RAGE™ Mobility

Register Reference Guide (OEM Version)

Technical Reference Manual

P/N: RRG-G04200-C Rev 1.00

The information contained in this manual 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
0.01	Nov. 1998
0.02	Apr. 1999
1.00	Jul. 1999
Refer to Appendix A for revision history.	

Related Manuals

RAGE Mobility series

- RAGE™ Mobility Register Reference Guide (RRG-G04200)
- RAGE™ Mobility Graphics Controller Specifications (GCS-C04200)
- RAGE™ Mobility Design Guide (DRS-D04200)

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 Registers

Appendix A: Revision History

Chapter 1
Introduction

1.1 Scope

This document is a reference guide for the registers used in the RAGE Mobility product.

1.2 Summary of the Contents

The following table summaries the chapters used in this document.

Table 1-1 A Summary of the Chapters

Chapter	Description or Contents
Chapter 1	An outline of the manual. Explains notations and conventions.
Chapter 2	An overview of the registers. Describes the memory and I/O mapping.
Chapter 3	A cross-reference summary of the registers. Two tables list the registers by register name (mnemonic) and by register address. To help quickly locate a specific register, both tables use a page number column. This column is hypertext linked for the on-line document.
Chapter 4	2D accelerator registers Setup and control registers Accelerator CRTC and DAC registers
Chapter 5	GUI engine registers
Chapter 6	Host interface registers PCI configuration space registers System bus mastering registers Draw engine bus mastering registers AGP registers
Chapter 7	VGA-compatible registers
Chapter 8	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:

Table 1-2 Explanation of the Mnemonic Nomenclature

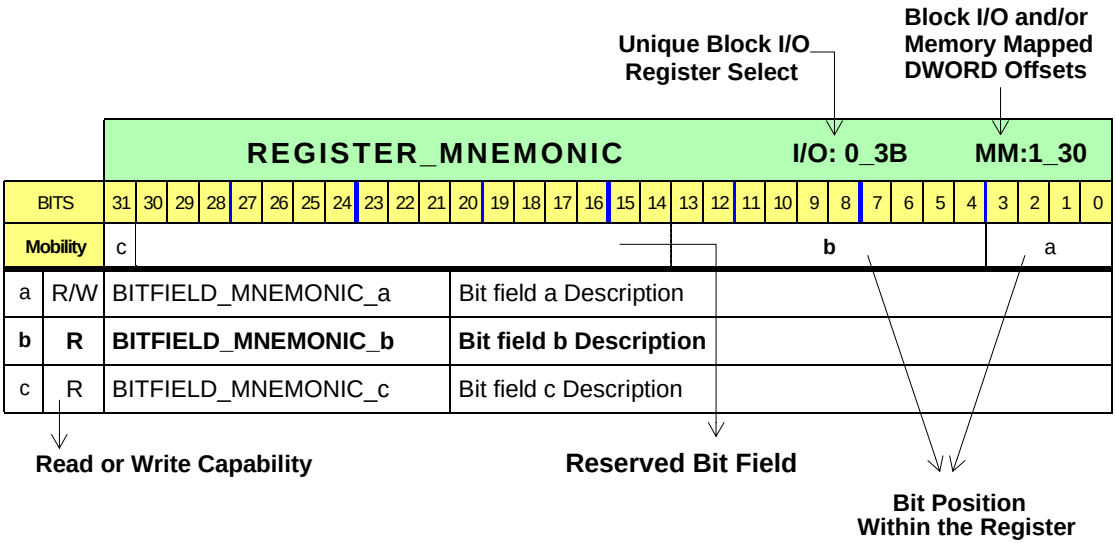
Mnemonic Nomenclature	Example and Explanation
REGISTER_MNEMONIC	CONFIG_CHIP_ID The register mnemonic for the Configuration Chip ID register.
REGISTER_MNEMONIC[bit#:bit#]	CONFIG_CHIP_ID[15:0] [15:0] is the bit field that occupies bit positions 0 through 15 within the CONFIG_CHIP_ID register.
FIELD_NAME@REGISTER_MNEMONIC	CFG_CHIP_TYPE@CONFIG_CHIP_ID An alternative to the above mnemonic (CONFIG_CHIP_ID[15:0]). Instead of providing the bit positions, this example gives the field name CFG_CHIP_TYPE (Product Type Code).

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.



NOTES:

A **bolded** bit descriptions means that this description is specific to the Mobility chip. This bolded description does not apply to any other ATI accelerator graphics chip.

Mobility means this register is specific to the Mobility chip. Therefore, all of the bit descriptions for a Mobility-specific register will be **bolded**. A Mobility specific register is not used by any other ATI accelerator graphics chip.

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 the following table below. Less frequently used or ATI-specific acronyms will be accompanied by the full expression when it appears for the first time in the document.

Table 1-3 Acronyms

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 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 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 (part of the *Bus Mastering Registers*)
- *Bus Mastering Registers*
- *AGP Registers*
- *LCD Panel Registers*
- *PCI Configuration Space Registers*
- *VGA Registers*

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. To view the contents of the PCI and VGA registers, refer to Chapters 6 and 7 respectively.

The following is an overview of all the registers. Some classes are further divided into sub-groups.

2.1.1 Setup and Control Registers

To view these registers, refer to *Chapter 4*. These registers are memory mapped and aliased at an I/O address. Most of these registers are initialized only once at boot time. They are divided into the following sub-groups:

Table 2-1 Sub-groups for the Setup and Control Register

Sub-group	Purpose
General I/O Control	Configuring the General Purpose I/O pins on the accelerator chip.
Scratch Pad	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	Configuring the on-chip bus interface unit.
Memory	Configuring the memory interfaces.
Test and Debug	Chip diagnostics and hardware debugging.
Configuration	Configuring the aperture and to read the current board configuration.
Custom Macros	Setting up custom macros.

2.1.2 Accelerator CRTC and DAC Registers

To view these registers, refer to *Chapter 4*. These registers are memory mapped and aliased at an I/O address. The accelerator CRTC registers are not the same as the VGA CRTC registers. They are divided into the following sub-groups:

Table 2-2 Sub-groups for the Accelerator CRTC and DAC Registers

Sub-group	Purpose
Second CRTC	Defining the secondary display parameters.
Overscan	Configuring the overscan borders.
Hardware Cursor	Defining and moving the hardware cursor.
Genlocking	Synchronizing the CRT with the video source.
Clock Control and PLL	Configuring the pixel clock.
DAC Control	Configuring the DAC.

2.1.3 Draw Engine Trajectory Registers

To view these registers, refer to *Chapter 5*. These registers are memory mapped. They set up the source and destination trajectories and initiate draw operations. They are divided into the following sub-groups:

Table 2-3 Sub-groups for the Draw Engine Trajectory Registers

Sub-group	Purpose
Destination Trajectory	Defining the region in which pixels are drawn. The region may be a a line, a rectangular, or a trapezoidal area.
Source Trajectory	Defining 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.

2.1.4 Draw Engine Control Registers

To view these registers, refer to *Chapter 5*. These 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 sub-groups:

Table 2-4 Sub-groups for the Draw Engine Control Registers

Sub-group	Purpose
Host Data	Transferring data from the host to the draw engine.
Pattern	Enabling and defining fixed patterns.
Scissor	Defining a draw region.
Data Path	Configuring the data path and ALU.
Color Compare	Configuring the source or destination color compare.
Command FIFO Status	Reporting the status of the command FIFO.
Draw Engine Composite Control	Abbreviated composites of other draw engine control registers.
Draw Engine Status	Reporting the current state of the draw engine.

2.1.5 Scaler and 3D Accelerator Registers (Omitted)

2.1.6 Multimedia Registers (Omitted)

2.1.7 Bus Mastering Registers

To view these registers, refer to *Chapter 6*. The RAGE Mobility 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.

Table 2-5 Sub-groups for the Bus Mastering Registers

Sub-group	Purpose
Draw Engine Bus Mastering	Specifying register address/data for bus mastering operations invoked through the Draw Engine.
System Bus Mastering	Controlling and determining the status of all the bus mastering operations.

2.1.8 AGP Registers

To view these registers, refer to *Chapter 6*. These registers configure the Accelerated Graphic Port.

2.1.9 LCD Panel Registers

These registers configure the LCD panel. To view these registers, refer to *Chapter 8*.

2.1.10 TV Out Support Registers (Omitted)

2.1.11 PCI Configuration Space Registers

To view these registers, refer to *Chapter 6*. The PCI Configuration Space registers determine the host bus configuration during system reset. For the RAGE Mobility, 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.

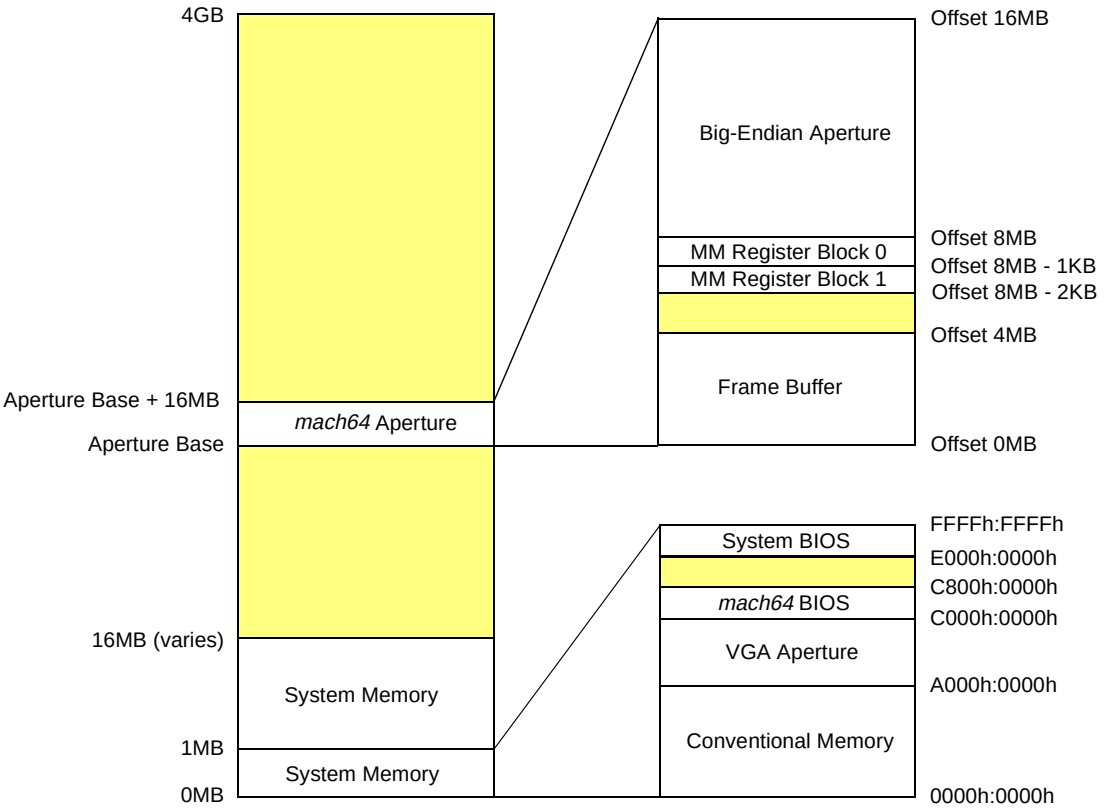
2.1.12 VGA Registers

To view these registers, refer to *Chapter 7*. 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 Mobility.

2.2 Memory Mapping

The RAGE Mobility 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’, as shown in the figure below.



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

2.2.1 Mapping Model

In terms of memory mapping locations, the registers can be grouped into six major categories. The table below summarizes the mapping of the register groups to memory and to the I/O space.

Table 2-6 Register Groups by Memory Mapping

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

- 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, can be directly readable and writable. In addition to being memory mapped, they are also mapped into a single continuous I/O block (referred to as “block” or “re-locatable” 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 Mobility.

2.2.2 Accessing Bytes, Words, and Dwords

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

Table 2-7 Register Accessing Sizes

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.

2.2.3 Non-Intel Based Memory Mapping

When incorporating the RAGE Mobility 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 Mobility in non-Intel environments, refer to Chapter 2 of the *mach64 Programmer's Guide*.

2.3 Mapping Modes

Depending on the system configuration, the RAGE Mobility 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.

2.3.1 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 auxiliary 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.

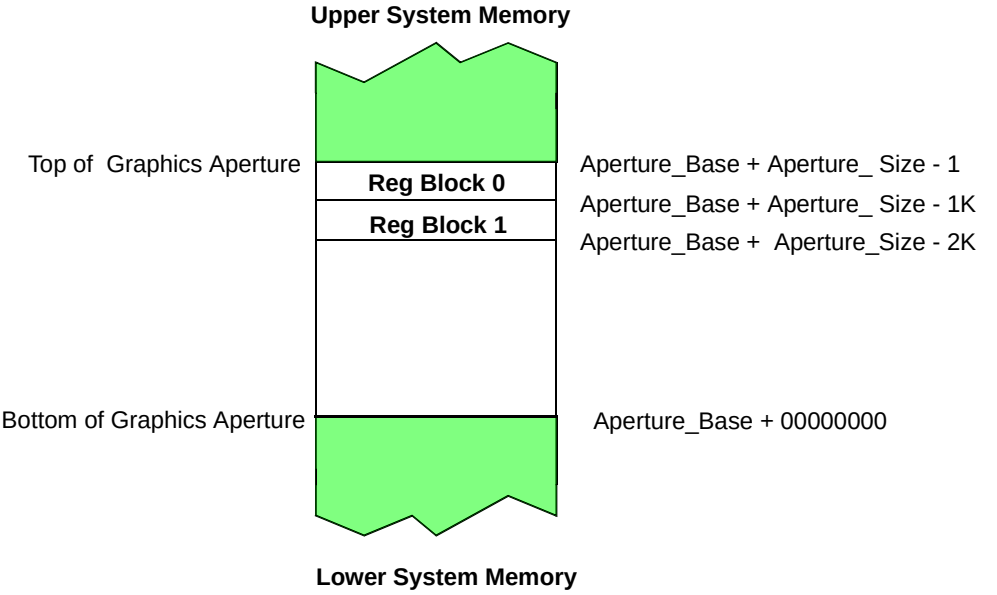


Figure 2-2 Primary Linear Aperture Register Map

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 2-8 Register Block Offsets

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

Auxiliary Aperture

The auxiliary 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 auxiliary 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 auxiliary aperture is shown in the figure below:

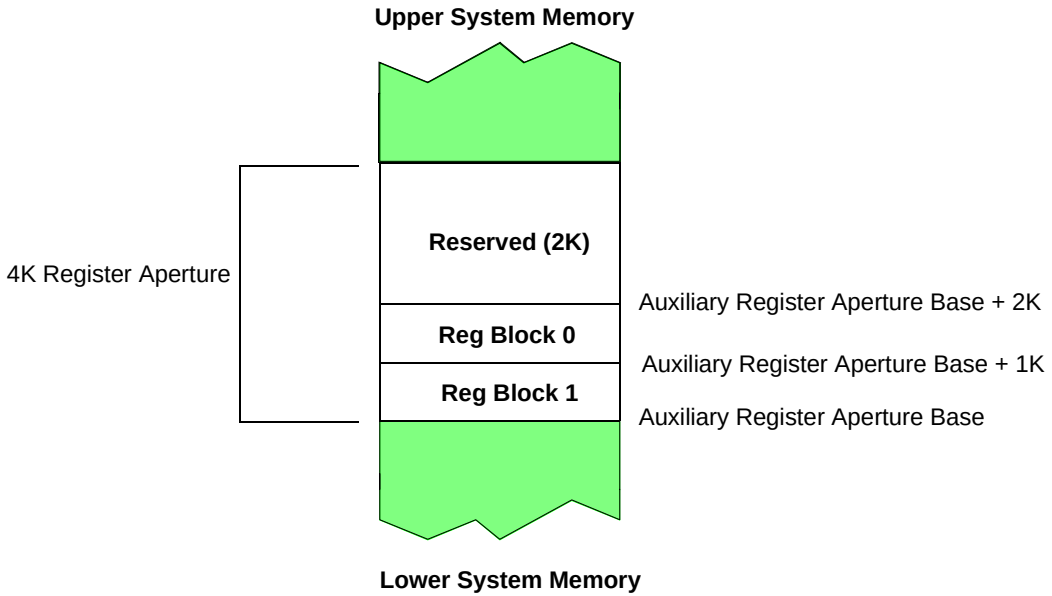


Figure 2-3 Auxiliary Register Aperture Memory Map

Note: 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.

2.3.2 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 or configured.

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.

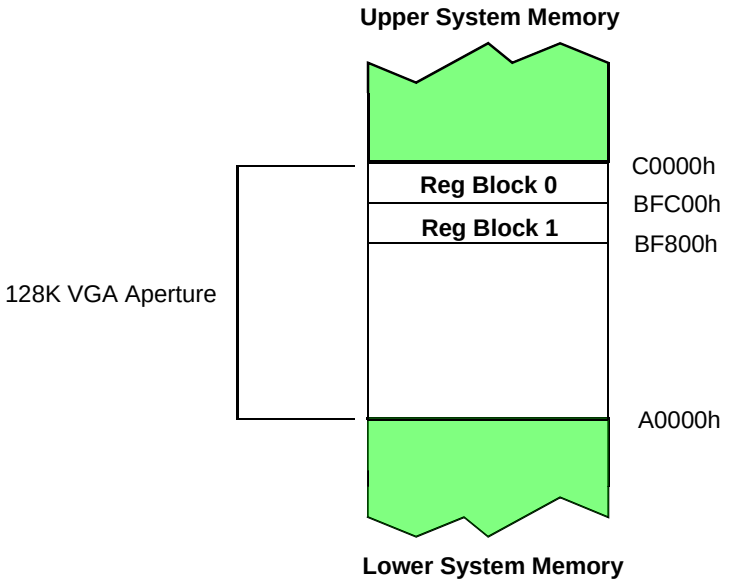


Figure 2-4 VGA Aperture Memory Map

The following table indicates the various bits that must be set to enable register mapping

into the VGA aperture.

Table 2-9 Mapping into the VGA Aperture

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

2.4 Determining Mapped Addresses

2.4.1 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 hexadecimal

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** proceeded by a block number are assumed to be in Block ‘0’.

2.4.2 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 Mobility 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.

2.4.3 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

Chapter 3
Cross Reference Tables

3.1 Using the Tables

This chapter comprises two main register summary tables, followed by 4 sub-tables. The two main tables list all but the PCI POS and the VGA Controller registers, which are covered in Chapters 6 and 7 respectively. The first table lists the registers by address while the second by mnemonic. The 4 sub-tables list the indexed PLL and LCD registers, with each group also sorted in two ways.

The tables in this chapter offer a convenient way to locate the full description of any of the registers contained in this manual. If you are using the online version of this document, the page numbers are hypertext linked to the register descriptions.

Table 3-1 A Summary of the Cross Reference of the Register Listings

3.1.1 Table Notations:

- For those familiar with the RAGE LT PRO, registers common to both the RAGE LT PRO and RAGE Mobility are in normal print, those that have been modified for the RAGE Mobility are in **bold**, and those that are specific to the RAGE Mobility are marked with ***Mob 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.
- *W* means write only.

3.2 Listings of Main Registers

3.2.1 Registers listed by Offset Address

Table 3-2 Registers by Offset Address

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	*4-46*
	CRTC2_H_TOTAL_DISP	R/W	✓	0_00	*4-47*
	CRTC_H_SYNC_STRT_WID	R/W	✓	0_01	*4-47*
	CRTC2_H_SYNC_STRT_WID	R/W	✓	0_01	*4-48*
	CRTC_V_TOTAL_DISP	R/W	✓	0_02	*4-49*
	CRTC2_V_TOTAL_DISP	R/W	✓	0_02	*4-49*
	CRTC_V_SYNC_STRT_WID	R/W	✓	0_03	*4-50*
	CRTC2_V_SYNC_STRT_WID	R/W	✓	0_03	*4-50*
	CRTC_VLINE_CRNT_VLINE	R/W	✓	0_04	*4-51*
	CRTC2_VLINE_CRNT_VLINE	R/W	✓	0_04	*4-51*
	CRTC_OFF_PITCH	R/W	✓	0_05	*4-52*
	CRTC_INT_CNTL	R/W	✓	0_06	*4-53*
	CRTC_GEN_CNTL	R/W	✓	0_07	*4-56*
Memory Buffer Control	DSP_CONFIG	R/W	✓	0_08	*4-10*
	PM_DSP_CONFIG (Mobility Only)	**R/W**	**✓**	**0_08**	***4-13***
	DSP_ON_OFF	R/W	✓	0_09	*4-10*
	PM_DSP_ON_OFF (Mobility Only)	**R/W**	**✓**	**0_09**	***4-13***
	TIMER_CONFIG	**R/W**	**✓**	**0_0A**	***4-11***
	MEM_BUF_CNTL	R/W	✓	0_0B	*4-15*
	MEM_ADDR_CONFIG	**R/W**	**✓**	**0_0D**	***4-16***
Accelerator CRTC	CRT_TRAP	R/W	✓	0_0E	*4-60*

Table 3-2 Registers by Offset Address **Cont'd**

Registers by Address					
Register Class	**Mnemonic**	**Read/ Write**	**Block I/O**	**DWORD Offset (h)**	**Page**
Overscan	OVR_CLR	R/W	✓	0_10	*4-61*
	OVR2_CLR	R/W	✓	0_10	*4-61*
	OVR_WID_LEFT_RIGHT	R/W	✓	0_11	*4-62*
	OVR2_WID_LEFT_RIGHT	R/W	✓	0_11	*4-63*
	OVR_WID_TOP_BOTTOM	R/W	✓	0_12	*4-63*
	OVR2_WID_TOP_BOTTOM	R/W	✓	0_12	*4-64*
Memory Buffer Control	VGA_DSP_CONFIG	R/W	✓	0_13	*4-16*
	PM_VGA_DSP_CONFIG (Mobility Only)	**R/W**	**✓**	**0_13**	***4-14***
	VGA_DSP_ON_OFF	R/W	✓	0_14	*4-16*
	PM_VGA_DSP_ON_OFF (Mobility Only)	**R/W**	**✓**	**0_14**	***4-14***
	DSP2_CONFIG	R/W	✓	0_15	*4-10*
	PM_DSP2_CONFIG (Mobility Only)	**R/W**	**✓**	**0_15**	***4-14***
	DSP2_ON_OFF	R/W	✓	0_16	*4-11*
	PM_DSP2_ON_OFF (Mobility Only)	**R/W**	**✓**	**0_16**	***4-15***
Accelerator CRTC	**CRTC2_OFF_PITCH**	**R/W**	**✓**	**0_17**	***4-53***
Hardware Cursor	CUR_CLR0	R/W	✓	0_18	*4-65*
	CUR2_CLR0	**R/W**	**✓**	**0_18**	***4-68***
	CUR_CLR1	R/W	✓	0_19	*4-66*
	CUR2_CLR1	**R/W**	**✓**	**0_19**	***4-68***
	CUR_OFFSET	R/W	✓	0_1A	*4-66*
	CUR2_OFFSET	**R/W**	**✓**	**0_1A**	***4-69***
	CUR_HORZ_VERT_POSN	R/W	✓	0_1B	*4-66*
	CUR2_HORZ_VERT_POSN	**R/W**	**✓**	**0_1B**	***4-69***
	CUR_HORZ_VERT_OFF	R/W	✓	0_1C	*4-67*
	CUR2_HORZ_VERT_OFF	**R/W**	**✓**	**0_1C**	***4-69***
General I/O Control	**GP_IO**	**R/W**	**✓**	**0_1E**	***4-2***
Test and Debug	HW_DEBUG	R/W	✓	0_1F	*4-30*

Table 3-2 Registers by Offset Address **Cont'd**

Registers by Address					
Register Class	**Mnemonic**	**Read/ Write**	**Block I/O**	**DWORD Offset (h)**	**Page**
Scratch Pad and Test	SCRATCH_REG0	R/W	✓	0_20	*4-4*
	SCRATCH_REG1	R/W	✓	0_21	*4-5*
	SCRATCH_REG2	R/W	✓	0_22	*4-5*
	SCRATCH_REG3	R/W	✓	0_23	*4-6*
Clock Control	CLOCK_CNTL	R/W	✓	0_24	*4-73*
Configuration	**CONFIG_STAT1**	**R**	**✓**	**0_25**	***4-38***
	CONFIG_STAT2	**R**	**✓**	**0_26**	***4-38***
Bus Control	BUS_CNTL	R/W	✓	0_28	*4-6*
For related registers see Table 3-3	LCD_INDEX	R/W	✓	0_29	*8-2*
	LCD_DATA	R/W	✓	0_2A	*8-3*
Memory Control	**EXT_MEM_CNTL**	**R/W**	**✓**	**0_2B**	***4-18***
	MEM_CNTL	**R/W**	**✓**	**0_2C**	***4-22***
	MEM_VGA_WP_SEL	R/W	✓	0_2D	*4-24*
	MEM_VGA_RP_SEL	R/W	✓	0_2E	*4-25*
DAC Control	DAC_REGS	R/W	✓	0_30	*4-99*
	DAC_CNTL	**R/W**	**✓**	**0_31**	***4-100***
Test and Debug	**GEN_TEST_CNTL**	**R/W**	**✓**	**0_34**	***4-27***
Custom Macros	CUSTOM_MACRO_CNTL	R/W	✓	0_35	*4-39*
Configuration	CONFIG_CNTL	R/W	✓	0_37	*4-34*
	CONFIG_CHIP_ID	**R**	**✓**	**0_38**	***4-35***
	CONFIG_STAT0	**R/W**	**✓**	**0_39**	***4-37***
Test and Debug	CRC_SIG	R	✓	0_3A	*4-30*
	CRC2_SIG	R	✓	0_3A	*4-30*

Table 3-2 Registers by Offset Address **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	*5-10*
	DST_X	R/W	-	0_41	*5-12*
	DST_Y	R/W	-	0_42	*5-14*
	DST_Y_X (aliased at 0_4Dh, 3D)	W	-	0_43	*5-15*
	DST_WIDTH	R/W	-	0_44	*5-11*
	DST_HEIGHT	R/W	-	0_45	*5-9*
	DST_HEIGHT_WIDTH	W	-	0_46	*5-9*
	DST_X_WIDTH	W	-	0_47	*5-13*
	DST_BRES_LNTH (LEAD_BRES_LNTH) (aliased at 0_51h, 3D)	R/W	-	0_48	*5-4*
	DST_BRES_ERR	R/W	-	0_49	*5-3*
	DST_BRES_INC (LEAD_BRES_INC, 3D)	R/W	-	0_4A	*5-3*
	DST_BRES_DEC (LEAD_BRES_DEC, 3D)	R/W	-	0_4B	*5-2*
	DST_CNTL	R/W	-	0_4C	*5-6*
	DST_Y_X (aliased at 0_43h, 3D)	W	-	0_4D	*5-15*
	TRAIL_BRES_ERR	R/W	-	0_4E	*5-16*
	TRAIL_BRES_INC	R/W	-	0_4F	*5-16*
	TRAIL_BRES_DEC	R/W	-	0_50	*5-16*
	LEAD_BRES_LNTH (aliased at 0_48h, 3D)	R/W	-	0_51	*5-4*
	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 3-2 Registers by Offset Address **Cont'd**

