mach64
Register Reference Guide

Technical Reference Manuals

P/N: RRG-S00700-05

ATI Technologies Inc.
33 Commerce Valley Drive East
Thornhill, Ontario
Canada L3T 7N6
Developer Support: 905-882-2600 ext.6000
Offices: 905-882-2600
Fax: 905-882-2620
BBS: 905-764-9404

P/N: RRG-S00700-05
RELEASE 5.0

Systems Publication Index

Technical Manuals

- ***mach64*** Graphics Controller Specifications (GCS-C015XX1-05)
- ***mach64*** Programmer's Guide (PRG-S00700-05)
- ***mach64*** VGA Register Guide (VGA-S00700-05)
- ***mach64*** Register Reference (RRG-S00700-05)
- ***mach64*** BIOS Kit (BIO-C012XX1-05)

Record of Revisions

Release	Date	Description of changes
01	93 Dec	PRELIMINARY
03	94 Feb	BETA RELEASE
04	94 May	BETA RELEASE
05	94 Oct	RELEASE

Contents

This page intentionally left blank

Chapter 1
Register Classifications

Introduction

The ***mach64*** accelerator has four major register classes:

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

The on-chip VGA register descriptions can be found in the ***mach64*** **VGA Register Guide**.

Setup and Control Registers

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

- Scratch registers are used for general purpose storage for the adapter ROM and for communicating the adapter ROM segment location to host applications. In test modes, these registers are used for chip diagnostics.
- Bus control registers are used to configure the on-chip bus interface unit.
- Memory control registers are used to configure the memory interface unit.
- Test registers are used for chip diagnostics.
- Configuration registers are used for aperture configuration and reading the current board configuration.

Accelerator CRTC and DAC Registers

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

- CRTC registers are used to configure the video mode.
- Clock control registers are used to configure the pixel clock.
- DAC control registers are used to configure the DAC.
- Overscan registers are used to configure overscan borders.
- Hardware cursor registers are used to define and move the hardware cursor.

Draw Engine Control Registers

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

- Host data registers are used for transferring data from the host to the draw engine.
- Pattern registers are used to enable and define fixed patterns.
- Scissor registers are used to define a draw region.
- Data path registers are used to configure the data path and ALU.
- Color compare registers are used to configure the source or destination color compare.
- FIFO status registers are used to report the status of the command FIFO.
- Context control registers are used to load contexts or execute context chains.
- Engine control registers are abbreviated composites of other draw engine control registers.
- Engine status registers report the current state of the draw engine.

Draw Engine Trajectory Registers

Draw engine trajectory registers are memory mapped. They set up the source and destination trajectories and initiate draw operations.

- Destination trajectory registers define the destination trajectory.
- Source trajectory registers define the blit source trajectory.

How To Find The Registers

- Registers are listed alphabetically, by register name. Register mnemonics are shown in the top outside margin of each page.
- The tables in the next chapter, *Cross Reference*, summarize all registers by class, mnemonic and address, and indicate the page number where each register is described.

Register Mapping

All registers not associated with the draw engine are I/O mapped, and all except CONFIG_CNTL have memory mapped register aliases. All registers are 32 bits wide, except for DAC_REGS, which are 4x8 bit registers.

- **If the small apertures are enabled**, memory mapped registers may be accessed through a 1K area at segment:offset of 0xB000:0xFC00.
- **If the big aperture is enabled**, the memory mapped registers occupy the address space located at the base address of the aperture, plus an offset of 0x3FFC00 for a 4M aperture or 0x7FFC00 for an 8M aperture configuration.
- **I/O mapped registers** are selected by the top 6 bits of the I/O address. The bottom 10 bits are the I/O base address, which may be 2EC, 1CC, or 1C8.

This page intentionally left blank

Chapter 2
Cross Reference

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

Registers by Classification

REGISTER CLASS	Mnemonic	Read /Write	I/O Select	DWORD Offset	PAGE
Direct Write Registers					
ACCELERATOR CRTC AND DAC					
Clock Control	CLOCK_CNTL	R/W	12h	24h	*3-4*
CRTC	CRTC_GEN_CNTL	R/W	7h	7h	*3-17*
	CRTC_H_SYNC_STRT_WID	R/W	1h	1h	*3-19*
	CRTC_H_TOTAL_DISP	R/W	0h,1Fh	0h	*3-20*
	CRTC_INT_CNTL	R/W	6h	6h	*3-21*
	CRTC_OFF_PITCH	R/W	5h	5h	*3-22*
	CRTC_V_SYNC_STRT_WID	R/W	3h	3h	*3-23*
	CRTC_V_TOTAL_DISP	R/W	2h	2h	*3-24*
	CRTC_VLINE_CRNT_VLINE	R/W	4h	4h	*3-25*
DAC Control	DAC_CNTL	R/W	18h	31h	*3-31*
	DAC_REGS	R/W	17h	30h	*3-33*
Hardware Cursor	CUR_CLR0	R/W	Bh	18h	*3-26*
	CUR_CLR1	R/W	Ch	19h	*3-27*
	CUR_HORZ_VERT_OFF	R/W	Fh	1Ch	*3-28*
	CUR_HORZ_VERT_POSN	R/W	Eh	1Bh	*3-29*
	CUR_OFFSET	R/W	Dh	1Ah	*3-30*
Overscan	OVR_CLR	R/W	8h	10h	*3-72*
	OVR_WID_LEFT_RIGHT	R/W	9h	11h	*3-73*
	OVR_WID_TOP_BOTTOM	R/W	Ah	12h	*3-74*
SETUP AND CONTROL					
Bus Control	BUS_CNTL	R/W	13h	28h	*3-2*
Configuration	CONFIG_CHIP_ID	R	18h	38h	*3-8*
	CONFIG_CNTL	R/W	1Ah	-	*3-9*
	CONFIG_STAT0	R	1Ch	39h	*3-12*
	CONFIG_STAT1	R	1Dh	3Ah	*3-14*
Memory Control	MEM_CNTL	R/W	14h	2Ch	*3-67*
	MEM_VGA_RP_SEL	R/W	16h	2Eh	*3-70*
	MEM_VGA_WP_SEL	R/W	15h	2Dh	*3-71*
Scratch Pad	SCRATCH_REG0	R/W	10h	20h	*3-84*
	SCRATCH_REG1	R/W	11h	21h	*3-85*
Test	GEN_TEST_CNTL	R/W	19h	34h	*3-57*
	TEST_REG0 (Test Mode 1)	R/W	10h	20h	*3-101*
	TEST_REG1 (Test Mode 1)	R/W	11h	21h	*3-102*
	TEST_REG2 (Test Mode 2)	R	10h	20h	*3-103*
	TEST_REG3 (Test Mode 2)	R	11h	21h	*3-104*
	TEST_REG4 (Test Mode 3)	R	10h	20h	*3-105*
	TEST_REG5 (Test Mode 4)	R/W	10h	20h	*3-106*
	TEST_REG6 (Test Mode 4)	R	11h	21h	*3-107*
	TEST_REG7 (Test Mode 5)	R/W	10h	20h	*3-109*
FIFOed Registers					
DRAW ENGINE CONTROL					
Color Compare	CLR_CMP_CLR	R/W	-	C0h	*3-5*
	CLR_CMP_CNTL	R/W	-	C2h	*3-6*
	CLR_CMP_MSK	R/W	-	C1h	*3-7*
Context Control	CONTEXT_LOAD_CNTL	R/W	-	CBh	*3-15*
	CONTEXT_MSK	R/W	-	C8h	*3-16*

Registers by Classification

REGISTER CLASS	Mnemonic	Read /Write	I/O Select	DWORD Offset	PAGE
DRAW ENGINE CONTROL (cont'd)					
Data Path	DP_BKGD_CLR	R/W	-	B0h	*3-34*
	DP_CHAIN_MSK	R/W	-	B3h	*3-35*
	DP_FRGD_CLR	R/W	-	B1h	*3-36*
	DP_MIX	R/W	-	B5h	*3-37*
	DP_PIX_WIDTH	R/W	-	B4h	*3-39*
	DP_SRC	R/W	-	B6h	*3-40*
	DP_WRITE_MSK	R/W	-	B2h	*3-41*
Engine Status	GUI_STAT	R	-	CEh	*3-61*
	GUI_TRAJ_CNTL	R/W	-	CCh	*3-62*
FIFO Status	FIFO_STAT	R	-	C4h	*3-56*
Host Data	HOST_CNTL	R/W	-	90h	*3-64*
	HOST_DATA[0:15]	W	-	80-8Fh	*3-65*
Pattern	PAT_CNTL	R/W	-	A2h	*3-75*
	PAT_REG0	R/W	-	A0h	*3-76*
	PAT_REG1	R/W	-	A1h	*3-77*
Scissor	SC_BOTTOM	R/W	-	ACh	*3-78*
	SC_LEFT	R/W	-	A8h	*3-79*
	SC_LEFT_RIGHT	W	-	AAh	*3-80*
	SC_RIGHT	R/W	-	A9h	*3-81*
	SC_TOP	R/W	-	ABh	*3-82*
	SC_TOP_BOTTOM	W	-	ADh	*3-83*
DRAW ENGINE TRAJECTORY					
Destination Draw Engine	DST_BRES_DEC	R/W	-	4Bh	*3-42*
	DST_BRES_ERR	R/W	-	49h	*3-43*
	DST_BRES_INC	R/W	-	4Ah	*3-44*
	DST_BRES_LNTH	R/W	-	48h	*3-45*
	DST_CNTL	R/W	-	4Ch	*3-46*
	DST_HEIGHT	R/W	-	45h	*3-48*
	DST_HEIGHT_WIDTH	W	-	46h	*3-49*
	DST_OFF_PITCH	R/W	-	40h	*3-50*
	DST_WIDTH	R/W	-	44h	*3-51*
	DST_X	R/W	-	41h	*3-52*
	DST_X_WIDTH	W	-	47h	*3-53*
	DST_Y	R/W	-	42h	*3-54*
	DST_Y_X	W	-	43h	*3-55*
Source Draw Engine	SRC_CNTL	R/W	-	6Dh	*3-86*
	SRC_HEIGHT1	R/W	-	65h	*3-88*
	SRC_HEIGHT1_WIDTH1	W	-	66h	*3-89*
	SRC_HEIGHT2	R/W	-	6Bh	*3-90*
	SRC_HEIGHT2_WIDTH2	W	-	6Ch	*3-91*
	SRC_OFF_PITCH	R/W	-	60h	*3-92*
	SRC_WIDTH1	R/W	-	64h	*3-93*
	SRC_WIDTH2	R/W	-	6Ah	*3-94*
	SRC_X	R/W	-	61h	*3-95*
	SRC_X_START	R/W	-	67h	*3-96*
	SRC_Y	R/W	-	62h	*3-97*
	SRC_Y_START	R/W	-	68h	*3-98*
	SRC_Y_X	W	-	63h	*3-99*
	SRC_Y_X_START	W	-	69h	*3-100*

Registers by Address

REGISTER CLASS	Mnemonic	Read /Write	I/O Select	DWORD Offset	PAGE
CRTC	CRTC_H_TOTAL_DISP	R/W	0h,1Fh	0h	*3-20*
	CRTC_H_SYNC_STRT_WID	R/W	1h	1h	*3-19*
	CRTC_V_TOTAL_DISP	R/W	2h	2h	*3-24*
	CRTC_V_SYNC_STRT_WID	R/W	3h	3h	*3-23*
	CRTC_VLINE_CRNT_VLINE	R/W	4h	4h	*3-25*
	CRTC_OFF_PITCH	R/W	5h	5h	*3-22*
	CRTC_INT_CNTL	R/W	6h	6h	*3-21*
	CRTC_GEN_CNTL	R/W	7h	7h	*3-17*
Overscan	OVR_CLR	R/W	8h	10h	*3-72*
	OVR_WID_LEFT_RIGHT	R/W	9h	11h	*3-73*
	OVR_WID_TOP_BOTTOM	R/W	Ah	12h	*3-74*
Hardware Cursor	CUR_CLR0	R/W	Bh	18h	*3-26*
	CUR_CLR1	R/W	Ch	19h	*3-27*
	CUR_OFFSET	R/W	Dh	1Ah	*3-30*
	CUR_HORZ_VERT_POSN	R/W	Eh	1Bh	*3-29*
	CUR_HORZ_VERT_OFF	R/W	Fh	1Ch	*3-28*
Scratch Pad and Test	SCRATCH_REG0	R/W	10h	20h	*3-84*
	TEST_REG0 (Test Mode 1)	R/W	10h	20h	*3-101*
	TEST_REG2 (Test Mode 2)	R	10h	20h	*3-103*
	TEST_REG4 (Test Mode 3)	R	10h	20h	*3-105*
	TEST_REG5 (Test Mode 4)	R/W	10h	20h	*3-106*
	TEST_REG6 (Test Mode 4)	R	11h	21h	*3-107*
	SCRATCH_REG1	R/W	11h	21h	*3-85*
	TEST_REG1 (Test Mode 1)	R/W	11h	21h	*3-102*
	TEST_REG3 (Test Mode 2)	R	11h	21h	*3-104*
	TEST_REG7 (Test Mode 5)	R/W	10h	20h	*3-109*
Clock Control	CLOCK_CNTL	R/W	12h	24h	*3-4*
Bus Control	BUS_CNTL	R/W	13h	28h	*3-2*
Memory Control	MEM_CNTL	R/W	14h	2Ch	*3-67*
	MEM_VGA_WP_SEL	R/W	15h	2Dh	*3-71*
	MEM_VGA_RP_SEL	R/W	16h	2Eh	*3-70*
DAC Control	DAC_REGS	R/W	17h	30h	*3-33*
	DAC_CNTL	R/W	18h	31h	*3-31*
Test	GEN_TEST_CNTL	R/W	19h	34h	*3-57*
Configuration	CONFIG_CNTL	R/W	1Ah	-	*3-9*
	CONFIG_CHIP_ID	R	1Bh	38h	*3-8*
	CONFIG_STAT0	R	1Ch	39h	*3-12*
	CONFIG_STAT1	R	1Dh	3Ah	*3-14*
Destination Draw Engine	DST_OFF_PITCH	R/W	-	40h	*3-50*
	DST_X	R/W	-	41h	*3-52*
	DST_Y	R/W	-	42h	*3-54*
	DST_Y_X	W	-	43h	*3-55*
	DST_WIDTH	R/W	-	44h	*3-51*
	DST_HEIGHT	R/W	-	45h	*3-48*
	DST_HEIGHT_WIDTH	W	-	46h	*3-49*
	DST_X_WIDTH	W	-	47h	*3-53*
	DST_BRES_LNTH	R/W	-	48h	*3-45*
	DST_BRES_ERR	R/W	-	49h	*3-43*
	DST_BRES_INC	R/W	-	4Ah	*3-44*
	DST_BRES_DEC	R/W	-	4Bh	*3-42*
	DST_CNTL	R/W	-	4Ch	*3-46*

Registers by Address

REGISTER CLASS	*Mnemonic*	*Read /Write*	*I/O Select*	*DWORD Offset*	*PAGE*
Source Draw Engine	SRC_OFF_PITCH	R/W	-	60h	*3-92*
	SRC_X	R/W	-	61h	*3-95*
	SRC_Y	R/W	-	62h	*3-97*
	SRC_Y_X	W	-	63h	*3-99*
	SRC_WIDTH1	R/W	-	64h	*3-93*
	SRC_HEIGHT1	R/W	-	65h	*3-88*
	SRC_HEIGHT1_WIDTH1	W	-	66h	*3-89*
	SRC_X_START	R/W	-	67h	*3-96*
	SRC_Y_START	R/W	-	68h	*3-98*
	SRC_Y_X_START	W	-	69h	*3-100*
	SRC_WIDTH2	R/W	-	6Ah	*3-94*
	SRC_HEIGHT2	R/W	-	6Bh	*3-90*
	SRC_HEIGHT2_WIDTH2	W	-	6Ch	*3-91*
	SRC_CNTL	R/W	-	6Dh	*3-86*
Host Data	HOST_DATA[0:15]	W	-	80-8Fh	*3-65*
	HOST_CNTL	R/W	-	90h	*3-64*
Pattern	PAT_REG0	R/W	-	A0h	*3-76*
	PAT_REG1	R/W	-	A1h	*3-77*
	PAT_CNTL	R/W	-	A2h	*3-75*
Scissor	SC_LEFT	R/W	-	A8h	*3-79*
	SC_RIGHT	R/W	-	A9h	*3-81*
	SC_LEFT_RIGHT	W	-	AAh	*3-80*
	SC_TOP	R/W	-	ABh	*3-82*
	SC_BOTTOM	R/W	-	ACh	*3-78*
	SC_TOP_BOTTOM	W	-	ADh	*3-83*
Data Path	DP_BKGD_CLR	R/W	-	B0h	*3-34*
	DP_FRGD_CLR	R/W	-	B1h	*3-36*
	DP_WRITE_MSK	R/W	-	B2h	*3-41*
	DP_CHAIN_MSK	R/W	-	B3h	*3-35*
	DP_PIX_WIDTH	R/W	-	B4h	*3-39*
	DP_MIX	R/W	-	B5h	*3-37*
	DP_SRC	R/W	-	B6h	*3-40*
Color Compare	CLR_CMP_CLR	R/W	-	C0h	*3-5*
	CLR_CMP_MSK	R/W	-	C1h	*3-7*
	CLR_CMP_CNTL	R/W	-	C2h	*3-6*
FIFO Status	FIFO_STAT	R	-	C4h	*3-56*
Context Control	CONTEXT_MSK	R/W	-	C8h	*3-16*
	CONTEXT_LOAD_CNTL	R/W	-	CBh	*3-15*
Engine Status	GUI_TRAJ_CNTL	R/W	-	CCh	*3-62*
	GUI_STAT	R	-	CEh	*3-61*

