit Mask	7-38
46E8	–	General	W	GENENA	Video Subsystem Enable (Add-On)	7-25

The following sections describe the registers grouped under the six register classes mentioned earlier. **I/O** indicates the read and write address of the register, and **Index** is included if the register is accessible indirectly.

7.2 VGA CRT Controller Registers

In the I/O address, note that:
? = B when GENMO[0]=0 (Monochrome emulation).
? = D when GENMO[0]=1 (Color/Graphics emulation).

	CRTC INDEX (CRTX)					I/O: 3?4		INDEX:–
BITS	7	6	5	4	3	2	1	0
			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.					

	HORIZONTAL TOTAL (CRT00)					I/O: 3?5		INDEX: 00
BITS	7	6	5	4	3	2	1	0
	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.					

	HORIZONTAL DISPLAY ENABLE END (CRT01)					I/O: 3?5		INDEX: 01
BITS	7	6	5	4	3	2	1	0
	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					

START HORIZONTAL BLANKING (CRT02) I/O: 3?5 INDEX: 02

BITS	7	6	5	4	3	2	1	0
	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.

END HORIZONTAL BLANKING (CRT03) I/O: 3?5 INDEX: 03

BITS	7	6	5	4	3	2	1	0
	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

START HORIZONTAL RETRACE (CRT04) I/O: 3?5 INDEX: 04

BITS	7	6	5	4	3	2	1	0
	a							

a	R/W	H_SYNC_START[7:0]	These bits define the horizontal character count at which the horizontal retrace pulse becomes active.

END HORIZONTAL RETRACE (CRT05) I/O: 3?5 INDEX: 05

BITS	7	6	5	4	3	2	1	0
	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]

VERTICAL TOTAL (CRT06) I/O: 3?5 INDEX: 06

BITS	7	6	5	4	3	2	1	0
	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.

CRTC OVERFLOW (CRT07) I/O: 3?5 INDEX: 07

BITS	7	6	5	4	3	2	1	0
	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).

Cont'd

CRTC OVERFLOW (CRT07) I/O: 3?5 INDEX: 07

BITS	7	6	5	4	3	2	1	0
	h	g	f	e	d	c	b	a

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)

PRESET ROW SCAN (CRT08) I/O: 3?5 INDEX: 08

BITS	7	6	5	4	3	2	1	0
		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).

	MAXIMUM SCAN LINE (CRT09)					I/O: 3?5		INDEX: 09
BITS	7	6	5	4	3	2	1	0
	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.

	CURSOR START (CRT0A)					I/O: 3?5		INDEX: 0A
BITS	7	6	5	4	3	2	1	0
			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.

CURSOR END (CRT0B) I/O: 3?5 INDEX: 0B

BITS	7	6	5	4	3	2	1	0
		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.

START ADDRESS (HIGH BYTE) (CRT0C) I/O: 3?5 INDEX: 0C

BITS	7	6	5	4	3	2	1	0
	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.

START ADDRESS (LOW BYTE) (CRT0D) I/O: 3?5 INDEX: 0D

BITS	7	6	5	4	3	2	1	0
	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.

CURSOR LOCATION (HIGH BYTE) (CRT0E) I/O: 3?5 INDEX: 0E

BITS	7	6	5	4	3	2	1	0
	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.

CURSOR LOCATION (LOW BYTE) (CRT0F) I/O: 3?5 INDEX: 0F

BITS	7	6	5	4	3	2	1	0
	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.

START VERTICAL RETRACE (CRT10)   I/O: 3?5   INDEX: 10

BITS	7	6	5	4	3	2	1	0
	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.

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?5   INDEX: 11

BITS	7	6	5	4	3	2	1	0
	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

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?5 INDEX: 12

BITS	7	6	5	4	3	2	1	0
	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.

OFFSET (CRT13) I/O: 3?5 INDEX: 13

BITS	7	6	5	4	3	2	1	0
	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.

UNDERLINE LOCATION (CRT14) I/O: 3?5 INDEX: 14

BITS	7	6	5	4	3	2	1	0
		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].

Cont'd

UNDERLINE LOCATION (CRT14) I/O: 3?5 INDEX: 14

BITS	7	6	5	4	3	2	1	0
		c	b	a				

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]

START VERTICAL BLANKING (CRT15) I/O: 3?5 INDEX: 15

BITS	7	6	5	4	3	2	1	0
	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.

END VERTICAL BLANKING (CRT16) I/O: 3?5 INDEX: 16

BITS	7	6	5	4	3	2	1	0

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.

	CRT MODE (CRT17)					I/O: 3?5		INDEX: 17
BITS	7	6	5	4	3	2	1	0
	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.

Cont'd

CRT MODE (CRT17) I/O: 3?5 INDEX: 17

BITS	7	6	5	4	3	2	1	0
	g	f	e		d	c	b	a

g	R/W	CRTC_SYNC_EN	H/V Retrace Enable: 0 = Disables horizontal and vertical retrace 1 = Enables horizontal and vertical retrace

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?5 INDEX: 18

BITS	7	6	5	4	3	2	1	0
	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].)