Registers by Address					
Register Class	**Mnemonic**	**Read/ Write**	**Block I/O**	**DWORD Offset (h)**	**Page**
Draw Engine Source Trajectory	SRC_OFF_PITCH	R/W	-	0_60	*5-26*
	SRC_X	R/W	-	0_61	*5-29*
	SRC_Y	R/W	-	0_62	*5-30*
	SRC_Y_X	W	-	0_63	*5-31*
	SRC_WIDTH1	R/W	-	0_64	*5-27*
	SRC_HEIGHT1	R/W	-	0_65	*5-24*
	SRC_HEIGHT1_WIDTH1	W	-	0_66	*5-25*
	SRC_X_START	R/W	-	0_67	*5-29*
	SRC_Y_START	R/W	-	0_68	*5-31*
	SRC_Y_X_START	W	-	0_69	*5-32*
	SRC_WIDTH2	R/W	-	0_6A	*5-28*
	SRC_HEIGHT2	R/W	-	0_6B	*5-25*
	SRC_HEIGHT2_WIDTH2	W	-	0_6C	*5-26*
	SRC_CNTL	R/W	-	0_6D	*5-21*
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	*6-13*
	BM_ADDR	**W**	**-**	**0_92**	***6-14***
	BM_DATA	W	-	0_92	*6-14*
	BM_GUI_TABLE_CMD	W	-	0_93	*6-16*
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*

Table 3-2 Registers by Offset Address **Cont'd**

Registers by Address					
Register Class	**Mnemonic**	**Read/ Write**	**Block I/O**	**DWORD Offset (h)**	**Page**
Data Path	USR1_DST_0FF_PITCH	W	-	0_AE	*5-52*
	USR2_DST_0FF_PITCH	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	*5-14*
	DST_WIDTH_HEIGHT	W	-	0_BB	*5-12*
Data Path	USR_DST_PITCH	W	-	0_BC	*5-52*
	DP_SET_GUI_ENGINE2	W	-	0_BE	*5-57*
	DP_SET_GUI_ENGINE	**W**	**-**	**0_BF**	***5-53***
Color Compare	CLR_CMP_CLR	R/W	-	0_C0	*5-60*
	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	*5-65*
Engine Status/FIFO	**GUI_STAT**	**R**	**-**	**0_CE**	***5-67***
Draw Engine Destination Trajectory	COMPOSITE_SHADOW_ID	R/W	-	0_E6	*5-21*
GenLocking	SNAPSHOT_VH_COUNTS	R	-	1_1C	*4-70*
	SNAPSHOT_F_COUNT	R	-	1_1D	*4-70*
	N_VIF_COUNT	R/W	-	1_1E	*4-71*
	SNAPSHOT_VIF_COUNT	R/W	-	1_1F	*4-71*

Table 3-2 Registers by Offset Address **Cont'd**

Registers by Address					
Register Class	**Mnemonic**	**Read/ Write**	**Block I/O**	**DWORD Offset (h)**	**Page**
GenLocking	**SNAPSHOT2_VH_COUNTS**	**R**	**-**	**1_2C**	***4-70***
	SNAPSHOT2_F_COUNT	**R**	**-**	**1_2D**	***4-70***
	N_VIF2_COUNT	**R/W**	**-**	**1_2E**	***4-71***
	SNAPSHOT2_VIF_COUNT	**R/W**	**-**	**1_2F**	***4-72***
Test and Debug	CRT_HORZ_VERT_LOAD	R/W	-	1_51	*4-33*
AGP	AGP_BASE	R/W	-	1_52	*6-16*
	AGP_CNTL	R/W	-	1_53	*6-17*
Command FIFO	**GUI_CMDFIFO_DEBUG**	**R/W**	**-**	**1_5C**	***5-63***
	GUI_CMDFIFO_DATA	R/W	-	1_5D	*5-64*
	GUI_CNTL	**R/W**	**-**	**1_5E**	***5-64***
Bus Mastering	BM_FRAME_BUF_OFFSET	R	-	1_60	*6-11*
	BM_SYSTEM_MEM_ADDR	R	-	1_61	*6-12*
	BM_COMMAND	R	-	1_62	*6-12*
	BM_STATUS	R	-	1_63	*6-12*
	BM_GUI_TABLE	R/W	-	1_6E	*6-15*
	BM_SYSTEM_TABLE	R/W	-	1_6F	*6-13*
Setup Engine	VERTEX_1_SECONDARY_T	R/W	-	1_CB	*6-43*
	VERTEX_1_SECONDARY_W	R/W	-	1_CC	*6-43*
	VERTEX_2_SECONDARY_S	R/W	-	1_CD	*6-44*
	VERTEX_2_SECONDARY_T	R/W	-	1_CE	*6-44*
	VERTEX_2_SECONDARY_W	R/W	-	1_CF	*6-45*

3.2.2 Registers listed by Mnemonic

Table 3-3 Registers Listed by Mnemonic

Registers by Mnemonic					
Register Class	**Mnemonic**	**Read/ Write**	**Block I/O**	**DWORD Offset (h)**	**Page**
AGP	AGP_BASE	R/W	-	1_52	*6-16*
	AGP_CNTL	R/W	-	1_53	*6-17*
Draw Engine Destination Trajectory	**ALPHA_TST_CNTL**	**R/W**	**-**	**0_54**	***5-18***
GUI Bus Mastering	**BM_ADDR**	**W**	**-**	**0_92**	***6-14***
Bus Mastering	BM_COMMAND	R	-	1_62	*6-12*
GUI Bus Mastering	BM_DATA	W	-	0_92	*6-14*
Bus Mastering	BM_FRAME_BUF_OFFSET	R	-	1_60	*6-11*
	BM_GUI_TABLE	R/W	-	1_6E	*6-15*
GUI Bus Mastering	BM_GUI_TABLE_CMD	W	-	0_93	*6-16*
	BM_HOSTDATA	W	-	0_91	*6-13*
Bus Mastering	BM_STATUS	R	-	1_63	*6-12*
	BM_SYSTEM_MEM_ADDR	R	-	1_61	*6-12*
	BM_SYSTEM_TABLE	R/W	-	1_6F	*6-13*
Bus Control	BUS_CNTL	R/W	✓	0_28	*4-6*
Clock Control	CLOCK_CNTL	R/W	✓	0_24	*4-73*
Color Compare	CLR_CMP_CLR	R/W	-	0_C0	*5-60*
	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	*5-21*
Configuration	**CONFIG_CHIP_ID**	**R**	**✓**	**0_38**	***4-35***
	CONFIG_CNTL	R/W	✓	0_37	*4-34*
	CONFIG_STAT0	**R/W**	**✓**	**0_39**	***4-37***
	CONFIG_STAT1	**R**	**✓**	**0_25**	***4-38***
	CONFIG_STAT2	**R**	**✓**	**0_26**	***4-38***

Table 3-3 Registers Listed by Mnemonic Cont'd

Registers by Mnemonic					
Register Class	**Mnemonic**	**Read/ Write**	**Block I/O**	**DWORD Offset (h)**	**Page**
Test and Debug	CRC_SIG	R	✓	0_3A	*4-30*
	CRC2_SIG	R	✓	0_3A	*4-30*
	CRT_HORZ_VERT_LOAD	R/W	-	1_51	*4-33*
Accelerator CRTC	CRT_TRAP	R/W	✓	0_0E	*4-60*
	CRTC2_H_SYNC_STRT_WID	**R/W**	✓	0_01	*4-48*
	CRTC2_H_TOTAL_DISP	**R/W**	✓	**0_00**	*4-47*
	CRTC2_OFF_PITCH	**R/W**	✓	**0_17**	***4-53***
	CRTC2_V_SYNC_STRT_WID	**R/W**	✓	**0_03**	*4-50*
	CRTC2_V_TOTAL_DISP	R/W	✓	0_02	*4-49*
	CRTC2_VLINE_CRNT_VLINE	**R/W**	✓	**0_04**	*4-51*
	CRTC_GEN_CNTL	**R/W**	✓	**0_07**	***4-56***
	CRTC_H_SYNC_STRT_WID	R/W	✓	0_01	*4-47*
	CRTC_H_TOTAL_DISP	R/W	✓	0_00	*4-46*
	CRTC_INT_CNTL	**R/W**	✓	**0_06**	*4-53*
	CRTC_OFF_PITCH	**R/W**	✓	**0_05**	***4-52***
	CRTC_V_SYNC_STRT_WID	R/W	✓	0_03	*4-50*
	CRTC_V_TOTAL_DISP	R/W	✓	0_02	*4-49*
	CRTC_VLINE_CRNT_VLINE	R/W	✓	0_04	*4-51*
Hardware Cursor	**CUR2_CLR0**	**R/W**	✓	**0_18**	***4-68***
	CUR2_CLR1	**R/W**	✓	**0_19**	***4-68***
	CUR2_HORZ_VERT_OFF	**R/W**	✓	**0_1C**	***4-69***
	CUR2_HORZ_VERT_POSN	**R/W**	✓	**0_1B**	***4-69***
	CUR2_OFFSET	**R/W**	✓	**0_1A**	***4-69***
	CUR_CLR0	R/W	✓	0_18	*4-65*
	CUR_CLR1	R/W	✓	0_19	*4-66*
	CUR_HORZ_VERT_OFF	R/W	✓	0_1C	*4-67*
	CUR_HORZ_VERT_POSN	R/W	✓	0_1B	*4-66*
	CUR_OFFSET	R/W	✓	0_1A	*4-66*
Custom Macros	CUSTOM_MACRO_CNTL	R/W	✓	0_35	*4-39*

Table 3-3 Registers Listed by Mnemonic Cont'd

Registers by Mnemonic					
Register Class	**Mnemonic**	**Read/ Write**	**Block I/O**	**DWORD Offset (h)**	**Page**
DAC Control	**DAC_CNTL**	**R/W**	✓	**0_31**	***4-100***
	DAC_REGS	R/W	✓	0_30	*4-99*
Data Path	DP_BKGD_CLR	R/W	-	0_B0	*5-43*
	DP_FOG_CLR (DP_FRGD_CLR)	R/W	-	0_B1	*5-43*
	DP_FRGD_BKGD_CLR	W	-	0_B8	*5-44*
	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	*5-57*
	DP_SRC	R/W	-	0_B6	*5-58*
	DP_WRITE_MSK	R/W	-	0_B2	*5-45*
Memory Buffer Control	DSP2_CONFIG	R/W	✓	0_15	*4-10*
	DSP2_ON_OFF	R/W	✓	0_16	*4-11*
	DSP_CONFIG	R/W	✓	0_08	*4-10*
	DSP_ON_OFF	R/W	✓	0_09	*4-10*

Table 3-3 Registers Listed by Mnemonic 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, 3D)	R/W	-	0_4B	*5-2*
	DST_BRES_ERR	R/W	-	0_49	*5-3*
	DST_BRES_INC (LEAD_BRES_INC, 3D)	R/W	-	0_4A	*5-3*
	DST_BRES_LNTH (LEAD_BRES_LNTH) (aliased at 0_51h, 3D)	R/W	-	0_48	*5-4*
	DST_CNTL	R/W	-	0_4C	*5-6*
	DST_HEIGHT	R/W	-	0_45	*5-9*
	DST_HEIGHT_WIDTH	W	-	0_46	*5-9*
	DST_OFF_PITCH	R/W	-	0_40	*5-10*
	DST_WIDTH	R/W	-	0_44	*5-11*
	DST_WIDTH_HEIGHT	W	-	0_BB	*5-12*
	DST_X	R/W	-	0_41	*5-12*
	DST_X_WIDTH	W	-	0_47	*5-13*
	DST_X_Y	W	-	0_BA	*5-14*
	DST_Y	R/W	-	0_42	*5-14*
	DST_Y_X (aliased at 0_43h, 3D)	W	-	0_4D	*5-15*
	DST_Y_X (aliased at 0_4Dh, 3D)	W	-	0_43	*5-15*
Memory Control	**EXT_MEM_CNTL**	**R/W**	✓	**0_2B**	***4-18***
Command FIFO	FIFO_STAT	R	-	0_C4	*5-62*
Test and Debug	**GEN_TEST_CNTL**	**R/W**	✓	**0_34**	***4-27***
General I/O Control	**GP_IO**	**R/W**	✓	**0_1E**	***4-2***
Command FIFO	GUI_CMDFIFO_DATA	R/W	-	1_5D	*5-64*
	GUI_CMDFIFO_DEBUG	**R/W**	**-**	**1_5C**	***5-63***
	GUI_CNTL	**R/W**	**-**	**1_5E**	***5-64***
Engine Status/FIFO	**GUI_STAT**	**R**	**-**	**0_CE**	***5-67***
Engine Control	GUI_TRAJ_CNTL	R/W	-	0_CC	*5-65*
Host Data	HOST_CNTL	R/W	-	0_90	*5-34*
	HOST_DATA[15:0]	W	-	0_80-8F	*5-33*

Table 3-3 Registers Listed by Mnemonic Cont'd

Registers by Mnemonic					
Register Class	**Mnemonic**	**Read/ Write**	**Block I/O**	**DWORD Offset (h)**	**Page**
Test and Debug	HW_DEBUG	R/W	✓	0_1F	*4-30*
For related registers see Table 3-3	LCD_DATA	R/W	✓	0_2A	*8-3*
	LCD_INDEX	R/W	✓	0_29	*8-2*
Draw Engine Destination Trajectory	LEAD_BRES_LNTH (aliased at 0_48h, 3D)	R/W	-	0_51	*5-4*
Memory Buffer Control	**MEM_ADDR_CONFIG**	**R/W**	✓	**0_0D**	***4-16***
	MEM_BUF_CNTL	R/W	✓	0_0B	*4-15*
Memory Control	**MEM_CNTL**	**R/W**	✓	**0_2C**	***4-22***
	MEM_VGA_RP_SEL	R/W	✓	0_2E	*4-25*
	MEM_VGA_WP_SEL	R/W	✓	0_2D	*4-24*
GenLocking	N_VIF2_COUNT	R/W	-	1_2E	*4-71*
	N_VIF_COUNT	R/W	-	1_1E	*4-71*
Overscan	OVR2_CLR	R/W	✓	0_10	*4-61*
	OVR2_WID_LEFT_RIGHT	R/W	✓	0_11	*4-63*
	OVR2_WID_TOP_BOTTOM	R/W	✓	0_12	*4-64*
	OVR_CLR	R/W	✓	0_10	*4-61*
	OVR_WID_LEFT_RIGHT	R/W	✓	0_11	*4-62*
	OVR_WID_TOP_BOTTOM	R/W	✓	0_12	*4-63*
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*
Memory Buffer Control	**PM_DSP2_CONFIG (Mobility Only)**	**R/W**	✓	**0_15**	***4-14***
	PM_DSP2_ON_OFF (Mobility Only)	**R/W**	✓	**0_16**	***4-15***
	PM_DSP_CONFIG (Mobility Only)	**R/W**	✓	**0_08**	***4-13***
	PM_DSP_ON_OFF (Mobility Only)	**R/W**	✓	**0_09**	***4-13***
	PM_VGA_DSP_CONFIG (Mobility Only)	**R/W**	✓	**0_13**	***4-14***
	PM_VGA_DSP_ON_OFF (Mobility Only)	**R/W**	✓	**0_14**	***4-14***

Table 3-3 Registers Listed by Mnemonic 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	*4-4*
	SCRATCH_REG1	R/W	✓	0_21	*4-5*
	SCRATCH_REG2	R/W	✓	0_22	*4-5*
	SCRATCH_REG3	R/W	✓	0_23	*4-6*
GenLocking	SNAPSHOT2_F_COUNT	R	-	1_2D	*4-70*
	SNAPSHOT2_VH_COUNTS	R	-	1_2C	*4-70*
	SNAPSHOT2_VIF_COUNT	R/W	-	1_2F	*4-72*
	SNAPSHOT_F_COUNT	R	-	1_1D	*4-70*
	SNAPSHOT_VH_COUNTS	R	-	1_1C	*4-70*
	SNAPSHOT_VIF_COUNT	R/W	-	1_1F	*4-71*

Table 3-3 Registers Listed by Mnemonic 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	*5-24*
	SRC_HEIGHT1_WIDTH1	W	-	0_66	*5-25*
	SRC_HEIGHT2	R/W	-	0_6B	*5-25*
	SRC_HEIGHT2_WIDTH2	W	-	0_6C	*5-26*
	SRC_OFF_PITCH	R/W	-	0_60	*5-26*
	SRC_WIDTH1	R/W	-	0_64	*5-27*
	SRC_WIDTH2	R/W	-	0_6A	*5-28*
	SRC_X	R/W	-	0_61	*5-29*
	SRC_X_START	R/W	-	0_67	*5-29*
	SRC_Y	R/W	-	0_62	*5-30*
	SRC_Y_START	R/W	-	0_68	*5-31*
	SRC_Y_X	W	-	0_63	*5-31*
	SRC_Y_X_START	W	-	0_69	*5-32*
Memory Buffer Control	**TIMER_CONFIG**	**R/W**	**✓**	**0_0A**	***4-11***
Draw Engine Destination Trajectory	TRAIL_BRES_DEC	R/W	-	0_50	*5-16*
	TRAIL_BRES_ERR	R/W	-	0_4E	*5-16*
	TRAIL_BRES_INC	R/W	-	0_4F	*5-16*
Data Path	USR1_DST_0FF_PITCH	W	-	0_AE	*5-52*
	USR2_DST_0FF_PITCH	W	-	0_AF	*5-52*
	USR_DST_PITCH	W	-	0_BC	*5-52*
Setup Engine	VERTEX_3_SECONDARY_W	R/W	-	1_B2	*6-46*
	VERTEX_3_SPEC_ARGB	R/W	-	1_A3,B6	*6-41*
	VERTEX_3_T	R/W	-	1_A1,B2	*6-40*
	VERTEX_3_W	R/W	-	1_A2,B3	*6-41*
	VERTEX_3_X_Y	R/W	-	1_A6,BF	*6-42*
	VERTEX_3_Z	R/W	-	1_A4,B9	*6-41*

Table 3-3 Registers Listed by Mnemonic Cont'd

Registers by Mnemonic					
Register Class	**Mnemonic**	**Read/ Write**	**Block I/O**	**DWORD Offset (h)**	**Page**
Memory Buffer Control	VGA_DSP_CONFIG	R/W	✓	0_13	*4-16*
	VGA_DSP_ON_OFF	R/W	✓	0_14	*4-16*
Draw Engine Destination Trajectory	Z_CNTL	R/W	-	0_53	*5-17*
	Z_OFF_PITCH	R/W	-	0_52	*5-17*

3.3 Listings of Indexed Registers

3.3.1 PLL Registers Sorted by Address

Table 3-4 PLL Registers Sorted by Address

PLL Registers Sorted by Address			
Address	**Name**	**Read/Write**	**Page**
0	MPLL_CNTL	R/W	*4-76*
1	VPLL_CNTL	R/W	*4-77*
2	PLL_REF_DIV	R/W	*4-77*
3	PLL_GEN_CNTL	R/W	*4-77*
4	MCLK_FB_DIV	R/W	*4-78*
5	PLL_VCLK_CNTL	R/W	*4-79*
6	VCLK_POST_DIV	R/W	*4-79*
7	VCLK0_FB_DIV	R/W	*4-80*
8	VCLK1_FB_DIV	R/W	*4-80*
9	VCLK2_FB_DIV	R/W	*4-80*
10	VCLK3_FB_DIV	R/W	*4-80*
11	PLL_EXT_CNTL	R/W	*4-81*
12	DLL1_CNTL	R/W	*4-82*
13	VFC_CNTL	R/W	*4-82*
14	PLL_TEST_CNTL	R/W	*4-83*
15	PLL_TEST_COUNT	R/W	*4-84*
16	LVDS_CNTL0	R/W	*4-84*
17	LVDS_CNTL1	R/W	*4-84*
18	AGP1_CNTL	R/W	*4-85*
19	AGP2_CNTL	R/W	*4-85*
20	DLL2_CNTL	R/W	*4-86*
21	SCLK_FB_DIV	R/W	*4-86*
22	SPLL_CNTL1	R/W	*4-86*
23	SPLL_CNTL2	R/W	*4-87*
24	APLL_STRAPS	R/W	*4-88*
25	EXT_VPLL_CNTL	R/W	*4-89*

Table 3-4 PLL Registers Sorted by Address **Cont'd**

PLL Registers Sorted by Address			
Address	**Name**	**Read/Write**	**Page**
26	EXT_VPLL_REF_DIV	R/W	*4-89*
27	EXT_VPLL_FB_DIV	R/W	*4-90*
28	EXT_VPLL_MSB	R/W	*4-90*
29	HTOTAL_CNTL	R/W	*4-91*
30	BYTE_CLK_CNTL	R/W	*4-91*
31	TV_PLL_CNTL1	R/W	*4-92*
32	TV_PLL_CNTL2	R/W	*4-92*
33	TV_PLL_CNTL	R/W	*4-92*
34	EXT_TV_PLL	R/W	*4-93*
35	V2PLL_CNTL	R/W	*4-93*
36	PLL_V2CLK_CNTL	R/W	*4-94*
37	EXT_V2PLL_REF_DIV	R/W	*4-94*
38	EXT_V2PLL_FB_DIV	R/W	*4-95*
39	EXT_V2PLL_MSB	R/W	*4-95*
40	HTOTAL2_CNTL	R/W	*4-96*
41	PLL_YCLK_CNTL	R/W	*4-96*
42	PM_DYN_CLK_CNTL	R/W	*4-97*

3.3.2 PLL Registers Sorted by Name

Table 3-5 PLL Registers by Name

PLL Registers Sorted by Name			
Address	**Name**	**Read/Write**	**Page**
18	AGP1_CNTL	R/W	*4-85*
19	AGP2_CNTL	R/W	*4-85*
24	APLL_STRAPS	R/W	*4-88*
30	BYTE_CLK_CNTL	R/W	*4-91*
12	DLL1_CNTL	R/W	*4-82*
20	DLL2_CNTL	R/W	*4-86*
34	EXT_TV_PLL	R/W	*4-93*
38	EXT_V2PLL_FB_DIV	R/W	*4-95*
39	EXT_V2PLL_MSB	R/W	*4-95*
37	EXT_V2PLL_REF_DIV	R/W	*4-94*
25	EXT_VPLL_CNTL	R/W	*4-89*
27	EXT_VPLL_FB_DIV	R/W	*4-90*
28	EXT_VPLL_MSB	R/W	*4-90*
26	EXT_VPLL_REF_DIV	R/W	*4-89*
40	HTOTAL2_CNTL	R/W	*4-96*
29	HTOTAL_CNTL	R/W	*4-91*
16	LVDS_CNTL0	R/W	*4-84*
17	LVDS_CNTL1	R/W	*4-84*
4	MCLK_FB_DIV	R/W	*4-78*
0	MPLL_CNTL	R/W	*4-76*
11	PLL_EXT_CNTL	R/W	*4-81*
3	PLL_GEN_CNTL	R/W	*4-77*
2	PLL_REF_DIV	R/W	*4-77*
14	PLL_TEST_CNTL	R/W	*4-83*
15	PLL_TEST_COUNT	R/W	*4-84*
36	PLL_V2CLK_CNTL	R/W	*4-94*
5	PLL_VCLK_CNTL	R/W	*4-79*
41	PLL_YCLK_CNTL	R/W	*4-96*

Table 3-5 PLL Registers by Name Cont'd

PLL Registers Sorted by Name			
Address	**Name**	**Read/Write**	**Page**
42	PM_DYN_CLK_CNTL	R/W	*4-97*
21	SCLK_FB_DIV	R/W	*4-86*
22	SPLL_CNTL1	R/W	*4-86*
23	SPLL_CNTL2	R/W	*4-87*
33	TV_PLL_CNTL	R/W	*4-92*
31	TV_PLL_CNTL1	R/W	*4-92*
32	TV_PLL_CNTL2	R/W	*4-92*
35	V2PLL_CNTL	R/W	*4-93*
7	VCLK0_FB_DIV	R/W	*4-80*
8	VCLK1_FB_DIV	R/W	*4-80*
9	VCLK2_FB_DIV	R/W	*4-80*
10	VCLK3_FB_DIV	R/W	*4-80*
6	VCLK_POST_DIV	R/W	*4-79*
13	VFC_CNTL	R/W	*4-82*
1	VPLL_CNTL	R/W	*4-77*

3.3.3 LCD Panel Registers Sorted by Index Number

Table 3-6 LCD Panel Registers Sorted by Index Number

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

Table 3-6 LCD Panel Registers Sorted by Index Number **Cont'd**

LCD Panel Registers Sorted by Index Number				
Name	**Read/Write**	**Index**	**Offset(h)**	**Page**
LT_GIO	R/W	7	0_2A	*8-13*
POWER_MANAGEMENT	**R/W**	**8**	**0_2A**	***8-16***
ZVGPIO	R/W	9	0_2A	*8-15*
ICON_CLR0	**R/W**	**A**	**0_2A**	***8-20***
ICON_CLR1	**R/W**	**B**	**0_2A**	***8-21***
ICON_OFFSET	**R/W**	**C**	**0_2A**	***8-21***
ICON_HORZ_VERT_POSN	**R/W**	**D**	**0_2A**	***8-21***
ICON_HORZ_VERT_OFF	**R/W**	**E**	**0_2A**	***8-22***
ICON2_CLR0	**R/W**	**F**	**0_2A**	***8-22***
ICON2_CLR1	**R/W**	**10**	**0_2A**	***8-22***
ICON2_OFFSET	**R/W**	**11**	**0_2A**	***8-22***
ICON2_HORZ_VERT_POSN	**R/W**	**12**	**0_2A**	***8-23***
ICON2_HORZ_VERT_OFF	**R/W**	**13**	**0_2A**	***8-22***
LCD_MISC_CNTL	**R/W**	**14**	**0_2A**	***8-23***
APC_CNTL	**R/W**	**1C**	**0_2A**	***8-25***
POWER_MANAGEMENT_2	**R/W**	**1D**	**0_2A**	***8-18***
ALPHA BLENDING	**R/W**	**25**	**0_2A**	***8-34***
PORTRAIT_GEN_CNTL	**R/W**	**26**	**0_2A**	***8-35***
APC_CTRL_IO	**R/W**	**27**	**0_2A**	***8-26***
TEST_IO	**R/W**	**28**	**0_2A**	***8-27***
TEST_OUTPUTS	**R**	**29**	**0_2A**	***8-28***
DP1_MEM_ACCESS	**R**	**2A**	**0_2A**	***8-28***
DP0_MEM_ACCESS	**R/W**	**2B**	**0_2A**	***8-28***
DP0_DEBUG_A	**R/W**	**2C**	**0_2A**	***8-29***
DP0_DEBUG_B	**R**	**2D**	**0_2A**	***8-29***
DP1_DEBUG_A	**R/W**	**2E**	**0_2A**	***8-30***
DP1_DEBUG_B	**R**	**2F**	**0_2A**	***8-30***
DPCTRL_DEBUG_A	**R**	**30**	**0_2A**	***8-31***
DPCTRL_DEBUG_B	**R**	**31**	**0_2A**	***8-31***