Registers Alphabetically by Mnemonic

REGISTER CLASS	Mnemonic	Read /Write	I/O Select	DWORD Offset	PAGE
Bus Control	BUS_CNTL	R/W	13h	28h	*3-2*
Clock Control	CLOCK_CNTL	R/W	12h	24h	*3-4*
Color Compare	CLR_CMP_CLR	R/W	-	C0h	*3-5*
	CLR_CMP_CNTL	R/W	-	C2h	*3-6*
	CLR_CMP_MSK	R/W	-	C1h	*3-7*
Configuration	CONFIG_CHIP_ID	R	1Bh	38h	*3-8*
	CONFIG_CNTL	R/W	1Ah	-	*3-9*
	CONFIG_STAT0	R	1Ch	39h	*3-12*
	CONFIG_STAT1	R	1Dh	3Ah	*3-14*
Context Control	CONTEXT_LOAD_CNTL	R/W	-	CBh	*3-15*
	CONTEXT_MSK	R/W	-	C8h	*3-16*
CRTC	CRTC_GEN_CNTL	R/W	7h	7h	*3-17*
	CRTC_H_SYNC_STRT_WID	R/W	1h	1h	*3-19*
	CRTC_H_TOTAL_DISP	R/W	0h,1Fh	0h	*3-20*
	CRTC_INT_CNTL	R/W	6h	6h	*3-21*
	CRTC_OFF_PITCH	R/W	5h	5h	*3-22*
	CRTC_V_SYNC_STRT_WID	R/W	3h	3h	*3-23*
	CRTC_V_TOTAL_DISP	R/W	2h	2h	*3-24*
	CRTC_VLINE_CRNT_VLINE	R/W	4h	4h	*3-25*
Hardware Cursor	CUR_CLR0	R/W	Bh	18h	*3-26*
	CUR_CLR1	R/W	Ch	19h	*3-27*
	CUR_HORZ_VERT_OFF	R/W	Fh	1Ch	*3-28*
	CUR_HORZ_VERT_POSN	R/W	Eh	1Bh	*3-29*
	CUR_OFFSET	R/W	Dh	1Ah	*3-30*
DAC Control	DAC_CNTL	R/W	18h	31h	*3-31*
	DAC_REGS	R/W	17h	30h	*3-33*
Data Path	DP_BKGD_CLR	R/W	-	B0h	*3-34*
	DP_CHAIN_MSK	R/W	-	B3h	*3-35*
	DP_FRGD_CLR	R/W	-	B1h	*3-36*
	DP_MIX	R/W	-	B5h	*3-37*
	DP_PIX_WIDTH	R/W	-	B4h	*3-39*
	DP_SRC	R/W	-	B6h	*3-40*
	DP_WRITE_MSK	R/W	-	B2h	*3-41*
Destination Draw Engine	DST_BRES_DEC	R/W	-	4Bh	*3-42*
	DST_BRES_ERR	R/W	-	49h	*3-43*
	DST_BRES_INC	R/W	-	4Ah	*3-44*
	DST_BRES_LNTH	R/W	-	48h	*3-45*
	DST_CNTL	R/W	-	4Ch	*3-46*
	DST_HEIGHT	R/W	-	45h	*3-48*
	DST_HEIGHT_WIDTH	W	-	46h	*3-49*
	DST_OFF_PITCH	R/W	-	40h	*3-50*
	DST_WIDTH	R/W	-	44h	*3-51*
	DST_X	R/W	-	41h	*3-52*
	DST_X_WIDTH	W	-	47h	*3-53*
	DST_Y	R/W	-	42h	*3-54*
	DST_Y_X	W	-	43h	*3-55*
FIFO Status	FIFO_STAT	R	-	C4h	*3-56*
Test	GEN_TEST_CNTL	R/W	19h	34h	*3-57*
Engine Status	GUI_STAT	R	-	CEh	*3-61*
	GUI_TRAJ_CNTL	R/W	-	CCh	*3-62*

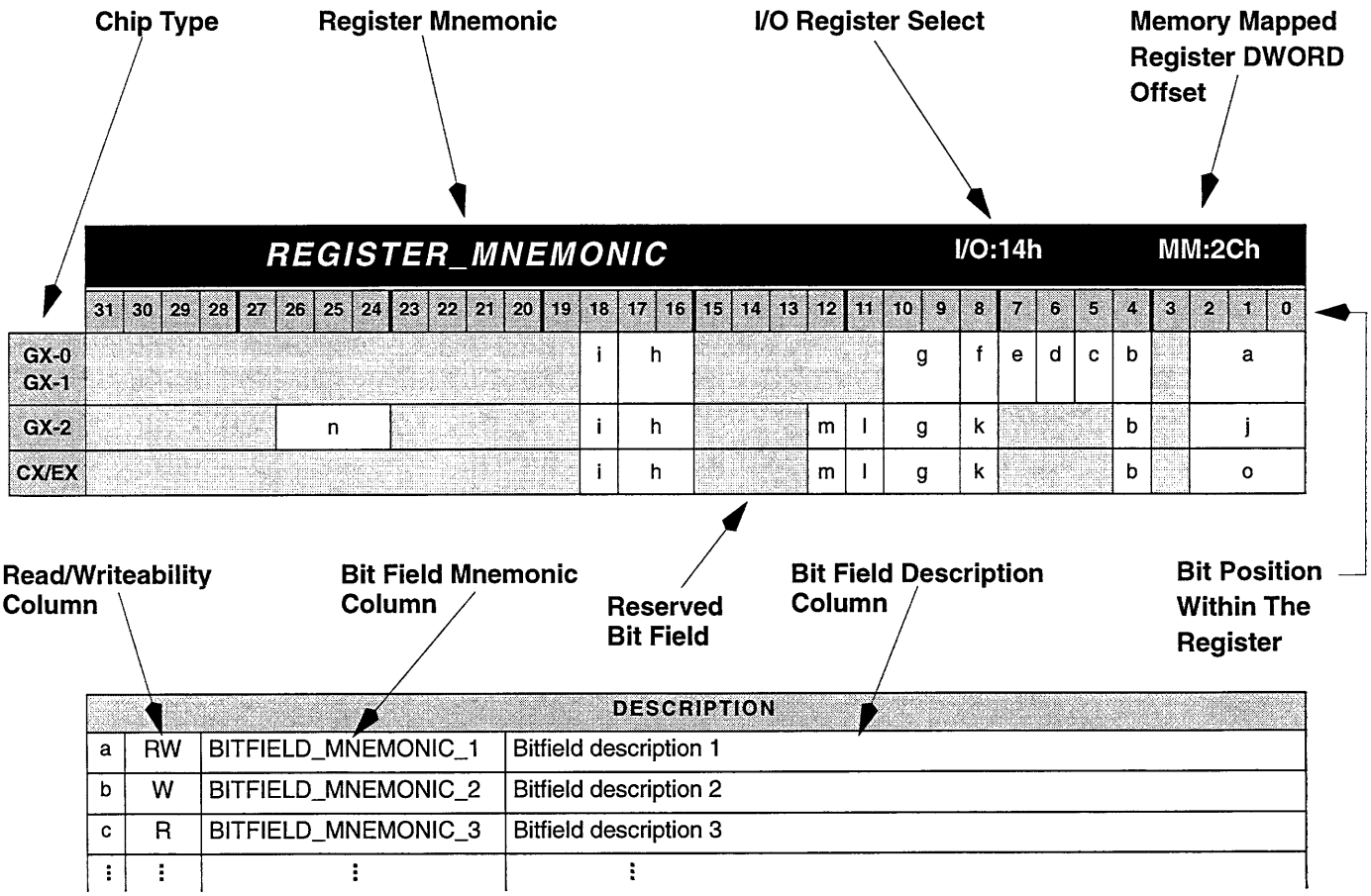
Registers Alphabetically by Mnemonic

REGISTER CLASS	Mnemonic	Read /Write	I/O Select	DWORD Offset	PAGE
Host Data	HOST_CNTL	R/W	-	90h	*3-64*
	HOST_DATA[0:15]	W	-	80-8Fh	*3-65*
Memory Control	MEM_CNTL	R/W	14h	2Ch	*3-67*
	MEM_VGA_RP_SEL	R/W	16h	2Eh	*3-70*
	MEM_VGA_WP_SEL	R/W	15h	2Dh	*3-71*
Overscan	OVR_CLR	R/W	8h	10h	*3-72*
	OVR_WID_LEFT_RIGHT	R/W	9h	11h	*3-73*
	OVR_WID_TOP_BOTTOM	R/W	Ah	12h	*3-74*
Pattern	PAT_CNTL	R/W	-	A2h	*3-75*
	PAT_REG0	R/W	-	A0h	*3-76*
	PAT_REG1	R/W	-	A1h	*3-77*
Scratch Pad	SCRATCH_REG0	R/W	10h	20h	*3-84*
	SCRATCH_REG1	R/W	11h	21h	*3-85*
Scissor	SC_BOTTOM	R/W	-	ACh	*3-78*
	SC_LEFT	R/W	-	A8h	*3-79*
	SC_LEFT_RIGHT	W	-	AAh	*3-80*
	SC_RIGHT	R/W	-	A9h	*3-81*
	SC_TOP	R/W	-	ABh	*3-82*
	SC_TOP_BOTTOM	W	-	ADh	*3-83*
Source Draw Engine	SRC_CNTL	R/W	-	6Dh	*3-86*
	SRC_HEIGHT1	R/W	-	65h	*3-88*
	SRC_HEIGHT1_WIDTH1	W	-	66h	*3-89*
	SRC_HEIGHT2	R/W	-	6Bh	*3-90*
	SRC_HEIGHT2_WIDTH2	W	-	6Ch	*3-91*
	SRC_OFF_PITCH	R/W	-	60h	*3-92*
	SRC_WIDTH1	R/W	-	64h	*3-93*
	SRC_WIDTH2	R/W	-	6Ah	*3-94*
	SRC_X	R/W	-	61h	*3-95*
	SRC_X_START	R/W	-	67h	*3-96*
	SRC_Y	R/W	-	62h	*3-97*
	SRC_Y_START	R/W	-	68h	*3-98*
	SRC_Y_X	W	-	63h	*3-99*
	SRC_Y_X_START	W	-	69h	*3-100*
Test	TEST_REG0 (Test Mode 1)	R/W	10h	20h	*3-101*
	TEST_REG1 (Test Mode 1)	R/W	11h	21h	*3-102*
	TEST_REG2 (Test Mode 2)	R	10h	20h	*3-103*
	TEST_REG3 (Test Mode 2)	R	11h	21h	*3-104*
	TEST_REG4 (Test Mode 3)	R	10h	20h	*3-105*
	TEST_REG5 (Test Mode 4)	R/W	10h	20h	*3-106*
	TEST_REG6 (Test Mode 4)	R	11h	21h	*3-107*
	TEST_REG7 (Test Mode 5)	R/W	10h	20h	*3-109*

This page intentionally left blank.

Chapter 3
Register Reference

The table below describes the layout of all the Register Reference Tables within Chapter 3.



BUS_CNTL																															
	I/O:13h																MM:28h														
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GX-0 GX-1 GX-2	p		o	n	m	l			k	j	i	h	g					f	e	d	c				b				a			
CX/EX	p		o	n	m	l			k	j	i	h	g					f		d	c				b				a			
CT	r		q	n					k	j	i	h	g					f		d	c				b				a			

DESCRIPTION			
a	RW	BUS_WS	Bus wait states (Default = Fh)
b	RW	BUS_ROM_WS	ROM access wait states (Default = Fh)
c	RW	BUS_ROM_PAGE	ROM page select
d	RW	BUS_ROM_DIS	ROM disabled (Default = 0)
e	RW	BUS_IO_16_EN	16-bit I/O enabled 0 = 8-bit I/O 1 = 16-bit I/O
f	RW	BUS_DAC_SNOOP_EN	DAC snooping enabled
g	RW	BUS_FIFO_WS	Maximum wait states that the FIFO can generate when it is full — if maximum is exceeded, it sets BUS_FIFO_ERR_INT
h	RW	BUS_FIFO_ERR_INT_EN	Command FIFO error interrupt enabled (Default = 0)
i	R	BUS_FIFO_ERR_INT	Command FIFO error interrupt
	W	BUS_FIFO_ERR_ACK	Command FIFO error acknowledge
j	RW	BUS_HOST_ERR_INT_EN	Command FIFO host data error interrupt enabled (Default = 0)
k	R	BUS_HOST_ERR_INT	Command FIFO host data error interrupt
	W	BUS_HOST_ERR_ACK	Command FIFO host data error acknowledge
l	RW	BUS_PCI_DAC_WS	DAC access wait states (PCI bus specific)
m	RW	BUS_PCI_DAC_DLY	DAC access delayed (PCI bus specific)
n	RW	BUS_PCI_MEMW_WS	Wait state select for memory writes (Default = 0, PCI bus specific) 0 = 0 wait state 1 = 1 wait state
o	RW	BUS_PCI_BURST_DEC	Decrement addressing for burst memory writes (PCI bus specific) 0 = increment addressing 1 = decrement addressing
p	RW	BUS_RDY_READ_DLY	Bus RDY delay control for memory read operations 0 = RDY early by 1 memory clock 1 = No RDY delay 2 = RDY delayed by 1 memory clock 3 = RDY delayed by 2 memory clocks
q	RW	BUS_BURST	Enable burst write transfers (default = 0) 0 = write burst transfers disabled 1= write bursts enabled

(Continued on next page)

BUS_CNTL																		I/O:13h								MM:28h			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GX-0 GX-1 GX-2	p		o	n	m	l			k	j	i	h	g					f	e	d	c				b				a			
CX/EX	p		o	n	m	l			k	j	i	h	g					f		d	c				b				a			
CT	r		q	n					k	j	i	h	g					f		d	c				b				a			

DESCRIPTION			
r	RW	BUS_RDY_READ_DLY	Bus memory read RDY signal delay (default = 2) 0 = *Reserved* 1 = no RDY delay 2 = RDY delayed by 1 memory clock 3 = *Reserved*

Description

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

Usage

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

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

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

See Also

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

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

CLOCK_CNTL																				I/O:12h								MM:24h			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GX-0 GX-1 GX-2 CX/EX																								e	d	c	b		a		
CT									h										g				f		c			a			

DESCRIPTION			
a	RW	CLOCK_SEL	Video clock frequency select
b	RW	CLOCK_DIV	Video clock frequency divider select 0 = Divide-by-1 1 = Divide-by-2 2 = Divide-by-4 3 = *Reserved*
c	W	CLOCK_STROBE	Clock frequency select activation
d	RW	CLOCK_SERIAL_DATA_EN	Enables serial data to be read from programmable clock synthesizer
e	RW	CLOCK_SERIAL_DATA	Serial data for programmable clock synthesizer
f	RW	PLL_WR_EN	PLL register write enable. High enables writing to PLL register as set by PLL_ADDR and PLL_DATA. Low latches PLL_ADDR for following read cycles.
g	RW	PLL_ADDR	PLL register address. Selects register in PLL to read or write
h	RW	PLL_DATA	PLL data. If PLL_WR_EN is high then data written to this field is written to the register indexed by PLL_ADDR, else no write occurs (PLL_DATA ignored). If PLL_WR_EN is low then PLL_DATA will contain the contents of the PLL register set by the last write to PLL_ADDR.

Description

CLOCK_CNTL is used to select a pixel clock for the current video mode. An ATI1881x type clock chip will provide 16 fixed or programmable frequencies, selectable with CLOCK_SEL and divisible with CLOCK_DIV. Fully programmable clock synthesizers are programmed serially using CLOCK_SERIAL_DATA_EN, CLOCK_SERIAL_DATA, and CLOCK_STROBE. Consult the manufacturer's clock chip reference.

