1.0 Hombre Peripheral Chip

1.1 Peripheral Pins

Clock signals

Name	*#*	*Type*	*Description*
sys_clk	1	Input	50 Mhz system clock
sys_reset0	1	Input	System reset

Peripheral Component Interface (PCI)

Name	*#*	*Type*	*Description*
pci_ad[31:0]	32	In/Out	PCI Mulitplexed Address/Data
pci_c1_be0[3:0]	4	In/Out	PCI Bus Command - Byte Enables
pci_par	1	In/Out	PCI Parity
pci_frame0	1	In/Out	PCI Cycle Frame
pci_trdy0	1	In/Out	PCI Target Ready
pci_irdy0	1	In/Out	PCI initiator Ready
pci_stop0	1	In/Out	PCI Stop
pci_devsel0	1	In/Out	PCI Device Select
pci_id_sel	1	In/Out	PCI Initialization Device Select
pci_req0[5:0]	6	Input	PCI Requests to arbiter
pci_gnt[[5:0]	6	Output	PCI Grants from arbiter
pci_clk	1	Output	PCI clock
pci_rst	1	In/Out	PCI reset

SCSI Interface

Name	*#*	*Type*	*Description*
scsi_ack0	1	Input	SCSI Data handshake signal from initiator device
scsi_req0	1	Input	SCSI Data handshake signal from target device
scsi_bsy0	1	Input	SCSI bus arbitration signal, busy
scsi_sel0	1	Input	SCSI bus arbitration signal, select device
scsi_i_o0	1	Input	SCSI phase line, input/output
scsi_c_d0	1	Input	SCSI phase line, command/data
scsi_msg0	1	Input	SCSI phase line, message
scsi_atn0	1	Input	SCSI attention, the initiator is requesting a message.
scsi_rst0	1	Input	SCSI bus reset
scsi_d1[7:0]	1	In/Out	SCSI data
scsi_dbp0	1	In/Out	SCSI parity

Floppy Disk

Name	*#*	*Type*	*Description*
fd_rd0	1	Input	Floppy disk read data
fd_wd0	1	Output	Floppy disk write data
fd_we0	1	Output	Floppy disk write enable
fd_rdy0	1	Input	Floppy disk ready
fd_mtr0	1	Output	Floppy disk motor control
fd_sel0[1:0]	2	Output	Floppy disk select lines
fd_chng0	1	Input	Floppy disk changed
fd_side0	1	Output	Floppy disk side select
fd_wprot0	1	Input	Floppy disk write protect
fd_trk00	1	Input	Floppy disk track 0 detect
fd_step0	1	Output	Floppy disk motor step control

fd_dir	1	Output	Floppy disk motor direction
fd_index0	1	Input	Floppy disk index detect
fd_eject0	1	Output	Floppy disk eject disk (future)

Keyboard Interface

Name	*#*	*Type*	*Description*
kb_clk	1	Output	Serial Keyboard clock
kb_data	1	Input	Serial Keyboard data
kb_reset0	1	Input	Serial Keyboard reset

Parallel Port

Name	*#*	*Type*	*Description*
prll_d1[7:0]	8	In/Out	Parallel Port data
prll_ack0	1	Input	Parallel Port acknowledge
prll_busy	1	Output	Parallel Port busy
prll_pout	1	Output	Parallel Port direction
prll_sel	1	Input	Parallel Port select
prll_strb	1	Input	Parallel Port strobe

UART Interface

Name	*#*	*Type*	*Description*
uart_rxd[1:0]	2	Input	Receive data
uart_txd[1:0]	2	Output	Transmit data
uart_rts0[1:0]	2	Output	Request to send
uart_cts0[1:0]	2	Input	Clear to send
uart_dsr0[1:]	2	Input	Data set ready
uart_cd0[1:0]	2	Input	Carrier detect
uart_dtr0[1:0]	2	Output	Data terminal ready
uart_ri0[1:0]	2	Input	Ring Indicator

Test Signals

Name	*#*	*Type*	*Description*
jtag_tclk	1	Input	JTAG Clock
jtag_tms	1	Input	JTAG Test Mode Select
jtag_di	1	Input	JTAG Data Input
jtag_do	1	Input	JTAG Data Output

Supplies

Name	*#*	*Type*	*Description*
Power	10	VDD	Power
Ground	20	VSS	Ground

1.0 Assumptions:

1. The time frame for this chip set is CY 2H 1994.

2. In the time frame specified:

a. The technology of choice will be 0.6u three-layer metal with stacked contacts.

b. 60ns/page, 20ns/serial VRAMS will be commonplace.