Table 3-6 LCD Panel Registers Sorted by Index Number **Cont'd**

LCD Panel Registers Sorted by Index Number				
Name	**Read/Write**	**Index**	**Offset(h)**	**Page**
MEMBLK_DEBUG	**R**	**32**	**0_2A**	***8-32***
APC_LUT_AB	**R/W**	**33**	**0_2A**	***8-32***
APC_LUT_CD	**R/W**	**34**	**0_2A**	***8-32***
APC_LUT_EF	**R/W**	**35**	**0_2A**	***8-33***
APC_LUT_GH	**R/W**	**36**	**0_2A**	***8-33***
APC_LUT_IJ	**R/W**	**37**	**0_2A**	***8-33***
APC_LUT_KL	**R/W**	**38**	**0_2A**	***8-33***
APC_LUT_MN	**R/W**	**39**	**0_2A**	***8-34***
APC_LUT_OP	**R/W**	**3A**	**0_2A**	***8-34***

3.3.4 LCD Panel Registers Sorted by Name

Table 3-7 LCD Panel Registers Sorted by Name

LCD Registers by Sorted Name				
Name	**Read/Write**	**Index**	**Offset(h)**	**Page**
ALPHA BLENDING	**R/W**	**25**	**0_2A**	***8-34***
APC_CNTL	**R/W**	**1C**	**0_2A**	***8-25***
APC_CTRL_IO	**R/W**	**27**	**0_2A**	***8-26***
APC_LUT_AB	**R/W**	**33**	**0_2A**	***8-32***
APC_LUT_CD	**R/W**	**34**	**0_2A**	***8-32***
APC_LUT_EF	**R/W**	**35**	**0_2A**	***8-33***
APC_LUT_GH	**R/W**	**36**	**0_2A**	***8-33***
APC_LUT_IJ	**R/W**	**37**	**0_2A**	***8-33***
APC_LUT_KL	**R/W**	**38**	**0_2A**	***8-33***
APC_LUT_MN	**R/W**	**39**	**0_2A**	***8-34***
APC_LUT_OP	**R/W**	**3A**	**0_2A**	***8-34***
CONFIG_PANEL	R/W	0	0_2A	*8-3*
DP0_DEBUG_A	**R/W**	**2C**	**0_2A**	***8-29***
DP0_DEBUG_B	**R**	**2D**	**0_2A**	***8-29***

Table 3-7 LCD Panel Registers Sorted by Name **Cont'd**

LCD Registers by Sorted Name				
Name	**Read/Write**	**Index**	**Offset(h)**	**Page**
DP0_MEM_ACCESS	**R/W**	**2B**	**0_2A**	***8-28***
DP1_DEBUG_A	**R/W**	**2E**	**0_2A**	***8-30***
DP1_DEBUG_B	**R**	**2F**	**0_2A**	***8-30***
DP1_MEM_ACCESS	**R**	**2A**	**0_2A**	***8-28***
DPCTRL_DEBUG_A	**R**	**30**	**0_2A**	***8-31***
DPCTRL_DEBUG_B	**R**	**31**	**0_2A**	***8-31***
DSTN_CONTROL	**R/W**	**2**	**0_2A**	***8-8***
EXT_VERT_STRETCH	**R/W**	**6**	**0_2A**	***8-12***
HFB_PITCH_ADDR	R/W	3	0_2A	*8-9*
HORZ_STRETCHING	**R/W**	**4**	**0_2A**	***8-10***
ICON2_CLR0	**R/W**	**F**	**0_2A**	***8-22***
ICON2_CLR1	**R/W**	**10**	**0_2A**	***8-22***
ICON2_HORZ_VERT_OFF	**R/W**	**13**	**0_2A**	***8-22***
ICON2_HORZ_VERT_POSN	**R/W**	**12**	**0_2A**	***8-23***
ICON2_OFFSET	**R/W**	**11**	**0_2A**	***8-22***
ICON_CLR0	**R/W**	**A**	**0_2A**	***8-20***
ICON_CLR1	**R/W**	**B**	**0_2A**	***8-21***
ICON_HORZ_VERT_OFF	**R/W**	**E**	**0_2A**	***8-22***
ICON_HORZ_VERT_POSN	**R/W**	**D**	**0_2A**	***8-21***
ICON_OFFSET	**R/W**	**C**	**0_2A**	***8-21***
LCD_GEN_CTRL	**R/W**	**1**	**0_2A**	***8-5***
LCD_INDEX	R/W		0_29	*8-2*
LCD_MISC_CNTL	**R/W**	**14**	**0_2A**	***8-23***
LT_GIO	R/W	7	0_2A	*8-13*
MEMBLK_DEBUG	**R**	**32**	**0_2A**	***8-32***
PORTRAIT_GEN_CNTL	**R/W**	**26**	**0_2A**	***8-35***
POWER_MANAGEMENT	**R/W**	**8**	**0_2A**	***8-16***
POWER_MANAGEMENT_2	**R/W**	**1D**	**0_2A**	***8-18***
TEST_IO	**R/W**	**28**	**0_2A**	***8-27***

Table 3-7 LCD Panel Registers Sorted by Name **Cont'd**

LCD Registers by Sorted Name				
Name	**Read/Write**	**Index**	**Offset(h)**	**Page**
TEST_OUTPUTS	**R**	**29**	**0_2A**	***8-28***
VERT_STRETCHING	R/W	5	0_2A	*8-11*
ZVGPIO	R/W	9	0_2A	*8-15*

3.3.5 Sorting of the VGA Compatible Registers

For a sorted list of the VGA compatible registers, refer to page *7-1*.

Chapter 4
Display and Configuration

This chapter describes the registers used for configuring the GUI draw engine functions. For an explanation of the notations used to describe the registers, refer to Chapter 1, section *1.3.3*

.

Table 4-1 Register Classes

4.1 Setup and Control Registers

4.1.1 General I/O Control Registers

		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
		dd	cc	bb	aa	z	y	x	w	v	u	t	s	r	q	p	o			n	m	l	k	j	i	h	g	f	e	d	c	b	a
a	R/W	GP_IO_0																															
		Write/Read (DIR0 = output/input) for pin: GPIO(0)																															
b	R/W	GP_IO_1																															
		Write/Read (DIR1 = output/input) for pin: GPIO(1)																															
c	R/W	GP_IO_2																															
		Write/Read (DIR2 = output/input) for pin: GPIO(2)																															
d	R/W	GP_IO_3																															
		Write/Read (DIR3 = output/input) for pin: GPIO(3)																															
e	R/W	GP_IO_4																															
		Write/Read (DIR4 = output/input) for pin: GPIO(4)																															
f	R/W	GP_IO_5																															
		Write/Read (DIR5 = output/input) for pin: GPIO(5)																															
g	R/W	GP_IO_6																															
		Write/Read (DIR6 = output/input) for pin: GPIO(6)																															
h	R/W	GP_IO_7																															
		Write/Read (DIR7 = output/input) for pin: GPIO(7)																															
i	R/W	GP_IO_8																															
		Write/Read (DIR8 = output/input) for pin: GPIO(8)																															
j	R/W	GP_IO_9																															
		Write/Read (DIR9 = output/input) for pin: GPIO(9)																															
k	R/W	GP_IO_A																															
		Write/Read (DIRA = output/input) for pin: GPIO(10)																															
l	R/W	GP_IO_B																															
		Write/Read (DIRB = output/input) for pin: GPIO(11)																															
m	R/W	GP_IO_C																															
		Write/Read (DIRC = output/input) for pin: GPIO(12)																															
n	R/W	GP_IO_D																															
		Write/Read (DIRD = output/input) for pin: GPIO(13)																															
o	R/W	GP_IO_DIR_0																															
		I/O direction for pin GPIO(0) 0 = Input 1 = Output																															
p	R/W	GP_IO_DIR_1																															
		I/O direction for pin GPIO(1) 0 = Input 1 = Output																															
q	R/W	GP_IO_DIR_2																															
		I/O direction for pin GPIO(2) 0 = Input 1 = Output																															
r	R/W	GP_IO_DIR_3																															
		I/O direction for pin GPIO(3) 0 = Input 1 = Output																															
s	R/W	GP_IO_DIR_4																															
		I/O direction for pin GPIO(4) 0 = Input 1 = Output																															

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	
		dd cc bb aa z y x w v u t s r q p o	n m l k j i h g f e d c b a
t	R/W	GP_IO_DIR_5	I/O direction for pin GPIO(5) 0 = Input 1 = Output
u	R/W	GP_IO_DIR_6	I/O direction for pin GPIO(6) 0 = Input 1 = Output
v	R/W	GP_IO_DIR_7	I/O direction for pin GPIO(7): 0 = Input 1 = Output
w	R/W	GP_IO_DIR_8	I/O direction for pin GPIO(8): 0 = Input 1 = Output
x	R/W	GP_IO_DIR_9	I/O direction for pin GPIO(9) 0 = Input 1 = Output
y	R/W	GP_IO_DIR_A	I/O direction for pin GPIO(10) 0 = Input 1 = Output
z	R/W	GP_IO_DIR_B	I/O direction for pin GPIO(11) 0 = Input 1 = Output
aa	R/W	GP_IO_DIR_C	I/O direction for pin GPIO(12) 0 = Input 1 = Output
bb	R/W	GP_IO_DIR_D	I/O direction for pin GPIO(13) 0 = Input 1 = Output
cc	**R/W**	**MPP_IO_DRIVE**	**MPP I/O output drive strength** **Applies to GPIO pins 0 to 11 inclusive** **0 = No boost** **1 = Boost**
dd	**R/W**	**I2C_IO_DRIVE**	**I2C I/O output drive strength** **Applies to GPIO pins 12 and 13** **0 = No boost** **1 = Boost**

Description

This register specifies the data and direction for pins GPIO[13:0] of the General Purpose I/O bus. For the data/direction of pins GPIO[46:44], refer to register *LT_GIO on page 8-13*.

Usage

For details on the typical pin configurations used to support the various operational modes (VFC, DVS, etc.), refer to the RAGE Mobility Graphics Controller Specifications.

4.1.2 Scratch Pad Registers

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

This register is a general-purpose, scratch-pad, storage register. Use these registers to allow two decoupled programs to exchange information. Typically, these registers 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 *4-5*, SCRATCH_REG2 on *4-5*, and SCRATCH_REG3 on *4-6*.

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

This register is similar to the SCRATCH_REG0 register on *4-4*.

Usage

Refer to the SCRATCH_REG0 register on *4-4*.

See Also

SCRATCH_REG0 on *4-4*, SCRATCH_REG2 on *4-5*, and SCRATCH_REG3 on *4-6*.

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				
	a																																			
a	R/W	SCRATCH_REG2																		Scratch pad 2																

Description

This register is similar to the SCRATCH_REG0 register on *4-4*.

Usage

Refer to the SCRATCH_REG0 register on *4-4*.

See Also

SCRATCH_REG0 on *4-4*, SCRATCH_REG1 on *4-5* and SCRATCH_REG3 on *4-6*

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
	a																															

a	R/W	SCRATCH_REG3	Scratch pad 3

Description

This register is similar to the SCRATCH_REG0 register on *4-4*.

Usage

Refer to the SCRATCH_REG0 register on *4-4*.

See Also

SCRATCH_REG0 on *4-4*, SCRATCH_REG1 on *4-5*, SCRATCH_REG2 on *4-5*.

mach64 Programmer's Guide:

- *Advanced Topics: Boot-time Initialization*

4.1.3 Bus Control Registers

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				l		h	g	f	e	d	c	b	a

a	R/W	BUS_DBL_RESYNC	Add 1 clock-settling time to BCLK and MCLK re-synchronizers: (default = 1)
b	W	BUS_MSTR_RESET	To reset the bus master, write a '1' to this bit (one shot, no need to write '0')
c	W	BUS_FLUSH_BUF	To flush data from the buffer, write a '1' to this bit (one shot, no need to write '0')
d	R/W	BUS_STOP_REQ_DIS	Disable the burst read requests after stop has been asserted: (default = 0) 0 = Normal operation 1 = Disable
e	R/W	BUS_APER_REG_DIS	Disable memory-mapped register decoding in the linear aperture: (default = 0) 0 = Enable 1 = Disable

Cont'd		BUS_CNTL																													
BITS		31	30	29	28	27	26	25	24	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				l		h	g	f	e	d	c	b	a

f	R/W	BUS_EXTRA_PIPE_DIS	Disable extra pipeline stage: (default = 0) 0 = Enable 1 = Disable
g	R/W	BUS_MASTER_DIS	Disable bus master operation: 0 =Enable 1= Disable
h	R/W	ROM_WRT_EN	Enable write to flash memory BIOS: (default = 0) 0 = Disable 1 = Enable (ROMWRTEN strap must also be enabled)
i	**R**	CHIP_HIDDEN_REV	These bits are used for driver/BIOS to identify chip rev from A2x and up. If these bits are writable: mobility A2x 00 = mobility A31 01 = reserved 10 = reserved 11 = reserved
j	R/W	BUS_PCI_READ_RETRY_EN	Enable 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 expires)
k	R/W	BUS_PCI_WRT_RETRY_EN	Enable 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 greater than cache line size reg., use 'read line' command 1 = When transfer length greater than cache line size reg., use 'read multiple' command
n	R/W	BUS_MSTR_RD_LINE	Set the 'read line' command for bus master**:** (default = 0) 0 = Use 'read' command exclusively 1 = When transfer length greater than 1, use 'read line' command
o	R/W	BUS_SUSPEND	Set mode for current bus master transfer 1 = Suspend (this transfer will resume when this bit is cleared) 0 = Resume
p	R	LAT16X	Set value of the latency timer 1 = Multiply the latency timer value by 16 0 = Use the latency timer value as is

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				l		h	g	f	e	d	c	b	a

q	R/W	BUS_RD_DISCARD_EN	Enable PCI slave read data: (default = 0) 1 = Enable 0 = Disable
r	R/W	BUS_RD_ABORT_EN	Enable the slave's delayed read transaction abort (for BM conflicts) 1 = Enable 0 = Disable
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	Enable extended register block 1: (default = 0) 0 = Disable 1 = Enable
u	R/W	BUS_MSTR_DISCONNECT_EN	Enable bus master disconnect after allowed wait states has elapsed: (default = 0) 1 = Enable 0 = Disable
v	R/W	BUS_WRT_BURST	Enable burst write transfers: (default = 0) 0 = Disable 1 = Enable
w	R/W	BUS_READ_BURST	Enable burst read transfers: (default = 0) 0 = Disable 1 = Enable
x	R/W	BUS_RDY_READ_DLY	Set delay for the bus memory read RDY signal: (default = 1) 0 = No delay 1 = Delay RDY by 1 memory clock

Description

This register configures the on-chip bus interface, controls the bus mastering, and controls the 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 4.2 below 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 4-2 Conversion of the BUS_TETRY_WS Field

BUS_RETRY_WS	Actual Number of Clocks to TRDY or STOP	Time (@33 MHz)
0	3	90 ns
1	5	150 ns
2h	7h	210 ns
3h	8h	240 ns
4h	9h	270 ns
5h	Bh	330 ns
6h	Eh	420 ns
7h	Fh	470 ns
8h	13h	570 ns
9h	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 Registers

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
										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
										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

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
	e			**d**									**c**				**b**								a							

a	R/W	VID_INTRA_ACCESS_TIMER	Video intra-access time for memory accesses (in XCLKs)
b	**R/W**	**PM_VID_INTRA_ACCESS_TIMER**	**Video intra-access time for memory accesses (in PM_XCLKs)**

Cont'd	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
	e			d									c				b								a							

c	**R/W**	**PM_MEM_REFRESH_RATE**	**Refresh rate for Set depending on PM_XCLK frequency:** **0000 = 10 to 49 MHz, 1 refresh every 156 PM_XCLKs** **0001 = 50 to 65 MHz, 1 refresh every 781 PM_XCLKs** **0010 = 66 to 74 MHz, 1 refresh every 1031 PM_XCLKs** **0011 = 75 to 82 MHz, 1 refresh every 1172 PM_XCLKs** **0100 = 83 to 89 MHz, 1 refresh every 1297 PM_XCLKs** **0101 = 90 to 94 MHz, 1 refresh every 1406 PM_XCLKs** **0110 = 95 to 99 MHz, 1 refresh every 1484 PM_XCLKs** **0111 = 100 to 104 MHz, 1 refresh every 1563 PM_XCLKs** **1000 = 105 to 109 MHz, 1 refresh every 1641 PM_XCLKs** **1001 = 110 to 114 MHz, 1 refresh every 1719 PM_XCLKs** **1010 = 115 to 119 MHz, 1 refresh every 1797 PM_XCLKs** **1011 = 120 to 124 MHz, 1 refresh every 1875 PM_XCLKs** **1100 = 125 to 132 MHz, 1 refresh every 1953 PM_XCLKs** **1101 = 133 to 142 MHz, 1 refresh every 2078 PM_XCLKs** **1110 = 143 to 165 MHz, 1 refresh every 2234 PM_XCLKs** **1111 = 166 and above, 1 refresh every 2594 PM_XCLKs**
d	R/W	VID_TIMER_MODE	Video timer mode 0 = Free running (independent) 1 = Relative to end of display burst
e	**R/W**	**PM_DSP_RW_SEL**	**Toggles register access/definition from XCLK display FIFO registers: (default = 0)** **DSP_CONFIG, DSP_ON_OFF, DSP2_CONFIG, DSP2_ON_OFF, VGA_DSP_CONFIG and VGA_DSP_ON_OFF) to PM_XCLK display FIFO registers (PM_DSP_CONFIG, PM_DSP_ON_OFF, PM_DSP2_CONFIG, PM_DSP2_ON_OFF, PM_VGA_DSP_CONFIG, PM_VGA_DSP_ON_OFF)** **These registers share the same offset** **0 = Register read/writes go to XCLK version of display FIFO registers** **1 = Register read/writes go to PM_XCLK version of display FIFO registers**

4.1.5 Power Management Mode Registers

This mode is used only in the RAGE Mobility. The power management mode uses the following memory controller registers (denoted by the prefix PM_). To use the slower version of XCLK, referred to as PM_XCLK, set the requester activity in a specific state that is defined by the POWER_MANAGEMENT_2 registers.

The PM_DSP_RW_SEL bit in TIMER_CONFIG will select which display FIFO registers (which share the same offset) to access. For example:

- **To assess DSP_ON_OFF, set PM_DSP_RW_SEL=0.**
- **To access PM_DSP_ON_OFF, set PM_DSP_RW_SEL=1 (for that register offset).**

PM_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
Mobility		c												b						a											
a	R/W	PM_DSP_XCLKS_PER_QW												Amount of time (in PM_XCLKs) that one QWORD in the display FIFO occupies																		
b	R/W	PM_DSP_LOOP_LATENCY												Display FIFO control parameter (for PM_XCLK mode)																		
c	R/W	PM_DSP_PRECISION												Integer.fraction precision point for (for PM_XCLK mode): DSP_XCLKS_PER_QW DSP_ON DSP_OFF																		

PM_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
Mobility		b																					a									
a	R/W	PM_DSP_OFF												The display memory request off threshold time in terms of PM_XCLKs																		
b	R/W	PM_DSP_ON												The display memory request on threshold time in terms of PM_XCLKs																		

PM_VGA_DSP_CONFIG																															Offset: 0_13
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mobility						c				b									a													

a	R/W	PM_VGA_DSP_XCLKS_PER_QW	Amount of time in PM_XCLKs that one QWORD in the display FIFO occupies
b	R/W	PM_VGA_DSP_PREC_PCLKBY2	Integer.fraction precision point for (for PM_XCLK mode): VGA_DSP_PREC_PCLK+1
c	R/W	PM_VGA_DSP_PREC_PCLK	Integer.fraction precision point for (for PM_XCLK mode): PM_VGA_DSP_XCLKS_PER_QW PM_VGA_DSP_ON PM_VGA_DSP_OFF

PM_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
Mobility																	b		a													

a	R/W	PM_VGA_DSP_OFF	The display memory request off threshold time in terms of PM_XCLKs
b	R/W	PM_VGA_DSP_ON	The display memory request on threshold time in terms of PM_XCLKs

PM_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
Mobility											c		b						a													

a	R/W	PM_DSP2_XCLKS_PER_QW	Amount of time in PM_XCLKs that one QWORD in the second display FIFO occupies
b	R/W	PM_DSP2_LOOP_LATENCY	Display FIFO control parameter (for PM_XCLK mode)
c	R/W	PM_DSP2_PRECISION	Integer.fraction precision point for (for PM_XCLK mode): DSP2_XCLKS_PER_QW DSP2_ON DSP2_OFF

PM_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
Mobility	**b**																					**a**										

a	**R/W**	**PM_DSP2_OFF**	**The display memory request off threshold time in terms of PM_XCLKs**
b	**R/W**	**PM_DSP2_ON**	**The display memory request on threshold time in terms of PM_XCLKs**

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

a	R/W	Z_WB_FLUSH	Z write buffer flush 0000 = Flush when buffer is full 0001-1011 = Flush when [0001-1011] QWORDs are present in write buffer
b	R/W	VID_WB_FLUSH	Video write buffer flush (see also VID_WB_FLUSH_MSB above) 0000 = Flush when buffer is full 0001-1011 = Flush when [0001-1011] QWORDs are present in write buffer
c	R/W	GUI_WB_FLUSH	GUI write buffer flush 00000 = Flush when buffer is full 00001-10111 = Flush when [00001-10111] QWORDs are present in write buffer
d	R/W	HST_WB_FLUSH	Host write buffer flush 000 = Flush when buffer is full 001-111 = Flush when [001-111] 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

VGA_DSP_CONFIG — Offset: 0_13

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

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
						h	**g**	**f**				**e**			**d**									**c**		b			a			

a	R/W	MEM_ROW_MAPPING	Row Address Mapping: (default = 0) MA: 12 11 10 9 8 7 6 5 4 3 2 1 0 0 = X A22 A21 A11 A20 A19 A18 A17 A16 A15 A14 A13 A12 1 = X A22 A11 A21 A20 A19 A18 A17 A16 A15 A14 A13 A12 2 = X A11 A22 A21 A20 A19 A18 A17 A16 A15 A14 A13 A12 3 = A22 A10 A21 A20 A19 A18 A17 A16 A15 A14 A13 A12 A11 4 - 7 = reserved
b	R/W	MEM_COL_MAPPING	Column Address Mapping: (default = 0) MA: 12 11 10 9 8 7 6 5 4 3 2 1 0 0 = A22 A11 A11 A11 A11 A10 A9 A8 A7 A6 A5 A4 A3 1 = A22 A10 A10 A10 A10 A9 A8 A7 A6 A5 A4 A3 A2 2 - 7 = reserved

Cont'd	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
						h	g	f			e				d								c			b			a		

c	R/W	MEM_GROUP_SIZE	Memory group size: 00 = 2MB 01 = 4MB 10 = 8MB or 4MB 32 bit interface 11 = 8MB for 32 bit interface This register field will be used in 11x8,12x8 and 13x8 SD/SGRAM configurations. Otherwise it is 00. The SGRAM's address pins will connect to our chip as follows: SGRAM's A0-A9 <=> RAGE MOBILITY's MA0-MA9 SGRAM's A10 <=> RAGE MOBILITY's CS2(2) SGRAM's A11 <=> RAGE MOBILITY's CS2(3) SGRAM's A12 <=> RAGE MOBILITY's DSF For 4MB with 11x8 SGRAM, tie SGRAM's CS to low. For 8MB with 11x8 SGRAM, tie SGRAM's CS to RAGE MOBILITY's CS0 and CS1.
d	R/W	MEM_LATCHL	Memory data latching mechanism for lower 32 bits: 00 = Inverted HCLK feedback by 2 (used for 2:1 SDR mode) 01 = HCLK feedback 10 = Positive edge of XCLK 11 = Negative edge of XCLK
e	R/W	MEM_LATCHU	Memory data latching mechanism for upper 32-bits: 00 = Reserved 01 = HCLK feedback 10 = Positive edge of XCLK 11 = Negative edge of XCLK
f	R/W	MEM_LATCH_ALWAYS	Dynamically controls read latching flops in memory data pads to reduce power 0 = Clock selected by MEM_LATCH[U|L] only active during read cycles 1 = Clock selected by MEM_LATCH[U|L] always active
g	R/W	MEM_CKE_ALWAYS	Control CKE signal to memory chip for power 0 = CKE pin always active except for power management modes 1 = CKE pin controlled dynamically
h	R/W	MEM_REQ_LOCK	Memory lock out control (default = 0) 0 = Unlocked 1 = Lock out requestors

Table 4-3 Memory Combinations

Interface	Memory Device	MEM_ROW_ MAPPING	MEM_COL_ MAPPING	MEM_GROUP_ SIZE
64 bit	10x8x32 bit	0	0	0
64 bit	11x8x32 bit	1	0	1
64 bit	12x8x32 bit	2	0	2
64 bit	12x8x16 bit	2	0	2
32 bit	12x8x16 bit	3	1	2

4.1.6 Memory Control Registers

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
	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**	**MEM_TBWC**	**Set block write cycle control** **0 = 1 clock block write** **1 = 2 clock block write (previous default)**
b	R/W	MEM_SDRAM_RESET	Invoke reset of SDRAM 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 configurations.
c	R/W	MEM_CYC_TEST	Invoke memory cycle test mode. (note**:** the chip WILL NOT FUNCTION NORMALLY in this mode) 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'.
d	**R/W**	**MEM_MA_YCLK**	**Output clock select for MA pins** **0 = Propagate off of XCLK** **1 = Propagate off of YCLK**

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	
		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**	**MEM_CNTL_YCLK**	**Output clock select for RAS, CAS, WE and CKE pins** **0 = Propagate off of XCLK** **1 = Propagate off of YCLK**
f	**R/W**	**MEM_CS_YCLK**	**Output clock select for CS pins** **0 = Propagate off of XCLK** **1 = Propagate off of YCLK**
g	**R/W**	**MEM_EXTND_ERST**	**Extend enable reset (for MD pad) by 1 clock cycle: (default = 1)** **0 = No extension** **1 = Extend**
h	**R/W**	**MEM_ERST_CNTL**	**Enable reset (for MD pad) timing control: (default = 00)** **00 = (CL - 1) clocks** **01 = (CL - 1/2) clocks** **10 = CL clock** **11 = Always enabled**
i	**R/W**	**MEM_CAS_LATENCY**	**Memory latency between CAS and data present** **Typically, same setting as MEM_CAS_LATENCY** **00 = 1 clock** **01 = 2 clocks (SDRAM CL = 1)** **10 = 3 clocks (SDRAM CL = 2)** **11 = 4 clocks (SDRAM CL = 3)**
j	**R/W**	**MEM_SSTL_EN**	**Interface select for MD pins** **0 = LVTTL interface** **1 = SSTL interface**
k	**R/W**	**MEM_MD_REC**	**Receiver select for MD pins** **0 = Hysteresis receiver** **1 = Differential receiver**
l	**R/W**	**MEM_HCLK_DRIVE**	**Boost drive strength of HCLK pin** **0 = No boost** **1 = Boost**
m	**R/W**	**MEM_CS_DRIVE**	**Boost drive strength of CS pins** **0 = No boost** **1 = Boost**
n	**R/W**	**MEM_MDA_DRIVE**	**Boost drive strength of upper 32 MD pins not connected to BIOS (49, 52-55)** **0 = No boost** **1 = Boost**
o	**R/W**	**MEM_MDB_DRIVE**	**Boost drive strength of pins not connected to BIOS (32-48, 50-51, 56-63)** **0 = No boost** **1 = Boost**