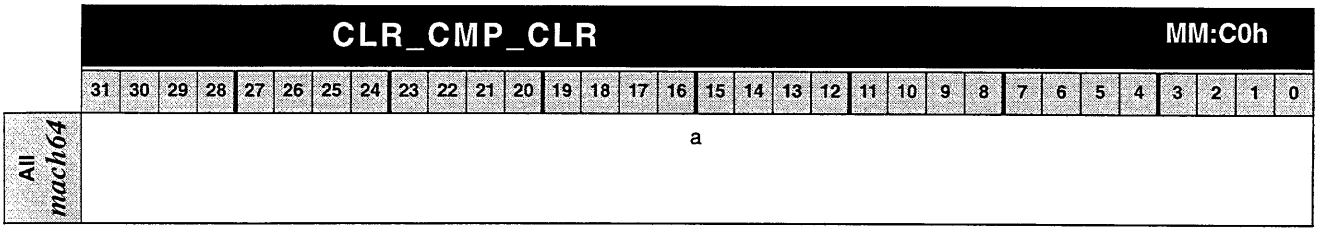
Usage

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

See Also

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

- *Programming Model: Mode Switching: Manual Mode Switching*
- *Programming Model: Mode Switching: Designing a Custom CRT Mode*
- *Appendix C: CRTC ParametersAppendix*
- *D: Clock Chip Reference*
- *Programming the Internal PLL*



DESCRIPTION			
a	RW	CLR_CMP_CLR	Color comparison color

Description

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

Usage

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

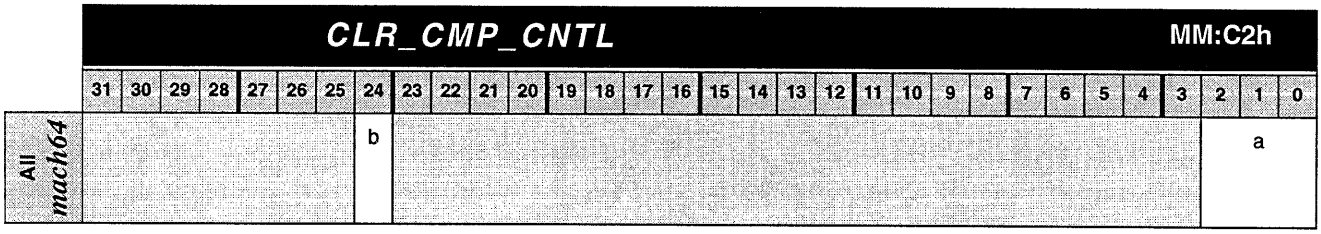
See Also

CLR_CMP_CNTL on page 3-6

CLR_CMP_MSK on page 3-7

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

- *Advanced Topics I: Transparent Blits*



			DESCRIPTION
a	RW	CLR_CMP_FN	Color comparison function 0 = FALSE 1 = TRUE 2 = *Reserved* 3 = *Reserved* 4 = DST_CLR != CLR_CMP_CLR 5 = DST_CLR = CLR_CMP_CLR 6 = *Reserved* 7 = *Reserved*
b	RW	CLR_CMP_SRC	Source for color keying 0 = Destination 1 = Source

Description

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

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

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

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

Usage

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

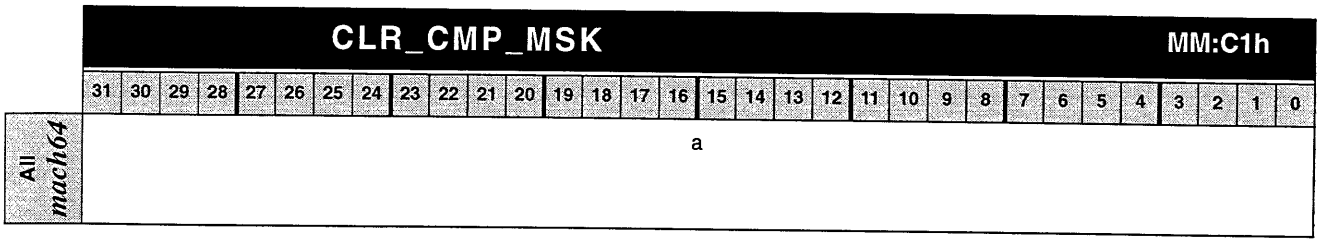
See Also

CLR_CMP_CLR on page 3-5

CLR_CMP_MSK on page 3-7

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

- *Programming Model: Logical Pixel Data Path*
- *Advanced Topics I: Transparent Blits*



DESCRIPTION			
a	RW	CLR_CMP_MSK	Color comparison mask

Description

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

Usage

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

See Also

CLR_CMP_CLR on page 3-5

CLR_CMP_CNTL on page 3-6

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

- *Advanced Topics I: Transparent Blits*

CONFIG_CHIP_ID																I/O:1Bh												MM:38h			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
All *mach64*	c							b								a															

		DESCRIPTION	
a	R	CFG_CHIP_TYPE	Product type code
b	R	CFG_CHIP_CLASS	Product class code
c	R	CFG_CHIP_REV	Production revision number

Description

CONFIG_CHIP_ID is a read-only register. It returns the revision details of the queried chip. CFG_CHIP_TYPE is an alphanumeric code; CFG_CHIP_CLASS and CFG_CHIP_REV are numeric codes. Known chip IDs are listed in the table below:

ASIC Identification			
ASIC Designation	CFG_CHIP_TYPE	CFG_CHIP_CLASS	CFG_CHIP_REV
GX-0	'GX'=0xD7	0x00	0x00
GX-1	'GX'=0xD7	0x00	0x01
GX-2	'GX'=0xD7	0x00	0x02
CX	'CX'=0x57	0x00	0x03
EX	'EX'=0x97	0x00	0x03
CT	'CT'=0x53	0x00	0x00

The CFG_CHIP_TYPE field is a translation of two uppercase ASCII characters. Each character is represented by a number from 0-25 and occupies 5 bits within the field. The upper 6 bits of CFG_CHIP_TYPE are set to zero.

Usage

This register is used for chip revision identification.

See Also

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

- *Programming Model: **mach64** Detection*

CONFIG_CNTL	I/O:1Ah																															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GX-0 GX-1													e	d					c											b	a	
GX-2													e						c											b	a	
CX/EX																			c											b	a	
CT													e						c											b	f	

DESCRIPTION			
a	RW	CFG_MEM_AP_SIZE	Linear memory aperture size (Default = 0) 0 = Linear aperture disabled 1 = 4M aperture 2 = 8M aperture 3 = *Reserved*
b	RW	CFG_MEM_VGA_AP_EN	VGA aperture enabled (Default = 0)
c	RW	CFG_MEM_AP_LOC	Linear memory aperture location in 4M increments. With 8M apertures, bit 4 is ignored
d	RW	CFG_CARD_ID	Card select control (Default = CFG_INIT_CARD_ID@CONFIG_STAT0)
e	RW	CFG_VGA_DIS	VGA disabled (Default = 0) 0 = VGA enabled 1 = VGA disabled
f	RW	CFG_MEM_AP_SIZE	Linear memory aperture size (Default = 0) 0 = *Reserved* 1 = *Reserved* 2 = 2 x 8M apertures 3 = *Reserved*

Description

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

See Also

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

- *Advanced Topics II: Boot-time Initialization*

CONFIG_STAT0																													
I/O:1Ch MM:39h																													
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GX-0 GX-1	r	q	p	o	n	m	l	k	j	i	h						g	f			e			d		c	b			a		
GX-2	r	q	p	o	n	m	l	k	j	i	h						g		u	t	e			d		c	s			a		
CX/EX		q	p	o	n	m	l			i							g				y			x		c	w			v		
CT																											bb		aa	z		

DESCRIPTION			
a	R	CFG_BUS_TYPE	Host bus interface 0 = PCI 1 = VLB 2 -5 = *Reserved* 6 = EISA 7 = ISA
b	R	CFG_MEM_TYPE	Memory type 0 = DRAM (256Kx4) 1 = VRAM (256Kx4, x8, x16) 2 = VRAM (256Kx16 short shift register) 3 = DRAM (256Kx16) 4 = Graphics DRAM (256Kx16) 5 = Enhanced VRAM (256Kx4, x8, x16) 6 = Enhanced VRAM (256Kx16 short shift register) 7 = *Reserved*
c	R	CFG_DUAL_CAS_EN	Dual CAS support enable 0 = Dual CAS disabled 1 = Dual CAS enable
d	R	CFG_LOCAL_BUS_OPTION	Local bus option 0 = *Reserved* 1 = Local option 2 2 = Local option 3 3 = Local option 1
e	R	CFG_INIT_DAC_TYPE	DAC type 0 = *Reserved* 1 = TVP3020 2 = ATI68875 3 = BT476/BT478 4 = BT481 5 = ATI68860/ATI68880 6 = STG1700 7 = SC15021
f	R	CFG_INIT_CARD_ID	Card ID 0 = Card ID 0 1 = Card ID 1 2 = Card ID 2 3 = Card ID 3 4 = Card ID 4 5 = Card ID 5 6 = Card ID 6 7 = Card ID feature disabled

(*Continued on next page*)

CONFIG_STAT0 — I/O:1Ch — MM:39h

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

			DESCRIPTION
g	R	CFG_TRI_BUF_DIS	Tri-stating of output buffers during reset disable 0 = Enables tri-stating of the output buffers 1 = Disables tri-stating of the output buffers
h	R	CFG_EXT_ROM_ADDR	Extended mode ROM base address 0 = C0000h 1 = C1000h ... 3F = FE000h
i	R	CFG_ROM_DIS	ROM disable 0 = Enables ROM 1 = Disables ROM
j	R	CFG_VGA_EN	VGA controller enable
k	R	CFG_LOCAL_BUS_CFG	Local bus configuration select 0 = Local bus configuration 2 1 = Local bus configuration 1
l	R	CFG_CHIP_EN	Chip enable 0 = Disables chip 1 = Enables chip
m	R	CFG_LOCAL_READ_DLY_DIS	Delay read cycle termination (by 1 bus clock) disable 0 = Delays read cycle termination 1 = No read cycle termination delay
n	R	CFG_ROM_ADDR	ROM address 0 = E0000 1 = C0000
o	R	CFG_BUS_OPTION	EISA bus: 0 = Disables POS register; enables chip 1 = Enables POS registers VESA Local Bus: 0 = Enables decode of I/O address 102 1 = Disables decode of I/O address 102
p	R	CFG_LOCAL_DAC_WR_EN	DAC write enable in local bus configuration 0 = Disables local bus DAC writes 1 = Enables local bus DAC writes
q	R	CFG_VLB_RDY_DIS	VESA local bus compliant RDY format disable 0 = Enables VLB RDY 1 = Disables VLB RDY
r	R	CFG_AP_4GBYTE_DIS	4GB aperture addressing disable 0 = Enable 1 = Disable

(*Continued on next page*)

CONFIG_STAT0											I/O:1Ch								MM:39h											
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GX-0 GX-1	r	q	p	o	n	m	l	k	j	i	h						g	f			e			d		c	b			a		
GX-2	r	q	p	o	n	m	l	k	j	i	h						g		u	t	e			d		c	s			a		
CX/EX		q	p	o	n	m	l			i							g				y			x		c	w			v		
CT																											bb		aa	z		

DESCRIPTION			
s	R	CFG_MEM_TYPE	Memory type 0 = DRAM (256Kx4) 1 - 2 = *Reserved* 3 = DRAM (256Kx16) 4 = *Reserved* 5 = Enhanced VRAM (256Kx4, x8, x16) 6 = Enhanced VRAM (256Kx16 short shift register) 7 = *Reserved*
t	R	CFG_BLK_WR_SIZE	Setup the block size for block writes 0 = block size is 4 words deep 1 = block size is 8 words deep
u	R	CFG_INT_QSF_EN	Enables internal QSF generation 0 = use external QSF 1 = use internal QSF
v	R	CFG_BUS_TYPE	Host bus interface 0 = PCI 1 = VLB 2 - 7 = *Reserved*
w	R	CFG_MEM_TYPE	Memory type 0 = DRAM (256Kx4, x8, x16) 1 = EDO DRAM (256Kx4, x8, x16) 2 = *Reserved* 3 = DRAM (256Kx16, asymmetric RAS/CAS) 4 - 7 = *Reserved*
x	R	CFG_LOCAL_BUS_OPTION	Local bus option 0 - 1 = *Reserved* 2 = Local option 3 3 = Local option 1
y	R	CFG_INIT_DAC_TYPE	DAC type 0 - 3 = *Reserved* 4 = BT481 5 - 6 = *Reserved* 7 = STG1702, STG1703, ATT20C498, ATT20C499
z	RW	CFG_MEM_TYPE	Memory type: 0 = Memory type unknown (Disable memory access) 1 = DRAM 2 = EDO DRAM 3-7 = *Reserved*
aa	RW	CFG_DUAL_CAS_EN	Dual CAS support enable: 0 = dual CAS support disabled 1 = dual CAS support enabled

(*Continued on next page*)

CONFIG_STAT0																													
I/O:1Ch MM:39h																													
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GX-0 GX-1	r	q	p	o	n	m	l	k	j	i	h						g	f			e			d		c	b			a		
GX-2	r	q	p	o	n	m	l	k	j	i	h						g		u	t	e			d		c	s			a		
CX/EX		q	p	o	n	m	l			i							g				y			x		c	w			v		
CT																											bb		aa	z		

DESCRIPTION			
bb	RW	CFG_CLOCK_EN	0 = GUI clock controlled by GUI activity 1 = GUI clock always on

Description

CONFIG_STAT0 is a read-only register. It returns the configuration of the current board.

Usage

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

See Also

CONFIG_STAT1 on page 3-14

CONFIG_STAT1 / CRC_SIG	I/O:1Dh	MM:3Ah

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

			DESCRIPTION
a	R	CFG_PCI_DAC_CFG	PCI local bus: DAC data bus configuration 0 = DAC data bus connected directly to the DAC 1 = DAC data bus interfaced through a latch to the DAC
b	R	CFG_1C8_IO_SEL	Base I/O address 0 = 2EC 1 = 1C8
c	R	CRC_SIG	Result of 24 bit CRC accumulation (test mode 15)

Description

CONFIG_STAT1 is a continuation of register CONFIG_STAT0.

On a *mach64*CT, this register is used to accumulate the display CRC check.

Usage

See CONFIG_STAT0 on page 3-12 for a usage description of CONFIG_STAT1.

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

See Also

CONFIG_STAT0 on page 3-12

GEN_TEST_CNTL on page 3-57

CONTEXT_LOAD_CNTL — MM:CBh

All *mach64*

31	30	29	28	27	26	25	24	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														

			DESCRIPTION
a	RW	CONTEXT_LOAD_PTR	Context load pointer
b	RW	CONTEXT_LOAD_CMD	Context load command 0 = No context load 1 = Load context from CONTEXT_LOAD_PTR 2 = Load context from CONTEXT_LOAD_PTR and initiate rectangular fill 3 = Load context from CONTEXT_LOAD_PTR and initiate Bresenham line
c	RW	CONTEXT_LOAD_DIS	Context load disable 0 = Execute context command 1 = Do not execute context command

Description

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

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

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

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

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

Usage

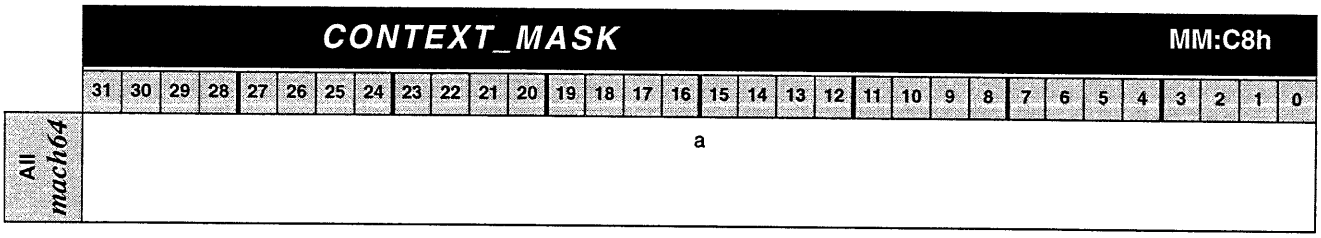
This register is used to load a default context into the draw engine or to execute a context chain.