GRAPHICS CONTROLLER INDEX DECODE (CRT1E,1F) I/O: 3?5 INDEX:1E,1F

BITS	7	6	5	4	3	2	1	0
	a							

a	R	GRPH_DEC[8:0]	This register is used to read back the graphics controller index decode.

	RAM DATA LATCH READBACK (CRT22)					I/O: 3?5		INDEX: 22
BITS	7	6	5	4	3	2	1	0
	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.

7.3 VGA Attribute Controller Registers

Notes:
After initialization, OUT commands toggle between writing to the ATTRX and the indexed Attribute registers.
The Attribute registers operate with the Palette registers to establish the video DAC PEL definition.

	ATTR INDEX (ATTRX)					I/O: 3C0		INDEX:–
BITS	7	6	5	4	3	2	1	0
			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.

	PALETTE 00-0F (ATTR00_0F)					I/O: 3C1(R) 3C0(W)		INDEX: 00 to 0F
BITS	7	6	5	4	3	2	1	0
			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: 3C1(R) 3C0(W)		INDEX: 10
BITS	7	6	5	4	3	2	1	0
	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.

Cont'd	MODE CONTROL (ATTR10)					I/O: 3C1(R) 3C0(W)		INDEX: 10
BITS	7	6	5	4	3	2	1	0
	g	f	e		d	c	b	a

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: 3C1(R) 3C0hW)		INDEX: 11
BITS	7	6	5	4	3	2	1	0
	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: 3C1(R)
3C0(W) INDEX: 12

BITS	7	6	5	4	3	2	1	0
			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: 3C1(R)
3C0(W) INDEX: 13

BITS	7	6	5	4	3	2	1	0
					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. (See table 9-3 below.)

Table 7-3

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: 3C1(R)** **3C0(W)**		**INDEX: 14**
BITS	7	6	5	4	3	2	1	0
					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.

7.4 General VGA Status and Configuration Registers

	VGA SLEEP (GENVS)					I/O:102		INDEX:–
BITS	7	6	5	4	3	2	1	0
								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: 3CA(R) 3?A(W)		INDEX:–
BITS	7	6	5	4	3	2	1	0
					a			

a	R/W	VSYNC_SEL	Vertical Sync Select: 0 = Normal vertical sync 1 = Sync is ‘vertical sync’ ORed’ vertical display enable’

INPUT STATUS 1 (GENS1)   I/O: 3?A   INDEX:–

BITS	7	6	5	4	3	2	1	0
			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
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

Note:
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: 3CC(R) 3C2(W)   INDEX:–

BITS	7	6	5	4	3	2	1	0
	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

Cont'd

MISCELLANEOUS OUTPUT (GENMO) I/O: 3CC(R) 3C2(W) INDEX:–

BITS	7	6	5	4	3	2	1	0
	e		d		c		b	a

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: 3C2 INDEX:–

BITS	7	6	5	4	3	2	1	0
	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: 3C3 INDEX:–

BITS	7	6	5	4	3	2	1	0
								a

a	R	VGA_ENABLE1	VGA Enable: Read back status of GENVS[0](0102)

VIDEO SUBSYSTEM ENABLE (ADD ON) (GENENA) I/O: 46E8 INDEX:–

BITS	7	6	5	4	3	2	1	0
				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:

The decode of this register is optionally controlled by the PCI configuration space. Refer to *Chapter 8, PCI Configuration Registers*.

7.5 VGA Sequencer Registers

	SEQUENCER INDEX (SEQX)					I/O: 3C4		INDEX:–
BITS	7	6	5	4	3	2	1	0
						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: 3C5		INDEX: 00
BITS	7	6	5	4	3	2	1	0
							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: 3C5		INDEX: 01
BITS	7	6	5	4	3	2	1	0
			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.

Cont'd

CLOCK MODE (SEQ01) I/O: 3C5 INDEX: 01

BITS	7	6	5	4	3	2	1	0
			e	b	d	c		a

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: 3C5 INDEX: 02

BITS	7	6	5	4	3	2	1	0
					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

Cont'd

MAP MASK (SEQ02) I/O: 3C5 INDEX: 02

BITS	7	6	5	4	3	2	1	0
					d	c	b	a

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: 3C5 INDEX: 03

BITS	7	6	5	4	3	2	1	0
			f	e	d	c	b	a

a, b, e	R/W	SEQ_FONTB[2:0]	Character Map Select B Bits 2:0
c, d, 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: 3C5	INDEX: 04
BITS	3	2	1	0
	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]

7.6 VGA DAC Registers

	DAC MASK (DAC_MASK)					**I/O: 03C6**		**INDEX:–**
BITS	7	6	5	4	3	2	1	0
	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: 03C7**		**INDEX:–**
BITS	7	6	5	4	3	2	1	0
	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: 03C8**		**INDEX:–**
BITS	7	6	5	4	3	2	1	0
	a							

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: 03C9		INDEX:–
BITS	7	6	5	4	3	2	1	0
	a							

a	R/W	DAC_DATA	DAC Data

7.7 VGA Graphics Controller Registers

	GRAPHICS CONTROLLER INDEX (GRAX)					I/O: 3CE		INDEX:–
BITS	7	6	5	4	3	2	1	0
					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: 3CF		INDEX: 00
BITS	7	6	5	4	3	2	1	0
					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.

