Pixel
Semiconductor
A Cirrus Logic Company

CL-PX208X

Technical Reference Manual

FEATURES

- **VESA® Advanced Feature Connector (VAFC™) compatible**
- **Mixes graphics and video in realtime**
- **Supports multiple, occluded video windows**
- **Pixel-clock rates up to 135 MHz**
  - — Software-selectable clock doubler
  - — 1024 x 768 at 2^{24} colors, 76-Hz refresh rate
  - — 1280 x 1024 at 64K colors, 76-Hz refresh rate
- **Pixel-clock rates up to 85 MHz (CL-PX2080)**
- **Extensive software support available**
  - — Contact Cirrus Logic sales office for complete details
- **Video input formats**
  - — RGB — 8:8:8, (1)5:5:5, (T)5:6:5
  - — YCrCb — tagged or untagged 4:2:2
  - — Automatic sync polarity detection
- **Graphics input formats**
  - — Pseudocolor — 4-bit or 8-bit
  - — RGB — 5:6:5, 5:5:5, 8:8:8
- **Display functions**
  - — Pseudocolor display
  - — True-color RGB display
  - — Interpolated, continuously variable zoom
  - — Hardware cursor controls
  - — Graphics overlay controls: tagged chroma color key, graphics overlay color key, x/y window
- **Local bus interface**
- **Direct ISA/MicroChannel® Bus (MCB) interface**

MediaDAC™

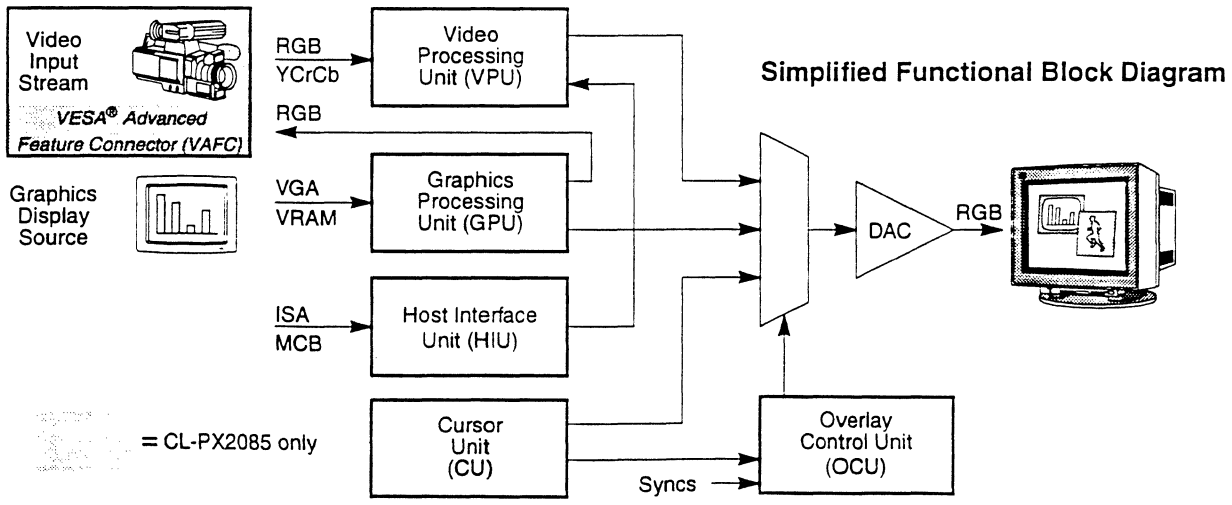
- **Interlaced or non-interlaced output**
- **Hardware cursor**
  - — 32 x 32 pixel
  - — 64 x 64 pixel
- **Flexible register access modes**

APPLICATIONS

- **Presentations**
- **Video Editing**
- **Video Authoring**
- **Video Teleconferencing**
- **Interactive Education**
- **Games**

OVERVIEW

The CL-PX208X MediaDAC™ accepts both video and graphics data. It converts the video data stream to the output format, color space, and color resolution of the graphics subsystem, then mixes and/or overlays it with processed graphics and cursor data.



Simplified Functional Block Diagram

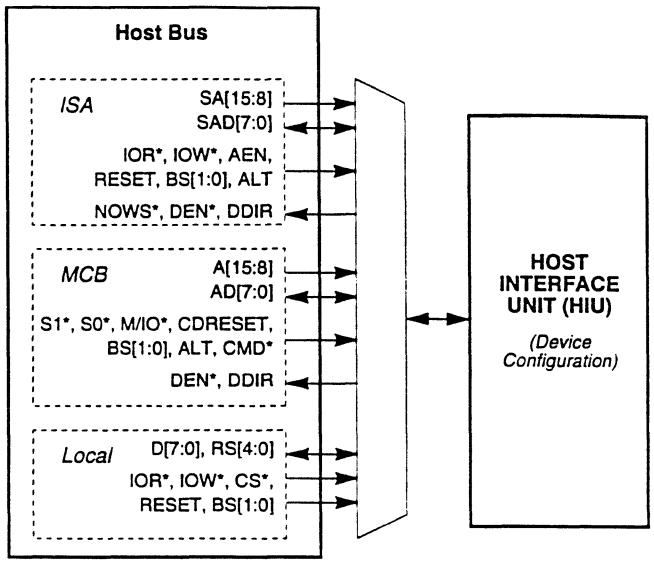
OVERVIEW (cont.)

The MediaDAC contains six functional blocks:

- Host Interface Unit (HIU)
- Video Processing Unit (VPU)
- Graphics Processing Unit (GPU)
- Cursor Unit (CU)
- Overlay Control Unit (OCU)
- Monitor Interface Unit (MIU).

Host Interface Unit (HIU)

The HIU contains the bus interface and the configuration, control, and status registers. It connects the MediaDAC directly to ISA and MCB buses, internally decoding a 16-bit address and responding as an 8-bit peripheral. (This interface eliminates most of the costly 'glue' circuitry common to PC expansion boards.) The HIU also interfaces with local hardware.



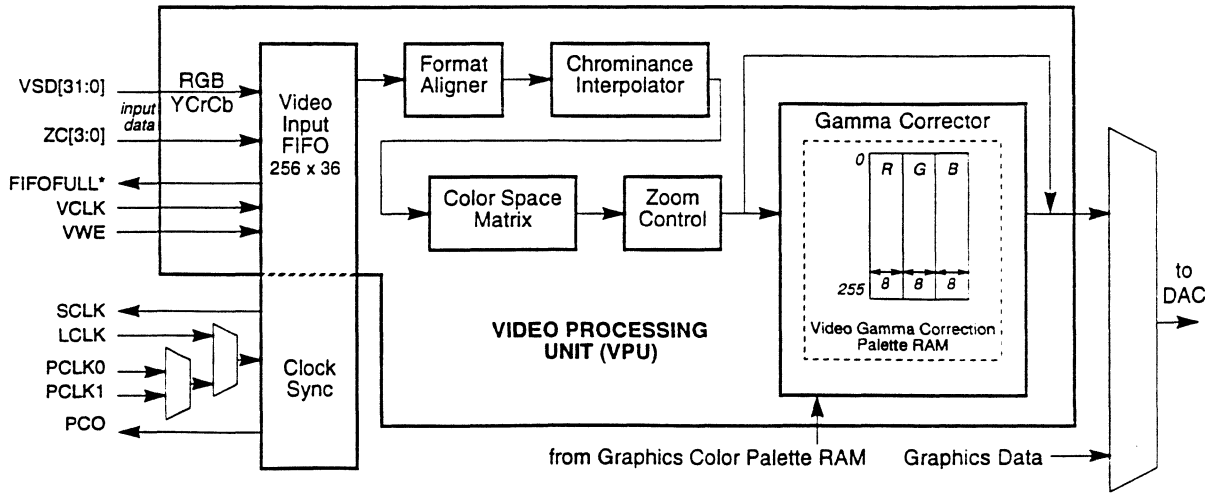
Host Bus Interface

Video Processing Unit (VPU)

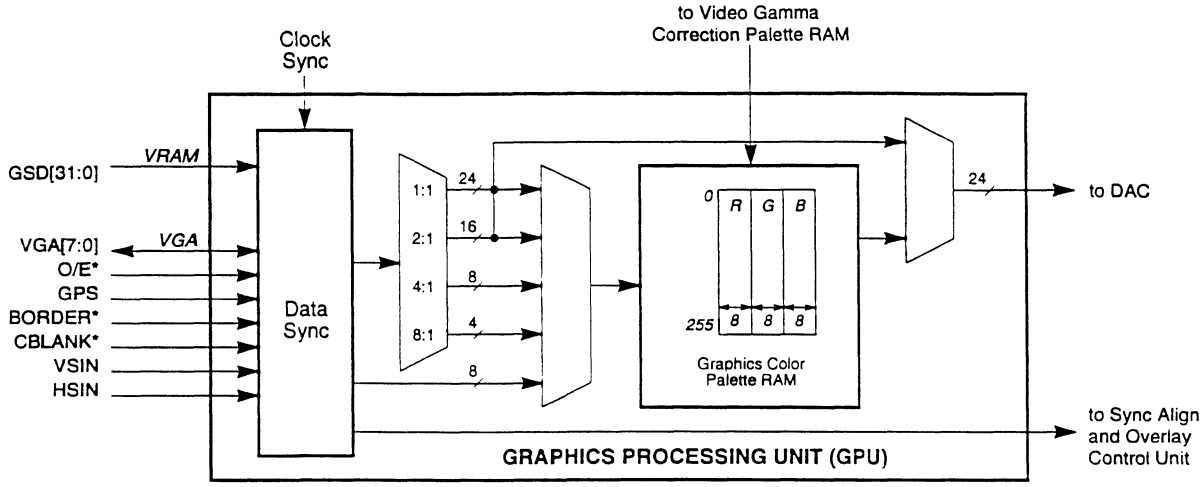
The VPU accepts digitized RBG and YCrCb video data in a wide range of formats. Its video processing functions are illustrated in the figure below.

VPU features:

- 32-bit input data path
- Continuously variable zoom (to 256x)
- Format alignment
- Chrominance interpolation
- YCrCb-to-RGB color-space conversion
- Internal 256 x 36-bit input FIFO that supports:
  - — 24-bit RGB data (up to 40 million pixels per second)
  - — 16-bit RGB or YCrCb data
- Chrominance interpolation
- Programmable gamma-correction lookup table.



Video Processing Functions



Graphics Processing Functions

Graphics Processing Unit (GPU)

The GPU accepts graphics data through either of two paths — VGA or VRAM — as shown above. As a result, PC graphics subsystems based on the MediaDAC can maintain compatibility with both types of systems.

GPU features:

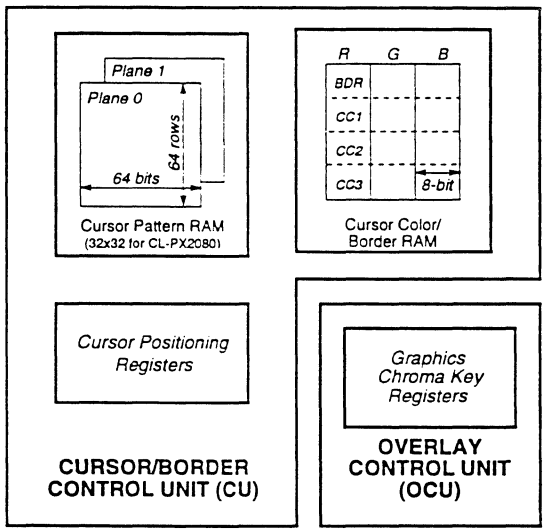
- VGA interface:
  - — 8-bit VGA data port
  - — Supports the large, existing installed base of systems and VGA-specific software.
- VRAM interface:
  - — 32-bit high-resolution VRAM serial data port; supports a variety of architectures
  - — Supports next generation of higher-performance and higher-resolution products
  - — Efficient pixel mapping within graphics-data words
- True-color (CLUT bypass) option.

Overlay Control Unit (OCU)

The OCU contains the Graphics Chroma Key registers. Its variety of operations allows the combination of video and graphics images.

Every graphics pixel is either opaque (its color information is displayed on the screen) or transparent (its color information is not displayed, allowing the video pixel behind it to be displayed instead). The OCU determines which graphics pixels are transparent.

The MediaDAC has 256 possible overlay combinations based on the video-pixel tag bit, the graphics-pixel overlay color, and the XY window of the video data.



Cursor/Overlay Control Functions

Cursor Unit (CU)

The MediaDAC implements an on-chip, user-definable, three-color hardware cursor that works in both interlaced and non-interlaced systems. It can be defined as either 32 x 32 x 2 (default on power-up and reset) or 64 x 64 x 2 in the CL-PX2085.

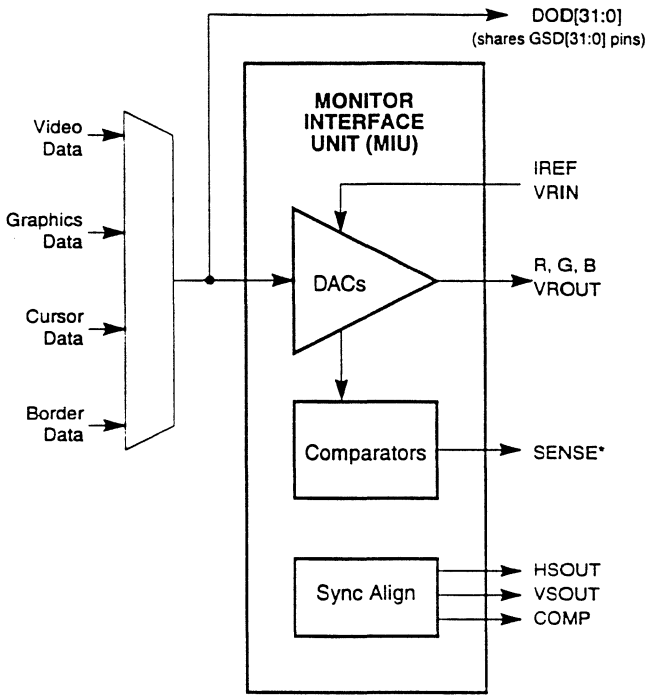
The CU controls the cursor color, pattern, and position.

Monitor Interface Unit (MIU)

The MIU contains three subunits, as illustrated:

- Three video-speed, 8-bit digital-to-analog converters
- Internal comparators to provide the sense function
- Sync alignment logic.

During the MediaDAC's power-down condition, the DACs are turned off and the RAM enters a low-power, data-retaining standby mode.



RGB Monitor Interface

TABLE OF CONTENTS

PART III: APPLICATION NOTES

PART IV: REFERENCE

LIST OF FIGURES

LIST OF TABLES

CONVENTIONS, ABBREVIATIONS, AND TRADEMARKS

CONVENTIONS

GCKc — Register names that contain lowercase variables represent groups of registers with similar functions. For example, GCKc represents *all* of the Graphics Chroma Key registers: GCKR (Red), GCKG (Green), and GCKB (Blue). In this data book, the following register variables are used:

b (byte) = L (Low) or H (High)
c (color space) = Y, U, V, R, G, or B

SAR[VL] — Register SAR, bit VL.

(shaded) — Shaded areas apply *only* to the CL-PX2085, not the CL-PX2080.

ABBREVIATIONS, ACRONYMS, and MNEMONICS

CLUT	Color LookUp Table	N/C	No Connect
CMOS	Complementary Metal Oxide Silicon	OCU	Overlay Control Unit
CPU	Central Processing Unit	OD	Open Drain
CRT	Cathode Ray Tube	PQFP	Plastic Quad Flat Pack
CU	Cursor Unit	PWR	PoWeR
DAC	Digital-to-Analog Converter	RGB	Red, Green, Blue
FIFO	First In, First Out	RAM	Random Access Memory
GPU	Graphics Processing Unit	TBD	To Be Determined
HIU	Host Interface Unit	TS	Tristate
ISA	Industry Standard Architecture	TTL	Transistor/Transistor Logic
LSB	Least-Significant Byte	VAFC™	VESA Advanced Feature Connector
LSb	Least-Significant bit	VESA®	Video Electronics Standards Association
LUT	LookUp Table	VGA	Video Graphics Architecture
MCB	MicroChannel® Bus	VRAM	Video dynamic Random Access Memory
MIU	Monitor Interface Unit	VSVI	VESA Synchronous Video Interface
MSB	Most-Significant Byte	YCrCb	Components of the CCIR601 color representation standard. Y = luminance; CrCb = chrominance Y-blue, chrominance Y-red
MSb	Most-Significant bit	VPU	Video Processing Unit
MUX	MUltipleX	YUV	Y = luminance; UV = chrominance

TRADEMARKS

MediaDAC™ is a trademark of Pixel Semiconductor, Inc.

HiCOLOR™ is a trademark of Sierra Semiconductor, Inc.

MicroChannel® is a registered trademark of IBM® Corporation.

VAFC™ is a registered trademark of Pixel Semiconductor, Inc.

VESA® is a registered trademark of the Video Electronics Standards Association.

PART I: PRODUCT DATA

1. PIN INFORMATION

The CL-PX208X (CL-PX2080 or CL-PX2085) MediaDAC™ is available in a 160-lead Plastic Quad Flat Pack (PQFP) surface-mount package. It can be configured for ISA, MCB (MicroChannel® Bus), and local hardware configurations, as shown in Figure 1-1.

1.1 Pin Diagram

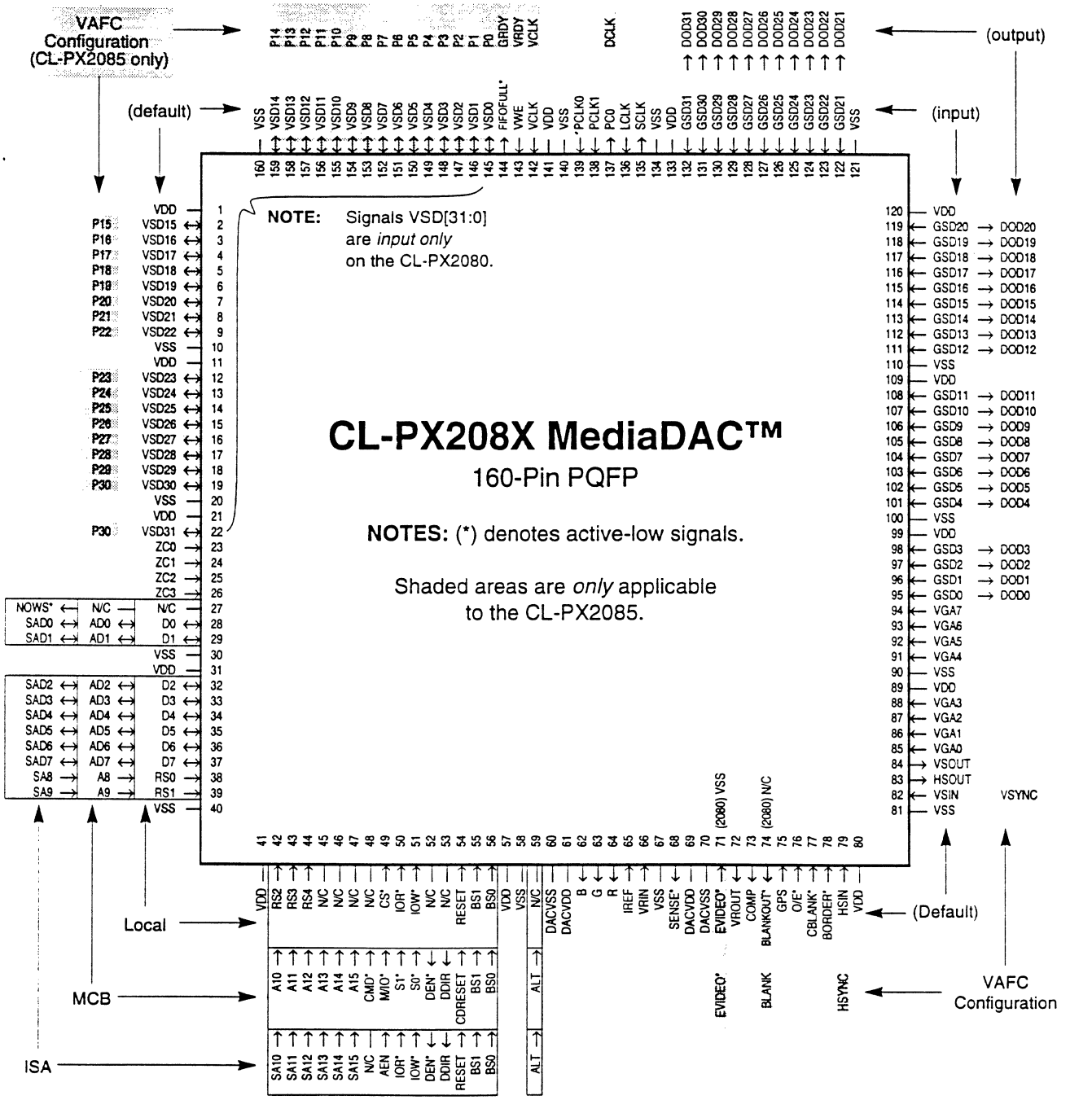


Figure 1-1. CL-PX208X MediaDAC™ Pin Diagram

1.2 Functional Signal Groups

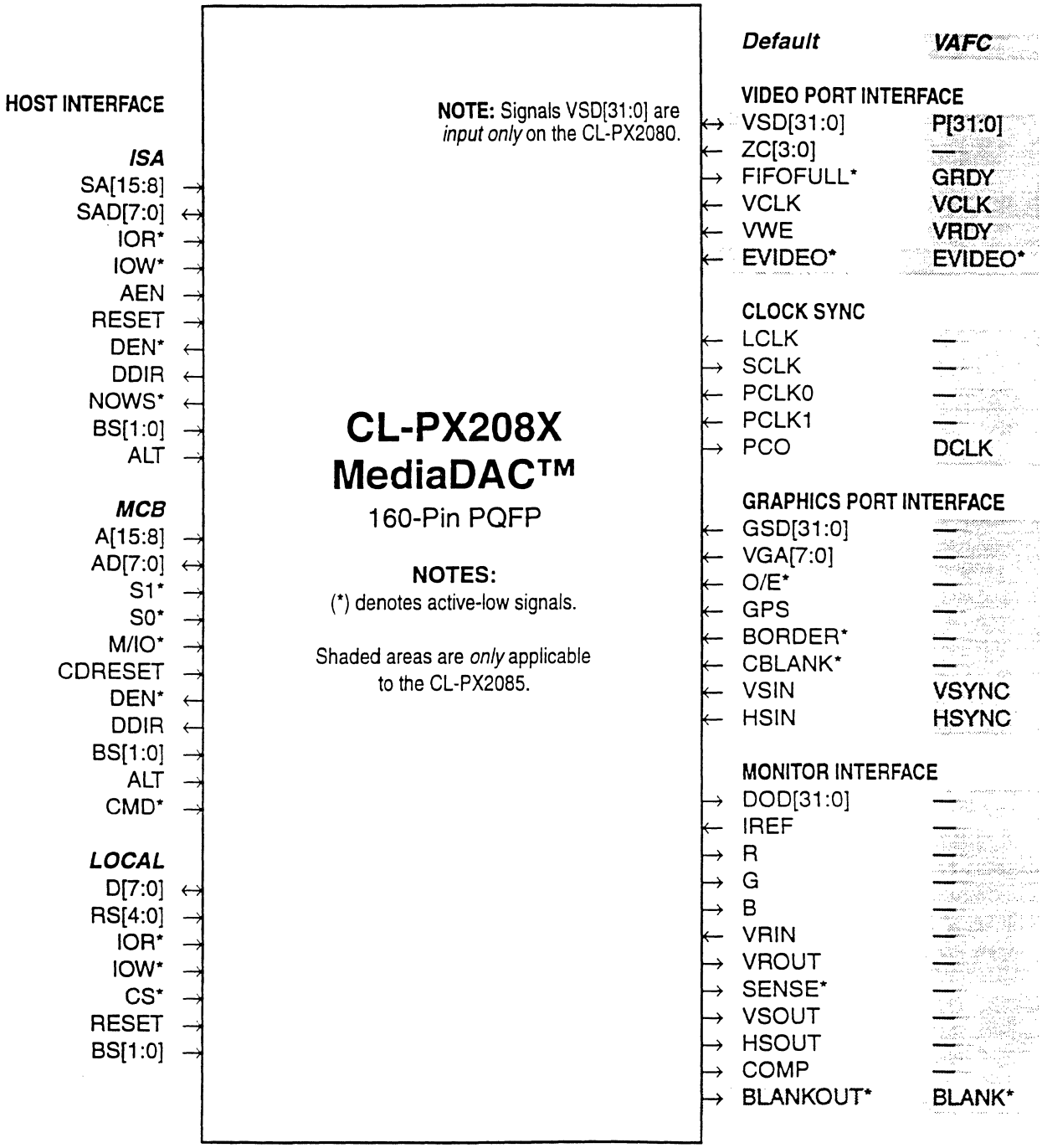


Figure 1-2. MediaDAC™ Functional Signal Groups

1.3 Pin Assignment Table

The following conventions are used in the pin assignment table:

(*)	= active-low signal	PD	= internal pull down
I	= input	TTL	= standard TTL input threshold and TTL output levels
O	= output	TS	= tristate TTL drive capability
AN	= analog signal	OD	= open drain, TTL inputs
PU	= internal pull-up	8	= 8-mA sink and 3-mA source drive capability
N/C	= no connect	12	= 12-mA sink and 4-mA source drive capability
PWR	= power	24	= 24-mA sink and 8-mA source drive capability

NOTE: Shaded areas apply to the CL-PX2085 only.

SIGNAL NAME			PIN	TYPE	CELL	FUNCTION
HOST INTERFACE						
ISA *(BS[1:0]=00)*	***MCB*** *(BS[1:0]=01)*	***LOCAL*** *(BS[1:0]=10,11)*				
SA[15:8]	A[15:8]	–	47:42, 39:38	I	PD, TTL	Address Bus — High Byte
SAD[7:0]	AD[7:0]	–	37:32, 29:28	I/O	PD, 3S, 8	Address/Data Bus — Low Byte
–	–	D[7:0]	37:32, 29:28	I/O	3S, 8	Data Bus
–	–	RS[4:0]	44:42, 39:38	I	PD, TTL	Register Select
AEN	–	–	49	I	TTL	Address Enable
–	M/IO*	–	49	I	TTL	Memory or I/O Cycle
–	–	CS*	49	I	TTL	Chip Select
IOR*	–	IOR*	50	I	TTL	I/O Read
–	S1*	–	50	I	TTL	Status 1
IOW*		IOW*	51	I	TTL	I/O Write
–	S0*	–	51	I	TTL	Status 0
RESET	CDRESET	RESET	54	I	TTL	Reset
DEN*	DEN*	–	52	O	OD, TTL, 8	Data Buffer Enable
DDIR	DDIR	–	53	O	OD, TTL, 8	Data Buffer Direction
NOWS*	–	–	27	O	OD, TTL, 24	No Wait State
BS[1:0]	BS[1:0]	BS[1:0]	55:56	I	TTL	Bus Select
ALT	ALT	–	59	I	TTL	BIR Alternate Address
–	CMD*	–	48	I	TTL	Command

NAME	PIN	TYPE	FUNCTION
POWER			
VDD	1, 11, 21, 31, 41, 57, 80, 89, 99, 109, 120, 133, 141	PWR	+5 VDC for Digital Logic and Interface Buffers
VSS	10, 20, 30, 40, 58, 67, 71 (CL-PX2080 only) 81, 90, 100, 110, 121, 134, 140, 160	PWR	Ground for Digital Logic and Interface Buffers
DACVDD	61, 69	PWR	+5 VDC for DAC
DACVSS	60, 70	PWR	Ground for DAC

SIGNAL NAME		PIN	TYPE	CELL	FUNCTION	
Default	***VAFC***				***Default***	**VAFC**
VAFC VESA Advanced Feature Connector[V]=1*	*VAFC VESA Advanced Feature Connector[V*]=0*				*VAFC VESA Advanced Feature Connector[V*]=1*	*VAFC VESA Advanced Feature Connector[V*]=0*
VIDEO PORT INTERFACE						
VSD[31:0] [a]	P[31:0]	22, 19:12, 9:2, 159:145	I/O (2085) I (2080)	PU, TTL, 8	Video Source Data	VAFC Port
ZC[3:0]	—	26:23	I	TTL	Zoom Control Code	—
FIFOFULL*	GRDY	144	O	TTL, 8	FIFO Full	VAFC Graphics Ready
VCLK	VCLK	142	I	TTL	Video Data Clock	VAFC Video Clock
VWE	VRDY	143	I	TTL	Video FIFO Write Enable	VAFC Video Ready
EVIDEO*	EVIDEO*	71	I	TTL	Enable Video	VAFC Enable Video
CLOCK SYNC						
LCLK	—	136	I	TTL	Latch Clock Input	—
SCLK	—	135	O	TTL, 12	VRAM Shift Clock Output	—
PCLK0	—	139	I	TTL	Pixel Input Clock 0	—
PCLK1	—	138	I	TTL	Pixel Input Clock 1	—
PCO	DCLK	137	O	TTL, 12	Pixel Clock Output	VAFC Graphics Clock
GRAPHICS PORT INTERFACE						
GSD[31:0][b]	—	132:122, 119:111, 108:101, 98:95	I	PU, 3S, 8	Graphics Source Data (VRAM)	—
VGA[7:0]	—	94:91, 88:85	I	PU, TTL	VGA Graphics Source Data	—
O/E*	—	76	I	TTL	Odd/Even Field Input	—
GPS	—	75	I	TTL	Graphics Port Select	—
BORDER*	—	78	I	TTL	Active Display Border	—
CBLANK*	—	77	I	TTL	Composite Blank Input	—
VSIN	VSYNC	82	I	TTL	Vertical Sync Input	VAFC Vertical Sync
HSIN	HSYNC	79	I	TTL	Horizontal Sync Input	VAFC Horizontal Sync
MONITOR INTERFACE						
DOD[31:0][c]	—	132:122,119:111, 108:101, 98:95	O	PU, 3S, 8	Display Output Data	—
IREF	—	65	I	AN	Current Reference	—
R	—	64	O	AN	Analog Red	—
G	—	63	O	AN	Analog Green	—
B	—	62	O	AN	Analog Blue	—
VRIN	—	66	I	AN	Voltage Reference In	—
VROUT	—	72	O	AN	Voltage Reference Out	—
SENSE*	—	68	O	TTL, 8	Monitor Sense	—
VSOUT	—	84	O	TTL, 8	Vertical Sync Output	—
HSOUT	—	83	O	TTL, 8	Horizontal Sync Output	—
COMP	—	73	O	AN	Compensation	—
BLANKOUT*	BLANK*	74	O	TTL, 8	Blank Out	VAFC Blank

a. Signals VSD[31:0] are input/output on the CL-PX2085, and input only on the CL-PX2080.
b. Input signals GSD[31:0] share pins with output signals DOD[31:0]; enabled when register CSC[DPS] = 01 or 10.
c. Output signals DOD[31:0] share pins with input signals GSD[31:0; enabled when register CSC[DPS] = 11.

2. DETAILED SIGNAL DESCRIPTIONS

2.1 Host Interface

2.1.1 ISA Configuration

NOTE: ISA signals are enabled when signals BS[1:0]=00.

Signal	Pin	Type	Cell	Function
SA[15:8]	47:42, 39:38	I	TTL	**Address Bus — High Byte.** Specifies the resource to be accessed during an I/O or memory cycle.
SAD[7:0]	37:32, 29:28	I/O	3S, 8	**Address/Data Bus — Low Byte.** Bidirectional, multiplexed address/data bus that transfers video data and operation status and commands between the host system and the MediaDAC.
AEN	49	I	TTL	**Address Enable.** 0 I/O cycle in progress 1 DMA cycle in progress
IOR*	50	I	TTL	**I/O Read.** 0 Specifies an I/O read cycle
IOW*	51	I	TTL	**I/O Write.** 0 Specifies an I/O write cycle
RESET	54	I	TTL	**Reset.** 1 Stops all MediaDAC activity and resets the hardware
DEN*	52	O	OD, TTL, 8	**Data Buffer Enable.** 0 Enables the host data bus buffer
DDIR	53	O	OD, TTL, 8	**Data Buffer Direction.** Specifies the direction of data flow on SAD[7:0]. 0 Host is reading data from SAD[7:0] 1 Host is writing data to SAD[7:0]
NOWS*	27	O	OD, TTL, 24	**No Wait State.** Instructs the host system to run a zero-wait-state cycle. The default ISA bus cycle is one wait state.
BS[1:0]	55:56	I	TTL	**Bus Select.** Specifies MediaDAC bus mode. 00 ISA 01 MCB 10 Local Hardware Mode A (RS[4:0] address registers) 11 Local Hardware Mode B (RS[3:0] address registers)
ALT	59	I	TTL	**BIR Alternate Address.** Selects ISA address range for BIR access. 0 Primary ISA address range 1 Secondary ISA address range

2.1.2 MCB Configuration

NOTE: MCB signals are enabled when signals BS[1:0]=01.

Signal	Pin	Type	Cell	Function
A[15:8]	47:42, 39:38	I	TTL	**Address Bus — High Byte.** Specifies the resource to be accessed during an I/O or memory cycle.
AD[7:0]	37:32, 29:28	I/O	3S, 8	**Address/Data Bus — Low Byte.** Bidirectional, multiplexed data bus that transfers video data and operation status and commands between host system and the MediaDAC.
M/IO*	49	I	TTL	**Memory or I/O Cycle.** M/IO*, S1*, and S0* specify the current bus cycle:
S1*	50	I	TTL	**Status 1.**
S0*	51	I	TTL	**Status 0.**
CDRESET	54	I	TTL	**Reset.** 1 Stops all MediaDAC activity and resets the hardware
DEN*	52	O	OD, TTL, 8	**Data Buffer Enable.** 0 Enables host data bus buffer
DDIR	53	O	OD, TTL, 8	**Data Buffer Direction.** Specifies the direction of data flow on AD[7:0]. 0 Host is reading data from AD[7:0] 1 Host is writing data to AD[7:0]
BS[1:0]	55:56	I	TTL	**Bus Select.** Specifies MediaDAC bus mode. 00 ISA 01 MCB 10 Local Hardware Mode A (RS[4:0] address registers) 11 Local Hardware Mode B (RS[3:0] address registers)
ALT	59	I	TTL	**BIR Alternate Address.** Selects ISA address range for BIR access. 0 Primary ISA address range 1 Secondary ISA address range
CMD*	48	I	TTL	**Command.** 0 (write cycle) valid data is on bus AD[7:0] 1c (read cycle) MediaDAC should place valid data on bus AD[7:0]

*M/IO**	*S1**	*S0**	*Current Bus Cycle*
0	0	0	Reserved
0	1	0	I/O Write
0	0	1	I/O Read
0	1	1	Inactive
1	0	0	Reserved
1	1	0	Memory Write
1	0	1	Memory Read
1	1	1	Inactive

2.1.3 Local Hardware Configuration

NOTE: Local hardware signals are enabled when signals BS[1:0]=10 (Mode A) or BS[1:0]=11 (Mode B).

Signal	Pin	Type	Cell	Function
D[7:0]	37:32, 29:28	I/O	3S, 8	**Data Bus.** Bidirectional data bus that accesses the internal control registers.
RS[4:0]	44:42, 39:38	I	PD, TTL	**Register Select.** Specifies the internal register to be accessed during a MediaDAC I/O cycle. **NOTE:** In Local Hardware Mode B, only RS[3:0] are used.
CS*	49	I	TTL	**Chip Select.** 0 Host is accessing the MediaDAC
IOR*	50	I	TTL	**I/O Read.** 0 Specifies an I/O read cycle
IOW*	51	I	TTL	**I/O Write.** 0 Specifies an I/O write cycle
RESET	54	I	TTL	**Reset.** 1 Stops all MediaDAC activity and resets the hardware
BS[1:0]	55:56	I	TTL	**Bus Select.** Specifies MediaDAC bus mode. 00 ISA 01 MCB 10 Local Hardware Mode A (RS[4:0] address registers) 11 Local Hardware Mode B (RS[3:0] address registers).

2.2 Video Port Interface

NOTE: VAFC signals (shown in italics) are enabled when register VAFC[V*]=0.

Signal	Pin	Type	Cell	Function
VSD[31:0]	22, 19:12, 9:2, 159:145	I/O (2085) I (2080)	PU, TTL, 8	**Video Source Data.** (input mode, see definition of EVIDEO* on page 23) Port through which video data enters the MediaDAC. The MediaDAC supports the following tagged and untagged data formats: 16-bit (4:2:2) YCrCb; 16-bit (5:6:5) and 24-bit (8:8:8) RGB. VCLK transfers 16-bit modes as two pixels per pixel word.
P[31:0]				**VAFC Port.** (output [power-up] mode) Port over which the MediaDAC outputs VGA[7:0]/GSD[31:0] bus data. The I/O condition of this port is defined by EVIDEO*. Powers up in 8-bit mode.
ZC[3:0]	26:23	I	TTL	**Zoom Control Code.** (used with the CL-PX2070 DVP) Specifies several options for interpolation and alignment of the input video stream on VSD[31:0].
FIFOFULL*	144	O	TTL, 8	**FIFO Full.** 0 Notifies the external video source that the 256-deep, double-pixel FIFO is within 8 pixels of a full condition
GRDY				**VAFC Graphics Ready.** 1 The graphics card is ready to latch data on bus

2.2 Video Port Interface *(cont.)*

NOTE: VAFC signals (shown in italics) are enabled when register VAFC[V*]=0.

Signal	Pin	Type	Cell	Function
VCLK	142	I	TTL	**Video Data Clock.** On rising edge, clocks VSD[31:0] and ZC[3:0] data into the MediaDAC input video FIFO; generated by the source video processor when VWE = 1.
VCLK				**VAFC Video Clock.** Continuous master clock; signals from video card into graphics card.
VWE	143	I	TTL	**Video FIFO Write Enable.** 0 Writes are disabled 1 Data is written on the rising edge of VCLK
VRDY				**VAFC Video Ready.** 1 Video card has placed valid data on the bus
EVIDEO*	71	I	TTL	**Enable Video.**
*EVIDEO**				**VAFC Enable Video.** Driven by video card. 0 Drives P[31:0] into graphics card on VSD[31:0] 1 Drives P[31:0] into video card on VSD[31:0]

2.3 Clock, Sync

NOTE: VAFC signals (shown in italics) are enabled when register VAFC[V*]=0.

Signal	Pin	Type	Cell	Function
LCLK	136	I	TTL	**Latch Clock Input.** On rising edge, latches GSD[31:0] or VGA[7:0], and CBLANK*, HSIN, VSIN, GPS, O/E*, and BORDER*. When the input data multiplexing rate is *x*:1, LCLK must equal PCLKn ÷ *x*. (*x*:1 = operating mode; *x* = 8, 4, 2, or 1.) **NOTE:** To avoid metastability, LCLK must maintain setup and hold requirements to SCLK.
SCLK	135	O	TTL, 12	**VRAM Shift Clock Output.** SCLK = PCLKn ÷ *x*. (*x*:1 = operating mode; *x* = 8, 4, 2, or 1.)
PCLKn	139	I	TTL	**PCLK0: Pixel Input Clock 0.** High-speed GSD[31:0] input clock used during multiplexed operation of the 32-bit VRAM serial pixel port; enabled when register CSC[CS] = 0.
	138	I	TTL	**PCLK1: Pixel Input Clock 1.** VGA input clock; enabled when register CSC[CS] = 1.
PCO	137	O	TTL, 12	**Pixel Clock Output.** Buffered output of PCLKn.
DCLK				**VAFC Graphics Clock.** Continuous master clock; signals from graphics card into video card. Typically based on DAC pixel clock.

2.4 Graphics Port Interface

NOTE: VAFC signals (shown in italics) are enabled when register VAFC[V*] = 0.

Signal	Pin	Type	Cell	Function
GSD[31:0]	132:121, 119:111, 108:101, 98:95	I	PU, 3S, 8	**Graphics Source Data (VRAM).** 32-bit VRAM serial data port that accepts data at 4, 8, 16, and 24 bits per pixel. Shares pins with output signals DOD[31:0]. See GPS table below for GSD[31:0] data selection.
VGA[7:0]	94:91, 88:85	I	PU, TTL	**VGA Graphics Source Data.** Latched on the rising edge of LCLK. All unused bits must be connected to VSS. See GPS table, page 24, for VGA[7:0] data selection.
O/E*	76	I	TTL	**Odd/Even Field Input.** Ensures proper operation of the cursor controller in interlaced mode (not used in non-interlaced mode). O/E* should be changed only during vertical blanking.
GPS	75	I	TTL	**Graphics Port Select.** Selects GSD[31:0] or VGA[7:0]. **NOTE:** PORTSEL is an internal signal based on the logical combination of signal GPS and register CSC[DPS] (see page 64), as shown in the following table.
BORDER*	78	I	TTL	**Active Display Border.** Used with CBLANK* and PORTSEL to specify DAC output characteristics (BORDER* = 1 when a display border is not used). PORTSEL is defined in GPS table, above.
CBLANK*	77	I	TTL	**Composite Blank Input.** Applies a color value of '0' to the DAC inputs to produce black at the DAC outputs. The cursor position counters are referenced to BORDER*.
VSIN	82	I	TTL	**Vertical Sync Input.** Generates a vertical sync pulse once per frame in non-interlaced mode, and once per field in interlaced mode.
VSYNC				**VAFC Vertical Sync.** Graphics vertical sync sent to video card as basic reference signal.
HSIN	79	I	TTL	**Horizontal Sync Input.** Generates a horizontal sync pulse every line.
HSYNC				**VAFC Horizontal Sync.** Graphics horizontal sync sent to video card as basic reference signal.

Register CSC[DPS]	*Signal GPS*	*Int. Signal PORTSEL*	*Port Selected*		
			GSD[31:0]	*VGA[7:0]*	*DOD[31:0]*
00	X	0	—	enabled	—
01	0	0	—	enabled	—
	1	1	enabled	—	—
10	X	1	enabled	—	—
11	X	0	—	enabled	enabled

*BORDER**	*CBLANK**	*PORTSEL*	*DAC Output Characteristics*
X	0	X	Blanked area
0	1	0	VGA data
0	1	1	Overscan (border)
1	1	0	VGA data, cursor data
1	1	1	GSD data, cursor data

2.5 Monitor Interface

NOTE: VAFC signals (enabled when register VAFC[V*]=0) are shown in italics.

Signal	Pin	Type	Cell	Function
DOD[31:0]	132:122, 119:111, 108:101, 98:95	O	PU, 3S, 8	**Display Output Data.** (shares pins with input signals GSD[31:0]) Enabled when register CSC[DPS] = 11, and outputs graphics data as follows: • GSD[23:16] = R[7:0] • GSD[15:8] = G[7:0] • GSD[7:0] = B[7:0] • GSD[26] = Window Active • GSD[25] = DODVSOUT • GSD[24] = DODHSOUT DOD[31:0] is latched on the rising edge of LCLK. All unused bits must be connected to VSS. DOD[31:0] will not run at maximum frequency.
IREF	65	I	AN	**Current Reference.** Reference current for the DAC.
R	64	O	AN	**Analog Red.** The analog red channel from the 8-bit DAC.
G	63	O	AN	**Analog Green.** The analog green channel from the 8-bit DAC.
B	62	O	AN	**Analog Blue.** The analog blue channel from the 8-bit DAC.
VRIN	66	I	AN	**Voltage Reference In.** 1.23V reference voltage for the DAC.
VROUT	72	O	AN	**Voltage Reference Out.**
SENSE*	68	O	TTL, 8	**Monitor Sense.** A logical OR of the comparator outputs. Three level-detecting comparators individually monitor the red, green, and blue DAC outputs. A maximum analog DAC output level generates a low-level comparator output. A minimum analog level produces a high-level comparator output.
VSOUT	84	O	TTL, 8	**Vertical Sync Output.** VSOUT (along with data) is delayed 31 PCLKs from VSIN.
HSOUT	83	O	TTL, 8	**Horizontal Sync Output.** HSOUT (or a composite sync generated from HSIN and VSIN, as determined by register SAR) is delayed 21 PCLKns after HSIN. Data is delayed 10 PCLKns after HSOUT (default).
COMP	73	O	AN	**Compensation.** For best image quality, connect COMP to DACVDD (pin 69) with minimum length PC-board wiring through a series 0.1 µf capacitor. This stabilizes an internal reference voltage, minimizes inductive parasitics, and allows maximum benefit of the external capacitor.
BLANKOUT* *BLANK**	74	O	TTL, 8	**Blank Out.** **VAFC Blank.** Only inactive during video window. Graphics data may be in the region where BLANK* is active.

2.6 Power

Signal	Pin	Type	Function
VDD	1, 11, 21, 31, 41, 57, 80, 89, 99, 109, 120, 133, 141	PWR	**+5 VDC for Digital Logic and Interface Buffers.** Each pin must be connected directly to the VDD plane.
VSS	10, 20, 30, 40, 58, 67, 81, 90, 100, 110, 121, 134, 140, 160	PWR	**Ground for Digital Logic and Interface Buffers.** Each pin must be connected directly to the ground plane.
DACVDD	61, 69	PWR	**+5 VDC for DAC.** Refer to Section 10.1 for more information.
DACVSS	60, 70	PWR	**Ground for DAC.** Refer to Section 10.1 for more information.

3. FUNCTIONAL DESCRIPTION

The CL-PX208X MediaDAC™ is a family of high-speed digital-to-analog video converters that can mix separate graphics and video streams that have different color spaces and resolutions. As shown in the simplified functional block diagram in Figure 3-1, the MediaDAC has three input ports:

- A video input port for YCrCb or RGB data, which can function as a VAFC (VESA Advanced Feature Connector) port in the CL-PX2085
- Two graphics input ports for 8-bit VGA or 32-bit high-resolution graphics.

The MediaDAC also includes a hardware cursor and a combination of three graphics overlay controls. Its video-processing functions include:

- Format alignment
- Chrominance interpolation
- Color-space conversion.
- Zoom
- Gamma correction

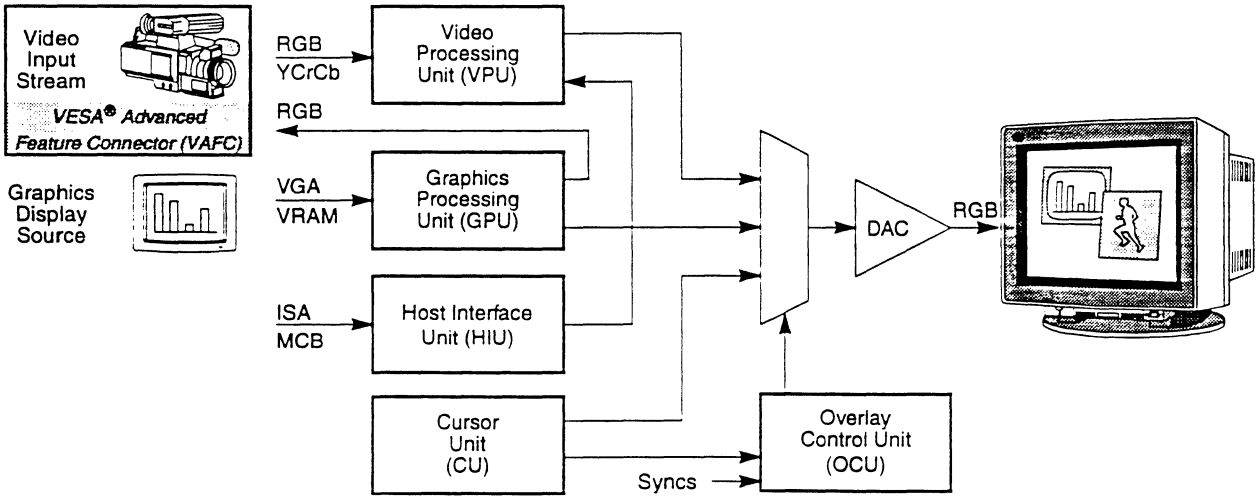


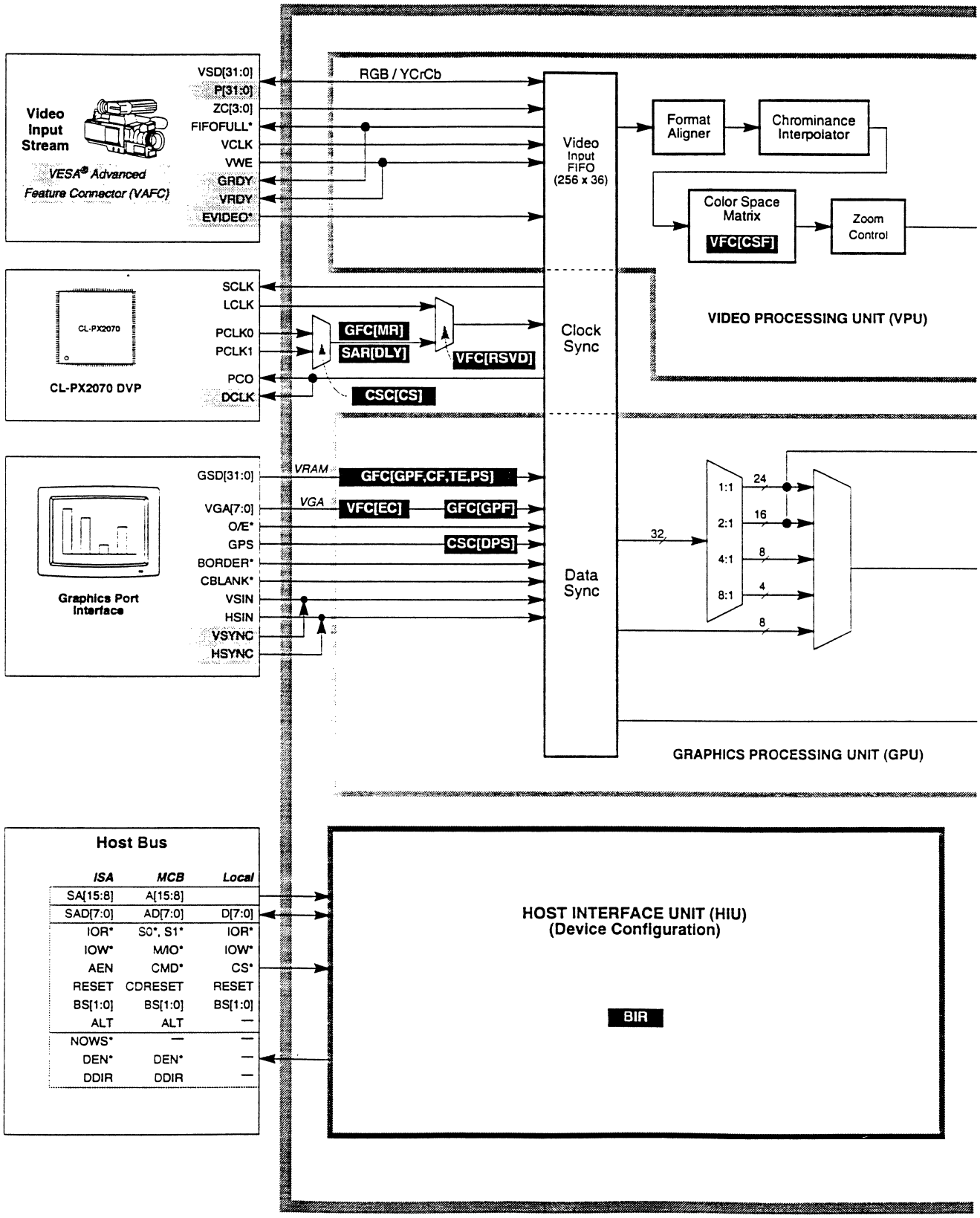
Figure 3-1. CL-PX208X MediaDAC Simplified Functional Block Diagram

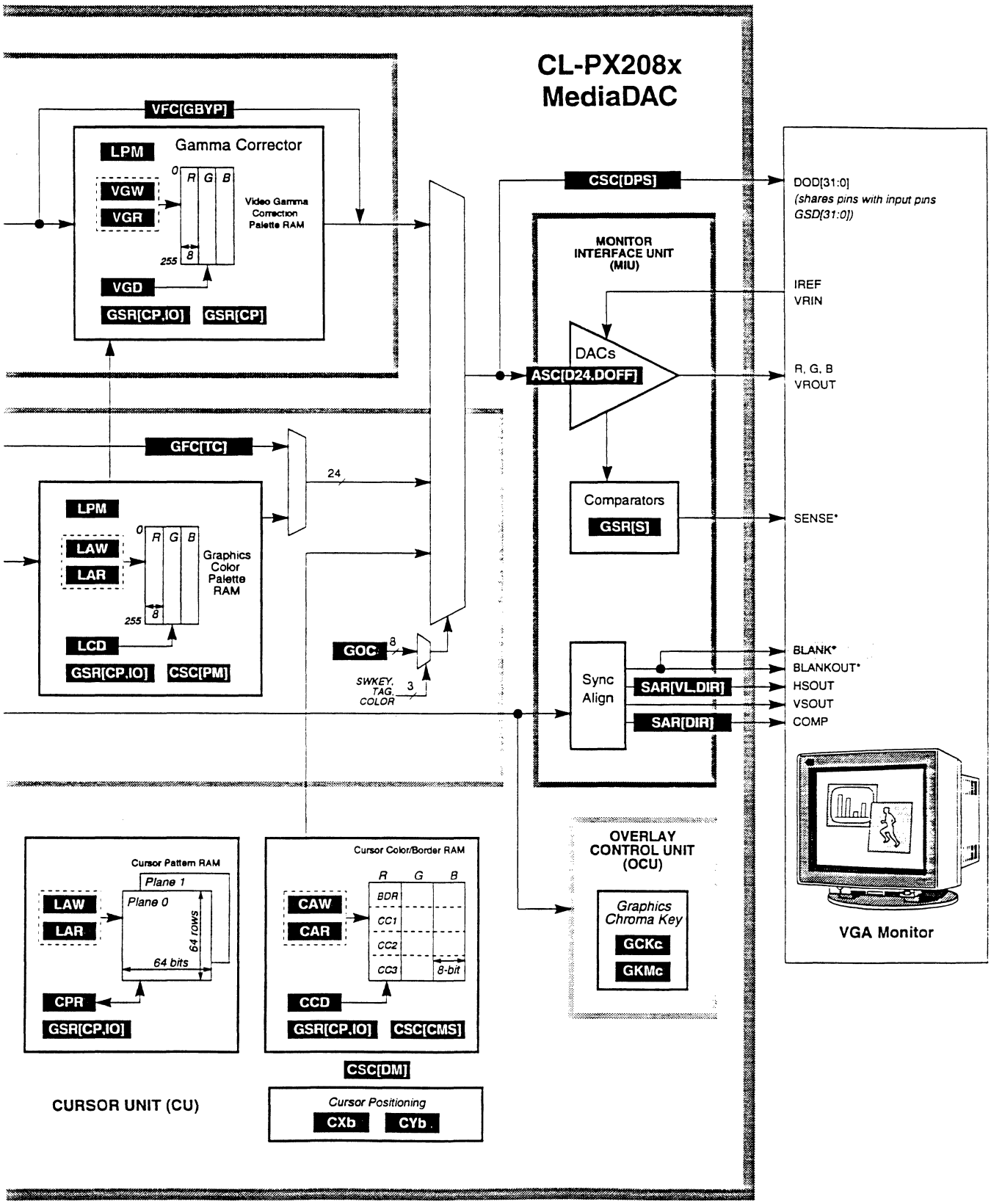
Figure 3-2 shows a more detailed MediaDAC block diagram. Refer to the figure while reading the following sections.