See Also

CONTEXT_MASK on page 3-16

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

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



DESCRIPTION			
a	RW	CONTEXT_MASK	Context mask

Description

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

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

Usage

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

See Also

CONTEXT_LOAD_CNTL on page 3-15

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

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

CRTC_GEN_CNTL — I/O:7h — MM:7h

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

DESCRIPTION			
a	RW	CRTC_DBL_SCAN_EN	Double scan enable
b	RW	CRTC_INTERLACE_EN	Interlace enable
c	RW	CRTC_HSYNC_DIS	Horizontal sync output disable
d	RW	CRTC_VSYNC_DIS	Vertical sync output disable
e	RW	CRTC_CSYNC_EN	Composite sync on horizontal sync output enable
f	RW	CRTC_PIX_BY_2_EN	CRTC to advance 2 pixels per pixel clock enable
g	RW	CRTC_DISPLAY_DIS	Disables the display, forcing the blanking signal to be active
h	RW	CRTC_PIX_WIDTH	Display pixel width 0 = *Reserved* 1 = 4 bpp 2 = 8 bpp 3 = 15 bpp (5,5,5) 4 = 16 bpp (5,6,5) 5 = 24 bpp 6 = 32 bpp 7 = *Reserved*
i	RW	CRTC_BYTE_PIX_ORDER	Pixel order reversal within memory byte (4 bpp) enable 0 = Pixel order from MSNibble to LSNibble 1 = Pixel order from LSNibble to MSNibble
j	RW	CRTC_FIFO_LWM	16-entry deep display FIFO low water mark
k	RW	CRTC_EXT_DISP_EN	Extended display mode enable (Default = 0) 0 = VGA display 1 = Extended mode display
l	RW	CRTC_EN	CRT controller enable (Default = 0) 0 = Resets CRTC 1 = Enable CRTC
m	RW	CRTC_DISP_REQ_ENb	Enables display requests (Default = 0) 0 = enable display requests 1 = disable display requests
n	RW	VGA_ATI_LINEAR	Enables linear addressing through VGA aperture. 0 = disable linear addressing 1 = enable linear addressing
o	RW	CRTC_VSYNC_FALL_EDGE	Select VSYNC edge to start frame sequence. 0 = rising edge of VSYNC 1 = falling edge of VSYNC
p	RW	VGA_TEXT_132	Extended text mode select. (linear address 132 column text mode) 1 = Active 0 = Inactive

(Continued on next page)

CRTC_GEN_CNTL	I/O:7h	MM:7h

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

DESCRIPTION			
q	RW	VGA_XCRT_CNT_EN	Extended CRTC display address counter enable. Active High
r	RW	VGA_CUR_B_TEST	Test cursor blinking. Active High
s	RW	CRTC_EXTRA_PIPE_ DELAY	Introduce an extra character clock of pipeline delay into the VGA data path
t	RW	CRCT_EXTRA_FIFO_ READ	1 = Enables extra request from DFIFO
u	RW	CRTC_VSTATUS_VSYNC	0 = VSTATUS 1 = VSYNC

Description

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

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

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

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

Usage

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

See Also

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

- *Programming Model: Mode Switching: Manual Mode Switching*
- *Programming Model: Mode Switching: Designing a Custom CRT Mode*
- *Appendix C, CRTC Parameters*

CRTC_H_SYNC_STRT_WID																		I/O:1h								MM:1h			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
All *mach64*										d	c										b			a							

			DESCRIPTION
a	RW	CRTC_H_SYNC_STRT	Horizontal sync start
b	RW	CRTC_H_SYNC_DLY	Horizontal sync start delay in pixels
c	RW	CRTC_H_SYNC_WID	Horizontal sync width
d	RW	CRTC_H_SYNC_POL	Horizontal sync polarity

Description

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

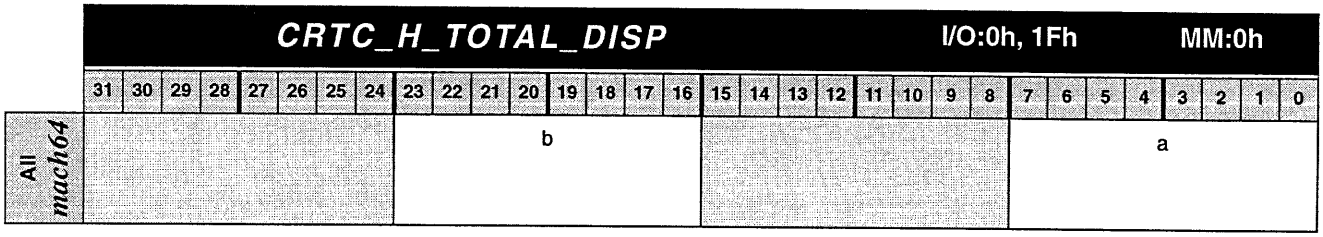
Usage

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

See Also

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

- *Programming Model: Mode Switching: Manual Mode Switching*
- *Programming Model: Mode Switching: Designing a Custom CRT Mode*
- *Appendix C, CRTC Parameters*



DESCRIPTION			
a	RW	CRTC_H_TOTAL	Horizontal total
b	RW	CRTC_H_DISP	Horizontal display end

Description

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

Usage

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

See Also

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

- *Programming Model: Mode Switching: Manual Mode Switching*
- *Programming Model: Mode Switching: Designing a Custom CRT Mode*
- *Appendix C, CRTC Parameters*

All *mach64*

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

		DESCRIPTION	
a	R	CRTC_VBLANK	Vertical blank
b	RW	CRTC_VBLANK_INT_EN	Vertical blank interrupt enable (Default = 0)
c	R	CRTC_VBLANK_INT	Vertical blank interrupt
	W	CRTC_VBLANK_ACK	Vertical blank acknowledge
d	RW	CRTC_VLINE_INT_EN	Vertical line interrupt enable (Default = 0)
e	R	CRTC_VLINE_INT	Vertical line interrupt
	W	CRTC_VLINE_ACK	Vertical line interrupt acknowledge
f	R	CRTC_VLINE_SYNC	Vertical line sync 0 = Even scan line 1 = Odd scan line
g	R	CRTC_FRAME	Interlaced odd/even frame 0 = even frame 1 = odd frame

Description

CRTC_INT_CNTL is used for enabling interrupts generated by the accelerator CRTC, for acknowledging those interrupts, and for reading the status of the CRTC.

Usage

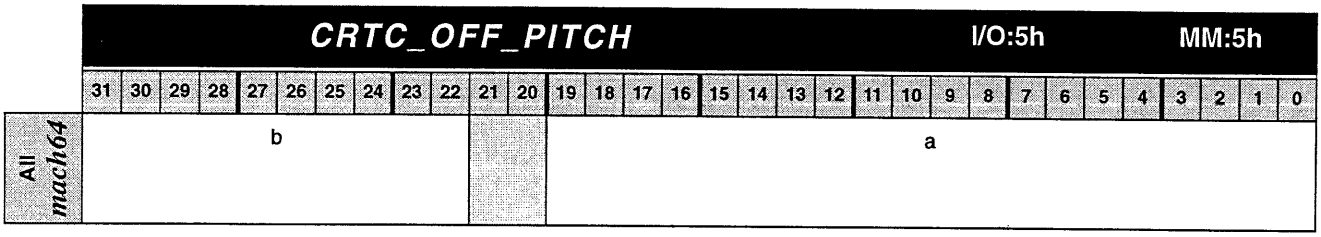
Applications may use this register to achieve smooth animation or to reduce flickering and tearing.

See Also

CRTC_VLINE_CRNT_VLINE on page 3-25

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

- *Advanced Topics I: Interrupts*
- *Advanced Topics I: CRT Synchronization*



DESCRIPTION			
a	RW	CRTC_OFFSET	Display address offset, in terms of 64-bit words
b	RW	CRTC_PITCH	Display pitch in pixels-times-8

Description

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

Usage

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

See Also

MEM_CNTL on page 3-67

SRC_OFF_PITCH on page 3-92

DST_OFF_PITCH on page 3-50

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

- *Programming Model: VGA Interaction*
- *Advanced Topics I: Scrolling and Panning*
- *Advanced Topics I: CRT Synchronization: Double Buffering (Memory)*

CRTC_V_SYNC_STRT_WID																				I/O:3h								MM:3h			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
All *mach64*											c	b										a										

		DESCRIPTION	
a	RW	CRTC_V_SYNC_STRT	Vertical sync start
b	RW	CRTC_V_SYNC_WID	Vertical sync width
c	RW	CRTC_V_SYNC_POL	Vertical sync polarity

Description

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

Usage

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

See Also

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

- *Programming Model: Mode Switching: Manual Mode Switching*
- *Programming Model: Mode Switching: Designing a Custom CRT Mode*
- *Appendix C, CRTC Parameters*

CRTC_V_TOTAL_DISP I/O:2h MM:2h

All *mach64*

31	30	29	28	27	26	25	24	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											

DESCRIPTION			
a	RW	CRTC_V_TOTAL	Vertical total
b	RW	CRTC_V_DISP	Vertical display end

Description

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

Usage

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

See Also

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

- *Programming Model: Mode Switching: Manual Mode Switching*
- *Programming Model: Mode Switching: Designing a Custom CRT Mode*
- *Appendix C, CRTC Parameters*

CRTC_VLINE_CRNT_VLINE																		I/O:4h												MM:4h			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
All *mach64*					b																a												

DESCRIPTION			
a	RW	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 interrupts are enabled. The CRTC_CRNT_VLINE field is read-only. It returns the current value of the accelerator CRTC vertical line counter.

Usage

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

See Also

CRTC_INT_CNTL on page 3-21

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

- *Advanced Topics I: Interrupts*
- *Advanced Topics I: CRT Synchronization*

CUR_CLR0																				I/O:Bh								MM:18h			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
All *mach64*	d								c								b								a						

			DESCRIPTION
a	RW	CUR_CLR0_8	Cursor color 0 for 4 bpp and 8 bpp display modes
b	RW	CUR_CLR0_B	Blue cursor color 0
c	RW	CUR_CLR0_G	Green cursor color 0
d	RW	CUR_CLR0_R	Red cursor color 0

Description

CUR_CLR0 contains color 0 for the hardware cursor. The color is specified in the lower 8 bits for pseudocolor modes or the upper 24 bits in direct color modes.

Usage

This register is used when defining the hardware cursor attributes.

See Also

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

- *Programming Model: Hardware Cursor*

CUR_CLR1																																I/O:Ch	MM:19h
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
All *mach64*	d								c								b								a								

			DESCRIPTION
a	RW	CUR_CLR1_8	Cursor color 1 for 4 bpp and 8 bpp display modes
b	RW	CUR_CLR1_B	Blue cursor color 1
c	RW	CUR_CLR1_G	Green cursor color 1
d	RW	CUR_CLR1_R	Red cursor color 1

Description

CUR_CLR1 contains color 1 for the hardware cursor. The color is specified in the lower 8 bits for pseudocolor modes or the upper 24 bits in direct color modes.

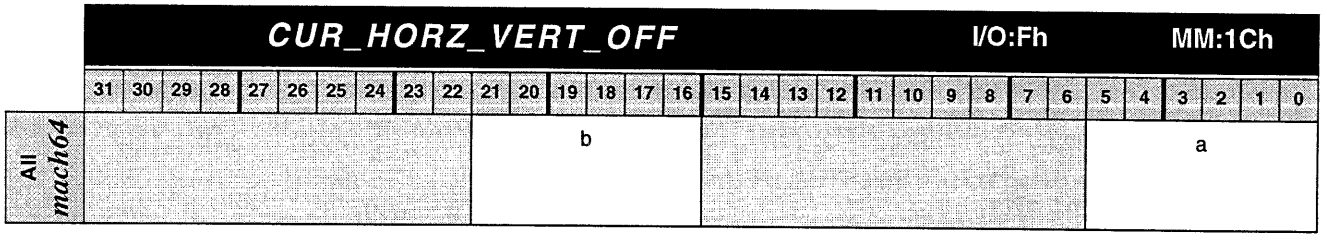
Usage

This register is used when defining the hardware cursor attributes.

See Also

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

- *Programming Model: Hardware Cursor*



			DESCRIPTION
a	RW	CUR_HORZ_OFF	Cursor horizontal offset
b	RW	CUR_VERT_OFFSET	Cursor vertical offset

Description

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

Usage

This register is used when defining the hardware cursor attributes.

See Also

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

- *Programming Model: Hardware Cursor*

CUR_HORZ_VERT_POSN																		I/O:Eh										MM:1Bh			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
All *mach64*						b																a										

		DESCRIPTION	
a	RW	CUR_HORZ_POSN	Cursor horizontal position
b	RW	CUR_VERT_POSN	Cursor vertical position

Description

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

Usage

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

See Also

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

- *Programming Model: Hardware Cursor*

CUR_OFFSET																				I/O:Dh								MM:1Ah			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
All *mach64*												a																			

DESCRIPTION			
a	RW	CUR_OFFSET	Cursor address offset, in terms of 64-bit words

Description

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

Usage

This register is used to define the hardware cursor.

See Also

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

- *Programming Model: Hardware Cursor*

DAC_CNTL — I/O:18h — MM:31h

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

	DESCRIPTION		
a	RW	DAC_EXT_SEL	DAC extended register select inputs
b	RW	DAC_8BIT_EN	8-bit DAC enable
c	RW	DAC_PIX_DLY	Blank delay, in terms of pixel clock periods 0 = No delay 1 = 1 pixel clock period delay 2 = 2 pixel clock period delay 3 = *Reserved*
d	RW	DAC_BLANK_ADJ	Blank delay, in terms of pixel clock periods 0 = No delay 1 = 1 pixel clock period delay 2 = 2 pixel clock period delay 3 = *Reserved*
e	RW	DAC_VGA_ADR_EN	DAC Addressing at the VGA I/O DAC address enable, when CRTC_EXT_DISP_EN@CRTC_GEN_CNTL = 1
f	RW	DAC_TYPE	DAC type — Initially, this register will reflect the value of the CFG_INIT_DAC_TYPE@CONFIG_STAT0 configuration straps
g	RW	DAC_MON_ID_STATE	Monitor ID pin state: the monitor ID is determined by manipulating the state and direction of three monitor ID pins and reading back the state
h	RW	DAC_MON_ID_DIR	Monitor ID direction: Each of the three monitor ID pins may be set to input or output based on the direction field as follows: (default=0) 0 = All pins are input 1 = Monitor ID pin 0 is output, other pins are input 2 = Monitor ID pin 1 is output, other pins are input 3 = *Reserved* 4 = Monitor ID pin 2 is output, other pins are input 5 - 6 =*Reserved*
i	R	DAC_MON_CMP_STATE	Output of DAC RGB level comparator
j	RW	DAC_BLANKING	Enable 7.5 IRE blanking pedestal 0 = 0 IRE blanking pedestal 1 = 7.5 IRE blanking pedestal
k	R	DAC_CMP_OUTPUT	DAC comparator outputs x x x 0 = Red comparator > 0.42 V x x x 1 = Red comparator < 0.28 V x x 0 x = Green comparator > 0.42 V x x 1 x = Green comparator < 0.28 V x 0 x x = Blue comparator > 0.42 V x 1 x x = Blue comparator < 0.28 V 0 x x x = At least 1 comparator > 0.42 V 1 x x x = All 3 comparators < 0.28 V

(Continued on next page)

DAC_CNTL																		I/O:18h								MM:31h							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GX-0 GX-1														f					e	d		c		b							a	
GX-2		i		h			g							f					e			c		b							a	
CX/EX				h			g							f					e			c		b							a	
CT				h			g							n			m	l	e					b		k			j			

DESCRIPTION			
l	RW	DAC_FEA_CON_EN	Enables feature connector signal outputs. Turns on output of 8 bit pixel data, clock and blank signals to feature connectors. Feature connector should be disabled in 24 bpp, 32 bpp modes and high resolution modes when pixel clock rate is too high.
m	RW	DAC_PDWN	Power down internal DAC (DAC macro only). Feature connector outputs can still run normally.
n	R	DAC_TYPE	DAC Type 0 = internal DAC 1= *Reserved* 2= *Reserved* 3= *Reserved* 4= *Reserved* 5= *Reserved* 6= *Reserved* 7= *Reserved*

Description

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

The DAC_TYPE can be overwritten to override the initial DAC type. Please consult the manufacturer's DAC specification.

Usage

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

See Also

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

- *Advanced Topics II: DAC Programming*

DAC_REGS																				I/O:17h				MM:30h			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
All *mach64*	d							c								b								a							

			DESCRIPTION
a	RW	DAC_W_INDEX	DAC write index register: automatically increments after three consecutive writes to DAC_DATA.
b	RW	DAC_DATA	DAC data register. Will read/write red, green, and blue data from/to the DAC palette. Writes are indexed with DAC_W_INDEX and reads are indexed with DAC_R_INDEX.
c	RW	DAC_MASK	DAC mask register. In pseudocolor modes, this register will mask the 4 or 8 bit color index data before the palette table lookup.
d	RW	DAC_R_INDEX	DAC read index register. Automatically increments after three consecutive writes to DAC_DATA.

Description

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

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

Usage

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

See Also

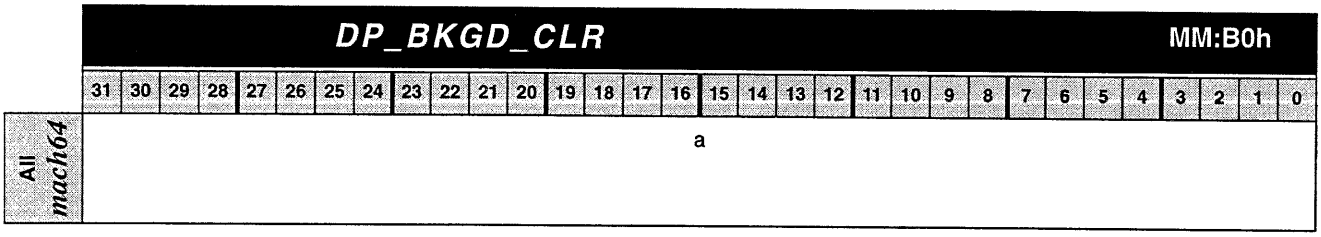
DAC_CNTL on page 3-31

***mach64* VGA Register Guide**:

- *VGA-Compatible Registers*

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

- *Advanced Topics I: CRT Synchronization: Double Buffering (Palette)*
- *Advanced Topics II: DAC Programming*



DESCRIPTION			
a	RW	DP_BKGD_CLR	Background color

Description

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

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

Usage

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

See Also

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

- *Programming Model: Logical Pixel Data Path*

DP_CHAIN_MSK																												MM:B3h
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
All *mach64*																a															

DESCRIPTION			
a	RW	DP_CHAIN_MSK	Chain mask

Description

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

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

Pixel Depth	DP_CHAIN_MASK
1	N/A
4 bpp pseudocolor	0x8888
7 bpp, aRGB 1232	0xD2D2
8 bpp pseudocolor	0x8080
8 bpp, RGB 332	0x9292
15 bpp, aRGB 1555	0x4210
16 bpp, RGB 565	0x8410
24 bpp, RGB 888	0x8080
32 bpp, RGBa 8888	0x8080

Usage

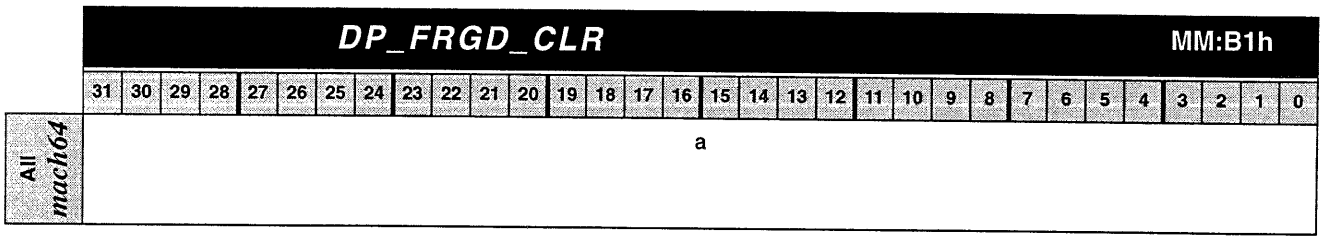
Set this register only when the foreground mix or background mix is set to function 0x17.