Cont'd

SET/RESET (GRA00) I/O: 3CF INDEX: 00

BITS	7	6	5	4	3	2	1	0
					d	c	b	a

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: 3CF INDEX: 01

BITS	7	6	5	4	3	2	1	0
					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.

Cont'd

ENABLE SET/RESET (GRA01) I/O: 3CF INDEX: 01

BITS	7	6	5	4	3	2	1	0
					d	c	b	a

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: 3CF INDEX: 02

BITS	7	6	5	4	3	2	1	0
					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: 3CF INDEX: 03

BITS	7	6	5	4	3	2	1	0
				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: 3CF INDEX: 04

BITS	7	6	5	4	3	2	1	0
							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: 3CF		INDEX: 05
BITS	7	6	5	4	3	2	1	0
		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.

Cont'd

GRAPHIC MODE (GRA05) I/O: 3CF INDEX: 05

BITS	7	6	5	4	3	2	1	0
		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: 3CF INDEX: 06

BITS	7	6	5	4	3	2	1	0
					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.

Cont'd

GRAPHICS MISCELLANEOUS (GRA06) I/O: 3CF INDEX: 06

BITS	7	6	5	4	3	2	1	0
					c		b	a

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: 3CF INDEX: 07

BITS	7	6	5	4	3	2	1	0
					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: 3CF		INDEX: 08
BITS	7	6	5	4	3	2	1	0
	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.

Chapter 8
LCD Panel

8.1 LCD Panel Registers

The LCD panel registers can only be accessed through memory mapped I/O and relocatable block I/O space. All registers are indexed.

8.1.1 Index and Data

	LCD_INDEX																								**Offset: 0_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
LT																						d	c	b					a			

a	R/W	LCD_REG_INDEX	Selects register in ltreg block to read or write. Access to LCD_DATA register (offset 2A) is using this index to read or write.
b	R/W	LCD_DISPLAY_DIS	Disables the display, forcing the LCD data to be 0
c	R/W	LCD_SRC_SEL	Selects display path for LCD out 0 - CRTC1 (primary display) 1 - CRTC2 (secondary display)
d	R/W	CRTC2_DISPLAY_DIS	Disables the display from second display path by forcing blank signal

	LCD_DATA																								**Offset: 0_2A**							
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
LT	a																															

a	R/W	LCD_REG_DATA	Data that will be written or read from the indexed LCD registers. Indexed registers are described on the following pages.

8.1.2 Configuration and Timing

	CONFIG_PANEL																					Offset: 0_2A Index: 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
LT	p				o		n		m	l	k		j	i	h	g		f	e	d		c			b					a		

a	R/W	PANEL_FORMAT	Specifies what kind of pixel format the LCD panel uses. For split-panel color STN panels. Panel_format Function 000 PACK6 (12-bit interface, 6-bit to upper panel, 6-bit to lower panel) 001 PACK8 (16-bit interface, 8-bit to upper panel, 8-bit to lower panel) 010 PACK12 (24-bit interface, 12-bit to upper panel, 12-bit to lower panel) For single-panel color STN panels. Panel_format Function 000 PACK12 (12-bit interface) 001 PACK16 (16-bit interface) For TFT panels Panel_format Panel Selected 000 8-color panel - 111 RGB 001 512-color panel - 333 RGB 010 4096-color panel - 444 RGB 100 18-bit/pixel panel - 666 RGB (LT mode) 101 24-bit/pixel panel - 888 RGB 110 18-bit/pixel panel - 666 RGB (FPDI-2 mode) (Combinations not specified are reserved)
b	R/W	PANEL_TYPE (Monochrome panel is not supported)	Specifies what kind of panel is being used. Panel_type Panel Type Selected 0001 Split Panel STN Color 0011 Single Panel STN Color 010x Reserved 0111 Color TFT (1 pixel per clock) 1111 Color TFT (2 pixels per clock) **(**Combinations not specified are reserved)
c	R/W	NO_OF_GREY	Determines how many levels of grey levels will be supported. (In the case of a color panel, it is the number of grey levels per color component) 000 = Indicates no frame modulation should be done. (applies only to TFT panels) 001 = 2 levels of grey support (applies only to TFT panels) 010 = 4 levels of grey support (applies only to TFT panels) 011 = 8 levels of grey support (applies only to STN panels) 100 = 16 levels of grey support (applies only to STN panels) 101 = 32 levels of grey support (applies only to monochrome STN panels) 110 = 64 levels of grey support (applies only to STN panels) 111 = Reserved