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

p	**R/W**	**MEM_MDL_DRIVE**	**Boost drive strength of low 32 MD pins (0-31)** **0 = No boost** **1 = Boost**
q	**R/W**	**MEM_GROUP_CHANGE_EN**	**Enable group change signaling: (default = 0)** **0 = Disable** **1 = Enable**
r	**R/W**	**MEM_MA_DRIVE**	**Boost drive strength of MA pins** **0 = No boost** **1 = Boost**
s	**R/W**	**MEM_CNTL_DRIVE**	**Boost drive strength of RAS, CAS, WE, DSF pins** **0 = No boost** **1 = Boost**
t	**R/W**	**MEM_DQM_DRIVE**	**Boost drive strength of DQM pins** **0 = No boost** **1 = Boost**
u	R/W	MEM_GCMRS	Set mode 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
v	**R**	**MEM_CS_STRAP**	**Chip select enable/disable strap read at hardware reset (read only)** **0 = Enable CS pins (strap MD(47) to VCC)** **1 = Disable CS pins (no strap on MD(47), built-in pull-down)** **Forced to 0 (enable) in UMA mode.**
w	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
x	R/W	MEM_ALL_PAGE_DIS	Controls all page memory cycles 0 = Enable 1 = Disable

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
	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
y	R/W	MEM_GROUP_FAULT_EN										Controls page faulting between 2MB groups 0 = Enable 1 = Disable																			

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

a	R/W	MEM_SIZE	Memory size: 0000 = Reserved 0001 = 1MB 0010 = Reserved 0011 = 2MB 0100 - 0110 = Reserved 0111 = 4MB 1000 = Reserved 1001 = 6MB 1010 = Reserved 1011 = 8MB 1100 - 1111 = Reserved **Note:** Only the above sizes are implemented. Reserved settings are for future use according to the following: a: Memory sizes for 0 to 7 increment by 0.5MB steps b: Memory sizes for 8 - 11 increment by 1MB steps c: Memory sizes for 12 - 15 increment by 2MB steps
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_WDOE_CNTL**	**Data pad output enable timing control (default = 00)** **00 = 1 clock before write** **01 = 1/2 clock before write** **10 = 0 clock before write** **11 = 1/2 clock after write**
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
f	R/W	MEM_TCRD	CAS to RAS delay, provided primarily for EDO: 0 = no clock delay between CAS high and RAS high 1 = one clock delay between CAS high and RAS high

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

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_TWR**	**Write recovery time (default = 01)** **00 = 0 clock** **01 = 1 clock** **10 = 2 clock** **11 = 3 clock**
i	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
j	R/W	MEM_REFRESH_DIS	Disable memory refresh 0 = Enable 1 = Disable
k	**R/W**	**MEM_REFRESH_RATE**	**Set memory refresh rate (depends on XCLK frequency)** **0000 = 10 to 49 MHz 1 refresh every 156 XCLKs** **0001 = 50 to 65 MHz 1 refresh every 781 XCLKs** **0010 = 66 to 74 MHz 1 refresh every 1031 XCLKs** **0011 = 75 to 82 MHz 1 refresh every 1172 XCLKs** **0100 = 83 to 89 MHz 1 refresh every 1297 XCLKs** **0101 = 90 to 94 MHz 1 refresh every 1406 XCLKs** **0110 = 95 to 99 MHz 1 refresh every 1484 XCLKs** **0111 = 100 to 104MHz 1 refresh every 1563 XCLKs** **1000 = 105 to 109 MHz 1 refresh every 1641 XCLKs** **1001 = 110 to 114 MHz 1 refresh every 1719 XCLKs** **1010 = 115 to 119 MHz 1 refresh every 1797 XCLKs** **1011 = 120 to 124 MHz 1 refresh every 1875 XCLKs** **1100 = 125 to 132 MHz 1 refresh every 1953 XCLKs** **1101 = 133 to 142 MHz 1 refresh every 2078 XCLKs** **1110 = 143 to 165 MHz 1 refresh every 2234 XCLKs** **1111 = 166 MHz and above 1 refresh every 2594 XCLKs** **Note: No effect in shared configurations**
l	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
			n		**m**		**l**		**k**				**j**	**i**			**h**		g	f	e		d		**c**		b		a			

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

Description

This register configures 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 32KB aperture into 8MB video memory.
b	R/W	MEM_VGA_WPS1	Write page pointer for upper 32KB aperture into 8MB video memory.

Description

This register 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 *4-34*

MEM_VGA_RP_SEL on *4-25*

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 32KB aperture into 8MB video memory.
b	R/W	MEM_VGA_RPS1	Read page pointer for upper 32KB aperture into 8MB video memory.

Description

This register 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 *4-34*

MEM_VGA_WP_SEL on *4-24*

mach64 Programmer's Guide:

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

4.1.7 Test and Debug Registers

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

a		**Same as RAGE III**	
b	**R/W**	**GEN_ICON2_ENABLE**	**Enable second hardware icon: (default = 0)** **0 = Disable** **1 = Enable**
c	**R/W**	**GEN_CUR2_ENABLE**	**Enable second hardware cursor: (default = 0)** **0 = Disable** **1 = Enable**
d	**R/W**	**GEN_ICON_ENABLE**	**Enable hardware icon: (default = 0)** **0 = Disable** **1 = Enable**
e	**R/W**	**GEN_CUR_ENABLE**	**Enable hardware cursor*: (default = 0)** **0 = Disable** **1 = Enable**
f			**Same as RAGE III**
g	**R/W**	**GEN_MEM_TRISTATE**	**Tristate memory interface signals** **0 = Normal operation** **1 = Tristate memory interface signals**
h			**Same as RAGE III**
i	**R/W**	**GEN_TEST_MODE**	**Enable test modes:** **0011 = Palette2 test** **0100 = RAMDAC2 functional test** **others = same as Rage III**
j	**R/W**	**GEN_CRC_EN**	**Enable both CRC and CRC2 signature block: (default = 0)** **0 = Disable** **1 = Enable**
k			**Same as RAGE III**

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

l	R/W	GEN_DEBUG_MODE	Debug Modes: 00 = Memory cycle debug 01 = Display debug 02 = Reserved 03 = SDRAM state machine debug 04 = Display2 debug 05 = Memory 2:1 debug 06 = ROM Controller debug 07 - 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 20 = Top guieng pipe 21 = Bottom of guieng pipe 22 - 3D = interface to guieng 23 - 29 = GUI stuff 2A - 2B = MC/IDCT parser 2C- 2E = MC/IDCT buffer control 2F = reserved 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 5x = Display engine 60 - 6F = AGP debug 7x = Power management states 8x = Second dispeng 90 = IDCT1(9:0) 91 = IDCT2(10:1) 92 = IDCT3(10-7:0) 93 - 9F = Reserved Ax = APC debug Bx = Reserved for Mobility OR TMDS debug for desktop Cx = Subpic debug Dx = Scaler debug E0 - FF = Reserved

Description

This register is used for general control and diagnostic control. Most of the test modes are only used during ASIC testing or for debugging purposes.

Bit [4] enables the second hardware icon.

Bit [5] enables the second hardware cursor. Bit

Bit [6] (or bit [5] in LCD_GEN_CTRL, backward compatibility) enables the hardware icon.

Bit [7] enables the hardware cursor.

Bit [8] resets the Draw engine.

Bits [19:16] enable various test modes of the ASIC.

Bit [21] enables the cyclic redundancy checkers (CRC and CRC2).

Bits [31:24] enable various debug modes of the ASIC.

Usage

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

Application level programs should touch only the following bits:

GEN_ICON2_ENABLE
GEN_CUR2_ENABLE
GEN_ICON_ENABLE
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

Use this register for diagnostics of the CRTC, DAC, hardware cursor, and overscan.

See Also

GEN_TEST_CNTL on *4-27*

CRC2_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
Mobility									**a**																							

a	**R**	**CRC2_SIG**	**Second DAC CRC signature value: (default = 0)**

Description

This register accumulates the signature for the second display CRC.

Usage

Use this register to perform diagnostics on the second CRTC, DAC, hardware cursor, and overscan. Before reading this register, set CRTC_RW_SELECT = 1.

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
	dd	cc	bb	aa	z	y	x	w	v	u	t	s	r	**q**			p	o	n	m	l	k	j	i	h	g	f	e	d	c	b	**a**

a	**R/W**	**BYPASS_SUBPIC_DBF**	**Bypass double buffering of overlay/scaler/subpic registers: (default = 0)** **0 = Do not bypass** **1 = Bypass**

Cont'd		HW_DEBUG																													
BITS		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		dd	cc	bb	aa	z	y	x	w	v	u	t	s	r	**q**			p	o	n	m	l	k	j	i	h	g	f	e	d	c	b	**a**

b	R/W	SYNC_PD_EN	0 = Pull down HSYNC and VSYNC pads in suspend mode 1 = Do not pull down HSYNC and VSYNC pads in suspend mode (default) Note: available only in rev A21
c	R/W	DISP_QW_FIX_DIS	0 = Enable extra display qword fix 1 = Disable extra display qword fix
d	R/W	AGPPLL_FIX_EN	0 = Disable AGPPLL fix 1 = Enable AGPPLL fix to output '0' for X1 and X2 clocks when AGPPLL is being reset (APLL_RESET) and when it is not being bypassed. Note: available only in rev A23 and after.
e	R/W	GUI_BEATS_HOST	Set arbitration between host and GUI request channels 0 = Normal operation 1 = Lock out arbitration to host when GUI is active
f	R/W	INTER_BLIT_FIX_DIS	0 = Enable inter-blit performance improvement 1 = Disable inter-blit performance improvement
g	R/W	INTER_PRIM_DIS	0 = Enable fast-fill/block-write scissoring 1 = Disable fast-fill/block-write scissoring
h	R/W	SRC_TRACK_DST_DIS	0 = Enable SRC_TRACK_DIS fix 1 = Disable SRC_TRACK_DIS fix
i	R/W	AUTO_BLKWRT_COLOR_DIS	0 = Enable auto-color register updates for block writes 1 = Disable auto-color register updates for block writes
j	R/W	INTER_LINE_OVERLAP_DIS	0 = Enable inter-line overlapping 1 = Disable inter-line overlapping
k	R/W	DBL_BUFFER_EN	0 = Disable double buffering feature 1 = Enable double buffering feature
l	R/W	CMDFIFO_SIZE_MODE_EN	0 = Disable CMDFIFO size change 1 = Enable CMDFIFO size change
m	R/W	AUTO_FF_DIS	0 = Enable auto-fast-fills 1 = Disable auto-fast-fills
n	R/W	AUTO_BLKWRT_DIS	0 = Enable auto-block writes 1 = Disable auto-block writes
o	R/W	ORed_INVLD_RB_CACHE	0 = invalidate the readback cache 1 = invalidate the readback cache (ORed with INVALIDATE_RB_CACHE pulse)
p	R/W	BLOCK_DBL_BUF	Blocks double-buffering of CRTC_OFFSET value if set.

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
	dd	cc	bb	aa	z	y	x	w	v	u	t	s	r	**q**			p	o	n	m	l	k	j	i	h	g	f	e	d	c	b	**a**

q	**R/W**	**HCLK_FB_SKEW**	**HCLK feedback skew adjustment for memory read data latching. This register only has an effect when MEM_LATCHL and MEM_LATCHU @ MEMCNTL = "01" (HCLK feedback) or "11" (XCLKb).** **000 = earliest** **111 = latest**
r	R/W	MCLK_START_EN	0 = Do not start MCLK during VBLANK 1 = Re-start MCLK during VBLANK when host write buffer is not empty Note: available only in rev A21
s	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
t	R/W	CMDFIFO_64EN	Control of GUI bus mastering path. Must be set to 1 if IDCT_EN bit is set to 1. 0 = cmdfifo2 and mcidctparser are not in the path (default) 1 = cmdfifo2 and mcidctparser are in the path Note: available from rev A22.
u	R/W	BM_FIX_DIS	0 = Enable bus master mode 1 = Disable bus master mode
v	R/W	Z_SWITCH_EN	0 = Do not flush the z-buffer 1 = Flush the z-buffer before accepting new data
w	R/W	FLUSH_HOST_WB	0 = Not flushing the host write buffer 1 = Flushing the host write buffer
x	R/W	HW_DEBUG_WRITE_MSK_FIX_DIS	0 = Enable write mask fix 1 = Disable write mask fix
y	R/W	Z_NO_WRITE_EN	0 = Not write to the memory when Z compare fail 1 = Write to memory when Z compare fail
z	R/W	DON'T_RST_PCLK	0 = Generate RSTPCLK when GEN_SOFT_RESET is set to '1' 1 = Not generate RSTPCLK when GEN_SOFT_RESET is set to '1'
aa	R/W	PM_D3_SUPPORT_EN	This bit is to be used in power management feature. 0 = Do not generate reset when going from D3 to D0 mode 1 = Generate internal reset when going from D3 to D0 mode

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
	dd	cc	bb	aa	z	y	x	w	v	u	t	s	r	**q**			p	o	n	m	l	k	j	i	h	g	f	e	d	c	b	**a**

bb	R/W	DON'T_RST_CHAREN	This bit is to be used in vga mode with panel connected only. 0 = Reset character clock with sequencer register index 0 (default) 1 = Do not reset character clock with sequencer register index 0 Note: available only in rev A21.
cc	R/W	HOSTRA_SET_EN	This bit is to block slave read coming in from HBIU during bus mastering. 0 = Disable the feature 1 = Enable the feature (default) Note: available only in rev. A21
dd	R/W	AUTOEXP_HORZ_FIX	This bit is to fix auto horizontal expansion in 40 column & to shadow hblank_end w/ LCD on. 0 = Disable the fix 1 = Enable the fix Note: available only in rev A21

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	Horizontal 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

Use this register for ASIC test purposes.

4.1.8 Configuration Registers

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 = 10) 00 = Reserved 01 = Reserved 10 = 2x8MB apertures 11 = 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	Disable VGA: (default = 0) 0 = Enable if CFG_VGA_EN@CONFIG_STAT0 = 1 1 = Disable

Description

This register configures 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*

.

CONFIG_CHIP_ID																												Offset: 0_38	
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	e		**d**			c			b								a														

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 (0x4) (TSMC)**
e	**R**	**CFG_CHIP_REV**	**ASIC Minor revision number, refer to the following table(chip_rev)**

Description

This register 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 Mobility in the PCI address 2 and CONFIG_CHIP_ID non-GUI register are as follows:

Table 4-4 Device ID

DEVICE ID	Description
4C4Dh (LM)	**AGP-133/BGA (AGP bus)**
4C52h (LR)	**PCI-33/BGA (PCI bus)**
4C4Eh (LN)	**AGP-133/BGA (reduced functionality)**
4C53h (LS)	**PCI-33/BGA (reduced functionality)**

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 table shows a list of ASIC IDs used to date:

Table 4-5 ASIC ID

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

Table 4-6 Chip Rev

Chip Rev	Description
0	**A11 and A12**
1	**A21 = all layer spin from A11 w/o P+/N+ in single port memory macro's** **A21 = A21- w/ PAI process** **A21 = all layer spin from A21-Aw/ P+/N+ in single/dual port memory macro's** **A21++ = A21+ w/ PAI process**
2 - 7	**Reserved**

See Also

mach64 Programmer's Guide:

- *Getting Started: mach64 Detection: Card Detection*

CONFIG_STAT0																															
BITS	31	30	29	28	27	26	25	24	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	CFG_MEM_TYPE	Select memory type: 000 = Disable memory access 001 = Reserved for basic DRAM 010 = Reserved for EDO 011 = Reserved for hyper page DRAM or EDO 100 = SDRAM (1:1) 101 = SGRAM (1:1) (default) 110 = SDRAM (2:1) (32-bit memory interface) 111 = Reserved
b	R/W	CFG_VGA_EN	0 = Disable VGA 1 = Enable VGA Default setting = strap setting
c	R/W	CFG_CLOCK_EN	Select control for GUI clock: (default = 0) 0 = clock controlled by GUI activity 1 = clock always on
d	R	MACROVISION_ENABLE	Enable macrovision 0 = Disable 1 = Enable
e	R	ARITHMOS_ENABLE	Enable arithmos 0 = Disable 1 = Enable

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*

For a complete description of the strapping scheme, refer to the RAGE Mobility Controller Specifications.

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
	b	a	

a	R/W	SUBSYS_DEV_ID (15:0)	PCI subsystem Device ID
b	R/W	SUBSYS_VEN_ID (15:0)	PCI subsystem Vendor ID

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
	i h g f e	d c b a	

a	R/W	AGPVCOGAIN	Set VCO gain 00 = Default
b	R/W	BUS_CLK_SEL	Select bus type 0 = Normal bus type (AGP with 352 BGA or PCI66 with 352 BGA) 1 = Reverse bus type (PCI33 with 352 BGA)
c	R/W	X1CLKSKEW	Adjustment for AGP X2 clock skew 00 = Default
d	R/W	AGPSKEW	Adjustment for AGP clock skew 00 = Default
e	R/W	ID_DISABLE	0 = Normal operation 1 = Not connected
f	R/W	BUS_TYPE	Select PCI bus signaling 0 = PCI 3.3 V signaling 1 = PCI 5 V signaling
g	R/W	VGA_DISABLE	Disable VGA 0 = Enable 1 = Disable
h	R/W	ENINTB	Disable interrupt 0 = Enable 1 = Disable
i	R/W	IDSEL	Set connection for IDSEL 0 = Connect IDSEL to AD16 1 = Connect IDSEL to AD17

4.1.9 Custom Macros Registers

CUSTOM_MACRO_CNTL																																
BITS		31	30	29	28	27	26	25	24	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**	**IDCT_FIFO_EXTSENSE**	**IDCT FIFO mode:** **hardware default = '0', BIOS should set '1'** **0 = Normal** **1 = Enable extended sense**
b	R/W	MEM_DQML_DRIVE	Boost drive strength of lower 4 DQM pins (0-3): 0 = Normal 1 = Slow Note: only available on Chip rev A21.
c	R/W	MEM_DQML_SLEWRATE	Slew rate control of lower 4 DQM pins (0-3): 0 = Normal 1 = Slow Note: only available on Chip rev A21.
d	R/W	GWBUF_FIFO_EXTSENSE	GUI write buffer FIFO mode 0 = Normal operation 1 = Enable extended sense
e	**R/W**	**RDRET_FIFO_EXTPRE**	**Read return FIFO precharge** **0 = Normal operation** **1 = Extended precharge**
f	**R/W**	**RDRET_FIFO_EXTDIS**	**Read return FIFO discharge** **0 = Normal operation** **1 = Extended discharge**
g	R/W	RDRET_FIFO_EXTSENSE	Read Return FIFO mode 0 = Normal operation 1 = Enable extended sense
h	**R/W**	**MEM_CS_SLEWRATE**	**Slew rate control of CS pins** **0 = Normal operation** **1 = Slow**
i	**R/W**	**MEM_MDA_SLEWRATE**	**Slew rate control of upper 32 MD pins not connected to BIOS (49, 52-55)** **0 = Normal operation** **1 = Slow**
j	**R/W**	**MEM_MDB_SLEWRATE**	**Slew rate control of pins connected to BIOS (32-48, 50-51, 56-63):** **0 = Normal operation** **1 = Slow**
k	**R/W**	**MEM_MDL_SLEWRATE**	**Slew rate control of low 32 MD pins (0-31)** **0 = Normal operation** **1 = Slow**

Cont'd	CUSTOM_MACRO_CNTL																															
BITS	31	30	29	28	27	26	25	24	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

l	R/W	MEM_MA_SLEWRATE	Slew rate control of MA pins 0 = Normal operation 1 = Slow
m	R/W	MEM_CNTL_SLEWRATE	Slew rate control of RAS, CAS, WE, DSF pins: (default = 0) 0 = Normal operation 1 = Slow
n	R/W	MEM_DQM_SLEWRATE	Slew rate control of DQM pins: (default = 0) 0 = Normal operation 1 = Slow

4.2 Accelerator CRTC and DAC Registers

4.2.1 Accelerator CRTC Registers

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 (*8 pixels). 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 *8-5*). In some cases, only part of the CRTC register is shadowed, as summarized below:

Table 4-7 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

Table 4-7 Shadowed VGA CRTC Registers Cont'd

Name	I/O	Index	Bits Shadowed
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
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).

Table 4-8 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 ‘1’ if written
	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

Auto-horizontal enable is used mainly for VGA compatibility when the LCD panel is connected. It allows the calculation of the horizontal expansion ratio based on the CRTC register parameters. The following table shows how the shadowed horizontal CRTC parameters are used:

Use of the Horizontal CRTC Parameters			
USE_SHADOW	**DONT_USE_ SHADOW**	**AUTO_HORZ_E N**	**Shadowed CRTC Registers Used**
1	0	?	VGA CRTC INDEX 0 (H_TOTAL) VGA CRTC INDEX 1 (H_DISP_END) VGA CRTC INDEX 2 (H_BLANK_START) VGA CRTC INDEX 3 (H_BLANK_END) BITS [4:0] VGA CRTC INDEX 4 (H_DE_SKEW) BITS [6:5] VGA CRTC INDEX 5 (H_SYNC_SKEW) BITS [6:5]
1	1	0	VGA CRTC INDEX 0 (H_TOTAL) VGA CRTC INDEX 2 (H_BLANK_START) VGA CRTC INDEX 3 (H_BLANK_END) BITS [4:0] VGA CRTC INDEX 4 (H_DE_SKEW) BITS [6:5] VGA CRTC INDEX 5 (H_SYNC_SKEW) BITS [6:5]
1	1	1	VGA CRTC INDEX 0 (H_TOTAL) VGA CRTC INDEX 3 (H_BLANK_END) BITS [4:0] VGA CRTC INDEX 5 (H_SYNC_SKEW) BITS [6:5]

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, the same address space will be used to access accelerator CRT registers. Registers necessary to define timing, hardware cursor, and CRC signature for the second CRT controller are duplicated, i.e.:

CRTC2_H_TOTAL_DISP	CRTC2_H_SYNC_STRT_WID
CRTC2_V_TOTAL_DISP	CRTC2_V_SYNC_STRT_WID
CRTC2_VLINE_CRNT_VLINE	CRTC2_OFF_PITCH
OVR2_WID_LEFT_RIGHT	OVR2_WID_TOP_BOTTOM
CUR2_CLR0	CUR2_CLR1
CUR2_OFFSET	CUR2_HORZ_VERT_POSN
CUR2_HORZ_VERT_OFF	CRC2_SIG

Note: To select the register set for the active display (i.e., CRT or LCD), set the bit CRTC_RW_SELECT in the LCD_GEN_CRTL register (refer to page *8-5*).

To select the register set for the first display, set CRTC_RW_SELECT@LCD_GEN_CRTL = 0.

To select the register set for the second display, set CRTC_RW_SELECT@LCD_GEN_CRTL = 1

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

Description

This register specifies the horizontal total and horizontal displayed parameters for the accelerator CRTC. This register is used for the first display path.

(*8) means the unit value (in decimal) contained in the bits represents 8 pixels.

Example: to set the CRTC for a total horizontal display of 640 pixels (i.e. a 640x480 display format), insert 50h (i.e., 80 decimal) into bits [8:0]; therefore, 80*8 = 640.

Usage

Use this register 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_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	CRTC2_H_TOTAL	Horizontal display total (*8 pixels)
b	R/W	CRTC2_H_DISP	Horizontal display end (*8 pixels)

Description

This register is similar to the CRTC_H_TOTAL_DISP register. This register is used for the second display path.

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 (*8 pixels)
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 (*8 pixels)
e	R/W	CRTC_H_SYNC_POL	Horizontal sync polarity (1 -> active low)

Description

This register specifies the horizontal sync attributes for the accelerator CRTC. This register is used for the first display path.

(*8 pixels) means the unit value (in decimal) contained in the bits represents 8 pixels. For an complete explanation, refer to CRTC_H_TOTAL_DISP (offset = 0_00).

Usage

Use this register 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
											e	d								c		b							a				

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

Description

This register is similar to the CRTC_H_SYNC_STRT_WID register. This register is used for the second display path.

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

This register specifies the vertical total and vertical displayed parameters for the accelerator CRTC. This register is used for the first display path. All vertical parameters are specified in lines.

Usage

Use this register 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
	b																					a										

a	R/W	CRTC2_V_TOTAL	Vertical total
b	R/W	CRTC2_V_DISP	Vertical display end

Description

This register is similar to the CRTC_V_TOTAL_DISP register. This register is used for the second display path.

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

This register specifies the vertical sync attributes for the accelerator CRTC. This register is used for the first display path. All vertical parameters are specified in lines.

Usage

Use this register 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_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	CRTC2_V_SYNC_STRT	Vertical sync start
b	R/W	CRTC2_V_SYNC_WID	Vertical sync width
a	R/W	CRTC2_V_SYNC_POL	Vertical sync polarity (1 -> active low)

Description

This register is similar to the CRTC_V_SYNC_STRT_WID register. This register is used for the second display path.

CRTC_VLINE_CRNT_VLINE																											Offset: 0_04
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						b																a										

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

Description

The CRTC_VLINE field determines the line at which a CRTC interrupt will be triggered if the interrupts are enabled. The CRTC_CRNT_VLINE field is read-only and it returns the current value of the accelerator CRTC vertical line counter. This register is used for the first display path.

Usage

Use this register only in applications that require synchronization to the CRTC, such as smooth animation.

See Also

CRTC_INT_CNTL on *4-53*

mach64 Programmer's Guide:

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

CRTC2_VLINE_CRNT_VLINE																											Offset: 0_04
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						b																a										

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

Description

This register is similar to the CRTC_VLINE_CRNT_VLINE register. This register is used for the second display path.

CRTC_OFF_PITCH																												Offset: 0_05	
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	c												b	a																		

a	R/W	CRTC_OFFSET	Display address offset in qword of primary display path
b	**R**	**CRTC_OFFSET_LOCK**	**CRTC display address offset status bit 0 = Indicates CRTC_OFFSET has been used 1 = Indicates CRTC_OFFSET has not been used**
c	**R/W**	**CRTC_PITCH**	**Display pitch (*8 pixels) of primary display path**

Description

This register specifies the starting memory offset and pitch of the accelerator CRTC. This register is used for the first display path. The pitch value must correspond exactly to the destination draw engine pitch for visible screen memory. If the memory boundary is enabled, the offset must be set to a value above or equal to the boundary offset.

(*8 pixels) means the unit value (in decimal) contained in the bits represents 8 pixels. For an complete explanation, refer to CRTC_H_TOTAL_DISP (offset = 0_00).