As described in the following sections, the primary functional blocks of the MediaDAC include:

- Host Interface Unit
- Video Processing Unit
- Graphics Processing Unit
- Overlay Control Unit
- Cursor Unit
- Monitor Interface Unit.

Figure 3-2. CL-PX208X MediaDAC Detailed Block Diagram




CL-PX208x
MediaDAC
VFC[GBYP]
LPM
Gamma Corrector
VGW
VGR
R G B
Video Gamma Correction Palette RAM
255
8
VGD
GSR[CP,IO]
GSR[CP]
CSC[DPS]
MONITOR INTERFACE UNIT (MIU)
DACs
ASC[D24,DOFF]
DOD[31:0]
(shares pins with input pins GSD[31:0])
IREF
VRIN
R, G, B
VROUT
GFC[TC]
24
LPM
LAW
LAR
Graphics Color Palette RAM
LCD
GSR[CP,IO]
CSC[PM]
GOC
8
SWKEY, TAG, COLOR
3
Comparators
GSR[S]
SENSE*
Sync Align
SAR[VL,DIR]
SAR[DIR]
BLANK*
BLANKOUT*
HSOUT
VSOUT
COMP
OVERLAY CONTROL UNIT (OCU)
Graphics Chroma Key
GCKc
GKMc
VGA Monitor
Cursor Pattern RAM
Plane 1
Plane 0
64 rows
64 bits
LAW
LAR
CPR
GSR[CP,IO]
Cursor Color/Border RAM
R G B
BDR
CC1
CC2
CC3
8-bit
CAW
CAR
CCD
GSR[CP,IO]
CSC[CMS]
CSC[DM]
Cursor Positioning
CXb
CYb
CURSOR UNIT (CU)

3.1 Architectural Overview

The MediaDAC is configured by the host system through the HIU, which provides access to all control registers for the device. For detailed register descriptions, refer to Section 4.

- The Video Processing Unit (VPU) accepts a digital-video data stream, and performs any necessary operations such as format and color-space conversion, zooming, and gamma correction/removal
- The Graphics Processing Unit (GPU) can accept 8-bit data from a VGA port or 32-bit data from a more advanced graphics subsystem. Multiple pixels can be transferred in one data word when lower color resolutions are used. A VGA-compatible color-palette RAM is provided, or 24-bit color can be passed directly to the output. Finally, graphics color keying and overlay controls provide sophisticated control over mixing of data that is sent to the Monitor Interface Unit
- The Monitor Interface Unit (MIU) includes a triple 8-bit DAC capable of operating at up to 135 MHz, an output-current comparator, and sync outputs. It directly drives standard analog SVGA-type CRT displays.

3.2 Host Interface Unit (HIU)

The MediaDAC interfaces with three bus protocols:

- Industry Standard Architecture (ISA)
- MicroChannel™ Bus (MCB)
- Local Hardware.

As shown in Figure 1-1 on page 16, the bus interface signals share a common set of I/O pins. For the complete pin assignment table, refer to Section 1. For detailed signal descriptions, refer to Section 2.

Table 3-1. Bus Selection Signals

BS[1:0]	Bus Selected
00	ISA
01	MCB
10	Local hardware interface (RS[4:0] addresses registers)
11	Local hardware interface (RS[3:0] addresses registers)

The MediaDAC connects directly to the ISA and MCB buses, internally decoding a 16-bit address issued by the host system and responding as an 8-bit peripheral. An index and data register pair provide access to the internal registers. In local hardware interface mode, the address range is externally decoded to drive signal CS*. Signals RS[4:0] select individual MediaDAC registers. Bus-selection signals BS[1:0] specify the host bus interface, as shown in Table 3-1.

The following sections describe the configuration method for each bus.

3.2.1 ISA Bus Interface

The MediaDAC interfaces with the ISA bus to support I/O read and write cycles. Since the LSB of the address and the data pins are multiplexed, a circuit similar to that shown in Figure 3-3 is required to prevent contention between address and data buffers.

The MediaDAC responds to I/O cycles on the ISA bus. Figure 3-4 shows a typical ISA 8-bit I/O cycle and compares the read and write cycles.

The following is the sequence of events for a read cycle:

1. A valid address within the address range of the MediaDAC stabilizes on the address bus. The address is decoded and asserts NOWS*.

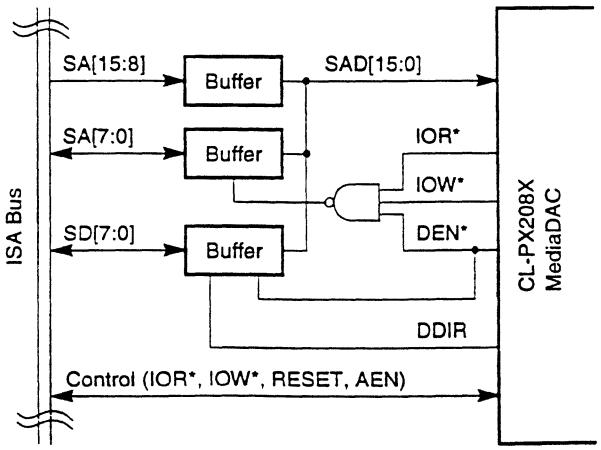


Figure 3-3. Example ISA Interface Circuit

2. The system asserts IOR*, causing the following:
   a. The MediaDAC latches the address on SA[15:8] and SAD[7:0] on the falling edge of IOR*;
   b. The I/O buffers of SAD[7:0] change from input to output mode (the circuit should be designed to disable the low-byte address buffer on IOR* assertion);
   c. DDIR goes low, using the external buffers to output to bus SAD[7:0];
   d. DEN* is asserted, disabling the address buffers and enabling the data buffers.
   e. After the appropriate time interval, the system negates IOR* and latches the data from SAD[7:0].
3. DDIR goes high.
4. DEN* is negated.
5. The I/O buffers of SAD[7:0] change from output to input mode.

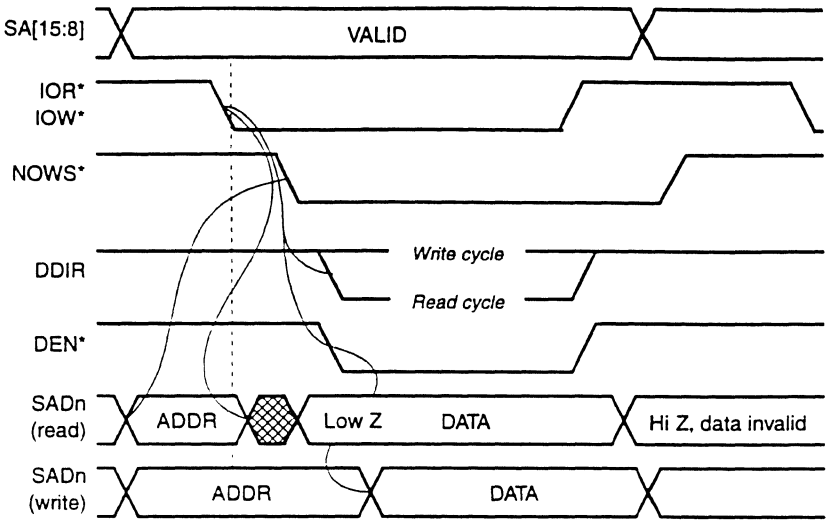


Figure 3-4. ISA 8-Bit I/O Cycle

3.2.2 MCB Interface

The MediaDAC supports I/O read and write cycles for the MCB environment. The connection of the multiplexed address and data pins to the MCB bus is similar to the ISA bus connection shown in Figure 3-4. (Refer to page 21 for the decoding performed for signals M/IO*, S0*, and S1*.)

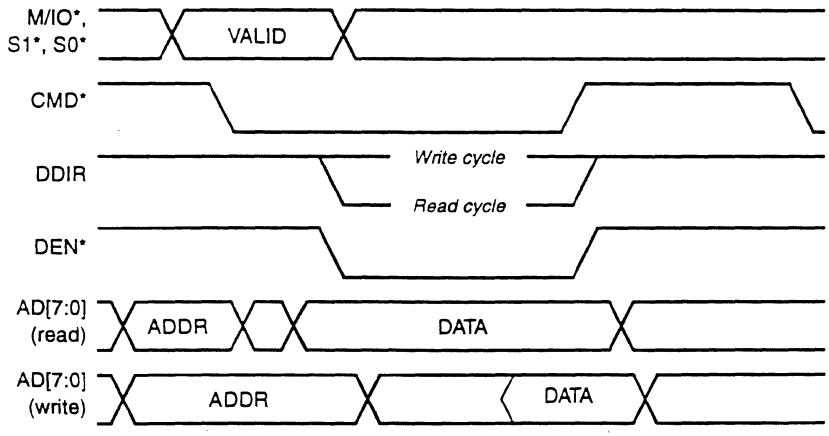


Figure 3-5. MCB 8-Bit I/O Cycle

3.2.2.1 MCB I/O Read

Figure 3-5 shows a typical MCB 8-bit I/O cycle. The MediaDAC latches the address present on A[15:8] and AD[7:0] on the falling edge of CMD*. During read operations, the MediaDAC provides valid data on bus AD[7:0] before the rising edge of CMD*. The MediaDAC outputs the data fast enough that no wait states are required. The signal CD-CHRDY is normally pulled high in the MCB environment and does not need to be driven by the MediaDAC. Since the MediaDAC is an 8-bit device, the MCB environment does not require it to drive signal CDDS16*.

3.2.3 Local Hardware Interface

The local hardware interface is used in a manner similar to a static RAM. The CL-PX2085 provides two access modes, one for direct addressing via RS[4:0], and another mode that uses only RS[3:0], described in Section 10.2.2. The interface has four components (described in the following sections) that determine the read and write operations of the local hardware interface:

- An 8-bit, bi-directional data bus
- Chip-select input signal CS*, which is driven by an external address decoder
- Signals RS[4:0], which select the register to be accessed and are typically connected to the lower five bits of the processor address bus
- Signals IOR* and IOW*, which define read and write cycles.

3.2.3.1 Local Access Cycles

Figure 3-6 shows the timing of a local hardware access. A read from the MediaDAC occurs when signals CS* and IOR* are low. Data from the addressed control register is placed on D[7:0] where it may be sampled by the host between the minimum specified access time (see Section 5.2.5) and the rising edge of IOR*.

A write occurs when CS* and IOR* are low. The host system asserts CS* after RS[4:0] are stable, then asserts IOW*. Data must be valid for the specified setup and hold times relative to the rising edge of IOW*.

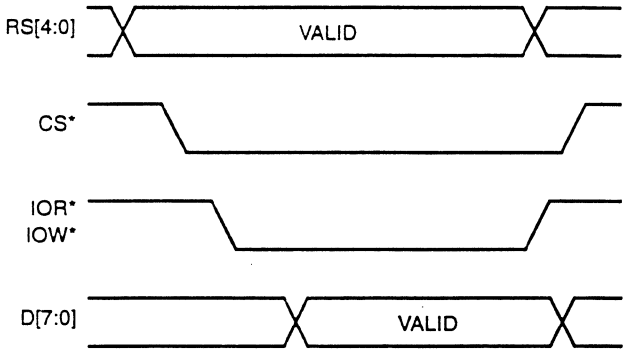


Figure 3-6. Local Hardware Interface Cycle

3.3 Video Processing Unit

The video frame buffer interface has a 32-bit data bus referred to as the video port. Four associated control bits, ZC[3:0], are called the *beta field*, which is used internally by the MediaDAC for X-zoom operations. The 32 data bits of the video port, VSD[31:0], can contain up to two pixels, depending on register VFC. The MediaDAC supports both tagged and untagged versions of 4:2:2 YCrCb, 5:5:5 RGB, and 8:8:8 RGB. The bitmapping of these formats onto VSD[31:0] is shown in Table 3-2. The processing of the incoming video stream is shown in Figure 3-9, and is detailed in the sections that follow.

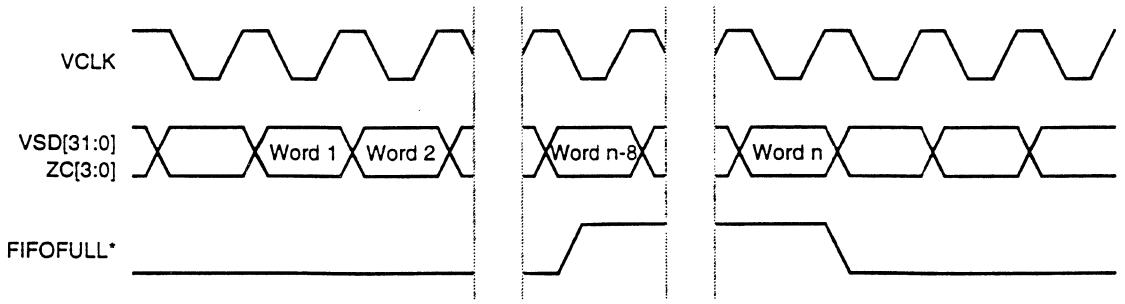


Figure 3-7. Video Input Timing

3.3.1 Video Input FIFO

Video port data is clocked into the MediaDAC independent of the graphics data. The MediaDAC latches video port data on the rising edge of signal VCLK. When the video input FIFO (full-8) flag goes active, the FIFOFULL is asserted internally before the next VCLK rising edge. Signal FIFOFULL* is asserted two PCLKn's later due to synchronizing circuitry. The video port data must be stable before and after the rising edge of VCLK. The video input FIFO is 256 double-pixels deep. The input format and other aspects of the VPU are programmed into register VFC. The Video Input FIFO supports:

- 24-bit RGB color data (≤ 1:1 multiplex) at up to 40-MHz pixel rates
- 16-bit RGB and 16-bit YCrCb (≤ 2:1 multiplex) at up to 80-MHz pixel rates.

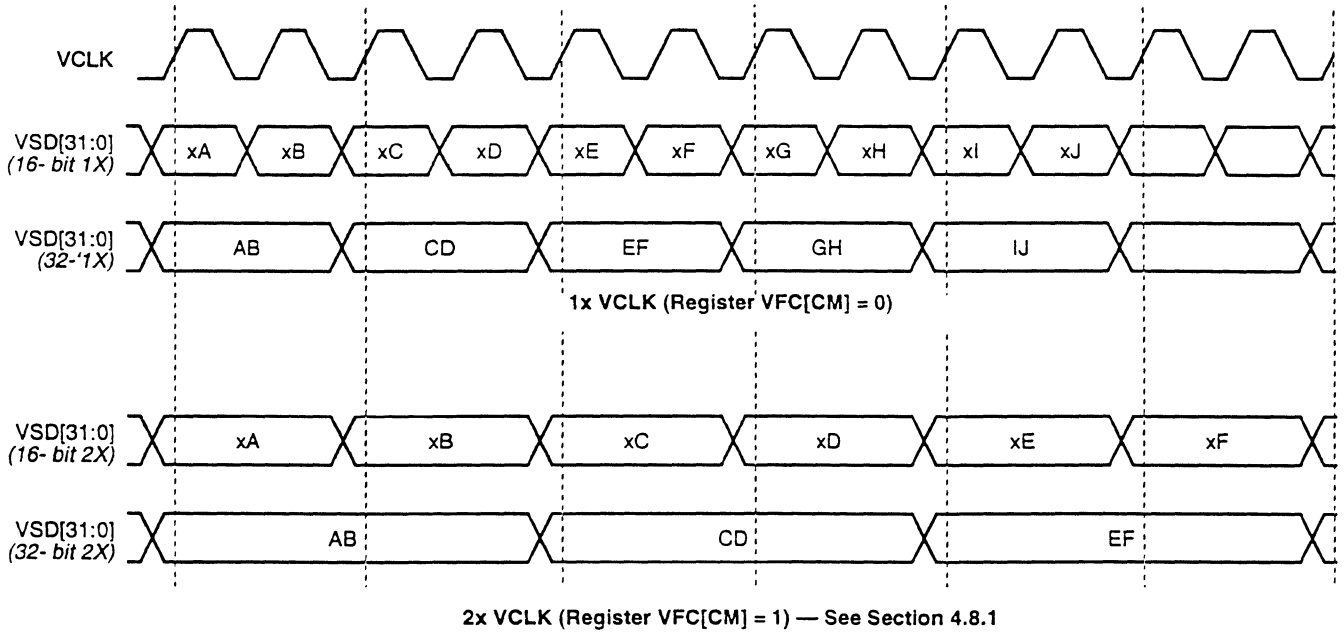


Figure 3-8. Video Input Formats

3.3.2 Video Processing Elements

The remaining video processing elements convert a variety of input formats into linear RGB:

- Format Aligner
- Chrominance Interpolator
- Zoom Control.
- Color-Space Converter
- Gamma Corrector

These processing elements operate in sequence as shown in Figure 3-9. Any stage not needed for a specific application can be bypassed using internal control registers.

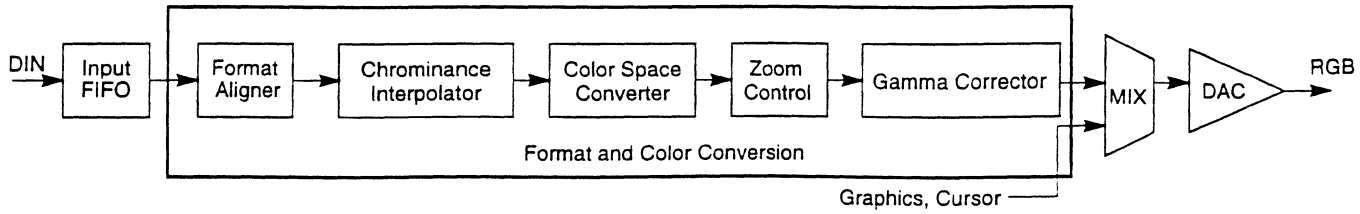


Figure 3-9. Video Input Processing Elements

All elements in the video pipeline are adjusted (or bypassed) automatically, depending on the format chosen in register VFC and on the zoom-control codes that are clocked in with the data. The gamma corrector can be bypassed using register VFC[GBYP].

3.3.3 Format Aligner

The format aligner accepts pixel input (signals VSD[31:0]) for the various pixel input formats described in Table 3-2 and converts it to 4:4:4 format, 1 YCrCb or RGB component per pixel clock. For formats requiring 16 bits per pixel or less, two pixels are packed in the 32-bit pixel word VSD[31:0]. Pipeline registers reformat data before it is passed into the color-conversion circuitry. The final format fed to the output DAC is 8:8:8 or 6:6:6 RGB. The notation is based on alignment to resultant 8-bit pixel values, where bit 7 is the MSb. For example, if Y_7, Y_6, Y_5, and Y_4 are specified, then data is left-justified out of the pipeline with the 4 LSbs padded with '0's. Also, when Y_0 and Y_1 are specified in the same input frame, Y_0 is the first luminance component in time.

3.3.3.1 RGB Video Input Data

For RGB video input data, the format aligner can be programmed to accept either:

- 5-6-5, 5-5-5 tagged, 8-8-8, or 8-8-8 tagged formats, where n-n-n indicates the bits allocated to the red, green, and blue planes respectively; or
- Pseudocolor, where the most-significant input byte is passed to the red, green and blue outputs.

Unused bit locations should be grounded prior to FIFO input. For RGB data, the input simply passes to the output in proper bit alignment.

3.3.3.2 YCrCb 4:2:2 Video Input Data

For 4:2:2 YCrCb video input data, the format aligner can be programmed to simultaneously accept two pixels of data in CCIR 601 format. The format aligner video input formats are shown on Table 3-3. An optional input tag may be used in place of the LSb of the chrominance values. Note that the chrominance values align with the odd luminance values. For 4:2:2 YCrCb data, the format aligner simply passes input data to the chrominance interpolator.

Table 3-2. Supported Pixel Word Input Formats

VSD	YCrCb		RGB			
	Non-Tag	Tagged	Non-Tagged		Tagged	
	16-Bit		16-Bit	24-Bit	16-Bit	24-Bit
31	$Y1_7$	$Y1_7$	$R1_7$	—	TAG_1	TAG_0
30	$Y1_6$	$Y1_6$	$R1_6$	—	$R1_7$	—
29	$Y1_5$	$Y1_5$	$R1_5$	—	$R1_6$	—
28	$Y1_4$	$Y1_4$	$R1_4$	—	$R1_5$	—
27	$Y1_3$	$Y1_3$	$R1_3$	—	$R1_4$	—
26	$Y1_2$	$Y1_2$	$G1_7$	—	$R1_3$	—
25	$Y1_1$	$Y1_1$	$G1_6$	—	$G1_7$	—
24	$Y1_0$	$Y1_0$	$G1_5$	—	$G1_6$	—
23	$V0_7$	$V0_7$	$G1_4$	$R0_7$	$G1_5$	$R0_7$
22	$V0_6$	$V0_6$	$G1_3$	$R0_6$	$G1_4$	$R0_6$
21	$V0_5$	$V0_5$	$G1_2$	$R0_5$	$G1_3$	$R0_5$
20	$V0_4$	$V0_4$	$B1_7$	$R0_4$	$B1_7$	$R0_4$
19	$V0_3$	$V0_3$	$B1_6$	$R0_3$	$B1_6$	$R0_3$
18	$V0_2$	$V0_2$	$B1_5$	$R0_2$	$B1_5$	$R0_2$
17	$V0_1$	$V0_1$	$B1_4$	$R0_1$	$B1_4$	$R0_1$
16	$V0_0$	TAG_1	$B1_3$	$R0_0$	$B1_3$	$R0_0$
15	$Y0_7$	$Y0_7$	$R0_7$	$G0_7$	TAG_0	$G0_7$
14	$Y0_6$	$Y0_6$	$R0_6$	$G0_6$	$R0_7$	$G0_6$
13	$Y0_5$	$Y0_5$	$R0_5$	$G0_5$	$R0_6$	$G0_5$
12	$Y0_4$	$Y0_4$	$R0_4$	$G0_4$	$R0_5$	$G0_4$
11	$Y0_3$	$Y0_3$	$R0_3$	$G0_3$	$R0_4$	$G0_3$
10	$Y0_2$	$Y0_2$	$G0_7$	$G0_2$	$R0_3$	$G0_2$
9	$Y0_1$	$Y0_1$	$G0_6$	$G0_1$	$G0_7$	$G0_1$
8	$Y0_0$	$Y0_0$	$G0_5$	$G0_0$	$G0_6$	$G0_0$
7	$U0_7$	$U0_7$	$G0_4$	$B0_7$	$G0_5$	$B0_7$
6	$U0_6$	$U0_6$	$G0_3$	$B0_6$	$G0_4$	$B0_6$
5	$U0_5$	$U0_5$	$G1_2$	$B0_5$	$G0_3$	$B0_5$
4	$U0_4$	$U0_4$	$B1_7$	$B0_4$	$B0_7$	$B0_4$
3	$U0_3$	$U0_3$	$B1_6$	$B0_3$	$B0_6$	$B0_3$
2	$U0_2$	$U0_2$	$B1_5$	$B0_2$	$B0_5$	$B0_2$
1	$U0_1$	$U0_1$	$B1_4$	$B0_1$	$B0_4$	$B0_1$
0	$U0_0$	TAG_0	$B1_3$	$B0_0$	$B0_3$	$B0_0$

3.3.3.3 Chrominance Interpolator

The chrominance interpolator increases the sampling rate of the color-difference signals when they are input at a rate less than 4:4:4, and it acts as a data source for the color space matrix. The chrominance interpolator always operates in the same mode as the format aligner.

The chrominance interpolator contains two identical circuits — one for the U and one for the V component. The output of the time demultiplexer feeds the chrominance interpolator input, setting the missing U and V values to '0' on display window boundary conditions.

Table 3-3. YCrCb 4:2:2 Format

Luminance Values									Chrominance Values								
Y Frame	1	2	3	4	5	6	7	8	UV Frame	1	2	3	4	5	6	7	8
Y7	Y07	Y17	Y07	Y17	Y07	Y17	Y07	Y17	UV7	U7	V7	U7	V7	U7	V7	U7	V7
Y6	Y06	Y16	Y06	Y16	Y06	Y16	Y06	Y16	UV6	U6	V6	U6	V6	U6	V6	U6	V6
Y5	Y05	Y15	Y05	Y15	Y05	Y15	Y05	Y15	UV5	U5	V5	U5	V5	U5	V5	U5	V5
Y4	Y04	Y14	Y04	Y14	Y04	Y14	Y04	Y14	UV4	U4	V4	U4	V4	U4	V4	U4	V4
Y3	Y03	Y13	Y03	Y13	Y03	Y13	Y03	Y13	UV3	U3	V3	U3	V3	U3	V3	U3	V3
Y2	Y02	Y12	Y02	Y12	Y02	Y12	Y02	Y12	UV2	U2	V2	U2	V2	U2	V2	U2	V2
Y1	Y01	Y11	Y01	Y11	Y01	Y11	Y01	Y11	UV1	U1	V1	U1	V1	U1	V1	U1	V1
Y0	Y00	Y10	Y00	Y10	Y00	Y10	Y00	Y10	UV0	U0	V0	U0	V0	U0	V0	U0	V0

3.3.3.4 Color Space Converter

The color space converter is enabled automatically when it is necessary to convert the video data to the RGB format that is passed to the DAC. For more information, see register VFC, page 68.

3.3.3.5 RGB Video Input Data

The chrominance interpolator is not required when using RGB video input data.

3.3.4 Gamma Correction

The gamma corrector comprises three 256 x 8 memories, one for each color channel. It accepts input from the zoom-control circuitry and outputs video data to the mixing circuitry for alignment with incoming graphics, cursor, and background border. It can be programmed either with a custom correction table or to remove the gamma coding that is normally present in a YCrCb television signal. The transfer function is user-definable and programmable.

The gamma corrector is an optional feature; to bypass it, set register VFC[GBYP] to '1'.

See Section 4.9 for more details on accessing the Video Gamma Correction Palette RAM.

3.3.5 VAFC Port (2085 only)

The CL-PX2085 contains a video port that complies with the VESA Synchronous Video Interface (VSVI).

3.3.5.1 VAFC Output Mode

The CL-PX2085 outputs signals VGA[7:0] or GSD[31:0] data on pins VSD[31:0]. The VGA[7:0]/ GSD[31:0] data is sampled on the rising edge of LCLK, and output through VSD[31:0], as shown in Figure 3-10. The format of the GSD[31:0] data is unchanged from the input. The VSD port options are shown in Table 3-4.

NOTE: PCO is the VAVI DCLK signal.

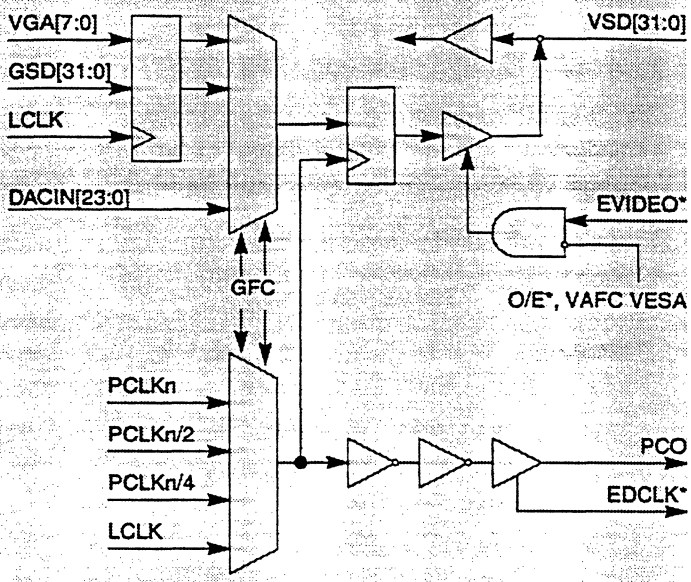


Figure 3-10. VAFC Output Mode

Table 3-4. VSD Port Options

Mode	CLK Ratio	PCO
Output		
VGA	1:1	LCLK
GSD4	8:1	LCLK
GSD8	4:1	LCLK
GSD16	2:1	LCLK
GSD16	1:1	LCLK
GSD24	1:1	LCLK
DAC-IN	1:1	PCLK
Input		
16-BIT 1x	1:1	PCLK
16-BIT 2x	2:1	PCLK/2
32-BIT 1x	2:1	PCLK/2
32-BIT 2x	4:1	PCLK/4
24-BIT 1x	1:1	PCLK

3.3.5.2 Default Power-up Conditions

When the CL-PX2085 powers up, register VAFC VESA Advanced Feature Connector[OE*] is reset to '0'; VGA[7:0] is driven out on VSD[7:0], with VSD[31:8] in output mode in tristate condition. EVIDEO* is asserted by an external device when data is to be driven into VSD[31:0].

3.3.5.3 VAFC Input Mode

The CL-PX2085 accepts input data on VSD[31:0] in the formats summarized in Table 3-5. Register bits VAFC[VM] support VAVI 16- and 32-bit modes. Register bit VAFC[V*] defines the VAFC mode. When VAFC[V*] = '0', the device is in standard VAFC mode, and VAFC[VM] selects the interface mode of the VAFC connector.

Offset pins are not supported. In 32-bit 2x mode, the CL-PX2085 interpolates by inserting appropriate zoom codes for a 2x zoom after the FIFO. The video pixel(s) appearing on the VAFC interface in all 2x modes will be blanked when signals ZC[3:0] are equal to 0x0C [0Ch]. The first pixel in the video stream is the LSB.

Table 3-5. CL-PX2085 Input Formats for VSD[31:0]

Register VAFC [V*]	Register VAFC [VM]	Signal VFC[3:0]	Input CLK VCLK	Description
1	xx	0000	PCLK/2	32-bit YCrCb 422
1	xx	0001	PCLK/2	32-bit YCrCb 422T
1	xx	1000	PCLK/2	32-bit RGB 565
1	xx	1010	PCLK/2	32-bit RGB 555
1	xx	1011	PCLK/2	32-bit RGB 555T
1	xx	1110	PCLK	24-bit RGB 888
1	xx	1111	PCLK	24-bit RGB 888T
0	00	0000	PCLK	16-bit 1x YCrCb 422
0	01	0000	PCLK/2	16-bit 2x YCrCb 422
0	10	0000	PCLK/2	32-bit 1x YCrCb 422
0	11	0000	PCLK/4	32-bit 2x YCrCb 422
0	00	1000	PCLK	16-bit 1x RGB 565
0	01	1000	PCLK/2	16-bit 2x RGB 565
0	10	1000	PCLK/2	32-bit 1x RGB 565
0	11	1000	PCLK/4	32-bit 2x RGB 565

x = don't care

3.4 Graphics Processing Unit

The MediaDAC accepts data from the graphics display source through either of two paths — an 8-bit VGA data path (VGA[7:0]) or a 32-bit VRAM serial data path (GSD[31:0]). One data path is selected at a time. Signal GPS (and conditions described in Section 2.4 on page 24) determine which input is selected. These two paths are provided to allow next-generation PC-graphics subsystems based on the MediaDAC to maintain compatibility with the large installed base of VGA systems while achieving higher performance and higher resolution via the VRAM serial data path.

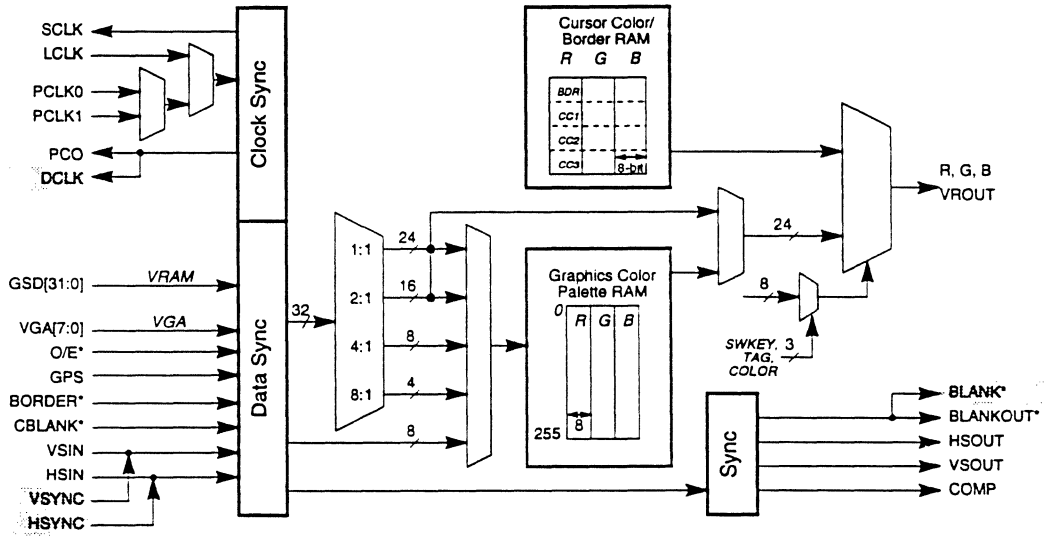


Figure 3-10. MediaDAC Graphics Datapath

3.4.1 VRAM Support

The graphics serial bus interface has a 32-bit data bus. The data on this bus is multiplexed under control of the graphics format control register GFC and is latched internally on the rising edge of LCLK. LCLK must be supplied by the graphics controller and should be derived from SCLK. SCLK is derived from PCLK, and register bits GFC[MR, GPF] control the SCLK rate.

3.4.2 VRAM Operation

3.4.2.1 Graphics Data — GSD[31:0]

The VRAM shift clock (SCLK) is generated by the MediaDAC. In 1:1 VGA mode, SCLK equals LCLK. Table 3-6 describes the relationship of SCLK and LCLK in various pixel modes.

GSD[31:0] is the input pixel data, 8 bits per pixel (4:1 multiplexed) and 4 bits per pixel (8:1 multiplexed) for four and eight horizontally consecutive output pixels. GSD[31:0] is always latched on the rising edge of LCLK.

Table 3-6. SCLK and LCLK Relationships

Multiplex Ratio	SCLK/LCLK Relationship
8:1	SCLK = LCLK = 8 PCLK
4:1	SCLK = LCLK = 4 PCLK
2:1	SCLK = LCLK = 2 PCLK
1:1	SCLK = LCLK = PCLK

3.4.2.2 GSD[31:0] Mapping to Pixel Port Interface

In all but 8:1 mode, the least-significant word, byte, or nibble is the first to be displayed in time. For example, when in 4:1 mode, there are four 8-bit pixel ports encoded within GSD[31:0]. Port GSD[7:0] corresponds to the first pixel of the first line of the display. This is the first pixel fed to the analog outputs, followed by GSD[15:8], then GSD[23:16], and finally GSD[31:24], repeating the pattern from LSB to MSB until the first scan line is completely displayed. The pixel ordering for all modes, including 8:1, are illustrated in Figure 3-12.

The graphics pixel data is used to look up color information in the Graphics Color Palette, and can be masked before the lookup occurs. Register LPM masks the address. The graphics pixel is logically AND'ed with the LPM data, and the result is used to address the Graphics Color Palette RAM. See Table 3-7.

Table 3-7. Pixel Index Mapping — Pixel Mask vs. VGA[7:0] and GSD[31:0] Bit Locations

Register/Format Type Description	MSb							LSb	Function
Register LPM	7	6	5	4	3	2	1	0	Register Bits
VGA[7:0]	7	6	5	4	3	2	1	0	Palette Index
4 Bits/Pixel	x	x	x	x	3	2	1	0	Palette Index
8 Bits/Pixel	7	6	5	4	3	2	1	0	Palette Index
16 Bits/Pixel 5:5:5 Format Sparse	14 9 4	13 8 3	12 7 2	11 6 1	10 5 0	x x x	x x x	x x x	Red Palette Index Green Palette Index Blue Palette Index
16 Bits/Pixel 5:5:5 Format Contiguous	x x x	x x x	x x x	14 9 4	13 8 3	12 7 2	11 6 1	10 5 0	Red Palette Index Green Palette Index Blue Palette Index
16 Bits/Pixel 5:6:5 Format Sparse	15 10 4	14 9 3	13 8 2	12 7 1	11 6 0	x 5 x	x x x	x x x	Red Palette Index Green Palette Index Blue Palette Index
16 Bits/Pixel 5:6:5 Format Contiguous	x x x	x x x	x 10 x	15 9 4	14 8 3	13 7 2	12 6 1	11 5 0	Red Palette Index Green Palette Index Blue Palette Index
24 Bits/Pixel 8:8:8 Format	23 15 7	22 14 6	21 13 5	20 12 4	19 11 3	18 10 2	17 9 1	16 8 0	Red Palette Index Green Palette Index Blue Palette Index

3.4.2.3 Odd/Even Field Definition

The output data sequence depends on register bit CSC[DM] and signal O/E* input. The MediaDAC treats interlaced graphics data in the same manner as non-interlaced data, merely transferring it for output processing. Interlaced data alignment must be performed and controlled outside the MediaDAC. Cursor pattern RAM data, however, is managed by the MediaDAC in interlaced mode.

In interlaced mode, scan line 1 is always displayed first and is considered the first line of the even field. In non-interlaced mode, scan line 2 immediately follows scan line 1. In interlaced mode, scan line 2 is considered to be the first line of the odd field and is displayed only after the entire even field has been displayed and the ODD/EVEN* pin has toggled.

Only the odd lines or only the even lines will be displayed if the ODD/EVEN* input signal does not change. Figure 3-11 shows the interlaced and non-interlaced display scan. A non-interlaced display scan is equal to one frame. An interlaced display scan is equal to one frame with odd and even fields.

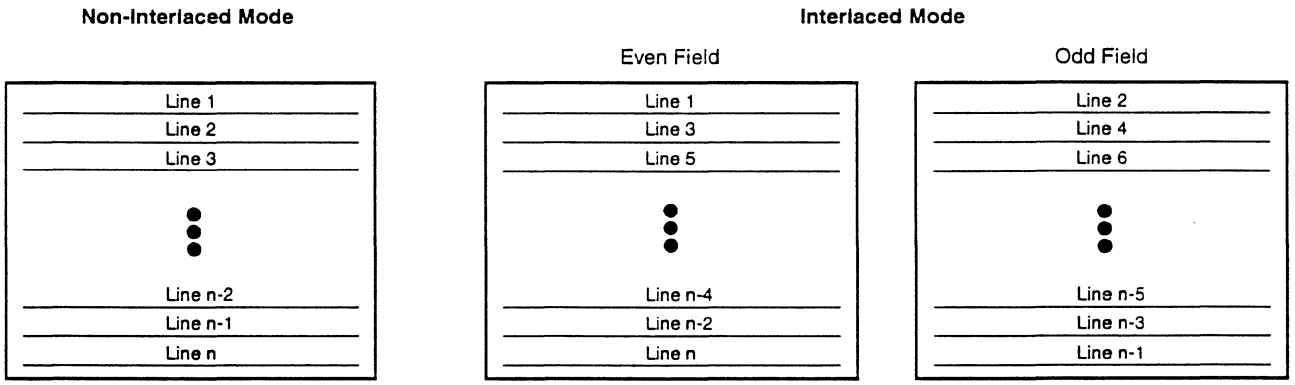


Figure 3-11. Interlaced and Non-Interlaced Display Scans

3.4.2.4 Pixel Read-Mask Register

Each pixel clock cycle, GSD[31:0] pixel data is AND'ed with the contents of the pixel read-mask register LPM, and the result is used to address the color palette RAM. The addressed location provides 24 bits of color information to the three D/A converters. Pixel masking is enabled for all modes of operation except when the true-color bypass is enabled.

3.4.3 VRAM Port Modes of Operation

The 32-bit VRAM port can be configured to act as a varying number of parallel pixel ports, depending on the number of bits defined for each pixel. Figure 3-12 describes four possible combinations. In each case, pixel data is latched on the rising edge of LCLK.

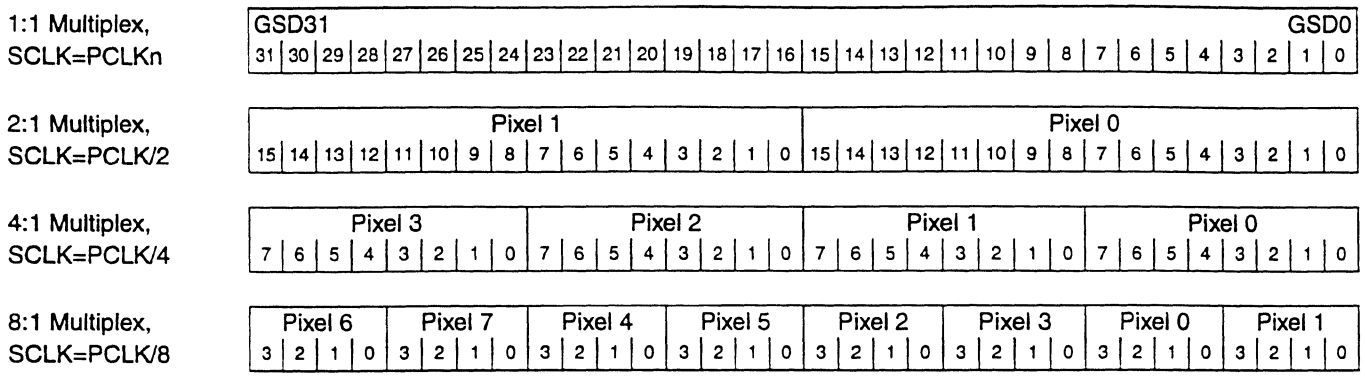


Figure 3-12. Pixel Mapping with Graphics Port

3.4.3.1 16-Bits/Pixel Operation (1:1 Multiplexed)

The 1:1 multiplexing mode is selected when GFC[MR] = 1. In this mode, two independent 16-bit pixel ports, GSD[31:16] (as pixel 1) and GSD[15:0] (as pixel 0), are latched on the rising edge of LCLK, and are multiplexed 1:1. When GFC[TE] = 0, GFC[PS] selects between the two ports. One LCLK rising edge occurs every PCLK cycle. SCLK is equal to the current PCLK selected.

GSD[31] switches between the two ports on a pixel-by-pixel basis when 5:5:5 RGB color format (GFC[CF] = 0) and real-time pixel port switching (GFC[TE] = 1) are enabled. GSD[15] is ignored internally when in 5:5:5 mode. Real-time pixel port switching is not supported for 5:6:5 RGB color format. (See Table 3-8.)

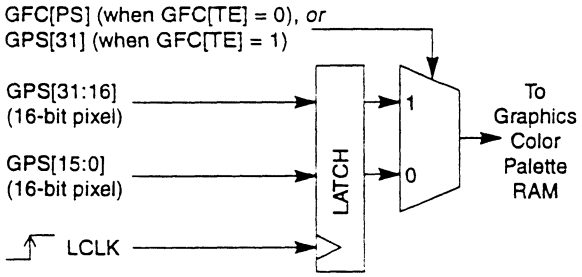


Figure 3-13. Graphics Port Function, Two-Port Mode

Table 3-8. Color Mapping to GSD[31:0] Bus

Mode	MSb															LSb
5:5:5	S	R4:0					G4:0					B4:0				
Port 1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Port 2	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
5:6:5	R4:0					G5:0						B4:0				
Port 1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Port 2	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

3.4.3.2 True-Color Operation

Register GFC[TC], enables or disables the true-color palette bypass feature. When bypass mode is selected, the pixel data are transferred directly to the proper bytes of the respective DACs, bypassing the palette and the pixel mask. When the bypass mode is not selected, the pixel data index the proper locations in the palette, and the correct color information is passed to the respective DACs. Register bit CSC[PM] selects sparse or contiguous palette mapping.

For sparse palette mapping, each independent color component of pixel data is mapped to the most-significant bits of the respective palette address; the least-significant bits are set to '0'. For contiguous palette mapping, each independent color component of the pixel data is mapped to the least-significant bits of the respective palette address; the most-significant bits are set to '0'. For either sparse or contiguous mapping, the specified color palette values are transferred to the DACs.

When 5:5:5 or 5:6:5 color format is selected, the display can contain 32K or 64K simultaneous colors respectively. The DACs can be configured for 6 or 8 bits of resolution in this mode.

Table 3-9. Graphics Port Operating Mode

Mode	Mux Rate	Signals		Register GFC					Ports/Mux Order
		GPS*	GSD[31]	GPF	CF	MR	TE	PS	
VGA	1:1	0	x	00	x	x	x	x	VGA[7:0]
4 Bits/Pixel	8:1	x	Data	11	x	x	x	x	GSD[7:4] GSD[3:0] GSD[15:12] GSD[11:8] GSD[23:20] GSD[19:16] GSD[31:28] GSD[27:24]
8 Bits/Pixel	4:1	x	Data	10	x	x	x	x	GSD[7:0] GSD[15:8] GSD[23:16] GSD[31:24]
16 Bits/Pixel (5:5:5)	2:1	x	x	01	0	0	x	x	GSD[15:0] GSD[31:16]
	1:1	1	x	01	0	1	0	0	GSD[15:0]
		1	x	01	0	1	0	1	GSD[31:16]
		1	0	01	0	1	1	x	GSD[15:0]
16 Bits/Pixel (5:6:5)	2:1	x	Data	01	1	0	x	x	GSD[15:0] GSD[31:16]

3.4.4 VGA Support

The VGA[7:0] data path is an 8-bit input bus multiplexed with the VRAM serial data path under control of CSC[DPS].

3.4.4.1 VGA-Compatible Modes

The MediaDAC has the flexibility to operate in a system with a separate VGA controller/DAC or to operate as the sole display output. Many currently used VGA/DAC ICs internally decode the ISA-standard addresses for the palette RAM. The MediaDAC is capable of transparently replacing an existing DAC, operating in parallel, or operating as a separate subsystem at a unique address. Three access modes are available to maintain compatibility with software designed for VGA registers.

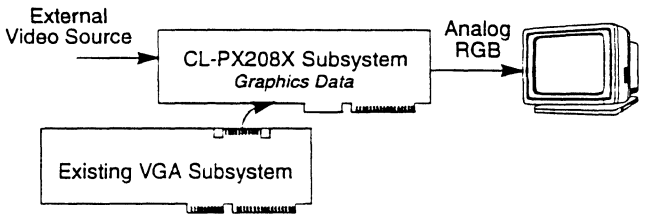


Figure 3-14. VGA Mode 0 System Configuration

These modes are used with ISA and MCB buses, and are selected by register BIR[IE, RE, and RO], as described in Section 4.2.1.

VGA Mode 0 is for a system with a separate, pre-existing VGA controller and palette DAC. Graphic data from the auxiliary (or 'feature') connector of the existing controller board enters the VGA graphics port of the MediaDAC where it is mixed with video data from an external source.

As shown in Figure 3-16, a single external monitor is connected to the analog-RGB output of the CL-PX208X. To preserve the same functionality with all VGA software drivers, the MediaDAC must accept writes to the standard VGA palette addresses but *not* respond to reads, allowing the existing VGA controller board to respond. The MediaDAC palette RAM shadows the VGA controller RAM on writes only. Mode 0 is also useful in designs configured like that shown in Figure 3-16, when the CL-PX208X is used with a self-decoding VGA controller.

VGA Mode 1 is designed for a system that has a VGA controller/palette DAC and a MediaDAC in the same subsystem. Graphic data from the auxiliary (or 'feature') connector of the existing controller board enters the VGA graphics port of the MediaDAC where it is mixed with video data from an external source, as shown in Figure 3-16.

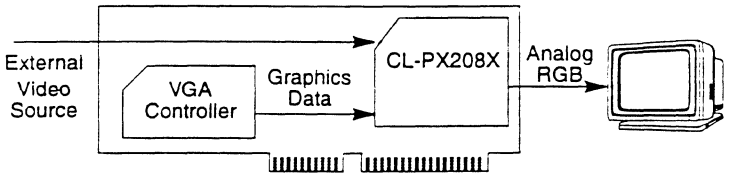


Figure 3-15. VGA Mode 1 System Configuration

Mode 1 differs from Mode 0 in that it responds to reads at the palette RAM address. This system is compatible with existing software that manipulates the VGA palette RAM.