Cont'd

CONFIG_PANEL Offset: 0_2A Index: 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
LT	p				o		n		m	l	k		j	i	h	g		f	e	d		c			b					a		

d	R/W	EXT_LVDS_CLK	Clock select for external LVDS/Panel Link with DSTN panel 00 - disabled 01 - output VCLK on LCDTMG(0) pin 10 - output VCLK/2 on LCDTMG(0) pin 11 - reserved
e	R/W	BLINK_RATE	Cursor blink: 0 = same as CRT 1 = Blink every 32 frames
f	R/W	DONT_SHADOW_HEND	0 = Use shadowed value for horizontal display end 1 = Use non shadow value for horizontal display end
g	R/W	FP_POL	0 = active high Frame Pulse / Vsync 1 = active low Frame Pulse / Vsync
h	R/W	LP_POL	0 = active high Line Pulse / Hsync 1 = active low Line Pulse / Hsync
i	R/W	DTMG_POL	0 = active high Display Enable / MOD 1 = active low Display Enable / MOD
j	R/W	SCK_POL	0 = active high Shift Clock / PCLK 1 = active low Shift Clock / PCLK
k	R/W	DITHER_SEL	00 = Disable dithering 01 = Dither to 4 bits 10 = Dither to 5 bits 11 = Dither to 6 bits
l	R/W	INVERSE_VIDEO_EN	0 = Disable inverse video 1 = Enable inverse video
m	R/W	BL_CLK_SEL	Backlight modulation clock selection: 0 = 32KHz (version A22) or 29MHz reference clock (from version A31 on) 1 = 32khz divided by 3 or 29MHz reference clock divided by 3 (from version A31 on)
n	R/W	BL_LEVEL (Up to version A22 inclusive) (Bit 24 reserved from version A31 on)	Backlight brightness level (PWM of the BLON signal): Up to version A22 00 = dimmest ... 11 = brightest From version A31 on: 0 = the 256 levels are controled by the value programmed in ZVGPIO register 1 = disable PWM on BLON pin

Cont'd

	CONFIG_PANEL																						Offset: 0_2A Index: 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
LT	p				o		n		m	l	k		j	i	h	g		f	e	d		c			b					a		

o	R/W	BIAS_LEVEL	Contrast level (PWM of the BLON signal) 00 = dimmest ... 11 = brightest
p	R/W	HSYNC_DELAY(3:0)	Hsync delay for the LCD Panel: 0000 = no delay 0001 = delay by 1 VCLK ... 1111 = delay by 15 VCLKs

Description

CONFIG_PANEL is used to configure the LCD Engine in order to support different types of LCD panels.

Usage

LCD Engine configuration should be done in the adapter BIOS only and before LCD panel is turned on. Most of the parameters depend on the panel type that is connected to the Graphics Controller. Panel type can be detected by reading PANEL_ID field in the CONFIG_STAT0 register. Only the adapter BIOS should touch this register.

See Also

LCD_GEN_CTRL below; specifications for the LCD panel that will be connected to the 3D RAGE LT PRO Graphics Controller; and documents about LCD panel interface.

8.1.3 General Control

	LCD_GEN_CTRL																						Offset: 0_2A Index: 1									
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
LT	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	CRT_ON	0 = CRT off 1 = CRT on
b	R/W	LCD_ON	0 = LCD off 1 = LCD on

Cont'd

LCD_GEN_CTRL Offset: 0_2A Index: 1

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
LT	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

c	R/W	HORZ_DIVBY2_EN	This bit is to control the following horizontal CRTC parameters: Htotal, Hdisp, Hblank,Hsync and SHsync 0 = Do not divide horizontal CRT parameters when shadow registers are used and SEQPCLKBY2 is active 1 = Divide horizontal CRT parameters when SEQ_PCLKBY2 = ‘1’
d	R/W	DONT_DS_ICON	Double scan hardware icon: 0 = expand the icon id double scan mode (default) 1 = do not expand the icon in double scan mode
e	R/W	LOCK_8DOT	0 = Character clock can be 9-dot or 8-dot 1 = Always 8-dot per character
f	R/W	ICON_ENABLE	0 = Hardware cursor is enabled. 1 = Hardware icon is enabled.
g	R/W	DONT_SHADOW_VPAR	This bit is to control the use of Vsync, Vblank, and Vtotal register values. 0 = Use shadowed values for vertical CRT parameters 1 = Use non-shadowed values for vertical CRT parameters
h	R/W	V2CLK_ALWAYS_ONb	0 - V2CLK is on regardless of CRTC2_PIX_WIDTH field 1 - V2CLK is off if CRTC2_PIX_WIDTH is 0, otherwise V2CLK is on
i	R/W	RST_FM	0 = Enable frame modulation circuitry to function. 1 = Resets the frame modulation circuit.
j	R/W	DIS_HOR_CRT_DIVBY2	0 = use non shadow H sync for CRT if HORZ_DIVBY2_EN = 1 in 40 column VGA mode 1 = use divided shadow H sync for CRT
k	R/W	SCLK_SEL	0 = Select the divided LCD_VCLK as the LCD output shift clock 1 = Select the PLL output as the LCD output shift clock
l	R/W	SCLK_DELAY	Adjust setup/hold time for LCD data (1 ns increment) 0 – 7 = SCLK comes before LCD data 8 = SCLK and LCD data have the same delay (default after reset) 9 – 15 = SCLk comes after LCD data
m	R/W	TVCLK_ALWAYS_ONb	0 = TVCLK is on regardless of the TV_ON bit 1 = TVCLK is off if TV_ON is 0 otherwise TVCLK is on
n	R/W	VCLK_DAC_ALWAYS_ONb	0 = VCLK_DAC is always on 1 = VCLK_DAC is off druing blank time