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 *4-22*

SRC_OFF_PITCH on *5-26*

DST_OFF_PITCH on *5-10*

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
		c											b	a																		

a	R/W	CRTC2_OFFSET	Display address offset in terms of 64 bit words
b	**R**	**CRTC2_OFFSET_LOCK**	**CRTC2 display address offset status bit.** **0 = Indicates address offset has been used** **1 = Indicates address offset has not been used**
c	**R/W**	**CRTC2_PITCH**	**Display pitch (*8 pixels)**

Description

This register is similar to the CRTC_OFF_PITCH register. This register defines the start address for the second display. This register is used for the second display path.

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
		F	**E**	**D**	**C**	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 *	Enable vertical blank interrupt (active high): (default = 0) 0 = Disable 1 = Enable
c	R	CRTC_VBLANK_INT *	Vertical blank interrupt (active high)
	W	CRTC_VBLANK_INT_AK *	Vertical blank acknowledge (to clear interrupt, write ‘1’)
d	R/W	CRTC_VLINE_INT_EN *	Enable vertical line interrupt (active high): (default = 0) 0 = Disable 1 = Enable
e	R	CRTC_VLINE_INT *	Vertical line interrupt (active high)
	W	CRTC_VLINE_INT_AK *	Vertical line interrupt acknowledge (to clear interrupt, write ‘1’)
f	R	CRTC_VLINE_SYNC	Vertical line sync 0 = Even scan line 1 = Odd scan line
g	R	CRTC_FRAME	Interlace odd/even frame 0 = Even frame 1 = Odd frame

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
	F	**E**	**D**	**C**	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

h	**R/W**	**SNAPSHOT_INT_EN**	**Enable snapshot interrupt: (default = 0)** **0 = Disable** **1 = Enable**
i	**R**	**SNAPSHOT_INT**	**0 = No snapshot taken** **1 = Snapshot taken**
	W	**SNAPSHOT_INT_AK**	**1 = Clear snapshot interrupt**
j	R/W	I2C_INT_EN	Enable I2C interrupt: (default = 0) 0 = Disable 1 = Enable
k	R	I2C_INT	0 = No I2C interrupt 1 = I2C interrupt took place
	W	I2C_INT_AK	1 = Clear I2C interrupt
l	**R**	**CRTC2_VBLANK**	**Vertical blank of the second CRT**
m	**R/W**	**CRTC2_VBLANK_INT_EN ***	**Enable vertical blank interrupt for second CRT (active high): (default = 0)** **0 = Disable** **1 = Enable**
n	**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 (to clear interrupt, write '1')**
o	**R/W**	**CRTC2_VLINE_INT_EN ***	**Enable vertical line interrupt for the second CRT (active high): (default = 0)** **0 = Disable** **1 = Enable**
p	**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 (to clear interrupt, write '1')s**
q	R/W	CUPBUF0_INT_EN	Enable continuous capture buffer 0 interrupt: (default = 0) 0 = Disable 1 = Enable
r	R	CUPBUF0_INT *	Continuous capture buffer 0 interrupt (active high)
	W	CUPBUF0_INT_AK *	Continuous capture buffer 0 (to clear interrupt, write '1')
s	R/W	CUPBUF1_INT_EN	Enable continuous capture buffer 1 interrupt: (default = 0) 0 = Disable 1 = 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
	F	**E**	**D**	**C**	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

t	R	CUPBUF1_INT *	Continuous capture buffer 1 interrupt (active high)
	W	CUPBUF1_INT_AK *	Continuous capture buffer 1 acknowledge (to clear interrupt, write '1')
u	R/W	OVERLAY_EOF_INT_EN	Enable overlay end-of-frame interrupt: (default = 0) 0 = Disable 1 = Enable
v	R	OVERLAY_EOF_INT *	Overlay end-of-frame interrupt (active high)
	W	OVERLAY_EOF_INT_AK *	Overlay end-of-frame acknowledge (to clear interrupt, write '1')
w	R/W	ONESHOT_CAP_INT_EN	Enable one-shot host capture complete interrupt: (default = 0) 0 = Disable 1 = Enable
x	R	ONESHOT_CAP_INT *	One-shot host capture complete interrupt (active high)
	W	ONESHOT_CAP_INT_AK *	One-shot host capture complete acknowledge (to clear interrupt, write '1')
y	R/W	BUSMASTER_EOL_INT_EN	Enable bus master end of system list interrupt: (default = 0) 0 = Disable 1 = Enable
z	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 (to clear interrupt, writer '1')
A	R/W	GP_INT_EN	Enable general purpose I/O interrupt: (default = 0) 0 = Disable 1 = Enable
B	R	GP_INT *	General purpose I/O interrupt
	W	GP_INT_AK *	General purpose I/O acknowledge (to clear interrupt, write '1')
C	**R**	**CRTC2_VLINE_SYNC**	**Vertical line sync of the second CRT** **0 = Even scan line** **1 = Odd scan line**
D	**R/W**	**SNAPSHOT2_INT_EN ***	**Enable snapshot interrupt (active high): (default = 0)** **0 = Disable** **1 = 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
	F	**E**	**D**	**C**	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

E	**R**	**SNAPSHOT2_INT ***	**Snapshot interrupt (active high)**
	W	**SNAPSHOT2_INT_AK ***	**Snapshot interrupt acknowledge (to clear interrupt, write '1')**
F	R	VBLANK_BIT2_INT *	Secondary VBLZNK bit interrupt
	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 of interrupts should **not** be attempted in a single write.

Description

This register 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

Use this register to achieve smooth animation, or reduce flickering and tearing.

See Also

CRTC_VLINE_CRNT_VLINE on *4-51*

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
	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/W	CRTC_DBL_SCAN_EN	Enable double scan: (default = 0) 0 = Disable 1= Enable

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
		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

b	R/W	CRTC_INTERLACE_EN	Enable Interlace: (default = 0) 0 = Disable 1= Enable
c	R/W	CRTC_HSYNC_DIS	Disable horizontal sync output: (default = 0) 0 = Enable 1= Disable
d	R/W	CRTC_VSYNC_DIS	Disable vertical sync output: (default = 0) 0 = Enable 1= Disable
e	R/W	CRTC_CSYNC_EN	Enable composite sync on horizontal sync output: (default = 0) 0 = Disable 1= Enable
f	R/W	CRTC2_DBL_SCAN_EN	Enable double scan on the secondary display: (default = 0) 0 = Disable 1= Enable
g	R/W	CRTC_DISPLAY_DIS	Disable the display (forcing the blanking signal to be active): (default = 0) 0 = Enable 1= Disable
h	R/W	CRTC_VGA_XOVERSCAN	Enable overscan in VGA mode: (default = 0) 0 = Disable 1 = Enable
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, OR 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	Enable reversing pixel order within each memory byte in 4 bpp mode. 0 = Pixel order from MSNibble to LSNibble. 1 = Pixel order from LSNibble to MSNibble.
k	**R/W**	**CRTC_VSYNC_INT_EN**	**Enable vertical sync interrupt of primary display: (default = 0)** **0 = Disable** **1 = Enable**

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
	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

l	R	CRTC_VSYNC_INT	Vertical sync started since last cleared 0 = No event 1 = Event has occurred, interrupting if enabled
	W	CRTC_VSYNC_INT_AK	To clear the status of CRTC_VSYNC_INT, write '1'
m	R/W	CRTC2_VSYNC_INT_EN	Enable vertical sync interrupt of second display: (default = 0) 0 = Disable 1 = Enable
n	R	CRTC2_VSYNC_INT	Vertical sync started since last cleared 0 = No event 1 = Event has occurred, interrupting if enabled
	W	CRTC2_VSYNC_INT	To clear the status of CRTC2_VSYNC_INT, write '1'
o	R/W	HVSYNC_IO_DRIVE	Set output drive strength for the HSYNC and VSYNC pins: (default = 1) 0 = No boost 1 = Boost
p	R/W	CRTC2_PIX_WIDTH	Set second display pixel width 000 = Disable second CRT 001 = Reserved 010 = 8 bpp 011 = 15 bpp (5,5,5) 100 = 16 bpp (5,6,5) 101 = 24 bpp (8,8,8) 110 = 32 bpp (a,8,8,8) 111 = YUV422
q	R/W	VGA_128KAP_PAGING	Enable extended aperture paging in 128K VGA aperture mode: (default = 0) 0 = Disable 1 = Enable
r	R/W	CRTC2_ENABLE	Enable second CRT controller: (default = 0) 0 = Reset CRTC2 1 = Enable
s	R/W	CRTC_LOCK_REGS	Lock extended CRTC registers from being written to: 0 = Unlock (read and write) 1 = Lock (read only)
t	R/W	CRTC_SYNC_TRISTATE	0 = Normal 1 = Tri-state hsync and vsync
u	R/W	CRTC_EXT_DISP_EN	Enable extended display mode: (default = 0) 0 = Disable (i.e. use VGA display) 1 = Enable

Cont'd		CRTC_GEN_CNTL																														
BITS		31	30	29	28	27	26	25	24	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/W	CRTC_ENABLE											Enable CRT controller: (default = 0) 0 = Disable (i.e. resets CRTC) 1 = Enable CRTC																				
w	R/W	CRTC_DISP_REQ_ENB											Enable display requests: (default = 1) 0 = Enable 1 = Disable																				
x	R/W	VGA_ATI_LINEAR											Enable linear addressing through VGA aperture: (default = 0) 0 = Disable 1 = Enable																				
y	R/W	CRTC_VSYNC_FALL_EDGE											Select VSYNC edge to start frame sequence 0 = Rising edge of VSYNC 1 = Falling edge of VSYNC																				
z	R/W	VGA_TEXT_132											Select extended text mode (linear address 132 column text mode) 1 = Active 0 = Inactive																				
A	R/W	VGA_XCRT_CNT_EN											Enable extended CRTC display address counter (active high)																				
B	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

Use this register only for mode switching. It 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 colors written 01 = Red color index written (await green/blue) 10 = Green color index written (await blue) 11 = Reserved
c	R/W	CRT_TRAP_EN	Enable VGA CRT register trapping: (default = 0) 0 = Disable 1 = Enable

Usage

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

4.2.2 Overscan Registers

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 Mobilty. 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 for the first display path.

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 *4-65*

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 path
b	R/W	OVR2_CLR_G	Green overscan color for second display path

Cont'd	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															

c	R/W	OVR2_CLR_R	Red overscan color for second display path

Description

This register is similar to the OVR_CLR register. This register is only used for the second display path.

	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 (8*pixels)
b	R/W	OVR_WID_RIGHT	Right overscan width (8*pixels)

Description

This register specifies the left and right overscan widths in characters for the first display path.

(*8 pixels) means the unit value (in decimal) contained in the bits represents 8 pixels.

Example: to set the left overscan width to 16 pixels, insert 02h (i.e., 2 decimal) into bits [5:0]; therefore, 2*8 = 16.

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 *4-63*

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
											b																a					

a	R/W	OVR2_WID_LEFT	Left overscan width (8*pixels)
b	R/W	OVR2_WID_RIGHT	Right overscan width (8*pixels)

Description

This register is similar to the OVR_WID_LEFT_RIGHT register. This register is only used only for the 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

This register specifies the top and bottom overscan widths in lines for the first display path.

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 *4-62*

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
	b																							a								

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

Description

This register is similar to the OVR_WID_TOP_BOTTOM register. This register is only used for the second display path.

4.2.3 Hardware Cursor Registers

The RAGE Mobility supports independent hardware cursor and hardware icon on both displays. Only the registers for the hardware cursor are shown here, the registers for the hardware icon are contained with the LCD indexed registers.

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.

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.

4.2.3.1 First Display Hardware Cursor Registers

The following registers are used to control the hardware cursor for the first display path.

Table 4-9 Cursor Pixel Color Definition

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)

	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.

Usage

Use this register 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

This register is similar to the CUR_CLR0 register, but used for color 1.

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

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

Usage

Use this register 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 horizontal position**
b	**R/W**	**CUR_VERT_POSN**	**Cursor vertical position**

Description

This register 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

Use this register 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
											b																a				

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

Description

This register 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 minus size), and each icon offset should be set such that offset = (128 minus size).

Usage

Use this register when defining the hardware cursor attributes.

See Also

mach64 Programmer's Guide:

- *Engine Operations: Miscellaneous Operations: Hardware Cursor*

4.2.4 Second Display Hardware Cursor Registers

Use the following registers to control the hardware cursor for the second display path.

CUR2_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
Mobility	d								c								b								a							

a	R/W	CUR2_CLR0_8	Cursor color 0 INDEX, output on feature connector in 4 and 8 bpp
b	R/W	CUR2_CLR0_B	Blue cursor color 0, to internal DAC
c	R/W	CUR2_CLR0_G	Green cursor color 0, to internal DAC
d	R/W	CUR2_CLR0_R	Red cursor color 0, to internal DAC

Description

This register is similar to the CUR_CLR0 register. This register is used for the second display hardware cursor. It uses the same address as the CUR_CLR0 register.

CUR2_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
Mobility	d								c								b								a							

a	R/W	CUR2_CLR1_8	Cursor color 1 INDEX, output on feature connector in 4 & 8 bpp
b	R/W	CUR2_CLR1_B	Blue cursor color 1, to internal DAC
c	R/W	CUR2_CLR1_G	Green cursor color 1, to internal DAC
d	R/W	CUR2_CLR1_R	Red cursor color 1, to internal DAC

Description

This register is similar to the CUR_CLR1 register. This register is used for the second display hardware cursor. It uses the same address as the CUR_CLR1 register.

CUR2_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
Mobility													a																			

a	R/W	CUR2_OFFSET	Cursor address offset in terms of 64 bit words

Description

This register is similar to the CUR_OFFSET register. This register is used for the second display hardware cursor. It uses the same address as the CUR_OFFSET register.

CUR2_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
Mobility							b																a									

a	R/W	CUR2_HORZ_POSN	Cursor horizontal position
b	R/W	CUR2_VERT_POSN	Cursor vertical position

Description

This register is similar to the CUR_HORZ_VERT_POSN register. This register is used for the second display hardware cursor. It uses the same address as the CUR_HORZ_VERT_POSN register.

CUR2_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
Mobility											b																a					

a	R/W	CUR2_HORZ_OFF	Cursor horizontal offset
b	R/W	CUR2_VERT_OFF	Cursor vertical offset

Description

This register is similar to the CUR_HORZ_VERT_OFF register. This register is used for the second display hardware cursor. It uses the same address as the CUR_HORZ_VERT_OFF register.

4.2.5 GenLocking (CRT-Sync to Video) Registers

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
	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
												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

This register 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
																							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. Refer to CRTC_INT_CNTL [8:7] for the snapshot interrupt specification

See also:

CRTC_INT_CNTL register (*4-53*), 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

Use this register to calculate the total number of Video-in-Field frames. Use the following equation:

*Total VIF frame count = (N_VIF_COUNT_VAL * USNAPSHOT_VIF_COUNT) + 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		
	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	R/W	AUTO_SNAPSHOT2_TAKEN	0 = writing '0' enables both auto and manual snapshot taking, and clears SNAPSHOT2_F_COUNT and SNAPSHOT2_VIF_COUNT counters (W). 1 = indicates that a snapshot for second CRT has been taken (R)
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)

4.2.6 Clock Control Registers

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

This register selects 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*

4.2.7 PLL Control Registers

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 PLLs. Gain bits affect lock and jitter of PLLs. These registers should only be adjusted by the BIOS.
- Oscillator is always turned on regardless of whether 14.3 or 29.5 MHz crystal 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 PLLs.

The table below is a listing of the PLL registers with their default values.

Table 4-10 PLL Register Default Values

Address	Register Name	Default	Default for Scan
0	MPLL_CNTL	ACh	ACh
1	VPLL_CNTL	ACh	ACh
2	PLL_REF_DIV	24h	24h
3	PLL_GEN_CNTL	DFh	FFh
4	MCLK_FB_DIV	F6h	F6h
5	PLL_VCLK_CNTL	04h	05h
6	VCLK_POST_DIV	00h	00h
7	VCLK0_FB_DIV	FDh	FDh
8	VCLK1_FB_DIV	8Eh	8Eh
9	VCLK2_FB_DIV	9Eh	9Eh
10	VCLK3_FB_DIV	65h	65h
11	PLL_EXT_CNTL	05h	05h
12	DLL1_CNTL	00h	00h
13	VFC_CNTL	00h	00h
14	PLL_TEST_CNTL	00h	00h
15	PLL_TEST_COUNT	00h	00h
16	LVDS_CNTL0	06h	06h
17	LVDS_CNTL1	CFh	CFh
18	AGP1_CNTL	80h	80h
19	AGP2_CNTL	00h	00h
20	DLL2_CNTL	10h	10h
21	SCLK_FB_DIV	F6h	F6h
22	SPRG	ACh	ACh
23	SPLL_CNTL	53h	53h
24	APLL_STRAPS	-- (straps)	-- (straps)
25	EXT_VPLL_CNTL	80h	80h

Table 4-10 PLL Register Default Values Cont'd

Address	Register Name	Default	Default for Scan
26	EXT_VPLL_REF_DIV	24h	24h
27	EXT_VPLL_FB_DIV	FDh	FDh
28	EXT_VPLL_MSB	00h	00h
29	HTOTAL_CNTL	00h	00h
30	BYTE_CLK_CNTL	00h	00h
31	TV_PLL_CNTL1	02h	02h
32	TV_PLL_CNTL2	06h	06h
33	TV_PLL_CNTL	ACh	ACh
34	EXT_TV_PLL	06h	2Eh
35	V2PLL_CNTL	ACh	ACh
36	PLL_V2CLK_CNTL	14h	15h
37	EXT_V2PLL_REF_DIV	24h	24h
38	EXT_V2PLL_FB_DIV	FDh	FDh
39	EXT_V2PLL_MSB	00h	00h
40	HTOTAL2_CNTL	00h	00h
41	PLL_YCLK_CNTL	05h	04h
42	PM_DYN_CLK_CNTL	55h	55h

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

This register sets the controls for the 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

This register sets the controls for the 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

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

c	R/W	OSC_EN	1 = Enable oscillator
d	R/W	EXT_CLK_EN	1 = Force DCLK pin output to tri-state
e	R/W	MCLK_SRC_SEL	Select MCLK (MCLK is GUI and 3D Engine clock) 000 = MCLK set to PLLMCLK (MPLL primary output) 001 = MCLK set to PLLMCLK/2 010 = MCLK set to PLLMCLK/4 011 = MCLK set to PLLMCLK/8 100 = MCLK set to PLLMCLK/3 101 = MCLK set to CPUCLK 110 = MCLK set to SCLK HCLK (direct, no DLL) 111 = MCLK set to XTALIN
f	R/W	DLL_PWDN	0 = Enable DLL 1 = Power down DLL (for full powerdown, set DLL_GAIN = 10)

Description

This register sets up the 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

This register sets up the 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	Select VLCK_SRC (VLC is the pixel clock) 00 = VCLK_SRC set to CPUCLK 01 = VCLK_SRC set to DCLK 10 = VCLK_SRC set to GIO(1) 11 = VCLK_SRC set to 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	Set ECP (ECP is the Scaler/Ovelay clock) 00 = ECP set to VCLK 01 = ECP set to VCLK/2 10 = ECP set to VCLK/4 11 = Reserved
e		(scratch)	Reserved for future use

Description

This register sets up the 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

This registers sets up the 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 *4-73*) 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

This register sets up the VPLL feedback divider. Selection made by VGA_CKSEL@GENM0 in VGA mode (see page *7-23*) or by CLOCK_SEL@ CLOCK_CNTL (see page *4-73*) 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	Select XCLK (XCLK is the memory interface clock) 000 = XCLK set to PLLMCLK (MPLL primary output) 001 = XCLK set to PLLMCLK/2 010 = XCLK set to PLLMCLK/4 011 = XCLK set to PLLMCLK/8 100 = XCLK set to PLLMCLK/3 101 = XCLK set to CPUCLK 110 = XCLK set to HCLK (direct, no DLL) 111 = XCLK set to DLL_CLK						
b	R/W	MFB_TIMES_4_2b	Select ratio of MCLK_FB_DIV to effective feedback value 0 = PLLMCLK feedback set to 2 * MCLK_FB_DIV 1 = PLLMCLK feedback set to 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

This register sets up the extended control of XCLK & VCLK (default = 05h).

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

a	R/W	DLL_REF_SRC	Select source for DLL_REF_CLK 00 = DLL_REF_CLK set to XCLK 01 = DLL_REF_CLK set to HCLK input 10 = DLL_REF_CLK set to not YCLK 11 = Reserved
b	**R/W**	**DLL_FB_SEL**	**Selection of source for DLL_FB_CLK** **00 = DLL_FB_CLK set to HCLK input** **01 = DLL_FB_CLK set to XCLK** **10 = DLL_FB_CLK set to internal feedback** **11 = Reserved**
c	R/W	DLL_RESET	DLL reset control. DLL resets on rising edge of this signal if DLL_REF_CLK is running
d	R/W	HCLK_OUT_EN	0 = HCLK forced to tri-state 1 = HCLK output enabled

Description

This register sets up the controls for 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	Select DCLK for VGA Modes: 0,1,4,5,D,13 (no affect in other display modes) 0 = DCLK set to 2 x VGA default 1 = DCLK set to VGA default
c	R/W	VFC_MULT_EN	Select true-color mode (no affect in VGA, 4bpp or 8bpp) 0 = Single clock VFC (DCLK set to VCLK) 1 = Multi-clock VFC (DCLK set by pixel depth and VCLK post divider)

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

d	R/W	VFC_DELAY	Adjustment for PIXEL and BLANKB hold 00 = Least delay 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

This register sets up the controls for the 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
d	R/W	ANALOG_MON_EN	Control PLL and Bandgap analog test mode 0 = Disable 1 = Enable

Description

This register sets up the 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

This register is used for ASIC production testing only. Do not use this register.

LVDSPLL_CNTL0								Addr: 16
BITS	7	6	5	4	3	2	1	0
	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 (default = 011)
c	R/W	LVDS_TEST_MODE	Define testmode for LVDS

LVDSPLL_CNTL1								Addr: 17
BITS	7	6	5	4	3	2	1	0
	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 = 001)
d	R/W	LPLL_CP_GAIN	CP gain control for LVDS PLL (default = 101)

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

a	R/W	X1_CLOCK_SKEW	Adjustment for AGP X1 phase (default = 000)
b	R/W	X2_CLOCK_SKEW	Adjustment for AGP X2 phase (default = 000)
c	R/W	PUMP_GAIN	AGP PLL charge-pump gain setting

Description

Use X1 and X2_CLOCK_SKEW to phase shift X1 clock (to roughly 0.5 ns step) with respect to X2 clock, and vice-versa.

Usage

This register should not be changed from the 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	Enable AGP PLL test mode: (default = 0) 0 = Disable 0 = Enable
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 and Bandgap analog test

Description

This register is used to test AGP PLL in ASIC production. Do not use this register.

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

a	R/W	DLL_SKEW	DLL reference skew control (default = 111 for minimum skew)
b	R/W	DLL_RANGE	DLL range control 10 = 80 - 110 MHz 11 = 110 - 150 MHz
c	R/W	DLL_FB_SKEW	DLL feedback skew control (default = 111 for minimum skew)

Description

This register sets up the 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

This register sets up the 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	Select clock for different divide downs of SCLK from secondary PLL 000 = SCLK set to PLLSCLK 001 = SCLK set to PLLSCLK/2 010 = SCLK set to PLLSCLK/4 011 = SCLK set to PLLSCLK/8 100 -111 = SCLK set to 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
	b				a			

a	R	APLL_SKEWS	[1:0] AGP skew control, provide a mean to compensate the input delay at the Reference Clock. Strap from MA(2:1) STRAP setting / Internal Setting (mapped to) / SKEW (ns) 00 / 011 / -0.221 01 / 100 / 0.14 10 / 010 / -0.596 11 / 101 / 0.51 [3:2] Inter-clock skew control, provide a mean to compensate the delay between the AGP X1 and AGP X2 Mode clock. Output is on AGP 1X. Strap from MA(4:3) STRAP setting / Internal Setting (mapped to) / SKEW (ns) 00 / 111 / 0.0 01 / 110 / 0.33 10 / 101 / -0.66 11 / 100 / 0.99
b	R	FILL_GAIN	VCO filter gain control, strap from MA(11:10) STRAP Setting / Register Value (internal setting of AGP PLL) 00 / 01 01 / 00 10 / 11 11 / 10

Description

This register sets up the 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 set to XTALIN 01 = VPLL_REF set to PLLMCLK/2 10 = VPLL_REF set to 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	Select reference divider value: (default = 1) 0 = Use reference divider value programmed in PLL_REF_DIV 1 = Use reference divider value programmed in EXT_VPLL_REF_DIV

Description

This register controls the 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

This register sets up the most significant bits (msb) 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

This register 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 set to GPIO(11) input, no change in VCLK_POST_DIV 01 = BYTE_CLK set to PLLVCLK/2, multiply VCLK_POST_DIV by 2 10 = BYTE_CLK set to PLLVCLK/3, multiply VCLK_POST_DIV by 3 11 = BYTE_CLK set to PLLVCLK/4, multiply VCLK_POST_DIV by 4

Cont'd

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

c	R/W	BYTE_CLK	0 = BYTE_CLK from ImpacTV to graphics controller 1 = Forces BYTE_CLK to be muxed out GPIO(111) to ImpacTV

Description

This register controls the 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
	a							

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

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

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

TVPLL_CNTL								Addr: 33
BITS	7	6	5	4	3	2	1	0
	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:

This register sets up the controls for the TVPLL analog macro (default = ADh).

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

a	R/W	TV_N_BIT9	Bit 9 of TV PLL feedback divider TV_N
b	R/W	TV_PLL_RST	1 = TV_PLL reset
c	R/W	TV_PLL_SLEEP	1 = TV_PLL power down
d	R/W	TVCLK_SRC_SEL	00 = CPUCLK 01 = XTALIN 10 = reserved 11 = TVPLLCLK
e	R/W	VCLKTV_SRC_SEL	000 = CPUCLK (inverted) 001 = DCLK 010 = VCLK 011 = VCLK2 100 = VCLK/2 101 = V2CLK/2 110 = Reserved 111 = Reserved

Description:

This register sets up the additional controls to TVPLL macro (default = 06h).

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

a	R/W	V2PLL_PC_GAIN	V2PLL 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:

This register sets up the controls to V2PLL analog macro (default = CCh).

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

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

Description:

This register sets up the second pixel clock control (default = 14h).

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

a	R/W	EXT_V2PLL_REF_DIV	EXT_V2PLL_REF_DIV bits 9:8
b	R/W	EXT_V2PLL_UPDATE	Controls transfer of EXT_V2PLL_REF_DIV and EXT_V2PLL_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_V2PLL_REF_DIV or EXT_V2PLL_FB_DIV
c	R/W	EXT_V2PLL_FB_DIV	EXT_V2PLL_FB_DIV bits 10:8

Description:

This register sets up the most significant bits (msb) 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
	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
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
c	R/W	EXT_V2PLL_INSYNC	Controls when EXT_V2PLL_UPDATE action occurs: 0 = Do atomic update ASAP 1 = Wait for vertical sync before atomic update

Description:

This register controls the 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).