VGA Mode 2 is designed for a system containing a VGA controller (with palette DAC) and a MediaDAC subsystem, each driving separate RGB monitors as shown in Figure 3-16. In this scenario the VGA subsystem occupies the standard VGA palette RAM addresses and the MediaDAC registers respond to a completely separate set of addresses.

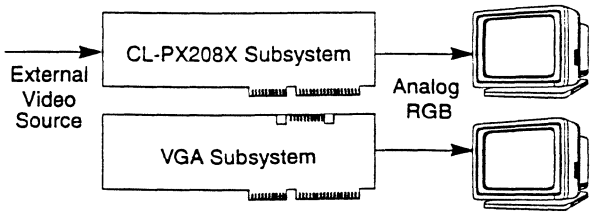


Figure 3-16. VGA Mode 2 System Configuration

3.4.4.2 Using the VGA Port in Extended Color Modes

In the Extended Color modes (specified by register VFC[EC]), the MediaDAC VGA port accepts 16-bit data, as illustrated in Figure 3-17. Extended color modes include compatibility with HiCOLOR™ modes used in some devices.

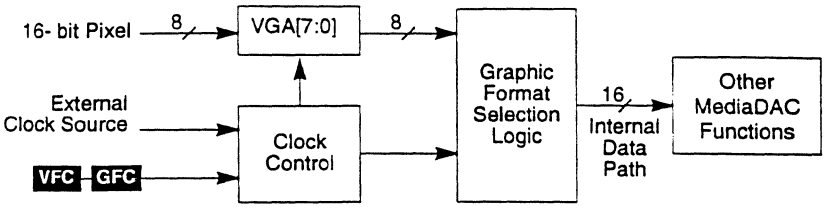


Figure 3-17. Accepting 16-Bit Color Data Through VGA[7:0]

In true-color mode, the color look-up table and read-mask circuitry are bypassed, allowing the display of a maximum of 64K colors.

Color-Data Clocking in EC Mode

In EC mode, two bytes are transferred each LCLK cycle. Data are transferred LSB first. The most-significant byte is sampled on the falling edge, and the least-significant byte is sampled on the rising edge as shown in Figure 3-18.

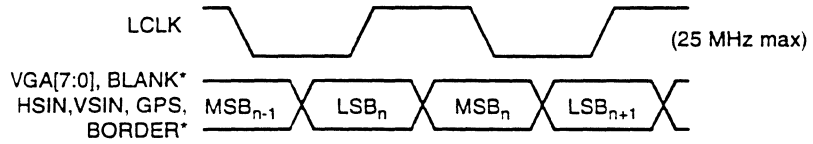


Figure 3-18. EC Mode 1 Timing

3.4.5 Hardware Cursor Operation

The MediaDAC has an on-chip, three-color, user-definable cursor, which is implemented in 32 x 32 x 2-bit memory (64 x 64 x 2 for the CL-PX2085). This cursor works in both interlaced and non-interlaced systems. The cursor can be programmed via register CSC for three modes of operation summarized in Table 3-10. For clarity, in the following descriptions and illustrations concerning the cursor, the cursor is shown with only 32 x 32-pixel dimensions.

The pattern for the cursor is provided by the cursor RAM, which can be accessed by the BIU at any time. Cursor positioning is performed in the cursor-position registers (CXH, CXL, CYH, and CYL). Figure 3-19 demonstrates its use.

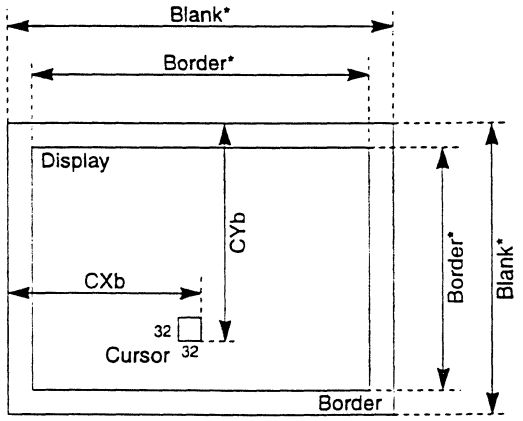


Figure 3-19. Cursor Operation

A value of (0,0) written to the cursor-position registers places the cursor completely off-screen, outside the viewing area. A cursor-position value of (1,1) places the lower-right pixel of the cursor on the upper left-hand corner of the screen. Only one cursor pattern per frame is displayed, regardless of updates to the cursor-position registers. The single restriction on position-register updates is that all position registers must be written when the cursor location is updated.

The internal position register is updated when the Y upper byte (register CYH) is written to ensure this operation. The cursor pattern is displayed at the last cursor location written prior to frame start. The reference point of the cursor, row 0 column 0, is the lower-right corner of the cursor relative to BORDER*.

The position of the cursor is not dependent upon CBLANK*. The cursor X position is relative to the first rising edge of LCLK when BORDER* is sampled at a logical '1'. The cursor Y position is relative to the first rising edge of LCLK when BORDER* is sampled at a logical '1' after the vertical blanking period has been determined.

The cursor pattern can be displayed in an interlaced system if register CSC[DM] is a logical '1'. The first cursor line displayed (row 31 of cursor pattern RAM) depends on the state of signal O/E* and the position value in registers CYH and CYL. If the Y position is an even number, the data in row 31 is displayed during the even field, starting at position (CX-31, CY-31), where CX is the concatenated position determined by registers CXH and CXL (and similarly with Y). Each subsequent scan line displayed in the even field corresponds to every alternate active cursor line after row 31 in the cursor RAM array. During odd fields, the even rows from the cursor pattern RAM are displayed starting with row 30 at position (CX-31, CY-30). Each subsequent scan line displayed in the odd field corresponds to every alternate active cursor line after row 30 in the cursor RAM array. Similarly, if the Y-position value was an odd number in the first line displayed, then row 31 and subsequent odd rows would be displayed during an odd field. Row 30 data and subsequent even rows would be displayed during the even field.

3.4.6 Internal Memory Access

3.4.6.1 Color RAM Data Access

Color-palette, gamma-palette, cursor, and cursor-border colors are specified in terms of 24-bit RGB data, one byte per color. The host processor accesses a color memory location by first writing the index address, then performing three successive writes or reads to the data register. Upon completion of the third access, the index register points to the next location. The two internal bits used for this function are reset to '0' upon a write to the address register, but they are unaffected by a read.

For example, to update the color palette data, the processor writes the MediaDAC address register (RAM write mode) with the address of the color palette RAM location to be modified. The processor performs three successive write cycles (8 bits each of red, green, and blue). After the blue write cycle, the three bytes of color information are concatenated into a 24-bit word and written to the location specified by the address register. The address register then increments to the next location, which the processor can modify by simply writing another sequence of red, green, and blue data. To write to a block of color values in consecutive locations, write the start address and perform continuous R, G, B write cycles until the entire block has been written. Cursor and other color information is handled the same way. Refer to appropriate Bus Interface Unit configuration for read/write timing.

When accessing the color palette RAM, the address register resets to '00h' following a blue read or write to RAM location FFh. The processor interface operates asynchronously with respect to the pixel clock. Data transfers between the color palette RAM and the color registers (R, G, and B in the block diagram) are synchronized by internal logic; the transfers occur in the period between processor accesses. To reduce noticeable sparkling on the CRT screen during processor access to the color palette RAM, internal logic maintains the previous output color data on the analog outputs while the transfer between RGB registers and look-up table RAMs occurs.

3.5 Cursor Unit

The Cursor Unit (CU) includes the cursor-pattern RAM, cursor-color palette RAM (also contains border color), cursor-position registers.

3.5.1 Accessing the Cursor Color/Border RAM

The 32 x 32 x 2 cursor RAM is accessed in a planar format where plane 1 is bit 1 of the cursor data and plane 0 is bit 0. In the planar format, only seven address bits are used. The eighth bit determines which Plane (0 or 1) data of the cursor RAM array is accessed (see Figure 3-20). A single address presented to the cursor RAM accesses 8-bit locations in plane 0 or 1, depending on the state of CPR data bit 7.

The 64 x 64 cursor uses 8 bits with the ninth bit defining the plane.

Refer to Sections 4.4 and 4.6.5 for more information.

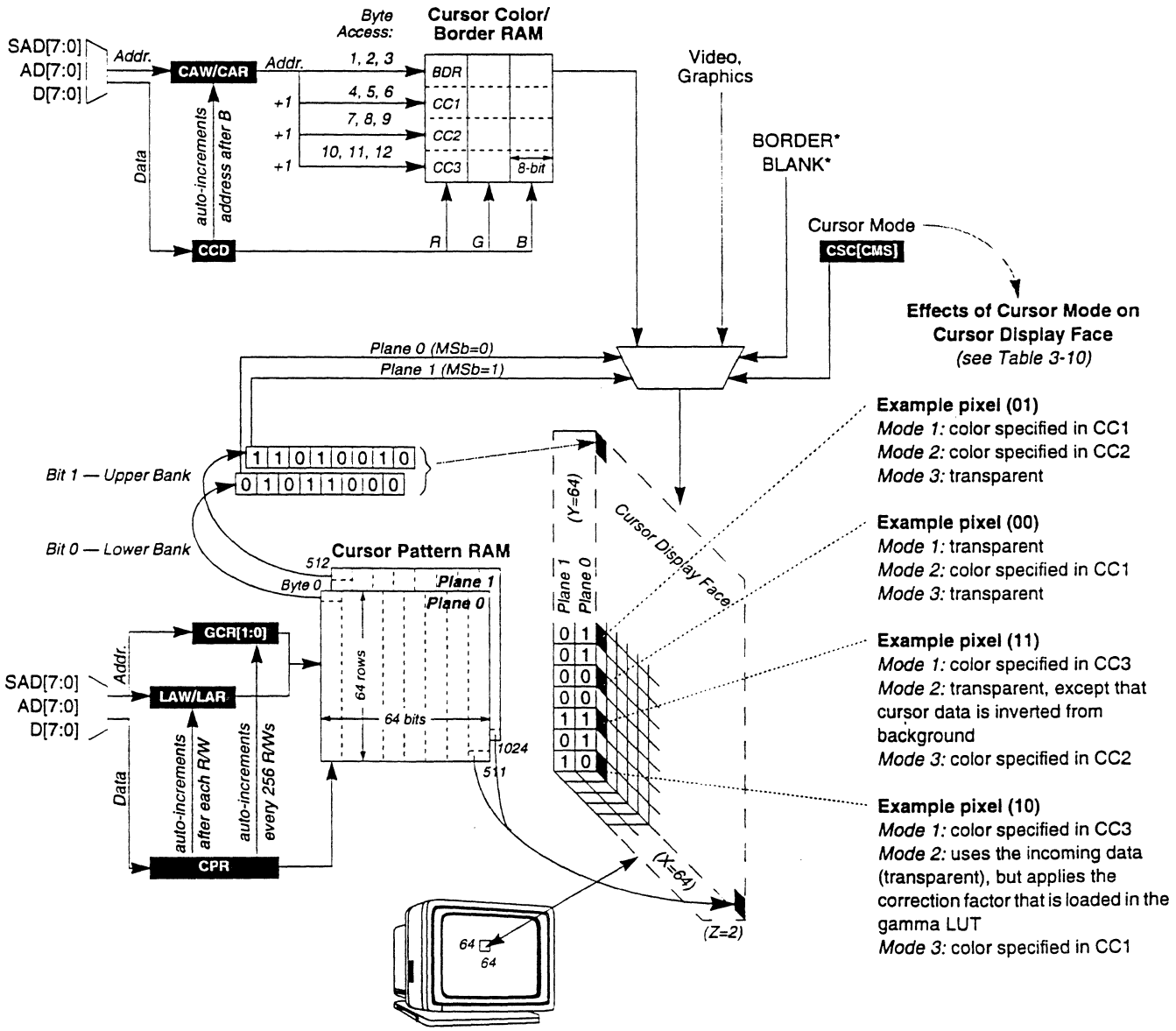


Figure 3-20. Cursor RAM Function Diagram

Table 3-10. Cursor Memory Mapping and Relationships for Display

Bit 1	Bit 0	Mode 1	Mode 2	Mode 3
0	0	Color/Gamma Palette Data	CC1	Color/Gamma Palette Data
0	1	CC1	CC2	Color/Gamma Palette Data
1	0	CC2	Color/Gamma Palette Data	CC1
1	1	CC3	Color/Gamma Palette Data (inverted)	CC2

3.6 Overlay Control Unit

The Overlay Control Unit (OCU) includes the functions that determine whether video or graphics data is sent to the DAC.

3.6.1 Mixing to Output DAC

Table 3-11. DAC Data Source Control

BORDER*	CBLANK*	PORTSEL	DAC Output Characteristics
X	0	X	Blanked area
0	1	0	VGA data
0	1	1	Overscan (border)
1	1	0	VGA data, cursor data
1	1	1	GSD data, cursor data

The Output DAC contains three 8-bit digital-to-analog converters. Table 3-11 shows how signals BORDER* and CBLANK* and the internal signal PORTSEL (defined with pin GPS on page 24) control the graphic data path at the input of the DAC.

3.6.2 Graphics Overlay Control

The graphics overlay controls consist of a 32-bit color key, a 32-bit color key mask, an 8-bit overlay opcode, and an 8:1 multiplexer.

To understand how the graphics overlay controls work, imagine the MediaDAC as managing two images, one in front of the other. The two image planes are the video and graphics images, with the video image behind the graphics image. Every graphics pixel is either transparent or opaque. If the graphics pixel is opaque, the graphics color information for that pixel is displayed on the screen. If the graphics pixel is transparent, the color information of the video pixel behind it is displayed on the screen.

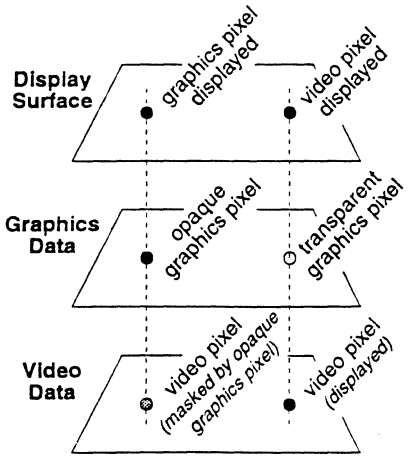


Figure 3-21. Graphics Overlay Operation

The graphics overlay controls determine which graphics pixels are transparent. The determination is made by a combination of three overlay control features — a TAG bit component in the video pixel data and the graphics COLOR key switch, and the SWKEY code that can be received from the video-input port on ZC[3:0]. The graphics COLOR key switch is generated by AND'ing the graphics pixel data with register GKMc and then comparing the results against register GCKc.

The tag bit is generated outside the MediaDAC and switches the multiplexer. Graphics Overlay OpCode register GOC is input to the multiplexer.

3.6.2.1 Graphics Overlay OpCode Register GOC

The graphics overlay opcode is an 8-bit value used as input to an 8:1 multiplexer. The select signals to the multiplexer (SWKEY, the TAG bit, and the graphics COLOR key match) determine which of the eight bits will become the transparency control for each pixel time. If a bit in register GOC is '1', the graphics pixel becomes transparent, enabling the video pixel for final display. Register GOC is initialized to '0x00' during reset, selecting the graphics path and ignoring the video input stream. This allows very sophisticated mixing of video and graphics, beyond simple windowing.

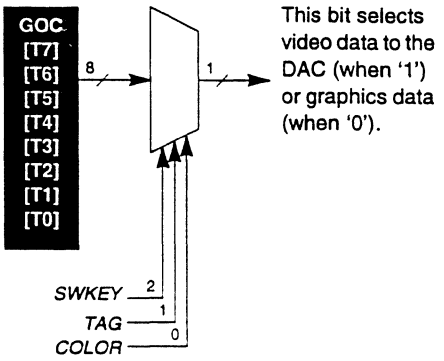


Figure 3-22. Graphics Overlay Opcode Register GOC

3.7 Monitor Interface Unit

3.7.1 6-Bit/8-Bit DAC Operation

Register ASC[D24] limits the resolution of the output DACs to 18 bits, resulting in a slight reduction in power consumption and provides software compatibility with many popular palette DACs.

For 8-bit operation, D0 is the LSb and D7 is the MSb of color data.

For 6-bit operation, color data is contained on the lower six bits of the data bus, with D0 being the LSb and D5 being the MSb of the color data. When writing color data, D6 and D7 are ignored. During color read cycles, D6 and D7 are a logical '0'. Accessing the cursor RAM array does not depend on the resolution of the DACs.

Note that in the 6-bit mode, the MediaDAC full-scale output current will be less than when it is in the 8-bit mode since the two LSbs of each 8-bit DAC are always a logical '0' in the 6-bit mode.

3.7.2 CLK Synchronizer and SYNC Alignment

The clock generator creates all device clocks. The rising edge of LCLK latches GSD[31:0] or VGA[7:0], and signals CBLANK*, HSIN, VSIN, GPS, O/E*, and BORDER*. The information latched by this signal is synchronized internally with SCLK. To avoid metastability, LCLK must maintain setup and hold requirements to SCLK. Data is synchronized with the selected pixel clock (PCLK0 or PCLK1) after being internally latched with SCLK. When the input data multiplexing rate is 8:1, 4:1, 2:1, or 1:1, LCLK must be the pixel clock divided by 8, 4, 2, or 1, respectively.

The sync alignment circuitry generates the external signals HSOUT and VSOUT required for the monitor. The output is delayed to match the internal PCLKn delay of the RGB outputs. Sync alignment register SAR programs polarities of signals HSIN, HSOUT, VSIN, and VSOUT. SAR also generates a programmable PCLKn delay of RGB relative to the matched HSOUT and VSOUT, as described above. This delay is programmable in both directions.

3.7.3 Power-Down Mode

The MediaDAC incorporates a power-down mode controlled by register bits ASC[COFF,DOFF]. While-COFF and DOFF are logical '0's, the MediaDAC functions normally.

- While COFF is a logical '0', all clock inputs, PCLKn, VCLK and LCLK, are disabled
- While DOFF is a logical '0', the DACs are turned off.

Note that the RAM still retains the data, and can be read or written to by the processor as long as the pixel clock is running. The RAM is enabled during processor read/write cycles and shuts down when the processor access is completed. The DACs output no current, and the command registers can still be written to or read by the processor.

NOTE: The output DACs require about 1 ms to turn off (sleep mode) or turn on (normal), depending on the compensation capacitor used.

3.7.4 Compensation Pin Connection

The compensation signal COMP (pin 73) is provided to ensure the stability and maximum resolution of the displayed image. COMP must be connected to DACVDD pin 69 through a series 0.1-uF capacitor. Lead lengths and PC-board conductors must be kept as short as possible. Please refer to Figure 10-3 for an application example.

3.8 Analog Video Output Signals

Table 3-12. Analog Output Operating Modes

Symbol	Parameter	Analog Output Modes[a]				
		RS343			IBM PS/2™	
		A	B	C	D (8-Bit)	E (6-Bit)
W_{lev}	White Level	26.7 mA 1.0 V	25.3 mA 0.95 V	17.6 mA 0.66 V	14.0 mA 0.70 V	14.0 mA 0.70 V
BK_{lev}	Black Level	9.04 mA 0.34 V	7.59 mA 0.28 V	0 mA 0 V	0 mA 0 V	0 mA 0 V
BL_{lev}	Blank Level	7.59 mA 0.28 V	7.59 mA 0.28 V	0 V	0 V	0 V
S_{lev}	Sync Level	0 mA 0 V	0 mA 0 V	—	—	—
I_{ref}	Reference Current	1.72 mA	1.72 mA	0.823 mA	0.679mA	0.660mA
R_{set}[b]	Reference Current Resistor Value	720 Ω	720 Ω	1.5 kΩ	1.87 kΩ	1.82 kΩ
L_{ped}	Luminance Pedestal	92.5 IRE	100 IRE	100 IRE	100 IRE	100 IRE
BL_{ped}	Blank Pedestal	7.5 IRE	—	—	—	—
S_{ped}	Sync Pedestal	40 IRE	40 IRE	—	—	—
	LSB Size	69 µA	69 µA	69 µA	55 µA	226 µA

[a] See Figure 3-23.
[b] Recommended values.

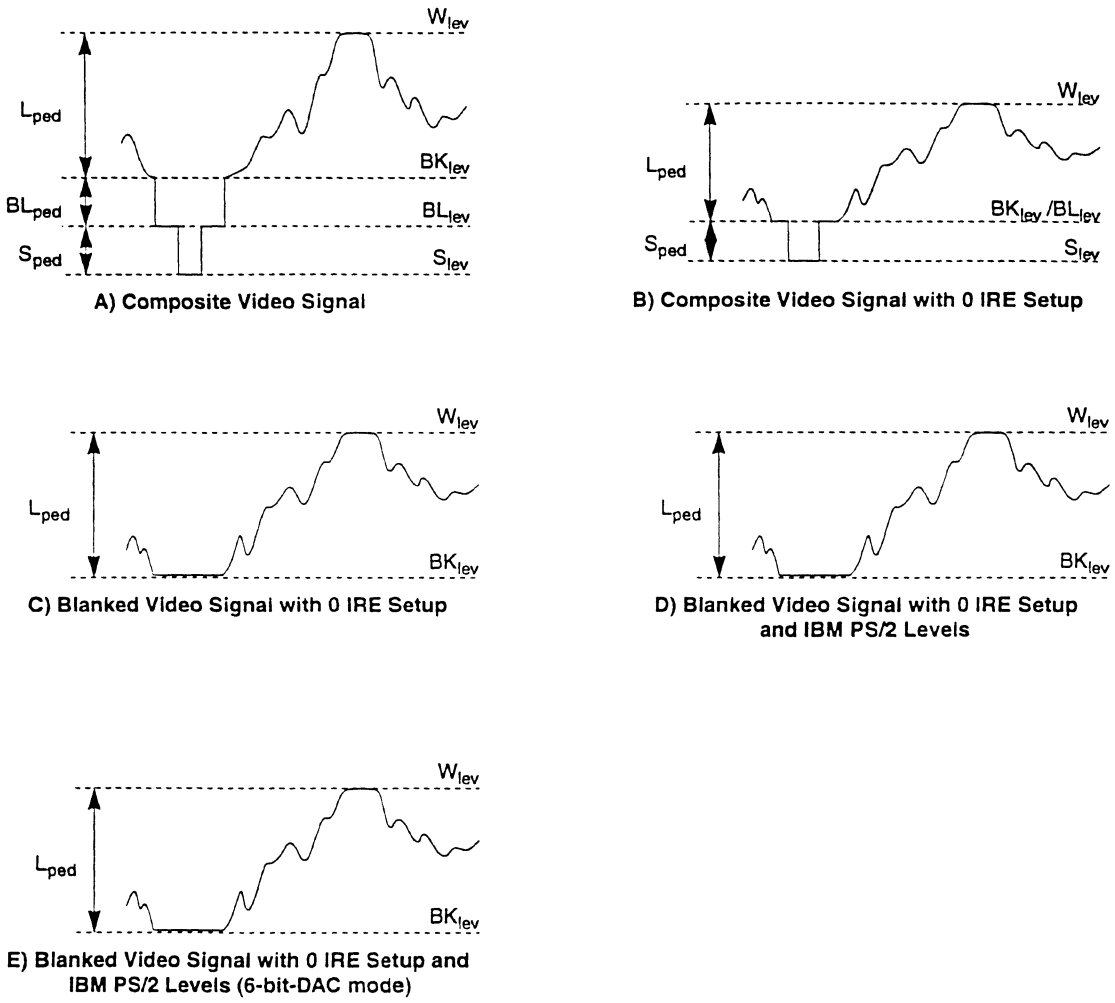


Figure 3-23. MediaDAC Analog Video Output Signals

4. REGISTERS

This section describes the internal registers that control MediaDAC™ operations. Table 4-1 shows the addresses for local hardware configuration; Table 4-2 shows the addresses for ISA and MCB configurations.

NOTES: 1) To maintain compatibility with future Pixel Semiconductor products, all reserved register bits must be written as '0'. Data values in reserved register locations are not guaranteed on readback.

2) Register names containing lowercase variables represent groups of registers with similar functions. Refer to the *Conventions* table on page 14 for a list of MediaDAC register variables.

3) Shaded areas are *only* applicable to the CL-PX2085.

4.1 MediaDAC Control Registers — Summary

4.1.1 Local Hardware Configuration

Table 4-1. MediaDAC™ Control Registers — Local Hardware Addresses

Register Definition	See Pg#	Mode A BS[1:0]=10 RS [4:0]	Mode B BS[1:0]=11 RS [3:0]	GCR [BLK]	Reg. Access	Definitions 7	6	5	4	3	2	1	0
Memory Access Addressing and Indexing													
BIR Block Index Register	56	N/A	N/A	N/A	N/A	IE	RE	SE	RO	RSVD	BLK		
Graphics Color Palette RAM Registers[a]													
LAW LUT Address Write	58	00h	0000	N/A	direct	WA[b]							
						P[c]	WA[c]						
LCD LUT Color Data	59	01h	0001	N/A	direct	D							
LPM LUT Pixel Mask	59	02h	0010	N/A	direct	M							
LAR LUT Address Read	59	03h	0011	N/A	direct	RA[b]							
						P[c]	RA[c]						
Cursor Color/Border RAM/Analog Setup Registers													
CAW Cursor Address Write	61	04h	0100	0000	direct	RSVD[d]						WA[d]	
						WA[e]							
CCD Cursor Color Data	61	05h	0101	0000	direct	D							
ASC Analog Setup Control	62	06h	0110	0000	direct	RSVD[f]	COFF	BPE	BS	GS	RS	D24	DOFF
						GCRE							
CAR Cursor Address Read	61	07h	0111	0000	direct	RSVD[d]						RA[d]	
						RA[e]							
Graphic and Cursor Setup Registers													
GFC Graphics Format Control	63	08h	1000	N/A	direct	RSVD	GPF		TC	CF	MR	TE	PS
CSC Cursor Setup Control	64	09h	1001	N/A	direct	SD	DPS		CS	DM	PM	CMS	
GSR[g] Graphics Status Register	64	0Ah	1010	N/A	direct	REV				S	IO	CP	
GCR[a] General Configuration Register	64	0Ah	1010	N/A	direct	BLK				2CLK	CU64	A9	A8
CPR Cursor Pattern RAM Data	66	0Bh	1011	N/A	direct	D							

Table 4-1. MediaDAC™ Control Registers — Local Hardware Addresses *(cont.)*

Register Definition	See Pg#	Mode A BS[1:0]=10 RS [4:0]	Mode B BS[1:0]=11 RS [3:0]	GCR [BLK]	Reg. Access	Definitions 7	6	5	4	3	2	1	0
Cursor Positioning Registers													
CXb Cursor X Position Registers													
CXL Cursor X Position, LSB	67	0Ch	1100	N/A	direct	X							
CXH Cursor X Position, MSB	67	0Dh	1101	N/A	direct	RSVD				X			
CYb Cursor Y Positions													
CYL Cursor Y Position, LSB	67	0Eh	1110	N/A	direct	Y							
CYH Cursor Y Position, MSB	67	0Fh	1111	N/A	direct	RSVD				Y			
Video, Graphics, and Sync Control Registers													
VFC Video Format Control	68	10h	0100	0100	indexed	EC	CS[f] / RSVD	RSVD	GBYP	CSF			
GOC Graphics Overlay Opcode	68	11h	0101	0100	indexed	T7	T6	T5	T4	T3	T2	T1	T0
SAR Sync Alignment Register	69	12h	0110	0100	indexed	RSVD			VL	DIR	DLY		
TEST Test Register	69	13h	1111	0100	indexed	GT	CCT	RSVD	PO	SO	WO[f]	RSVD[f]	BCE[f]
											RSVD		
Video Gamma Correction Palette RAM Registers													
VGW Video Gamma Address Write	71	14h	0100	0101	indexed	WA							
VGD Video Gamma Data	71	15h	0101	0101	indexed	D							
RSV2[f] Reserved 2[f]	—	16h	0110	0101	indexed	RSVD[f]							
VAFC VESA Advanced Feature Connector	72					RSVD	RSVD	RSVD	VM		V*	OE*	
VGR Video Gamma Address Read	71	17h	0111	0101	indexed	RA							
Graphics Chroma Key Registers													
GCKc Graphics Chroma Key Registers													
GCKR GCK Red	73	18h	0100	0110	indexed	CKR							
GCKG GCK Green	73	19h	0101	0110	indexed	CKG							
GCKB GCK Blue	73	1Ah	0110	0110	indexed	CKB							
RSV3 Reserved 3	—	1Bh	0110	0111	indirect	RSVD							
GKMc Graphics Chroma Key Mask													
GKMR GCK Mask Red	74	1Ch	0100	0111	indexed	MR							
GKMG GCK Mask Green	74	1Dh	0101	0111	indexed	MG							
GKMB GCK Mask Blue	74	1Eh	0110	0111	indexed	MB							
RSV4 Reserved 4	—	1Fh	0111	0111	indirect	RSVD							

a. The indexed registers are mapped to these addresses when ASC[GCRE] is enabled, and when BIR[BLK] contains the appropriate value.
b. Graphics Color Selection.
c. Cursor Color Selection.
d. 32 x 32 hardware cursor configuration.
e. 64 x 64 hardware cursor configuration.
f. CL-PX2080 only.
g. GSR and GCR share a register address. GSR is enabled when ASC[GCRE] = 0; GCR is enabled when ASC[GCRE] = 1.

4.1.2 ISA/MCB Configurations

Table 4-2. MediaDAC™ Control Registers — ISA/MCB Addresses

NOTE: ISA configuration is specified when signals BS[1:0] = 00;
MCB configuration is specified when signals BS[1:0] = 01.

Register Definition		See Pg#	Pri. Adr. BIR:SE=0	Sec. Adr. BIR:SE=1	BIR [BLK]	Bit Definitions 7	6	5	4	3	2	1	0
Memory Access Addressing and Indexing													
BIR	Block Index Register	56	0x27CE ALT=0 (BIR:SE N/A)	0x029E ALT=1 (BIR:SE N/A)	N/A	IE	RE	SE	RO	RSVD	BLK		
Graphics Color Palette RAM Registers[a]													
LAW	LUT Address Write	58	0x03C8[b] 0x27CC[c]	N/A[b] 0x029C[c]	N/A[b] 000c[c]	WA[d]							
						P[e]	WA[c]						
LCD	LUT Color Data	59	0x03C9[b] 0x27CD[c]	N/A[b] 0x029D[c]	N/A[b] 000c[c]	D							
LPM	LUT Pixel Mask	59	0x03C6[b] 0x27CA[c]	N/A[b] 0x029A[c]	N/A[b] 000c[c]	M							
LAR	LUT Address Read	59	0x03C7[b] 0x27CB[c]	N/A[b] 0x029B[c]	N/A[b] 000c[c]	RA[b]							
						P[c]	RA[c]						
Cursor Color/Border RAM / Analog Setup Registers													
CAW	Cursor Address Write	61	0x27CC	0x029C	001	RSVD[f]						WA[d]	
						WA[g]							
CCD	Cursor Color Data	61	0x27CD	0x029D	001	D							
ASC	Analog Setup Control	62	0x27CA	0x029A	001	RSVD[h] GCRE	COFF	BPE	BS	GS	RS	D24	DOFF
CAR	Cursor Address Read	61	0x27CB	0x029B	001	RSVD[d]						RA[d]	
						RA[e]							
Graphic and Cursor Setup Registers													
GFC	Graphics Format Control	63	0x27CC	0x029C	010	RSVD	GPF		TC	CF	MR	TE	PS
CSC	Cursor Setup Control	64	0x27CD	0x029D	010	SD	DPS		CS	DM	PM	CMS	
GSR[i]	Graphics Status Register	64	0x27CA	0x029A	010	REV				S	IO	CP	
GCR[a]	General Configuration Register	64	0x27CA	0x029A	010	BLK				2CLK	CU64	A9	A8
CPR	Cursor Pattern RAM Data	66	0x27CB	0x029B	010	D							
Cursor Positioning Registers													
CXb	*Cursor X Position Registers*												
CXL	Cursor X Position, LSB	67	0x27CC	0x029C	011	X							
CXH	Cursor X Position, MSB	67	0x27CD	0x029D	011	RSVD				X			
CYb	*Cursor Y Positions*												
CYL	Cursor Y Position, LSB	67	0x27CA	0x029A	011	Y							
CYH	Cursor Y Position, MSB	67	0x27CB	0x029B	011	RSVD				Y			

Table 4-2. MediaDAC™ Control Registers — ISA/MCB Addresses *(cont.)*

NOTE: ISA configuration is specified when signals BS[1:0] = 00;
MCB configuration is specified when signals BS[1:0] = 01.

Register Definition	See Pg#	Pri. Adr. BIR:SE=0	Sec. Adr. BIR:SE=1	BIR [BLK]	Bit Definitions 7	6	5	4	3	2	1	0
Video, Graphics, and Sync Control Registers												
VFC Video Format Control	68	0x27CC	0x029C	100	EC	CS[f] / RSVD	RSVD	GBYP	CSF			
GOC Graphics Overlay Opcode	68	0x27CD	0x029D	100	T7	T6	T5	T4	T3	T2	T1	T0
SAR Sync Alignment Register	69	0x27CA	0x029A	100	RSVD			VL	DIR	DLY		
TEST Test Register	69	0x27CB	0x029B	100	GT	CCT	RSVD	PO	SO	WO[f] / RSVD	RSVD[f] / RSVD	BCE[f] / RSVD
Video Gamma Correction Palette RAM Registers												
VGW Video Gamma Address Write	71	0x27CC	0x029C	101	WA							
VGD Video Gamma Data	71	0x27CD	0x029D	101	D							
RSV2[f] Reserved 2[f]	—	0x27CA	0x029A	101	RSVD[f]							
VAFC VAFC VESA Advanced Feature Connector	72				RSVD	RSVD	RSVD	VM		V*	OE*	
VGR Video Gamma Address Read	71	0x27CB	0x029B	101	RA							
Graphics Chroma Key Registers												
GCKc Graphics Chroma Key Registers												
GCKR GCK Red	73	0x27CC	0x029C	110	CKR							
GCKG GCK Green	73	0x27CD	0x029D	110	CKG							
GCKB GCK Blue	73	0x27CA	0x029A	110	CKB							
RSV3 Reserved 3	—	0x27CB	0x029B	110	RSVD							
GKMc Graphics Chroma Key Mask												
GKMR GCK Mask Red	74	0x27CC	0x029C	111	MR							
GKMG GCK Mask Green	74	0x27CD	0x029D	111	MG							
GKMB GCK Mask Blue	74	0x27CA	0x029A	111	MB							
RSV4 Reserved 4	—	0x27CB	0x029B	111	RSVD							

a. The indexed registers are mapped to these addresses when ASC[GCRE] is enabled, and when BIR[BLK] contains the appropriate value.
b. VGA Modes 0 and 1 only.
c. VGA Mode 2 only.
d. Graphics Color Selection.
e. Cursor Color Selection.
f. 32 x 32 hardware cursor configuration.
g. 64 x 64 hardware cursor configuration.
h. CL-PX2080 only.
i. GSR and GCR share a register address. GSR is enabled when ASC[GCRE] = 0; GCR is enabled when ASC[GCRE] = 1.

4.2 Memory Access Addressing and Indexing

4.2.1 BIR: Block Index Register

Register Definition		ISA: BS[1:0] = 00 MCB: BS[1:0] = 01 Pri. Adr. BIR:SE=0	ISA: BS[1:0] = 00 MCB: BS[1:0] = 01 Sec. Adr. BIR:SE=1	ISA: BS[1:0] = 00 MCB: BS[1:0] = 01 BIR [BLK]	Local (A) BS[1:0] = 10 RS [4:0]	Local (B) BS[1:0] = 11 RS [3:0]	Local (B) BS[1:0] = 11 GCR [BLK]	Local (B) BS[1:0] = 11 Reg. Access	7	6	5	4	3	2	1	0
BIR	Block Index Register	0x27CE ALT=0 (BIR:SE N/A)	0x029E ALT=1 (BIR:SE N/A)	N/A	N/A	N/A	N/A	N/A	IE	RE	SE	RO	RSVD	BLK		

BIR specifies the access modes for all MediaDAC™ control registers.

Bit #	Access	Reset	Description	
7	R/W	0	IE	**Index Enable.** Specifies direct (local hardware interface) or indexed addressing (ISA/MCB) for MediaDAC control registers. 0 Direct addressing; RS[4:0] overrides BIR[BLK] 1 Indexed addressing; I/O address specified by BIR[SE]
6	R/W	0	RE	**Read Access Enable.** Enables read/write access of VGA-compatible registers, as shown below. 0 Write-only access 1 Read/write access
5	R/W	0	SE	**Secondary Enable.** Specifies I/O address for MediaDAC registers. 0 Primary I/O address selected 1 Secondary I/O address selected
4	R/W	0	RO	**LAR Override.** Enables register LAR at I/O address 0x03C7 (VGA modes 0 and 1 only). 0 LAR disabled 1 LAR enabled
3	R/W	0	RSVD	**Reserved.** Read as '0'.
2:0	R/W	0	BLK	**Block Select.** Specifies which block of registers is to be read and written at 16-bit I/O addresses 0x27CA, 0x27CB, 0x27CC, 0x27CD .

VGA Mode	*IE*	*RE*	*RO*	*LPM*	*LAR*	*LAW*	*LCD*
0	0	0	x	W			
1a	0	1	0	R/W	W	R/W	R/W
1b	0	1	1	R/W			
2	1	x	x	R/W			

BLK	*Registers*	*27CA*	*27CB*	*27CC*	*27CD*
000	Graphics Color Palette RAM	LPM	LAR	LAW	LCD
001	Cursor Color/Border RAM / Analog Setup	ASC	CAR	CAW	CCD
010	Graphic and Cursor Setup	GSR	CPR	GFC	CSC
011	Cursor Positioning	CYL	CYH	CXL	CXH
100	Video, Graphics, and Sync Control	SAR	TEST	VFC	GOC
101	Video Gamma Correction Palette RAM	VAFC	VGR	VGW	VGD
110	Graphics Chroma Key	GCKB	RSV3	GCKR	GCKG
111	Graphics Chroma Key Mask	GKMB	RSV4	GKMR	GKMG

4.3 Graphics Color Palette RAM Registers

The Graphics Color Palette RAM is a 256 x 24-bit color LUT that is modified using data register LCD and write and read address registers LAW and LAR, as shown below. See also Section 3.5.1.

Writes:

The first byte of data from the host bus specifies the initial Graphics Color Palette RAM address to which color data will be written. This address is written to register LAW. The next three bytes of data contain R, G, and B color information, which is written into register LCD, then to the RAM.

When the blue component is written, the address stored in LAW auto-increments by 1 to point to the next RAM location, where LCD writes the next R, G, B components, etc. The process continues until all RAM locations are filled, or until the host writes a new address to LAW (resetting the RAM address and causing writes to continue from the new location).

Reads:

The host bus specifies the initial RAM address from which color data will be read. This address is written to register LAR.

Register LCD reads the R, G, and B color information from the specified RAM location and returns it to the host. When the blue component is read, the address stored in LAR auto-increments by 1 to point to the next RAM location, where LCD reads the next R, G, B components, etc. The process continues until all RAM locations are read, or until the host writes a new address to LAR (resetting the RAM address and causing the RGB reads to continue from the new location).

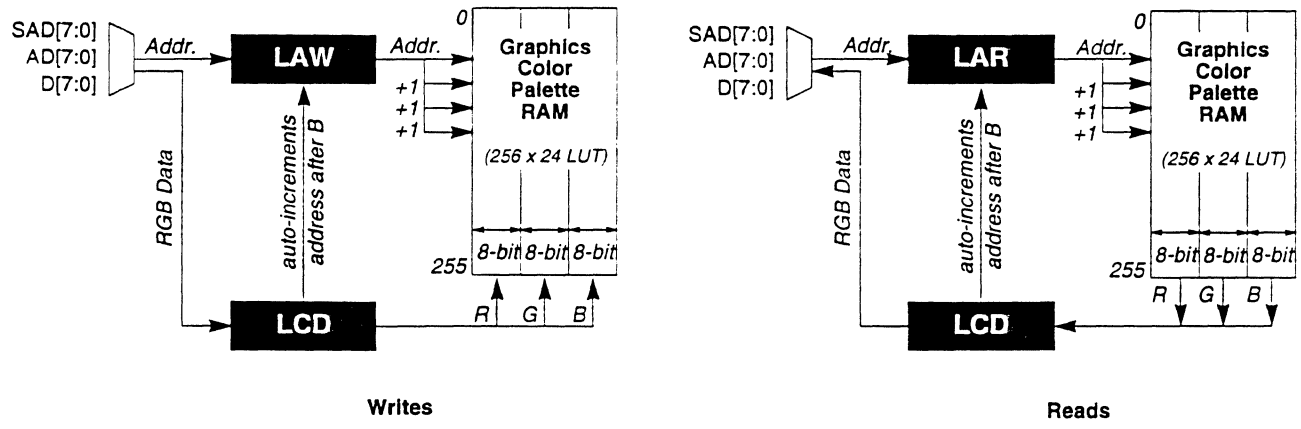


Figure 4-1. Accessing the Graphics Color Palette RAM

Register Definition	ISA: BS[1:0] = 00 MCB: BS[1:0] = 01 Pri. Adr. BIR:SE=0	Sec. Adr. BIR:SE=1	BIR [BLK]	Local (A) BS[1:0] = 10 RS [4:0]	Local (B) BS[1:0] = 11 RS [3:0]	GCR [BLK]	Reg. Access	7	6	5	4	3	2	1	0
LAW[a] LUT Address Write	0x03C8[b]	N/A[b]	N/A[b]	00h	N/A	0000	direct	WA[d]							
	0x27CC[c]	0x029C[c]	000[c]					P[e]	WA[c]						
LCD[a] LUT Color Data	0x03C9[b]	N/A[b]	N/A[b]	01h	N/A	0001	direct	D							
	0x27CD[c]	0x029D[c]	000[c]												
LPM[a] LUT Pixel Mask	0x03C6[b]	N/A[b]	N/A[b]	02h	N/A	0010	direct	M							
	0x27CA[c]	0x029A[c]	000[c]												
LAR[a] LUT Address Read	0x03C7[b]	N/A[b]	N/A[b]	03h	N/A	0011	direct	RA[b]							
	0x27CB[c]	0x029B[c]	000[c]					P[c]	RA[c]						

a. The indexed registers are mapped to these addresses when ASC[GCRE] is enabled, and when BIR[BLK] contains the appropriate value.
b. VGA Modes 0 and 1 only.
c. VGA Mode 2 only.
d. Graphics Color Selection.
e. Cursor Color Selection.

4.3.1 LAW: LUT Address Write

Write address register LAW has two functions:

- **Graphics Color Selection.** LAW addresses the Graphics Color Palette RAM when the host system writes 24- or 18-bit color data through register LCD(See Section 4.3 for more information)
- **Cursor Pattern Selection.** LAW addresses the Cursor Pattern RAM when the host system writes cursor pattern data through CPR (See Section 4.6.5 for more information)

Bit #	Access	Reset	Description
Graphics Color Selection			
7:0	R/W	0h	WA **Write Address for Graphics Color Palette RAM.**
Cursor Pattern Selection			
7	R/W	0	P **Plane of Cursor Pattern RAM Data to be addressed (32x32 cursor).** 0 Plane 0 1 Plane 1
6:0	R/W	0h	WA **Write Address for Cursor Pattern RAM Data.** **NOTE:** Byte 0 is located in the upper-left corner of plane 0.

4.3.2 LCD: LUT Color Data

LCD is an 8-bit-wide port from the host system to the Graphics Color Palette RAM. See Section 4.3 for a complete description of Graphics Color Palette RAM addressing.

Bit #	Access	Reset	Description	
7:0	R/W	0h	D	**Data for Graphics Color Palette RAM.** **NOTE:** For 18-bit color (specified by register ASC[D24]), valid data is on bits 5:0 of the data bus for both reads and writes.

4.3.3 LPM: LUT Pixel Mask

The graphics pixel data is used to look up color information in the Graphics Color Palette, and can be masked before the lookup occurs. LPM masks the address. The graphics pixel is logically AND'ed with the LPM data, and the result is used to address the Graphics Color Palette RAM. See also Table 3-7, page 39.

Bit #	Access	Reset	Description	
7:0	R/W	FFh	M	**Data for Graphics Color Palette RAM Pixel Mask.**

4.3.4 LAR: LUT Address Read

Read address register LAR has two functions:

- **Graphics Color Selection.** LAR addresses the Graphics Color Palette RAM when the host system reads 24- or 18-bit color data through register LCD (See Section 4.3 for more information)
- **Cursor Pattern Selection.** LAR addresses the Cursor Pattern RAM when the host system reads cursor pattern data through register CPR (See Section 4.6.5 for more information)

Bit #	Access	Reset	Description	
Graphics Color Selection				
7:0	R/W	0h	RA	**Read Address for Graphics Color Palette RAM.**
Cursor Pattern Selection				
7	R/W	0	P	**Plane of Cursor Pattern RAM Data to be addressed (64 x 64 cursor).** 0 Plane 0 1 Plane 1
6:0	R/W	0h	RA	**Read Address for Cursor Pattern RAM Data.** **NOTE:** Byte 0 is located in the upper-left corner of plane 0.

4.4 Cursor Color/Border RAM Registers

The Cursor Color/Border RAM is a 4 x 24-bit color LUT. Its four locations — Border (BDR), Cursor Color 1 (CC1), Cursor Color 2 (CC2), and Cursor Color 3 (CC3) — are accessed using data register CCD and write and read address registers CAW and CAR, as shown below. See also Section 3.5.1.

Writes:

The first byte of data from the host bus specifies the initial Cursor Color/Border RAM address to which color data will be written. This address is written to register CAW. The next three bytes of data contain R, G, and B color information. This data is written into register CCD, then to the RAM.

When the blue component is written, the address stored in CAW auto-increments by 1 to point to the next RAM location, where CCD writes the next R, G, B components, etc. The process continues until all RAM locations are filled, or until the host writes a new address to CAW (resetting the RAM address and causing writes to continue from the new location).

Reads:

The host bus specifies the initial RAM address from which color data will be read. This address is written to register CAR.

Register CCD reads the R, G, and B color information from the specified RAM location and returns it to the host. When the blue component is read, the address stored in CAR auto-increments by 1 to point to the next RAM location, where CCD reads the next R, G, B components, etc. The process continues until all RAM locations are read, or until the host writes a new address to CAR (resetting the RAM address and causing reads to continue from the new location).

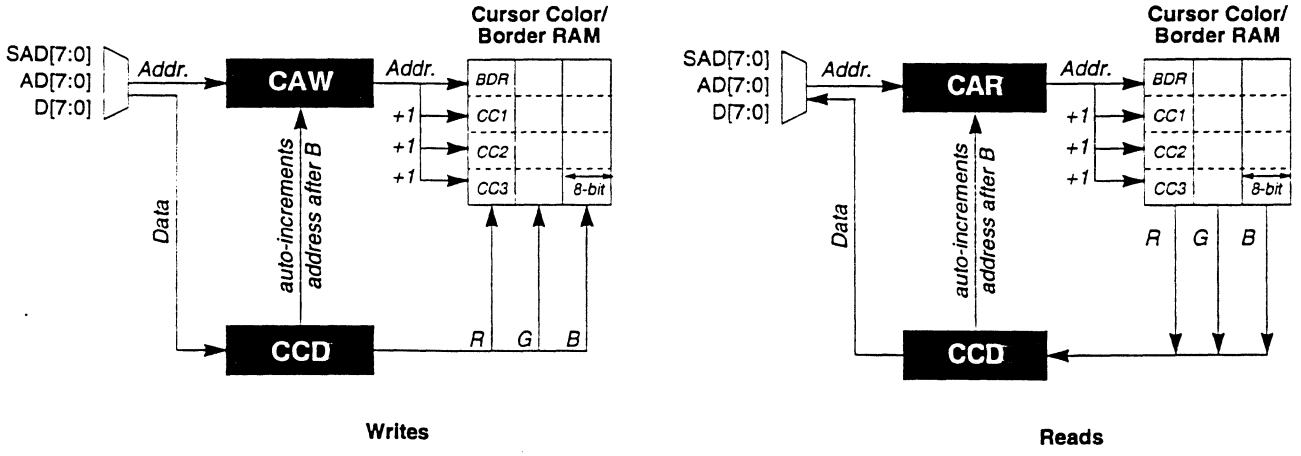


Figure 4-2. Accessing the Cursor Color/Border RAM

Register Definition	ISA: BS[1:0] = 00 MCB: BS[1:0] = 01 Prl. Adr. BIR:SE=0	Sec. Adr. BIR:SE=1	BIR [BLK]	Local (A) BS[1:0] = 10 RS [4:0]	Local (B) BS[1:0] = 11 RS [3:0]	GCR [BLK]	Reg. Access	f = 32x32 hardware cursor configuration g = 64x64 hardware cursor configuration 7	6	5	4	3	2	1	0
CAW Cursor Address Write	0x27CC	0x029C	001	04h	0000	0100	direct	RSVD[d]						WA[d]	
								WA[e]							
CCD Cursor Color Data	0x27CD	0x029D	001	05h	0000	0101	direct	D							
CAR Cursor Address Read	0x27CB	0x029B	001	07h	0000	0111	direct	RSVD[d]						RA[d]	
								RA[e]							

4.4.1 CAW: Cursor Address Write

Write address register CAW addresses the Cursor Color/Border RAM when the host system writes 24-bit cursor or border color data through register CCD. See Section 4.4 for more information.

Bit #	Access	Reset	Description	
7:0	R/W	00	WA	**Write Address for Cursor Color/Border RAM.** XXXX XX00 BDR XXXX XX01 CC1 XXXX XX10 CC2 XXXX XX11 CC3

4.4.2 CCD: Cursor Color Data

Register CCD is an 8-bit-wide port from the host system to the Cursor Color/Border RAM. See Section 4.4 for more information.

Bit #	Access	Reset	Description	
7	R/W	0	D	**Data for Cursor Color/Border RAM.** **NOTE:** For 18-bit color (specified by register ASC, bit D24), valid data is on bits 5:0 of the data bus for both reads and writes.

4.4.3 CAR: Cursor Address Read

Read address register CAR addresses the Cursor Color/Border RAM when the host system reads 24-bit cursor or border color data through register CCD. See Section 4.4 for more information.

Bit #	Access	Reset	Description	
7:0	R/W	00	RA	**Read Address for Cursor Color/Border RAM.** XXXX XX00 BDR XXXX XX01 CC1 XXXX XX10 CC2 XXXX XX11 CC3

4.5 Analog Setup Register

4.5.1 ASC: Analog Setup Control

Register Definition	ISA: BS[1:0] = 00 MCB: BS[1:0] = 01 Pri. Adr. BIR:SE=0	Sec. Adr. BIR:SE=1	BIR [BLK]	Local (A) BS[1:0] = 10 RS [4:0]	Local (B) BS[1:0] = 11 RS [3:0]	GCR [BLK]	Reg. Access	7	6	5	4	3	2	1	0
ASC Analog Setup Control	0x27CA	0x029A	001	06h	0000	0110	direct	RSVD GCRE	COFF	BPE	BS	GS	RS	D24	DOFF

Register ASC sets up the DAC and analog output for the MediaDAC™.

NOTE: All I/O registers and the Cursor Color Data can be R/W accessed when the internal clock is off. The Cursor Pattern RAM and Graphics Color Palette RAM maintain their integrity. Reads from register CSC or any I/O register do not affect the state of the internal clock.