Cont'd

LCD_GEN_CTRL **Offset: 0_2A Index: 1**

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
LT	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

o	R/W	VCLK_LCD_OFF	0 = VCLK_LCD is on regardless of the LCD_ON bit 1 = VCLK_LCD is off if LCD_ON is 0 otherwise VCLK_LCD is on
p	R/W	SELECT_WAIT_4MS (From A22 version)	0 = Wait 8 CPUCLK for PLL to be stable when going out from Suspend Mode 1 = Wait 4 ms for PLL to be stable when going out from Suspend Mode
q	R/W	V2CLK_DAC_ALWAYS_ON	0 - V2CLK_DAC is always on 1 - V2CLK_DAC is off during blank time
r	R/W	LVDS_EN	0 = On chip LVDS interface is disabled 1 = On chip LVDS interface is enabled
s	R/W	LVDS_PLL_EN	0 = Disable LVDS PLL 1 = Enable LVDS PLL
t	R/W	LVDS_PLL_RESET	Reset LVDS PLL
u	R/W	LVDS_RESERVED_BITS	Controls which value you want to output on the LVDS_B reserved bits
v	R/W	CRTC_RW_SELECT	0 = All CRTC register reads/writes go to primary CRT 1 = All CRTC register reads/writes through non VGA space go to secondary CRT
w	R/W	USE_SHADOWED_VEND	0 = Disable the shadow vertical end register 1 = Use the shadowed vertical end register CRTC_V_TOTAL_DISP(26:16)
x	R/W	USE_SHADOWED_ROWCUR	0 = Disable the shadow VGA cursor start and end, max_row_scan, underline register 1 = Use the shadow VGA cursor start and end, max_row_scan, underline register CRT0A bit 4:0, CRT0B bit 4:0, CRT9 bit 4:0, CRT14 bit 4:0
y	R/W	SHADOW_EN	0 = Use the normal CRTC registers to generate the CRT timing. 1 = Use the shadow registers to generate the CRT timing. (except shadow of CRT09(4,0), CRT14, CRTA, CRTB, CRTC_V_TOTAL_DISP(26:16))
z	R/W	SHADOW_RW_EN	0 = Disable read/write to the panel shadow registers. 1 = Enable read/write to the panel shadow registers.

Description

LCD_GEN_CTRL is used for general LCD panel configuration and control as implemented by the 3D RAGE LT PRO, and also to control the shadowed CRTC registers.

8.1.4 Dual Scan

	DSTN_CONTROL																					Offset: 0_2A Index: 2										
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
LT	h		g		f		e		d	c	b											a										

a	R/W	FP_POS	This register defines the position of the frame pulse for the LCD panel. The frame pulse happens one line after vertical count hits this value. If top_overscan is 0, then FP_POS = 0; otherwise FP_POS = VTOTAL - top_overscan + 1 (ADD 1 in VGA mode)
b	R/W	LOWER_PANEL_VPOS	This is the line where the LCD display switches from upper panel to lower panel. This panel is only used when driving a split panel. LOWER_PANEL_VPOS = {[no_of _active_display_lines + no_of_top_overscan – no_of_bottom_overscan] / 2} – 1
c	R/W	AUTO_LOWER_PANEL _VPOS	0 = use register value from LOWER_PANEL_VPOS 1 = calculate value based on Vertical Display End
d	R/W	USE_ADJUST0	0 = LP_VPOS_ADJUST0 /1/2/3 will be used if the value of the Vertical End register is 350/400/480/600 respectively 1 = use LP_VPOS_ADJUST0
e	R/W	LP_VPOS_ADJUST0	Adjust position of the vsync during ratiometric expansion (0 = no shift, 1 = shift by 1 line, and so on.)
f	R/W	LP_VPOS_ADJUST1	Adjust position of the vsync during ratiometric expansion (0 = no shift, 1 = shift by 1 line, and so on.
g	R/W	LP_VPOS_ADJUST2	Adjust position of the vsync during ratiometric expansion (0 = no shift, 1 = shift by 1 line, and so on.
g	R/W	LP_VPOS_ADJUST3	Adjust position of the vsync during ratiometric expansion (0 = no shift, 1 = shift by 1 line, and so on.

Description

DSTN_CONTROL register is used to configure sync signals for dual-scan STN panels. This register has to be programmed only if dual-scan STN panel is used.

Usage

Dual-scan STN panel configuration should be done in the adapter BIOS only.

See Also

CONFIG_PANEL (fields PANEL_TYPE, PANEL_FORMAT) on *page 8-2*:

8.1.5 Half Frame Buffer

	HFB_PITCH_ADDR																										Offset: 0_2A Index: 3					
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
LT													d			c					b				a							

a	R/W	XBUF_BASE	Base address of external frame buffer (bits 0XBUF_ADDRESS[22:15]). This will put the half frame buffer on a 32K byte boundary in the physical memory.
b	R/W	XBUF_SIZE	Defines the size of the half frame buffer in 32K increments 0 = size is unlimited 1-14 = size in 32K blocks 15 = stop write to the half frame buffer unconditionally
c	R/W	CRT_SYNC_SEL	Defines which CRT values should be used in centering mode for CRT monitor. 0 = Use non-shadow values for CRT syncs in centering mode. 1 = Use shadow (LCD) values for CRT syncs in centering mode
d	R/W	HFB_HW	Half frame buffer high watermark

Description

XBUF_PITCH_ADDR is used to configure the external frame buffer inside LCD Engine. This register has to be programmed only if dual-scan STN panel is used.