3. The chipset is being designed to work with RISC processors. In particular, a 64 bit data bus is assumed for the high end. A version which bonds out 32 bits will support low end applications where an external RISC is not necessary.

4. Given that the processor will change, code from AMIGAs will not be compatible and other changes to the graphics, etc. can be made without fear of incompatibilities...

5. We have the license to "clean things up", throw away features that are deemed not appropriate, and add features which we feel are necessary and desirable.

Definitions:

byte: 8 bit quantity

Halfword: 16 bit quantity

Word: 32 bit quantity

Doubleword: 64 bit quantity

Chunky pixel: An 8 bit quantity which contains an indexed color.

Canvas: An array of memory which contains chunky pixels. Must have a width which is a multiple of 8 pixels.

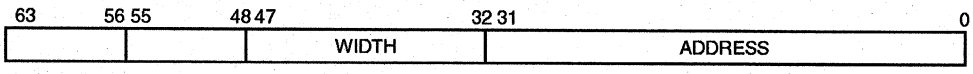
Appendix A Blitter Registers

Name: **BLT_ADDR_x**

Address: **0x**

Access:

Format:



Purpose: BLT_ADDR_x, 0 <= x <= 2 contain the addresses used by rectangular blits. BLT_ADDR_0 and BLT_ADDR_1 are used for sources; BLT_ADDR_2 is the destination address. Each contain a full byte address. The least significant 3 bits are used as byte specifiers within a 64 bit word. Pixels must be aligned on natural boundaries.

Fields:

ADDRESS: The pixel address of the rectangle to be used in the blit operation..

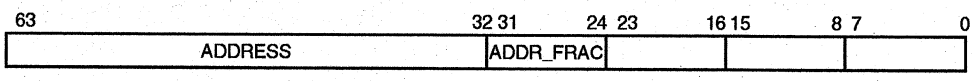
WIDTH: The width of the canvas that this rectangle is part of.

Name: **BLT_BRESH_ADDR_x**

Address: **0x**

Access:

Format:



Purpose: BLT_BRESH_ADDR_x, 0 <= x <= 2, is the register which contains the address used during line draws, area fills, and texture maps. BLT_BRESH_ADDR_0 is used during line draws. BLT_BRESH_ADDR_0 and BLT_BRESH_ADDR_1 are used during fills. BLT_BRESH_ADDR_0 defines the left line while BLT_BRESH_ADDR_1 defines the right line of the area being filled. All three are used for texture maps. BLT_BRESH_ADDR_2 defines the source line from the texture map. These registers contain a full byte address extended by 8 bits of fractional value. The least significant 3 bits of ADDRESS are used as byte specifiers within a 64 bit word. Pixels must be aligned on natural boundaries.

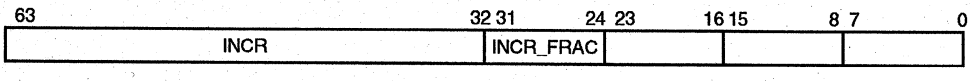
Fields: ADDRESS.ADDR_FRAC:The address of the starting pixel in the line.

Name: **BLT_BRESH_INCR_x_y**

Address: **0x**

Access:

Format:



Purpose: BLT_BRESH_INCR_x_y, 0 <= x <= 2, 0 <= y <= 1, contain the address increment values for the Breshenham address generators. The x index refers to one of three address generators. The y index refers to the two increment values within each of the address generators. Increment register 0 is the independent value; increment register 1 is the dependent value.

Fields:

INCR.INCR_FRAC: Value added to the Breshenham address (increment value 0 or 1) when appropriate.

ADDR_INCR_0: Defines an integer which is used as an address increment during blitter operations. Its use is a function of the blitter mode:

LINE_DRAW: ADDR_INCR_0 is added after each pixel calculation (Dindep).

SHADED_FILL: ADDR_INCR_0 is added to the left start position after each shaded line completion (DLx/Dy).

ADDR_INCR_1: Defines an integer which is used as an address increment during blitter operations. Its use is a function of the blitter mode:

LINE_DRAW: ADDR_INCR_1 is added when the error calculation crosses the zero threshold (Ddep).

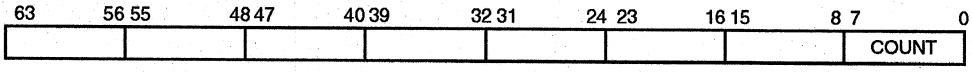
SHADED_FILL: ADDR_INCR_1 is added to the previous start position after each shaded line completion (DRx/Dy).