PLL_YCLK_CNTL							Addr: 41	
BITS	7	6	5	4	3	2	1	0
Mobility	**c**		**b**			**a**		

a	**R/W**	**YCLK_SRC_SEL**	**YCLK is the memory interface clock when running 2:1 mode** **000 = YCLK set to PLLMCLK** **001 = YCLK set to PLLMCLK/2** **010 = YCLK set to PLLMCLK/4** **011 = YCLK set to PLLMCLK/8** **100 = YCLK set to SSIN** **101 = YCLK set to CPUCLK** **110 = YCLK set to HCLK (direct, no DLL)** **111 = YCLK set to DLL_CLK**

Cont'd	PLL_YCLK_CNTL					Addr: 41		
BITS	7	6	5	4	3	2	1	0
Mobility	c		b			a		

b	R/W	HCLK_SEL	Select function for HCLK pin (default = 4) 000 = HCLK set to XCLK 001 = HCLK set to not XCLK 010 = HCLK set to not YCLK 011 = HCLK set to YCLK 100 = HCLKset to DLL_CLK (default) 101 = Reserved 110 = Reserved 111 = HCLK set to YCLK/2
c	R/W	HCLK_REC	HCLK receiver selection 0 = Hysteresis receiver 1 = Differential receiver

	PM_DYN_CLK_CNTL					Addr: 42		
BITS	7	6	5	4	3	2	1	0
Mobility	d	c			b	a		

a	R/W	PM_XCLK_SRC_SEL	PM_XCLK is the slower memory interface clock in the dynamic clock speed power management mode 000 = PM_XCLK set to PLLMCLK (MPLL primary output) 001 = PM_XCLK set to PLLMCLK/2 010 = PM_XCLK set to PLLMCLK/4 011 = PM_XCLK set to PLLMCLK/8 100 = PM_XCLK set to PLLMCLK/3 101 = PM_XCLK set to CPUCLK 110 = PM_XCLK set to HCLK (direct, no DLL) 111 = PM_XCLK set to DLL_CLK Note: This is not a separate mux. The select for XCLK_SRC will be dynamically chosen as XCLK_SRC_SEL or PM_XCLK_SRC_SEL depending on the memory requester activity and the programming of POWER_MANAGEMENT_2 registers
b	R/W	PWRMAN_CLK_SEL	Set power management clock: (default = 0) 0 = BCLK_PM = BCLK 1 = BCLK_PM = XTALIN (if XTALIN_ALWAYS_ONb = 0)

Cont'd	PM_DYN_CLK_CNTL							Addr: 42
BITS	7	6	5	4	3	2	1	0
Mobility	d	c			b	a		

c	R/W	PM_YCLK_SRC_SEL	PM_YCLK is the slower memory interface clock when running 2:1 mode in the dynamic clock speed power management mode 000 = PM_YCLK set to PLLMCLK 001 = PM_YCLK set to PLLMCLK/2 010 = PM_YCLK set to PLLMCLK/4 011 = PM_YCLK set to PLLMCLK/8 100 = PM_YCLK set to SSIN 101 = PM_YCLK set to CPUCLK 110 = PM_YCLK set to HCLK (direct, no DLL) 111 = PM_YCLK set to DLL_CLK Note: This is not a separate mux; the select for YCLK_SRC will be dynamically chosen as YCLK_SRC_SEL or PM_YCLK_SRC_SEL depending on the memory requester activity and the programming of POWER_MANAGEMENT_2 registers
d	R/W	SS_EN	Enable spread spectrum clocking Allows to broadband the VCLK source frequency for reducing measured EMI

Description:

This register sets up the control of PM_XCLK and PM_YCLK for dynamic memory clock speed power management (default 55h).

4.2.8 DAC Control Registers

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.
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

This register 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

Use these registers to reprogram the DAC look up table (LUT).

See Also

DAC_CNTL on *4-100*

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

a	R/W	DAC_RANGE_CNTL	DAC output standard 00 = PAL 01 = NTSC 10 = PS2 11 = RS343
b	R/W	DAC_BLANKING	0 = 0 IRE blanking pedestal 1 = Enable 7.5 IRE blanking pedestal
c	R/W	DAC_CMP_DISABLE	Disable DAC comparators: (default = 0) 0 = Enable 1 = Disable (power down DAC comparators)
d	R/W	DAC1_CLK_SEL	Select data source for DAC1 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	Enable snooping of primary palette writes: (default = 0) 0 = Disabled 1 = Enable (everything written to primary palette will be written to secondary palette as well)
g	R	DAC_CMP_OUTPUT	DAC comparator output 0 = At least one comparator > 0.42 V 1 = All three comparators < 0.28 V
h	R/W	DAC_8BIT_EN	Select 6- or 8-bit DAC operation 0 = 6-bit 1 = 8-bit
i	**R**	**CRT_SENSE**	**This bit is read only** **0 = CRT is not connected** **1 = CRT is connected**
j	**R/W**	**CRT_DETECTION_ON**	**This bit is to enable hardware monitor detection circuitry**
k	**R/W**	**DAC_VGA_ADR_EN**	**Enable addressing of the DAC at the VGA IO DAC address when CRTC_EXT_DISP_EN is a '1'**
l	**R/W**	**DAC_FEA_CON_EN**	**Enable feature connector signal outputs**
m	**R/W**	**DAC_PDWN**	**Power down internal DAC (DAC macro only)**

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

n	**R**	**DAC_TYPE**	**Select DAC type** **000 = Internal DAC, 18-bit palette, no gamma correction** **001 = Internal DAC, 24-bit palette, gamma correction** **010 - 111 = Reserved**

Description

This register 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.

To enable CRT monitor auto-detection through hardware, set CRT_DETECTION_ON = 1.

For manual detection of the CRT monitor through software, set MONITOR_DET_EN@LCD_MISC_CNTL = 1.

To properly use the bits for selecting either manual- or auto-detection of the CRT monitor, set the following bits to their proper values:

FORCE_DAC_DATA_SEL@LCD_MISC_CNTL

FORCE_DAC_DATA@LCD_MISC_CNTL

Usage

This register is used only for mode switching. Only the adapter BIOS touches this register.

See Also

mach64 Programmer's Guide:

- *Advanced Topics: DAC Programming*

This page intentionally left blank.

Chapter 5
GUI Draw Engine

This chapter describes the registers used for configuring the GUI draw engine functions. For an explanation of the notations used to describe the registers, refer to Chapter 1, section *1.3.3*.

Table 5-1 Register Class

5.1 Draw Engine Trajectory Registers

This chapter describes the GUI engine registers. Operations are initiated implicitly 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 monochrome pattern source will be expanded and rotated properly. Note that Bresenham line drawing operations are not generally supported in the partial 24 bpp mode.

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

5.1.1 Destination Trajectory Registers

This register has two names—DST_BRES_DEC or LEAD_BRES_DEC

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

a	R/W	DST_BRES_DEC	Bresenham decrement for line and trapezoid leading edge

Description

This register 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

Use this register for line draw or trapezoid draw operations. It is aliased as LEAD_BRES_DEC.

See Also

DST_BRES_ERR on *5-3*

DST_BRES_INC on *5-3*

DST_BRES_LNTH on *5-4*

mach64 Programmer's Guide:

- *Engine Operations: Background Information: Trajectories: Destination Trajectory 2, Line*
- *Engine Operations: Draw Operations: Color 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

This register 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

Use this register for line draw or trapezoid draw operations. It is aliased as LEAD_BRES_ERR.

See Also

DST_BRES_DEC on *5-2*

DST_BRES_INC on *5-3*

DST_BRES_LNTH on *5-4*

mach64 Programmer's Guide:

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

This register has two names—DST_BRES_INC or LEAD_BRES_INC

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

a	R/W	DST_BRES_INC	Bresenham increment for line and trapezoid leading edge

Description

This register 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

Use this register for line draw or trapezoid draw operations, and is aliased as LEAD_BRES_INC.

See Also

DST_BRES_DEC on *5-2*

DST_BRES_ERR on *5-3*

DST_BRES_LNTH on *5-4*

mach64 Programmer's Guide:

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

The register below has two names—DST_BRES_LNTH or LEAD_BRES_LNTH

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														

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 this register 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

Use this register for line draw or trapezoid draw operations.

See Also

DST_BRES_DEC on *5-2*

DST_BRES_ERR on *5-3*

DST_BRES_INC on *5-3*

mach64 Programmer's Guide:

- *Engine Operations: Background Information: Trajectories: Destination Trajectory 2, Line*
- *Engine Operations: Draw Operations: Color 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

This register describes the 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 *5-65*

SRC_CNTL on *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: Color 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

Use this register for drawing a rectangular or trapezoidal destination area.

See Also

DST_WIDTH on *5-11*

DST_HEIGHT_WIDTH on *5-9*

mach64 Programmer's Guide:

- *Engine Operations: Background Information: Trajectories: Destination Trajectory 1, Rectangular*
- *Engine Operations: Draw Operations: Color 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

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

Usage

Use these registers only for drawing rectangular destinations.

See Also

DST_HEIGHT on *5-9*

DST_WIDTH on *5-11*

DST_WIDTH_HEIGHT on *5-12*

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

This register specifies 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 *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

This register 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

Use this register only for drawing a rectangular destination area.

See Also

DST_HEIGHT on *5-9*

DST_HEIGHT_WIDTH on *5-9*

DST_X_WIDTH on *5-13*

mach64 Programmer's Guide:

- *Engine Operations: Background Information: Trajectories: Destination Trajectory 1, Rectangular*
- *Engine Operations: Draw Operations: Color 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

This register 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

Use these registers only for drawing rectangular destinations.

See Also

DST_HEIGHT on *5-9*

DST_WIDTH on *5-11*

DST_HEIGHT_WIDTH on *5-9*

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

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

Usage

Use this register for all draw operations.

See Also

DST_X_WIDTH on *5-13*

DST_Y on *5-14*

DST_Y_X on *5-15*

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

This register is a composite of registers DST_X and DST_WIDTH.

Usage

Use this register as an alternative for initiating rectangle fill operations when drawing a rectangular destination area.

See Also

DST_X on *5-12*

DST_WIDTH on *5-11*

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

This register is a composite of registers DST_X and DST_Y. It provides destination coordinates in the inverse order to DST_Y_X.

Usage

Use these registers for all draw operations.

See Also

DST_X on *5-12*

DST_Y on *5-14*

DST_Y_X on *5-15*

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

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

Usage

Use this register for all draw operations.

See Also

DST_X on *5-12*

DST_Y_X on *5-15*

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

This register is a composite of registers DST_X and DST_Y.

Usage

Use these registers for all draw operations.

See Also

DST_X on *5-12*

DST_Y on *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

This register 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

This register 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

This register 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 (*8)

Description

This register 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.

(*8) means the unit value (in decimal) contained in the bits represents 8 pixels.

Example: to set the pitch of the Z-buffer area to 16 pixels, insert 02h (i.e., 2 decimal) into bits [19:0]; therefore, 2*8 = 16.

Usage

Use this register for all 3D draw operations.

All other Z functions directly track the DST registers.

See Also

Z_CNTL on *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 =Disable Z testing 1 = Enable Z testing
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	Enable specific Z test 000 = Z test never passes 001 = Z < current Z 010 = Z <= current Z 011 = Z == current Z 100 = Z >= current Z 101 = Z > current Z 110 = Z != current Z 111 = Z test always passes A passing Z test will overwrite the existing value with the new source value
d	R/W	Z_MASK	Enable writing to the Z planes: 0 = Disable writing to the Z planes 1 = Enable writing to the Z planes

Description

This register controls the new Z source FIFO which supports the hardware Z buffering.

Usage

Z_EN toggles Z 'on' or '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 *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
	n	**m**	**l**	**k**	**j**				**i**								**h**	**g**	**f**	e		d			c	b						a

a	R/W	ALPHA_TST_EN	Enable use of Alpha testing functions 0 = Disable Alpha testing 1 = Enable Alpha testing

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

b	R/W	ALPHA_TEST	Enable specific Alpha 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: 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_MODE_MSB	If this bit is set, and TEX_MAP_AEN is set, the TEXEL alpha is used to modify the source alpha value used to blend with the destination. 0 = Use filtered texture alpha (if TEX_MAP_AEN is 1 and ALPHA_DECAL lighting is off) or alpha interpolator (otherwise) as the source alpha value. 1 = Use the (alpha interpolator * filtered TEXEL alpha) 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	Alpha value written to the destination 000 = Destination alpha is 0x00 001 = Destination alpha is 0xff 010 = Reserved 011 = (reserved) 100 = Destination alpha is As 101 = Destination alpha is 1-A_S 110 = Destination alpha Ad 111 = Destination alpha is 1-Ad
e	R/W	ALPHA_TST_SRC_SEL	Select the value that is compared to REF_ALPHA during alpha testing 0 = Use Texel Alpha 1 = Use Source Alpha

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

f	R/W	FOG_TABLE_EN	Enable the fog table If fog is enabled, this bit denotes that the fog table rather than the alpha/fog interpolator will be used for the fog "density" value. 0 = Use alpha/fog interpolator 1 = Use Z interpolator value to index into the 256x8 fog table. Use the low order bits of Z to interpolate between 2 adjacent entries. Note that when this bit is set and fog is enabled, the Setup engine should set up the Z interpolator.
g	R/W	TEX_CACHE_LINE_SIZE	Allows fetches to use 2QW rather than 4 QW fetches (in some cases this may be a performance improvement) 0 = Fetch a full line (4 QW) 1 = Fetch half lines (2 QW)
h	R/W	IDCT_EN	Indicates that the operation should take data from the IDCT engine, rather then the destination during alpha blending and add it to the generated pixel. For this bit to take effect SIGNED_DST_CLAMP_EN must be on. In this mode, the Alpha blending function will be overridden and assumed to be Src + Dst. If this bit is 1, CMDFIFO2_EN@HW_DEBUG has to be set to 1 as well, in order to have cmdfifo2 and MC/IDCT data parser ON. 0 = IDCT engine off 1 = IDCT engine on (Alpha Blending off)
i	R/W	REF_ALPHA	Reference alpha used for Alpha testing
j	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
k	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
l	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

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

m	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 un-expanded composite texels If they match, the SPECULAR color is substituted for the composite texel, otherwise, 0 is substituted for the composite texel
n	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 Registers

		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	Enable pattern source SRC_Y_END is only used if this bit is enabled
b	R/W	SRC_PATT_ROT_EN	Enable pattern source rotation SRC_X_START, SRC_Y_START is only used if this bit is enabled

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

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. 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
m	R/W	COLOR_REG_WRITE_EN	Enable 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	Enable block write blit using SGRAM color register Not needed if auto-block-writes are enabled (see HW_DEBUG). Write as '0'

Description

This register 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 5-2 Source Trajectories vs SRC_CNTL Enable Bits

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 color to be transferred must be replicated in all 32-bits of the foreground color 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 *5-6*

GUI_TRAJ_CNTL on *5-65*

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 specifies 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 *5-25*

SRC_WIDTH1 on *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 *5-24*

SRC_WIDTH1 on *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 specifies 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 *5-26*

SRC_WIDTH2 on *5-28*

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 *5-25*

SRC_WIDTH2 on *5-28*

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 specifies 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 *5-10*

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 specifies 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 *5-24*

SRC_HEIGHT1_WIDTH1 on *5-25*

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 specifies 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 *5-25*

SRC_HEIGHT2_WIDTH2 on *5-26*

mach64 Programmer's Guide:

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

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

a	R/W	SRC_X	Source X coordinate

Description

This register specifies the starting X coordinate of the blit source trajectory. This is a signed 14 bit number.

Usage

Use this register for any draw operation which selects a blit source in the pixel data path.

See Also

SRC_Y on *5-30*

SRC_Y_X on *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 *5-31*

SRC_Y_X_START on *5-32*

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

Use this register for any draw operation that selects a blit source in the pixel data path.

See Also

SRC_X on *5-29*

SRC_Y_X on *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 *5-29*

SRC_Y_X_START on *5-32*

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 *5-30*

SRC_X on *5-29*

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 *5-29*

SRC_Y_START on *5-31*

5.2 Draw Engine Control Registers

5.2.1 Host Data Registers

The host data registers provide pixel data that are used in the current drawing operation. The pixel data may be used as a monochrome 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

This register 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 *4-6*

HOST_CNTL on *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 = Disable big endian data translation 1 = Enable big endian data translation

Description

HOST_BYTE_ALIGN controls the host data consumption for 1 bpp data. When T_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 *5-65*

HOST_DATA on *5-33*

mach64 Programmer's Guide:

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

5.2.2 Pattern Registers

Two pattern registers (PAT_REG0 and PAT_REG1) 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 5-3 Data Alignment (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 5-4 Data Alignment (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 5-5 Data Alignment (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 5-6 Data Alignment (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

This register 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 *5-38*

PAT_REG1 on *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

This register 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 *5-38*

PAT_REG0 on *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

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

Usage

Only use this register 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 *5-65*

PAT_REG0 on *5-37*

PAT_REG1 on *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 Registers

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

This register 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 *5-41*

SC_BOTTOM on *5-41*

SC_RIGHT on *5-40*

SC_LEFT_RIGHT on *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

This register 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 *5-41*

SC_LEFT on *5-39*

SC_LEFT_RIGHT on *5-40*

SC_BOTTOM on *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

This register 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 *5-39*

SC_RIGHT on *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

This register 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 *5-41*

SC_LEFT on *5-39*

SC_RIGHT on *5-40*

SC_TOP_BOTTOM on *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

This register 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 *5-41*

SC_TOP_BOTTOM on *5-42*

SC_LEFT on *5-39*

SC_RIGHT on *5-40*

mach64 Programmer's Guide:

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

SC_TOP_BOTTOM **MM: 0_AD**

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

a	W	SC_TOP	Top scissor
b	W	SC_BOTTOM	Bottom scissor

Description

This register 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 *5-41*

SC_BOTTOM on *5-41*

5.2.4 Data Path Registers

	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

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

Table 5-7 Video Modes vs Bits Used

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, use this register 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

This register holds a solid color source. The number of bits used varies depending on graphics modes, as follows:

Table 5-8 Video Modes vs Bits Used

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, use this register 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*

	DP_FRGD_BKGD_CLR																												MM: 0_B8			
BITS	31	30	29	28	27	26	25	24	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	DP_FRGD_CLR	Foreground color [0..15]
b	W	DP_BKGD_CLR	Background color [0..15]

Description

This register sets the pixel depth.

Usage

Set this register 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

This register inhibits the 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 5-8 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																															
BITS	31	30	29	28	27	26	25	24	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**	DP_DST_PIX_WIDTH	Destination datapath pixel width 0000 = Monochrome 0001 = Reserved 0010 = 8 bpp pseudocolor 0011 = 16 bpp aRGB 1555 0100 = 16 bpp RGB 565 0101 = 24 bpp. Valid only when SCALE_3D_FCN is set to SCALING. No ALPHA blending is allowed in this mode 0110 = 32 bpp aRGB 8888 0111 = 8 bpp RGB 332 1000 = Y8 greyscale 1001 = RGB8 greyscale (8 bit intensity, duplicated for all 3 channels. Green channel is used on writes) 1010 = Reserved 1011 = YUV 422 packed (VYUY) 1100 = YUV 422 packed (YVYU) 1101 = Reserved 1110 = aYUV 444 (8:8:8:8) 1111 = 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 0000 = Reserved 0001 = Reserved 0010 = 8 bpp pseudocolor 0011 = 16 bpp aRGB 1555 0100 = 16 bpp RGB 565 0101 = Reserved 0110 = 32 bpp aRGB 8888 0111 = 8 bpp RGB 332 1000 = Y8 greyscale 1001 = RGB8 greyscale (8 bit intensity, duplicated for all 3 channels. Green channel is used on writes) 1010 = Reserved 1011 = YUV 422 packed (VYUY) 1100 = YUV 422 packed (YVYU) 1101 = (reserved) 1110 = aYUV 444 (8:8:8:8) 1111 = 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
	l				k	j	i	h	g				f					e	d		c				b				a			

c	R/W	DP_SRC_PIX_WIDTH	Source datapath pixel width 0000 = Monochrome 0001 = Reserved 0010 = 8 bpp pseudocolor 0011 = 16 bpp aRGB 1555 0100 = 16 bpp RGB 565 0101 = Reserved 0110 = 32 bpp aRGB 8888 0111 = 8 bpp RGB332 1000 = Y8 greyscale 1001 = Reserved 1010 = Reserved 1011 = YUV 422 packed (VYUY) 1100 = YUV 422 packed (VYUY) 1101 = Reserved 1110 = aYUV 444 (8:8:8:8) 1111 = aRGB4444
d	R/W	DP_HOST_TRIPLE_EN	0 = Disable host data triplication 1 = Enable host data triplication
e	**R/W**	DP_PALETTE_TYPE	If set to 1, this indicates that the Texture Palette contains 32 bit ARGB entries This bit is assumed to be 0 unless SCALE_3D_FCN is set to SCALING and SCALE_PIX_REP is 1
f	R/W	DP_HOST_PIX_WIDTH	Host datapath pixel width 0000 = Monochrome 0001 = Reserved 0010 = 8 bpp pseudocolor 0011 = 16 bpp aRGB 1555 0100 = 16 bpp RGB 565 0101 = (reserved) 0110 = 32 bpp aRGB 8888 0111 = 8 bpp RGB8 332 1000 = Y8 greyscale 1001 = RGB8 greyscale 1010 = (reserved) 1011 = YUV 422 packed (VYUY) 1100 = YUV 422 packed (YVYU) 1101 = (reserved) 1110 = aYUV444 (8:8:8:8) 1111 = aRGB4444
g	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 palettes

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

h	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
i	R/W	DP_CONVERSION_TEMP	YUV to RGB conversion temperature 0 = Red@6500 K, GB@9300 K 1 = RGB@9300K
j	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
k	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
l	R/W	DP_SCALE_PIX_WIDTH	Scaler source and 3D (texture and shading) datapath pixel width 0000 = Reserved 0001 = Reserved 0010 = 8 bpp pseudocolor 0011 = 15 bpp aRGB 1555 0100 = 16 bpp RGB 565 0101 = (reserved) 0110 = 32 bpp aRGB 8888 0111 = 8 bpp RGB8 332 1000 = Y8 greyscale 1001 = RGB8 greyscale (8 bit intensity, duplicated for all three channels, Green channel is used on writes) 1010 = Reserved 1011 = YUV 422 packed (VYUY) 1100 = YUV 422 packed (YVYU) 1101 = (reserved) 1110 = aYUV444 (8:8:8:8) 1111 = 16 bpp aRGB 4444

Description

This register 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

Use this register 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 color sources:

Table 5-9 Host Destination Pixel Width Combinations

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 5-10 Scalar 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 Pseudocolor-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

This register 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 5-11 Mix Function Descriptions

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

Always set this register. 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 *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

Use this register with DP_SET_GUI_ENGINE and DP_SET_GUI_ENGINE2.

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
	b												a																			

a	W	USR1_DST_OFFSET	DST1_OFFSET value
b	W	USR1_DST_PITCH	DST1_PITCH value

Usage

Use this register to support 2D/3D operations on the secondary display.

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
	b												a																			

a	W	USR2_DST_OFFSET	DST2_OFFSET value
b	W	USR2_DST_PITCH	DST2_PITCH value

Usage

Use this register to support 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	000 = Mono 001 = Reserved 010 = 8 Bpp 011 = 15 Bpp 100 = 16 Bpp 101 = Reserved 110 = 32 Bpp 111 = Reserved
b	W	SET_DP_SRC_PIX_WIDTH	0 = Mono 1 = Set same as SET_DP_DST_PIX_WIDTH setting
c	W	SET_DST_OFFSET	000 = 0 001 = 256KB 010 = 512KB 011 = 768KB 100 = 1MB 101 = Reserved 110 = USR1_DST_OFFSET 111 = 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	0 = SRC_OFF_PITCH set to 0 1 = SRC_OFF_PITCH set to DST_OFF_PITCH
g	W	SET_SRC_HGTWID1_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	Enable bus mastering for any subsequent GUI operations
k	W	SET_BUS_MASTER_SYNC	Synchronize GUI operations to bus master No operations permitted until both GUI and bus master are complete
l	W	DP_HOST_TRIPLE_EN	Enable 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	Enable block write blit using SGRAM color 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 5-12 Registers Set by DP_SET_GUI_ENGINE

Register	Value
DST_Y_X	0
DST_HEIGHT_WIDTH	0
SRC_Y_X	0

Table 5-12 Registers Set by DP_SET_GUI_ENGINE

Register	Value
SC_TOP_BOTTOM	3FFF0000h = OPEN completely
SC_LEFT_RIGHT	1FFF0000h = OPEN completely
DP_WRITE_MSK	FFFFFFFFh = enable all destination writes
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 5-13 DRAWING_COMBO Table

	DP_SRC	DP_MIX	GUI_TRAJ_CNTL	Used in
0000	XXXXXXXX	XXXXXXXX	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)
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+ SrcPattEn+SrcPatt RotEn	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)

Table 5-13 DRAWING_COMBO Table

	DP_SRC	DP_MIX	GUI_TRAJ_CNTL	Used in
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
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

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			

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:

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: 000 = Background color 001 = Foreground color 010 = Host data 011 = Blit source 100 = Pattern registers 101 = Scaler/3D data 011 = Reserved 111 = Reserved

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		

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

This register 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 Registers

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

This register is compared against the source or destination data to determine whether the 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 *5-61*

CLR_CMP_MSK on *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

This 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 *5-60*

CLR_CMP_CNTL on *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 000 = False 001 = True 010 = Reserved 011 = Reserved 100 = DST_CLR != CLR_CMP_CLR 101 = DST_CLR = CLR_CMP_CLR 110 = Reserved 111 = Reserved
b	R/W	CLR_CMP_SRC	Defines source for color keying 00 = Destination 01 = 2D Source 10 = Texel Source/Scaler Source 11 = Reserved

Description

This register configures 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, the 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

Use this register for selectively inhibiting the drawing of certain pixels which key on the source data or destination data.

See Also

CLR_CMP_CLR on *5-60*

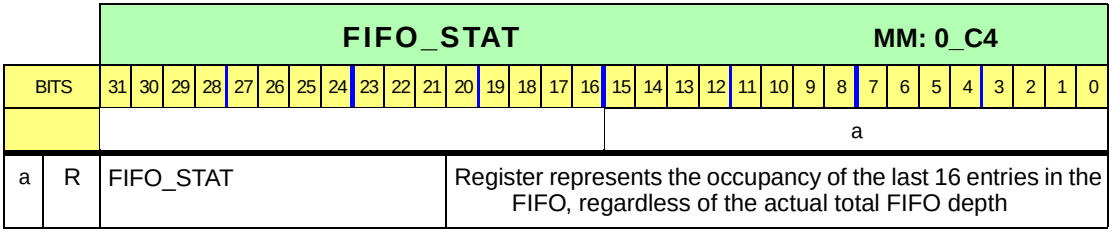
CLR_CMP_MSK on *5-60*

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 Registers

The command FIFO is 'n' words deep by 32 bits, where n > 16.
For the RAGE Mobility, n = 64, 256, or 512 as determined by CMD_FIFO_SIZE_MODE@GUI_CNTL.