Usage

External frame buffer configuration should be done in the adapter BIOS only and it is used only for dual-scan STN panels. Base address of the external frame buffer should be placed somewhere in off-screen memory. The memory size used for half frame could be calculated as:

Memory_size [byte] = Horiz_resolution × Vertical_resolution × 3/8

See Also

CONFIG_PANEL (fields PANEL_TYPE, PANEL_FORMAT) on *page 8-2*.

8.1.6 Horizontal Stretching

	HORZ_STRETCHING																									Offset: 0_2A Index: 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
LT	d	c												b			a															

a	R/W	HORZ_STRETCH_RATIO	Horizontal stretch ratio. The value in the register is shifted out serially. (the LSB first, i.e., bit 0). If the bit that is shifted out is 0, the same pixel is duplicated. If the bit that is shifted out is 1, the next pixel is sent out. The bit 0 is looped back to either bit 9,12,14 or 15 depending on the value of the LOOP_STRETCH bit (bit 0 has to be programmed to zero). When HORZ_BLEND_EN is 1, HORZ_STRETCH_RATIO is used as a ratio for horizontal blender adder. In that case, only 12 least significant bits are used and the ratio should be: HORZ_STRETCH_RATIO = [source_width/dest_width] * 4096.
b	R/W	LOOP_STRETCH	Horizontal stretching shift register loop back select 000 = Loop bit 0 back to bit 9 001 = Loop bit 0 back to bit 11 010 = Loop bit 0 back to bit 12 011 = Loop bit 0 back to bit 14 100 = Loop bit 0 back to bit 15 Others = Reserved
c	R/W	HORZ_STRETCH_MODE	Horizontal stretch mode: 0 = pixel replication (use shift register) 1 = horizontal blending
d	R/W	HORZ_STRETCH_EN	Horizontal stretch enable: 1 = Horizontal stretching is enabled

Description

HORIZ_STRETCHING is used to define parameters for horizontal stretching (expansion).

Usage

In order to support ratiometric expansion, HORZ_STRETCHING, VERT_STRETCHING and EXT_VERT_STRETCH registers have to be programmed. The values that have to be programmed in these registers depend on the panel resolution and the current graphics mode resolution. Ratiometric expansion in horizontal direction can be done in two ways — pixel replication or blending.

See Also

VERT_STRETCHING (next) and EXT_VERT_STRETCH. (after next)

8.1.7 Vertical Stretching

	VERT_STRETCHING																					**Offset: 0_2A Index: 5**										
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
LT	e	d	c										b										a									

a	R/W	VERT_STRETCH_RATIO0	Vertical stretch ratio: VERT_STRETCH_RATIO = [no_of_lines_of_source / no_of_lines_of _destination] * 1024
b	R/W	VERT_STRETCH_RATIO1	Vertical stretch ratio: VERT_STRETCH_RATIO = [no_of_lines_of_source / no_of_lines_of _destination] * 1024
c	R/W	VERT_STRETCH_RATIO2	Vertical stretch ratio: VERT_STRETCH_RATIO = [no_of_lines_of_source / no_of_lines_of _destination] * 1024
d	R/W	USE_RATIO0	0 = VERT_STRETCH_RATIO0 /1 /2 /3 will be used if the value of the vertical end register is 350/ 400/ 480 respectively. 1 = Always use VERT_STRETCH_RATIO0
e	R/W	VERT_STRETCH_EN	Vertical stretch enable: 1 = Vertical stretching is enabled

Description

VERT_STRETCHING is used to define parameters for vertical stretching (expansion).

Usage

In order to support ratiometric expansion, HORZ_STRETCHING, VERT_STRETCHING and EXT_VERT_STRETCH registers have to be programmed. The values that have to be programmed in these registers depend on the panel resolution and the current graphics mode resolution. Ratiometric expansion in the vertical direction can be done in two ways — line replication or vertical blending.

See Also

HORZ_STRETCHING, EXT_VERT_STRETCH (next)

EXT_VERT_STRETCH Offset: 0_2A Index: 6

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
LT								f	e	d	c											b	a									