Name: **BLT_BRESH_SKIP**

Address: **0x**

Access:

Format:



Purpose: BLT_BRESH_SKIP contains an integer value used by the texture map Breshenham loop. It determines which points along a line drawn through a texture map will be used as a source for data being placed into a texture filled area.

Fields:

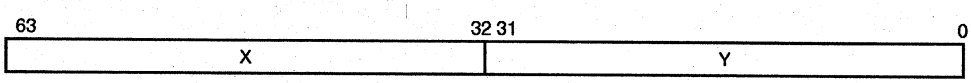
COUNT: The number of pixels between texture map samples being sourced for filling by the third Breshenham generator.

Name: **BLT_CLIP_BR**

Address: **0x**

Access:

Format:



Purpose: BLT_CLIP_BR contains pixel coordinates for the bottom right of a clipping rectangle.

Fields:

X: Defines an integer coordinate of the rightmost pixel to be included in a line draw operation.

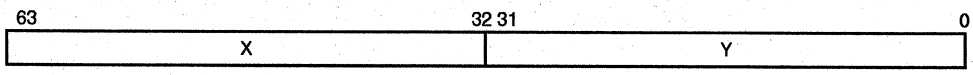
Y: Defines an integer coordinate of the bottommost pixel to be included in a line draw operation.

Name: **BLT_CLIP_UL**

Address: **0x**

Access:

Format:



Purpose: BLT_CLIP_UL contains pixel coordinates for the upper left of a clipping rectangle.

Fields:

X: Defines an integer coordinate of the leftmost pixel to be included in a line draw operation.

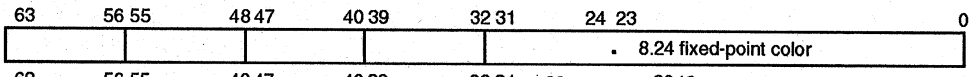
Y: Defines an integer coordinate of the topmost pixel to be included in a line draw operation.

Name: **BLT_COLOR_0**

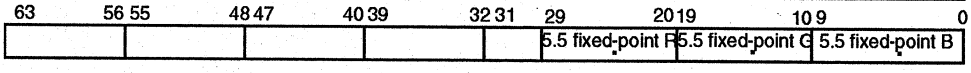
Address: **0x**

Access:

Format:



Format:



Purpose: BLT_COLOR_0 contains a color value for use by line draw, expand, and shaded fill operations.

Fields:

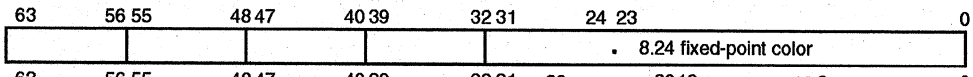
COLOR_0: Defines a fixed point value used by the line draw, expand, and shaded fill operations. When used with byte planes, the 32 bit value is interpretted as an 8.24 fixed-point number. When used with 16-bit pixels, the 32 bit value is interpretted as 3 5.5 fixed-point numbers.

Name: **BLT_COLOR_1**

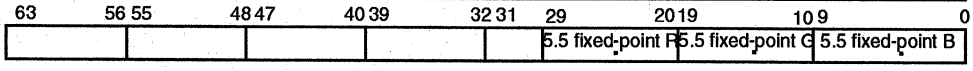
Address: **0x**

Access:

Format:



Format:



Purpose: BLT_COLOR_1 contains a color value for use by line draw, expand, and shaded fill operations.

Fields:

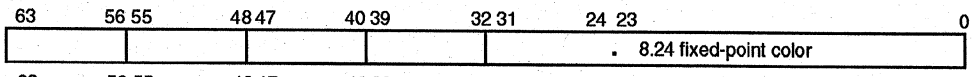
COLOR_1: Defines a fixed point value used by the line draw, expand, and shaded fill operations. When used with byte planes, the 32 bit value is interpretted as an 8.24 fixed-point number. When used with 16-bit pixels, the 32 bit value is interpretted as 3 5.5 fixed-point numbers.

Name: **BLT_COLOR_V_DELT**

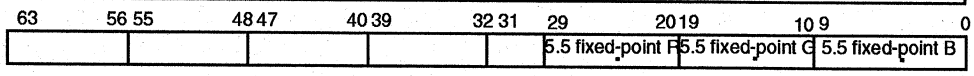
Address: **0x**

Access:

Format:



Format:



Purpose: BLT_COLOR_V_DELT contains an 8.24 color value for use by the shaded fill operation.