See Also

DP_MIX on page 3-37

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

- *Programming Model: Source and Destination Mixing Logic*



DESCRIPTION			
a	RW	DP_FRGD_CLR	Foreground color

Description

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

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

Usage

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

See Also

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

- *Programming Model: Logical Pixel Data Path*

DP_MIX MM:B5h

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
All *mach64*												b																a				

DESCRIPTION			
a		DP_BKGD_MIX	Background mix
b		DP_FRGD_MIX	Foreground mix

Description

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

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

Usage

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

See Also

DP_MONO_SRC@DP_SRC on page 3-40

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

- *Programming Model: Logical Pixel Data Path*
- *Programming Model: Source and Destination Mixing Logic*

DP_PIX_WIDTH — MM:B4h

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
All *mach64*								d						c								b								a		

			DESCRIPTION
a	RW	DP_DST_PIX_WIDTH	Destination data path pixel width 0 = Monochrome 1 = 4 bpp 2 = 8 bpp 3 = 15 bpp (5,5,5) 4 = 16 bpp (5,6,5) 5 = *Reserved* 6 = 32 bpp 7 = *Reserved*
b	RW	DP_SRC_PIX_WIDTH	Source data path pixel width — bit description same as those for DP_DST_PIX_WIDTH[2:0], shown above.
c	RW	DP_HOST_PIX_WIDTH	Host data path pixel width — bit description same as those for DP_DST_PIX_WIDTH[2:0], shown above.
d	RW	DP_BYTE_PIX_ORDER	Pixel order within each byte in monochrome and 4 bpp modes 0 = Pixel order from MSBit (nibble) to LSBit (nibble) 1 = Pixel order from LSBit (nibble) to MSBit (nibble)

Description

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

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

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

Usage

This register is used for setting draw engine pixel width and pixel ordering within a byte.

See Also

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

- *Simple Draw Operations: Monochrome Expansion Bitblit*

DP_SRC																														MM:B6h
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
All *mach64*															c							b								a		

			DESCRIPTION
a	RW	DP_BKGD_SRC	Background source 0 = Background color 1 = Foreground color 2 = Host data 3 = Blit source 4 = Pattern registers 5 = *Reserved* 6 = *Reserved* 7 = *Reserved*
b	RW	DP_FRGD_SRC	Foreground — bit description same as those for DP_BKGD_SRC[2:0], shown above.
c	RW	DP_MONO_SRC	Monochrome source 0 = Always_'1' 1 = Pattern registers 2 = Host data 3 = Blit source

Description

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

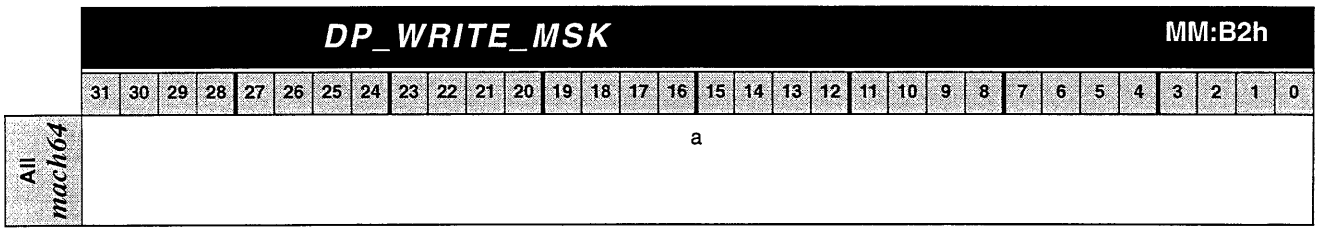
Usage

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

See Also

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

- *Programming Model: Logical Pixel Data Path*



DESCRIPTION			
a	RW	DP_WRITE_MSK	Write mask

Description

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

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

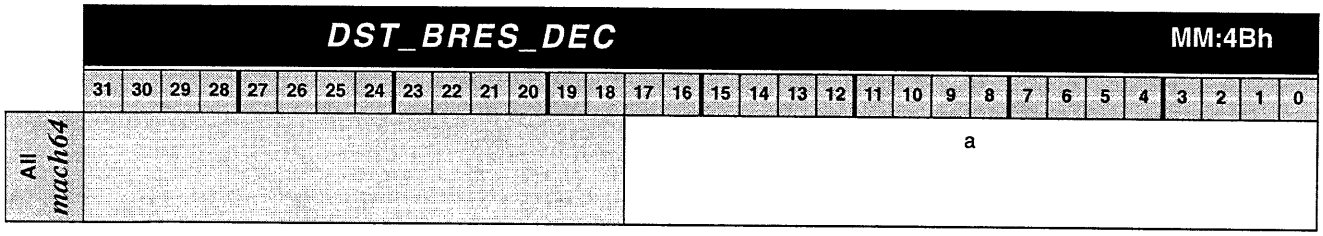
Usage

All draw operations require this register to be set.

See Also

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

- *Programming Model: Scissoring and Masking*



DESCRIPTION			
a	RW	DST_BRES_DEC	Bresenham decrement term

Description

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

The value written to this register should be calculated:

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

Usage

This register is used only for line draw operations.

See Also

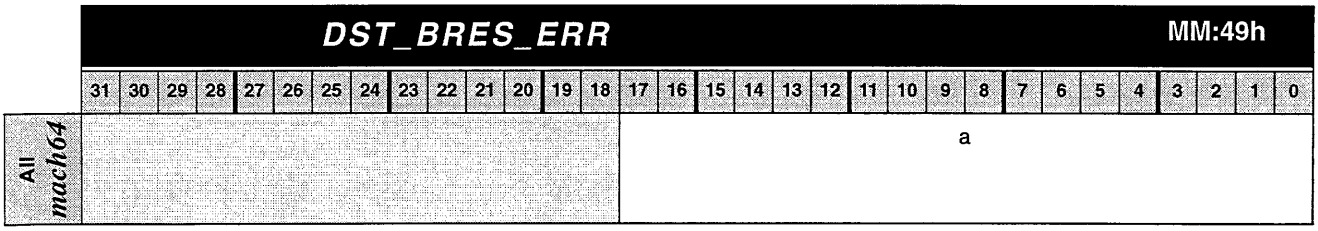
DST_BRES_INC on page 3-44

DST_BRES_ERR on page 3-43

DST_BRES_LNTH on page 3-45

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

- *Programming Model: Trajectories: Destination Trajectory 2, Line*
- *Simple Draw Operations: Line Draw*



DESCRIPTION			
a	RW	DST_BRES_DEC	Bresenham decrement term

Description

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

The initial value of the register field DST_BRES_ERR should be:

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

Usage

This register is used only for line draw operations.

See Also

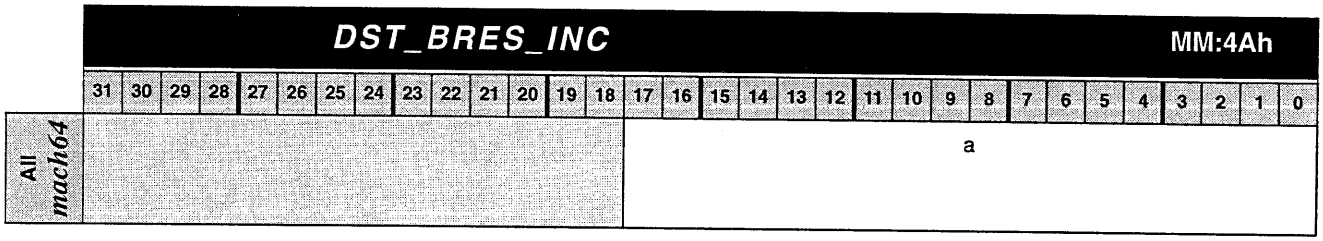
DST_BRES_DEC on page 3-42

DST_BRES_INC on page 3-44

DST_BRES_LNTH on page 3-45

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

- *Programming Model: Trajectories: Destination Trajectory 2, Line*
- *Simple Draw Operations: Line Draw*



DESCRIPTION			
a	RW	DST_BRES_INC	Bresenham increment term

Description

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

The value written to the field DST_BRES_INC should be:

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

Usage

This register is used only for line draw operations.

See Also

DST_BRES_DEC on page 3-42

DST_BRES_ERR on page 3-43

DST_BRES_LNTH on page 3-45

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

- *Programming Model: Trajectories: Destination Trajectory 2, Line*
- *Simple Draw Operations: Line Draw*

DST_BRES_LNTH MM:48h

All *mach64*

31	30	29	28	27	26	25	24	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															

DESCRIPTION			
a	RW	DST_BRES_LNTH	Bresenham line length
b	RW	DST_BRES_LNTH_LINE_DIS	Initiation of Bresenham line draw operations disabled 0 = Bresenham line draw operation initiated 1 = No Bresenham line draw operation initiated

Description

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

Writing to this register also overwrites the contents of DST_WIDTH

DST_BRES_LNTH = max(ldxl,ldyl) + 1

Usage

This register is used to draw lines.

See Also

DST_BRES_DEC on page 3-42

DST_BRES_ERR on page 3-43

DST_BRES_INC on page 3-44

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

- *Programming Model: Trajectories: Destination Trajectory 2, Line*
- *Simple Draw Operations: Line Draw*

DST_CNTL																													MM:4Ch
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GX-0 GX-1 GX-2 CX/EX																					j	i			h	g	f	e	d	c	b	a
CT																				k	j	i			h	g	f	e	d	c	b	a

			DESCRIPTION
a	RW	DST_X_DIR	Destination X direction 0 = Right to left 1 = Left to right
b	RW	DST_Y_DIR	Destination Y direction 0 = Bottom to top 1 = Top to bottom
c	RW	DST_Y_MAJOR	Destination Y major axis flag for Bresenham lines 0 = X major line 1 = Y major line
d	RW	DST_X_TILE	Rectangular tiling in the X direction enable
e	RW	DST_Y_TILE	Rectangular tiling in the Y direction enable
f	RW	DST_LAST_PEL	Destination last PEL enable
g	RW	DST_POLYGON_EN	Destination polygon outline and polygon fill enable
h	RW	DST_24_ROT_EN	24 bpp rotation enable — DST_PIX_WIDTH must be set to 8 bpp
i	RW	DST_24_ROT	The initial foreground color, background color, write mask, and monochrome pattern rotation when drawing packed 24 bpp.
j	RW	DST_BRES_SIGN	Sign of DST_BRES_ERR 0 = DST_BRES_ERR=0 is positive 1 = DST_BRES_ERR=0 is negative
k	RW	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

Description

Miscellaneous control bits for the destination area:

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

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

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

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

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

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

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

Usage

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

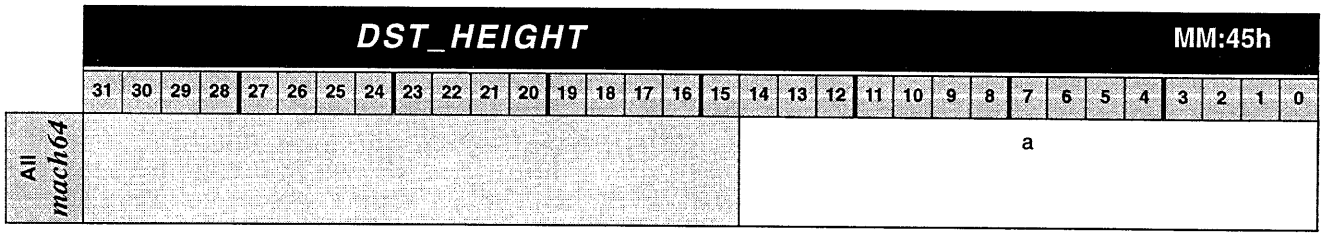
See Also

GUI_TRAJ_CNTL on page 3-62

SRC_CNTL on page 3-86

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

- *Programming Model: Trajectories: Destination Trajectory 1, Rectangular*
- *Programming Model: Trajectories: Destination Trajectory 2, Line*
- *Programming Model: Trajectories: Trajectory Modifier 2, DST_POLYGON_EN*
- *Programming Model: Side Effects*
- *Advanced Topics I: Polygons*
- *Advanced Topics I: Drawing in Packed 24 Bit Per Pixel Mode*



DESCRIPTION			
a	RW	DST_HEIGHT	Destination height

Description

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

Usage

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

See Also

DST_WIDTH on page 3-51
DST_HEIGHT_WIDTH on page 3-49

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

- *Programming Model: Trajectories: Destination Trajectory 1, Rectangular*
- *Simple Draw Operations: Rectangle Fill*
- *Simple Draw Operations: Bitblit*
- *Advanced Topics I: Polygons*
- *Advanced Topics I: Transparent Blits*

DST_HEIGHT_WIDTH MM:46h

All *mach64*

31	30	29	28	27	26	25	24	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														

			DESCRIPTION
a	W	DST_HEIGHT	Destination height
b	W	DST_WIDTH	Destination width

Description

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

Usage

These registers are used only for drawing rectangular destinations.

See Also

DST_HEIGHT on page 3-48

DST_WIDTH on page 3-51

DST_OFF_PITCH MM:40h

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
All *mach64*	b												a																			

DESCRIPTION			
a	RW	DST_OFFSET	Destination offset address, in terms of 64-bit words
b	RW	DST_PITCH	Destination pitch in pixel-times-8. Note: In monochrome modes the destination pitch must be a multiple of 64 pixels; in 4 bpp modes the destination pitch must be a multiple of 16 pixels.

Description

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

Usage

This register should be set for all draw operations.

See Also

SRC_OFF_PITCH on page 3-92

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

- *Programming Model: Trajectories: Destination Trajectory 1, Rectangular*
- *Programming Model: Trajectories: Destination Trajectory 2, Line*
- *Programming Model: VGA Interaction*
- *Advanced Topics I: CRT Synchronization: Double Buffering (Memory)*

DST_WIDTH — MM:44h

All *mach64*

31	30	29	28	27	26	25	24	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												

DESCRIPTION			
a	RW	DST_WIDTH	Destination width
b	RW	DST_WID_FILL_DIS	Initiation of rectangular fill operation disabled 0 = Initiates rectangular fill operation 1 = No rectangular fill operation initiation

Description

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

Writing to this register also overwrites the contents of DST_BRES_LNTH.

Usage

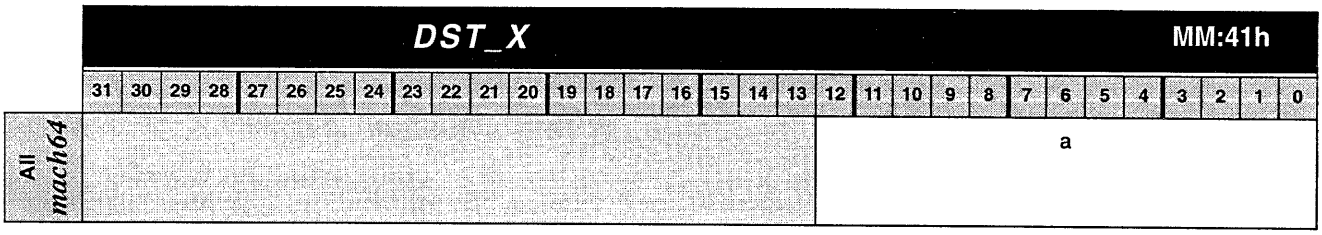
This register is used only when drawing a rectangular destination area.

See Also

DST_HEIGHT on page 3-48
DST_HEIGHT_WIDTH on page 3-49
DST_X_WIDTH on page 3-53

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

- *Programming Model: Trajectories: Destination Trajectory 1, Rectangular*
- *Simple Draw Operations: Rectangle Fill*
- *Simple Draw Operations: Bitblit*
- *Advanced Topics I: Polygons*
- *Advanced Topics I: Transparent Blits*



DST_X																												MM:41h
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
All *mach64*																			a												

DESCRIPTION			
a	RW	DST_X	Destination X coordinate

Description

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

Usage

This register is used for all draw operations.

See Also

DST_X_WIDTH on page 3-53

DST_Y on page 3-54

DST_Y_X on page 3-55

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

- *Programming Model: Trajectories: Destination Trajectory 1, Rectangular*
- *Programming Model: Trajectories: Destination Trajectory 2, Line*

DST_X_WIDTH — MM:47h

All *mach64*

31	30	29	28	27	26	25	24	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												

			DESCRIPTION
a	W	DST_X	Destination X coordinate
b	W	DST_WIDTH	Destination width

Description

DST_X_WIDTH is a composite of registers DST_X and DST_WIDTH

Usage

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

See Also

DST_X on page 3-52

DST_WIDTH on page 3-51

DST_Y MM:42h

All *mach64*

31	30	29	28	27	26	25	24	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														

DESCRIPTION			
a	RW	DST_Y	Destination Y coordinate

Description

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

Usage

This register is used for all draw operations.