a	R/W	VERT_STRETCH_RATIO3 (9:0) (Reserved for Ratio until version A22 inclusive; shared afterwards)	Vertical stretch ratio (for 600 non standard VGA mode) VERT_STRETCH_RATIO = [no_of_lines_of_source/ no_of_lines_of _destination] * 1024 (From version A31 on, if MONITOR_DET_EN = 0, look at POWER_MANAGEMENT register)
		FORCE_DAC_DATA (7:0) (From Version A31 on, shared with Ratio)	Used for improved monitor detection (if MONITOR_DET_EN = 1) The value written in this regiser will be forced directly to the input of the CRT D/A converter
		FORCE_DAC_DATA_SEL (9:8) (From Version A31 on, shared with Ratio)	Used to select which 24-bit D/A inputs will be forced with FORCE_DAC_DATA value 00 - R = FORCE_DAC_DATA; G = B = 0 01 - G = FORCE_DAC_DATA; R = B = 0 10 - B = FORCE_DAC_DATA; R = G = 0 11 - R = G = B = FORCE_DAC_DATA
b	R/W	VERT_STRETCH_MODE	Vertical stretch mode 0 = line replication 1 = vertical blending
c	R/W	VERT_PANEL_SIZE	Vertical panel size (line number -1)
d	R/W	AUTO_VERT_RATIO	Automatically calculate vertical ratio based on vertical display end, top and bottom overscan 0 = use VERT_STRETCH_RATIOX register values 1 = calculate vertical ratio
e	R/W	USE_AUTO_FP_POS	Enable generation of sync signals for DSTN panel based on VERT_PANEL_SIZE value
f	R/W	USE_AUTO_LCD_VSYNC	Enable generation of sync signal for TFT panel based on shadow sync start and end values

8.1.8 LT_GPIO and ZVGPIO

LT_GIO **Offset: 0_2A Index: 7**

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
LT		t	s	r						q	p	o	n	m	l	k		j	i	h						g	f	e	d	c	b	a

a	R/W	LT_GIO_0	Write/Read (DIR0 = output/input) Pin: LTGIO(0)
b	R/W	LT_GIO_1	Write/Read (DIR1 = output/input) Pin: LTGIO(1)
c	R/W	LT_GIO_2	Write/Read (DIR2 = output/input) Pin: LTGIO(2)
d	R/W	LT_GIO_3	Write/Read (DIR3 = output/input) Pin: LCDDO(20)
e	R/W	LT_GIO_4	Write/Read (DIR4 = output/input) Pin: LCDDO(21)
f	R/W	LT_GIO_5	Write/Read (DIR5 = output/input) Pin: LCDDO(22)
g	R/W	LT_GIO_6	Write/Read (DIR6 = output/input) Pin: LCDDO(23)
h	R/W	GPIO_14	Write/Read (DIR14 = output/input) Pin: GPIO(14)
i	R/W	GPIO_15	Write/Read (DIR15 = output/input) Pin: GPIO(15)
j	R/W	GPIO_16	Write/Read (DIR16 = output/input) Pin: GPIO(16)
k	R/W	LT_GIO_DIR_0	LT_GIO_0 Direction: 0 = Input 1 = Output (Default = 0)
l	R/W	LT_GIO_DIR_1	LT_GIO_1 Direction: 0 = Input 1 = Output (Default = 0)
m	R/W	LT_GIO_DIR_2	LT_GIO_2 Direction: 0 = Input 1 = Output (Default = 0)
n	R/W	LT_GIO_DIR_3	LT_GIO_3 Direction: 0 = Input 1 = Output (Default = 0)
o	R/W	LT_GIO_DIR_4	LT_GIO_4 Direction: 0 = Input 1 = Output (Default = 0)
p	R/W	LT_GIO_DIR_5	LT_GIO_5 Direction: 0 = Input 1 = Output (Default = 0)
q	R/W	LT_GIO_DIR_6	LT_GIO_6 Direction: 0 = Input 1 = Output (Default = 0)
r	R/W	GPIO_DIR_14	GPIO_14 Direction: 0 = Input 1 = Output (Default = 0)
s	R/W	GPIO_DIR_15	GPIO_15 Direction: 0 = Input 1 = Output (Default = 0)
t	R/W	GPIO_DIR_16	GPIO_16 Direction: 0 = Input 1 = Output (Default = 0)

Description

This register specifies the data/direction for the following pins: LTGIO[2:0], LCDDO[23:20] and GPIO[16:14].

Usage

Refer to the RAGE LT PRO Graphics Controller Specification for details on the typical pin configuration used to support various panel types. In some modes, LCDDO[23:20] pins will not be available as general purposed I/Os.

See Also

GP_IO on *page 4-1* .

	ZVGPIO																						Offset: 0_2A Index: 9									
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
LT	q								p	o	n	m	l	k	j	i									h	g	f	e	d	c	b	a