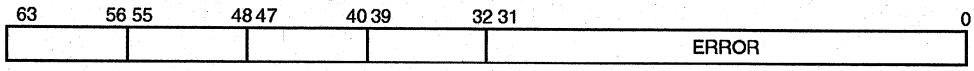
Fields: COLOR_V_DELT: Defines an 8.24 fixed point value used by the and shaded fill operations.

Name: **BLT_ERROR_x**

Address: **0x**

Access:

Format:



Purpose: BLT_ERROR _x, 0 <= x <= 2, contains an integer value for use by line draw, expand, shaded fill, and texture map operations. It is loaded with the Breshenham error term before line drawing or fills are started.

Fields:

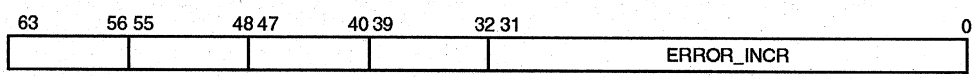
ERROR: Defines a two's complement integer which is used as the initial error during the execution of Breshenham's line drawing algorithm.

Name: **BLT_ERROR_INCR_x_y**

Address: **0x**

Access:

Format:



Purpose: BLT_ERROR_INCR_x_y, 0 <= x <= 2, 0 <= y <= 1, contains an integer value for use by line draw, expand, shaded fill, and texture map operations. The x index refers to one of threee Breshenham loops. The y index refers to the two increment values within each of the loops. Increment register 0 is the independent value; increment register 1 is the dependent value.

Fields:

ERROR_INCR_1: Defines a two's complement integer which is used as an error increment during blitter operations. Its use is a function of the blitter mode.

LINE_DRAW: ERROR_INCR_1 is added after each pixel calculation (Dindep).

SHADED_FILL: ERROR_INCR_1 is added to the left start position after each shaded line completion (DLx/Dy).

ERROR_INCR_2: Defines a two's complement integer which is used as an error increment during blitter operations. Its use is a function of the blitter mode:

LINE_DRAW: ERROR_INCR_2 is added when the error calculation crosses the zero threshold (Ddep).

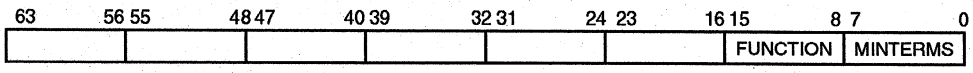
SHADED_FILL: ERROR_INCR_2 is added to the right end position after each shaded line completion (DRx/Dy).

Name: **BLT_FUNC**

Address: **0x**

Access:

Format:



Purpose: BLT_FUNC specifies the function that the blitter is to perform and the minterms that should be used during the operation. Writing to the BLT_FUNC register starts the blitter operation.

Fields:

FUNCTION: Defines the function that the blitter will perform

Code	Operation
0x1	0 source blit
0x2	1 source blit
0x3	2 source blit
0x4	Clipped line draw
0x5	Unclipped line draw
0x6	Dual Breshenham fill
0x7	Texture Map
0x8	Expand

MINTERMS: Defines the logical operation performed.

T: Consider pixel values of zero as transparent. Only applicable on rectangular blits, this bit allows insertion of graphics into canvas without the use of a mask.

S: Pixel SIze: 0 states that 8-bit pixels are being manipulated or expanded to. 1 states that 16-bit pixels are being manipulated to expanded to.

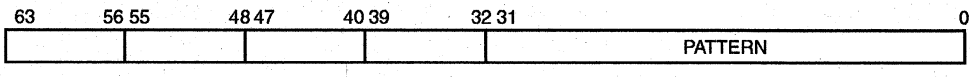
I: Interrupt on completion.

Name: **BLT_PATTERN**

Address:

Access:

Format:



Purpose: BLT_PATTERN contains a bit pattern used during line draw operations.

Fields:

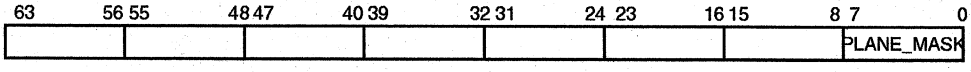
PATTERN: Defines a pattern used during line draw operations. Pixels drawn when the pattern bit is 0 use the value in BLT_COLOR_0. Pixels drawn when the pattern bit is 1 use the value in BLT_COLOR_1.