See Also

DST_X on page 3-52

DST_Y_X on page 3-55

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

- *Programming Model: Trajectories: Destination Trajectory 1, Rectangular*
- *Programming Model: Trajectories: Destination Trajectory 2, Line*

DST_Y_X — MM:43h

All *mach64*

31	30	29	28	27	26	25	24	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														

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

Description

DST_Y_X is a composite of registers DST_X and DST_Y

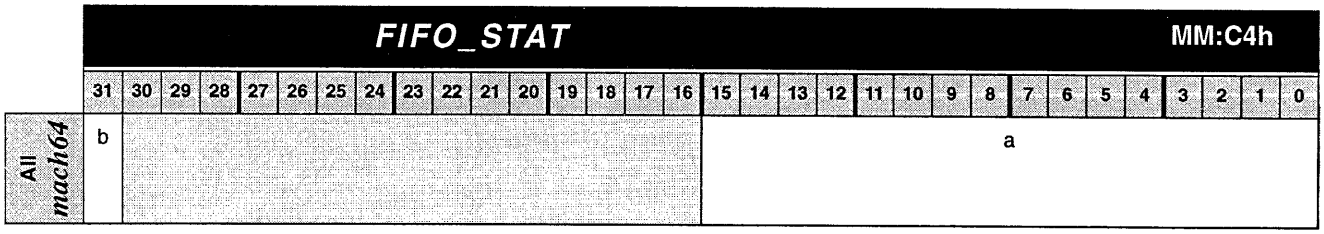
Usage

These registers are used for all draw operations.

See Also

DST_X on page 3-52

DST_Y on page 3-54



DESCRIPTION			
a	R	FIFO_STAT	Indicator of the number of filled command FIFO entries
b	R	FIFO_ERR	FIFO overrun error

Description

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

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

Usage

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

See Also

BUS_CNTL on page 3-2

GEN_TEST_CNTL on page 3-57

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

- *Programming Model: The Command FIFO*
- *Simple Draw Operations*

GEN_TEST_CNTL — I/O:19h — MM:34h

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

	DESCRIPTION		
a	W	GEN_EE_DATA_OUT	EEPROM data out
b	W	GEN_EE_CLOCK	EEPROM clock
c	W	GEN_EE_CHIP_SEL	EEPROM chip select
d	R	GEN_EE_DATA_IN	EEPROM data in
e	RW	GEN_EE_EN	EEPROM interface enable (Default = 0)
f	RW	GEN_OVR_OUTPUT_EN	Overscan signal to be output for external DAC support enable (Default = 0)
g	RW	GEN_OVR_POLARITY	Polarity of external overscan signal 0 = Active 1 1 = Active 0
h	RW	GEN_CUR_EN	Hardware cursor enable (Default = 0)
i	RW	GEN_GUI_EN	Draw engine enable 0 = Resets draw engine 1 = Enables draw engine
j	RW	GEN_BLOCK_WR_EN	Block write memory cycle enable
k	RW	GEN_TEST_FIFO_EN	Draw engine testing of the command FIFO enable
l	RW	GEN_TEST_GUI_REGS_EN	Draw engine register loading without triggering drawing operations or context switching enable
m	RW	GEN_TEST_VECT_EN	Bi-directional buses to have 1 clock turn-around period when transitioning from input to output enable (Default = 0)
n	R	GEN_TEST_CRC_DONE	When this bit goes high the CRC calculation is complete. The CRC can be read back from TEST_REG7. This bit is reset when a (1) is written to GEN_TEST_CRC_STR
	W	GEN_TEST_CRC_STROBE	Pixel data CRC initialization strobe. Writing a (1) to this bit will reset the CRC accumulator. The CRC accumulator will then CRC the next display frame. The CRC value includes overscan and hardware cursor pixel data. CRC is active in VGA display modes as well
o	RW	GEN_TEST_MODE	Test mode 0 = Test mode disabled 1 = Memory read/write test 2 = Draw engine dest/source length counter test 3 = Draw engine source read length counter test 4 = CRTC test 5 = Pixel data CRC test 6 = *Reserved* 7 = *Reserved*

(Continued on next page)

GEN_TEST_CNTL — I/O:19h — MM:34h

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

DESCRIPTION			
p	RW	GEN_TEST_MEM_WR	Test mode 1 memory write cycle 0 = Read cycle 1 = Write cycle
q	W	GEN_TEST_MEM_STROBE	Writing this bit with a value of (1) will initiate a memory cycle if GEN_TEST_MODE (bits 20:22) has a value of 1. Note: GEN_TEST_MEM_WR (bit 24) will determine if the cycle is to be read or write.
r	RW	GEN_TEST_DST_SS_EN	1 = Destination trajectory controller single stepping enable. Note: The engine processes multiple pixels in single steps for rectangular draw operations. Note: Bresenham line draw operations operate a single pixel at a time. (Default = 0)
s	W	GEN_TEST_DST_SS_STROBE	Setting this bit to (1) when GEN_TEST_DST_SS_EN (bit 26) is active advances the draw engine destination by one step
t	RW	GEN_TEST_SRC_SS_EN	1 = Source trajectory controller single stepping enable. Note: The engine processes multiple pixels in single steps for rectangular draw operations, and Bresenham line draw operations operate on a single pixel at a time. (Default = 0)
u	W	GEN_TEST_SRC_SS_STROBE	Setting this bit to (1) when GEN_TEST_SRC_SS_EN (bit 28) is active advances the draw engine destination by one step
v	RW	GEN_TEST_CC_EN	CRTC single stepping enable
w	W	GEN_TEST_CC_STROBE	Setting this bit to (1) when GEN_TEST_CC_EN (bit 30) is active advances the CRTC by one character clock
x	RW	GEN_TEST_MODE	Test mode 0 = Test mode disabled 1 = *Reserved* 2 = Draw engine dest/source length counter test 3 = Draw engine source read length counter test 4 = CRTC test 5 = *Reserved* 6 = *Reserved* 7 = *Reserved*
y	RW	GEN_EE_WRITE	EERPROM data write enable (default = 0 i.e. read enable)

(*Continued on next page*)

GEN_TEST_CNTL																													
I/O:19h MM:34h																													
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GX-0 GX-1	w	v	u	t	s	r	q	p		o			n	m	l	k							j	i	h	g	f	e	d	c	b	a
GX-2	w	v	u	t	s	r				x				m	l	k							j	i	h	g	f	e	d	c	b	a
CX/EX	w	v	u	t	s	r				x				m	l	k								i	h	g	f	e	d	c	b	a
CT			cc								bb	aa	z												h		y	e	d	c	b	a

DESCRIPTION			
z	RW	GEN_TEST_MODE	Enable test modes 0 = Test mode disabled 1 = CRTC (VGACRTC) test enabled 2 = Graphics controller (VGAGC) test enabled 3 = Attribute controller (VGAATTR) test enabled 4 = Display address generator (DADDRGEN) test enabled 5 = GUI engine address generator test 6 = Command FIFO test (Lock the FIFO) 7 = *Reserved* 8 = Delay path and ring oscillator test 9 = *Reserved* 10 = Register block test 11 = PLL test 12 = Palette test 13 = DAC test 14 = DAC functional test 15 = CRC full speed test (24 bit)
aa	RW	GEN_TEST_CNT_EN	Enables the Scan Counter (default to 0)
bb	RW	GEN_CRC_EN	Enables the CRC signature block (default to 0)
cc	RW	GEN_TEST_CNT_VALUE	Scan Counter Value (default to 0)

Description

GEN_TEST_CNTL is a multi-purpose register used for accessing the EEPROM, configuring overscan for external DACs, enabling the hardware cursor, resetting the draw engine after a FIFO lock, enabling the VRAM block write feature, and for general diagnostics.

Usage

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

Application level programs should touch only GEN_CUR_EN and GEN_GUI_EN.

See Also

TEST_REG0 through TEST_REG7 on pages 3-101 through 3-109.

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

- *Programming Model: The Command FIFO*
- *Programming Model: Hardware Cursor*
- *Advanced Topics II: Boot-time Initialization*
- *Advanced Topics II: Accessing the EEPROM*
- *Advanced Topics II: DAC Programming*
- *Advanced Topics II: Diagnostic Features*

All *mach64*

GUI_STAT																												MM:CEh
31	30	29	28	27	26	25	24	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

		DESCRIPTION	
a	R	GUI_ACTIVE	1 = Draw engine is busy
b	R	DSTX_LT_SCISSOR_LEFT	DSTX is left of the left scissor
c	R	DSTX_GT_SCISSOR_RIGHT	DSTX is right of the right scissor
d	R	DSTY_LT_SCISSOR_TOP	DSTY is above the top scissor
e	R	DSTY_GT_SCISSOR_BOTTOM	DSTY is below the bottom scissor

Description

GUI_STAT reports the status of the draw engine.

Usage

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

See Also

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

- *Programming Model: The Command FIFO*

GUI_TRAJ_CNTL																													MM:CCh
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GX-0 GX-1 CX/EX				s		r	q	p				o	n	m	l	k					j	i			h	g	f	e	d	c	b	a
GX-2			t	s		r	q	p				o	n	m	l	k					j	i			h	g	f	e	d	c	b	a
CT			t	s		r	q	p				o	n	m	l	k				u	j	i			h	g	f	e	d	c	b	a

			DESCRIPTION
a	RW	DST_X_DIR	Destination X direction 0 = Right-to-left 1 = Left-to-right
b	RW	DST_Y_DIR	Destination Y direction 0 = Bottom-to-top 1 = Top-to-bottom
c	RW	DST_Y_MAJOR	Destination Y major axis flag for Bresenham lines 0 = X major line 1 = Y major line
d	RW	DST_X_TILE	Rectangular tiling in the X direction enable
e	RW	DST_Y_TILE	Rectangular tiling in the Y direction enable
f	RW	DST_LAST_PEL	Destination last pel enable
g	RW	DST_POLYGON_EN	Destination polygon outline and polygon fill enable
h	RW	DST_24_ROT_EN	24 bpp rotation enable — DST_PIX_WIDTH must be set to 8 bpp
i	RW	DST_24_ROT	Initial foreground color, background color, write mask, and monochrome pattern rotation when drawing packed 24 bpp.
j	RW	DST_BRES_SIGN	Sign of DST_BRES_ERR 0 = DST_BRES_ERR=0 is positive 1 = DST_BRES_ERR=0 is negative
k	RW	SRC_PATT_EN	Pattern source enable — SRC_Y_END will be used only if this bit is enabled.
l	RW	SRC_PATT_ROT_EN	Pattern source rotation enable — SRC_X_START, SRC_Y_START will be used only if this bit is enabled.
m	RW	SRC_LINEAR_EN	Source linearly advanced in memory enable. The source starts at SRC_OFFSET and advances in the left-to-right direction. DST_X_DIR should also be set to left-to-right for proper operation. Note: All other source registers and control bits with the exception of SRC_BYTE_ALIGN are ignored
n	RW	SRC_BYTE_ALIGN	Skip to the next data byte boundary when the destination advances in the Y direction — SRC_LINEAR_EN must be set.
o	RW	SRC_LINE_X_DIR	Source X direction when drawing operation is a Bresenham line.
p	RW	PAT_MONO_EN	Monochrome 8x8 pattern enable

(Continued on next page)

GUI_TRAJ_CNTL																															MM:CCh
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GX-0 GX-1 CX/EX				s		r	q	p				o	n	m	l	k					j		i		h	g	f	e	d	c	b	a
GX-2			t	s		r	q	p				o	n	m	l	k					j		i		h	g	f	e	d	c	b	a
CT			t	s		r	q	p				o	n	m	l	k				u	j		i		h	g	f	e	d	c	b	a

		DESCRIPTION	
q	RW	PAT_CLR_4x2_EN	Color 4x2 pattern enable
r	RW	PAT_CLR_8x1_EN	Color 8x1 pattern enable
s	RW	HOST_BYTE_ALIGN	Byte alignment of host data enable
t	RW	HOST_BIG_ENDIAN_EN	Enables big endian data translation for 15 bpp, and 32 bpp pixel widths. In 15 bpp and 16 bpp modes the bytes within each word are swapped. In 32 bpp mode the order of the four bytes within each dword is reversed. 0=big endian data translation disables 1=big endian data translation enables
u	RW	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

Description

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

Usage

This register is used for general draw operations.

See Also

DST_CNTL on page 3-46

SRC_CNTL on page 3-86

PAT_CNTL on page 3-75

HOST_CNTL on page 3-64

HOST_CNTL	MM:90h																													
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GX-0 GX-1 CX/EX																																a
GX-2																															b	a
CT																															b	a

DESCRIPTION			
a	RW	HOST_BYTE_ALIGN	1 = Host data byte align enabled
b	RW	HOST_BIG_ENDIAN_EN	Enables big endian data translation for 15 bpp, and 32 bpp pixel widths. In 15 bpp and 16 bpp modes the bytes within each word are swapped. In 32 bpp mode the order of the four bytes within each dword is reversed. 0 = big endian data translation disables 1 = big endian data translation enables

Description

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

Usage

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

See Also

GUI_TRAJ_CNTL on page 3-62

HOST_DATA on page 3-65

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

- *Programming Model: Logical Pixel Data Path: Host Data Consumption*
- *Simple Draw Operations: Rectangle Fill*

HOST_DATA[0:15]																														MM:80h-8Fh
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
All *mach64*	a																															

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

Description

HOST_DATA is actually a single register mapped to 16 consecutive addresses, thus HOST_DATA[0:15]. 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.

See Also

BUS_CNTL on page 3-2

HOST_CNTL on page 3-64

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

- *Programming Model: Logical Pixel Data Path: Host Data Consumption*
- *Performance Issues*

MEM_CNTL													I/O:14h								MM:2Ch											
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GX-0 GX-1														i	h							g		f	e	d	c	b		a	
GX-2						n								i	h					m	l	g		k				b		j	
CX/EX														i	h					m	l	g		k				b		o	
CT						n								i	h					rl		q								p	

			DESCRIPTION
a	RW	MEM_SIZE	Memory size (Default = 512K) 1 = 1M 2 = 2M 3 = 4M 4 = 6M 5 = 8M 6 = *Reserved* 7 = *Reserved*
b	RW	MEM_RD_LATCH_EN	Latching of data on RAM port data enable
c	RW	MEM_RD_LATCH_DLY	Delay latching of data on RAM port by 1/2 memory clock period
d	RW	MEM_SD_LATCH_EN	Latching of memory serial access port data enable
e	RW	MEM_SD_LATCH_DLY	Delay latching of data on serial access port by 1/2 memory clock period
f	RW	MEM_FULL_PLS	One memory clock period set for width of data latch pulse
g	RW	MEM_CYC_LNTH	Non-page memory cycle length (Default = 2) 0 = 5 memory clock periods 1 = 6 memory clock periods 2 = 7 memory clock periods 3 = *Reserved*
h	RW	MEM_BNDRY	Memory boundary 0 = 0K 1 = 256K 2 = 512K 3 = 1M
i	RW	MEM_BNDRY_EN	Memory boundary enable
j	RW	MEM_SIZE	Memory size (Default = 1M) 0 = *Reserved* 1 = 1M 2 = 2M 3 = 4M 4 = 6M 5 = 8M 6 = *Reserved* 7 = *Reserved*
k	RW	MEM_FULL_PLS	Sets the memory latch pulse delay to 1 memory clock period. The default is 1/2 memory clock period

(Continued on next page)

MEM_CNTL																															
I/O:14h MM:2Ch																															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GX-0 GX-1														i	h							g		f	e	d	c	b		a	
GX-2						n								i	h					m	l	g		k				b		j	
CX/EX														i	h					m	l	g		k				b		o	
CT						n								i	h					r		q								p	

DESCRIPTION			
i	RW	MEM_WR_RDY_SEL	Delays the memory WE signal transition by 1/2 memory clock period. 0 = no delay of WE signal transition 1 = delay WE signal transition
m	RW	MEM_EXT_RMW_CYC_EN	Extends the length of the page write cycle to 3 memory clocks in read-modify-write operations (default=0). 0 = 2 clock page write cycle 1 = 3 clock page write cycle
n	RW	MEM_PIX_WIDTH	Big endian memory aperture pixel width (this field is only used in PCI bus configuration with the CFG_BIG_ENDIAN_EN strap set) 0 = 1 bpp 1 = 4 bpp 2 = 8 bpp 3 = 15 bpp 4 = 16 bpp 5 = 24 bpp 6 = 32 bpp 7 = *Reserved*
o	RW	MEM_SIZE	Memory size (Default = 512K) 0 = 512K 1 = 1M 2 = 2M 3 = 4M 4 = *Reserved* 5 = *Reserved* 6 = *Reserved* 7 = *Reserved*
p	RW	MEM_SIZE	Memory size: (default = 0) 0 = *Reserved* 1 = 1M 2 = 2M 3 = 4M 4 - 7 =*Reserved*
q	RW	MEM_CYC_LNTH	Non-page memory cycle latency: (default = 0) 0 = 7 memory clock periods 1 = 6 memory clock periods (Turbo CAS) 2 = 6 memory clock periods (Turbo RAS) 3 = 5 memory clock periods
r	RW	MEM_REFRESH_RATE	Memory refresh rate: (default = 3) 0 = 1 refresh rate per Horizontal Total 1 = 2 refresh rate per Horizontal Total 2 = 3 refresh rate per Horizontal Total 3 = 4 refresh rate per Horizontal Total