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 *4-6*

GEN_TEST_CNTL on *4-27*

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	31:0		
	a		
a	R	GUI_CMDFIFO_DATA	Value read from FIFO pointed to by the read pointer

GUI_CNTL			MM: 1_5E
BITS	31:18	17	16	15:2	1:0
		c	b		a
a	**R/W**	**CMDFIFO_SIZE_MODE**	**Set CMDFIFO size** **Mode PIO BM** **00 = 512 0** **01 = 384 128** **10 = 128 384** **11 = 256 256** **FIFO must be empty before writing this register** **Writing to this register should be followed by a reading from GUI_STAT for proper synchronization**
b	**R/W**	**IDCT_PRSR_MODE**	**0 = Writes two IDCT_LEVELS in the same order they are coming from HBIU block** **1 = Changes the order of the two incoming IDCT_LEVEL register writes**
c	**R/W**	**IDCT_BLOCK_GUI_ INITIATOR**	**0 = Writes to IDCT registers will trigger IDCT operation** **1 = Writes to IDCT registers will NOT trigger IDCT operation**

5.2.7 Draw Engine Composite Control Registers

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	Enable rectangular tiling in the X direction
e	R/W	DST_Y_TILE	Enable 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	Enable 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	Disable drawing of the right edge pixel of a polygon fill operation 0 = Enable drawing of right edge pixel 1 = Disable drawing of right edge pixel
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 pattern source SRC_Y_END will only be used if this bit is enabled
p	R/W	SRC_PATT_ROT_EN	Enable pattern source rotation SRC_X_START, SRC_Y_START will only be used if this bit is enabled
q	R/W	SRC_LINEAR_EN	Enable 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	Enable monochrome 8x8 pattern
x	R/W	PAT_CLR_4x2_EN	Enable color 4x2 pattern
y	R/W	PAT_CLR_8x1_EN	Enable color 8x1 pattern
z	R/W	HOST_BYTE_ALIGN	Enable byte alignment of the host data
aa	R/W	HOST_BIG_ENDIAN_EN	Enable 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 = Disable big endian data translation 1 = Enable big endian data translation

Description

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

Usage

Use this register for general draw operations.

See Also

DST_CNTL on *5-6*

SRC_CNTL on *5-21*

PAT_CNTL on *5-38*

HOST_CNTL on *5-34*

5.2.8 Draw Engine Status Registers

	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 DST_X is left of left scissor
c	R	DSTX_GT_SICISSOR_RIGHT	Indicates DST_X is right of right scissor
d	R	DSTY_LT_SCISSOR_TOP	Indicates DST_Y is above top scissor
e	R	DSTY_GT_SICISSOR_ BOTTOM	Indicates DST_Y 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

This register reports the status of the draw engine.

Usage

The GUI_ACTIVE bit determines 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 *5-62*

mach64 Programmer's Guide:

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

Chapter 6
Host Interface

This chapter describes the registers used for configuring the host interface functions. For an explanation of the notations used to describe the registers, refer to Chapter 1, section *1.3.3*.

Table 6-1 Register Class

6.1 PCI Configuration Space Registers

The RAGE Mobility 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' 4C47h = ASCII characters 'LG'															

Usage

Refer to Table 8-1.

The actual value is dependent on PCI33EN (bonding option), and straps BUS_TYPE (MA6), and CHANGE_ID (MA0) on power up (refer to page 15 of Mobility Pinout Specification).

Table 6-2 Device IDs

DEVICE ID	Description
4C4Dh (LM)	AGP 1X, 2X, full function

Table 6-2 Device IDs

4C4Eh (LN)	AGP 1X, 2X
4C52h (LR)	PCI 33, 66, full function
4C53h (LS)	PCI 33, 66

		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 6-3 ASIC IDs

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							

		LATENCY_TIMER	Byte Addr: 0D
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

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							

b	R/W	Block Decoded I/O Address

REGISTE_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 = 1h 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 = 0h)
d	R	AGP MAJOR: (always = 1h, 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 = 1h)
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: (defaults to 0h after reset) Indicates enabled transfer rate, bit [0:1X], bit [1:2X] Only one bit may be on
b	R	Reserved: (always = 0h)
c	R/W	Enable AGP: (defaults to 0h after reset)
d	R/W	Enable Side Band Addressing: (defaults to 0h after reset)
e	R	Reserved: (always 0h)
f	R/W	Number of requests the device is allowed to enqueue: (defaults to 0h 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)

Cont'd		POWER_MANAGEMENT_CAPABILITY Byte Addr: 5F:5E
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 Registers

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	000 = Transfer system memory data to frame buffer immediately 001 = Transfer frame buffer data to system memory immediately 010 = Transfer frame buffer to system memory on capture Buf0 completion 011 = Transfer frame buffer to system memory on capture Buf1 completion 100 = Transfer frame buffer to system memory on host one-shot completion 101 = Transfer system memory to MPP data port 110 - 111 = 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 Registers

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 6-14*

BM_DATA on *page 6-14*

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
	d	c							b																a							

a	R/W	GUIREG_ADDR	Index offset to desired GUI/ENG_3D/IDCT register, or 'don't care' in case of IDCT_FLAGS = 0b00, or index offset of IDCT_RUNS register in case of IDCT_FLAGS=0B01h
b	R/W	GUIREG_COUNTER	The number of GUI registers to be consecutively written IDCT IDCT_FLAGS Value to program Auto-increment 0 xx N-1 YES 1 00 X n/a 1 01 T*3 - 1 YES/NO 1 10 N - 1 YES Where: N = The number of BM_DATA writes that follow T = Number of BM_DATA R, L, L triplets that follow X = Don't care
c	R/W	IDCT_FLAGS	If IDCT = 1 these bits determine type of IDCT packet (no effect otherwise) 00 = End of IDCT block request (i.e. IDCT block with all 0 coefficients will be generated). No BM_DATA is required/allowed 01 = Triplets of RUN, LEVEL, LEVEL, RUN, LEVEL… (data will follow) 10 = Write to IDCT registers with auto-increment of GUIREG_ADDR 11 = Reserved
d	R/W	IDCT	0 = Data for command FIFO (i.e. guiengine, eng_3d) 1 = Data for IDCT

Including the following register:

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 transfers 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 6-13*

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	00 = 16KB (1K entries) 01 = 32KB (2K entries) 10 = 64KB (4K entries) 11 = 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	00 = 16KB (1K entries) 01 = 32KB (2K entries) 10 = 64KB (4K entries) 11 = 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 111110 = 8MB 111100 = 16MB 111000 = 32MB 110000 = 64MB 100000 = 128MB 000000 = 256MB (default)
b	R/W	MAX_IDLE_CLK	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

This chapter describes the registers used for configuring the VGA compatible register functions. For an explanation of the notations used to describe the registers, refer to Chapter 1, section *1.3.3*.

7.1 VGA Compatible Registers Summary – By I/O Port

VGA registers in the RAGE Mobility 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

Register 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)	*7-25*
DAC Registers (DACxx)	*7-29*
Graphics Controller Registers (GRAxx)	*7-30*

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	7-7
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	7-8
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	7-10
03?5	B		R/W	CRT0B	Cursor End	7-10
03?5	C		R/W	CRT0C	Start Address (High Byte)	7-11
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)	7-12
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	7-14
03?5	15		R/W	CRT15	Start Vertical Blanking	7-14
03?5	16		R/W	CRT16	End Vertical Blanking	7-15
03?5	17		R/W	CRT17	CRT Mode	7-15

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

Port	Index	Function	Type	Mnemonic	Register Name	Page
03?5	18	CRT Controller	R/W	CRT18	Line Compare	7-16
03?5	1E,1F		R	CRT1E, 1F	Graphic Controller Index Decode	7-17
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
03C0	–	Attribute Controller	R/W	ATTRX	Attribute Controller Index	7-18
03C0	00-0F		W	ATTR(00:0F)	Palette (00 to 0F)	7-18
03C0	10		W	ATTR10	Mode Control	7-19
03C0	11		W	ATTR11	Overscan Color	7-20
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	7-19
03C1	11		R	ATTR11	Overscan Color	7-20
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

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

Port	Index	Function	Type	Mnemonic	Register Name	Page
03C4	–	Sequencer	R/W	SEQX	Sequencer Index	7-25
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	7-27
03C5	4		R/W	SEQ04	Memory Mode	7-28
03C6	–	DAC	R/W	DAC_MASK	DAC Mask	7-29
03C7	–		R/W	DAC_R_INDEX	DAC Read Current Color Index	7-29
03C8	–		R/W	DAC_W_INDEX	DAC Write Current Color Index	7-29
03C9	–		R/W	DAC_DATA	DAC Data	7-29
03CA	–	General	R	GENFC	Feature Control	7-22
03CC	–		R	GENMO	Miscellaneous Output	7-23
03CE	–	Graphics Controller	R/W	GRAX	Graphics Controller Index	7-30
03CF	0		R/W	GRA00	Set/Reset	7-30
03CF	1		R/W	GRA01	Enable Set/Reset	7-31
03CF	2		R/W	GRA02	Color Compare	7-32
03CF	3		R/W	GRA03	Data Rotate	7-32
03CF	4		R/W	GRA04	Read Map Select	7-33
03CF	5		R/W	GRA05	Graphics Mode	7-34
03CF	6		R/W	GRA06	Graphics Miscellaneous	7-35
03CF	7		R/W	GRA07	Color Don't Care	7-36
03CF	8		R/W	GRA08	Bit Mask	7-37
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 = Enable write-only to CRT10 and CRT11 1 = Enable 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 horizontal 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)
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

	PRESET_ROW_SCAN (CRT08)				I/O: 3?5		INDEX: 08	
BITS	7	6	5	4	3	2	1	0
		b		a				

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’					

STAR_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

END_VERTICAL_RETRACE (CRT11)					I/O: 3?5		INDEX: 11	
BITS	7	6	5	4	3	2	1	0
	e	d	c	b	a			

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 = Enable 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]
c	R/W	DOUBLE_WORD	Double-Word Mode 0 = Allows addressing mode to be selected by CRT17[6] 1 = Enable 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]	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 = Enable 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

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

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 = Enable 256K video memory addressing. In word mode, address counter bits are rotated left by one, so bit 15(MA15) is wrapped around to bit 0 position at the CRTC output
f	R/W	BYTE_MODE	Byte/Word Mode 0 = Selects word mode memory addressing. The memory address is rotated left by one 1 = Selects byte mode memory addressing
g	R/W	CRTC_SYNC_EN	H/V Retrace Enable 0 = Disable horizontal and vertical retrace 1 = Enable 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} These bits 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 = Enable the special line graphics character codes for monochrome emulation, and sets the ninth dot of a line graphics character to be the same as the eighth dot
d	R/W	ATTR_BLINK_EN	Blink Enable/*Background intensity 0 = Allows bit [7] of the character attribute to control background intensity 1 = Allows bit [7] of the character attribute to control blinking
e	R/W	ATTR_PANTOPONLY	PEL Panning Compatibility 0 = Allows both halves of a split screen to pan together by preventing a line compare split screen function from affecting the output of PEL panning register ATTR13 and byte panning bits CRT08 [6:5] 1 = For panning only the top half of a split screen by forcing ATTR13 output to zero until the start of the next V sync pulse when line compare condition is “true”
f	R/W	ATTR_PCLKBY2	PEL Clock Select 0 = Shift registers are clocked every dot clock 1 = For 256 color mode 13h: eight bits of video data are packed to form a pixel
g	R/W	ATTR_CSEL_EN	Alternate Color Source 0 = Selects palette register bits [4] and [5] (in ATTR00-0F) as source for color output bits P4 and P5 1 = Selects ATTR14 [1:0] as source for color output bits P4 and P5, respectively

OVERSCAN_COLOR (ATTR11)						I/O: 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 = Disable data from maps 3-0 to be used for video output 1 = Enable 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 Left Shift vs Shift Count

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 = Disable VGA video subsystem (controller) The VGA video subsystem only responds to memory read operations to the BIOS ROM. All other I/O or memory read/write operations are suspended 1 = Enable 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 = Enable display of video data 1 = Disable display of video data
b	R	VGA_VSTATUS	Vertical Retrace Status
c	R	PIXEL_READ_BACK[1:0]	Diagnostic bits [0] and [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 = Disable CPU access to video RAM (0) (default value) 1 = Enable 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 = Select odd (high) video memory locations 1 = Select 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 = Enable 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 = Disable I/O write to GENVS(0102) 1 = Enable 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	d	c	b		a

a	R/W	SEQ_DOT8	8/9 Dot Clocks 0 = Selects 9-dot character clocks 1 = Selects 8-dot character clocks Modes 0, 1, 2, 3, 7 use 9-dot characters To change bit 0, GENVS[0] must be logical zero
b, c	R/W	SEQ_SHIFT4 SEQ_SHIFT2	Shift 4, Shift Load bits 00 = Loads video serializers every character clock 01 = Loads video serializers every other character clock 10 = Loads video serializers every fourth character clock 11 = Loads video serializers every fourth character clock
d	R/W	SEQ_PCLKBY2	Dot Clock 0 = Indicates dot clock is Master clock 1 = Indicates dot clock is Master clock divided by 2 Typically, 320 and 360 horizontal modes use divide-by-2 to provide 40 column displays. To change this bit SEQ00[0:0] must first be set to zero
e	R/W	SEQ_MAXBW	0 = Allows CPU:CRT interleaved access to video memory 1 = Blanks the screen and disables video-generation logic access to video memory. Allows CPU uninterrupted access to video memory

Note: To change this register, SEQ00[1 or 0] must first be logical zero.

MAP_MASK (SEQ02)						I/O: 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 = Disable write access to memory map 0 1 = Enable write access to memory map 0
b	R/W	SEQ_MAP1_EN	Enable Map 1 0 = Disable write access to memory map 1 1 = Enable write access to memory map 1
c	R/W	SEQ_MAP2_EN	Enable Map 2 0 = Disable write access to memory map 2 1 = Enable write access to memory map 2
d	R/W	SEQ_MAP3_EN	Enable Map 3 0 = Disable write access to memory map 3 1 = Enable write access to memory map 3

Notes:

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

CHARACTER_MAP_SELECT (SEQ03)						I/O: 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

*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 = Enable 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 = Enable 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) Refer to 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 Refer to 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
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

Cont'd

SET_RESET (GRA00) I/O: 3CF INDEX: 00

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

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
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 (alphanumeric mode): display data bypasses the graphics controller and latches into the attribute controller 1 = Selects APA (graphics) mode: color data is serialized in the shift registers before it is passed to the attribute controller
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 GRA05fs

This page intentionally left blank.

Chapter 8
LCD Panel Registers

This chapter describes the registers used for configuring the front-end scaler and 3D pipeline functions. For an explanation of the notations used to describe the registers, refer to Chapter 1, section *click here*.. The LCD panel registers can only be accessed through memory mapped I/O and relocatable block I/O space. All registers are indexed.

Table 8-1 Register Class

8.1 Index and Data Registers

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
				j	**i**	**h**	**g**	**f**													**e**	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	Disable the display from second display path by forcing blank signal
e	**R**	**GUI_ACTIVE**	**Indicates GUI engine is active**
f	**R**	**MONDET_SENSE**	**For detecting the connectivity of a panel Direct input from MONDET pin 1 = Panel connected 0 = No panel connected**
g	**R/W**	**MONDET_INT_POL**	**0 = Interrupt on falling edge of MON_DET 1 = Interrupt on the rising edge**
h	**R/W**	**MONDET_INT_EN**	**0 = No interrupt generated from MONDET 1 = Generate PCI interrupt when specified edge occurs on MONDET**
i	**R**	**MONDET_INT**	**1 = Specified edge has occurred on MONDET pin 0 = Edge has not occurred on MONDET pin**
i	**W**	**MONDET_INT_ACK**	**1 = Clear this bit to ‘0’**
j	**R/W**	**MONDET_EN**	**Enable MONDET feature 0 = Disable 1 = Enable**

	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
	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

8.2 Configuration and Timing Registers

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

a	R/W	PANEL_FORMAT	Specifies type of pixel format the LCD panel uses For split-panel color STN panels: 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: 000 = PACK12 (12-bit interface) 001 = PACK16 (16-bit interface) For TFT panels: 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) All other non-specified combinations are reserved
b	R/W	PANEL_TYPE	Specifies type panel being used 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) Monochrome panel is not supported All other non-specified combinations are 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
	o								**n**	m	l		k	j	i	h	**g**	f	e	d		c			**b**					**a**		

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
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**	**LCD_IO_DRIVE**	**LCD I/O output drive strength control (applies to all LCDDO and LCDTMG pins)** **0 = No boost** **1 = Boost**
h	R/W	FP_POL	0 = Active high Frame Pulse / Vsync 1 = Active low Frame Pulse / Vsync
i	R/W	LP_POL	0 = Active high Line Pulse / Hsync 1 = Active low Line Pulse / Hsync
j	R/W	DTMG_POL	0 = Active high Display Enable / MOD 1 = Active low Display Enable / MOD
k	R/W	SCK_POL	0 = Active high Shift Clock / PCLK 1 = Active low Shift Clock / PCLK
l	R/W	DITHER_SEL	00 = Disable dithering 01 = Dither to 4 bits 10 = Dither to 5 bits 11 = Dither to 6 bits
m		RESERVED	Program to '0' at all times

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

n	**R/W**	**BL_CLK_SEL**	**Backlight modulation clock selection** **0 = 29 MHz reference clock** **1 = 29 MHz reference clock divided by 3**
		RESERVED	Bits [24:25] are reserved bits. They must be programmed to zero.
o	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

This register configures 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 RAGE Mobility Graphics Controller; and documents about LCD panel interface.

8.3 LCD General Control Registers

	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
	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/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
	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

c	R/W	HORZ_DIVBY2_EN	0 - do not divide horizontal CRT parameters when shadow registers are used and SEQPCLKBY2 is active 1 - divide horizontal CRT parameters
d	R/W	DONT_DS_ICON	Do not double scan hardware icon and cursor in double scan modes: 0 = Expand the cursor & icon id in double scan mode (default) 1 = Do not expand the cursor & icon in double scan mode when both cursor & icon are enabled.
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 = Enable hardware cursor 1 = Enable hardware icon
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 = Reset the frame modulation circuit
j	**R/W**	**DISABLE_PCLK_RESET**	**0 = Generate PCLK reset when memory cntl (reset is generated)** **1 = Disable PCLK reset when memory cntl (reset is generated)**
k	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
l	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
m	R/W	SCLK_DELAY	Adjust setup/hold time for LCD data (1 ns increment) 0000 - 0111 = SCLK comes before LCD data 1000 = SCLK and LCD data have the same delay (default after reset) 1001 - 1111 = SCLk comes after LCD data
n	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

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
	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

o	R/W	VCLK_DAC_ALWAYS_ONb	0 = VCLK_DAC is always on 1 = VCLK_DAC is off druing blank time
p	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
q	R/W	XTALIN_ALWAYS_ONb	0 = XTALIN is always on 1 = XTALIN depends on power management state (e.g. stops in suspend)
r	R/W	V2CLK_DAC_ALWAYS_ON	0 = V2CLK_DAC is always on 1 = V2CLK_DAC is off during blank time
s	R/W	LVDS_EN	0 = Disable on-chip LVDS interface 1 = Enable on-chip LVDS interface
t	R/W	LVDS_PLL_EN	0 = Disable LVDS PLL 1 = Enable LVDS PLL
u	R/W	LVDS_PLL_RESET	Reset LVDS PLL
v	R/W	LVDS_RESERVED_BITS	Controls value to output on the LVDS_B reserved bits
w	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
x	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]
y	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]
z	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]
A	**R/W**	**SHADOW_RW_EN**	**0 = Disable read/write to the panel shadow registers** **1 = Enable read/write to the panel shadow registers**

Description

This register is used for general LCD panel configuration. Control as implemented by the RAGE Mobility, and also to control the shadowed CRTC registers.

8.4 Dual Scan Registers

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
	i		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**	**HFB_TEST_EN**	**Enable testing of HFB, data is from the system directly**
c	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
d	R/W	AUTO_LOWER_PANEL _VPOS	0 = Use register value from LOWER_PANEL_VPOS 1 = Calculate value based on Vertical Display End
e	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
f	R/W	LP_VPOS_ADJUST0	Adjust position of the vsync during ratiometric expansion 00 = No shift 01 = Shift by 1 line 10 = Shift by 2 lines 11 = Shift by 3 lines
g	R/W	LP_VPOS_ADJUST1	Adjust position of the vsync during ratiometric expansion 00 = No shift 01 = Shift by 1 line 10 = Shift by 2 lines 11 = Shift by 3 lines
h	R/W	LP_VPOS_ADJUST2	Adjust position of the vsync during ratiometric expansion 00 = No shift 01 = Shift by 1 line 10 = Shift by 2 lines 11 = Shift by 3 lines

Cont'd	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
	i		h		g		f		e	d	c										**b**	a										

i	R/W	LP_VPOS_ADJUST3	Adjust position of the vsync during ratiometric expansion 00 = No shift 01 = Shift by 1 line 10 = Shift by 2 lines 11 = Shift by 3 lines

Description

This register configures the 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 8-3*:

8.5 Half Frame Buffer Registers

	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
													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 0000 = Size is unlimited 0001 - 1110 = Size in 32K blocks 1111 = 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

This register configures 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 8-3*.

8.6 Horizontal and Vertical Stretching Registers

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
	f	e	**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] All others = Reserved
c	**R/W**	**HORZ_PANEL_SIZE**	**Panel horizontal display in characters (pixels/8 –1)**

	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
	f	e	**d**		**c**									b			a															

d	**R/W**	**AUTO_HORZ_RATIO**	**Enable the calculation of horizontal stretch ratio based on actual horizontal display and overscan instead of H_DISP_END, OVR_WID_LEFT & OVR_WID_RIGHT registers** **0 = Disable auto horizontal stretch ratio** **1 = Enable auto horizontal stretch ratio**
e	R/W	HORZ_STRETCH_MODE	Horizontal stretch mode 0 = Pixel replication (use shift register) 1 = Horizontal blending
f	R/W	HORZ_STRETCH_EN	Horizontal stretch enable 1 = Enable horizontal stretching

Description

This register defines the 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)

	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
	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

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
	e	d	c										b										a									

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_RATIO
e	R/W	VERT_STRETCH_EN	Vertical stretch enable 1 = Enable vertical stretching

Description

This register defines the 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
								f	e	d	c											b	**a**									

a	**R/W**	**VERT_STRETCH_RATIO3**	**Vertical stretch ratio (for 600 non standard VGA mode)** **VERT_STRETCH_RATIO =** **[no_of_lines_of_source/ no_of_lines_of _destination] * 1024**
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)

	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
								f	e	d	c											b	a								

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

8.7 LT_GPIO and ZVGPIO Registers

	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
	u	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: (default = 0) 0 = Input 1 = Output
l	R/W	LT_GIO_DIR_1	LT_GIO_1 Direction: (default = 0) 0 = Input 1 = Output

Cont'd		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
	u	t	s	r						q	p	o	n	m	l	k		j	i	h						g	f	e	d	c	b	a

m	R/W	LT_GIO_DIR_2	LT_GIO_2 Direction: (default = 0) 0 = Input 1 = Output
n	R/W	LT_GIO_DIR_3	LT_GIO_3 Direction: (default = 0) 0 = Input 1 = Output
o	R/W	LT_GIO_DIR_4	LT_GIO_4 Direction: (default = 0) 0 = Input 1 = Output
p	R/W	LT_GIO_DIR_5	LT_GIO_5 Direction: (default = 0) 0 = Input 1 = Output
q	R/W	LT_GIO_DIR_6	LT_GIO_6 Direction: (default = 0) 0 = Input 1 = Output
r	R/W	GPIO_DIR_14	GPIO_14 Direction: (default = 0) 0 = Input 1 = Output
s	R/W	GPIO_DIR_15	GPIO_15 Direction: (default = 0) 0 = Input 1 = Output
t	R/W	GPIO_DIR_16	GPIO_16 Direction: (default = 0) 0 = Input 1 = Output
u	R/W	DDC_IO_DRIVE	DDC IO output drive strength (applies to GPIO pins 14 to 16 inclusive) 0 = No boost 1 = Boost

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 Mobility Graphics Controller Specifications 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 4-2* .

		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	
		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)
i	R/W	ZVGPIO_DIR_0	ZVGPIO_0 Direction 0 = Input 1 = Output
j	R/W	ZVGPIO_DIR_1	ZVGPIO_1 Direction 0 = Input 1 = Output
k	R/W	ZVGPIO_DIR_2	ZVGPIO_2 Direction 0 = Input 1 = Output
l	R/W	ZVGPIO_DIR_3	ZVGPIO_3 Direction 0 = Input 1 = Output
m	R/W	ZVGPIO_DIR_4	ZVGPIO_4 Direction 0 = Input 1 = Output
n	R/W	ZVGPIO_DIR_5	ZVGPIO_5 Direction 0 = Input = Output
o	R/W	ZVGPIO_DIR_6	ZVGPIO_6 Direction 0 = Input 1 = Output

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

p	R/W	ZVGPIO_DIR_7	ZVGPIO_7 Direction 0 = Input 1 = Output
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**

8.8 Power Management Registers

	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
	u		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

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

i	R/W	KEYBD_SNOOP	0 = Disable keyboard access snooping 1 = Enable keyboard access to exit the chip from power down modes
j	R/W	DONT_USE_F32KHZ	0 = Use F32 KHz input port as source 1 = Use divided reference clock as a source
k	R/W	TRISTATE_MEM_EN	0 = 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	PM_D3_RST_ENb	0 = Enable global software reset when exiting D3 state (default) 1 = Do not initiate reset when exiting D3 state
s	R/W	STANDBY_NOW	1 = Force the chip to go into standby mode unconditionally
t	R/W	SUSPEND_NOW	1 = Force the chip to go into suspend mode unconditionally
u	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 controls the Power Management features implemented in RAGE Mobility.