Name: **BLT_PLANE_MASK**

Address: **0x**

Access:

Format:



Purpose: BLT_PLANE_MASK contains a bit pattern indicating bits to be modified during blitter operation.

Fields:

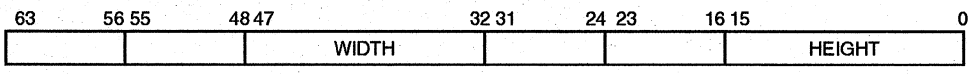
PLANE_MASK: Defines a pattern used to influence bit modifications within pixels during blit operations. Pixels written during blitter operations will be modified only in the bit positions which have a "1" in the plane mask. Those bits where "0" appears in the plane mask will be unchanged in the destination pixel. To write all bits of the pixel, a value of 0xff should be placed in the BLT_PLANE_MASK.

Name: **BLT_SIZE**

Address: **0x**

Access:

Format:



Purpose: BLT_SIZE specifies the size of the rectangle to be operated on by the blitter.

Fields:

WIDTH: Specifies the width in pixels.

HEIGHT: Specifies the height in pixels.

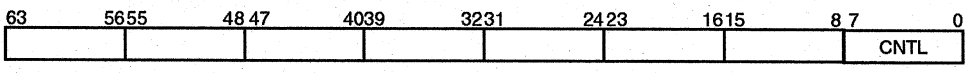
Appendix A CD-ROM Registers

Name: **CD_CNTL**

Address: **0x**

Access:

Format:



Purpose: CD_CNTL controls the operation of the CD-ROM interface.

Fields: CNTL: Controls the features of the CD-ROM interface.

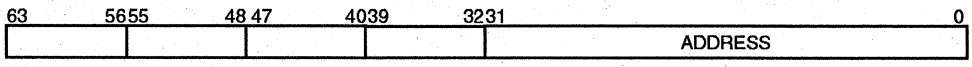
Bit	Description
0	Raw Mode. When set, received data is considered raw audio data
1	When clear, the CD-ROM decoder is disabled and reset. The incrementing buffer index is reset to 0. When set, the CD-ROM decoder and DMA are enabled. If INDX == CMP, and OVERFLOW occurs, no DMA occurs. Otherwise, data is transferred into the buffer referenced by INDX. When a sector has been received and placed into the buffer, INDX is incremented and the staus bit set and an interrupt generated if enabled. If INDX == CMP after being incremented, the OVERFLOW bit is set.
2	Enable DMA of subcode data into memory. When set, the subcode decoder and DMA are enabled. When clear, subcode data is ignored.
3	Enable receiving drive status/response.
4	Enable transmitting drive command data.

Name: **CD_DATA_ADDR**

Address: **0x**

Access:

Format:



Purpose: CD_DATA_ADDR controls the location where CD data buffers will be located.

Fields:

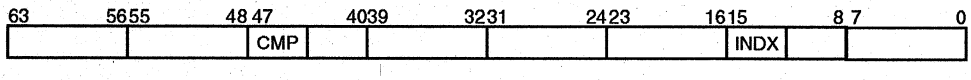
ADDRESS:	Defines the address where a block of 16 4k byte CD-ROM data buffers will be located. The address must be aligned to a 64k byte boundary.

Name: **CD_DATA_BUF_CNTL**

Address: **0x**

Access: **Fullword, RW**

Format:



Purpose: CD_DATA_BUF_CNTL manages the CD data buffers.

Fields:

INDX: Writing this register clears it to zero. This register is incremented at the end of each sector. It is used as an offset to the CD_DATA_ADDR to determine which buffer within the block will be used to receive data from the CD data interface.

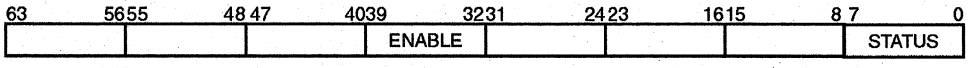
CMP: Contains the number of the buffer within the block which will cause an interrupt to be generated. When INDX is equal to CMP, an interrupt is generated.

Name: **CD_INTERRUPT**

Address: **0x**

Access:

Format:



Purpose: CD_CNTL controls the generation of interrupts from the CD interface.

Fields:

ENABLE: When set, permits an interrupt to be generated when the specified condition is met. When clear, inhibits an interrupt from being set.

STATUS: Indicates the source of CD interrupts.