Description

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

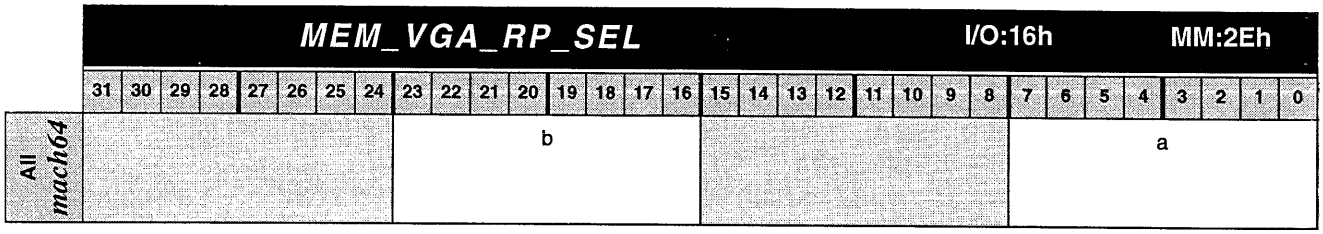
Usage

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

See Also

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

- *Programming Model: VGA Interaction*
- *Advanced Topics II: Boot-time Initialization*



			DESCRIPTION
a	RW	MEM_VGA_RPS0	Read page pointer for the lower 32K aperture, for up to 8M of video memory
b	RW	MEM_VGA_RPS1	Read page pointer for the upper 32K aperture, for up to 8M of video memory

Description

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

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

Usage

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

See Also

CONFIG_CNTL on page 3-9

MEM_VGA_WP_SEL on page 3-71

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

- *Programming Model: The Linear and Paged Memory Apertures*

MEM_VGA_WP_SEL																				I/O:15h								MM:2Dh			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
All *mach64*								b																a							

DESCRIPTION			
a	RW	MEM_VGA_WPS0	Write page pointer for the lower 32K aperture, for up to 8M of video memory
b	RW	MEM_VGA_WPS1	Write page pointer for the upper 32K aperture, for up to 8M of video memory

Description

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

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

Usage

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

See Also

CONFIG_CNTL on page 3-9

MEM_VGA_RP_SEL on page 3-70

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

- *Programming Model: The Linear and Paged Memory Apertures*

OVR_CLR																I/O:8h								MM:10h							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
All *mach64*	d								c								b								a							

			DESCRIPTION
a	RW	OVR_CLR_8	Overscan color for 4 bpp and 8 bpp display modes
b	RW	OVR_CLR_B	Blue overscan color
c	RW	OVR_CLR_G	Green overscan color
d	RW	OVR_CLR_R	Red overscan color

Description

This register specifies the overscan color.

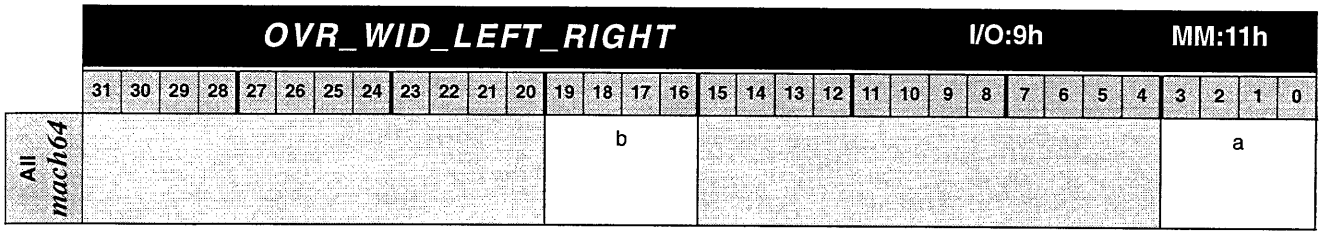
Usage

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

See Also

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

- *Programming Model: Mode Switching: Designing a Custom CRT Mode*



			DESCRIPTION
a	RW	OVR_WID_LEFT	Left overscan width
b	RW	OVR_WID_RIGHT	Right overscan width

Description

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

Usage

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

See Also

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

- *Programming Model: Mode Switching: Designing a Custom CRT Mode*

OVR_WID_TOP_BOTTOM																																I/O:Ah		MM:12h	
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
All *mach64*									b																a										

		DESCRIPTION	
a	RW	OVR_WID_TOP	Top overscan width
b	RW	OVR_WID_BOTTOM	Bottom overscan width

Description

OVR_WID_TOP_BOTTOM specifies the top and bottom overscan widths, in lines.

Usage

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

See Also

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

- *Programming Model: Mode Switching: Designing a Custom CRT Mode*

PAT_CNTL — MM:A2h

All *mach64*

31	30	29	28	27	26	25	24	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

			DESCRIPTION
a	RW	PAT_MONO_EN	Monochrome 8x8 pattern enable
b	RW	PAT_CLR_4x2_EN	Color 4x2 pattern enable
c	RW	PAT_CLR_8x1_EN	Color 8x1 pattern enable

Description

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

Usage

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

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

See Also

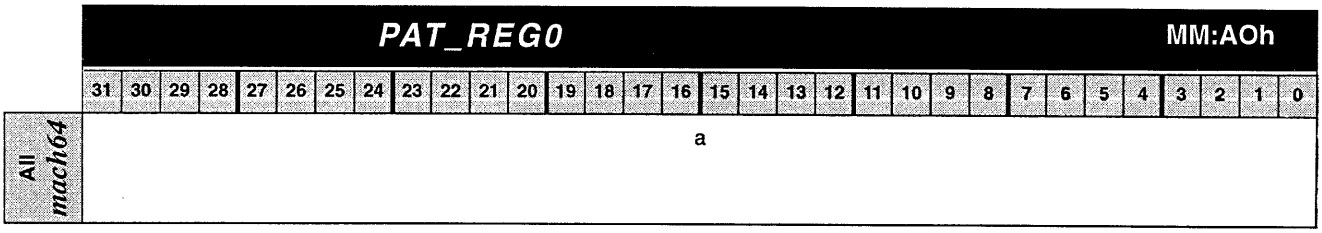
GUI_TRAJ_CNTL on page 3-62

PAT_REG0 on page 3-76

PAT_REG1 on page 3-77

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

- *Programming Model: Logical Pixel Data Path*
- *Programming Model: Logical Pixel Data Path: Pattern Consumption*
- *Simple Draw Operations: Fixed Patterns*



DESCRIPTION			
a	RW	PAT_REG0	Pattern register 0

Description

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

Usage

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

See Also

PAT_CNTL on page 3-75

PAT_REG1 on page 3-77

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

- *Programming Model: Logical Pixel Data Path*
- *Programming Model: Logical Pixel Data Path: Pattern Consumption*
- *Simple Draw Operations: Fixed Patterns*

PAT_REG1 MM:A1h

31	30	29	28	27	26	25	24	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																															

All *mach64*

DESCRIPTION			
a	RW	PAT_REG1	Pattern register 1

Description

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

Usage

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

See Also

PAT_CNTL on page 3-75

PAT_REG0 on page 3-76

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

- *Programming Model: Logical Pixel Data Path*
- *Programming Model: Logical Pixel Data Path: Pattern Consumption*
- *Simple Draw Operations: Fixed Patterns*

SC_BOTTOM MM:ACh

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
All *mach64*																		a														

DESCRIPTION			
a	RW	SC_BOTTOM	Bottom scissor

Description

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

Usage

This register must be set for all draw operations.

See Also

SC_TOP on page 3-82

SC_TOP_BOTTOM on page 3-83

SC_LEFT on page 3-79

SC_RIGHT on page 3-81

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

- *Programming Model: Scissoring and Masking*

SC_LEFT MM:A8h

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
All *mach64*																				a												

DESCRIPTION			
a	RW	SC_LEFT	Left scissor

Description

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

Usage

This register must be set for all draw operations.

See Also

SC_TOP on page 3-82

SC_BOTTOM on page 3-78

SC_RIGHT on page 3-81

SC_LEFT_RIGHT on page 3-80

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

- *Programming Model: Scissoring and Masking*

SC_LEFT_RIGHT MM:AAh

All *mach64*

31	30	29	28	27	26	25	24	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												

		DESCRIPTION	
a	W	SC_LEFT	Left scissor
b	W	SC_RIGHT	Right scissor

Description

SC_LEFT_RIGHT is a composite of registers SC_LEFT and SC_RIGHT.

Usage

This register must be set for all draw operations.

See Also

SC_LEFT on page 3-79

SC_RIGHT on page 3-81

SC_RIGHT																														MM:A9h
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
All *mach64*																				a												

DESCRIPTION			
a	RW	SC_RIGHT	Right scissor

Description

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

Usage

This register must be set for all draw operations.

See Also

SC_TOP on page 3-82

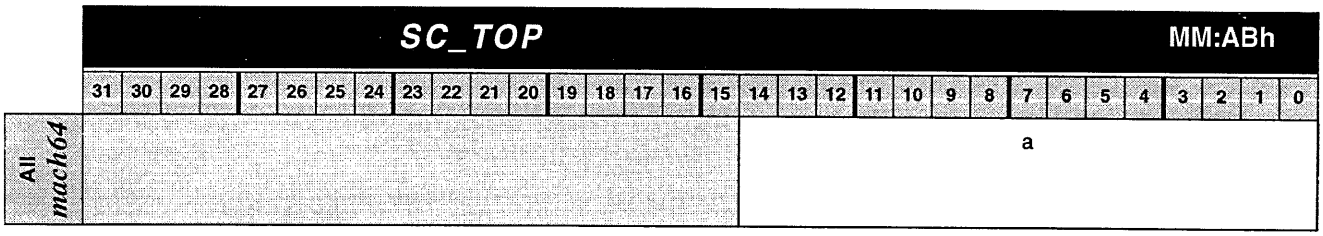
SC_LEFT on page 3-79

SC_LEFT_RIGHT on page 3-80

SC_BOTTOM on page 3-78

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

- *Programming Model: Scissoring and Masking*



SC_TOP																															
MM:ABh																															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
All *mach64*																		a													

DESCRIPTION			
a	RW	SC_TOP	Top scissor

Description

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

Usage

This register must be set for all draw operations.

See Also

SC_BOTTOM on page 3-78

SC_LEFT on page 3-79

SC_RIGHT on page 3-81

SC_TOP_BOTTOM on page 3-83

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

- *Programming Model: Scissoring and Masking*

SC_TOP_BOTTOM MM:ADh

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
All *mach64*		b																a														

			DESCRIPTION
a	W	SC_TOP	Top scissor
b	W	SC_BOTTOM	Bottom scissor

Description

SC_TOP_BOTTOM is a composite of registers SC_TOP and SC_BOTTOM.

Usage

This register must be set for all draw operations.

See Also

SC_TOP on page 3-82

SC_BOTTOM on page 3-78

SCRATCH_REG0 (Test Mode 0) I/O:10h MM:20h

All *mach64*

31	30	29	28	27	26	25	24	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																															

DESCRIPTION			
a	RW	SCRATCH_REG0	Scratch pad 0

Description

SCRATCH_REG0 is a general purpose storage register. In test modes, this register is aliased to a diagnostic register.

Usage

Only the adapter BIOS should touch this register.

See Also

SCRATCH_REG1 on page 3-85

TEST_REG0 on page 3-101

TEST_REG2 on page 3-103

TEST_REG4 on page 3-105

TEST_REG5 on page 3-106

TEST_REG7 on page 3-109

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

- *Advanced Topics II: Boot-time Initialization*

SCRATCH_REG1 (Test Mode 0)																																I/O:11h	MM:21h
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
All *mach64*	a																																	

DESCRIPTION			
a	RW	SCRATCH_REG1	Scratch pad 1

Description

SCRATCH_REG1 is a general purpose storage register. In test modes, this register is aliased to a diagnostic register.

Usage

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

See Also

SCRATCH_REG0 on page 3-84

TEST_REG1 on page 3-102

TEST_REG3 on page 3-104

TEST_REG6 on page 3-107

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

- *Advanced Topics II: Boot-time Initialization*

SRC_CNTL I/O: MM:6Dh

All *mach64*

31	30	29	28	27	26	25	24	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

			DESCRIPTION
a	RW	SRC_PATT_EN	General pattern — SRC_Y_END will be used only if this bit is enabled
b	RW	SRC_PATT_ROT_EN	General pattern with rotation — SRC_X_START and SRC_Y_START will be used only if this bit is enabled
c	RW	SRC_LINEAR_EN	Linearly advanced source enable — the source starts at SRC_OFFSET, and advances in the left-to-right direction. Note: DST_X_DIR should be set to left-to-right for proper operation. Note: All other source registers with the exception of SRC_BYTE_ALIGN are ignored
d	RW	SRC_BYTE_ALIGN	Source to skip to the next data byte boundary, if not at boundary when the destination advances in the Y direction. Note: SRC_LINEAR_EN must be set. This is only for 1 bpp and 4 bpp source pixel width definitions.
e	RW	SRC_LINE_X_DIR	Source X direction when drawing operation is a Bresenham line.

Description

SRC_CNTL contains various enable bits for blit source trajectory control.

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

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

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

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

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

Usage

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

See Also

DST_CNTL on page 3-46

GUI_TRAJ_CNTL on page 3-62

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

- *Programming Model: Trajectories*
- *Programming Model: Source and Destination Alignment*
- *Simple Draw Operations: Bitblit*

SRC_HEIGHT1 MM:65h

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
All *mach64*																		a														

			DESCRIPTION
a	RW	SRC_HEIGHT1	Source height 1

Description

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

Usage

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

See Also

SRC_HEIGHT1_WIDTH1 on page 3-89

SRC_WIDTH1 on page 3-93

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

- *Programming Model: Trajectories: Source Trajectory 3, General Pattern*
- *Programming Model: Trajectories: Source Trajectory 4, General Pattern with Rotation*
- *Simple Draw Operations: Bitblit: General Pattern*
- *Simple Draw Operations: Bitblit: General Pattern with Rotation*

SRC_HEIGHT1_WIDTH1 MM:66h

All *mach64*

31	30	29	28	27	26	25	24	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														

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

Description

This register is a composite of SRC_HEIGHT1 and SRC_WIDTH1.

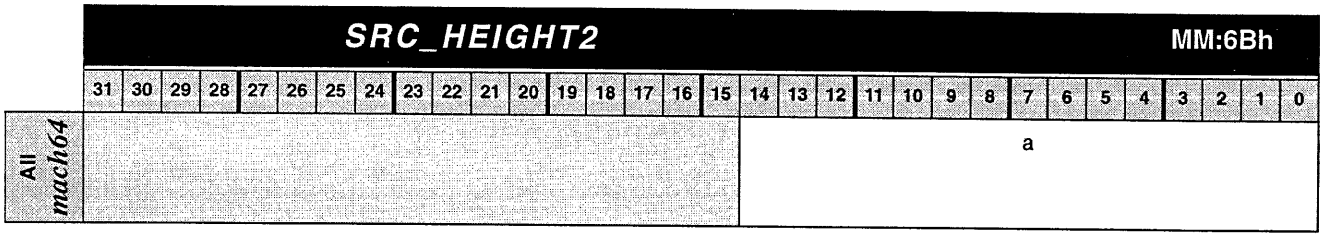
Usage

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

See Also

SRC_HEIGHT1 on page 3-88

SRC_WIDTH1 on page 3-93



DESCRIPTION			
a	RW	SRC_HEIGHT2	Source height 2

Description

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

Usage

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

See Also

SRC_HEIGHT2_WIDTH2 on page 3-91

SRC_WIDTH2 on page 3-94

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

- *Programming Model: Trajectories: Source Trajectory 4, General Pattern with Rotation*
- *Simple Draw Operations: Bitblit: General Pattern with Rotation*

SRC_HEIGHT2_WIDTH2 MM:6Ch

All *mach64*

31	30	29	28	27	26	25	24	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														

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

Description

This register is a composite of SRC_HEIGHT2 and SRC_WIDTH2.

Usage

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

See Also

SRC_HEIGHT2 on page 3-90

SRC_WIDTH2 on page 3-94

SRC_OFF_PITCH																												MM:60h
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
All *mach64*	b												a																			

			DESCRIPTION
a	RW	SRC_OFFSET	Source offset address, in terms of 64-bit words
b	RW	SRC_PITCH	Source pitch in pixel-times-8. Note: In monochrome modes the destination pitch must be a multiple of 64 pixels, in 4 bpp modes it must be a multiple of 16 pixels.