Bit #	Access	Reset		Description
7	R/W	0	GCRE	**GCR Enable.** 0 Register GCR disabled 1 Register GCR enabled
6	R/W	0	COFF	**Clock Off.** 0 Normal clocking 1 Powers down the MediaDAC; forces all clocks high When DOFF also = 1, internal clock forced high
5	R/W	0	BPE	**Blanking Pedestal Enable.**
4	R/W	0	BS	**Blue Sync.**
3	R/W	0	GS	**Green Sync.**
2	R/W	0	RS	**Red Sync.**
1	R/W	0	D24	**Specifies DAC data as 18- or 24-bit.** 0 18-bit DAC 1 24-bit DAC
0	R/W	0	DOFF	**DAC Off** 0 Normal operation 1 Disables DAC output When COFF also = 1, internal clock forced high

4.6 Graphic and Cursor Setup Registers

4.6.1 GFC: Graphics Format Control

	ISA: BS[1:0] = 00 MCB: BS[1:0] = 01			Local (A) BS[1:0] = 10	Local (B) BS[1:0] = 11									
Register Definition	Pri. Adr. BIR:SE=0	Sec. Adr. BIR:SE=1	BIR [BLK]	RS [4:0]	RS [3:0]	GCR [BLK]	Reg. Access	7	6	5	4	3	2	1	0
GFC Graphics Format Control	0x27CC	0x029C	010	08h	N/A	1000	direct	RSVD	GPF		TC	CF	MR	TE	PS

Register GFC sets up the graphics interface timing and color format controls.

NOTE: PORTSEL is defined in the GPS table, page 24.

Bit #	Access	Reset	Description
7	R/W	0	RSVD **Reserved.** Write as '0'.
6:5	R/W	00	GPF **Graphics Port Format.** Specifies data type for GSD[31:0]/VGA[7:0]; works with PORTSEL and GFC[MR] to specify SCLK. 00 24-bit pixel 8:8:8 RGB data (PCLKn:LCLK = 1:1) When PORTSEL = 1, SCLK = PCLKn 01 16-bit pixel data (multiplex ratios set by bit TE) When PORTSEL = 1 and GFC[MR] = 0, SCLK = PCLKn/2 When PORTSEL = 1 and GFC[MR] = 1, SCLK = PCLKn 10 Four 8-bit pixels (PCLKn:LCLK = 4:1) When PORTSEL = 1, SCLK = PCLKn/4 11 Eight 4-bit pixels (PCLKn:LCLK = 8:1) When PORTSEL = 1, SCLK = PCLKn/8 XX When PORTSEL = 0, SCLK = PCLKn
4	R/W	0	TC **True Color Graphics Color Palette RAM bypass.** 0 Pixel data is pseudocolor When PORTSEL = 1, GSD[31:0] data addresses palette 1 Pixel data is true color When PORTSEL = 1, GSD[31:0] data bypasses palette X When PORTSEL = 0, VGA[7:0] data addresses palette
3	R/W	0	CF **Color Format.** Selects GSD[31:0] RGB color format. 0 5:5:5 1 5:6:5
2	R/W	0	MR **Multiplexing Rate.** Controls 16-bit graphics port data multiplexing rate. 0 PCLKn:LCLK ratio = 2:1 (two-pixel bus) 1 PCLKn:LCLK ratio = 1:1 (one-pixel bus)
1	R/W	0	TE **Tag Enable.** Specifies pixel selector for 5:5:5 RGB graphics data. 0 GFC[PS] is pixel selector 1 GSD[31] is video pixel selector
0	R/W	0	PS **Pixel Selector.** Selects graphics pixel for 5:6:5 and 5:5:5 RGB 1:1 multiplex data. 0 Graphics pixel 0 (graphics data on port GSD[15:0]) 1 Graphics pixel 1 (graphics data on port GSD[31:16])

4.6.2 CSC: Cursor Setup Control

Register Definition	ISA: BS[1:0] = 00 / MCB: BS[1:0] = 01 Pri. Adr. BIR:SE=0	Sec. Adr. BIR:SE=1	BIR [BLK]	Local (A) BS[1:0] = 10 RS [4:0]	Local (B) BS[1:0] = 11 RS [3:0]	GCR [BLK]	Reg. Access	7	6	5	4	3	2	1	0
CSC Cursor Setup Control	0x27CD	0x029D	010	09h	N/A	1001	direct	SD	DPS			CS	DM	PM	CMS		

Register CSC sets cursor modes and controls Graphics Color Palette RAM indexing for 16-bit graphics data modes and clock selections.

Bit #	Access	Reset	Description	
7	R/W	0	SD	**SCLK Disable.** Enables or disables signal SCLK. 0 SCLK enabled 1 SCLK disabled
6:5	R/W	00	DPS	**Data Port Select.** (See also signal GPS, page 24). **NOTE:** Signals GSD[31:0] and DOD[31:0] share pins. 00 *GSD[31:0]* disabled; *VGA[7:0]* enabled; *DOD[31:0]* disabled 01 *GSD[31:0]* enabled *(when GPS=1)*; *VGA[7:0]* enabled *(when GPS=0)*; *DOD[31:0]* disabled 10 *GSD[31:0]* enabled; *VGA[7:0]* disabled; *DOD[31:0]* disabled 11 *GSD[31:0]* disabled; *VGA[7:0]* enabled; *DOD[31:0]* enabled
4	R/W	0	CS	**Pixel Clock Select.** 0 PCLK0 1 PCLK1
3	R/W	0	DM	**Display Mode.** 0 Progressive-scan display 1 Interlaced scan display
2	R/W	0	PM	**Palette Mapping** for 16-bit graphics data to 18-bit DAC. 0 Color components mapped to MSb's of Graphics Color Palette 1 Color components mapped to LSb's of Graphics Color Palette **NOTE:** All unused bits are padded with '0's.
1:0	R/W	00	CMS	**Cursor Mode Select.** 00 Cursor disabled 01 Three-color cursor 10 Two-color cursor with highlighting 11 Two-color cursor

4.6.3 GSR: Graphics Status Register

Register Definition	ISA: BS[1:0] = 00 / MCB: BS[1:0] = 01 Pri. Adr. BIR:SE=0	Sec. Adr. BIR:SE=1	BIR [BLK]	Local (A) BS[1:0] = 10 RS [4:0]	Local (B) BS[1:0] = 11 RS [3:0]	GCR [BLK]	Reg. Access	7	6	5	4	3	2	1	0
GSR[a] Graphics Status Register	0x27CA	0x029A	010	0Ah	N/A	1010	direct	REV				S	IO	CP			

a. GSR and GCR share a register address. GSR is enabled when ASC[GCRE] = 0; GCR is enabled when ASC[GCRE] = 1.

Register GSR is a read-only, modulo-3 counter that sets up and monitors the MediaDAC™ revisions and I/O cycles. It shares an address with general configuration register GCR.

Bit #	Access	Reset	Description
GSR: Graphics Status Register (enabled when register ASC, bit GCRE = 0)			
7:4	R	0h	REV **MediaDAC Revision level.**
3	R	0	S **Sense bit status.** 0 All analog outputs are below the 335-mV level 1 One or more analog outputs have exceeded the 335-mV level
2	R	0	IO **I/O read/write access status.** 0 I/O write: LAW, CAW, or VGW has been specified 1 I/O read: LAR, CAR, or VGR has been specified
1:0	R	00	CP **Color Pointer.** Specifies color component for host to access on next I/O cycle to a palette. 00 Red 01 Green 10 Blue

4.6.4 GCR: General Configuration Register

	ISA: BS[1:0] = 00 MCB: BS[1:0] = 01			*Local (A) BS[1:0] = 10*	*Local (B) BS[1:0] = 11*									
Register Definition	**Prl. Adr. *BIR:SE=0***	**Sec. Adr. *BIR:SE=1***	**BIR [BLK]**	**RS [4:0]**	**RS [3:0]**	**GCR [BLK]**	**Reg. Access**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
GCR[a] General Configuration Register	0x27CA	0x029A	010	0Ah	N/A	1010	direct	BLK				2CLK	CU64	A9	A8

a. GSR and GCR share a register address. GSR is enabled when ASC[GCRE] = 0; GCR is enabled when ASC[GCRE] = 1.

GCR: General Configuration Register (enabled when register ASC, bit GCRE = 1)				
7:4	R/W	0h	BLK	**Index address** to extended registers (local hardware interface).
3	R/W	0h	2CLK	**Clock Doubler Enable.** 0 clock doubler disabled; MediaDAC operates at external clock frequency (default at power-up) 1 clock doubler enabled
2	R/W	0h	CU64	**64 x 64 Hardware Cursor Enable.** 0 32 x 32 cursor enabled (default at power-up) 1 64 x 64 cursor enabled
1	R/W	0h	A9	**Address 9.** Specifies the bit plane to be addressed in Cursor Pattern RAM. 0 Plane 0 1 Plane 1 **A9, A8.** MSb's for 64 x 64 cursor address. Must be loaded before LAR/LAW is written when 64 x 64 cursor is enabled (GCR[CU64] = 1)
0	R/W	0h	A8	**Address 8.**

4.6.5 CPR: Cursor Pattern RAM Data

Register Definition	ISA: BS[1:0] = 00 MCB: BS[1:0] = 01 Pri. Adr. BIR:SE=0	ISA: BS[1:0] = 00 MCB: BS[1:0] = 01 Sec. Adr. BIR:SE=1	ISA: BS[1:0] = 00 MCB: BS[1:0] = 01 BIR [BLK]	Local (A) BS[1:0] = 10 RS [4:0]	Local (B) BS[1:0] = 11 RS [3:0]	Local (B) BS[1:0] = 11 GCR [BLK]	Local (B) BS[1:0] = 11 Reg. Access	7	6	5	4	3	2	1	0
CPR Cursor Pattern RAM Data	0x27CB	0x029B	010	0Bh	N/A	1011	direct	D							

Register CPR is an 8-bit-wide port from the host system to the Cursor Pattern RAM.

NOTE: The Cursor Pattern RAM is a 1K-byte memory array comprising two 64 x 64-bit (512-byte) planes. (It can also be specified as two 32 x 32-bit, or 128-byte, planes by GCR:CU64.) This RAM is modified using data register CPR, and write and read registers LAW and LAR, as shown below.

Bit #	Access	Reset	Description
7:0	R/W	0h	D **Data** for Cursor Color/Border RAM.

Writes:

The address written to register LAW specifies the initial Cursor Pattern RAM address to which cursor pattern data will be written. Data is written into register CPR, then to the RAM. Following each write, the address stored in LAW auto-increments by 1 to point to the next RAM location, where CPR writes the next byte, etc. The process continues until all RAM locations are filled, or until the host writes a new address to LAW (resetting the RAM address, and causing writes to continue from the new location).

Reads:

The host bus writes to LAR the initial Cursor Pattern RAM address from which cursor pattern data will be read. CPR reads the data from the specified RAM location and returns it to the host. Following each read, the address stored in LAR auto-increments by 1 to point to the next RAM location, where CPR reads the next byte, etc. The process continues until all RAM locations are read, or until the host writes a new address to LAR, resetting the RAM address and causing reads to continue from the new location).

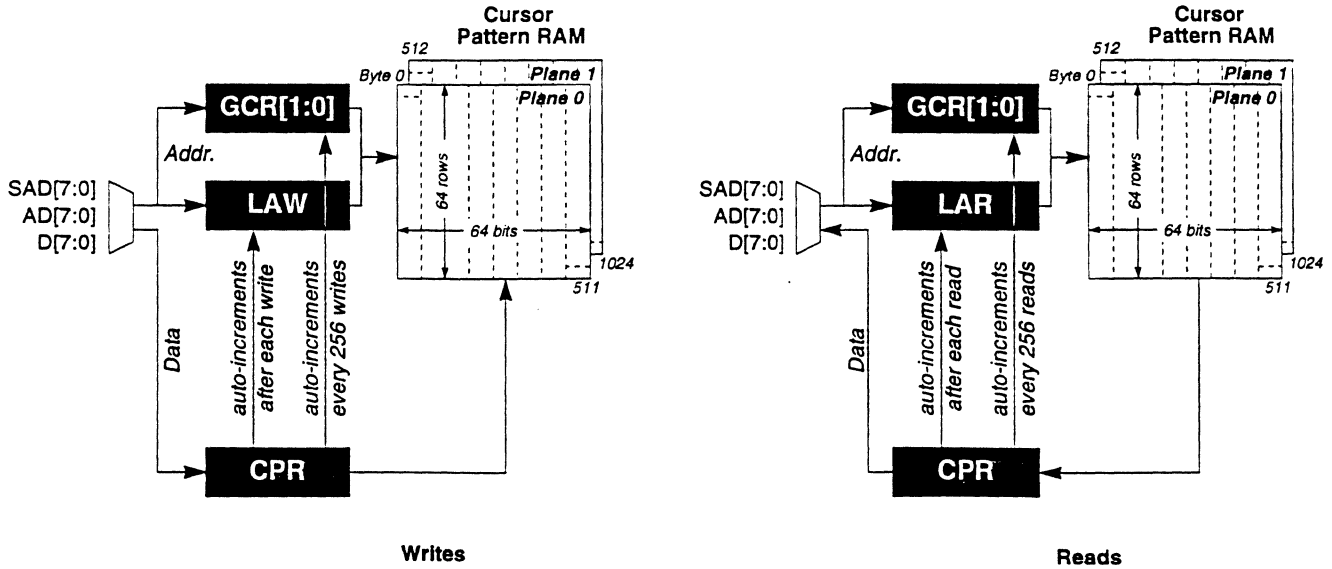


Figure 4-3. Accessing the Cursor Pattern RAM

4.7 Cursor Positioning Registers

Register Definition		ISA: BS[1:0] = 00 MCB: BS[1:0] = 01 Pri. Adr. BIR:SE=0	ISA: BS[1:0] = 00 MCB: BS[1:0] = 01 Sec. Adr. BIR:SE=1	ISA: BS[1:0] = 00 MCB: BS[1:0] = 01 BIR [BLK]	Local (A) BS[1:0] = 10 RS [4:0]	Local (B) BS[1:0] = 11 RS [3:0]	Local (B) BS[1:0] = 11 GCR [BLK]	Local (B) BS[1:0] = 11 Reg. Access	7	6	5	4	3	2	1	0
CXb	*Cursor X Position*															
CXL	Cursor X Position, LSB	0x27CC	0x029C	011	0Ch	N/A	1100	direct	X							
CXH	Cursor X Position, MSB	0x27CD	0x029D	011	0Dh	N/A	1101	direct	RSVD				X			
CYb	*Cursor Y Position*															
CYL	Cursor Y Position, LSB	0x27CA	0x029A	011	0Eh	N/A	1110	direct	Y							
CYH	Cursor Y Position, MSB	0x27CB	0x029B	011	0Fh	N/A	1111	direct	RSVD				Y			

4.7.1 CXb: Cursor X Position

Registers CXb specify the X position of the bottom-right corner of the cursor.

NOTE: During reset, when CXb are set to 0h, the cursor is positioned in the upper-left corner, offscreen. When CXL = 20h and CXH = 0h, the left column of cursor pixels is positioned at the left column of display-screen pixels.

Bit #	Access	Reset	Description	
CXL: Cursor X Position, LSB				
7:0	R/W	0	X	**Cursor X position, Low Byte.**
CXH: Cursor X Position, MSB				
7:4	R/W	0	RSVD	**Reserved.** Read as '0'.
3:0	R/W	0	X	**Cursor X position, High Byte.**

4.7.2 CYb: Cursor Y Position

Registers CYb specify the Y position of the bottom-right corner of the cursor relative to the top of the display screen.

NOTE: During reset, when CXb are set to 0h, the cursor is positioned in the upper-left corner, offscreen. When CYL = 20h and CYH = 0h, the top row of cursor pixels is positioned at the top row of display-screen pixels.

Bit #	Access	Reset	Description	
CYL: Cursor Y Position, LSB				
7:0	R/W	0	Y	**Cursor Y position, Low Byte.**
CYH: Cursor Y Position, MSB				
7:4	R/W	0	RSVD	**Reserved.** Read as '0'.
3:0	R/W	0	Y	**Cursor Y position, High Byte.**

4.8 Video, Graphics, and Sync Control Registers

Register Definition	ISA: BS[1:0] = 00 MCB: BS[1:0] = 01 Pri. Adr. BIR:SE=0	Sec. Adr. BIR:SE=1	BIR [BLK]	Local (A) BS[1:0] = 10 RS [4:0]	Local (B) BS[1:0] = 11 RS [3:0]	GCR [BLK]	Reg. Access	7	6	5	4	3	2	1	0
VFC Video Format Control	0x27CC	0x029C	100	10h	0100	0100	Indirect	EC	CS[f]	RSVD	GBYP	CSF			
									RSVD						
GOC Graphics Overlay Opcode	0x27CD	0x029D	100	11h	0100	0101	Indirect	T7	T6	T5	T4	T3	T2	T1	T0
SAR Sync Alignment Register	0x27CA	0x029A	100	12h	0100	0110	Indirect	RSVD			VL	DIR	DLY		
TEST Test Register	0x27CB	0x029B	100	13h	0100	0100	Indirect	GT	CCT	RSVD	PO	SO	WO[f]	RSVD[f]	BCE[f]
													RSVD		

4.8.1 VFC: Video Format Control

Register VFC sets up the video port interface timing and color format controls.

Bit #	Access	Reset	Description	
7	R/W	0	EC	**Extended Color mode for VGA data.** 0 Normal mode 1 Extended color mode
6	R/W	0	RSVD	**Reserved.** Read as '0'.
5	R/W	0	RSVD	**Reserved.** Read as '0'.
4	R/W	0	GBYP	**Gamma Correction Bypass.** 0 Gamma enabled 1 Gamma disabled
3:0	R/W	0h	CSF	**Color-Space Format.** Specifies color-space format of VSD[31:0]. 0000 YCrCb 4:2:2, non-tagged 0001 Y(U:T)(V:T) 4:(2):(2); LSb of U and V are tag data 1000 RGB 5:6:5 1010 RGB 5:5:5 1011 TRGB 1:5:5:5 (MSb = tag data) 1110 RGB 8:8:8, non-tagged 1111 TRGB 1:8:8:8 (bit 31= tag bit) XXXX All other bit configurations are reserved

4.8.2 GOC: Graphics Overlay Opcode

GOC contains an 8-bit value that inputs to an 8:1 multiplexer. Three select signals determine which of the eight bits become the transparency control for each pixel time: SWKEY = 1 when MediaDAC signals ZC[3:0] = 1100b; TAG = 1 when input data is tagged (tagged formats only); COLOR = 1 when data color matches contents of the graphics chroma-key and mask registers GCKc and GKMc.

Bit #	Access	Reset	Description	
7	R/W	0h	T7	111*
6	R/W	0h	T6	110*
5	R/W	0h	T5	101*
4	R/W	0h	T4	100*
3	R/W	0h	T3	011*
2	R/W	0h	T2	010*
1	R/W	0h	T1	001*
0	R/W	0h	T0	000*

Transparency controls.
1 video pixel enabled; graphics transparent
0 graphics pixel opaque; video pixel not visible

*Bit outputs from the multiplexer and becomes the transparency control for the current pixel time when SWKEY:TAG:COLOR = this value.

4.8.3 SAR: Sync Alignment Register

Register SAR programs outgoing sync signals to match the monitor, both in polarity and PCLKn delay, relative to the DAC outputs and internal pixel pipeline. SAR also sets the output sync signals VSOUT and HSOUT to align with the DAC RGB outputs in PCLKn time units.

Bit #	Access	Reset	Description	
7:5	R/W	0h	RSVD	**Reserved.** Read as '0'.
4	R/W	0	VL	**Horizontal and vertical sync output polarity in LCD mode.** 0 GSD24, GSD25 = active low 1 GSD24, GSD25 = active high
3	R/W	0	DIR	**Delay direction of HSOUT, VSOUT relative to R, G, B.** 0 Sync signals behind DAC outputs 1 Sync signals ahead of DAC outputs
2:0	R/W	000	DLY	**PCLKn delay** 000 No delay 001 PCLKn difference 010 2PCLKn difference 011 3PCLKn difference 100 4PCLKn difference 101 5PCLKn difference 110 6PCLKn difference 111 7PCLKn difference

4.8.4 TEST: Test Register

Bit #	Access	Reset	Description	
7	R/W	0	GT	**Graphics Test.** Graphics data (GSD[31:0] or VGA[7:0]) is output to VSD[31:0] .
6	R/W	0	CCT	**Cursor Counter Test.** Also bypasses cursor memory.
5	R/W	0	RSVD	**Reserved.** Write as '0'.
4	R/W	0	PO	**PCLKn off.**
3	R/W	0	SO	**Sense off** (active low).
2:0	R/W	0	RSVD	**Reserved.** Write as '0'.

4.9 Video Gamma Correction Palette RAM Registers

The Video Gamma Correction Palette RAM is a 256 x 24-bit color LUT that maps non-linear video pixel data to linear color data. It is modified using data register VGD and write and read registers VGW and VGR, as shown below. See also Section 3.3.4.

Writes:

The first byte of data from the host bus specifies the initial Video Gamma Correction Palette RAM address to which color data will be written. This address is written to register VGW. The next three bytes of data contain R, G, and B color information. This data is written into register VGD, then to the RAM.

When the blue component is written, the address stored in VGW auto-increments by 1 to point to the next RAM location, where VGD writes the next R, G, B components, etc. The process continues until all RAM locations are filled, or until the host writes a new address to VGW (resetting the RAM address and causing writes to continue from the new location).

Reads:

The host bus specifies the initial RAM address from which color data will be read. This address is written to register VGR.

Register VGD reads the R, G, and B color information from the specified RAM location, then returns it to the host. When the blue component is read, the address stored in VGR auto-increments by 1 to point to the next RAM location, where VGD reads the next R, G, B components, etc. The process continues until all RAM locations are read, or until the host writes a new address to VGR (resetting the RAM address and causing reads to continue from the new location).

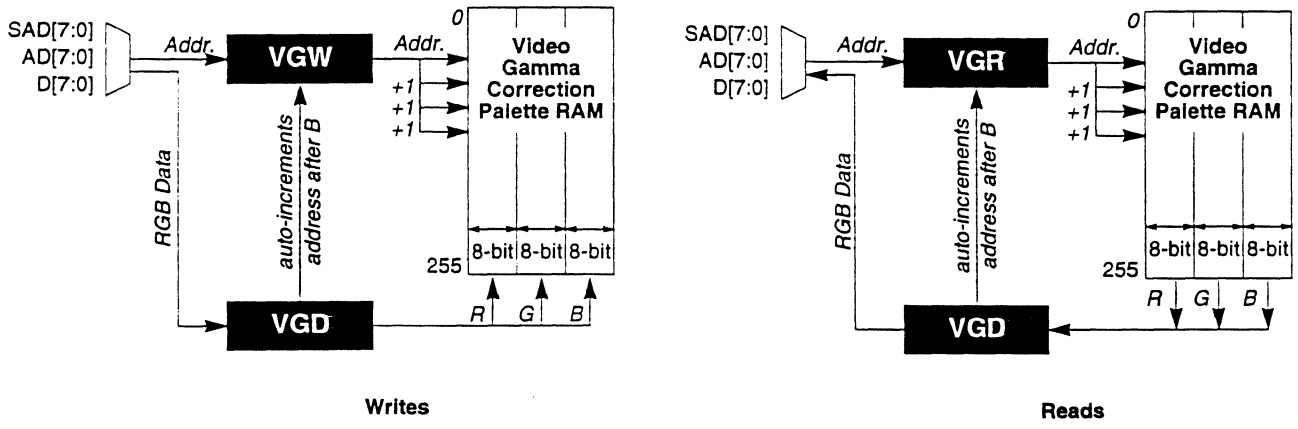


Figure 4-4. Accessing the Video Gamma Correction Palette RAM

Register Definition	ISA: BS[1:0] = 00 MCB: BS[1:0] = 01 Pri. Adr. BIR:SE=0	Sec. Adr. BIR:SE=1	BIR [BLK]	Local (A) BS[1:0] = 10 RS [4:0]	Local (B) BS[1:0] = 11 RS [3:0]	GCR [BLK]	Reg. Access	7	6	5	4	3	2	1	0
VGW Video Gamma Address Write	0x27CC	0x029C	101	14h	0101	0100	indirect	WA							
VGD Video Gamma Data	0x27CD	0x029D	101	15h	0101	0101	indirect	D							
VGR Video Gamma Address Read	0x27CB	0x029B	101	17h	0101	0111	indirect	RA							

4.9.1 VGW: Video Gamma Address Write

Write address register VGW addresses the Video Gamma Correction Palette RAM when the host system writes 24-bit color data through register VGD. See Section 4.9 for more information.

Bit #	Access	Reset	Description	
7:0	R/W	0h	WA	**Write Address for Video Gamma Correction Palette RAM.**

4.9.2 VGD: Video Gamma Data

Register VGD is an 8-bit-wide port from the host system to the Video Gamma Correction Palette RAM. The host system writes data to the RAM through VGD using VGW, and reads data from the RAM through VGD using VGR. See Section 4.9 for more information.

Bit #	Access	Reset	Description	
7:0	R/W	0	D	**Data for Video Gamma Correction Palette RAM Data.** **NOTE:** For 18-bit color (specified by register ASC, bit D24), valid data is on bits 5:0 of data bus for reads and writes.

4.9.3 VGR: Video Gamma Address Read

Read address register VGR addresses the Video Gamma Correction Palette RAM when the host system reads 24-bit color data through register VGD. See Section 4.9 for more information.

Bit #	Access	Reset	Description	
7:0	R/W	0h	RA	**Read Address for Video Gamma Correction Palette RAM.**

4.9.4 VAFC™ Register: VESA® Advanced Feature Connector

	ISA: BS[1:0] = 00 MCB: BS[1:0] = 01			Local (A) BS[1:0] = 10	Local (B) BS[1:0] = 11									
Register Definition	Prt. Adr. BIR:SE=0	Sec. Adr. BIR:SE=1	BIR [BLK]	RS [4:0]	RS [3:0]	GCR [BLK]	Reg. Access	7	6	5	4	3	2	1	0
VAFC VESA Advanced Feature Connector	0x27CA	0x029A	101	16h	0101	0110	Indirect	RSVD		RSVD	RSVD	VM		V*	OE*

Bit #	Access	Reset	Description
7:6	R/W	0h	RSVD **Reserved.** Read as '0'
5	R/W	0h	ZGT2 **Zoom Greater-Than 2.** Enhances image quality when using interpolated zoom factors greater than 2. 0 Normal zoom operation 1 Enhanced operation at zoom factors > 2[a]
4	R/W	0h	RSVD **Reserved.** Read as '0'.
3:2	R/W	0h	VM **VAFC Mode.** 00 16 B1x RGB 01 16 B2x RGB 10 32-bit mode 1x 11 32-bit mode 2x
1	R/W	0h	V* **VAFC Mode Enable.** 0 standard mode 1 better interp zoom over 2x zoom factors
0	R/W	0h	OE* **VAFC Output Mode Enable.** 0 enable 1 disable

VM[1:0]	*V**	*VFC[3:0]*	*Input CLK*	*Video Input Format Over VAVI Port*
xx	1	0000	PCLK/2	32-bit YUV422
xx	1	0001	PCLK/2	32-bit YUV422T
xx	1	1000	PCLK/2	32-bit RGB 565
xx	1	1010	PCLK/2	32-bit RGB 555
xx	1	1011	PCLK/2	32-bit RGB 555T
xx	1	1110	PCLK	24-bit RGB 888
xx	1	1111	PCLK	24-bit RGB 888T
10	0	0000	PCLK/2	32-bit 1x YCrCb 422
11	0	0000	PCLK/4	32-bit 2x YCrC b422
00	0	1000	PCLK	16-bit 1x RGB 565
01	0	1000	PCLK/2	16-bit 2x RGB 565
10	0	1000	PCLK/2	32-bit 1x RGB 565
11	0	1000	PCLK/4	32-bit 2x RGB 565

a. This feature is not implemented in Revision A devices.

4.10 Graphics Chroma Key Registers

Register Definition	ISA: BS[1:0] = 00 MCB: BS[1:0] = 01 Pri. Adr. BIR:SE=0	ISA: BS[1:0] = 00 MCB: BS[1:0] = 01 Sec. Adr. BIR:SE=1	ISA: BS[1:0] = 00 MCB: BS[1:0] = 01 BIR [BLK]	Local (A) BS[1:0] = 10 RS [4:0]	Local (B) BS[1:0] = 11 RS [3:0]	GCR [BLK]	Reg. Access	7	6	5	4	3	2	1	0
GCKc Graphics Chroma Key															
GCKR GCK Red	0x27CC	0x029C	110	18h	0110	0100	indexed	CKR							
GCKG GCK Green	0x27CD	0x029D	110	19h	0110	0101	indexed	CKG							
GCKB GCK Blue	0x27CA	0x029A	110	1Ah	0110	0110	indexed	CKB							
GKMc Graphics Chroma Key Mask															
GKMR GCK Mask Red	0x27CC	0x029C	111	1Ch	0111	0100	indirect	MR							
GKMG GCK Mask Green	0x27CD	0x029D	111	1Dh	0111	0101	indirect	MG							
GKMB GCK Mask Blue	0x27CA	0x029A	111	1Eh	0111	0110	indirect	MB							

4.10.1 GCKc: Graphics Chroma Key

Registers GCKR, GCKG, and GCKB control the Graphics Chroma Key function.

- When graphics data inputs on VGA[7:0], the data in registers GCKc is compared against four adjacent graphics pixels. GCKR is compared to the least-significant pixel.
- When graphics data inputs on GSD[31:0], the data in registers GCKc is used as specified by the PCLKn:SCLK ratio.

Bit #	Access	Reset	Description	
GCKR: GCK Red				
7:0	R/W	0	CKR	**Chroma Key Red Data.**
GCKG: GCK Green				
7:0	R/W	0	CKG	**Chroma Key Green Data.**
GCKB: GCK Blue				
7:0	R/W	0	CKB	**Chroma Key Blue Data.**

4.10.2 GKMc: Graphics Chroma Key Mask

Registers GKMR, GKMG, and GKMB control the Graphics Chroma Key Mask function.

- When graphics data inputs on VGA[7:0], the data in registers GKMc is AND'ed with four adjacent graphics pixels. GKMR is compared to the least-significant pixel.
- When graphics data inputs on GSD[31:0], the data in registers GKMc is used as specified by the PCLKn:SCLK ratio.

Bit #	Access	Reset	Description	
GKMR: GCK Mask Red				
7:0	R/W	1	MR	**Graphics Key Mask Red Data.**
GKMG: GCK Mask Green				
7:0	R/W	1	MG	**Graphics Key Mask Green Data.**
GKMB: GCK Mask Blue				
7:0	R/W	1	MB	**Graphics Key Mask Blue Data.**

5. ELECTRICAL SPECIFICATIONS

5.1 Electrical Specifications — CL-PX2080

5.1.1 Absolute Maximum Ratings

This section lists the absolute maximum ratings of the MediaDAC. Stresses above those listed can cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods can affect device reliability.

Ambient temperature under bias ..... 0° to + 70° C

Storage temperature ..... -65° to +150° C

Voltage on any pin with respect to ground ..... GND -0.5 to V_{CC} + 0.5 V

Power supply voltage ..... 7 V

Lead temperature (10 seconds) ..... 300° C

5.1.2 DC Specifications (Digital)

Symbol	Parameter	MIN	MAX	Conditions
V_{DD}	Power supply voltage	4.75 V	5.25 V	Normal Operation
V_{IL}	Input low voltage	0 V	0.8 V	
V_{IH}	Input high voltage	2.0 V	V_{DD} + 0.8 V	
V_{OL}	Output low voltage		0.4 V	I_{OL} = 4 mA
V_{OH}	Output high voltage	2.4 V		I_{OH} = 400 µA
V_{ILC}	Input low-voltage CMOS		0.8 V	
V_{IHC}	Input high-holtage CMOS	3.0 V		
V_{OLC}	Output low-voltage CMOS		0.4 V	I_{OLC} = 3.2 mA
V_{OHC}	Output high-voltage CMOS	3.5 V		I_{OHC} = -200 µA
I_{DD}	Digital supply current		300 mA	V_{DD} Nominal
I_{DDT}	Total supply current		450 mA	Note 1
I_L	Input leakage	-10 µA	10 µA	$0 < V_{IN} < V_{DD}$
C_{IN}	Input capacitance		10 pF	
C_{OUT}	Output capacitance		10 pF	

NOTES: 1) I_{DDT} is the sum of I_{DD} + $DACI_{DD}$ + $CLKI_{DD}$.
2) DACVSS must not exceed V_{DD}.

5.1.3 DC Specifications (RAMDAC)

(V_{DD} = 5 V ± 5%, T_A = 0° to 70° C, unless otherwise specified)

Symbol	Parameter	MIN	TYP	MAX	Conditions
DACVDD	Power supply voltage	4.75 V	5.0 V	5.25 V	Normal operation
$DACI_{DD}$	DAC supply current		45 mA	60 mA	Note 4

NOTES: 1) I_{REF} outside the specified limits may cause the analog outputs to become invalid.
2) The dotclock must be stable for a period of 100 µs after power-up before proper DAC operation is guaranteed.
3) See the I_{REF} description in Section 3.
4) $DACI_{DD}$ is specified with the three analog outputs (R,G,B) each loaded with 50 Ω.

5.1.4 DAC Characteristics

Test conditions generate PS/2 video-signal levels unless otherwise specified. Recommended test conditions are R_{set} = 1.87 kΩ, V_{ref} = 1.235 V, and I_{ref} = 0.679 mA. The parameters in the following table apply over the full voltage and temperature ranges specified in Sections 5.2.2 and 5.2.3.

V_{DD} = 5 V ± 5%, T_A = 0° to 70° C, unless otherwise specified.

Symbol	Parameter	MIN	TYP	MAX	Conditions
R	Resolution	8 bits			
I_{Omax}	Output current			-27 mA	V_O < 1 V
Co	Output capacitance			12 pF	
DT	DAC-to-DAC variability			± 5%	Notes 1, 2
DNL	Differential nonlinearity			± 1.0 ISB	
	White level relative to black	13.3 mA	14.0 mA	14.7 mA	
	Blank pedestal	1.25 mA	1.45 mA	1.65 mA	
	Sync pedestal	6.8 mA	7.59 mA	8.3 mA	
	Sense trip level	330 mV	400 mV	470 mV	
	Internal voltage reference output	1.11 V	1.235 V	1.35 V	

NOTES: 1) Outputs loaded identically.
2) Midpoint of the distribution of the three DACs measured at full-scale deflection.

5.1.5 AC Characteristics/Timing Information

This section includes system timing requirements for the MediaDAC. Timings are provided in nanoseconds (ns), at TTL input levels, with the ambient temperature varying from 0° C to 70° C, and V_{DD} varying from 4.75 to 5.25 VDC.

NOTES: 1) All timings assume a load of 50 pF.
2) TTL signals are measured at TTL threshold; CMOS signals are measured at CMOS threshold.

5.1.5.1 Index of Timing Information

5.1.5.2 Video Port Timing

Ref.	Parameter		MIN	MAX
t_1	Setup	ZC, EVIDEO*, VWE, VSD stable to VCLK rising edge	3 ns	
t_2	Hold	ZC, EVIDEO*, VWE, VSD stable after VCLK rising edge	3 ns	

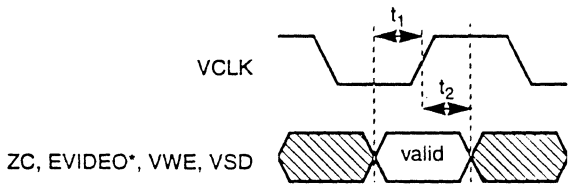


Figure 5-1. Video Port Timing

5.1.5.3 ISA Bus Timing

Ref.	Parameter		MIN	MAX
t_1	Setup	Valid address to IOR*/IOW* active	25 ns	
t_2	Delay	IOR*/IOW* active to DEN* active, DDIR change		20 ns
t_3	Delay	IOR* active to data output Low-Z	5 ns	25 ns
t_4	Delay	IOR* active to data out valid		40 ns
t_5	Pulse width	IOR*/IOW*	50 ns	
t_6	Delay	IOR*/IOW* inactive to DEN* inactive, DDIR changing		20 ns
t_7	Delay	IOR* inactive to three-state delay		15 ns
t_8	Hold	Address from IOR*/IOW* active	0 ns	
t_9	Setup	Data valid to IOW* inactive	20 ns	
t_{10}	Hold	IOW* inactive to data invalid	0 ns	
t_{11}	Delay	IOW* inactive to next IOW* or IOR* active	50 ns	
t_{12}	Setup	AEN falling edge to IOW* or IOR* active	10 ns	
t_{13}	Delay	IOW* or IOR* inactive to AEN rising edge	10 ns	

NOTE: The low-byte address-buffer enable must be qualified by IOR* to avoid data contention. Also, AEN must be low during cycle.

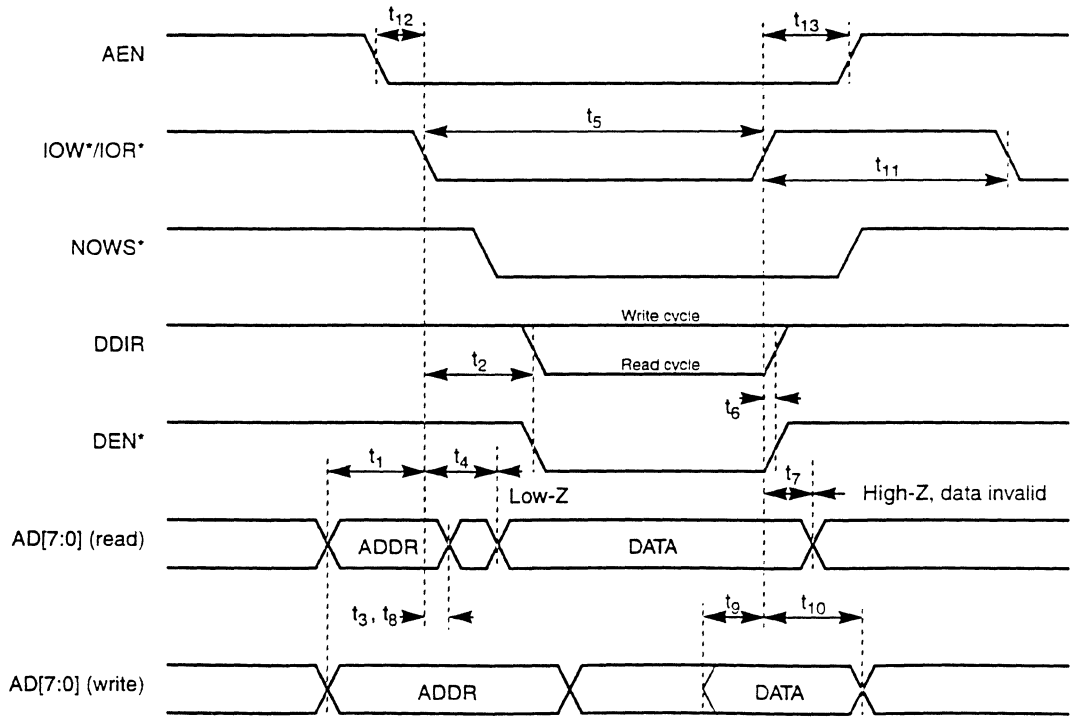


Figure 5-2. ISA Bus Timing

5.1.5.4 MCB Bus Timing

Ref.	Parameter		MIN	MAX
t_1	Setup	Address valid to CMD* active	25 ns	
t_2	Delay	CMD* active to DEN* active		20 ns
t_{3a}	Hold	Address valid from CMD* active	0 ns	
t_{3b}	Delay	CMD* active to data output Low-Z	5 ns	25 ns
t_4	Delay	CMD* active to read data valid		40 ns
t_5	Pulse width	CMD*	50 ns	
t_6	Delay	CMD* inactive to data invalid, High-Z		15 ns
t_7	Setup	Write data valid to CMD* inactive	20 ns	
t_8	Hold	Write data valid to CMD* inactive	0 ns	
t_9	Delay	CMD* inactive to next CMD* active	50 ns	
t_{10}	Setup	Status valid to CMD* active	25 ns	
t_{11}	Hold	CMD* active to status invalid	5 ns	
t_{12}	Delay	CMD* inactive to DEN* inactive, DDIR change		20 ns

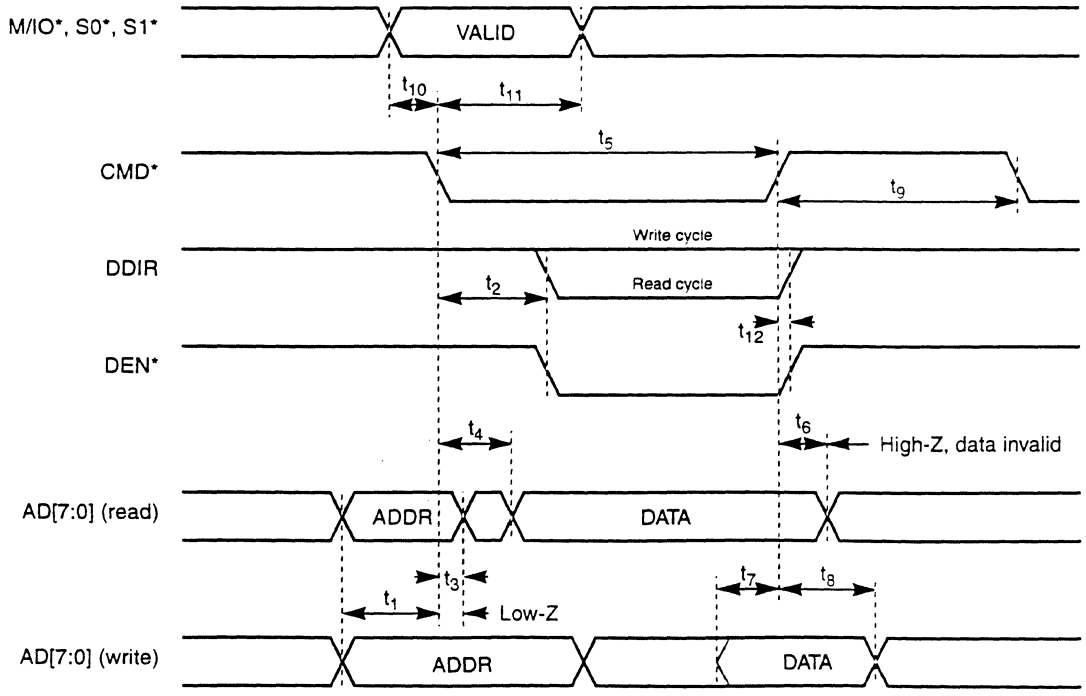


Figure 5-3. MCB Bus Timing

5.1.5.5 Local Hardware Interface Timing

Ref.	Parameter		MIN	MAX
t_1	Setup	Valid address to IOR*/IOW* active	25 ns	
t_2	Delay	IOR*/IOW* active to DEN* active, DDIR change		20 ns
t_3	Delay	IOR* active to data out Low-Z	5 ns	25 ns
t_4	Delay	IOR* active to data out valid		40 ns
t_5	Pulse width	IOR*/IOW*	50 ns	
t_6	Delay	IOR*/IOW* inactive to DEN* inactive, DDIR changing		20 ns
t_7	Delay	IOR* inactive to data High-Z		15 ns
t_8	Hold	From IOR*/IOW* active to Address invalid	0 ns	
t_9	Setup	Data valid to IOW* inactive	20 ns	
t_{10}	Hold	IOW* inactive to data invalid	0 ns	
t_{11}	Delay	IOW* inactive to next IOW* or IOR* active	50 ns	
t_{12}	Setup	CS* falling edge to IOW* or IOR* active	10 ns	
t_{13}	Delay	IOW* or IOR* inactive to CS* inactive	10 ns	

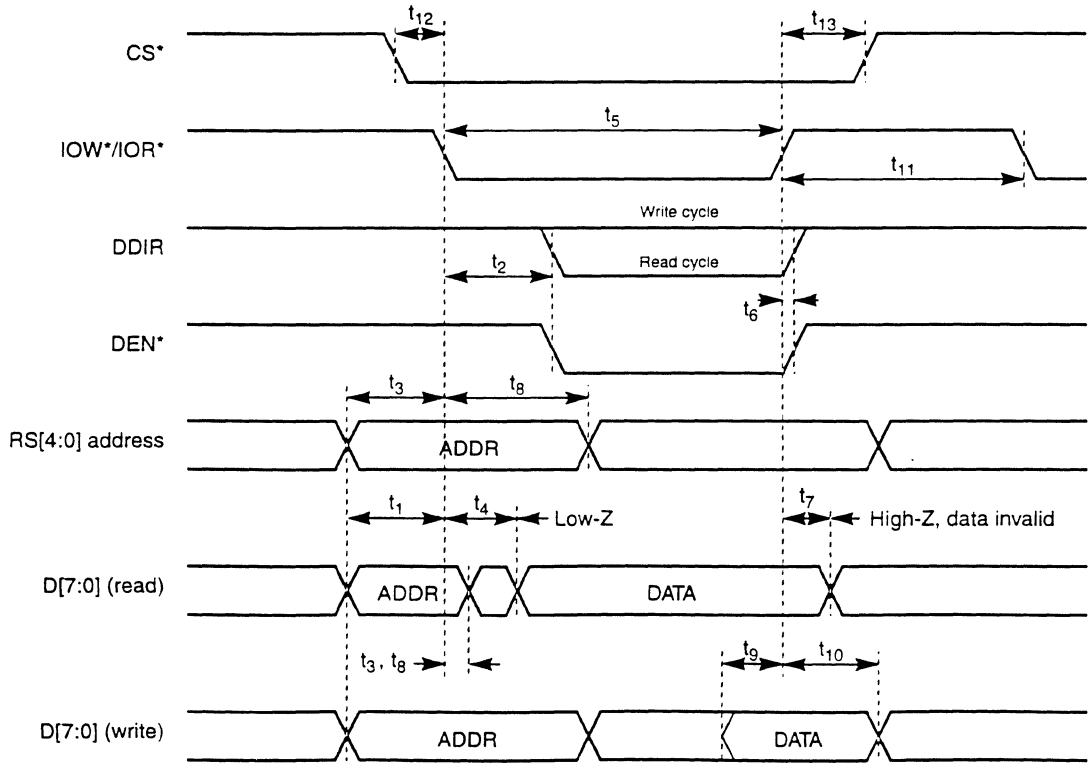


Figure 5-4. Local Hardware Interface Timing

5.1.5.6 Clocks as Inputs (85 MHz)

Ref.	Parameter		85 MHz MIN	85 MHz MAX	135 MHz MIN	135 MHz MAX
t_1	Rise time	PCLKn		3 ns		3 ns
		LCLK		3 ns		3 ns
		VCLK		3 ns		3 ns
t_2	Fall time	PCLKn		3 ns		3 ns
		LCLK		3 ns		3 ns
		VCLK		3 ns		3 ns
t_3	High period	PCLKn	4 ns		4 ns	
		LCLK	4 ns		4 ns	
		VCLK	5 ns		5 ns	
t_4	Low period	PCLKn	4 ns		4 ns	
		LCLK	4 ns		4 ns	
		VCLK	5 ns		5 ns	
t_5	Cycle time	PCLKn	11.5 ns		11.5 ns	
		LCLK	11.5 ns		11.5 ns	
		VCLK	14.8 ns		14.8 ns	

NOTE: LCLK and SCLK cycle and pulse width times are multiplied by 2, 4, and 8 in 2:1, 4:1, and 8:1 multiplexing modes, respectively.

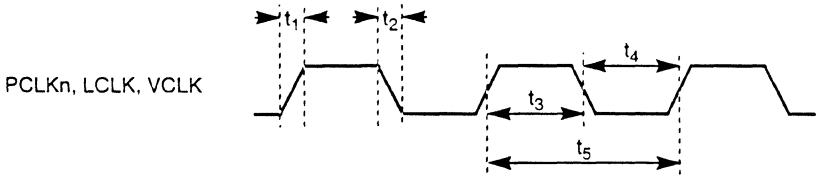


Figure 5-5. PCLKn, LCLK, and VCLK as Inputs (LCLK = 1:1 Multiplex Rate)

5.1.5.7 Sync, RGB, and GSD[23:0] Output Timing from PCLKn

Ref.	Parameter		MIN	MAX
t_1	Delay	PCLKn rise to RGB output		30 ns
t_2	Rise/Fall	RGB output	3 ns	
t_3	Settling Time	RGB output full-scale		15 ns
t_4	Delay	PCLKn rise to VSOUT, HSOUT output	2 ns	10 ns
t_5	Delay	PCLKn rise to GSD[23:0] output	2 ns	10 ns
(not shown)	Delay	RGB output to SENSE* output		1 µs

NOTES: 1) Output delay is measured from the 50% point of the rising edge of PCLKn to the 50% point of full-scale transition.

2) Settling time is measured from the 50% point of full-scale transition to the output remaining within ± 1 LSB.

3) Output rise/fall time is measured between the 10% and 90% points of full-scale transition.

4) In 1:1 multiplexing mode, RGB data is clocked out directly from LCLK, and synchronizing with SCLK and PCLKn is unnecessary and bypassed. All timing PCLKn references in the table above apply to LCLK when in 1:1 multiplexing mode.