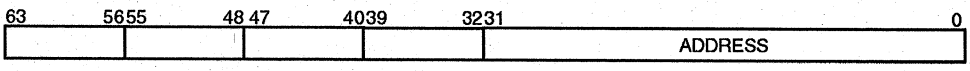
Bit	Use
0	Out of data buffers: cleared by writing CD_DATA_BUF_CNTL.CMP
1	Data sector available: cleared by reading CD_DATA_BUF_CNTL.INDX
2	Subcode frame available: cleared by reading CD_SUBCODE_-BUF_CNTL.INDX
3	Command buffer comparison made: cleared by writing CD_TX_CMD_-BUF_CNTL.CMP
4	Status/Response buffer comparison make: cleared by writing CD_RX_CM-D_BUF_CNTL.CMP

Name: **CD_RX_CMD_ADDR**

Address: **0x**

Access:

Format:



Purpose: CD_RX_CMD_ADDR controls the location where the CD drive status/response information is stored.

Fields:

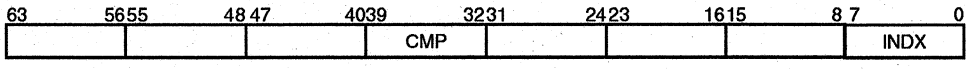
ADDRESS:	Defines the address where a 256 byte circular buffer for CD drive status/response information is located. The address must be aligned to a 256 byte boundary.

Name: **CD_RX_CMD_BUF_CNTL**

Address: **0x**

Access: **Fullword, RW**

Format:



Purpose: CD_RX_CMD_BUF_CNTL manages the CD drive status/response circular buffer..

Fields:

INDX: Writing this register clears it to zero. This register is incremented upon the receipt of each byte of status/response data from the CD drive..

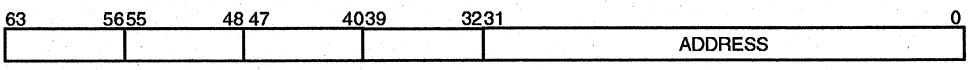
CMP: Contains the offset into the circular buffer that will cause an interrupt. When INDX is equal to CMP, an interrupt is generated.

Name: **CD_SUBCODE_ADDR**

Address: **0x**

Access:

Format:



Purpose: CD_SUBCODE_ADDR controls the location where the CD subcode information is stored.

Fields:

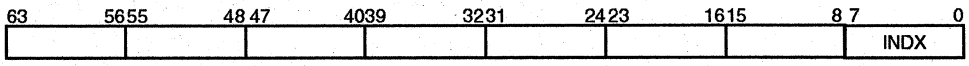
ADDRESS:	Defines the address where a 256 byte circular buffer for CD subcode information is located. The address must be aligned to a 256 byte boundary.

Name: **CD_SUBCODE_BUF_CNTL**

Address: **0x**

Access: **Fullword, RW**

Format:



Purpose: CD_SUBCODE_BUF_CNTL manages the CD subcode circular buffer..

Fields:

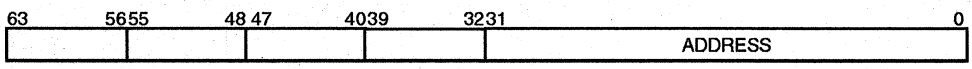
INDX: Writing this register clears it to zero. This register is incremented upon the receipt of each byte of subcode data from the CD drive. When 98 bytes have been received, an interrupt is generated and bit 7 of INDX is inverted. Bits 6:0 are cleared.

Name: **CD_TX_CMD_ADDR**

Address: **0x**

Access:

Format:



Purpose: CD_TX_CMD_ADDR controls the location where the CD drive command information is stored.

Fields:

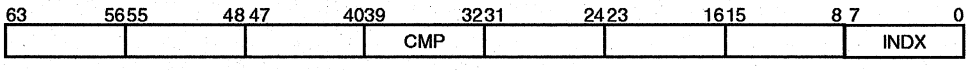
ADDRESS:	Defines the address where a 256 byte circular buffer for CD drive command information is located. The address must be aligned to a 256 byte boundary.

Name: **CD_TX_CMD_BUF_CNTL**

Address: **0x**

Access: **Fullword, RW**

Format:



Purpose: CD_TX_CMD_BUF_CNTL manages the CD drive command circular buffer..

Fields:

INDX: Writing this register clears it to zero. This register is incremented after sending each byte of command data to the CD drive..

CMP: Contains the offset into the circular buffer that will cause an interrupt. When INDX is equal to CMP, an interrupt is generated.