Description

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

Usage

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

See Also

DST_OFF_PITCH on page 3-50

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

- *Programming Model: Trajectories: Source Trajectory 1, Strictly Linear*
- *Programming Model: Trajectories: Source Trajectory 2, Unbounded Y*
- *Programming Model: Trajectories: Source Trajectory 3, General Pattern*
- *Programming Model: Trajectories: Source Trajectory 4, General Pattern with Rotation*

SRC_WIDTH1																														MM:64h
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
All *mach64*																				a												

		DESCRIPTION	
a	RW	SRC_WIDTH1	Source width 1

Description

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

Usage

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

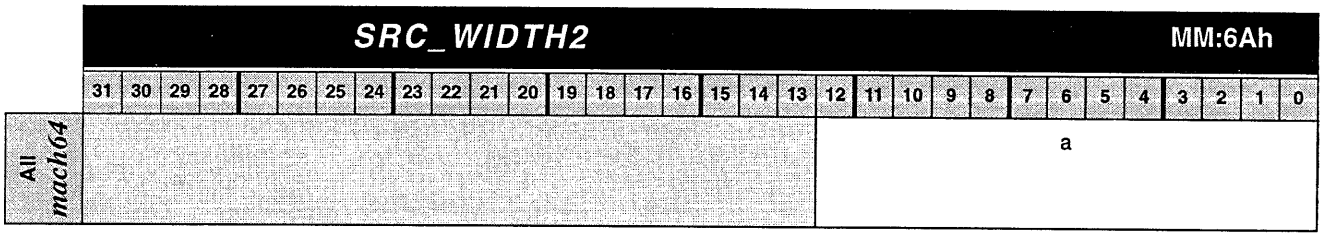
See Also

SRC_HEIGHT1 on page 3-88

SRC_HEIGHT1_WIDTH1 on page 3-89

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

- *Programming Model: Trajectories: Source Trajectory 2, Unbounded Y*
- *Programming Model: Trajectories: Source Trajectory 3, General Pattern*
- *Programming Model: Trajectories: Source Trajectory 4, General Pattern with Rotation*
- *Simple Draw Operations: Bitblit: Unbounded Y*
- *Simple Draw Operations: Bitblit: General Pattern*
- *Simple Draw Operations: Bitblit: General Pattern with Rotation*



SRC_WIDTH2																														MM:6Ah
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
All *mach64*																				a												

DESCRIPTION			
a	RW	SRC_WIDTH2	Source width 2

Description

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

Usage

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

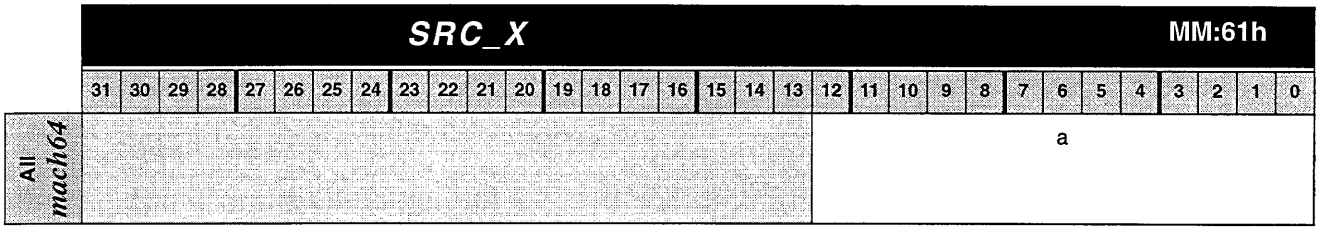
See Also

SRC_HEIGHT2 on page 3-90

SRC_HEIGHT2_WIDTH2 on page 3-91

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

- *Programming Model: Trajectories: Source Trajectory 4, General Pattern with Rotation*
- *Simple Draw Operations: Bitblit: General Pattern with Rotation*



DESCRIPTION			
a	RW	SRC_X	Source X coordinate

Description

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

Usage

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

See Also

SRC_Y on page 3-97

SRC_Y_X on page 3-99

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

- *Programming Model: Trajectories: Source Trajectory 1, Strictly Linear*
- *Programming Model: Trajectories: Source Trajectory 2, Unbounded Y*
- *Programming Model: Trajectories: Source Trajectory 3, General Pattern*
- *Programming Model: Trajectories: Source Trajectory 4, General Pattern with Rotation*

SRC_X_START MM:67h

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
All *mach64*																				a												

DESCRIPTION			
a	RW	SRC_X_START	Pattern source X start for pattern rotation in the X direction

Description

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

Usage

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

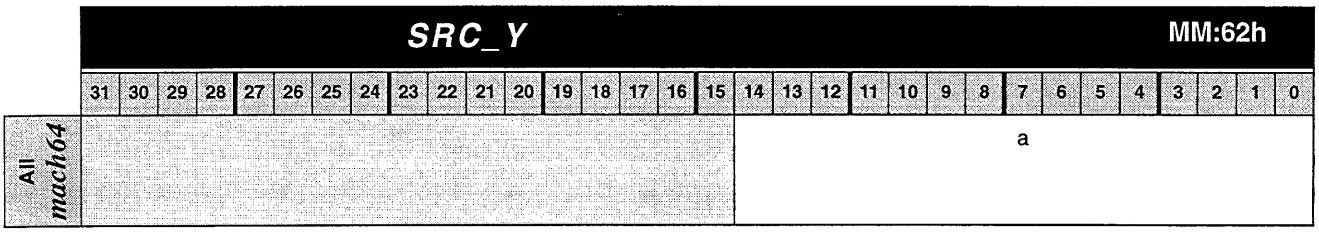
See Also

SRC_Y_START on page 3-98

SRC_Y_X_START on page 3-100

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

- *Programming Model: Trajectories: Source Trajectory 4, General Pattern with Rotation*
- *Simple Draw Operations: Bitblit: General Pattern with Rotation*



			DESCRIPTION
a	RW	SRC_Y	Source Y coordinate

Description

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

Usage

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

See Also

SRC_X on page 3-95

SRC_Y_X on page 3-99

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

- *Programming Model: Trajectories: Source Trajectory 1, Strictly Linear*
- *Programming Model: Trajectories: Source Trajectory 2, Unbounded Y*
- *Programming Model: Trajectories: Source Trajectory 3, General Pattern*
- *Programming Model: Trajectories: Source Trajectory 4, General Pattern with Rotation*

SRC_Y_START MM:68h

All *mach64*	31	30	29	28	27	26	25	24	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															

			DESCRIPTION
a	RW	SRC_Y_START	Pattern source Y start for pattern rotation in the Y direction

Description

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

Usage

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

See Also

SRC_X_START on page 3-96

SRC_Y_X_START on page 3-100

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

- *Programming Model: Trajectories: Source Trajectory 4, General Pattern with Rotation*
- *Simple Draw Operations: Bitblit: General Pattern with Rotation*

SRC_Y_X MM:63h

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

Description

This register is a composite of SRC_Y and SRC_X.

Usage

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

See Also

SRC_Y on page 3-97

SRC_X on page 3-95

SRC_Y_X_START																												MM:69h
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
All *mach64*				b														a														

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

Description

This register is a composite of SRC_X_START and SRC_Y_START.

Usage

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

See Also

SRC_X_START on page 3-96

SRC_Y_START on page 3-98

TEST_REG0 (Test Mode 1)	I/O:10h	MM:20h

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GX-0 GX-1												a																				
GX-2 CX/EX CT	NOT IMPLEMENTED																															

		DESCRIPTION	
a	RW	TEST_MEM_ADDR	DWORD memory address

Description

This register is aliased with SCRATCH_REG0 and is used only in test mode 1 to specify a DWORD memory address for the test mode.

Usage

This register is intended for memory diagnostics.

See Also

TEST_REG1 on page 3-102

GEN_TEST_CNTL on page 3-57

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

- *Advanced Topics II: Diagnostic Features*

TEST_REG1 (Test Mode 1)																															
	I/O:11h																MM:21h														
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GX-0 GX-1	a																														
GX-2 CX/EX CT	NOT IMPLEMENTED																														

			DESCRIPTION
a	RW	TEST_MEM_DATA	The data word which is read from or written to memory at DWORD address TEST_MEM_ADDR@TEST_REG0 when GEN_TEST_MEM_STROBE@GEN_TEST_CNTL = 1. GEN_TEST_MEM_WR@GEN_TEST_CNTL determines if the operation is a memory read or write.

Description

This register is aliased with SCRATCH_REG1 and is used only in test mode 1. This register holds the write data for memory-write tests or the latched memory data in memory-read tests as determined by GEN_TEST_MEM_WR@GEN_TEST_CNTL.

Usage

This register is intended for memory diagnostics.

See Also

TEST_REG0 on page 3-101

GEN_TEST_CNTL on page 3-57

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

- *Advanced Topics II: Diagnostic Features*

TEST_REG2 (Test Mode 2)																															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GX-0 GX-1 GX-2 CX/EX		b																		a												
CT	NOT IMPLEMENTED																															

I/O:10h MM:20h

			DESCRIPTION
a	R	DST_X_LNTH_CNTR	Destination X coordinate length counter
b	R	DST_Y_LNTH_CNTR	Destination Y coordinate length counter

Description

This register is aliased with SCRATCH_REG0 and is used only in test mode 2. This is a read-only register. It returns the contents of the internal destination length counters. This register is designed to be used when the draw engine is in destination single step mode.

Usage

This register is intended for validating the correct operation of the internal destination length counters in a diagnostic program.

See Also

TEST_REG3 on page 3-104

TEST_REG4 on page 3-105

GEN_TEST_CNTL on page 3-57

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

- *Advanced Topics II: Diagnostic Features*

TEST_REG3 (Test Mode 2)																		I/O:11h								MM:21h			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GX-0 GX-1 GX-2 CX/EX		b																		a												
CT	NOT IMPLEMENTED																															

		DESCRIPTION	
a	R	SRC_X_LNTH_CNTR	Source X coordinate length counter
b	R	SRC_Y_LNTH_CNTR	Source Y coordinate length counter

Description

This register is aliased with SCRATCH_REG1 and is used only in test mode 2. This is a read-only register. It returns the contents of the internal source length counters. This register is designed to be used when the draw engine is in source single step mode.

Usage

This register is intended for validating the correct operation of the internal source length counters in a diagnostic program.

See Also

TEST_REG2 on page 3-103

TEST_REG4 on page 3-105

GEN_TEST_CNTL on page 3-57

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

- *Advanced Topics II: Diagnostic Features*

TEST_REG4 (Test Mode 3)																				I/O:10h								MM:20h			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GX-0 GX-1 GX-2 CX/EX																				a												
CT	NOT IMPLEMENTED																															

		DESCRIPTION	
a	R	SRC_READ_LNTH_CNTR	Source FIFO read length counter

Description

This register is aliased with SCRATCH_REG0 and is used only in test mode 3. This is a read-only register. It returns the contents of the internal source FIFO length counters. This register is designed to be used when the draw engine is in source single step mode.

Usage

This register is intended for validating the correct operation of the internal source FIFO length counters in a diagnostic program.

See Also

TEST_REG2 on page 3-103

TEST_REG3 on page 3-104

GEN_TEST_CNTL on page 3-57

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

- *Advanced Topics II: Diagnostic Features*

TEST_REG5 (Test Mode 4)																I/O:10h								MM:20h							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GX-0 GX-1 GX-2 CX/EX				d					c					b											a							
CT	NOT IMPLEMENTED																															

			DESCRIPTION
a	RW	CRTC_H_CHAR_CNTR	CRTC horizontal character counter
b	RW	CRTC_V_LINE_CNTR	CRTC vertical line counter
c	RW	CRTC_H_SYNC_WID_CNTR	CRTC horizontal sync width counter
d	RW	CRTC_V_SYNC_WID_CNTR	CRTC vertical sync width counter

Description

This register is aliased with SCRATCH_REG0 and is used only in test mode 4. Internal CRTC counters may be read or written in this test mode. This register is designed to be used when the CRTC is in single step mode.

Usage

This register is used for diagnosing the CRTC functionality.

See Also

TEST_REG6 on page 3-107

GEN_TEST_CNTL on page 3-57

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

- *Advanced Topics II: Diagnostic Features*

TEST_REG6 (Test Mode 4)																													
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GX-0 GX-1 GX-2 CX/EX						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
CT	NOT IMPLEMENTED																															

			DESCRIPTION
a	R	CRTC_H_TOTAL_EQ	Horizontal counter = CRTC_H_TOTAL
b	R	CRTC_H_TOTAL_HALF_EQ	Horizontal counter = (CRTC_H_TOTAL/2)
c	R	CRTC_H_DISP_EQ	Horizontal counter = CRTC_H_DISP
d	R	CRTC_H_SYNC_STRT_EQ	Horizontal counter = CRTC_H_SYNC_STRT
e	R	CRTC_H_SYNC_WID_EQ	Horizontal sync counter = CRTC_H_SYNC_WID
f	R	CRTC_V_TOTAL_EQ	Vertical counter = CRTC_V_TOTAL
g	R	CRTC_V_DISP_EQ	Vertical counter = CRTC_V_DISP
h	R	CRTC_V_SYNC_STRT_EQ	Vertical counter = CRTC_V_SYNC_STRT
i	R	CRTC_V_SYNC_WID_EQ	Vertical sync width counter = CRTC_V_SYNC_WID
j	R	CRTC_V_LINE_EQ	Vertical counter = CRTC_V_LINE
k	R	CRTC_H_DISP_EN	Horizontal display enable
l	R	CRTC_V_DISP_EN	Vertical display enable
m	R	CRTC_DISP_EN	Display enable
n	R	CRTC_BLANK	Display blank
o	R	CRTC_H_SYNC	Horizontal sync
p	R	CRTC_V_SYNC	Vertical sync
q	R	CRTC_FRAME	Odd/even frame tag (strobed when CRTC_V_TOTAL_EQ = 1)
r	R	CRTC_FRAME_X	Odd/even frame tag (strobed when CRTC_V_DISP_EQ = 1)
s	R	OVR_LEFT_EQ	Horizontal counter = (CRTC_H_TOTAL) - (OVR_WID_LEFT)
t	R	OVR_RIGHT_EQ	Horizontal counter = (CRTC_H_DISP) +(OVR_WID_RIGHT)
u	R	OVR_TOP_EQ	Vertical counter = (CRTC_V_TOTAL) - (OVR_WID_TOP)
v	R	OVR_BOTTOM_EQ	Vertical counter = (CRTC_V_TOTAL) + (OVR_WID_BOTTOM)
w	R	OVR_LEFT_EN	Left overscan enable
x	R	OVR_RIGHT_EN	Right overscan enable
y	R	OVR_TOP_EN	Top overscan enable
z	R	OVR_BOTTOM_EN	Bottom overscan enable
aa	R	OVR_EN	Overscan enable

Description

This register is aliased with SCRATCH_REG1 and is used only in test mode 4. It returns the status of internal CRTC comparators and enable bits. This register is designed to be used when the CRTC is in single step mode.

Usage

This register is used for diagnosing the CRTC functionality.

See Also

TEST_REG5 on page 3-106

GEN_TEST_CNTL on page 3-57

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

- *Advanced Topics II: Diagnostic Features*

TEST_REG7 (Test Mode 5)																															
I/O:10h MM:20h																															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GX-0 GX-1									a																						
GX-2 CX/EX CT	NOT IMPLEMENTED																														

DESCRIPTION			
a	R	PIXEL_DATA_CRC	Pixel data CRC

Description

This register is used to accumulate the CRC of the digital pixel stream from the CRTC. Use GEN_TEST_CRC_STR@GEN_TEST_CNTL and GEN_TEST_CRC_DONE@GEN_TEST_CNTL to begin a CRC check and detect when it is finished.

Usage

This register is used to diagnose the hardware cursor, overscan, and the CRTC DAC interface.

See Also

GEN_TEST_CNTL on page 3-57

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

- *Advanced Topics II: Diagnostic Features*

This page intentionally left blank.

Index

M

O

P

R

S

T

Perfecting the PC

User Response Form

In our continuing effort to improve our products and documentation, ATI Technologies Inc. is anxious to obtain your feedback on this manual. Using a scale of 1 to 5, with 5 representing the most favorable, and 1 the least favorable, please circle the number that best reflects your opinion. Everyone who completes and returns this questionnaire will receive an ATI souvenir (e.g. golf shirt, mug etc.) depending on availability.

Name and Organization _____

Title _____ **Years in position** _____

Name of Manual *mach64* Register Reference_____ **Release No.** _____

How do you rate the manual, generally?	*1*	*2*	*3*	*4*	*5*
Is the technical level of the manual appropriate?	*1*	*2*	*3*	*4*	*5*
Are operating instructions clear and complete?	*1*	*2*	*3*	*4*	*5*
Are terms and concepts explained clearly?	*1*	*2*	*3*	*4*	*5*
Is the manual well organized?	*1*	*2*	*3*	*4*	*5*
Are the tables easy to understand?	*1*	*2*	*3*	*4*	*5*
Is the artwork clear and easy to understand?	*1*	*2*	*3*	*4*	*5*
How helpful is the index?	*1*	*2*	*3*	*4*	*5*
How does this manual compare with other, similar manuals you have used?	*1*	*2*	*3*	*4*	*5*

Comments

What did you like best about this manual ? _____

_____

What did you like least about this manual ? _____

_____

Are there any errors/omissions in the manual? _____

_____

THANK YOU FOR YOUR COMMENTS! Please mail to: ATI Technologies Inc., 33 Commerce Valley Drive East, Thornhill, Ontario, Canada L3T 7N6. You may also fax this to (905) 882-2620 (attn: Technical Publications)

This page intentionally left blank

Perfecting the PC

Printed in Canada