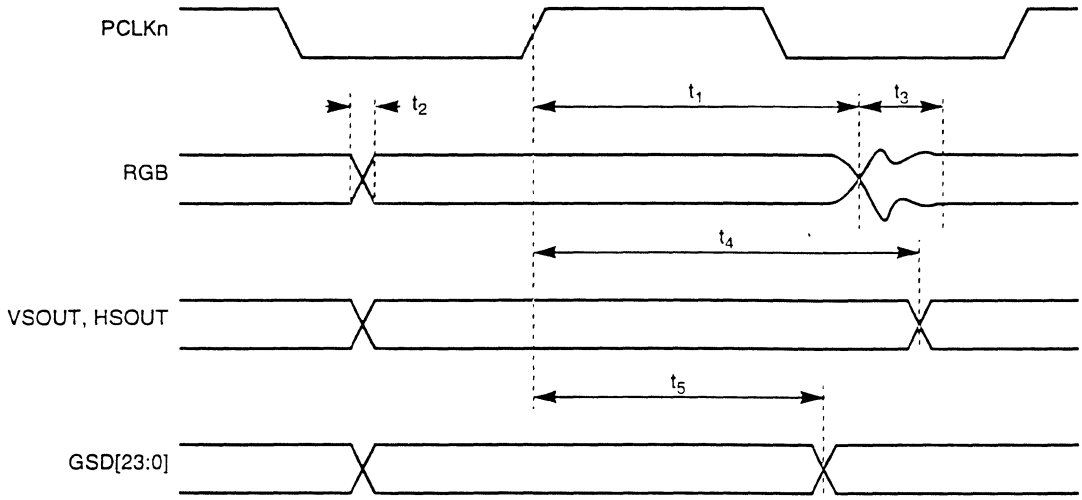


Figure 5-6. Sync, RGB, and GSD[23:0] Output Timing from PCLKn

5.1.5.8 Graphics Port Input Timing

Ref.	Parameter		MIN	MAX
t_1	Delay	PCLKn rise to PCO, SCLK output (2:1, 4:1, 8:1 mode)	2 ns	10 ns
t_2	Delay	LCLK rising edge to PCO, SCLK rising edge output (1:1 mode)	2 ns	10 ns
t_3	Setup	LCLK stable to PCLKn rising edge	3 ns	
t_4	Hold	LCLK stable after PCLKn rising edge	3 ns	
t_5	Setup	Graphics data, control to LCLK rising edge	1 ns	
t_6	Hold	Graphics data, control to LCLK rising edge	5 ns	

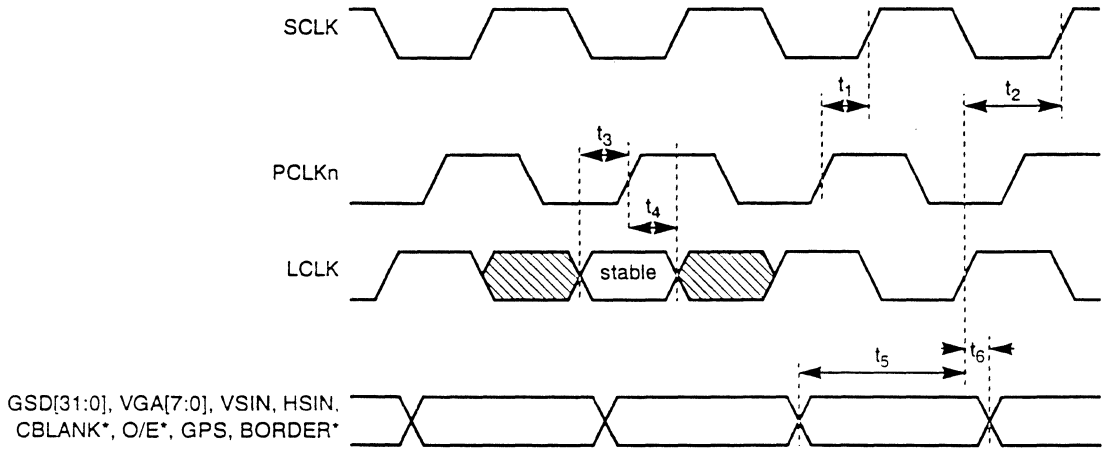


Figure 5-7. Graphics Port Input Timing

5.1.5.9 VGA Port Interface Timing, EC Mode

Ref.	Parameter		MIN	MAX
t_1	Setup	MSB data valid to falling edge PCLKn	1 ns	
t_2	Hold	LCLK falling edge to MSB data	5 ns	
t_3	Setup	LSB data and graphics controls valid to rising edge PCLKn	1 ns	
t_4	Hold	LCLK rising edge to LSB data, graphics controls invalid	5 ns	
t_5	Pulse Width	LCLK low	12 ns	
t_6	Pulse Width	LCLK high	12 ns	
t_7	Period	LCLK	30 ns	
t_8	Delay	LCLK rising edge to RGB output		15 ns
t_9	Delay	Valid data to full-scale RGB output		5 ns
t_{10}	Delay	RGB out to SENSE* valid	0 µs	1 µs
t_{11}	Delay	Pipeline		32 PCLKn cycles

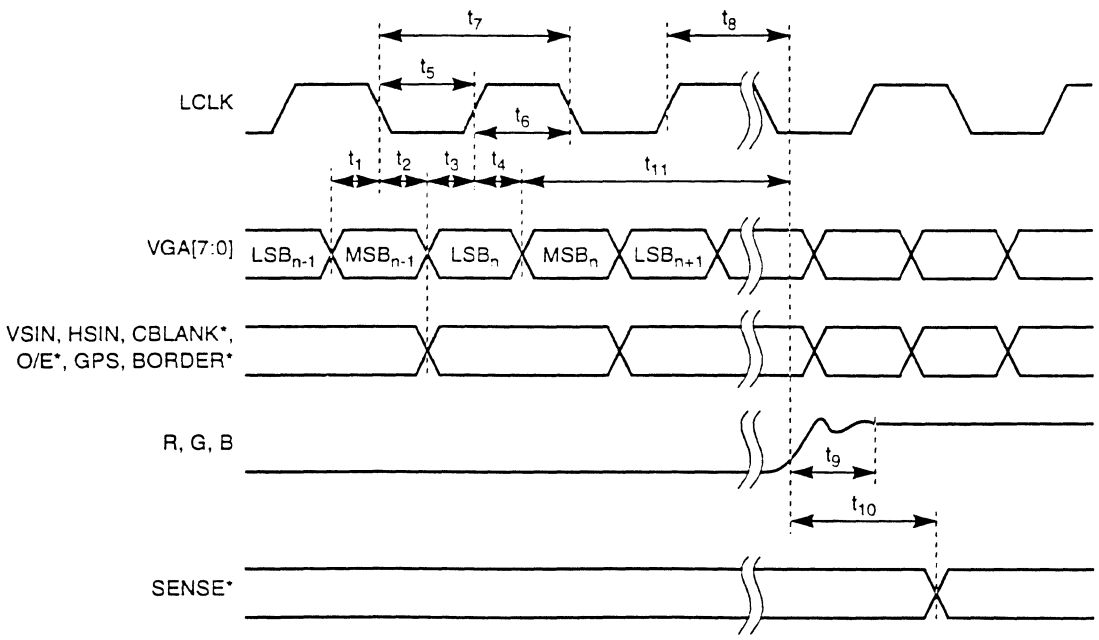


Figure 5-8. VGA Port Interface Timing, EC Mode (16-Bit)

5.1.5.10 VAFC Output Timing On VSD[31:0] from PCLKn

Ref.	Parameter		MIN	MAX
t_1	Delay	PCLKn rise to PCO output	2 ns	10 ns
t_2	Delay	LCLK fall to BLANKOUT* valid output	2 ns	10 ns
t_3	Delay	LCLK fall to VSD[31:0] valid output	2 ns	10 ns

NOTE: System configuration: 1:1 mode, 24-bit/pixel data input on GSD[31:0], VSD[31:0] configured for VAFC-output mode, EVIDEO* high.

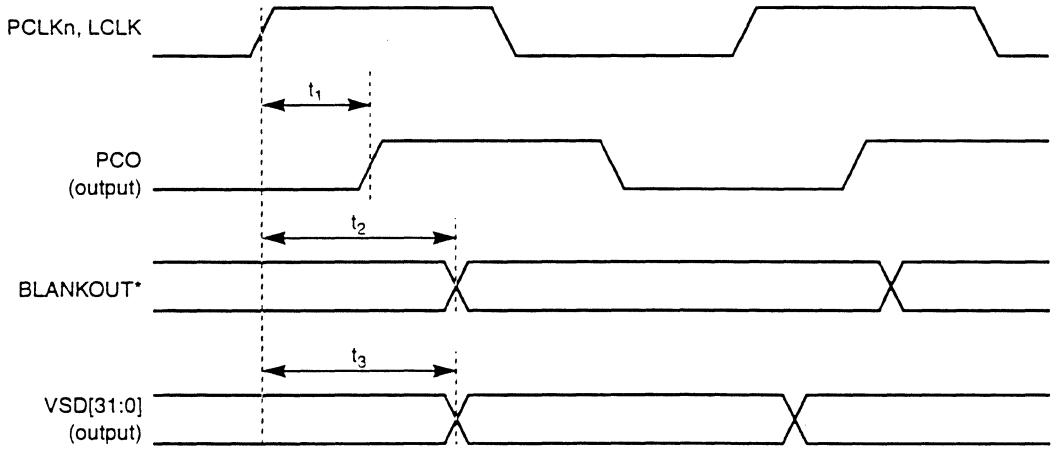


Figure 5-9. VAFC Output Timing on VSD[32:0] from PCLKn

5.2 Electrical Specifications — CL-PX2085

5.2.1 Absolute Maximum Ratings

This section lists the absolute maximum ratings of the MediaDAC. Stresses above those listed can cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods can affect device reliability.

Ambient temperature under bias ..... 0° to + 55° C

Storage temperature ..... -65° to +150° C

Voltage on any pin with respect to ground ..... GND -0.5 to V_{CC} + 0.5 V

Power supply voltage ..... 7 V

Lead temperature (10 seconds) ..... 300° C

5.2.2 DC Specifications (Digital)

Symbol	Parameter	MIN	MAX	Conditions
V_{DD}	Power supply voltage	4.75 V	5.25 V	Normal operation
V_{IL}	Input low voltage	0 V	0.8 V	
V_{IH}	Input high voltage	2.0 V	V_{DD} + 0.8 V	
V_{OL}	Output low voltage		0.4 V	I_{OL} = 4 mA
V_{OH}	Output high voltage	2.4 V		I_{OH} = 400 µA
V_{ILC}	Input low-voltage CMOS		0.8 V	
V_{IHC}	Input high-voltage CMOS	3.0 V		
V_{OLC}	Output low-voltage CMOS		0.4 V	I_{OLC} = 3.2 mA
V_{OHC}	Output high-voltage CMOS	3.5 V		I_{OHC} = -200 µA
I_{DD}	Digital supply current		300 mA	V_{DD} nominal
I_{DDT}	Total supply current		450 mA	Note 1
I_L	Input leakage	-10 µA	10 µA	$0 < V_{IN} < V_{DD}$
C_{IN}	Input capacitance		10 pF	
C_{OUT}	Output capacitance		10 pF	

NOTES: 1) I_{DDT} is the sum of I_{DD} + $DACI_{DD}$ + $CLKI_{DD}$.
2) DACVSS must not exceed V_{DD}.

5.2.3 DC Specifications (RAMDAC)

(V_{DD} = 5 V ± 5%, T_A = 0° to 70° C, unless otherwise specified)

Symbol	Parameter	MIN	TYP	MAX	Conditions
DACVDD	Power supply voltage	4.75 V	5.0 V	5.25 V	Normal operation
$DACI_{DD}$	DAC supply current		45 mA	60 mA	Note 4

NOTES: 1) I_{REF} outside the specified limits may cause the analog outputs to become invalid.
2) The dotclock must be stable for a period of 100 µs after power-up before proper DAC operation isguaranteed.
3) See the I_{REF} description in Section 3.
4) $DACI_{DD}$ is specified with the three analog outputs (RGB) each loaded with 50 Ω.

5.2.4 DAC Characteristics

Test conditions generate PS/2 video-signal levels unless otherwise specified. Recommended test conditions are R_{set} = 1.87 kΩ, V_{ref} = 1.235 V, and I_{ref} = 0.679 mA. The parameters in the following table apply over the full voltage and temperature ranges specified in Sections 5.2.2 and 5.2.3.

V_{DD} = 5 V ± 5%, T_A = 0° to 70° C, unless otherwise specified.

Symbol	Parameter	MIN	TYP	MAX	Conditions
R	Resolution	8 bits			
I_{Omax}	Output current			-27 mA	V_O < 1 V
Co	Output capacitance			12 pF	
DT	DAC-to-DAC variability			± 5%	Notes 1, 2
DNL	Differential nonlinearity			± 1.0 ISB	
	White level relative to black	13.3 mA	14.0 mA	14.7 mA	
	Blank pedestal	1.25 mA	1.45 mA	1.65 mA	
	Sync pedestal	6.8 mA	7.59 mA	8.3 mA	
	Sense trip level	330 mV	400 mV	470 mV	
	Internal voltage reference output	1.11 V	1.235 V	1.35 V	

NOTES: 1) Outputs loaded identically.
2) Midpoint of the distribution of the three DACs measured at full-scale deflection.

5.2.5 AC Characteristics/Timing Information

This section includes system timing requirements for the MediaDAC™. Timings are provided in nanoseconds (ns), at TTL input levels, with the ambient temperature varying from 0° to 70° C, and V_{DD} varying from 4.75 to 5.25 VDC.

NOTES: 1) All timings assume a load of 50 pF.

2) TTL signals are measured at TTL threshold; CMOS signals are measured at CMOS threshold.

5.2.5.1 Index of Timing Information

5.2.5.2 Video Port Timing

Ref.	Parameter		MIN	MAX
t_1	Setup	ZC, EVIDEO*, VWE, VSD stable to VCLK rising edge	3 ns	
t_2	Hold	ZC, EVIDEO*, VWE, VSD stable after VCLK rising edge	3 ns	

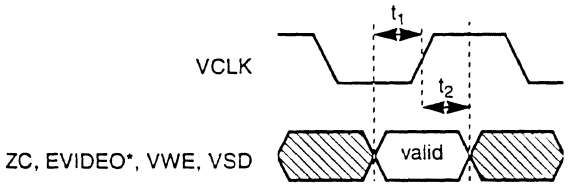


Figure 5-10. Video Port Timing

5.2.5.3 ISA Bus Timing

Ref.	Parameter		MIN	MAX
t_1	Setup	Valid address to IOR*/IOW* active	25 ns	
t_2	Delay	IOR*/IOW* active to DEN* active, DDIR change		20 ns
t_3	Delay	IOR* active to data output Low-Z	5 ns	25 ns
t_4	Delay	IOR* active to data out valid		40 ns
t_5	Pulse width	IOR*/IOW*	50 ns	
t_6	Delay	IOR*/IOW* inactive to DEN* inactive, DDIR changing		20 ns
t_7	Delay	IOR* inactive to tristate delay		15 ns
t_8	Hold	Address from IOR*/IOW* active	0 ns	
t_9	Setup	Data valid to IOW* inactive	20 ns	
t_{10}	Hold	IOW* inactive to data invalid	0 ns	
t_{11}	Delay	IOW* inactive to next IOW* or IOR* active	50 ns	
t_{12}	Setup	AEN falling edge to IOW* or IOR* active	10 ns	
t_{13}	Delay	IOW* or IOR* inactive to AEN rising edge	10 ns	

NOTE: The low-byte address-buffer enable must be qualified by IOR* to avoid data contention. Also, AEN must be low during cycle.

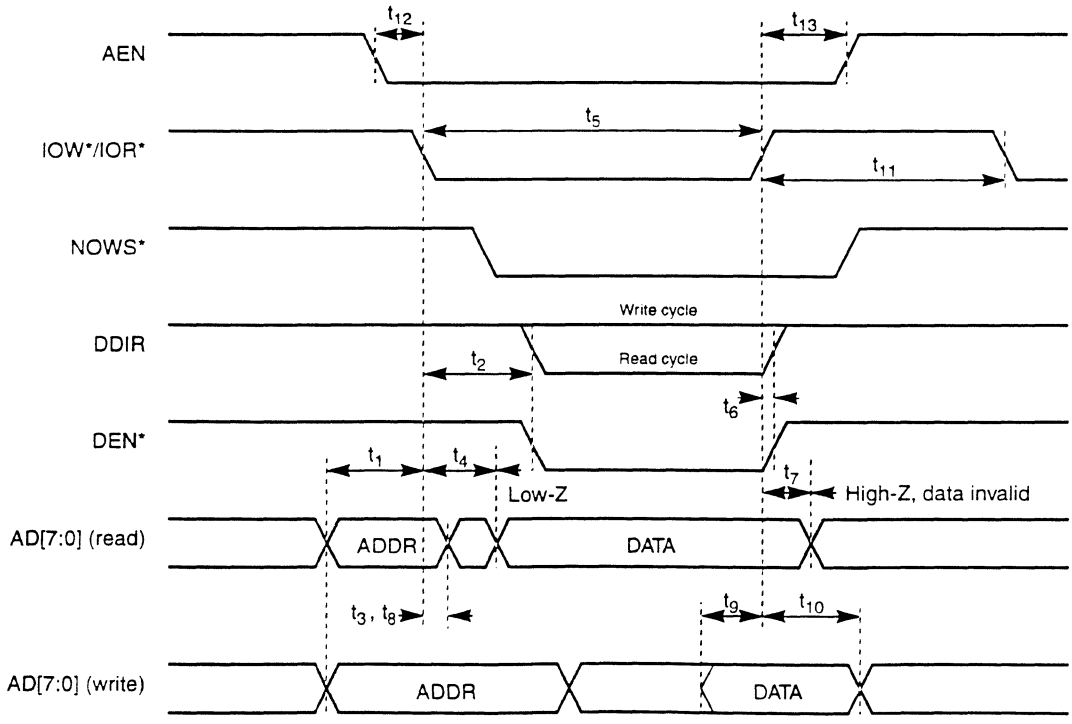


Figure 5-11. ISA Bus Timing

5.2.5.4 MicroChannel® Bus Timing

Ref.	Parameter		MIN	MAX
t_1	Setup	Address valid to CMD* active	25 ns	
t_2	Delay	CMD* active to DEN* active		20 ns
t_{3a}	Hold	Address valid from CMD* active	0 ns	
t_{3b}	Delay	CMD* active to data output low-Z	5 ns	25 ns
t_4	Delay	CMD* active to read data valid		40 ns
t_5	Pulse width	CMD*	50 ns	
t_6	Delay	CMD* inactive to data invalid, high-Z		15 ns
t_7	Setup	Write data valid to CMD* inactive	20 ns	
t_8	Hold	Write data valid to CMD* inactive	0 ns	
t_9	Delay	CMD* inactive to next CMD* active	50 ns	
t_{10}	Setup	Status valid to CMD* active	25 ns	
t_{11}	Hold	CMD* active to status invalid	5 ns	
t_{12}	Delay	CMD* inactive to DEN* inactive, DDIR change		20 ns

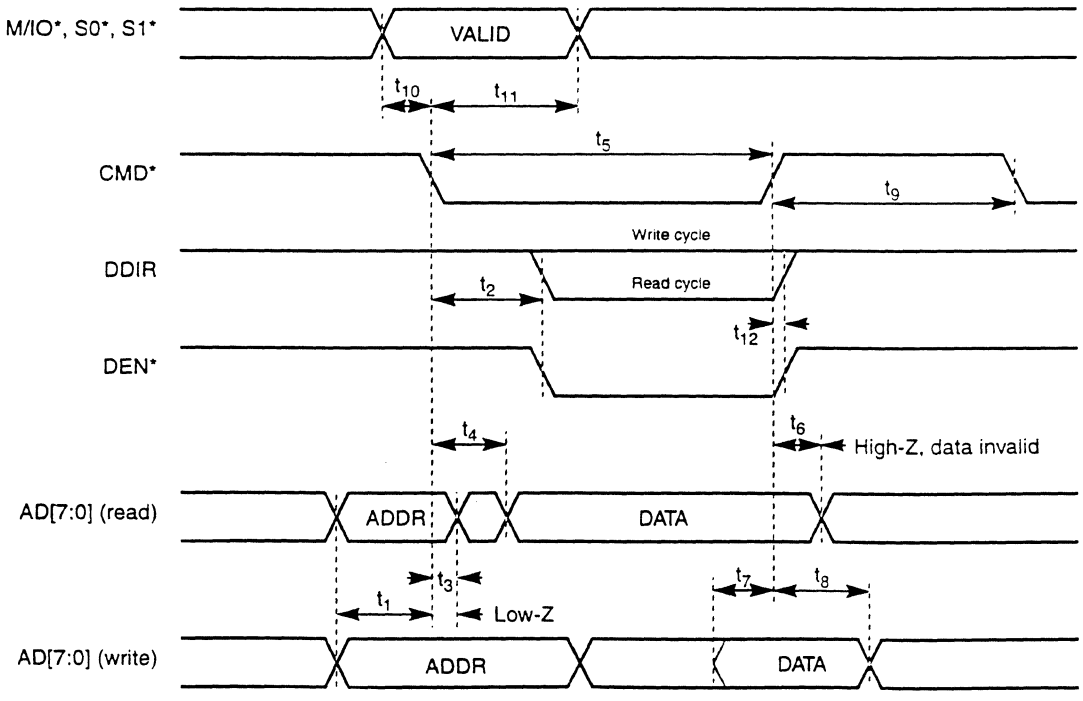


Figure 5-12. MicroChannel® Bus Timing

5.2.5.5 Local Hardware Interface Timing

Ref.	Parameter		MIN	MAX
t_1	Setup	Valid address to IOR*/IOW* active	25 ns	
t_2	Delay	IOR*/IOW* active to DEN* active, DDIR change		20 ns
t_3	Delay	IOR* active to data out Low-Z	5 ns	25 ns
t_4	Delay	IOR* active to data out valid		40 ns
t_5	Pulse width	IOR*/IOW*	50 ns	
t_6	Delay	IOR*/IOW* inactive to DEN* inactive, DDIR changing		20 ns
t_7	Delay	IOR* inactive to data High-Z		15 ns
t_8	Hold	From IOR*/IOW* active to Address invalid	0 ns	
t_9	Setup	Data valid to IOW* inactive	20 ns	
t_{10}	Hold	IOW* inactive to data invalid	0 ns	
t_{11}	Delay	IOW* inactive to next IOW* or IOR* active	50 ns	
t_{12}	Setup	CS* falling edge to IOW* or IOR* active	10 ns	
t_{13}	Delay	IOW* or IOR* inactive to CS* inactive	10 ns	

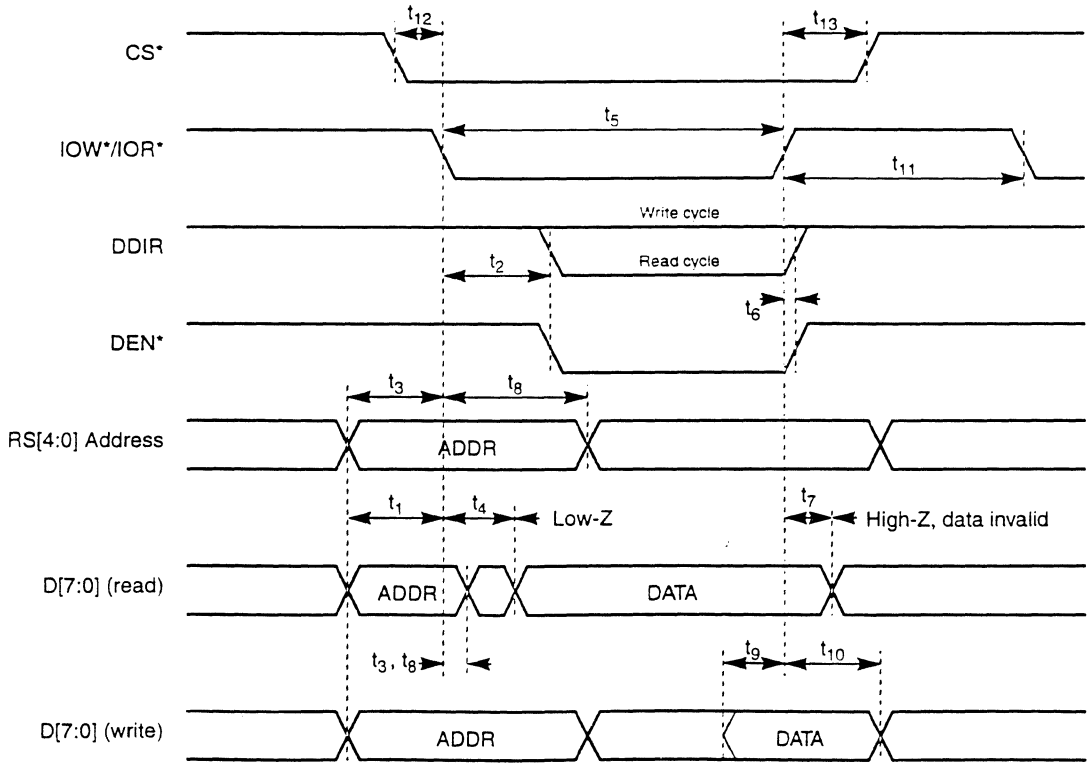


Figure 5-13. Local Hardware Interface Timing

5.2.5.6 Clocks as Inputs (85 MHz)

Ref.	Parameter		85 MHz MIN	85 MHz MAX	135 MHz MIN	135 MHz MAX
t_1	Rise time	PCLKn		3 ns		3 ns
		LCLK		3 ns		3 ns
		VCLK		3 ns		3 ns
t_2	Fall time	PCLKn		3 ns		3 ns
		LCLK		3 ns		3 ns
		VCLK		3 ns		3 ns
t_3	High period	PCLKn	4 ns		4 ns	
		LCLK	4 ns		4 ns	
		VCLK	5 ns		5 ns	
t_4	Low period	PCLKn	4 ns		4 ns	
		LCLK	4 ns		4 ns	
		VCLK	5 ns		5 ns	
t_5	Cycle time	PCLKn	11.5 ns		11.5 ns	
		LCLK	11.5 ns		11.5 ns	
		VCLK	14.8 ns		14.8 ns	

NOTE: LCLK and SCLK cycle and pulse width times are multiplied by 2, 4, and 8 in 2:1, 4:1, and 8:1 multiplexing modes, respectively.

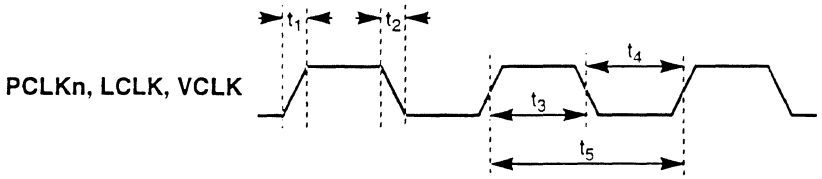


Figure 5-14. PCLKn, LCLK, and VCLK as Inputs (LCLK = 1:1 Multiplex Rate)

5.2.5.7 Sync, RGB, and GSD[23:0] Output Timing from PCLKn

Ref.	Parameter		MIN	MAX
t_1	Delay	PCLKn rise to RGB output		30 ns
t_2	Rise/Fall	RGB output	3 ns	
t_3	Settling time	RGB output full-scale		15 ns
t_4	Delay	PCLKn rise to VSOUT, HSOUT output	2 ns	10 ns
t_5	Delay	PCLKn rise to GSD[23:0] output	2 ns	10 ns
(not shown)	Delay	RGB output to SENSE* output		1 µs

NOTES: 1) Output delay is measured from the 50% point of the rising edge of PCLKn to the 50% point of full-scale transition.

2) Settling time is measured from the 50% point of full-scale transition to the output remaining within ±1 LSB

3) Output rise/fall time is measured between the 10% and 90% points of full-scale transition.

4) In 1:1 multiplexing mode, RGB data is clocked out directly from LCLK, synchronizing with SCLK and PCLKn is unnecessary and bypassed. All timing PCLKn references in the table above apply to LCLK when in 1:1 multiplexing mode.

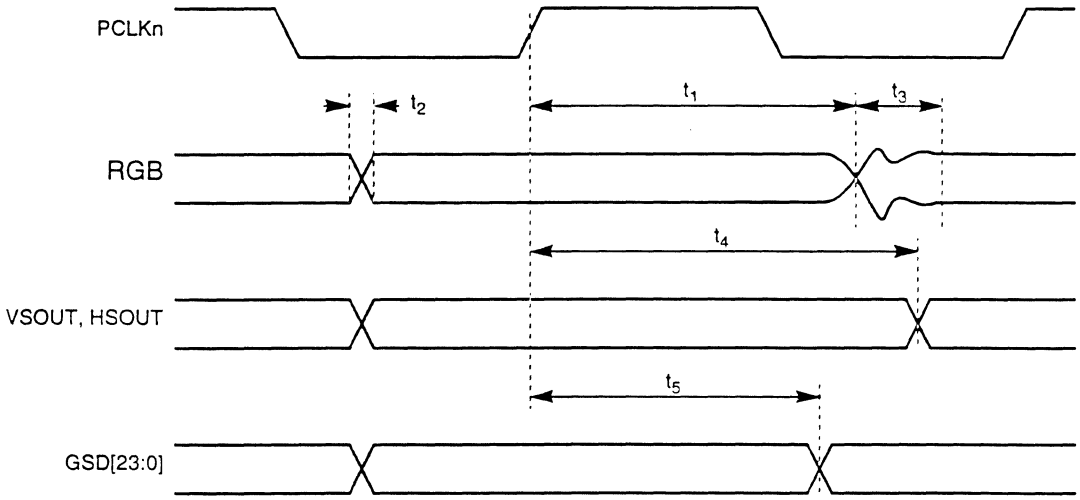


Figure 5-15. Sync, RGB, and GSD[23:0] Output Timing from PCLKn

5.2.5.8 Graphics Port Input Timing

Ref.	Parameter		MIN	MAX
t_1	Delay	PCLKn rise to PCO, SCLK output (2:1, 4:1, 8:1 mode)	2 ns	10 ns
t_2	Delay	LCLK rising edge to PCO, SCLK rising edge output (1:1 mode)	2 ns	10 ns
t_3	Setup	LCLK stable to PCLKn rising edge	3 ns	
t_4	Hold	LCLK stable after PCLKn rising edge	3 ns	
t_5	Setup	Graphics data, control to LCLK rising edge	1 ns	
t_6	Hold	Graphics data, control to LCLK rising edge	5 ns	

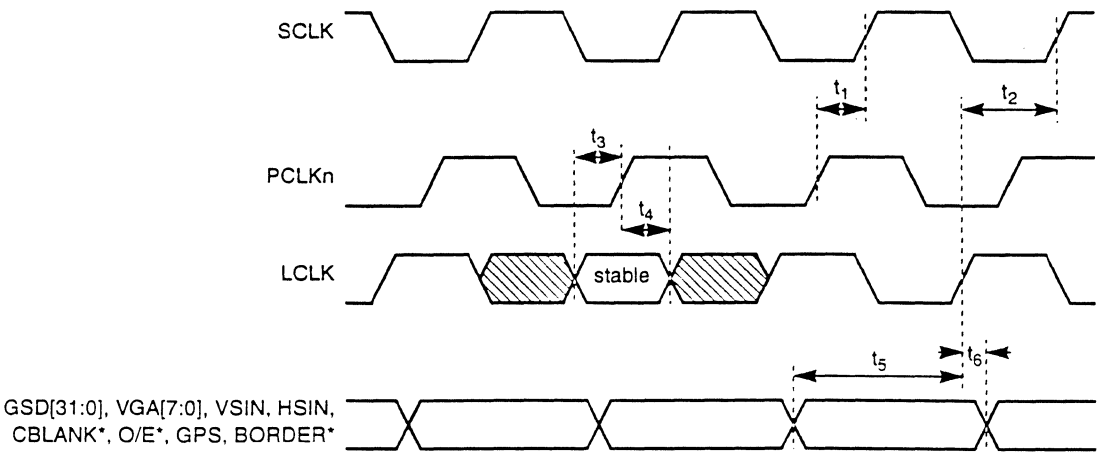


Figure 5-16. Graphics Port Input Timing

5.2.5.9 VGA Port Interface Timing, EC Mode

Ref.	Parameter		MIN	MAX
t_1	Setup	MSB data valid to falling edge PCLKn	1 ns	
t_2	Hold	LCLK falling edge to MSB data	5 ns	
t_3	Setup	LSB data and graphics controls valid to rising edge PCLKn	1 ns	
t_4	Hold	LCLK rising edge to LSB data, graphics controls invalid	5 ns	
t_5	Pulse Width	LCLK low	12 ns	
t_6	Pulse Width	LCLK high	12 ns	
t_7	Period	LCLK	30 ns	
t_8	Delay	LCLK rising edge to RGB output		15 ns
t_9	Delay	Valid data to full-scale RGB output		5 ns
t_{10}	Delay	RGB out to SENSE* valid	0 µs	1 µs
t_{11}	Delay	Pipeline		32 PCLKn cycles

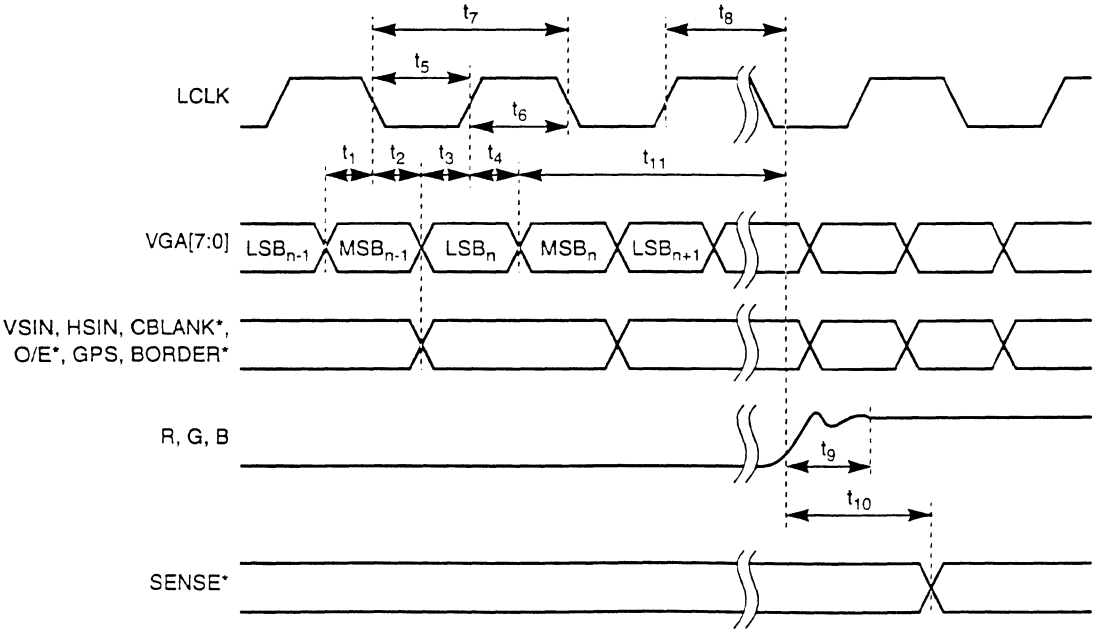


Figure 5-17. VGA Port Interface Timing, EC Mode (16-Bit)

5.2.5.10 VAFC Output Timing On VSD[31:0] from PCLKn

Ref.	Parameter		MIN	MAX
t_1	Delay	PCLKn fall to PCO output	2 ns	10 ns
t_2	Delay	LCLK fall to BLANKOUT* valid output	2 ns	10 ns
t_3	Delay	LCLK fall to VSD[31:0] valid output	2 ns	10 ns

NOTE: System configuration: 1:1 mode, 24-bit/pixel data input on GSD[31:0], VSD[31:0] configured for VAFC-output mode, EVIDEO* high.

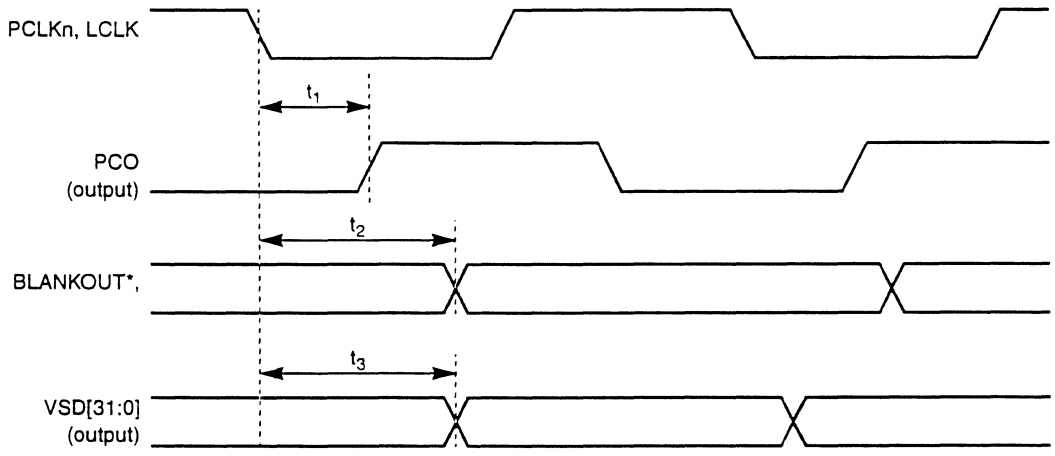
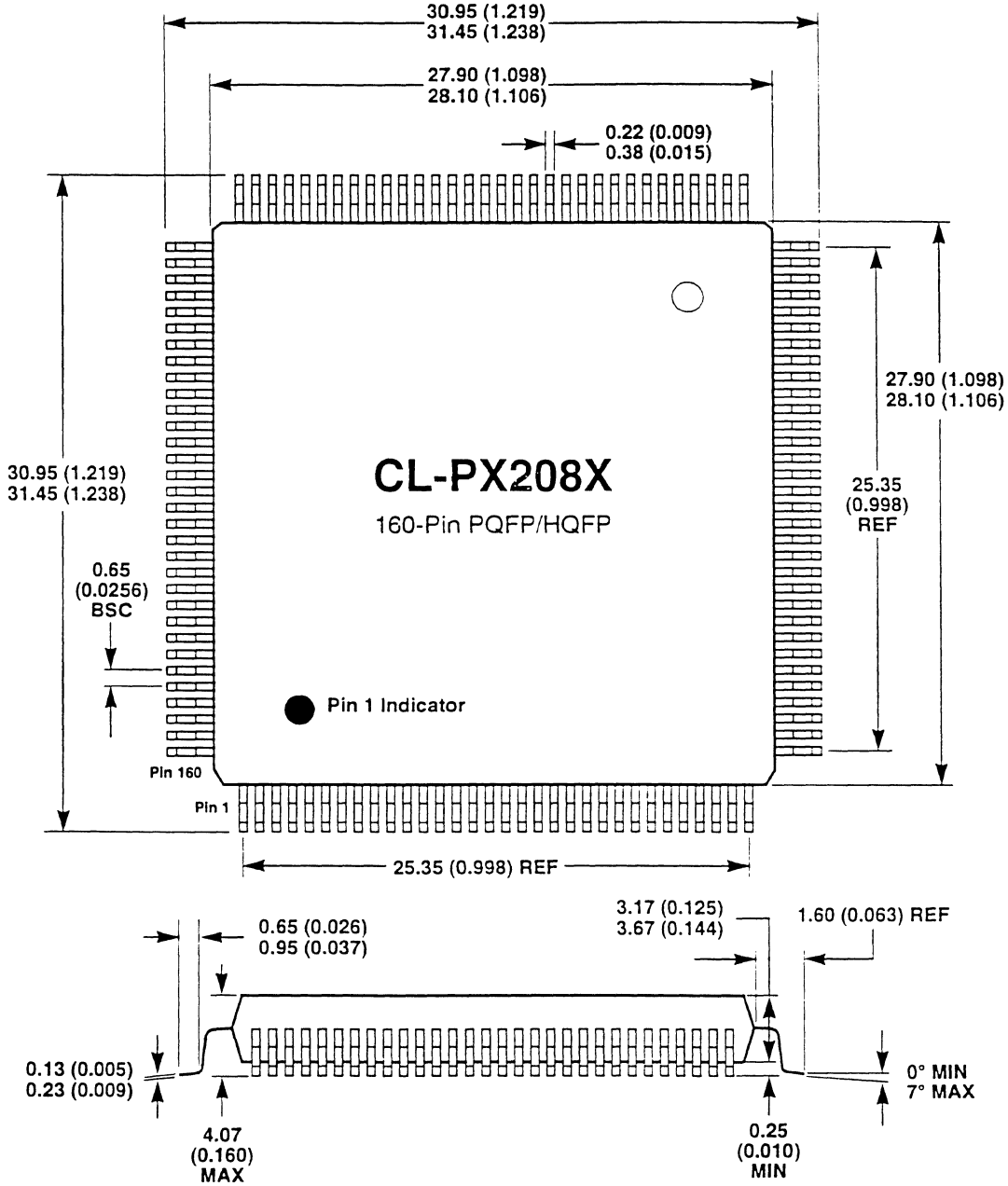


Figure 5-18. VAFC Output Timing on VSD[32:0] from PCLKn

6. PACKAGE SPECIFICATIONS



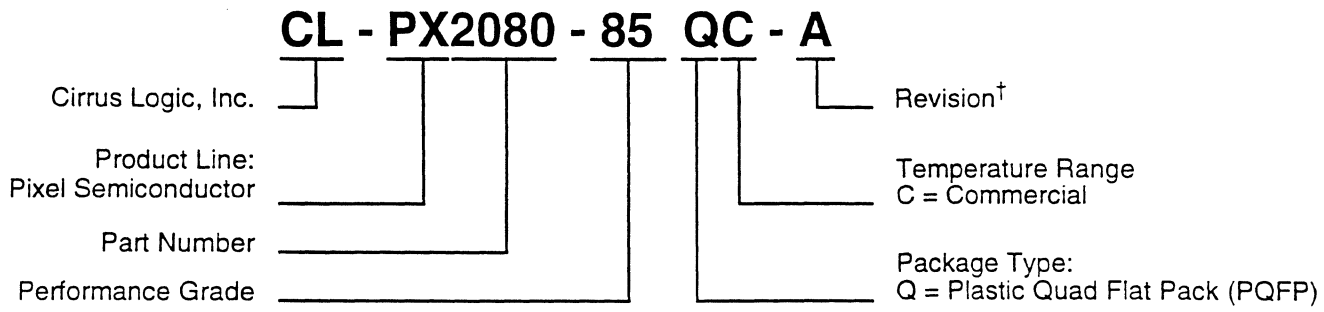
NOTES:

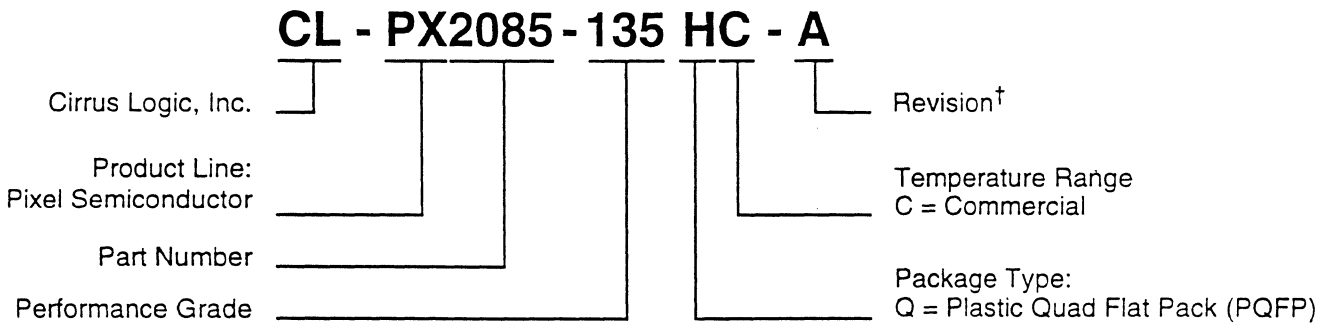
5) Dimensions are in millimeters (inches), and controlling dimension is millimeter.
6) Drawing above does not reflect exact package pin count.
7) Before beginning any new design with this device, please contact Cirrus Logic for the latest package information.
4) HQFP is a high-performance QFP with an exposed or unexposed heat sink.

Figure 6-1. CL-PX208X Package Information

7. ORDERING INFORMATION

When ordering the CL-PX208X MediaDAC™, use the following format:





† Contact Cirrus Logic, Inc., for up-to-date information.

PART II: SOFTWARE

Section Page

8. MediaDAC™ CONTROL FUNCTIONS

The group of functions listed in this section provides controls for communications between the PxVPS and the CL-PX208X MediaDAC device.

WARNING: The following functions are available but should be used very carefully. Calling these functions can change the behavior of all PxVPS applications currently running and can affect the graphics system. Therefore, they should only be used when it is clear to the application that it is the only PxVPS-based application running. Most PxVPS applications should never have a reason to call these functions.

8.1 Control Functions

8.1.1 pxCreateOverlayBrush

DESCRIPTION This routine creates a Windows-compatible brush to paint the current overlay color to allow video to display in an active display window. This brush has to be deleted (pxDeleteOverlayBrush) before the application terminates.

HBRUSH pxCreateOverlayBrush (void);

INPUT PARAMETERS None.

RETURN VALUE

Type	Argument	Meaning
HBRUSH	hOverlay	Overlay brush handle for current overlay color or NULL.
	NULL	Operation failed. Call pxGetSystemError to get error code.

8.1.2 pxDeleteOverlayBrush

DESCRIPTION This routine deletes a brush created by pxCreateOverlayBrush. Call this routine before the application terminates.

int pxDeleteOverlayBrush (HBRUSH);

INPUT PARAMETERS

Type	Argument	Meaning
HBRUSH	hOverlay	Overlay brush handle for current overlay color to delete.

RETURN VALUE

Type	Argument	Meaning
int	iStatus	See below.
	PX_OK	Successful completion of operation.
	PX_ERROR	Operation failed. Call pxGetSystemError to get error code.

8.2 Message-Based APIs

8.2.1 pxdrvSendMediaDACMessage

DESCRIPTION This routine sends a message to the MediaDAC driver.

#include "PxVPS.h"

int pxdrvSendMediaDACMessage (uMessage, lParam1, lParam2);

REMARKS The MediaDAC API is defined in PXMDAPI.H. This includes the available messages and their respective parameters (lParam1 and lParam2). The meanings of the two messages are message-specific. The debug version of the MediaDAC driver asserts all pointer parameters.

INPUT PARAMETER

Type	Argument	Meaning
UINT	uMessage;	The Media DAC message (defined in PXMDAPI.H).
LPARAM	lParam1;	Message specific parameter one.
LPARAM	lParam2;	Message specific parameter two.

RETURN VALUE

Type	Argument	Meaning
int	iStatus;	See below.
	PX_OK	Successful completion of operation.
	PX_ERROR	Operation failed. Call pxGetSystemError to get the error code.

8.2.2 pxGetMDGammaPalette

DESCRIPTION This routine returns the gamma palette and determines if the gamma correction is enabled/disabled.

#include "PxVPS.h"

int pxGetMDGammaPalette (unsigned char far *lpGammaPalette);

REMARKS lpGammaPalette must be of size GAMMA_ARRAY_SIZE or must be NULL. If it is NULL, the palette is ignored and this function simply returns the gamma correction status. The gamma palette contains 256 entries; each of which contains a 24-bit RGB value.

INPUT PARAMETER

Type	Argument	Meaning
unsigned char far *	lpGammaPalette;	Pointer to an application supplied buffer in which the gamma palette is to be stored.

RETURN VALUE

Type	Argument	Meaning
int	*1*	Gamma correction is enabled.
	0	Gamma correction is disabled.

EXAMPLES

```
.
.
/*
 * If the gamma palette is not enabled, then return FALSE.
 */
if (pxGetMDGammaPalette (lpGammaPalette) == FALSE)
{
      return (FALSE);
}
.
.
.
```

SEE ALSO pxSetMDGammaPalette

8.2.3 pxGetMDOvlyColor

DESCRIPTION This routine returns the current MediaDAC overlay color to the user.

#include "PxVPS.h"

unsigned char pxGetMDOvlyColor (void);

REMARKS The overlay color is the current value of the red chroma key register of the MediaDAC. As VGA data is sent to the MediaDAC, it is compared to the chroma-key registers. When there is a match, Image data can be displayed instead of graphics as determined by the graphics overlay mode.

INPUT PARAMETERS None.

RETURN VALUE

Type	Argument	Meaning
unsigned char	CurrentColor;	Current overlay color index.

EXAMPLES

```
.
.
.
      bOverlayColor = pxGetMDOvlyColor();
.
.
.
```

SEE ALSO pxGetMDOvlyMode
pxSetMDOvlyColor
pxSetMDOvlyMode

8.2.4 pxGetMDOvlyMode

DESCRIPTION This routine returns the current MediaDAC overlay mode to the user.

#include "PxVPS.h"

unsigned char pxGetMDOvlyMode (void);

REMARKS The overlay mode resides in the graphics overlay opcode register of the MediaDAC. This 8-bit value determines the manner in which the video and graphics data are displayed. See the *pxSetMDOvlyMode* command on page 110 for a description of how this register works.

No input parameters

RETURN VALUE

Type	Argument	Meaning
unsigned char.	OverlayMode; (for possible values, see table below).	Current overlay mode index. (for possible values, see table below).

Overlay Mode	Current Overlay Mode Index
CHROMA_AND_VALID_IMAGE	The image data are displayed as long as the chroma-key color matches and it resides within a valid display window. This is the default used by PxVPS.
GRAPHICS_ONLY	Displays graphics-only data. No image data are ever displayed.
CHROMA_MATCH_ONLY	The image data are displayed as long as the chroma-key color matches, regardless of data tagging or whether the chroma-key color resides in a valid display window.
VALID_IMAGE_ONLY	The image inside the valid display window is always displayed regardless of chroma-keying or data tagging.
TAG_ONLY	Tagged image data are always displayed regardless of chroma-key information or whether the tagged data resides in a valid display window.
CHROMA_AND_TAG	The image data are always displayed as long as the chroma-key matches and the data are tagged.
TAG_AND_VALID_IMAGE	The image data are always displayed as long as the data are tagged and it resides within a valid display window.
CHROMA_TAG_AND_VALID_IMAGE	The image data are only displayed if the chroma-key matches, the data are tagged, and it resides within a valid display window.
CHROMA_AND_NOT_TAG	The image data are displayed if the chroma-key matches and the data are not tagged.

EXAMPLES

```
.
.
.
        bOvlyMode = pxGetMDOvlyMode();
.
.
.
```

SEE ALSO pxGetMDOvlyColor/pxSetMDOvlyColor/pxSetMDOvlyMode

8.2.5 pxGetMDVideoFormat

DESCRIPTION This routine returns the color-space format of the video bus.

```
#include "PxVPS.h"
unsigned char pxGetMDVideoFormat (void);
```

REMARKS The color space format determines how the MediaDAC interprets incoming video data.

No Input Parameters.

RETURN VALUE

Type	Argument	Meaning
	bFormat:	The video buffer color space format.
unsigned char	YUV422	YUV 4:2:2 format non-tagged.
	TYUV422	Y(U:T)(V:T) 4:(2):*2) format (LSB of U and V are tag data).
	RGB565	RGB 5:6:5 format.
	RGB555	RGB 5:5:5 format.
	TRGB1555	TRGB 1:5:5:5 format.
	RGB888	RGB 8:8:8 format non-tagged.
	TRGB1888	TRGB 1:8:8:8 format (Bbit 31 is tag bit).

EXAMPLES

```
/*
 * Get the current format of the video frame buffer, and if it is a
 * tagged format, reprogram the CL-PX2080 to expect non-tagged data.
 */
switch (pxGetMDVideoFormat ())
{
       case TYUV422:
              if (pxSetMDVideoFormat (YUV422))
              {
                 iErrorCode = pxGetSystemError ();
              }
              break;

       case TRGB1555:
              if (pxSetMDVideoFormat (RGB565))
              {
                 iErrorCode = pxGetSystemError ();
              }
              break;

       case TRGB1888:
              if (pxSetMDVideoFormat (RGB888))
              {
                 iErrorCode = pxGetSystemError ();
              }
              break;

       case YUV422:
       case RGB565:
       case RGB555:
       case RGB888:
       break;
}
```

SEE ALSO pxSetMDVideoFormat

8.2.6 pxSetMDGammaPalette

DESCRIPTION This routine sets the gamma palette and enables/disables gamma correction.

#include "PxVPS.h"

int pxSetMDGammaPalette (unsigned char far *lpGammaPalette, int iState);

REMARKS lpGammaPalette must be of size GAMMA_ARRAY_SIZE or must be NULL. If it is NULL, the palette is ignored and this function simply enables/disables gamma correction. The gamma palette contains 256 entries; each of which contains a 24-bit RGB value.

NOTE: Calling this function affects the operation of all PxVPS-based applications that may be running. Use this function carefully.

INPUT PARAMETERS

Type	Argument	Meaning
unsigned char far *	lpGammaPalette;	Pointer to an application supplied buffer containing the new gamma palette.
int	iState;	Zero if gamma correction is disabled; non-zero if it is disabled.

RETURN VALUE

Type	Argument	Meaning
unsigned int	iStatus;	Handle to dialog created.
	iOk	Successful completion of operation.
	iError	Operation failed. Call pxGetSystemError to get the error code.