Usage

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

POWER_MANAGEMENT_2 **Offset: 0_2A Index: 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
Mobility					u	t	s	r	q	p	o	n	m	l	k	j		i	h					g			f	e	d	c	b	a

a	R/W	XCLK_DISP_ALWAYS_ONb	Enable stopping of primary display memory clock (XCLK_DISP): (default = 0) 0 = XCLK_DISP is running always 1 = Stop XCLK_DISP when CRTC_ENABLE = 0
b	R/W	XCLK_DISP2_ALWAYS_ONb	Enable stopping of secondary display memory clock (XCLK_DISP2): (default=0) 0 = XCLK_DISP2 is running always 1 = Stop XCLK_DISP2 when CRTC2_ENABLE = 0
c	R/W	XCLK_VID_ALWAYS_ONb	Enable stopping of video capture and half frame buffer (HFB) memory clock (XCLK_VID): (default = 0) 0 = XCLK_VID is running always 1 = Stop XCLK _VID when (CAPTURE_EN or HFB_ACTIVE) = 0
d	R/W	XCLK_SCL_ALWAYS_ONb	Enable stopping of scaler/overlay memory clock (XCLK_SCL): (default = 0) 0 = XCLK_SCL is running always 1 = Stop XCLK_SCL when (SCALE_EN) = 0
e	R/W	XCLK_GUI_ALWAYS_ONb	Enable stopping of GUI memory clock (XCLK_GUI): (default = 0) 0 = XCLK_GUI is running always 1 = Stop XCLK_GUI when 2D and 3D guieng are inactive (same condition as GCP)
f	R/W	XCLK_SUB_ALWAYS_ONb	Enable stopping of DVD subpicture memory clock (XCLK_SUB): (default = 0) 0 = XCLK_SUB is running always 1 = stop XCLK_SUB when (SUBPIC_ON) = 0
g	R/W	MCLK_ALWAYS_ONb	Enable stopping of MCLK: (default = 0) 0 = MCLK is running always 1 = Stop MCLK when (MPP_EN = 0, CAPTURE_EN = 0, I2C_SEL = 1, and gui and host are idle)

Cont'd	POWER_MANAGEMENT_2 Offset: 0_2A Index: 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
Mobility					u	t	s	r	q	p	o	n	m	l	k	j		i	h					g			f	e	d	c	b	a

h	R/W	PM_DYN_XCLK_SYNC	Synchronization condition for switching between XCLK speeds (i.e., when to make the decision for XCLK or PM_XCLK speed selection) (default = 10) 00 = Start of VBLANK (primary display only) 01 = Start of V2BLANK (secondary display only) 10 = Combination of VBLANK and V2BLANK (for both displays) as below: (VBLANK and not VSYNC or not CRTC_ENABLE) and (V2BLANK and not V2SYNC or not CRTC2_ENABLE) 11 = Reserved
i	R/W	SEL_W4MS	Wait 4 ms after suspend mode: (default = 1)
j	R/W	PM_DYN_XCLK_EN	Enable dynamic XCLK power saving mode: (default = 0) 0 = XCLK is always running at frequency defined by XCLK_SRC_SEL in CLOCK_CNTL register and corresponding PLL register settings 1 = XCLK dynamically changes between XCLK and PM_XCLK depending on activity. In this setting, PM_REFRESH_RATE and the power saving display settings (defined in PM_DSP_CONFIG and PM_DSP_ON_OFF) are used when running with PM_XCLK
k	R/W	PM_XCLK_ALWAYS	Override forcing to use PM_XCLK rather than XCLK when PM_DYN_XCLK_EN = 1: (default = 0) 0 = XCLK depends on PM_DYN_XCLK_* activity registers (see below) 1 = XCLK = PM_XCLK always; uses slower XCLK regardless of activity
l	R	PM_DYN_XCLK_STATUS	Status of XCLK in power management mode 0 = Running off XCLK 1 = Running off PM_XCLK
m	R/W	PCI_ACC_DIS	When 1, disable I/O and MEM access through bus when in power management state, D1, D2 , D3
n	R/W	PM_DYN_XCLK_DISP	Enable slower XCLK (PM_XCLK) when primary display is active (CRTC_ENABLE = 1): (default = 0) 0 = Must use XCLK when active 1 = Can use PM_XCLK when active depending on other requesters' activity
o	R/W	PM_DYN_XCLK_DISP2	Enable slower XCLK (PM_XCLK) when secondary display is active (CRTC2_ENABLE = 1): (default = 0) 0 = Must use XCLK when active 1 = Can use PM_XCLK when active depending on other requesters' activity

Cont'd	POWER_MANAGEMENT_2	Offset: 0_2A Index: 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
Mobility					u	t	s	r	q	p	o	n	m	l	k	j		i	h					g			f	e	d	c	b	a

p	R/W	PM_DYN_XCLK_VID	Enable slower XCLK (PM_XCLK) when video is active (CAPTURE_EN = 1): (default = 0) 0 = Must use XCLK when active 1 = Can use PM_XCLK when active depending on other requesters' activity
q	R/W	PM_DYN_XCLK_HFB	Enable slower XCLK (PM_XCLK) when half frame buffer is active (HFB_ACTIVE = 1): (default = 0) 0 = Must use XCLK when active 1 = Can use PM_XCLK when active depending on other requesters' activity
r	R/W	PM_DYN_XCLK_SCL	Enable slower XCLK (PM_XCLK) when overlay scaler is active (SCALE_EN = 1): (default = 0) 0 = Must use XCLK when active 1 = Can use PM_XCLK when active depending on other requesters' activity
s	R/W	PM_DYN_XCLK_SUB	Enable slower XCLK (PM_XCLK) when DVD subpicture is active (SUBPIC_ON = 1): (default = 0) 0 = Must use XCLK when active 1 = Can use PM_XCLK when active depending on other requesters' activity
t	R/W	PM_DYN_XCLK_GUI	Enable slower XCLK (PM_XCLK) when 2D/3D gui is active (same as GCP – based on gui state machine and host requests): (default = 0) 0 = Must use XCLK when active 1 = Can use PM_XCLK when active depending on other requesters' activity
u	R/W	PM_DYN_XCLK_HOST	Enable slower XCLK (PM_XCLK) when host writes/reads are active: (default = 0) 0 = Must use XCLK when active 1 = Can use PM_XCLK when active depending on other requesters' activity

8.9 Hardware Icon Registers

	ICON_CLR0	Offset: 0_2A, Index: A

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

a	R/W	ICON_CLR0_B	Blue Icon color 0, to internal DAC

Cont'd	ICON_CLR0																												Offset: 0_2A, Index: A
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mobility	c								b								a														

b	R/W	ICON_CLR0_G	Green Icon color 0, to internal DAC
c	R/W	ICON_CLR0_R	Red Icon color 0, to internal DAC

	ICON_CLR1																												Offset: 0_2A, Index: B
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mobility	c								b								a														

a	R/W	ICON_CLR1_B	Blue Icon color 0, to internal DAC
b	R/W	ICON_CLR1_G	Green Icon color 0, to internal DAC
c	R/W	ICON_CLR1_R	Red Icon color 0, to internal DAC

	ICON_OFFSET																												Offset: 0_2A, Index: C
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mobility								b					a																		

a	R/W	ICON_OFFSET	Icon address offset in terms of 64 bit words
b	R/W	BLANK_SCREEN1	Blank the screen except for ICON on the primary display path

	ICON_HORZ_VERT_POSN																												Offset: 0_2A, Index: D
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mobility						b															a										

a	R/W	ICON_HORZ_POSN	Icon horizontal position
b	R/W	ICON_VERT_POSN	Icon vertical position

ICON_HORZ_VERT_OFF — Offset: 0_2A, Index: E

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

a	R/W	ICON_HORZ_OFF	Icon horizontal offset
b	R/W	ICON_VERT_OFF	Icon vertical offset

ICON2_CLR0 — Offset: 0_2A, Index: F

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

a	R/W	ICON2_CLR0_B	Blue Icon color 0, to internal DAC
b	R/W	ICON2_CLR0_G	Green Icon color 0, to internal DAC
c	R/W	ICON2_CLR0_R	Red Icon color 0, to internal DAC

ICON2_CLR1 — Offset: 0_2A, Index: 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
Mobility									c								b								a							

a	R/W	ICON2_CLR1_B	Blue Icon color 1, to internal DAC
b	R/W	ICON2_CLR1_G	Green Icon color 1, to internal DAC
c	R/W	ICON2_CLR1_R	Red Icon color 1, to internal DAC

ICON2_OFFSET — Offset: 0_2A, Index: 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
Mobility								b					a																			

a	R/W	ICON2_OFFSET	Icon address offset in terms of 64 bit words
b	R/W	BLANK_SCREEN2	Blank the screen except for ICON on the primary display path

ICON2_HORZ_VERT_POSN — Offset: 0_2A, Index: 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
Mobility						b																a										

a	R/W	ICON2_HORZ_POSN	Icon horizontal position
b	R/W	ICON2_VERT_POSN	Icon vertical position

ICON2_HORZ_VERT_OFF — Offset: 0_2A, Index: 13

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

a	R/W	ICON2_HORZ_OFF	Icon horizontal offset
b	R/W	ICON2_VERT_OFF	Icon vertical offset

LCD_MISC_CNTL — Offset: 0_2A, Index: 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
Mobility	i								h		g	f	e	d		c	b								a							

a	R/W	BL_MOD_LEVEL	Blacklight modulation levels
b	R/W	BIAS_MOD_LEVEL	Bias modulation levels
c	R/W	BLMOD_EN	Enable backlight modulation
d	R/W	BIASMOD_EN	Enable bias modulation
e	R/W	PWRSEQ_MODE	0 = Old way to power on/off panel 1 = Turn on clock with digital power
f	R/W	APC_EN	Enable the use of APC algorithm
g	R/W	MONITOR_DET_EN	Enable improved CRT detection circuitry To be used with FORCE_DAC_SEL and FORCE_DAC_DATA
h	R/W	FORCE_DAC_DATA_SEL	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

Cont'd	LCD_MISC_CNTL																													Offset: 0_2A, Index: 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
Mobility	i								h	g	f	e		d	c	b									a							

i		FORCE_DAC_DATA	Used for improved monitor detection (if MONITOR_DET_EN = 1) The value written in this register will be forced directly to the input of the CRT D/A converter

8.10 Scratch Pad Registers

	SCRATCH_PAD_4																													Offset: 0_2A, Index: 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
Mobility	a																															

a	R/W	SCRATCH_PAD_4	Scratch pad register for BIOS/driver use

	SCRATCH_PAD_5																													Offset: 0_2A, Index: 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
Mobility	a																															

a	R/W	SCRATCH_PAD_5	Scratch pad register for BIOS/driver use

	SCRATCH_PAD_6																													Offset: 0_2A, Index: 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
Mobility	a																															

a	R/W	SCRATCH_PAD_6	Scratch pad register for BIOS/driver use

	SCRATCH_PAD_7																													Offset: 0_2A, Index: 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
Mobility	a																															

a	R/W	SCRATCH_PAD_7	Scratch pad register for BIOS/driver use

	SCRATCH_PAD_8																													Offset: 0_2A, Index: 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
Mobility	a																															

a	R/W	SCRATCH_PAD_8	Scratch pad register for BIOS/driver use

8.11 APC Registers

	APC_CNTL																													Offset: 0_2A, Index: 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
Mobility																											e		d	c	b	a

a	R/W	EXTSENSE_AI	Extent sensing for APC macro
b	R/W	EXTSENSE_AD	<no description>
c	R/W	EXTSENSE_AO	<no description>
d	R/W	APC_VIDEO_EN	Enable the use of new algorithm for video in APC
e	R/W	IP_MODE	Acceleration of the LCD frame 00 = 1x 01 = 2x 10 = 3x 11 = 4x

APC_CTRL_IO																														Offset: 0_2A, Index: 27	
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mobility	m																				l	k	j	i	h	g	f	e	d	c	b	a

a	R/W	VIRTUAL_CLOCK	Virtual clock used to drive the apc block during the test mode
b	R/W	APC_IN_SEL	Mux control signal 0 = APC input from normal data path 1 = APC input from registers
c	R/W	DP0_IN_SEL	Mux control signal 0 = DP0 input from normal data path 1 = DP0 input from registers
d	R/W	DPBUF_IN_SEL	Mux control signal 0 = DPBUF input from normal data path 1 = DPBUF input from registers
e	R/W	DP1_IN_SEL	Mux control signal 0 = DP1 input from normal data path 1 = DP1 input from registers
f	R/W	OUTBUF_IN_SEL	Mux control signal 0 = OUTBUF input from normal data path 1 = OUTBUF input from registers
g	R/W	DP0_ERR_IN_SEL	Mux control signal 0 = DP0 reads the error from the line buffers 1 = DP0 reads the error from registers
h	R/W	DP1_ERR_IN_SEL	Mux control signal 0 = DP1 reads the error from the line buffers 1 = DP1 reads the error from registers
i	R/W	APCSYNC_IN_SEL	Mux control signal 0 = APCSYNC input from normal input 1 = APCSYNC input from registers
j	R/W	TEST_MODE_MASTER	Master control bit for the test mode 0 = Test mode disabled 1 = Test mode enabled
k	R/W	DP1_DEBUG	0 = ALL DP1 control signals comes from APC_SYNC 1 = ALL DP1 control signals comes from the DP1_DEBUG register
l	R/W	DP0_DEBUG	0 = ALL DP0 control signals comes from APC_SYNC 1 = ALL DP0 control signals comes from the DP0_DEBUG register

Cont'd	APC_CTRL_IO																												Offset: 0_2A, Index: 27
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mobility	m																				l	k	j	i	h	g	f	e	d	c	b	a

m	R/W	RED_ERR_INPUT[15:0]	Red color of the line buffer output to DP0 or DP1 memory input

	TEST_IO																												Offset: 0_2A, Index: 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
Mobility	h				g					f											e	d	c	b	a							

a	R/W	RED_DATA_INPUT	Red color test input data
b	R/W	VIDEO_FLAG	Video flag bit for all colors during test mode
c	R/W	VIDEO_FM_MEM	Video flag to DP0 or DP1 memory input
d	R/W	IN_ENAB	Data input enable signal for APCSYNC
e	R/W	IN_VSYNC_B	Vsync signal for APCSYNC
f	R	DPBUF_OUT	DP buffer data output
g	R	DP1_OUT	DP1 data output
h	R	OBUF_OUT	Output buffer data output

TEST_OUTPUTS — 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
Mobility	g									f										e	d	c	b	a								

a	**R**	**RED_INPUT[8:0]**	**Red data going into the apc block**
b	**R**	**VFLAG_TO_MEM_BY_DP0**	**Video flag value written to memory by DP0**
c	**R**	**VFLAG_TO_MEM_BY_DP1**	**Video flag value written to memory by DP1**
d	**R**	**VFLAG_FM_MEM_TO_DP0**	**Video flag value read from memory by DP0**
e	**R**	**VFLAG_FM_MEM_TO_DP1**	**Video flag value read from memory by DP1**
f	**R**	**DP0_OUT[6:0]**	**DP0 data output**
g	**R**	**INBUF_OUT[8:0]**	**Input buffer data output**

DP1_MEM_ACCESS — Offset: 0_2A, Index: 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
Mobility	b																a															

a	**R**	**DP1_MEM_IN[15:0]**	**DP1 data input from the line buffer**
b	**R**	**DP1_MEM_OUT[15:0]**	**DP1 data output to the line buffer**

DP0_MEM_ACCESS — Offset: 0_2A, Index: 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
Mobility	b																a															

a	**R**	**DP0_MEM_IN[15:0]**	**DP0 data input from the line buffer**
b	**R**	**DP0_MEM_OUT[15:0]**	**DP0 data output to the line buffer**

DP0_DEBUG_A — Offset: 0_2A, Index: 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
Mobility		**l**	**k**	**j**	**i**	**h**									**g**		**f**		**e**		**d**		**c**		**b**		**a**						

a	**R/W**	**RANDERR_R[5:0]**	**Random error: (default = 0)**
b	**R/W**	**NIBBLE_SELECT[1:0]**	**Nibble select: (default = 0)**
c	**R/W**	**RELX_QUANTPIX[1:0]**	**X position of the input pixel: (default = 0)**
d	**R/W**	**RELX_ACC_ROW[1:0]**	**X position of the input pixel: (default = 0)**
e	**R/W**	**RELX_PARTERR[1:0]**	**Partition error: (default = 0)**
f	**R/W**	**RELY_ACC_ROW[1:0]**	**Y position of the input pixel: (default = 0)**
g	**R/W**	**RELY_PARTERR[1:0]**	**Y position of the input pixel: (default = 0)**
h	**R**	**PREV_PIX_EDATA[8:0]**	**Y position of the input pixel: (default = N/A)**
i	**R/W**	**FIRST_ROW_DP0**	**Default = 0** **0 = Not first row** **1 = First row**
j	**R/W**	**LD_FM_MEM_DP0**	**Default = 0** **0 = Not load memory to DP0** **1 = Load memory to DP0**
k	**R/W**	**LD_FM_MEM_DP0_1ST**	**Default = 0** **0 = Not first time load to DP0** **1 = First time load to DP0**
l	**R/W**	**DP0_MEM_SHIFT_RUN**	**Default = 0** **0 = Do not enable shifting the data registers** **1 = Enable shifting the data registers**

DP0_DEBUG_B — Offset: 0_2A, Index: 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
Mobility	**e**												**d**					**c**	**b**	**a**													

a	**R**	**PIXEL_TARGET_PIPE[13:0]**	**Pixel Target signal in quantpix.arc**
b	**R**	**VFLAG_ROW**	**Video flag for a row in dp0.arc**
c	**R**	**STOREEDATA_VFLAG**	**Stored error video flag in dp0.arc**
d	**R**	**ACCUM_ROW_EDATA[6:0]**	**Accumulated Error signal in dp0.arc**
e	**R**	**NEW_EDATA[8:0]**	**New Error signal in dp0.arc**

DP1_DEBUG_A																															
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mobility		j	i	h	g	f			e	d							c				b				a							

a	R/W	RANDOM_PHASE[5:0]	Random phase value
b	R/W	ACCEL_FRM_CTR[5:0]	Acceleration frame counter for graphics mode
c	R/W	VIDEO_FRM_CTR[3:0]	Acceleration frame counter for video mode
d	R	PH_INCR_PIPE_1[6:0]	Phase Increment pipeline 1 signal in bestph.arc
e	R/W	LD_FROM_MEM	Load CAM state value from line buffer
f	R	PH_SPREAD_PIPE[2:0]	Phase Spread signal in bestph.arc
g	R/W	RESET_CAM	Clear CAM state
h	R/W	TAIL_SHIFT_HALT	Tail shift control for late writing
i	R/W	APC_RESET	Reset APC
j	R	ZERO_FREQ_PIPE_0	Zero frequency pipeline 0 signal in bestph.arc

DP1_DEBUG_B																															
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mobility		j			i	h			g	f	e						d	c							b	a						

a	R	BEST_PHASE_PRE_MOD[6:0]	Best phase pre-mode signal in bestph.arc
b	R	CUR_MATCH_LOC_PIPE_0	Matched location pipeline 0 signal in bestph.arc
c	R	TIME_PHASE[6:0]	Time phase signal in outword.arc
d	R	NEW_ENTRY_PIPE_0	New entry pipeline 0 signal in bestph.arc
e	R	BEST_PHASE[5:0]	Best phase signal in bestph.arc
f	R	ZERO_FREQ_PIPE_1	Zero frequency pipeline 1 signal in bestph.arc
g	R	FIX_PH_FLAG_PIPE_0	Fixed Phase flag pipeline 0 signal in bestph.arc
h	R	KEY_COMP_PIPE_1[2:0]	Key componet pipeline 1 signal in bestph.arc
i	R	FOUND_MATCH_PIPE_0	Found match pipeline 0 signal in bestph.arc
j	R	PH_NOISE_PIPE[2:0]	Phase noise pipeline signal in bestph.arc

DPCTRL_DEBUG_A — Offset: 0_2A, Index: 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
Mobility				h								g					f	e	d							c	b	a				

a	**R**	**INBUF_WRADDR_B[5:0]**	**Input buffer write address**
b	**R**	**INBUF_WEN_B**	**Input buffer write enable**
c	**R**	**INBUF_REN_B**	**Input buffer read enable**
d	**R**	**INBUF_RDADDR_B[5:0]**	**Input buffer read address**
e	**R**	**DPBUF_WEN_B**	**Data path buffer write enable**
f	**R**	**DPBUF_REN_B**	**Data path buffer read enable**
g	**R**	**DPBUF_WRADDR_B[4:0]**	**Data path buffer write address**
h	**R**	**DPBUF_RDADDR_B[4:0]**	**Data path buffer read address**

DPCTRL_DEBUG_B — Offset: 0_2A, Index: 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
Mobility																g	f	e	d						c	b	a				

a	**R**	**OBUF_WRADDR_B[5:0]**	**Output buffer write address**
b	**R**	**OBUF_WEN_B**	**Output buffer write enable**
c	**R**	**OBUF_REN_B**	**Output buffer read enable**
d	**R**	**OBUF_RDADDR_B[5:0]**	**Output buffer read address**
e	**R**	**LD_INPUT_PIX_B**	**Control signal to MEMBLK**
f	**R**	**LD_QUANT_PIX_DP1_B**	**Control signal to MEMBLK**
g	**R**	**E_LD_OUTBUF_B**	**Control signal to MEMBLK**

MEMBLK_DEBUG			Offset: 0_2A, Index: 32
BITS	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Mobility	i h g f e d c b a		
a	R	MEM0_ADDR_B[7:0]	Line buffer 0 address
b	R	MEM1_ADDR_B[7:0]	Line buffer 1 address
c	R	MEM0_CE_B	Line buffer 0 chip enable
d	R	MEM1_CE_B	Line buffer 1 chip enable
e	R	MEM0_RW_B	Line buffer 0 read/write control
f	R	MEM1_RW_B	Line buffer 1 read/write control
g	R	DP0_MEM_RD_B	Control signal to DP0
h	R	DP0_LSTMEM_RD_B	Control signal to DP0
i	R	CAM_TAIL_SHIFT_B	Control signal to DP1

APC_LUT_AB			Offset: 0_2A, Index: 33
BITS	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Mobility	b	a	
a	R/W	VIDPAT_A	Lower 16-bit of VidPattern(1): (default = 16#0001)
b	R/W	VIDPAT_B	Lower 16-bit of VidPattern(2): (default = 16#0101)

APC_LUT_CD			Offset: 0_2A, Index: 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		
Mobility	b	a	
a	R/W	VIDPAT_C	Lower 16-bit of VidPattern(3): (default = 16#0111)
b	R/W	VIDPAT_D	Lower 16-bit of VidPattern(4): (default = 16#1111)

APC_LUT_EF																															Offset: 0_2A, Index: 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
Mobility	b																a															

a	R/W	VIDPAT_E	Lower 16-bit of VidPattern(5): (default = 16#1511)
b	R/W	VIDPAT_F	Lower 16-bit of VidPattern(6): (default = 16#1515)

APC_LUT_GH																															Offset: 0_2A, Index: 36
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mobility	b																a															

a	R/W	VIDPAT_G	Lower 16-bit of VidPattern(7): (default = 16#5515)
b	R/W	VIDPAT_H	Lower 16-bit of VidPattern(8): (default = 16#5555)

APC_LUT_IJ																															Offset: 0_2A, Index: 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
Mobility	b																a															

a	R/W	VIDPAT_I	Lower 16-bit of VidPattern (9): (default = 16#5755)
b	R/W	VIDPAT_J	Lower 16-bit of VidPattern (10): (default = 16#5757)

APC_LUT_KL																															Offset: 0_2A, Index: 38
BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mobility	b																a															

a	R/W	VIDPAT_K	Lower 16-bit of VidPattern (11): (default = 16#7757)
b	R/W	VIDPAT_L	Lower 16-bit of VidPattern (12): (default = 16#7777)

APC_LUT_MN																																Offset: 0_2A, Index: 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
Mobility	b																a															

a	R/W	VIDPAT_M	Lower 16-bit of VidPattern (13): (default = 16#7f77)
b	R/W	VIDPAT_N	Lower 16-bit of VidPattern (14): (default = 16#7f7f)

APC_LUT_OP																																Offset: 0_2A, Index: 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
Mobility	b																a															

a	R/W	VIDPAT_O	Lower 16-bit of VidPattern(15): (default = 16#ff7f)
b	R/W	VIDPAT_P	Lower 16-bit of VidPattern(16): (default = 16#ffff)

8.12 Alpha Blending Registers

ALPHA BLENDING																																Offset: 0_2A, Index: 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
Mobility																									d					c	b	a

a	R/W	ALPHA_BLD_EN	Enable the alpha blending of video and graphics: (default = 0) 0 = Disable alpha blending 1 = Enable alpha blending
b	R/W	YCBCR_BYPASS	Enable the RGB to YcrCb conversion: (default = 0) 0 = Enable RGB2YcbCr 1 = Bypass RGB2YcbCr
c	R/W	ALPHA_LOAD	Update the alpha value: (default = 0) 0 = Not update alpha value 1 = Load the new alpha value
d	R/W	ALPHA_VALUE	Alpha values indicating the scale of the blending of each component

8.13 Portrait Display Registers

PORTRAIT_GEN_CNTL																												Offset: 0_2A, Index: 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
Mobility						d												c												b	a

a	R/W	PORTRAIT_EN	0 = Disable portrait display feature (default) 1 = Enable portrait display
b	R/W	ROTATE_CLKW	0 = Indicate portrait coordinates were rotated 90 degrees counter clockwise from landscape coordinates 1 = Indicate portrait coordinates were rotated 90 degrees clockwise from landscape coordinates (default)
c	R/W	PORTRAIT_DISP_PITCH	Portrait display pitch in pixels. It is assumed to be on Dword boundaries
d	R/W	PORTRAIT_DISP_HEIGHT	Portrait display height in lines

Usage

Linear aperture access in portrait mode is restricted to the following cases:

8 bpp: byte, word, dword access

15/16 bpp: word, dword access

24 bpp: dword access

32 bpp: dword access

In addition, word or dword access cannot be done on odd address in portrait mode. Unsupported modes will read/write data in wrong address location.

This page intentionally left blank.

Appendix A
Revision History

A.1 Rev 0.01, RR42001.pdf (Nov 1998)

- Preliminary release.

A.2 Rev 0.02, RR42002.pdf (Apr 1999)

This document conforms to Eng. Spec. OBS-C032AC2, Rev. Level AAAD (3/31/99).

- Added MEM_DQML_DRIVE and MEM_DQML_SLEWRATE bits in CUSTOM_MACRO_CNTL to split up DQM signals controls into upper and lower regions.
- CONFIG_STAT0, moved bit 21, MACROVISION_ENABLE, to bit 22 to match hardware.
- Updated HW_DEBUG with additions of DON'T_RST_PCLK, PM_D3_SUPPORT_ENABLE, DON'T_RST_CHAREN, HOSTRA_SET_EN, and AUTOEXP_HORZ_FIX.
- Changed GEN_TEST_CNTL[31:24], GEN_DEBUG_MODE 8x to second dispeng from palette read/write.
- Updated APLL_STRAPS register to use default values.
- Use HW_DEBUG[1] as SYNC_PD_EN bit to pull down HSYNC/VSYNC pads.
- Use HW_DEBUG[19] as MCLK_START_EN to enable the fix on MCLK stopping bug.
- Put in comment for bit 24-25 of CONFIG_PANEL register.
- Updated comments for bit 3 in LCD_GEN_CTRL.
- Added CHIP_REV table in COFIG_CHIP_ID register.
- HW_DEBUG[21] will dissable/enable cmdfifo2 and parser.
- IDCT_EN@ALPHA_TST_CNTL description updated.
- Removed PM_D3_SUPPORT_ENABLE bit (POWER_MANAGEMENT[27]) and put it as reserved.

- Added AGPPLL_FIX_EN bit to HW_DEBUG[3] to enable/disable fix for AGPPLL to output ‘0’ rather than ‘1’ when being reset and not bypassed (for suspend current).
- CONFIG_PANEL[22], changed to reserved. Should be programmed to zero at all times.
- POWER_MANAGEMENT[10], changed name to DONT_USE_F32KHZ and its definition.

A.3 Rev 1.00, RR42100C.pdf (July 1999)

This document conforms to Eng. Spec. OBS-C032AC2, Rev. Level AAAG (6/19/99).

- Added PM_D3_RST_ENb bit in POWER_MANAGEMENT[27] to initiate a reset when exiting D3 state (Chapter 8)
- Changed BUS_CNTL(9:8) to read only for driver/BIOS to differentiate A3x from A2x (Chapter 4).
- Updated MEM_ADDR_CONFIG to support 512Kx32x4 memory (Chapter 4).

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

Z