a	R/W	ZVGPIO_0	Write/Read (DIR0 = output/input) Pin: ZVPORT(8)
b	R/W	ZVGPIO_1	Write/Read (DIR1 = output/input) Pin: ZVPORT(9)
c	R/W	ZVGPIO_2	Write/Read (DIR2 = output/input) Pin: ZVPORT(10)
d	R/W	ZVGPIO_3	Write/Read (DIR3 = output/input) Pin: ZVPORT(11)
e	R/W	ZVGPIO_4	Write/Read (DIR4 = output/input) Pin: ZVPORT(12)
f	R/W	ZVGPIO_5	Write/Read (DIR5 = output/input) Pin: ZVPORT(13)
g	R/W	ZVGPIO_6	Write/Read (DIR6 = output/input) Pin: ZVPORT(14)
h	R/W	ZVGPIO_7	Write/Read (DIR7 = output/input) Pin: ZVPORT(15)
	R/W	BL_PWM_LEVEL (7:0) (from A31 version on, shared with ZVGPIO)	Selects one of 256 posible pulse width modulated levels for backlight only if BL_PWM_LEVEL_DIS = 0. When BL_PWM_LEVEL_DIS = 1, this field is used for ZVGPIO as before.
i	R/W	ZVGPIO_DIR_0	ZVGPIO_0 Direction: 0 = Input 1 = Output (Default = 0)
j	R/W	ZVGPIO_DIR_1	ZVGPIO_1 Direction: 0 = Input 1 = Output (Default = 0)
k	R/W	ZVGPIO_DIR_2	ZVGPIO_2 Direction: 0 = Input 1 = Output (Default = 0)
l	R/W	ZVGPIO_DIR_3	ZVGPIO_3 Direction: 0 = Input 1 = Output (Default = 0)
m	R/W	ZVGPIO_DIR_4	ZVGPIO_4 Direction: 0 = Input 1 = Output (Default = 0)
n	R/W	ZVGPIO_DIR_5	ZVGPIO_5 Direction: 0 = Input 1 = Output (Default = 0)
o	R/W	ZVGPIO_DIR_6	ZVGPIO_6 Direction: 0 = Input 1 = Output (Default = 0)
p	R/W	ZVGPIO_DIR_7	ZVGPIO_7 Direction: 0 = Input 1 = Output (Default = 0)

Cont'd

ZVGPIO **Offset: 0_2A Index: 9**

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
LT	q								p	o	n	m	l	k	j	i									h	g	f	e	d	c	b	a

q	R/W	PWRSEQ_DELAY	Programmable value of panel power sequencing block. This value can be programmed up to 2048 ms in increments of 8 ms (generated from 32kHz clock). If 0 is programmed, FP will be used as clock to generate power up/down sequence. Value bigger than 1 has to be programmed.

8.1.9 Power Management

POWER_MANAGEMENT **Offset: 0_2A Index: 8**

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
LT	t		s	r		q	p	o	n				m				l				k	j	i	h	g	f	e	d	c	b		a

a	R/W	PWR_MGT_ON	0 = Disable power management 1 = Enable power management
b	R/W	PWR_MGT_MODE	00 = Pin mode 01 = Register mode 10 = Timer mode 11 = PCI Power Management mode
c	R/W	AUTO_PWRUP_EN	0 = Disable automatic power up sequence 1 = Enable automatic power up sequence
d	R/W	ACTIVITY_PIN_ON	0 = STANDBYb is used as STANDBY pin 1 = STANDBYb is used as ACTIVITY pin
e	R/W	STANDBY_POL	0 = STANDBYb pin is active low 1 = STANDBYb pin is active high
f	R/W	SUSPEND_POL	0 = SUSPENDb pin is active low 1 = SUSPENDb pin is active high
g	R/W	SELF_REFRESH	0 = Enable Self-Refresh 1 = Disable Self-Refresh
h	R/W	ACTIVITY_PIN_EN	0 = Don't allow activity pin to get the chip out of standby / suspend mode 1 = Enable the activity pin to force the chip out of standby / suspend mode
i	R/W	KEYBD_SNOOP	0 = Disable keyboard access snooping 1 = Enable keyboard access to exit the chip from power down modes

Cont'd **POWER_MANAGEMENT** **Offset: 0_2A Index: 8**

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
LT	t		s	r		q	p	o	n				m				l				k	j	i	h	g	f	e	d	c	b		a

j	R/W	USE_F32KHZ	Selects the source of the clock for Standby/Suspend timers, backlight modulation 0 = Use devided reference clock as a source 1 = Use F32KHZ input port as a source
k	R/W	TRISTATE_MEM_EN	o = disable tristating MA, MD and control signals to the memory during Suspend Mode 1 = enable tristating signals in Suspend Mode
l	R/W	LCDENG_TEST_MODE	0000 = Normal mode Others = Set the LCD Engine into test modes (to be defined) 0001 = CRC for LCD datapath 0010 = Framemod module goes into test mode 0011 = LVDS test mode 1000 = TV out test mode 1110 = MONITOR_DEC_EN (used for improved monitor detection; from A31 on) 1111 = Power Management test mode Others = Reserved
m	R/W	STANDBY_COUNT	4-bit Standby counter value
n	R/W	SUSPEND_COUNT	4-bit Suspend counter value
o	R/W	BIASON	Panel bias voltage control: 0 = Shut off bias voltage (Vee) 1 = Turn on bias voltage (Vee)
p	R/W	BLON	Backlight control: 0 = Shut off backlight voltage (Vb) 1 = Turn on backlight voltage (Vb)
q	R/W	DIGON	Panel digital power control: 0 = Shut off digital voltage (Vcc) 1 = Turn on digital voltage (Vcc)
r	R/W	STANDBY_NOW	1 = Force the chip to go into standby mode unconditionally
s	R/W	SUSPEND_NOW	1 = Force the chip to go into suspend mode unconditionally
t	R	PWR_MGT_STATUS	Power management status (Read only) 00 = On mode 01 = Standby mode 10 = Suspend mode 11 = Transition between Power Management modes

Description

This register is used to control Power Management features implemented in 3D RAGE LT PRO.

Usage

Three different power saving modes are supported: *On*, *Standby* and *Suspend*, with the power consumption decreasing from *On* to *Suspend* mode.

This page intentionally left blank.

Index