EXAMPLES

```
.
.
.
/*
 * Program the gamma palette of the CL-PX2080 and enable the gamma
 * correction.
 */
if (pxSetMDGammaPalette (lpGammaPalette, TRUE) == iError)
{
        iErrorCode = pxGetSystemError ();
}
.
.
.
```

SEE ALSO pxGetMDGammaPalette

8.2.7 pxSetMDOvlyColor

DESCRIPTION This routine sets the current MediaDAC overlay color as specified by the user.

#include "PxVPS.h"

void pxSetMDOvlyColor (unsigned char bValue);

REMARKS The overlay color is the value loaded into the red, green, and blue chroma-key registers of the MediaDAC. As VGA data issent to the MediaDAC, it is compared to the chroma-key registers. When there is a match, image data can be displayed instead of graphics as determined by the graphics overlay mode. In a typical programming example, the application fills the client area of a window with a solid color. The index of this solid color is then used in the *pxSetMdOvlyColor* call.

NOTE: Calling this function affects the operation of all PxVPS-based applications that may be running. Use this function carefully.

INPUT PARAMETERS

Type	Argument	Meaning
unsigned char	bValue;	New overlay color index.

RETURN VALUE None.

EXAMPLES

```
.
.
.
        pxSetMDOvlyColor (BValue);
.
.
.
```

SEE ALSO pxGetMDOvlyColor
pxGetMDOvlyMode
pxSetMDOvlyMode

8.2.8 pxSetMDOvlyMode

DESCRIPTION This routine sets the current MediaDAC overlay mode.

#include "PxVPS.h"

int pxSetMDOvlyMode (unsigned char bValue);

REMARKS The overlay mode determines how the MediaDAC mixes the video image with the graphics data. The new overlay mode must be one of the following values:

Value	Meaning
CHROMA_AND_VALID_IMAGE	The image data are displayed as long as the chroma-key color matches and it resides within a valid display window. This is the default used by PxVPS.
GRAPHICS_ONLY	Displays graphics-only data. No image data are ever displayed.
CHROMA_MATCH_ONLY	The image data are displayed as long as the chroma-key color matches, regardless of data tagging or whether the chroma-key color resides in a valid display window.
VALID_IMAGE_ONLY	The image inside the valid display window is always displayed regardless of chroma-keying or data tagging.
TAG_ONLY	Tagged image data are always displayed regardless of chroma-key information or whether the tagged data resides in a valid display window.
CHROMA_AND_TAG	The image data are always displayed as long as the chroma-key matches and the data are tagged.
TAG_AND_VALID_IMAGE	The image data are always displayed as long as the data are tagged and it resides within a valid display window.
CHROMA_TAG_AND_VALID_IMAGE	The image data is only displayed if the chroma-key matches, the data is tagged, and it resides within a valid display window.
CHROMA_AND_NOT_TAG	The image data are displayed if the chroma-key matches and the data are not tagged.

NOTE: Calling this function affects the operation of all PxVPS-based applications and possibly every Windows applications that may be running. Use this function carefully.

INPUT PARAMETERS

Type	Argument	Meaning
unsigned char	bValue;	Specifies the new overlay mode.

RETURN VALUE

Type	Argument	Meaning
unsigned int	iStatus;	
	iOk	Successful completion of operation.
	iError	Operation failed. Call pxGetSystemError to get the error code.

EXAMPLES

```
/* Set up the CL-PX208x to overlay image if it receives the proper
 * overlay color from the graphics stream while getting valid image
 * data from the video frame buffer.
 */
bValue = CHROMA_AND_VALID_IMAGE;
if (pxSetMDOvlyMode (bValue) == iError)
{
      iErrorCode = pxGetSystemError ();
}
```

SEE ALSO pxGetMDOvlyMode

8.2.9 pxSetMDVideoFormat

DESCRIPTION This routine sets the color space format of the video bus.

#include "PxVPS.h"

int pxSetMDVideoFormat (unsigned char Format);

REMARKS The color space format determines how the MediaDAC interprets incoming video data.

NOTE: Calling this function affects the operation of all PxVPS-based applications that may be running. Use function carefully.

INPUT PARAMETERS

Type	Argument	Meaning
	bFormat;	The video color space format.
unsigned char	YUV422;	YUV 4:2:2 format non-tagged.
	TYUV422	Y(U:T)(V:T) 4:(2):(2) format (LSB of U and V are tag data).
	RGB565	RGB 5:6:5 format.
	RGB555	RGB 5:5:5 format.
	TRGB1555	TRGB 1:5:5:5 format.
	RGB888	RGB 8:8:8 format non-tagged.
	TRGB1888	TRGB 1:8:8:8 format (bit 31 is tag bit).

RETURN

Type	Argument	Meaning
unsigned int	iStatus;	
	iOk	Successful completion of operation.
	iError	Operation failed. Call pxGetSystemError to get the error code.

EXAMPLES

```
.
.
.
/*
 * Get the current format of the video frame buffer, and, if it is a
 * tagged format, reprogram the CL-PX208x to expect non-tagged data.
 */
switch (pxGetMDVideoFormat ())
{
       case TYUV422:
              if (pxSetMDVideoFormat (YUV422))
              {
                   iErrorCode = pxGetSystemError ();
              }
              break;


       case TRGB1555:
              if (pxSetMDVideoFormat (RGB565))
              {
                   iErrorCode = pxGetSystemError ();
              }
              break;
```

```
            case TRGB1888:
                  if (pxSetMDVideoFormat (RGB888))
                  {
                        iErrorCode = pxGetSystemError ();
                  }
                  break;


            case YUV422:
            case RGB565:
            case RGB555:
            case RGB888:
            break;
      }
      .
      .
      .
```

SEE ALSO pxGetMDVideoFormat

8.2.10 DRV_PXMD_GET_ANALOG_SETUP

DESCRIPTION This message returns the current MediaDAC analog setup register to the caller.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE FAR *) lpbAnalogSetup;
LPARAM2	(LPARAM) 0L;

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.11 DRV_PXMD_GET_BIR

DESCRIPTION This message returns the current MediaDAC base index register to the caller. This register is only available if the MediaDAC is configured in ISA bus mode.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE FAR *)lpbBIR;
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.12 DRV_PXMD_GET_CLOCK_RATE

DESCRIPTION This message returns the clock rate being sent to the CL-PX2070. The following values are valid.

Value	Meaning
GRAPHICSCLK_1X	Graphics clock is passed on as is.
GRAPHICSCLK_1HALFX	Graphics clock is divided by two.
GRAPHICSCLK_1QUARTERX	Graphics clock is divided by four.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE FAR *) llpbClockRate;
LPARAM2	(LPARAM) 0L;

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.13 DRV_PXMD_GET_COM_REG_3

DESCRIPTION This message returns the current MediaDAC command register 3 value to the caller. In the event that the MediaDAC is a CL-PX208X, the value is undefined, and the message returns zero.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE FAR *) llpbComReg3;
LPARAM2	(LPARAM) 0L;

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.14 DRV_PXMD_GET_CURSOR_SETUP

DESCRIPTION This message returns the current MediaDAC cursor setup register to the caller.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE FAR *) lpbCursorSetup;
LPARAM2	(LPARAM) 0L;

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.15 DRV_PXMD_GET_CURSOR_X_HI_POSITION

DESCRIPTION This message returns the current MediaDAC hi byte of the hardware cursor X position to the caller.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE FAR *) lpbCrsrPos;
LPARAM2	(LPARAM) 0L;

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

NOTE: The upper nibble of this value is reserved and should always read 0.

8.2.16 DRV_PXMD_GET_CURSOR_X_LOW_POSITION

DESCRIPTION This message returns the current MediaDAC low byte of the hardware cursor X position to the caller.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE FAR *) lpbCrsrPos
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.17 DRV_PXMD_GET_CURSOR_Y_HI_POSITION

DESCRIPTION This message returns the current MediaDAC highbyte of the hardware cursor Y position to the caller.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE FAR *) lpbCrsrPos;
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

NOTE: The upper nibble of this value is reserved and should always read 0.

8.2.18 DRV_PXMD_GET_CURSOR_Y_LOW_POSITION

DESCRIPTION This message returns the current MediaDAC low byte of the hardware cursor Y position to the caller.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE FAR *) lpbCrsrPos;
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.19 DRV_PXMD_GET_ERROR

DESCRIPTION This message returns and clears the current error code of the MediaDAC driver.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (DWORD FAR *) lpdwErrorCode;
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.20 DRV_PXMD_GET_EXT_OVERLAY_COLOR

DESCRIPTION This message sets the current MediaDAC overlay color as specified by the caller. It sets all values (as shown below) independently, not assuming any color mode, for example, DRV_PXMD_GET_OVERLAY_COLOR and DRV_PXMD_SET_OVERLAY_

COLOR 31 23 1615 7 0

XXXXXXXXRRRRRRRRGGGGGGGGBBBBBBBB - where each color component is 8-bits.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (DWORD FAR *) lpdwOverlayColor;
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.21 DRV_PXMD_GET_GAMMA_PALETTE

DESCRIPTION This message returns the current MediaDAC gamma palette to the caller as well as the status bit indicating whether the gamma palette is disabled (zero is enabled, non-zero is disabled).

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (LPGAMMA_PALETTE)lpGammaPalette;
LPARAM2	(LPARAM) (BYTE FAR *) lpbStatus

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.22 DRV_PXMD_GET_GRAPHICS_FORMAT

DESCRIPTION This message returns the current MediaDAC graphics format register to the caller.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE FAR *) lpbGraphicsFormat;
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.23 DRV_PXMD_GET_OVERLAY_COLOR

DESCRIPTION This message returns the current MediaDAC overlay color to the caller. This message is 256-color mode specific.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE FAR *) lpbOverlayColor;
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.24 DRV_PXMD_GET_OVERLAY_MASK

DESCRIPTION This message returns the current MediaDAC overlay mask value to the caller.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE FAR *) lpbOvlyMask;
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.25 DRV_PXMD_GET_OVERLAY_MODE

DESCRIPTION This message returns the current MediaDAC overlay mode to the caller. This mode defines how the image and graphics data should be mixed and displayed. The valid modes are listed below:

Value	Meaning
GRAPHICS_ONLY	Display no image data.
CHROMA_MATCH_ONLY	Display image where overlay key matches graphics color.
VALID_IMAGE_ONLY	Display image where zoom codes are valid.
CHROMA_AND_VALID_IMAGE	Display image where zoom codes are valid and overlay key matches graphics color.
TAG_ONLY	Display image where image data are tagged.
CHROMA_AND_TAG	Display image where overlay key matches graphics color and image data are tagged.
TAG_AND_VALID_IMAGE	Display image where zoom codes are valid and the image data are tagged.
CHROMA_TAG_AND_VALID_IMAGE	Display image where zoom codes are valid, the overlay key matches the graphics pixel color, and the image data are tagged.
CHROMA_AND_NOT_TAG	Display image where overlay key matches the graphics pixel color and the image data are not tagged.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE FAR *) lpbOverlayMode;
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.26 DRV_PXMD_GET_PIXEL_MASK

DESCRIPTION This message returns the current MediaDAC pixel mask to the caller. The pixel mask is AND'ed with the graphics pixels.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE FAR *) lpbPixelMask;
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.27 DRV_PXMD_GET_RESERVED_1

DESCRIPTION This message returns the current MediaDAC reserved 1 register to the caller.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE) bRSV1;
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.28 DRV_PXMD_GET_RESERVED_2

DESCRIPTION This message returns the current MediaDAC reserved 2 register to the caller.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE FAR *) lpbRSV2;
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.29 DRV_PXMD_GET_REV

DESCRIPTION This message returns the current MediaDAC revision to the caller. The revision can be masked to determine what kind of DAC is being used.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE FAR *) lpbRev;
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.30 DRV_PXMD_GET_SYNC_ALIGN

DESCRIPTION This message returns the current MediaDAC sync alignment register to the caller.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE FAR *) lpbSAR
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.31 DRV_PXMD_GET_SYNC_POLARITY

DESCRIPTION This message returns the current MediaDAC sync polarities to the caller. Please note that if the MediaDAC is a CL-PX2085, the return values are undefined since the CL-PX2085 auto-detects sync polarities.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE FAR *) lpbVSyncPol;
LPARAM2	(LPARAM) (BYTE FAR *) lpbHSyncPol

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.32 DRV_PXMD_GET_VGA_PALETTE

DESCRIPTION This message returns the current MediaDAC VGA palette to the caller.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (LPVGA_PALETTE) lpVGAPalette;
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.33 DRV_PXMD_GET_VIDEO_FORMAT

DESCRIPTION This message returns the current MediaDAC video format to the caller. The available formats are defined in PXVPS.H.

This message differs from the DRV_PXMD_GET_VIDEO_FORMAT_REG message in that this message gets the format code, while the latter gets the register value of the VFC register. The valid codes are defined below:

Value	Meaning
YUV422	16-bit YCrCb (YUV) 4:2:2.
TYUV422	16-bit Tagged YCrCb (YUV) 4:2:2.
RGB565	16-bit RGB 5:6:5.
RGB555	16-bit RGB 5:5:5.
TRGB1555	16-bit Tagged RGB.
RGB888	32-bit RGB X:8:8:8.

Value	Meaning
TRGB1888	32-bit Tagged RGB 1:8:8:8.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE FAR *) lpbVideoFormat;
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.34 DRV_PXMD_GET_VIDEO_FORMAT_REG

DESCRIPTION This message returns the current MediaDAC video format register to the caller.

This message differs slightly from DRV_PXMD_GET_VIDEO_FORMAT in that this message gets the register value itself, while the latter gets the video format field and returns the video format codes specified at the top of the header.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE FAR *) lpbVideoFormat;
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.35 DRV_PXMD_INIT_HARDWARE

DESCRIPTION This message initializes the MediaDAC.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) 0L;
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.36 DRV_PXMD_SET_ANALOG_SETUP

DESCRIPTION This message sets the current MediaDAC analog setup register as specified by the caller. Please note that a call to this function could adversely affect all Windows applications.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE) bAnalogSetup;
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.37 DRV_PXMD_SET_BIR

DESCRIPTION This message sets the current MediaDAC base index register as specified by the caller. This register is only available if the MediaDAC is configured in ISA bus mode. Please note that a call to this function could adversely affect all Windows applications.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE) bBIR;
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.38 DRV_PXMD_SET_COM_REG_3

DESCRIPTION This message sees the current MediaDAC command register 3 as specified by the caller. Please note that a call to this function could adversely affect all Windows applications. This message has no affect if the MediaDAC is a CL-PX2080.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE) bComReg3
LPARAM2	(LPARAM) 0L

RESULT Non-zero if successful, or zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.39 DRV_PXMD_SET_CURSOR_SETUP

DESCRIPTION This message sets the current MediaDAC cursor setup register as specified by the caller. Please note that a call to this function could adversely affect all Windows applications.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE) bCursorSetup
LPARAM2	(LPARAM) 0L

RESULT Non-zero if successful, or zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.40 DRV_PXMD_SET_CURSOR_X_HI_POSITION

DESCRIPTION This message sets the current MediaDAC hardware cursor X hi position as specified by the caller. Please note that a call to this function could adversely affect all Windows applications.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE) bCrsrPos
LPARAM2	(LPARAM) 0L

RESULT Non-zero if successful, or zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

NOTE: The upper nibble of this value is reserved and should always read back 0.

8.2.41 DRV_PXMD_SET_CURSOR_X_LOW_POSITION

DESCRIPTION This message sets the current MediaDAC hardware cursor X low position as specified by the caller. Please note that a call to this function could adversely affect all Windows applications.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE) bCrsrPos
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.42 DRV_PXMD_SET_CURSOR_Y_HI_POSITION

DESCRIPTION This message sets the current MediaDAC hardware cursor Y high position as specified by the caller. Please note, a call to this function could adversely affect all Windows applications.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE) bCrsrPos
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

NOTE: The upper nibble of this value is reserved and should always read back 0.

8.2.43 DRV_PXMD_SET_CURSOR_Y_LOW_POSITION

DESCRIPTION This message sets the current MediaDAC hardware cursor Y low position as specified by the caller. Please note, a call to this function could adversely affect all Windows applications.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE) bCrsrPos;
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.44 DRV_PXMD_SET_EXT_OVERLAY_COLOR

DESCRIPTION This message sets the current MediaDAC overlay color as specified by the caller. It sets all values (as shown below) independently, not assuming any color mode, for example DRV_PXMD_GET_OVERLAY_COLOR and DRV_ PXMD_SET_OVERLAY_ COLOR.

31 23 1615 7 0

XXXXXXXXRRRRRRRRGGGGGGGGBBBBBBBB - where each color component is 8-bits.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (DWORD) dwOverlayColor;
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.45 DRV_PXMD_SET_GAMMA_PALETTE

DESCRIPTION This message sets the current MediaDAC gamma palette as specified by the caller, as well as enabling or disabling the palette. bStatus should be zero to leave the palette enabled, or one to disable the palette. Please note, a call to this function could adversely affect all Windows applications.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (LPGAMMA_PALETTE) lpGammaPalette;
LPARAM2	(LPARAM) (BYTE) bStatus

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.46 DRV_PXMD_SET_GRAPHICS_FORMAT

DESCRIPTION This message sets the current MediaDAC graphics format register as specified by the caller. Please note that a call to this function could adversely affect all Windows applications.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE) bGraphicsFormat;
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.47 DRV_PXMD_SET_OVERLAY_COLOR

DESCRIPTION This message sets the current MediaDAC overlay color as specified by the caller. This message is 256-color-mode-specific.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE) bOverlayColor;
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.48 DRV_PXMD_SET_OVERLAY_MASK

DESCRIPTION This message sets the current MediaDAC overlay mask value as specified by the caller. Please note that a call to this function could adversely affect all Windows applications.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE) bOvlyMask;
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.49 DRV_PXMD_SET_OVERLAY_MODE

DESCRIPTION This message sets the current MediaDAC overlay mode as specified by the caller. It is the lower byte of LPARAM1. This mode defines how the image and graphics data should be mixed and displayed. The valid modes are listed below:

Value	Meaning
GRAPHICS_ONLY	Display no image data.
CHROMA_MATCH_ONLY	Display image where overlay key matches graphics color.
VALID_IMAGE_ONLY	Display image where zoom codes are valid.
CHROMA_AND_VALID_IMAGE	Display image where zoom codes are valid and overlay key matches graphics color.
TAG_ONLY	Display image where image data are tagged.
CHROMA_AND_TAG	Display image where overlay key matches graphics color and image data are tagged.
TAG_AND_VALID_IMAGE	Display image where zoom codes are valid and the image data are tagged.
CHROMA_TAG_AND_VALID_IMAGE	Display image where zoom codes are valid, the overlay key matches the graphics pixel color, and the image data are tagged.
CHROMA_AND_NOT_TAG	Display image where overlay key matches the graphics pixel color and the image data are not tagged.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE) bOverlayMode;
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure

8.2.50 DRV_PXMD_SET_PIXEL_MASK

DESCRIPTION This message sets the current MediaDAC pixel mask as specified by the caller. The pixel mask is AND'ed with the graphics pixels. Please note that a call to this function could adversely affect all Windows applications.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE) bPixelMask;
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.51 DRV_PXMD_SET_RESERVED_1

DESCRIPTION This message sets the current MediaDAC reserved 1 register as specified by the caller. Please note that a call to this function could adversely affect all Windows applications.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE) bRSV1;
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.52 DRV_PXMD_SET_RESERVED_2

DESCRIPTION This message set the current MediaDAC reserved 2 register as specified by the caller. Please note, a call to this function could adversely affect all Windows applications.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE) bRSV2;
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.53 DRV_PXMD_SET_SYNC_ALIGN

DESCRIPTION This message sets the current MediaDAC sync alignment register as specified by the caller. Note that a call to this function could adversely affect all Windows applications.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE) bSAR;
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.54 DRV_PXMD_SET_SYNC_POLARITY

DESCRIPTION This message sets the current MediaDAC sync polarities as specified by the caller. Please note that if the MediaDAC is a CL-PX2085, this message has no effect since the CL-PX2085 auto-detects sync polarities.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE) bVSyncPol;
LPARAM2	(LPARAM) (BYTE) bHSyncPol

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.55 DRV_PXMD_SET_VGA_PALETTE

DESCRIPTION This message sets the current MediaDAC VGA palette as specified by the caller. Please note, a call to this function could adversely affect all Windows applications.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (LPVGA_PALETTE) lpVGAPalette;
LPARAM2	(LPARAM) 0L

RESULT Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.56 DRV_PXMD_SET_VIDEO_FORMAT

DESCRIPTION

This message sets the current MediaDAC video format as specified by the caller. It is the lower byte of LPARAM1.

This message differs slightly from DRV_PXMD_SET_VIDEO_FORMAT_REG in that it sets the format code, while the latter sets the register and VFC directly. The valid codes are defined below:

Value	Meaning
YUV422	16-bit YCrCb (YUV) 4:2:2.
TYUV422	16-bit tagged YCrCb (YUV) 4:2:2.
RGB565	16-bit RGB 5:6:5.
RGB555	16-bit RGB 5:5:5.
TRGB1555	16-bit tagged RGB.
RGB888	32-bit RGB X:8:8:8.
TRGB1888	32-bit tagged RGB 1:8:8:8.

INPUT PARAMETERS

Type	Argument
LPARAM1	(BYTE) bValue;
LPARAM2	(LPARAM) 0L

RESULT

Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

8.2.57 DRV_PXMD_SET_VIDEO_FORMAT_REG

DESCRIPTION

This message sets the current MediaDAC video format register as specified by the caller. It is the lower byte of LPARAM1.

This message differs slightly from DRV_PXMD_SET_VIDEO_FORMAT in that it sets the video register with the value specified, while the latter sets the video format field of the register only.

INPUT PARAMETERS

Type	Argument
LPARAM1	(LPARAM) (BYTE) bVideoFormat;
LPARAM2	(LPARAM) 0L

RESULT

Non-zero if the call was successful; zero if the call failed. Use the DRV_PXMD_GET_ERROR message to get the error code for the failure.

PART III: APPLICATION NOTES

NOTE: These application notes apply *only* to the CL-PX2085.

9. VAFC OVERVIEW

The VESA® has published a new standard for extending the performance and functionality of the 'feature connector' found on many popular graphics controllers. This overview section briefly describes the limitations of the previous feature-connector standard and the advantages of the VAFC™.

9.1 The Standards Gap

The old feature connector found on most VGA-type graphics boards served its intended purpose well. It provided a data path between the graphics card and video-processing cards for such functions as genlocked graphic overlays onto video, and video still-frame capture.

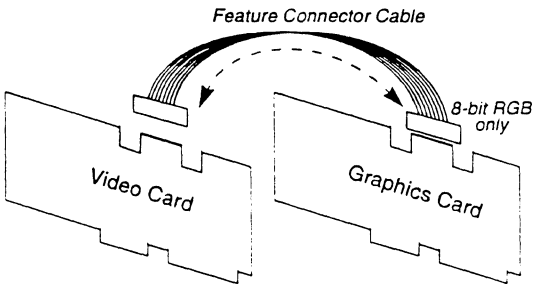


Figure 9-1. 8-Bit Feature Connector

However, as graphic-display resolutions increased, the higher frequencies caused problems for the feature-connector interface. The ribbon cable was not designed for frequencies approaching 100 MHz, and because of the widespread use of gradient shading and photographic images, computer users have recently developed a taste for color resolutions greater than 8 bits.

A typical use for the feature connector is illustrated by one of the first evaluation manufacturing kits provided by Pixel Semiconductor. The Videographer was an ISA-bus example design incorporating the CL-PX0070 (and, later the CL-PX0072) Video Window Generator. In the 'Hot-Key' mode, full-screen, full-motion video data at 30 frames per second could be viewed on a 640 x 480 VGA display. This was accomplished by transmitting the digitized video stream from the Videographer board to the VGA board through the feature connector. Since the color resolution was limited to 8 bits, the intelligent color-interpolation algorithms of the CL-PX0072 were required to bring the quality of the display to an acceptable level. This application consumed most (if not all) the capacity of the 8-bit VESA feature-connector interface.

Some products are available that use multiplexed pixel clocking to achieve 16-bit resolution, but for this to work, both the graphics card and the video card must use the same scheme. Thus, the 8-bit standard interface is changed to a 16-bit proprietary interface with a limited choice of compatible equipment. To make matters worse, EMI levels become more difficult to control. Other systems use the VESA Local Bus VLB to transfer the video data stream, but this consumes host-CPU bandwidth, adversely affecting system throughput. A new standard was clearly needed.

9.2 VAFC to the Rescue!

The VAFC standard resulted from the cooperative efforts of several manufacturers of personal computer equipment to address the technical issues presented by desktop video in multimedia applications. The new standard overcomes many of the system-design problems of multimedia subsystems.

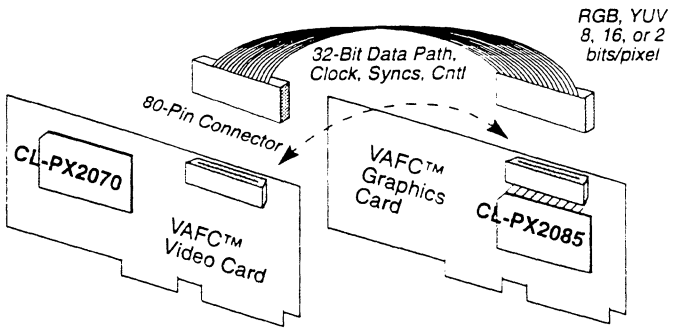


Figure 9-2. VAFC™ Links Separate Video, Graphics Cards

The VAFC standard offers significant advantages to graphics and video subsystem designers such as:

- Published standard — compatible video and graphics cards are now available from several vendors
- 16- or 32-bit data path for higher color resolution and multi-pixel clocking
- Standardized clocking modes to allow multiple-pixel transfer on each clock cycle
- Signals carefully arranged so that the ribbon cable is self-shielding, holding crosstalk to a minimum
- A maximum VAFC clock frequency 37.5 MHz
  - — Allows real-time video at 16-bit color resolution
  - — If YUV data is transmitted over the interface, 24-bit color quality is realized while enjoying the economy of 16-bit per-pixel transmission
- Standardized software drivers allow an application to configure the video and graphics cards to a common, compatible operating mode.

The CL-PX2085 offers an easy path to VAFC compatibility. CL-PX2085 advantages include:

- VAFC-compatible, 32-bit digital video input port with integral FIFO
- 32-bit graphics port, 8-bit VGA port, software selectable
- Real-time video/graphics mixing, multi-window capability
- 135-MHz, triple-8-bit DAC
- Integral zoom, color-space conversion.

Subsequent sections of this application note present techniques for using the CL-PX2085 to implement a VAFC-compatible graphics system, or to upgrade an existing graphics system.

9.3 VAFC™ Connector and Pinout Information

The VESA advanced feature connector is an extension of the previous standard VESA 8-bit feature-connector interface. A 32-bit data path and new control signals were added to facilitate desktop video and multimedia.

The connector for the VAFC port was placed so that pin 80 was located horizontally 2 inches from the front plate, as shown in Figure 9-3. The signal names and pin numbers are shown in Figure 9-4.

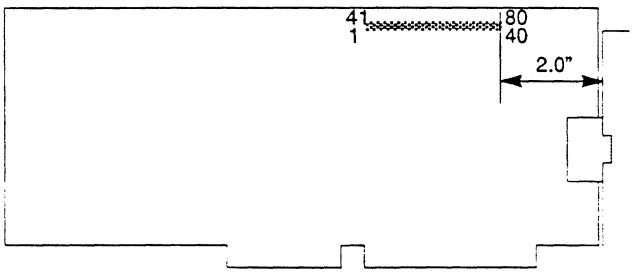


Figure 9-3. VAFC Placement and Orientation

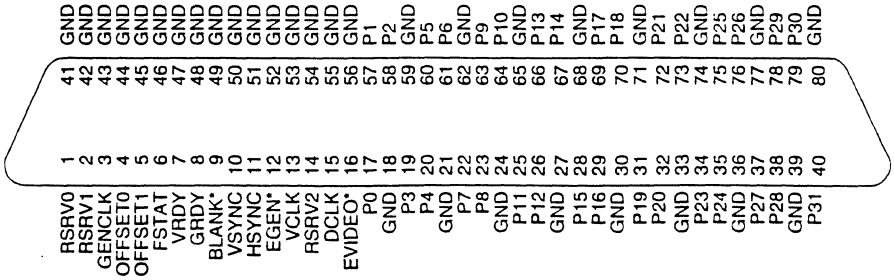


Figure 9-4. VAFC Pin Assignments

10. VAFC™ APPLICATION NOTES

10.1 MediaDAC™ Graphics Subsystem with VAFC™ Port

This section illustrates a general graphics-system architecture and describes some of the implementation options available when using the CL-PX2085 MediaDAC™.

10.1.1 Advantages of CL-PX2085-Based VAFC™ Upgrade Path

- Software compatible with standard PC palette DACs (Bt485 and compatibles)
- No software changes to API, GDI drivers, or BIOS
- No graphics-controller redesign
- The CL-PX2085 interpolates rather than replicates pixels in VAFC 32-bit, 2X, 1280-pixel mode, and in VAFC 16-bit, 2X, 1024-pixel mode
- Compatible with future video boards from many manufacturers, including CL-PX2070 boards with PxVPS software.
  - — PxVPS-equipped boards recognize the CL-PX2085 and fully utilizes its extended features
  - — PxVPS software uses the CL-PX2085 extended register set, then returns the register interface to the 485-compatible mode.

10.1.2 From a Graphics Card Perspective

Now that the VAFC specification is published, graphics board manufacturers can update existing Bt485 (or equivalent) board designs to include a CL-PX2085 with a VAFC port.

10.1.3 From a Multimedia Video Card Perspective

The introduction of the VAFC port allows customers with CL-PX2070/'80 designs (either in the design process or already on the market) an easy option to establish compatibility with graphics boards from many different manufacturers. The existing design can follow a multifunction or a modular approach.

10.1.3.1 Advantages of Single-Board, Multifunction Approach

An existing CL-PX2080-based board design can be upgraded to a CL-PX2085-based full-function, high-performance 135-MHz video/graphics multimedia board. Some advantages include:

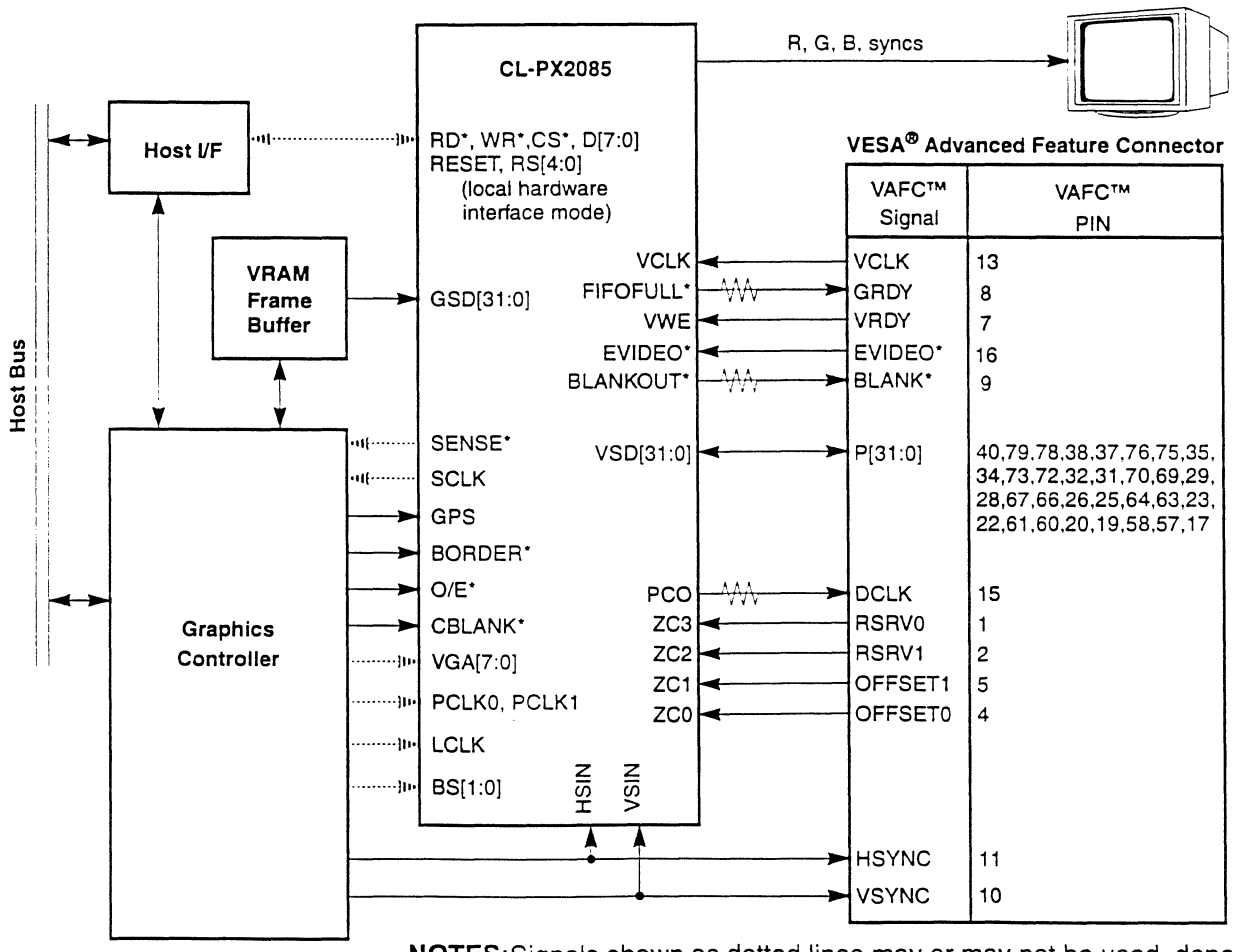
- Lower manufacturing cost
- Fewer connectors, boards, cables; components are designed to work as a system, thus optimizing cost/performance trade-offs
- Improved price/performance ratio
- Easier for end users to incorporate into their systems.

Pixel components bring video functionality to multimedia products, making multifunction boards affordable.

10.1.3.2 Advantages of a Multiple-Board, Component Approach

- The VAFC specification will be implemented by a number of graphics card suppliers. For existing CL-PX2070/CL-PX2080 multimedia board designs, the CL-PX2080 may be removed and replaced with a VAFC port, resulting in a CL-PX2070-based video-processing card. This card is then compatible with a number of CL-PX2085-based graphics cards, and is backward-compatible with application software developed for the previous multimedia board. The cost of adding the VAFC connector is offset by the savings realized by removing the CL-PX2080 from the multimedia board design
- The end user has a video component available that can be paired with the VAFC graphics card that meets his specific needs
- Each function can be upgraded separately
- A malfunctioning component can be replaced more easily than an entire multi-function module, upgrading in the process
- Most upgrades are transparent to the application software.

A multiple-component approach for assembling a multimedia system is available to the end user. This approach provides a flexible, modular system, similar in concept to component audio systems. The same freedom to address specific application needs that the audio consumer has enjoyed is now a reality in the multimedia marketplace.



NOTES: Signals shown as dotted lines may or may not be used, depending on the individual design and type of graphics controller used.

Figure 10-1. CL-PX2085 Example System Block Diagram

10.2 Upgrading from Bt485 to CL-PX2085

When upgrading from the Bt485 to the CL-PX2085, design considerations include signal and register compatibility.

10.2.1 Signal Compatibility

The pins on the two devices may be classified into pin groups, simplifying the device-substitution task. These groups are illustrated in Figure 10-2, and are discussed in Table 10-1. Table 10-2 gives a detailed, pin-by-pin description of signal compatibility between the Bt485 and the CL-PX2085.

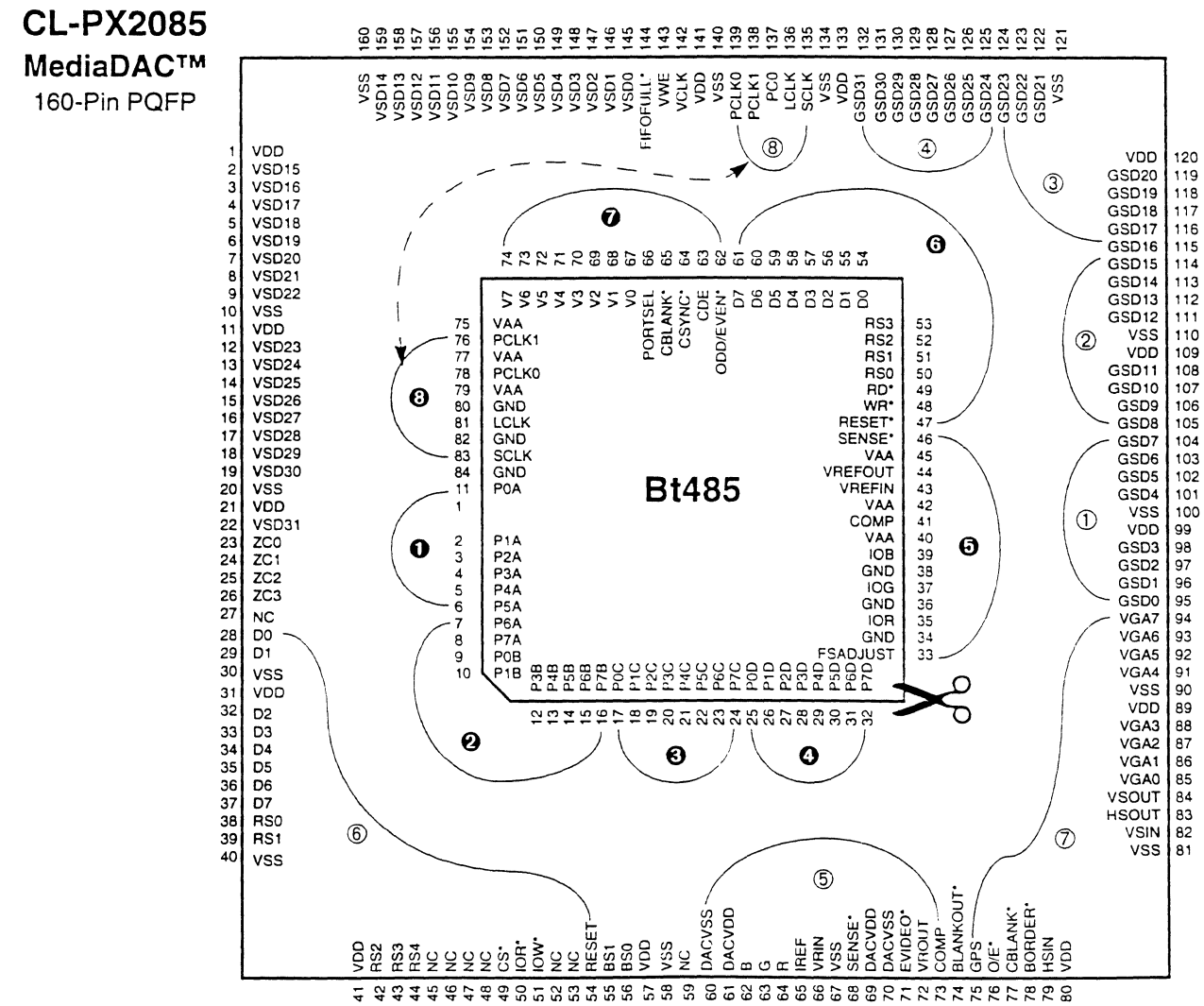


Figure 10-2. Comparative Pin Diagram, CL-PX2085 and Bt485

Table 10-1. Overview — Bt485/CL-PX208x Signal Compatibility

#	Description	Corresponding Signals		Notes
		Bt485	CL-PX2085	
1	Graphics Input Data, Pixel "A"	P7A:P0A	GSD[7:0]	Signals correspond in descending order of bits. When using 8-bit pixels, this is the first pixel of a 32-bit word to be displayed (written to the CRT).
2	Graphics Input Data, Pixel "B"	P7B:P0B	GSD[15:8]	When using 8-bit pixels, this is the second pixel of a 32-bit word to be displayed.
3	Graphics Input Data, Pixel "C"	P7C:P0C	GSD[23:16]	When using 8-bit pixels, this is the third pixel of a 32-bit word to be displayed.
4	Graphics Input Data, Pixel "D"	P7D:P0D	GSD[31:24]	When using 8-bit pixels, this is the fourth pixel of a 32-bit word to be displayed.
5	Analog Monitor Interface	as shown in Figure 10-2 and Table 10-2		The analog connection to one of the popular analog graphics monitors is identical between the two devices. The value of the RSET resistor differs between the two devices. The CL-PX2085 provides a separate set of DACVDD and DACVSS signals for the analog output section.
6	Host Interface	as shown in Figure 10-2 and Table 10-2		The host interface in this example is shown in mode 3 (BS[1:0] are sampled high (11b) immediately after reset) in order to maintain complete software compatibility with the Bt485. RS[4] is not used in this mode; rather, RS[3:0] are used along with an index value programmed into register GCR[REV]. The Bt485 has no chip-select signal. The CL-PX2085 signal CS* is tied to VSS, assuring compatible operation of IOW* (WR*), and IOR* (RD*). For more information, refer to the registeraccess discussion in Section 10.2.2.
7	VGA/Graphics Interface	as shown in Figure 10-2 and Table 10-2		The CL-PX2085 provides a direct replacement for all VGA pins of the Bt485, except CSYNC*. The CL-PX2085 has two additional signals in this group, VSIN and HSIN, used by internal circuitry to align graphics and video frames.
8	Clock Sync	as shown in Figure 10-2 and Table 10-2		CL-PX2085 clock signals are direct replacements for their Bt485 counterparts. In addition, the CL-PX2085 has an output signal PCO that drives the VAFC port signal DCLK through a series-termination resistor.

An existing Bt485 design is enhanced by substituting a CL-PX2085. Similar signals are connected according to Table 10-2. A CL-PX2085 signal is connected to the adjacent Bt485 signal. In the few cases where no direct substitution exists, signals are shown within the same signal group, but on separate rows.

Table 10-2. Detailed Description — Bt485/CL-PX208x Signal Compatibility

CL-PX2085		Bt485			
Signal[a]	**Pin**	**Signal**	**Pin**	**I/O**	**Function**
HOST INTERFACE					
D[7:0]	37:32, 29:28	D[7:0]	61:54	I/O	Data Bus
RS[4:0][b]	44:42, 39:38	RS[3:0]	53:50	I	Register Select
IOR*	50	RD*	49	I	I/O Read
IOW*	51	WR*	48	I	I/O Write
CS*[c]	49	—	—	I	Chip Select
RESET[d]	54	RESET*	47	I	Reset
BS[1:0]	55:56	—	—	I	Bus Select
VIDEO PORT INTERFACE					
VSD[31:0]	22, 19:12, 9:2, 159:145	—	—	I	Video Source Data
ZC[3:0]	26:23	—	—	I	Zoom Control Code
FIFOFULL*	144	—	—	O	FIFO Full
VCLK	142	—	—	I	Video Data Clock
VWE	143	—	—	I	Video FIFO Write Enable
EVIDEO*	71	—	—	I	Enable Video Input
CLOCK SYNC					
LCLK	136	LCLK	81	I	Latch Clock Input
SCLK	135	SCLK	83	O	VRAM Shift Clock Output
PCLK0	139	PCLK0	78	I	Pixel Input Clock 0
PCLK1	138	PCLK1	76	I	Pixel Input Clock 1
PCO	137	—	—	O	Pixel Clock Output
GRAPHICS PORT INTERFACE					
GSD[31:24]	132:125	P[7:0]D	32:25	I	Graphics Source Data
GSD[23:16]	124:122, 119:115	P[7:0]C	24:17	I	(VRAM)[e]
GSD[15:8]	114:111, 108:105	P[7:0]B	16:9	I	
GSD[7:0]	104:101, 98:95	P[7:0]A	8:1	I	
VGA[7:0]	94:91, 88:85	VGA[7:0]	74:67	I	VGA Graphics Source Data
O/E*	76	ODD/EVEN*	62	I	Odd/Even Field Input
GPS	75	PORTSEL	66	I	Graphics Port Select
BORDER*	78	CDE	63	I	Active Display Border
CBLANK*	77	CBLANK*	65	I	Composite Blank Input
VSIN	82	—	—	I	Vertical Sync Input
HSIN	79	—	—	I	Horizontal Sync Input
—	—	CSYNC*	64		Composite Sync Input

Table 10-2. Detailed Description — Bt485/CL-PX208x Signal Compatibility *(cont.)*

CL-PX2085		Bt485			
Signal[a]	**Pin**	**Signal**	**Pin**	**I/O**	**Function *(cont.)***
MONITOR INTERFACE					
IREF	65	FSADJUST	33	I	Current Reference[f]
R	64	IOR	35	O	Analog Red
G	63	IOG	37	O	Analog Green
B	62	IOB	39	O	Analog Blue
VRIN	66	VREFIN	43	I	Voltage Reference In[c]
VROUT	72	VREFOUT	44	O	Voltage Reference Out[g]
SENSE*	68	SENSE*	46	O	Monitor Sense
VSOUT	84	—	—	O	Vertical Sync Output
HSOUT	83	—	—	O	Horizontal Sync Output
COMP	73	COMP	41	O	Compensation
BLANKOUT*	74	—	—	O	Blank Out
POWER					
VDD	1, 11, 21, 31, 41, 57, 80, 89, 99, 109, 120, 133, 141	VAA	40, 42, 45, 75, 77, 79		+5 VDC for Digital Logic and Interface Buffers
VSS	10, 20, 30, 40, 58, 67, 81, 90, 100, 110, 121, 134, 140, 160	GND	34, 36, 38, 80, 82, 84		Ground for Digital Logic and Interface Buffers
DACVDD	61, 69		—		+5 VDC for DAC[h]
DACVSS	60, 70		—		Ground for DAC

a. In this comparison, the CL-PX2085 is shown in local hardware interface mode. The CL-PX2085 also directly interfaces to ISA and MicroChannel® buses.
b. The full CL-PX2085 extended register set is accessible through two local-bus addressing schemes, one using RS[4:0], and the other using RS[3:0] (plus an offset address loaded in a register).
c. IOR* directly replaces RD*, and IOW* directly replaces WR* if CS* is tied to VSS (ground).
d. The CL-PX2085 RESET signal is active-high. It must be inverted as part of the Bt485 substitution. In ISA-bus-based designs, RESET can be driven directly from the ISA bus.
e. The GSD bus canbe programmed to form the 'Display Output Data' bus. See the CSC register description in Section 4.4.2 of the *CL-PX2085 Data Book*.
f. The value of the RSET resistor varies between the CL-PX2085 and the Bt485.
g. The operation of VROUT has not been fully characterized at the time of printing. Until this information is published, it is recommended that VROUT be left floating and that VRIN be driven from an external, 1.23 V precision voltage-reference circuit, as shown in Figure 10-3 on page 141.
h. Refer to Section 10.4.5 on page 159 for recommended treatment.

10.2.2 Register Compatibility

The CL-PX2085 is fully register-compatible with the Bt485. Even though the CL-PX2085 register set is more extensive, no software changes are required when upgrading to the CL-PX2085. The PxVPS identifies a CL-PX2085 when present, and can access the extended CL-PX2085 registers, then return the device to a Bt485-compatible mode.

The register-access Mode B (signals BS[1:0]=11) allows software drivers written for the Bt485 to run successfully in a CL-PX2085-based system. This mode uses only RS[3:0] and a block index in register GCR[BLK] to address the entire extended register set. Equivalent CL-PX2085 register functions appear at the same addresses as in the Bt485.

PxVPS software, running on a separate CL-PX2070-based video board and connected to the CL-PX2085 through the VAFC port, is able to identify that the CL-PX2085 is present. In this scenario, when PxVPS accesses the extended CL-PX2085 registers, the interface is returned to a Bt485-compatible mode after the PxVPS access. Drivers written for the Bt485 and the advanced PxVPS software modules run on the same system without interfering with each other.

10.2.2.1 GCR/CR3 Access

The CL-PX2085 register GCR is the equivalent function to the Bt485 register CR3. To access GCR, which is at the same physical I/O and index address as register GSR (read-only), write a '1' to register bit ASC[GCRE].

The access method of the Bt485 requires that a '1' to be written to CR0, bit 7, then 01h to the address register (equivalent to the CL-PX2085 register LAW) before register CR3 can be accessed.

When Bt485 software runs on a CL-PX2085-based system, the address written to register LAW is ignored, and register GCR is accessed transparently.

Table 10-3 compares the register maps and register-access modes of the CL-PX2085 and Bt485.

Table 10-3. Bt485/CL-PX2085 Register Compatibility

CL-Px2085 Name	CL-Px2085 Function	Bt485	GCR [7:4]	RS [3:0]
Memory Access Addressing and Indexing				
BIR	Block Index Register			
Graphics Color Palette RAM Registers[a]				
LAW	LUT Address Write	palette wr addr	N/A	0
LCD	LUT Color Data	palette data	N/A	1
LPM	LUT Pixel Mask	pixel mask	N/A	2
LAR	LUT Address Read	palette rd addr	N/A	3
Cursor Color/Border RAM / Analog Setup Registers				
CAW	Cursor Address Write	wr addr, cursor/overscan color	0	4
CCD	Cursor Color Data	cursor/overscan color data	0	5
ASC	Analog Setup Control	command register 0	0	6
CAR	Cursor Address Read	rd addr, cursor/overscan color	0	7

Table 10-3. Bt485/CL-PX2085 Register Compatibility *(cont.)*

CL-Px2085 Name	CL-Px2085 Function	Bt485	GCR [7:4]	RS [3:0]
Graphic and Cursor Setup Registers				
GFC	Graphics Format Control	command register 1	N/A	8
CSC	Cursor Setup Control	command register 2	N/A	9
GSR[b] /GCR[b]	Graphics Status Register/ General Configuration Register	status register (CR3)	N/A	A
CPR	Cursor Pattern RAM Data	cursor RAM array data	N/A	B
Cursor Positioning Registers				
CXL	Cursor X Position, LSB	cursor X low register	N/A	C
CXH	Cursor X Position, MSB	cursor X high register	N/A	D
CYL	Cursor Y Position, LSB	cursor Y low register	N/A	E
CYH	Cursor Y Position, MSB	cursor Y high register	N/A	F
Video, Graphics, and Sync Control Registers				
VFC	Video Format Control	—	4	4
GOC	Graphics Overlay Opcode	—	4	5
SAR	Sync Alignment Register	—	4	6
TEST	Test Register	—	4	7
Video Gamma Correction Palette RAM Registers				
VGW	Video Gamma Address Write	—	5	4
VGD	Video Gamma Data	—	5	5
VAFC VESA Advanced Feature Connec-tor	VAFC VESA Advanced Feature Connector	—	5	6
VGR	Video Gamma Address Read	—	5	7
Graphics Chroma Key Registers				
GCKR	GCK Red	—	6	4
GCKG	GCK Green	—	6	5
GCKB	GCK Blue	—	6	6
GKMR	GCK Mask Red	—	7	4
GKMG	GCK Mask Green	—	7	5
GKMB	GCK Mask Blue	—	7	6

a. The indexed registers are mapped to these addresses when ASC[GCRE] is enabled, and when BIR[BLK] contains the appropriate value.
b. GSR and GCR share a register address. GSR is enabled when ASC[GCRE] = 0; GCR is enabled when ASC[GCRE] = 1.

10.3 CL-PX2085 Graphics Programming Considerations

This section explains programming considerations affecting CPU-to-I/O-addressed registers.

10.3.1 Register Access Sequence

Accesses to Graphics Color Palette RAM or cursor Color/Border RAM registers must be separate address-then-data operations. Intermixed accesses will result in incorrect data, as with sequences similar to the following:

write LAW @ 00h write LCD @ 01h write CCD @ 05h write CCD @ 05h	Incorrect RGB component data is written to the Cursor Color registers.	write CAW @ 04h write CCD @ 05h write CCD @ 05h write LCD @ 01h	Incorrect RGB component data is written to the palette.

It is recommended that register CAW always be used to address register CCD, and that LAW be used to address register LCD. However, the following access sequences result in valid data written to the registers:

write LAW @ 00h write CCD @ 05h write CCD @ 05h write CCD @ 05h	Correct RGB component data is written to the Cursor Color register. Not recommended.	write LAW @ 00h write LCD @ 01h write LCD @ 01h write LCD @ 01h	Correct RGB component data is written to the palette. Recommended.
write LAW @ 00h write CPR @ 0Bh	Correct data is written to Cursor RAM. Not recommended	write CAW @ 04h write CPR @ 0Bh	Correct data is written to Cursor RAM. Recommended.
write CAW @ 04h write CCD @ 05h write CCD @ 05h write CCD @ 05h	Correct RGB component data is written to the Cursor Color register. Recommended.	write CAW @ 04h write LCD @ 01h write LCD @ 01h write LCD @ 01h	Correct RGB component data is written to the palette. Not recommended.

The Cursor RAM cannot be read using register CAR as the address pointer. Addressing must be done with LAR. For example:

write LAR @ 03h read CPR @ 0Bh	Valid.	write CAR @ 07h read CPR @ 0Bh	Invalid.

10.3.1.1 Accessing the Graphics/Cursor Color Data In 6- or 8-Bit DAC Mode

The CL-PX2085 can be configured for 6- or 8-bits per DAC. These modes must not be mixed. If the DAC mode is changed, the Graphics Color Pallette RAM/Cursor Color/Border RAM registers must be reloaded for the appropriate mode.

10.3.1.2 Address Pointer Reads by the CPU

Cursor registers CAR/CAW have no internal or functional connection to registers LAW/LAR, except for the two LSBs in each. A read/write operation from CAR/CAW will not return the same 8-bit value as a read/write from LAW/LAR. The internal Cursor Color address registers (CAR/CAW) contain the two MSBs. The value returned is {XXXXXX, [bit1], [bit2]}.

10.3.1.3 Palette and Cursor RAM Access, Internal Clock Disabled

The CL-PX2085 internal clock is disabled (forced high) when register ASC[COFF,DOFF] = 1. When the internal clock is off, all I/O and Cursor Color/Border RAM registers can be accessed, but no accesses are allowed to the Graphics Color Pallette and Cursor Color/Border RAMs. The Graphics Color Pallette and Cursor Color/Border RAMs maintain data integrity during disabled-clock operation.

10.3.1.4 VGA Port When Palette Bypass is Enabled

When the VGA port is selected (that is ,PORTSEL = 0) and the palette bypass enabled, VGA data bypasses instead of addresses the palette. To ensure that VGA data addresses the palette, palette bypass should be disabled when the VGA port is selected. Consider the example of a Windows application operating with a true-color (palette bypassed) mode. If a full-screen DOS shell is opened, the driver should reenable the color-palette access (that is disable bypass mode) so that VGA data addresses the palette. When the DOS shell is closed, the driver should reinstate the palette-bypass mode.

10.3.1.5 Accessing the Bt485 Command Register 3/CL-PX2085 Status Register

Bt485 Operation:

The following sequence is recommended to access Command Register 3:

1. Set Command register 0 (0x83c6), bit 7 to "1".
2. Write 01h to Palette Write Address register (0x03c8).[1]
3. Read/Write Command Register 3 through Status register address (0x13c6).

With the Bt485, when Command register 0, bit 7 = 1 and the clock double bit in Command register 3 is enabled, the external PCLK selected is doubled internally. When in the clock double mode and Command register 0, bit 7 is then reset to '0', internal clock doubling is disabled even though Command register 3 has not been modified with an I/O write. This means that when in the clock doubled mode, the Status register cannot be read (that is, Command register 0, bit 7 = 0 and a read access to the Status register (0x13c6)) while maintaining internal clock doubling because Command register 0, bit 7 = 0, thus disabling internal clock doubling. Command register 3, bit 3 (the clock double enable bit) does not behave like a "normal" register bit in that what is written to the bit is not retained regardless of the state of Command register 3, bit 7. To get around this problem another qualifier must be added so that the Status register can be read when in clock double mode. This qualifier is the decode of the Palette Write Address register, as previously described.

Px2085 Operation:

For the CL-PX2085, bit 3 in Command register 3 behaves like a "normal" register bit in that whatever is written is retained regardless of the state of Command register 0, bit 7. To write/read Command register 3, Command register 0, bit 7 should = 1, and a write/read to/from the Status register address (0x13c6) should be performed. To read the Status register, Command register 0, bit 7 should = 0, and a read from the Status register address (0x13c6) should be performed. The CL-PX2085 does not have a qualifier based on the decode of the Palette Write Address register.

Consequence:

- If the true Status register is read in clock double mode with the Bt485 operation sequence described above, the Px2085 will return the state of Command register 3.
- If system software relies on Command Register 0, bit 7, disabling the internal clock doubling with Command register 3, bit 3 set to "1", the Px2085 will maintain internal clock doubling.

1. Write 00h to Palette Write Address register (0x03c8) to read Status register via 0x13c6 when bit 7 of Command Register 0 is "1."

10.4 Case Study: Converting a Design to the CL-PX2085

Weitek® Corporation produced a board to demonstrate the capabilities of the Power 9000 graphics coprocessor. The design described in this application note was released by Weitek Corporation as manufacturing kit, number 4800-0098-00, Rev. 01. This board essentially has two graphics subsystems — a VGA-compatible W5286 subsystem with a 1-Mbyte DRAM frame buffer, and the Power 9000 subsystem with a separate 1- or 2-Mbyte frame buffer. The two subsystems share the same palette DAC. The original design used a Bt485 palette DAC. Application engineers at Pixel Semiconductor altered the design, replacing the Bt485 with a CL-PX2085 MediaDAC and adding a VAFC in the process. This upgrade was accomplished with no modifications to BIOS or software drivers.

This section is divided into four subsections.

- Section 10.4.1 provides a brief description of the overall design, with emphasis on those parts of the circuit common to both designs.
- Section describes the interface to the Bt485, as it was in the original design.
- Section 10.4.3 describes the design changes to accommodate the interface to the CL-PX2085 MediaDAC.
- Section 10.4.4 briefly lists the steps taken to update the Weitek board design.

10.4.1 General Design Description

This section identifies parts of the board that remained the same in both designs. The general circuit description is based on the block diagram in Figure 10-3. References to specific schematic pages are given in the following general format:

page nn(nn)
- page number in Bt485 version
- page number in CL-PX2085 version

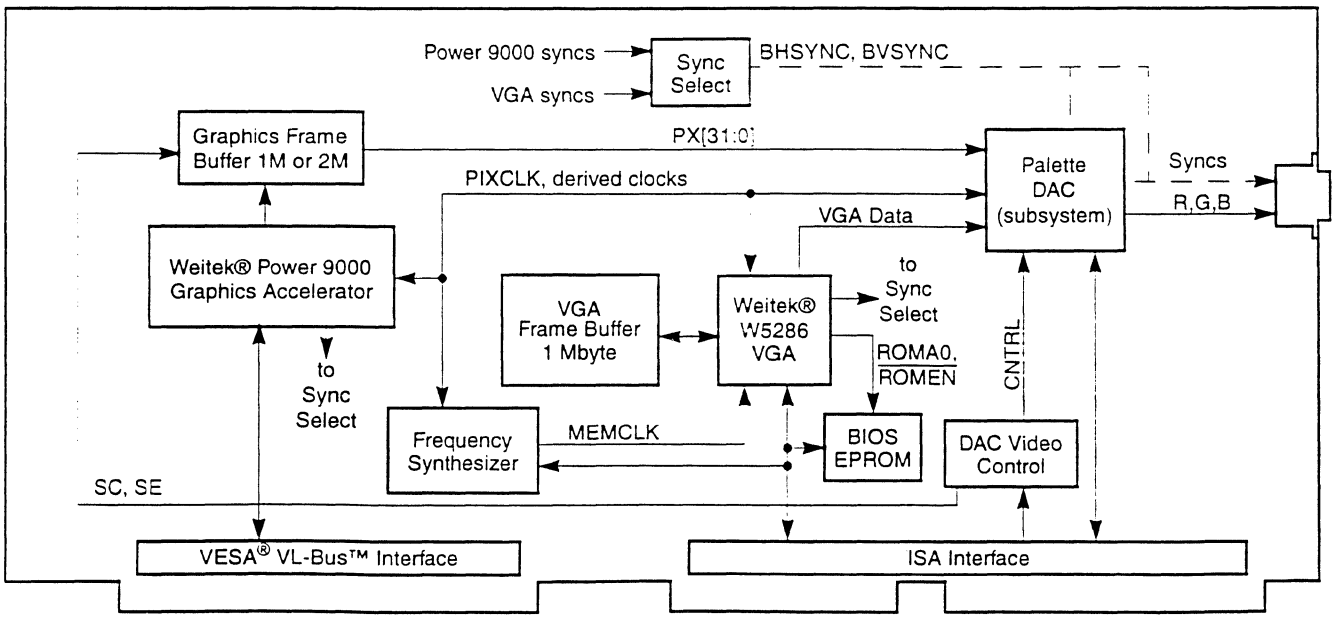


Figure 10-3. Block Diagram, Functional Blocks Common to Both Designs

10.4.1.1 Host Interface

The board is controlled by the host system through ISA and VESA® VL-Bus™ interfaces.

ISA Interface

The ISA interface controls most of the functions on the board. These are the:

- Palette DAC
- DAC video control
- VGA subsystem
- Frequency synthesizer
- BIOS ROM access

The interface design uses two 74LS245 bus transceivers on the ISA data lines. A 74F38 provides open-collector ISA outputs as needed. A 16L8-15 PLD decodes control signals for the palette DAC and bus transceivers. ISA address lines connect directly to the BIOS EPROM, the VGA controller, the ISA RAM-DAC interface, and the palette DAC.

VESA® Local-Bus Interface

A separate VESA® VL-Bus™ interface provides high-speed host access to the Weitek Power 9000 graphics accelerator and associated frame buffer. The interface is contained primarily in a 22V10 PLD. Address and data lines connect directly between the Power 9000 and the VESA® VL-Bus™.

10.4.1.2 DAC/Video Control

A 16L8-15 PLD decodes control signals $\overline{\text{DACWR}}$, $\overline{\text{DACRD}}$, $\overline{\text{BUSOUT}}$, and $\overline{\text{ISABUSOUT}}$. A 16R4-15 PLD decodes and registers S/CK and S/D, used by the frequency synthesizer, P9000SYNCPOLARITY, and P9000VIDEOENABLE.

10.4.1.3 VGA Subsystem

The Weitek W5286 VGA controller provides backward compatibility with lower-resolution VGA modes. It is clocked by MEMCLK and PIXCLK from the frequency synthesizer. Eight-bit VGA data (VIDOUT[7:0]), drives the VGA input of the palette DAC. Sync signals are passed through a sync-select PLD to the palette-DAC subsystem.

The VGA subsystem has a separate 256K x 32 frame buffer.

10.4.1.4 Frequency Synthesizer

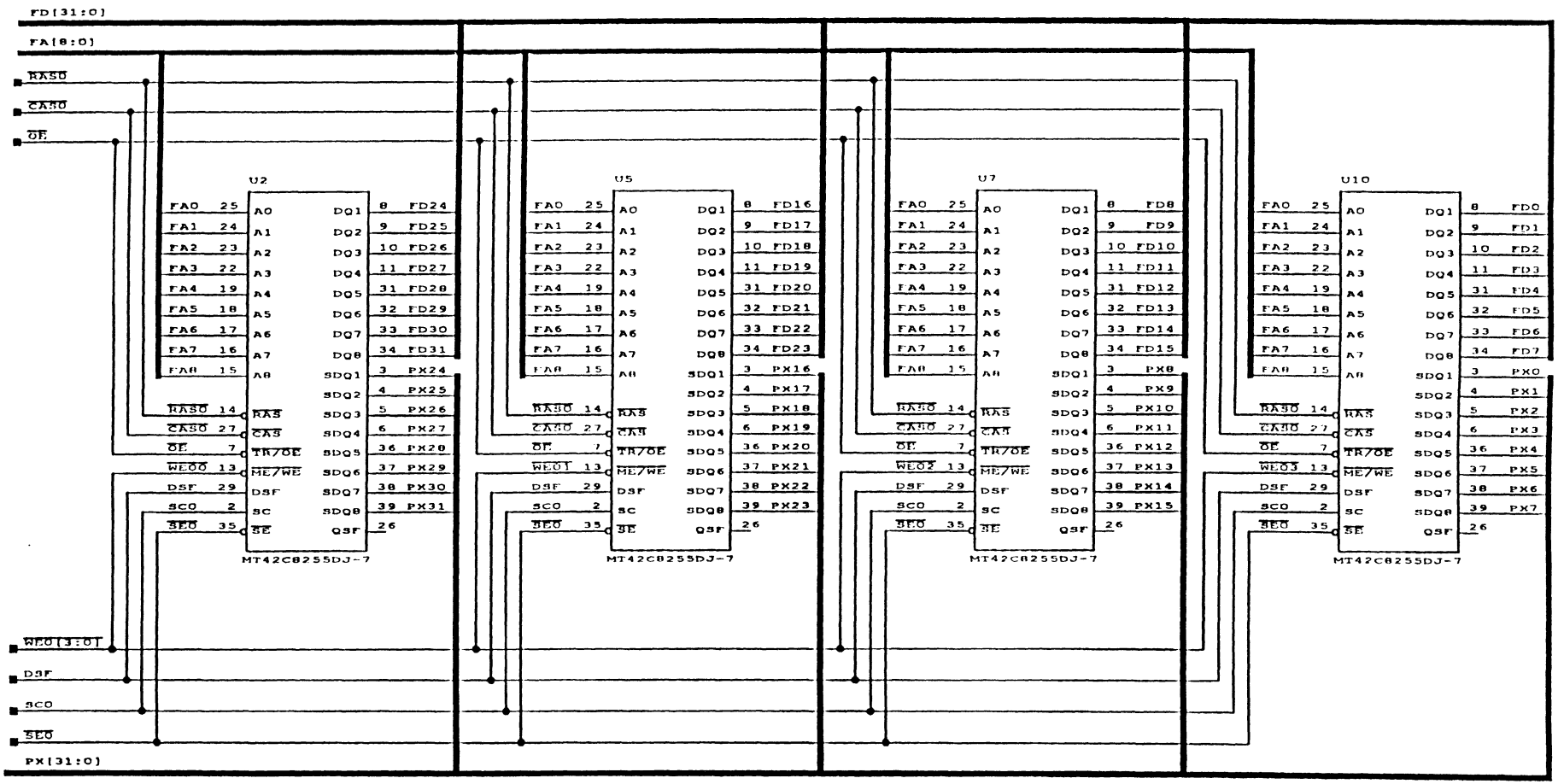
An ICD2061 device using a 14.31818-MHz crystal generates MEMCLK and PIXCLK, which are series terminated by 33-Ω resistors. An analog supply, AVDD, is provided by a 78L05 series-pass regulator from the +12-V supply. S/CK and S/D are generated by a DAC/Video control PLD.

10.4.1.5 BIOS ROM Access

ROMA0 and $\overline{\text{ROMEN}}$ are generated by the W5286 VGA controller. SA14-SA1 are connected to the remaining EPROM address lines.

10.4.1.6 Power 9000 Graphics Accelerator Subsystem

The Weitek Power 9000 graphics accelerator communicates with the host system through the VESA® VL-Bus™ interface. A separate power pin for the internal phase-locked loop is driven by a 78L05 series-pass regulator from the +12-V supply. The control signals for the VRAM frame buffer are series-terminated by 33-Ω resistors.

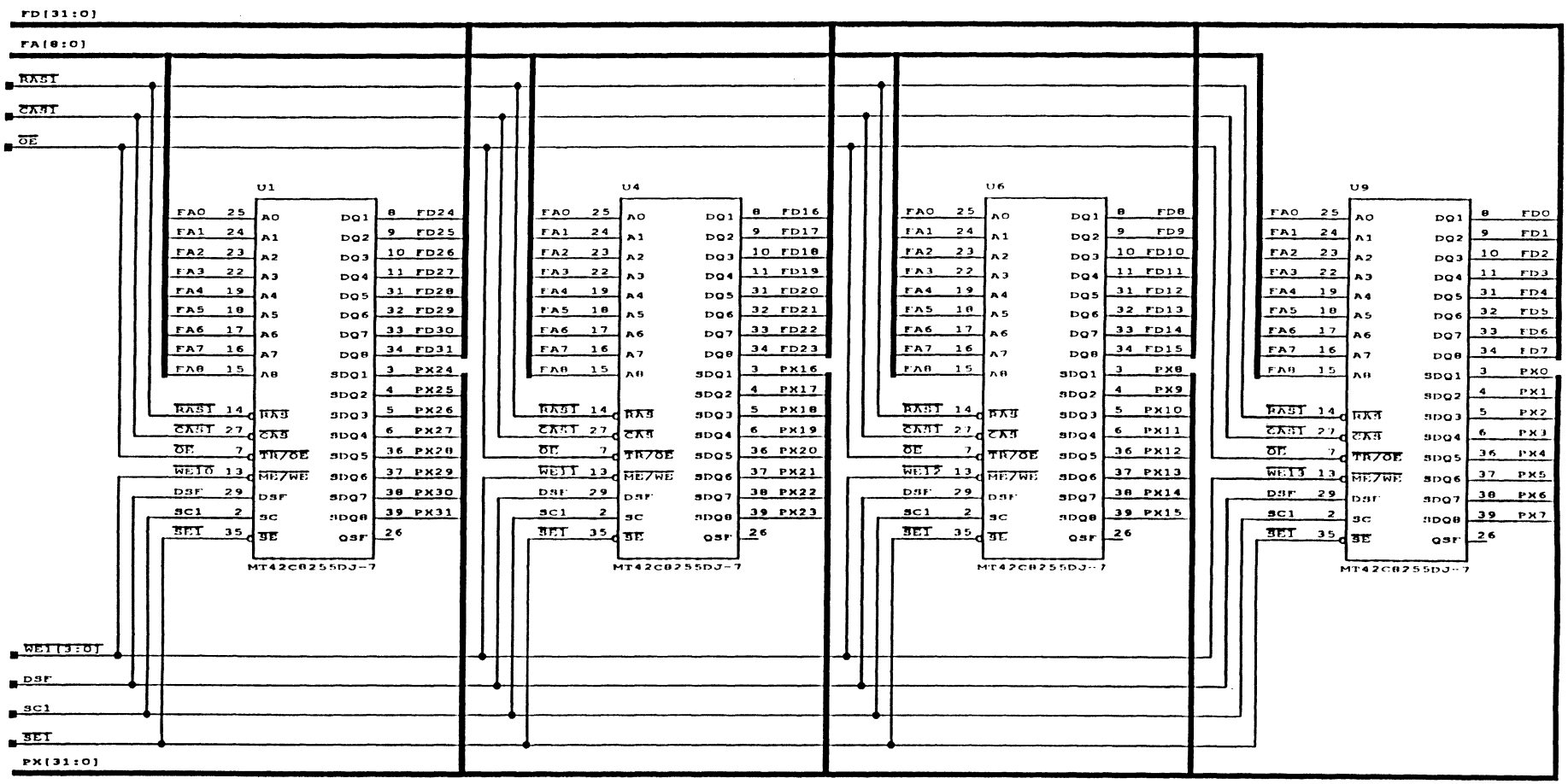


2MB Frame buffer bank 0

Pixel Semiconductor

DWG Title: 2085 EVALUATION BOARD

Size: B	Drawn By:	Rev: APP
Date: 11/05/93		Page: 1 OF 11



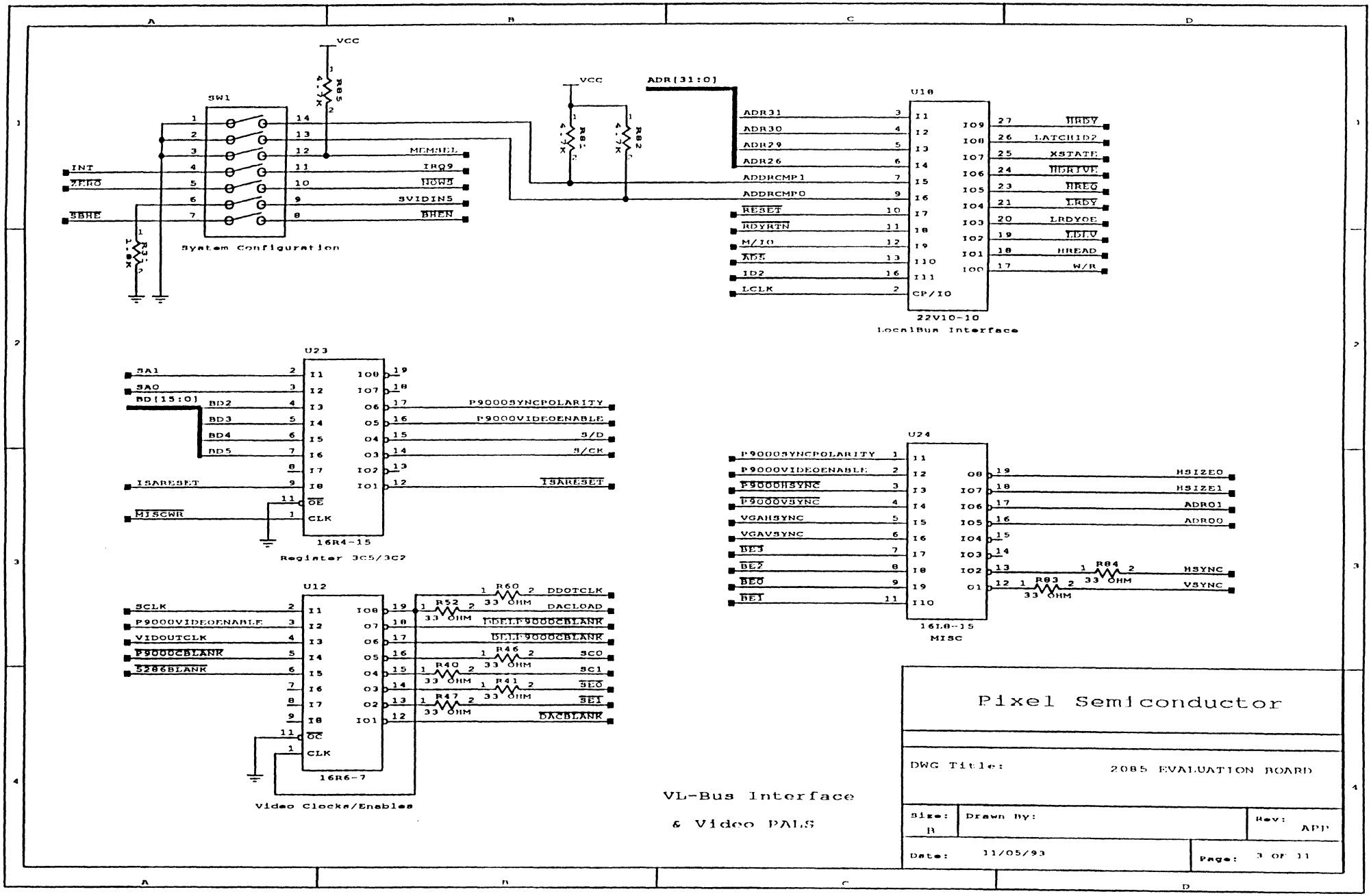
2MB Frame buffer bank 1

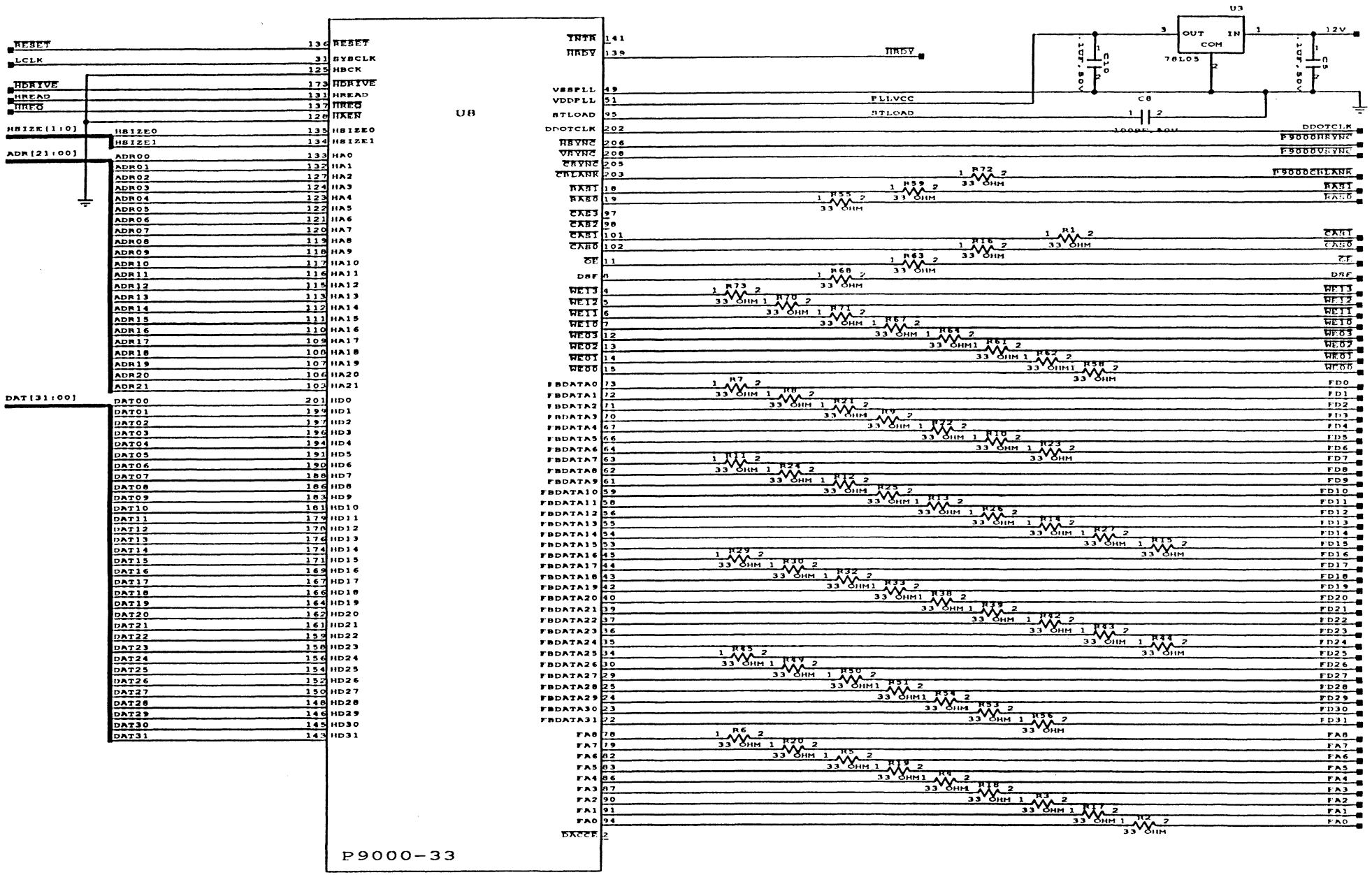
Pixel Semiconductor

DWG Title: 2085 EVALUATION BOARD

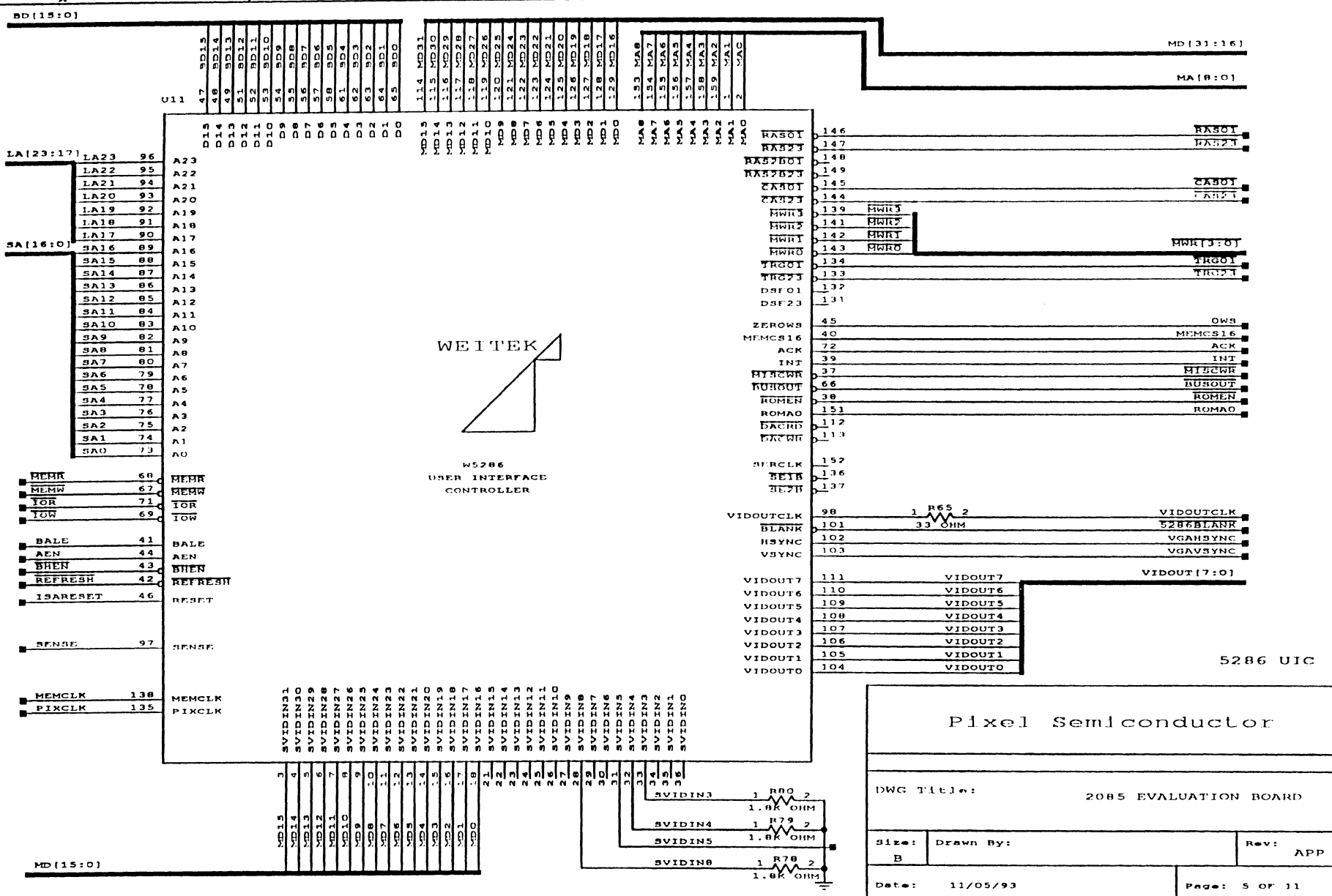
Size: B | Drawn By: | Rev: APP

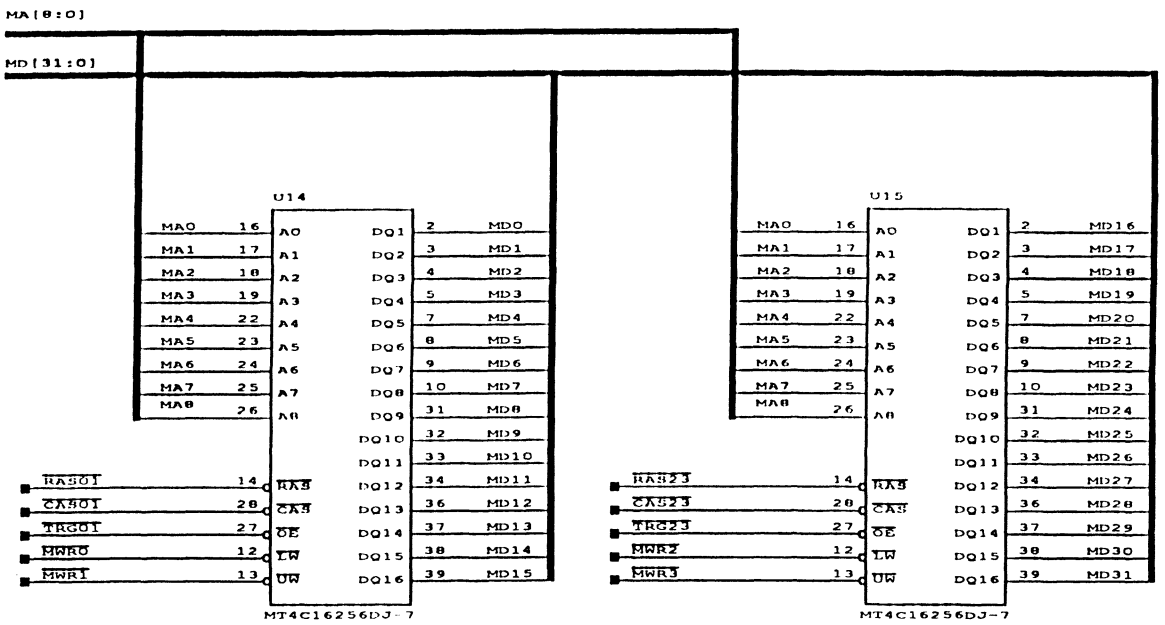
Date: 11/05/93 | Page: 2 OF 11


VCC
SW1
R85
4.7K
INT
ZERO
SBHE
MEMSEL
IRQ9
NOWS
SVIDIN5
BHEN
System Configuration
R81
R82
R83
1.0K
ADR[31:0]
ADR31
ADR30
ADR29
ADR26
ADDRCMP1
ADDRCMP0
RESET
RDYRTN
M/IO
ADS
ID2
LCLK
U18
22V10-10
LocalBus Interface
HRDY
LATCHID2
XSTATE
HDRIVE
HREQ
LRDY
LRDYOE
LDIV
HREAD
W/R
SA1
SA0
BD[15:0]
BD2
BD3
BD4
BD5
ISARESET
MISCWR
U23
16R4-15
Register 3C5/3C7
P9000SYNCPOLARITY
P9000VIDEOENABLE
S/D
S/CK
ISARESET
U24
P9000HSYNC
P9000VSYNC
VGAHSYNC
VGAVSYNC
BE3
BE2
BE0
BE1
16L8-15
MISC
HSIZE0
HSIZE1
ADR01
ADR00
R84
R83
33 OHM
HSYNC
VSYNC
SCLK
VIDOUTCLK
P9000CBLANK
5286BLANK
U12
16R6-7
Video Clocks/Enables
R60
R52
R46
R40
R41
R47
DDOTCLK
DACLOAD
DDELP9000CBLANK
SC0
SC1
SE0
SE1
DACBLANK
VL-Bus Interface
& Video PALS
Pixel Semiconductor
DWG Title:
2085 EVALUATION BOARD
Size:
B
Drawn By:
Rev:
APP
Date:
11/05/93
Page:
3 OF 11



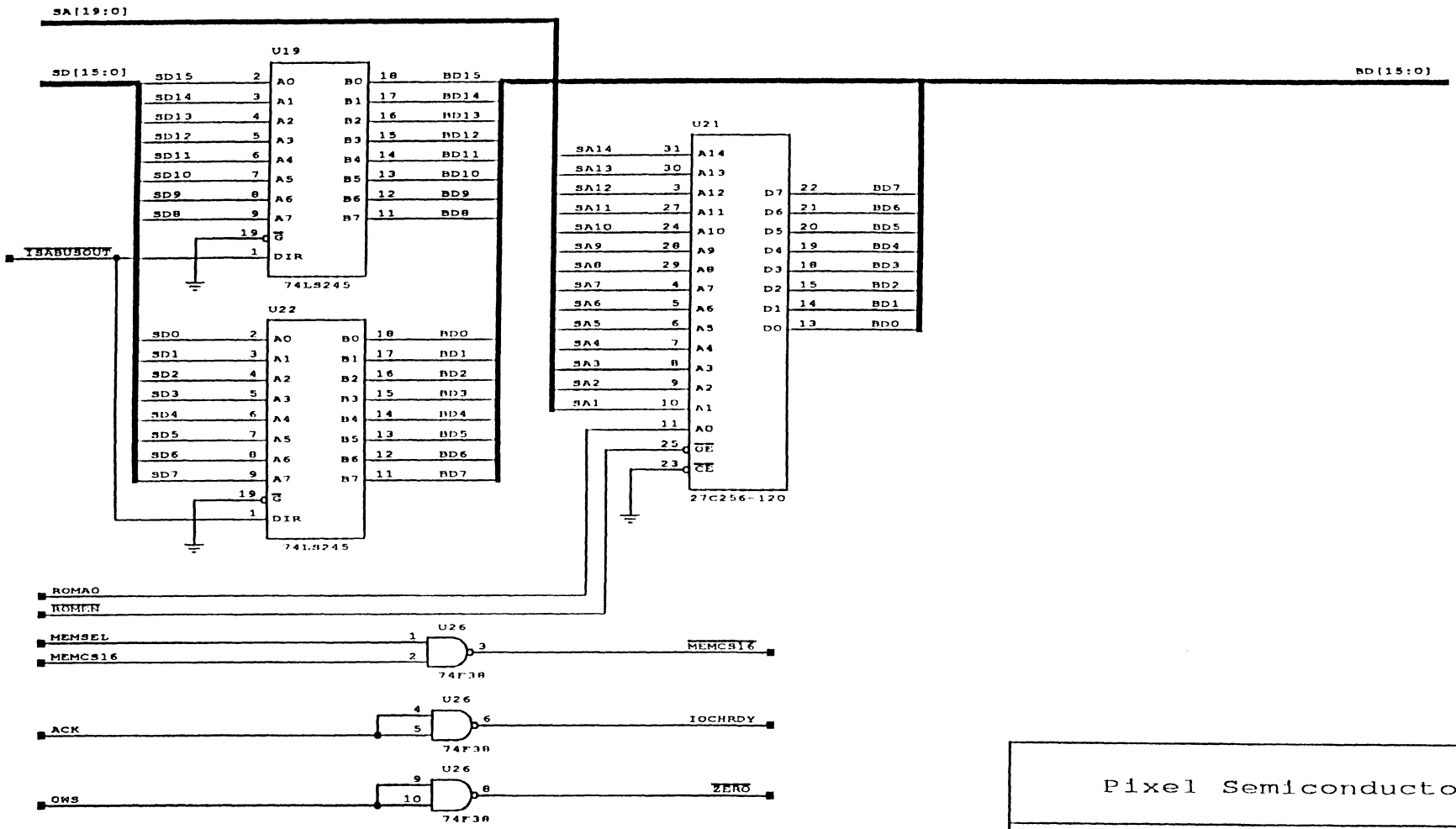
208PQFP
1
STD
9000-033-PFP
WEITEK, 9000-033-PFP
IC, 9000-033-PFP,GUI ACCEL.
GND;1,9,16,20,26,28,32,38,46,47,57,65,68,74,75,77,81,85,89,93,99
GND;105,114,126,130,140,144,149,151,155,160,165,170,172,177,182,185,189,193,195,200,204
VDD;3,10,17,21,27,33,41,52,60,69,76,80,84,88,92,96,100,104
VDD;129,138,142,147,153,157,163,168,175,180,184,187,192,198,207

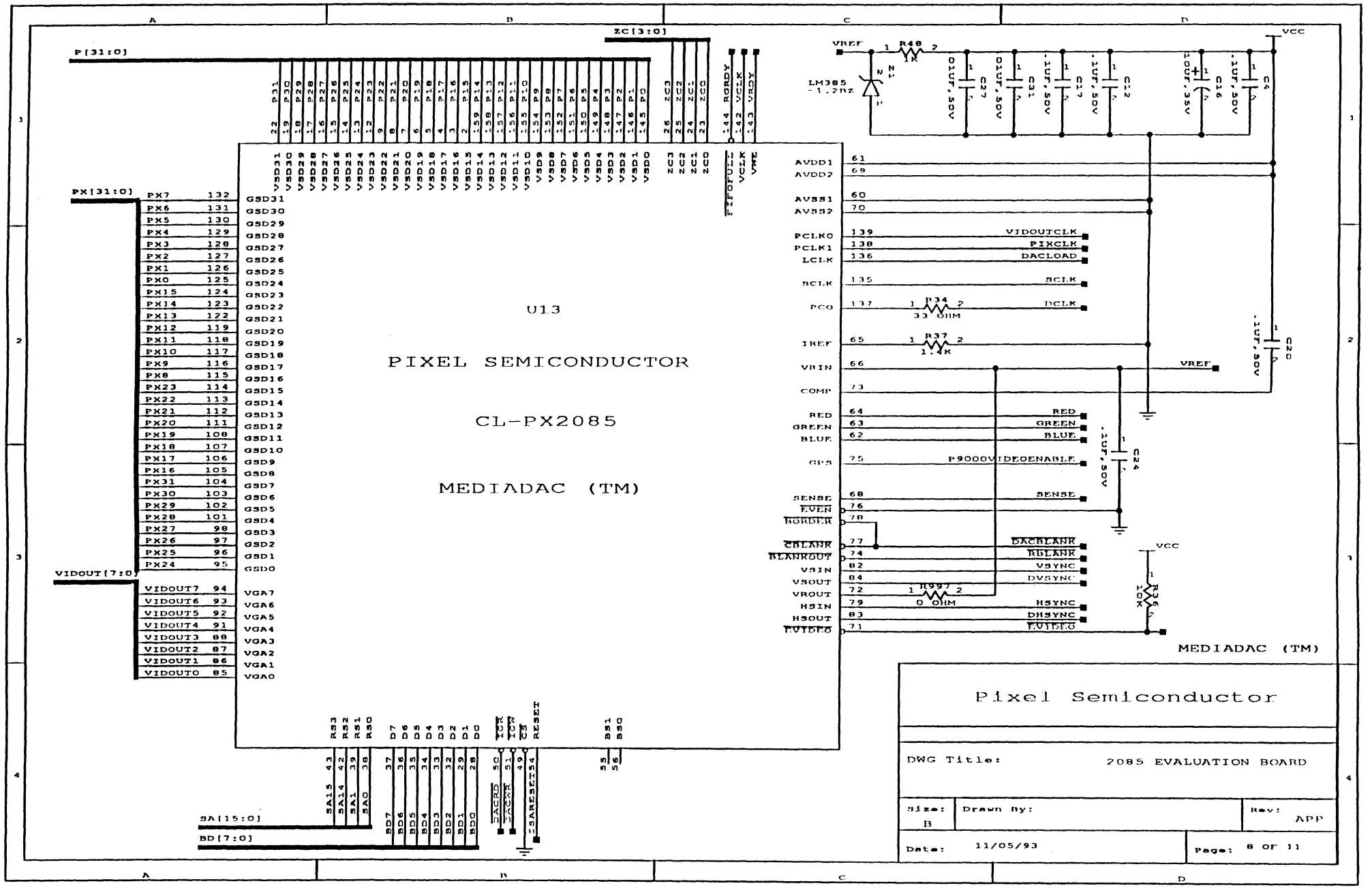

BD[15:0]
MD[31:16]
MA[8:0]
U11
LA[23:17]
SA[16:0]
WEITEK
W5286
USER INTERFACE
CONTROLLER
MD[15:0]
5286 UIC
Pixel Semiconductor
DWG Title: 2085 EVALUATION BOARD
Size: B
Drawn By:
Rev: APP
Date: 11/05/93
Page: 5 OF 11

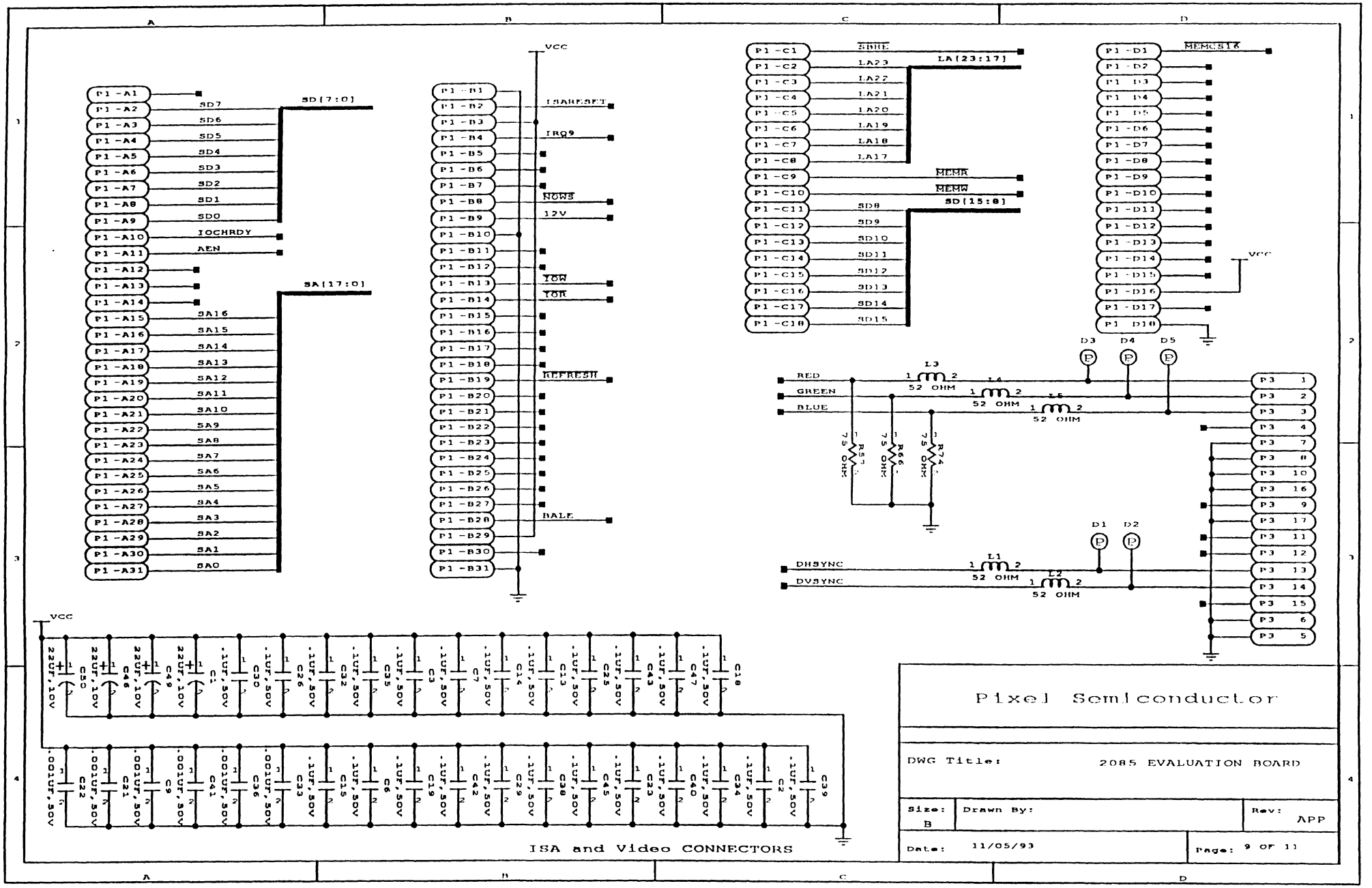


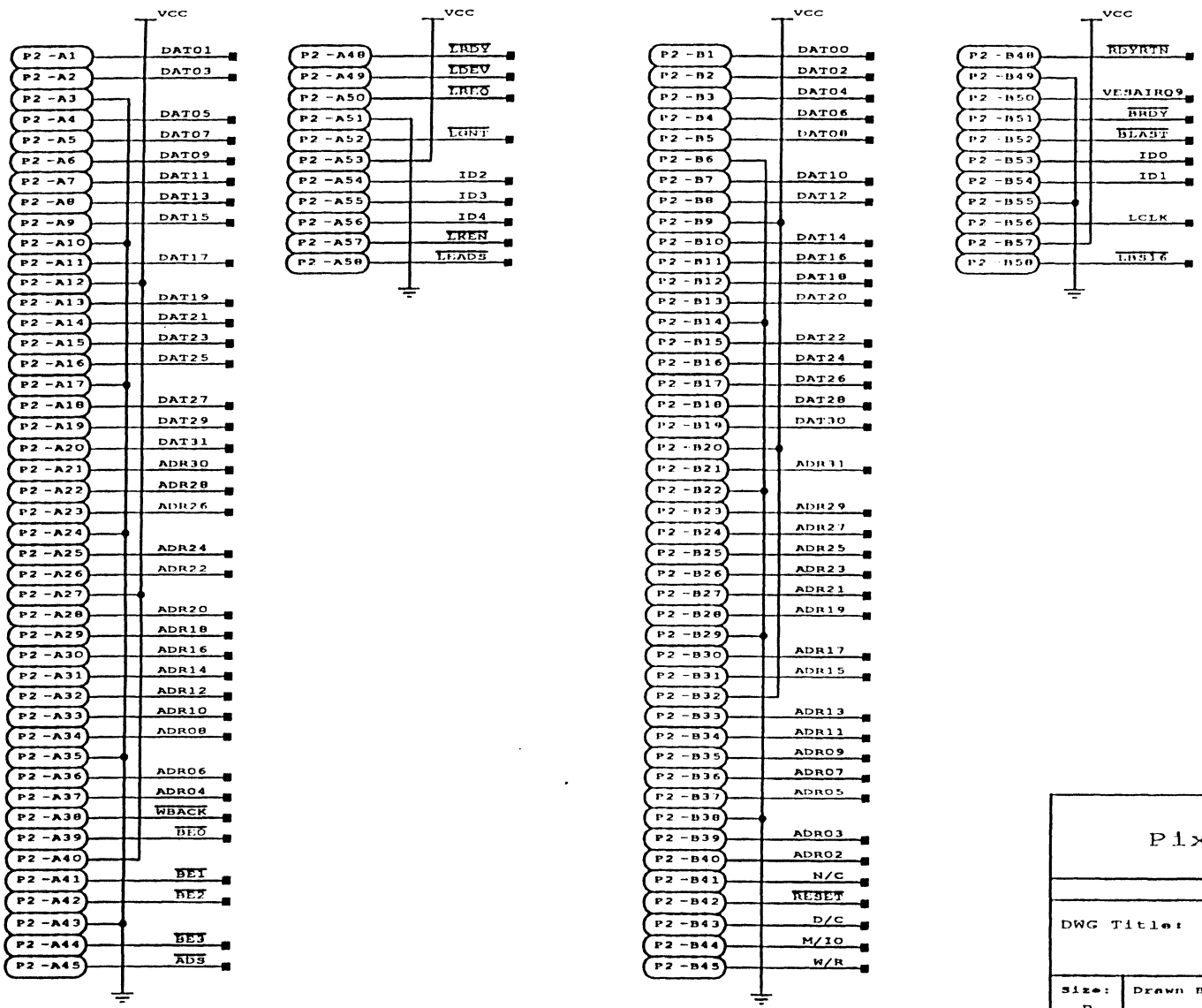
VGA FRAME BUFFER

Pixel Semiconductor		
DWG Title:	2085 EVALUATION BOARD	
Size: B	Drawn By:	Rev: APP
Date: 11/05/93		Page: 6 OF 11


SA[19:0]
SD[15:0]
BD[15:0]
U19
74LS245
U22
74LS245
U21
27C256-120
ISABUSOUT
ROMA0
ROMEN
MEMSEL
MEMCS16
U26
74F38
ACK
IOCHRDY
OWS
ZERO
5286 ISA INTERFACE & BIOS
Pixel Semiconductor
DWG Title: 2085 EVALUATION BOARD
Size: B
Drawn By:
Rev: APP
Date: 11/05/93
Page: 7 OF 11


U13
PIXEL SEMICONDUCTOR
CL-PX2085
MEDIADAC (TM)
P[31:0]
ZC[3:0]
PX[31:0]
VIDOUT[7:0]
SA[15:0]
BD[7:0]
VCC
VREF
R40 1K
LM385 -1.2DZ
D1
C27 .01UF,50V
C31 .1UF,50V
C17 .1UF,50V
C12 .1UF,50V
C16 10UF,35V
C4 .1UF,50V
C20 .1UF,50V
C24 .1UF,50V
R34 33 OHM
R37 1.4K
R36 10K
R99 0 OHM
AVDD1 61
AVDD2 69
AVSS1 60
AVSS2 70
PCLK0 139 VIDOUTCLK
PCLK1 138 PIXCLK
LCLK 136 DACLOAD
SCLK 135 SCLK
PCO 137 DCLK
IREF 65
VRIN 66 VREF
COMP 73
RED 64 RED
GREEN 63 GREEN
BLUE 62 BLUE
GPS 75 P9000VIDEOENABLE
SENSE 68 SENSE
EVEN 76
BORDER 78
CBLANK 77 DACBLANK
BLANKOUT 74 KBLANK
VSIN 82 VSYNC
VSOUT 84 DVSYNC
VROUT 72
HSIN 79 HSYNC
HSOUT 83 DHSYNC
EVIDEO 71 EVIDEO
MEDIADAC (TM)
Pixel Semiconductor
DWG Title: 2085 EVALUATION BOARD
Size: B
Drawn By:
Rev: APP
Date: 11/05/93
Page: 8 OF 11


P1 -A1
P1 -A2
P1 -A3
P1 -A4
P1 -A5
P1 -A6
P1 -A7
P1 -A8
P1 -A9
P1 -A10
P1 -A11
P1 -A12
P1 -A13
P1 -A14
P1 -A15
P1 -A16
P1 -A17
P1 -A18
P1 -A19
P1 -A20
P1 -A21
P1 -A22
P1 -A23
P1 -A24
P1 -A25
P1 -A26
P1 -A27
P1 -A28
P1 -A29
P1 -A30
P1 -A31
SD7
SD6
SD5
SD4
SD3
SD2
SD1
SD0
SD[7:0]
IOCHRDY
AEN
SA16
SA15
SA14
SA13
SA12
SA11
SA10
SA9
SA8
SA7
SA6
SA5
SA4
SA3
SA2
SA1
SA0
SA[17:0]
VCC
P1 -B1
P1 -B2
P1 -B3
P1 -B4
P1 -B5
P1 -B6
P1 -B7
P1 -B8
P1 -B9
P1 -B10
P1 -B11
P1 -B12
P1 -B13
P1 -B14
P1 -B15
P1 -B16
P1 -B17
P1 -B18
P1 -B19
P1 -B20
P1 -B21
P1 -B22
P1 -B23
P1 -B24
P1 -B25
P1 -B26
P1 -B27
P1 -B28
P1 -B29
P1 -B30
P1 -B31
ISARESET
IRQ9
NOWS
12V
IOW
IOR
REFRESH
BALE
P1 -C1
P1 -C2
P1 -C3
P1 -C4
P1 -C5
P1 -C6
P1 -C7
P1 -C8
P1 -C9
P1 -C10
P1 -C11
P1 -C12
P1 -C13
P1 -C14
P1 -C15
P1 -C16
P1 -C17
P1 -C18
SBHE
LA23
LA22
LA21
LA20
LA19
LA18
LA17
LA[23:17]
MEMR
MEMW
SD8
SD9
SD10
SD11
SD12
SD13
SD14
SD15
SD[15:8]
P1 -D1
P1 -D2
P1 -D3
P1 -D4
P1 -D5
P1 -D6
P1 -D7
P1 -D8
P1 -D9
P1 -D10
P1 -D11
P1 -D12
P1 -D13
P1 -D14
P1 -D15
P1 -D16
P1 -D17
P1 -D18
MEMCS16
VCC
D3
D4
D5
RED
GREEN
BLUE
L3
L4
L5
52 OHM
52 OHM
52 OHM
R57
R66
R24
75 OHM
75 OHM
75 OHM
D1
D2
DH3SYNC
DV3SYNC
L1
L2
52 OHM
52 OHM
P3 1
P3 2
P3 3
P3 4
P3 7
P3 8
P3 10
P3 16
P3 9
P3 17
P3 11
P3 12
P3 13
P3 14
P3 15
P3 6
P3 5
VCC
220UF,10V
C50
220UF,10V
C46
220UF,10V
C49
220UF,10V
C1
.1UF,50V
C30
.1UF,50V
C26
.1UF,50V
C32
.1UF,50V
C35
.1UF,50V
C3
.1UF,50V
C7
.1UF,50V
C14
.1UF,50V
C13
.1UF,50V
C25
.1UF,50V
C43
.1UF,50V
C47
.1UF,50V
C10
.001UF,50V
C22
.001UF,50V
C21
.001UF,50V
C9
.001UF,50V
C41
.001UF,50V
C36
.001UF,50V
C33
.1UF,50V
C15
.1UF,50V
C6
.1UF,50V
C19
.1UF,50V
C42
.1UF,50V
C29
.1UF,50V
C38
.1UF,50V
C45
.1UF,50V
C23
.1UF,50V
C40
.1UF,50V
C34
.1UF,50V
C2
.1UF,50V
C39
ISA and Video CONNECTORS
Pixel Semiconductor
DWG Title:
2085 EVALUATION BOARD
Size:
B
Drawn By:
Rev:
APP
Date:
11/05/93
Page:
9 OF 11

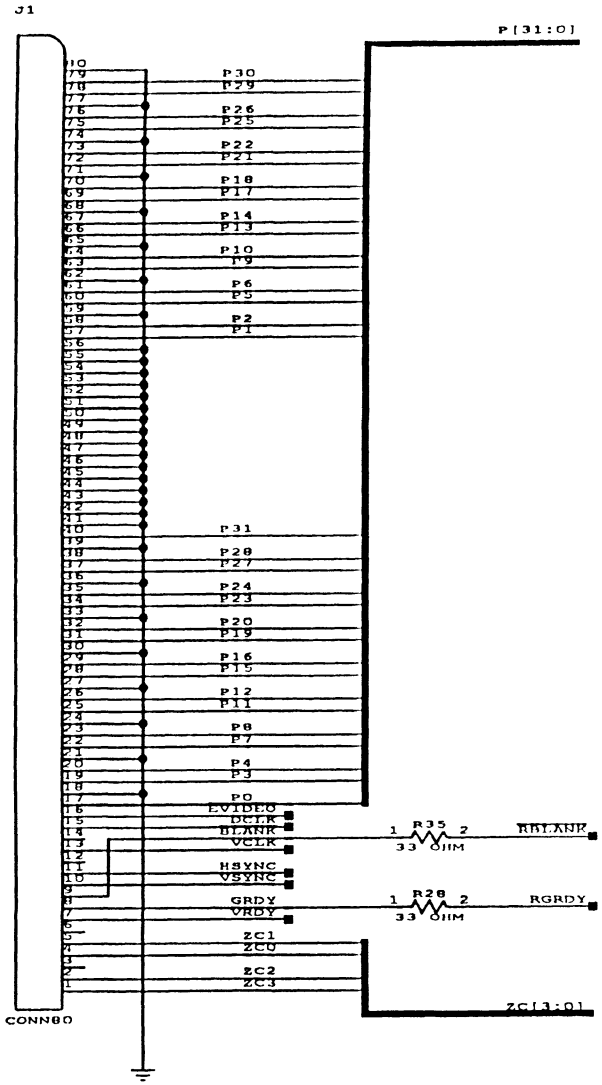


VESA Connectors

Pixel Semiconductor

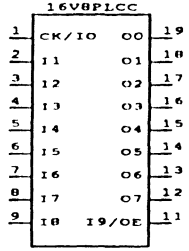
DWG Title: 2085 EVALUATION BOARD

Size: B	Drawn By:	Rev: APP
Date: 11/05/93		Page: 10 OF 11

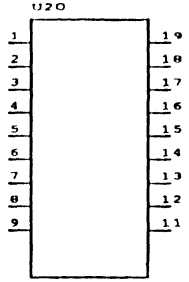


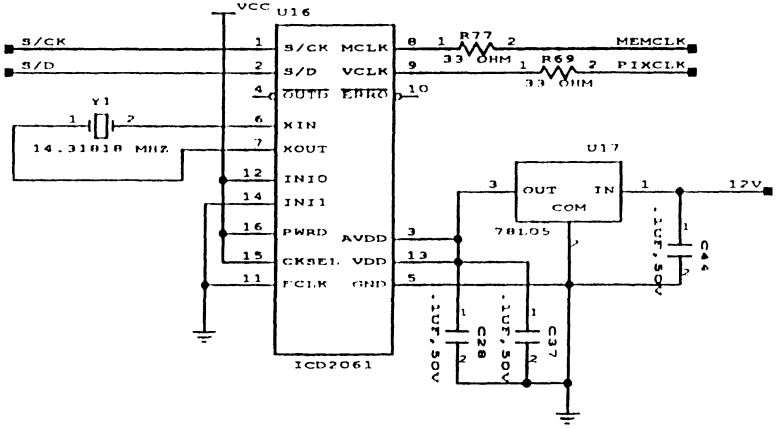
VAFC CONNECTOR

SPARE PAL LOCATION

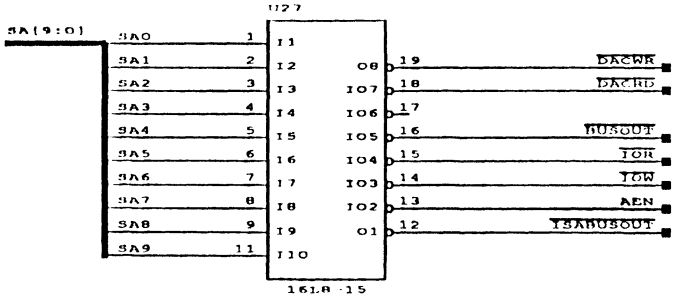


SPARE 20PIN SOIC LOCATION





FREQ SYNTH



ISA RAMDAC Interface

Pixel Semiconductor		
DWG Title: 2085 EVALUATION BOARD		
Size: B	Drawn By:	Rev: APP
Date: 11/05/93		Page: 11 OF 11

10.4.2 Existing Design: Weitek® Evalutaion Board with Bt485

This section describes the palette DAC interface of the existing design which incorporated a Bt485 RAM-DAC. References to schematic pages and device reference designators are to Weitek schematic 4120-0115-00, Rev. 01.

Figure 10-3 is a block diagram showing the major details of the Bt485 interface. Figure 10-3 on page 141 shows a simplified schematic page with the Bt485. Portions of the system that are common to both versions are shown with shaded lines, while those parts of the design unique to the Bt485 version are shown with solid lines.

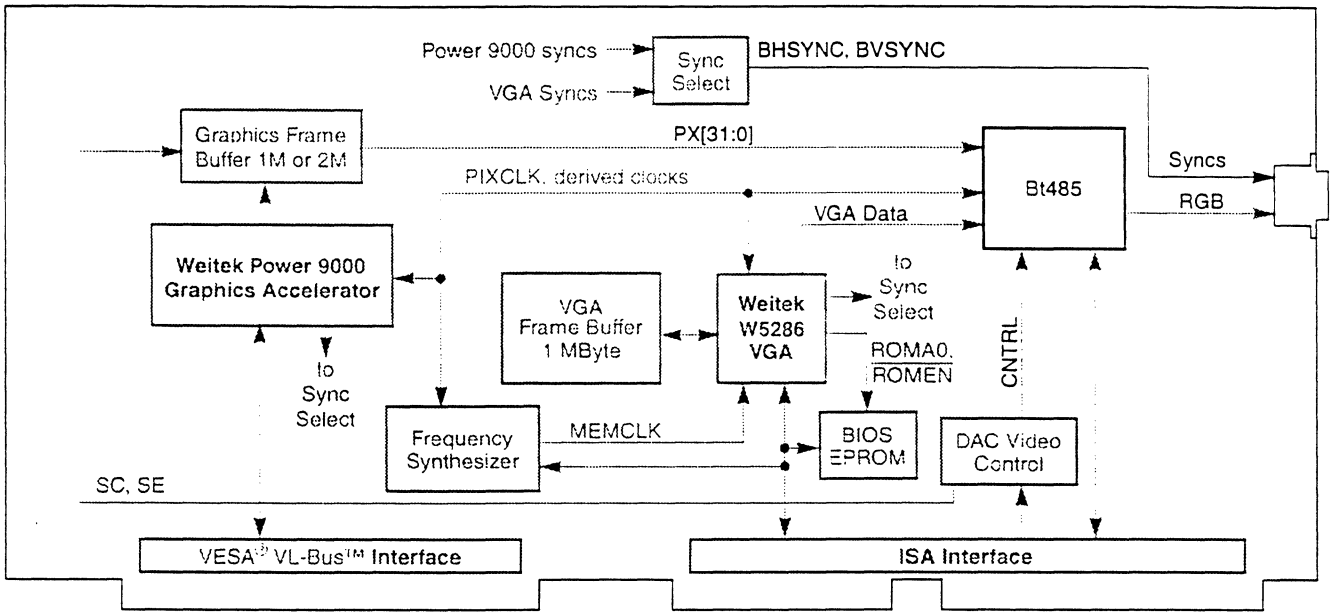


Figure 10-4. Block Diagram, Original Board with Bt485

Graphics 32-bit data from the VRAM serial ports enters the Bt485 through a 32-bit interface subdivided into 8-bit groups corresponding to 8-bit pixels. When 8-bit pixels are used, four pixels are clocked into this port in parallel. The first pixel displayed is pixel A, clocked in on signals PA[7:0], followed by pixels B, C, and D. VGA data from the W5286 VGA controller enters the Bt485 on signals V[7:0].

The DAC/Video Control block decodes the addresses and generates RS[3:0], $\overline{\text{DACWR}}$, and $\overline{\text{DACRD}}$ for Bt485 register access.

The monitor interface has the analog outputs IOR, IOG, and IOB from the Bt485. These outputs each drive a 75-Ω resistor to ground. When a monitor is connected, each output drives 37.5 — 75-Ω from the resistor in parallel with 75-Ω from the impedance of the monitor input. Syncs BHSYNC and BVSYNC are directly output from the sync-select logic. All monitor outputs are fed through EMI filters.

The Bt485 reference voltage is set by an external 147-Ω resistor. The COMP Pin is connected through a series 0.1-µF capacitor to VAA. VREFIN is directly connected to VREFOUT, and both pins are decoupled to ground through a 0.1-µF capacitor.

VAA is derived by filtering the digital VCC +5-V supply through a ferrite-bead inductor.

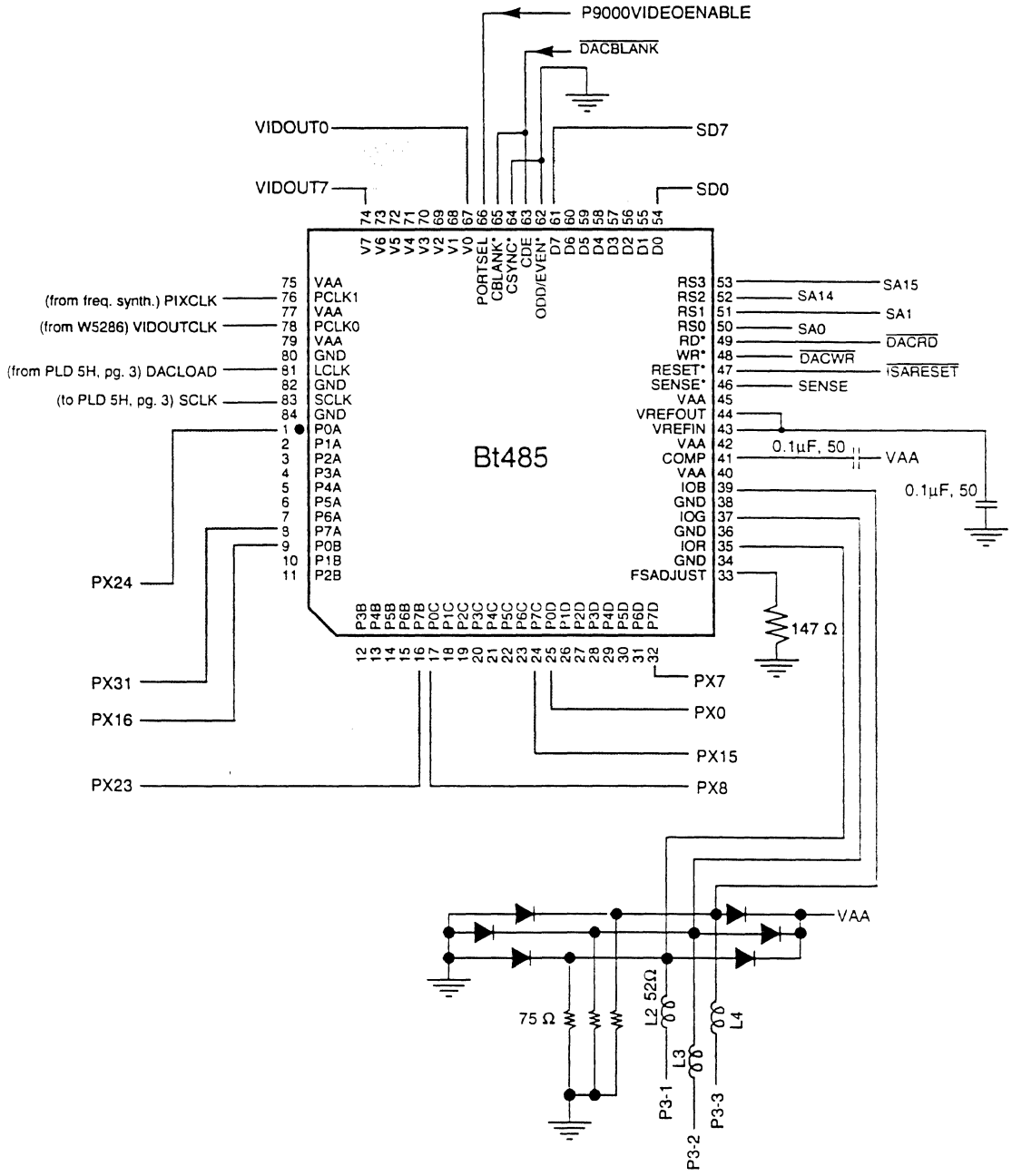


Figure 10-5. Signal Interface to Bt485

10.4.3 Upgraded Design: Weitek® Evaluation Board with CL-PX2085

The areas that were redesigned are indicated by solid black lines in Figure 10-6. Much of the circuit remains
unchanged, as indicated by the shaded lines. The two main areas of change are the source of the monitor sync signals and the addition of the VAFC port.

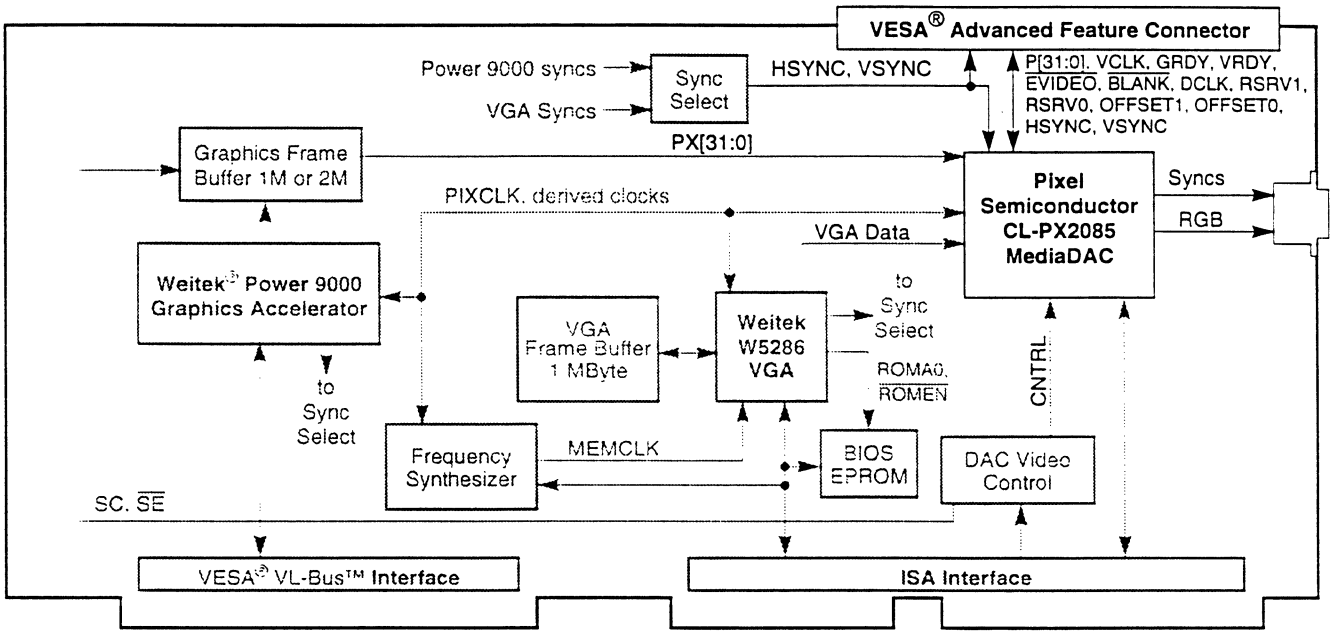


Figure 10-6. Block Diagram, Redesigned Board with CL-PX2085

10.4.3.1 Monitor Sync Source

The original monitor sync signals BHSYNC and BVSYNC are renamed to HSYNC and VSYNC, and pass-through the sync-alignment unit of the CL-PX2085. The syncs emerge from the CL-PX2085 as the signals DHSYNC and DVSYNC, that drive the monitor sync inputs.

10.4.3.2 VAFC Port

The VAFC signals exist almost exclusively between the connector and the CL-PX2085. Exceptions are the VAFC horizontal and vertical sync signals, which are driven by the HSYNC and VSYNC from the sync select logic. The signals include: P[31:0], VCLK, GRDY, VRDY, $\overline{\text{EVIDEO}}$, $\overline{\text{BLANK}}$, DCLK, RSRV1, RSRV0, OFFSET1, OFFSET0, HSYNC, and VSYNC.

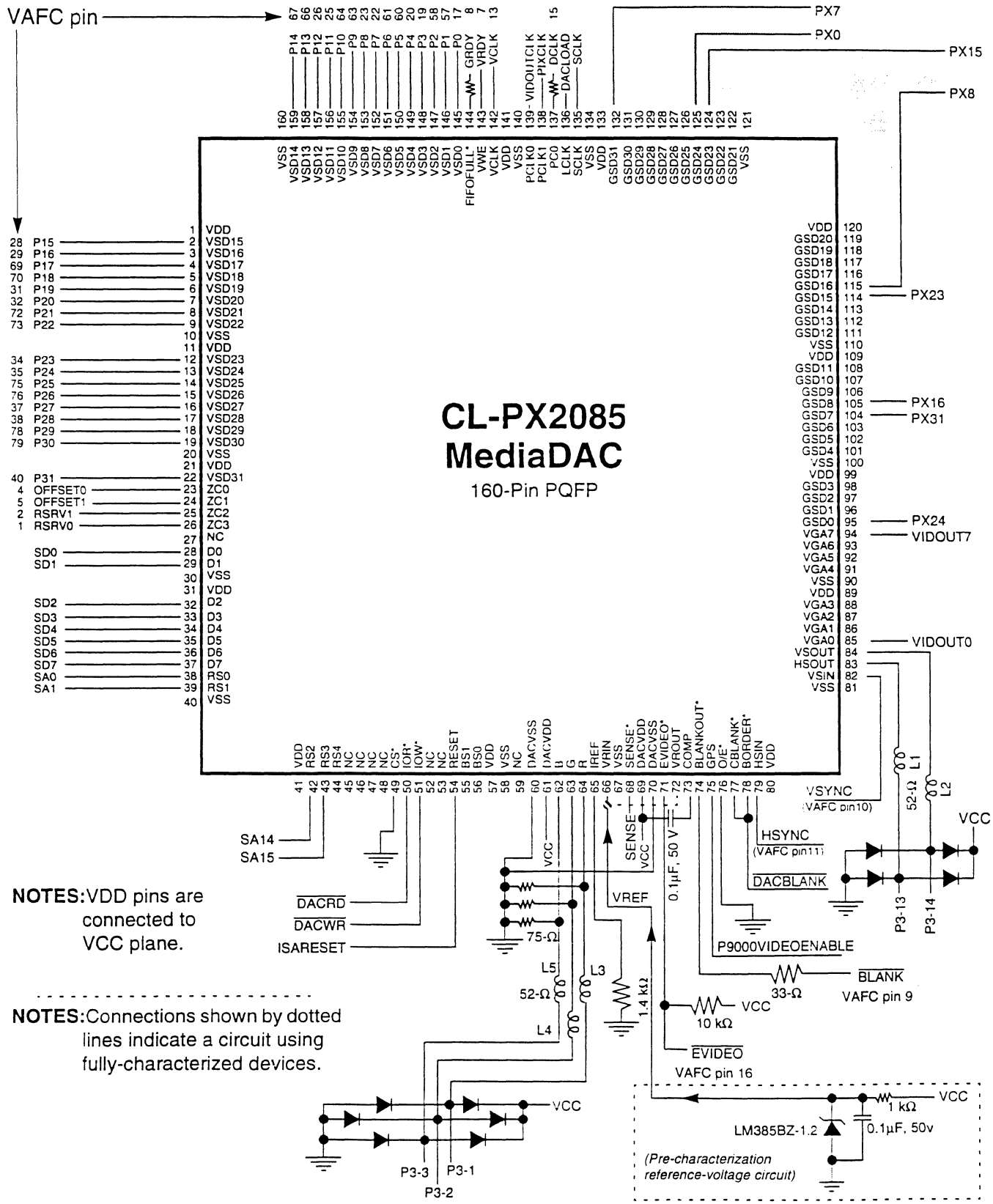


Figure 10-7. Signal Interface to CL-PX2085

Table 10-4. Parts List for Weitek Evaluation Board with CL-PX2085

Item	Qty/ Bd	Reference Designator	Part Description	Part#	Package
1	1	U8	IC, Power 9000, 33 MHz, Weitek®		208 PQFP
2	1	U11	IC, 5286-080-PFP, Weitek®		160 PQFP
3	2	U3, U17	IC, voltage regulator, 12 V/5 V	LM78L05ACZ	TO-92
4	8	U1,U2,U4,U5,U6,U7,U9,U10	IC, VRAM, 256K x 8, 70 NS, SOJ	MT42C8255DJ-7	SOJ 40
5	2	U14, U15	IC, DRAM, 256K x 16, 70 NS, SOJ	MT4C16256DJ-7	SOJ 40
6	2	U19, U22	IC, octal bus transceiver, SMD	SN74LS245DW	20 SOIC
7	1	U26	IC, Quad 2-In NAND, OC, SMD	M74F38D	14 SOIC
8	1	U21	IC, 32K x 8 PROM, 120 NS, 32 PLCC	AM27C256-120	32 PLCC
9	1	U18	IC, PAL, 22V10-10, PLCC	PAL22V10-10JC	28 PLCC
10	2	U24, U27	IC, PAL, 16L8-15, PLCC	PAL16L8JC	20 PLCC
11	1	U16	IC, Dual Pgm Graphics Clock Gen	ICD2061ASC-1	16 SOIC
12	1	Z1	IC, 1.2 V voltage reference	LM385BZ-1.2	TO-92
13	1	U13	IC, CL-PX2085, Pixel Semiconductor		160-pin PQFP
14	1	U12	IC, PAL, 16R6-7, PLCC	PAL16R6-7JC	20-pin PLCC
15	1	U23	IC, PAL, 16R4-15, PLCC	PAL16R4-15JC	20-pin PLCC
16	1	Y1	Crystal, 14.31818 MHz		HC-49/US
17	5	L1, L2, L3, L4, L5	Inductor, ferrite bead, 52-Ω		SMD 1210
18	1	C8	Capacitor, ceramic, 100 pF, 50 V		SMD 1206
19	6	C9, C19, C21,C33, C36, C41	Capacitor, ceramic, 0.001 µF, 50 V		SMD 1206
20	2	C27, C31	Capacitor, ceramic, 0.01 µF, 50 V		SMD 1206
21	34	C2 - C7, C10, C12 - C15, C17, C18, C20, C22 - C26, C28 - C30,C32. C34, C35, C37 - C40, C42 - C45, C47	Capacitor, ceramic, 0.1 µF, 50 V		SMD 1206
22	4	C1, C46, C49, C50	Capacitor, Tantalum, 22 µF, 10V		SMD 7343
23	1	C16	Capacitor, Tantalum, 10 µF, 10V		SMD 7343
24	1	R36	Resistor, 10 kΩ, 1/10 W, 5%		SMD 1206
25	3	R81, R82, R85	Resistor, 4.7 kΩ, 1/10 W, 5%		SMD 1206
26	4	R31, R78 - R80	Resistor, 1.8 kΩ, 1/10 W, 5%		SMD 1206
27	1	R37	Resistor, 1.4 kΩ, 1/10 W, 5%		SMD 1206
28	1	R48	Resistor, 1.0 kΩ, 1/10 W, 5%		SMD 1206
29	3	R57, R66, R74	Resistor, 75 Ohm, 1/10 W, 5%		SMD 1206
30	70	R1 - R30, R32 - R35, R38 - R47, R49 - R56, R58 - R65, R67 - R73,R77, R83, R84	Resistor, 33 Ohm, 1/10 W, 5%		SMD 1206
31					
32	5	D1, D2, D3, D4, D5	Diode, Dual	MMBD7000L	SOT-23
33	1	P3	Connector, D-Sub, 15 pin, HD		HD DSub 15
34	1	J1	Conn, plug, 80 pin, 0.050 x .10, Rt. Ang	5-175472-9	80 Pin Plug
35	1	XU21	Socket, 32-pin PLCC, Surface-Mount		32 PLCC
36	1	XP3	FAB, Mtg. Bracket, AT, w/DSUB9		G97.
37	1	SW1	Switch, DIP, 7 position, SPST	CTS 207-7LPST	14 DIP
38	2	XP3	Jack screw, 4-40 x 5/16		Screw
39	1	PCB	PCB, CL-PX2085 135-MHz Evaluation		PCB

10.4.4 Summary of Changes

This section summarizes the steps taken to upgrade the Weitek schematic (4120-0115-00 Rev. 01) design by adding a CL-PX2085 and a VAFC.

There are no changes to PALs or software drivers. A page was added for the VAFC connector, and the frequency synthesizer and RAMDAC PAL were moved to this page to make room for the CL-PX2085 on the RAMDAC page. With reference to the Weitek sample layout to make room for the VAFC connector the two DRAMs for the 5286 were moved to a position below the eight VRAMs for the P9000. Other components were rearranged to enhance routeability.

Changes to the original design were as follows:

1. Replace Bt485 with CL-PX2085, and remove VAA inductor.
2. Tie ISARESET directly to the CL-PX2085 RESET pin.
3. Ground the CS* signal on the CL-PX2085.
4. Leave BS[1:0] floating, since they have internal pull-up resistors in the CL-PX2085.
5. Add the VAFC connector, with a 10 kΩ pullup resistor on EVIDEO* and 33-Ω series-termination resistors on the CL-PX2085 outputs that drive the VAFC connector (BLANK*, GRDY, DCLK).
6. Change the resistor value for IREF (FSADJUST).
7. Connect the clamping diodes for the final RED, GREEN and BLUE pins of the connector on the connector side of the EMI filter inductors.
8. Drive the horizontal and vertical sync signals for the monitor from the CL-PX2085 instead of the HSYNC and VSYNC outputs of the Misc. PAL (BHSync renamed HSync, and BVSync renamed VSync).
9. Add two more dual clamping diodes (one each for the horizontal and vertical sync signals on the connector side of the EMI filter inductors) to protect the CL-PX2085 from electrostatic discharge.
10. Add one spare 20-pin PLCC location and one spare 20-pin SOIC location.
11. Add an LM385BZ-1.2 precision voltage reference IC and supporting passive components to provide the option to drive the external voltage reference input (VRIN on the CL-PX2085) with an external reference voltage.

10.4.5 CL-PX2085 PC-Board Layout Considerations

The CL-PX2085 provides a separate set of DACVDD and DACVSS pins to power the analog output section. For the CL-PX2085 design, DACVDD is powered by VCC, and DACVSS connects to the ground area. These connection points are in a specially prepared analog region of the VCC and ground areas.

WARNING: Using a ferrite bead inductor to supply DACVDD from VCC and/or DACVSS from ground may damage the device, and is strongly discouraged.

This example design uses a current-steering method to separate the digital and analog current paths by providing partially isolated power and ground areas for the analog signals and components. Areas that overlay or overlap digital areas are not recommended. See Figure 10-8.

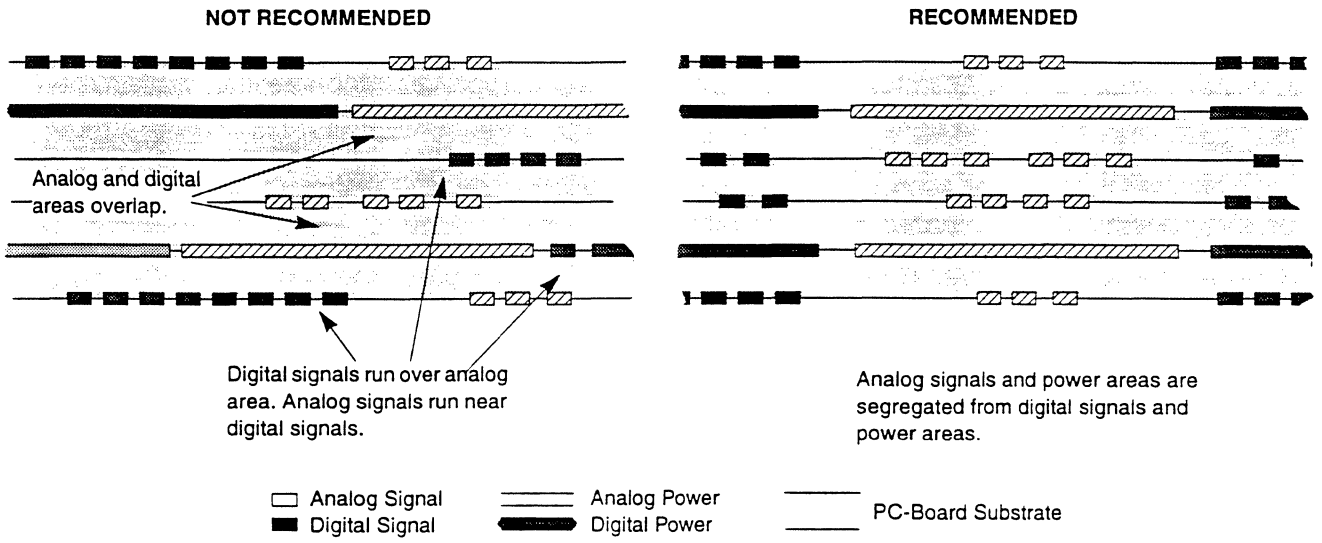


Figure 10-8. Cross Section of Example PC-Board Layout

The preferred method for establishing the analog power and ground areas is illustrated in Figure 10-9. A slot surrounds the analog circuitry creating isolated areas within the digital power areas. The ends of the slot are near the monitor-connector mounting holes, but sufficient space is left to provide a substantial ground connection from the board to the chassis and the monitor cable through the mounting holes, as shown in Figure 10-9.

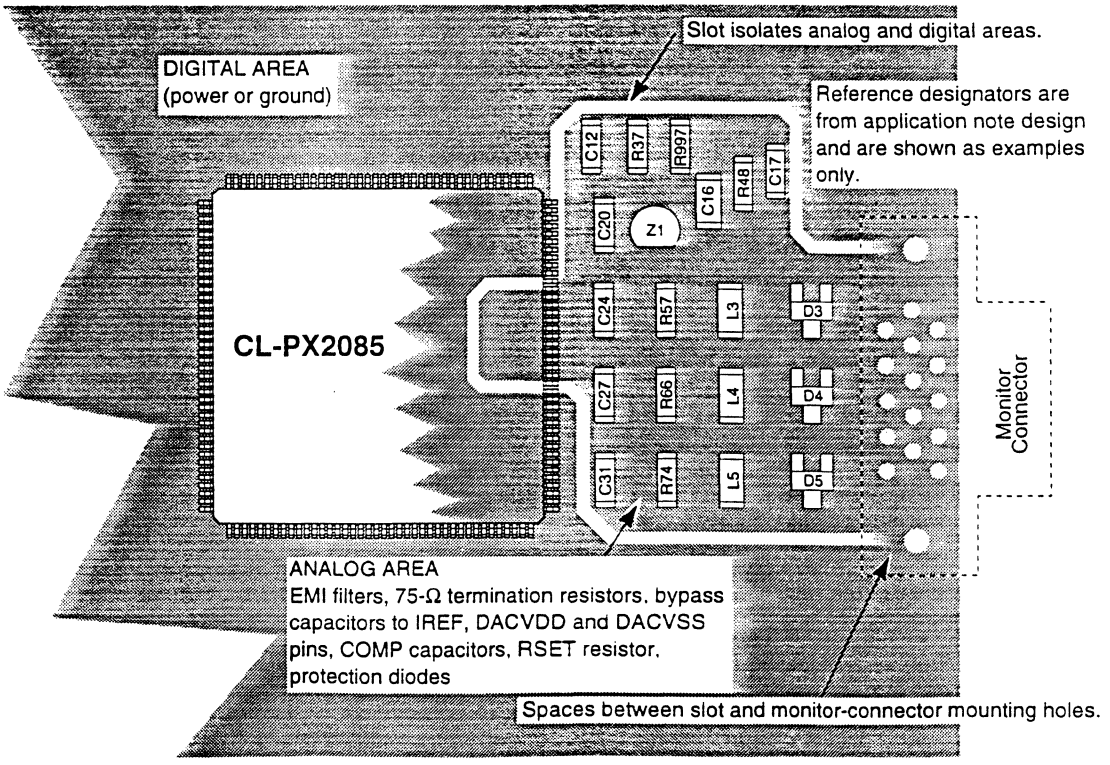


Figure 10-9. Analog and Digital Power Areas

To minimize digital noise on the analog outputs, all analog components, (bypass capacitors, 75-Ω termination resistors, EMI filters, COMP capacitor, etc.) should be placed within the perimeter of the analog area. No digital signals should be run near or through this area. The analog return paths for the analog RGB outputs should be connected within the analog area.

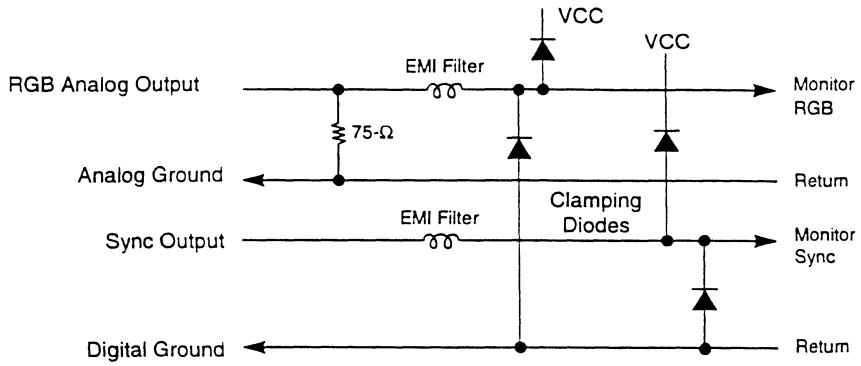


Figure 10-10. RGB Output Circuit

The digital return for the sync output signals should be connected within the digital ground area. In addition, the reverse-biased clamping diodes should clamp the analog outputs to the digital power and ground areas (Figure 10-10), to help protect the analog output section from electrostatic discharge.

10.5 Programmable Logic Device Equations

No changes were required for the upgrade to CL-PX2085 functionality and VAFC compatibility. Changes detailed within these listings was made to the original Weitek design before the CL-PX2085 upgrade.

10.5.1 MISC - P16L8 at U24

```
module MISC;
" Options '-reduce bypin fixed';
flag '-r2';
title 'BE translation and sync muxing.';
misc device 'P16L8';

" This PAL translates the BE signals into HA/HSIZE for the P9000,
" muxes the appropriate sync signals to the monitor.

" History
" =======
" 2/3/93
" Created file for the combo board.

"power
 VCC              pin 20;
 GND              pin 10;

" inputs

   p9000syncpol  pin 1;
   p9000enable   pin 2; " Enables P9000 video

   p9000hsync_   pin 3; p9000hsync macro {!p9000hsync_};
   p9000vsync_   pin 4; p9000vsync macro {!p9000vsync_};
   p9000syncpol  pin 1;

   vgahsync      pin 5;
   vgavsync      pin 6;

 be3_            pin 7;    " BE's from VL-Bus
```

```
 be2_            pin 8;
 be1_            pin 11;
 be0_            pin 9;

" outputs
   hsyncout      pin 13;
   vsyncout      pin 12;

 ha1 pin 17;
 ha0 pin 16;
 hsize1 pin 18;
 hsize0 pin 19;

" CONSTANTS

 T,F,X,C,Z = 1,0,.X.,.C.,.Z.;

equations

    " Multiplex the VGA and P9000 syncs to the video connector.
    " Reverse the polarity of the P9000 syncs if requested.
    " (Some video modes require positive sync polarity.
    " Since the P9000 believes that negative syncs are the
    " only way to go, we reverse the polarity here if
    " necessary. This will probably never get used since
    " only the lower res modes such as 800x600 need
    " positive syncs.)
    " We don't have the ability to mix sync polarities.

    hsyncout = p9000enable & (p9000hsync_ $ p9000syncpol) #
                  !p9000enable & vgahsync;
    vsyncout = p9000enable & (p9000vsync_ $ p9000syncpol) #
                  !p9000enable & vgavsync;

" Generate the HA[1..0] and HSIZE[1..0] signals from the BE's.
" BE -> HA/HSIZE -> WE's for DFB writes
"
truth_table ([be3_, be2_, be1_, be0_] -> [ha1, ha0, hsize1, hsize0])
 [0,0,0,0] -> [0,0,0,0]; " long
 [0,0,0,1] -> [0,0,1,1]; " triplet
 [0,0,1,0] -> [X,X,X,X]; " illegal
 [0,0,1,1] -> [0,0,1,0]; " half
 [0,1,0,0] -> [X,X,X,X]; " illegal
 [0,1,0,1] -> [X,X,X,X]; " illegal
 [0,1,1,0] -> [X,X,X,X]; " illegal
 [0,1,1,1] -> [0,0,0,1]; " byte
 [1,0,0,0] -> [0,1,1,1]; " triplet
 [1,0,0,1] -> [0,1,1,0]; " half
 [1,0,1,0] -> [X,X,X,X]; " illegal
 [1,0,1,1] -> [0,1,0,1]; " byte
 [1,1,0,0] -> [1,0,1,0]; " half
 [1,1,0,1] -> [1,0,0,1]; " byte
 [1,1,1,0] -> [1,1,0,1]; " byte
 [1,1,1,1] -> [X,X,X,X]; " illegal

test_vectors
([p9000enable,p9000syncpol,p9000hsync_,p9000vsync_,vgahsync,vgavsync]->[hsyncout,vsyncout])
 [ 1 , 0 , 0 , 0 , X , X ]->[ 0 , 0 ];
```

```
 [ 1 , 0 , 0 , 1 , X , X ]->[ 0 , 1 ];
 [ 1 , 0 , 1 , 0 , X , X ]->[ 1 , 0 ];
 [ 1 , 0 , 1 , 1 , X , X ]->[ 1 , 1 ];
 [ 1 , 1 , 0 , 0 , X , X ]->[ 1 , 1 ];
 [ 1 , 1 , 0 , 1 , X , X ]->[ 1 , 0 ];
 [ 1 , 1 , 1 , 0 , X , X ]->[ 0 , 1 ];
 [ 1 , 1 , 1 , 1 , X , X ]->[ 0 , 0 ];
 [ 0 , X , X , X , 0 , 0 ]->[ 0 , 0 ];
 [ 0 , X , X , X , 0 , 1 ]->[ 0 , 1 ];
 [ 0 , X , X , X , 1 , 0 ]->[ 1 , 0 ];
 [ 0 , X , X , X , 1 , 1 ]->[ 1 , 1 ];

test_vectors
([be3_, be2_, be1_, be0_] -> [ha1, ha0, hsize1, hsize0])
 [0,0,0,0] -> [0,0,0,0]; " long
 [0,0,0,1] -> [0,0,1,1]; " triplet
 [0,0,1,0] -> [X,X,X,X]; " illegal
 [0,0,1,1] -> [0,0,1,0]; " half
 [0,1,0,0] -> [X,X,X,X]; " illegal
 [0,1,0,1] -> [X,X,X,X]; " illegal
 [0,1,1,0] -> [X,X,X,X]; " illegal
 [0,1,1,1] -> [0,0,0,1]; " byte
 [1,0,0,0] -> [0,1,1,1]; " triplet
 [1,0,0,1] -> [0,1,1,0]; " half
 [1,0,1,0] -> [X,X,X,X]; " illegal
 [1,0,1,1] -> [0,1,0,1]; " byte
 [1,1,0,0] -> [1,0,1,0]; " half
 [1,1,0,1] -> [1,0,0,1]; " byte
 [1,1,1,0] -> [1,1,0,1]; " byte
 [1,1,1,1] -> [X,X,X,X]; " illegal

end MISC;
```

10.5.1.1 LOCALSM, P22V10C at U18

```
module LOCALSM;
Options '-reduce bypin fixed';
title 'LocalBus Interface.';
localsm device 'P22V10C';

" This PAL contains the logic to interface the WEITEK P9000 Graphics
" controller chip to the VESA local bus. A -10 or faster PAL should
" be used because of the low (7 ns) set-up time given by the VESA local bus.

" History (most recent changes are listed first)
" =======
"
" *** 7/6/93
" Changed the address mapping to be compatible with more
" chipsets. (Some of the chipset manufacturers are not handling
" at all of the address lines on VL.)
"
" *** 5/19/93 ***
" Changed the method that is used to generate the output enable
" for lrdy. It is now driven high for half-of a cycle before
" it is tristated during active termination. Previously it was
" driven high for a shorter duration equal to the feedback time
```

```
" through the array. Some PALs feedback and tristate so
" quickly that lrdy was not getting properly terminated.
"
" NOTE: Fast writes are not supported by this PAL any more
" because we need the latched id2 pin to help generate
"  the lrdy output enable. This will have a small but
"  measurable performance hit. (If we hadn't added the
"  hread output on 2/3 then we could have fit this in...)
"
" Also deleted comments that pertained to ABEL removing a
" hazard term that was needed to build the id2 latch.
"
" *** 2/3/93 ***
" Added the hread output for the combo board.
"
" *** 10/2/92 ***
" Changed pin numbers to support a surface mounted PAL.
"

"power
 VCC             pin 28;
 GND             pin 14;

" inputs

   lclk                  pin 2;

 adr31           pin 3;
 adr30           pin 4;
 adr29           pin 5;
 adr26           pin 6;
 adr = [adr31,adr30,adr29,adr26];
                                                  " STUFF FOR TEST VECTORS
   XADR_c = [.X.,.X.,.X.,.X.];" Don't care address
   PADR_c = [1,1,1,0];           " P9000 address
   OADR_c = [0,0,0,0];           " Other address

   dipsw1                pin 7;
   dipsw0                pin 9;
   dipsw = [dipsw1,dipsw0];
   DIPSW_c = [1,1];


   " Figure out the P9000 address from the dipswitch
   " Dipsw1 Dipsw0 Address
   " 1 1 $E0000000
   " 1 0 $C0000000
   " 0 1 $80000000
   " 0 0 $04000000
   p9000addr = [dipsw1#dipsw0,dipsw1,dipsw1&dipsw0,!dipsw1&!dipsw0];

   reset_                pin 10; reset macro {!reset_};

   rdy_rtn_      pin 11; rdyrtn macro {!rdy_rtn_};
   m_io                  pin 12;
   ads_                  pin 13; ads macro {!ads_};
   id2                   pin 16;
   w_r                   pin 17;
```

```
   hrdy_                    pin 27; hrdy macro {!hrdy_};

" outputs
 hreq_            pin 23 = 'neg,reg'; hreq macro {!hreq_};
   hdrive_                  pin 24 = 'neg,reg'; hdrive macro {!hdrive_};
   ldev_                    pin 19 = 'pos,com'; ldev macro {!ldev_};
   lrdy_                    pin 21 = 'neg,reg'; lrdy macro {!lrdy_};
   xstate                   pin 25 = 'pos,reg';
   lrdyoe                   pin 20 = 'pos,reg';
   gateoe                   pin 26 = 'pos,com';

   hread                    pin 18 = 'pos,com';
" Logical value/clock definitions
 T,F,X,C,Z = 1,0,.X.,.C.,.Z.;


" State register
   sreg = [hreq,lrdy,lrdyoe,hdrive,xstate];

" State definitions
"  NOTE: Logical values (not physical) values used in state assignment.
"
"                hllhs
"                rrrdt
"                eddra
"                qyyit
"                ove
"                ee        Hex   Signals Active
"                =====     ===   ==============
   idle         = ^b00000;" 00   <none>
   rreq         = ^b10000;" 10   hreq, hdrive
   rrqwait      = ^b10110;" 16   hreq, hdrive, lrdy driven false
   rwait        = ^b00110;" 06   hdrive, lrdy driven false
   rdone        = ^b01110;" 0e   hdrive, lrdy driven true
   rdata        = ^b00111;" 07   hdrive - May need to drive lrdy false based on lclk high
   term         = ^b00101;" 05   <none> - May need to drive lrdy false based on lclk high
   wdone        = ^b11100;" 1c   hreq, lrdy driven true
   wdelay       = ^b01000;" 08   <none>
   wwait        = ^b10100;" 14   hreq, lrdy driven false


   " Define states for test vectors
   " Slightly different from the above state definitions
   idle_c                   = [1,Z,1,0];" {hreq_, lrdy_, hdrive_, xstate}
   rreq_c                   = [0,Z,1,0];
   rrqwait_c     = [0,1,0,0];
   rwait_c                  = [1,1,0,0];
   rdone_c                  = [1,0,0,0];
   rdata_a                  = [1,1,0,1];" lrdy driven high (first half of cycle)
   rdata_c                  = [1,Z,0,1];" lrdy tristate (second half of cycle)
   term_a                   = [1,1,1,1];" lrdy driven high (first half of cycle)
   term_c                   = [1,Z,1,1];" lrdy tristate (second half of cycle)
   wdone_c                  = [0,0,1,0];
   wdelay_c      = [1,Z,1,0];
   wwait_c                  = [0,1,1,0];
```

```
" Convenient macros

P9000WRITEmacro {(ads & (adr == p9000addr) & m_io & w_r)}
P9000READmacro {(ads & (adr == p9000addr) & m_io & !w_r)}

equations

@dcset

   hread = !w_r;                          " Invert w/r polarity for P9000

   ldev = m_io & (adr == p9000addr);" LDEV is combinatorially decoded


   gateoe = !(xstate & !lclk);" gateoe goes low when we are in
                                          " a state with xstate asserted and
                                          " lclk is low. This is used to disable
                                          " the lrdy output for active termination.

   hreq_.AR      = reset;         " The reset signal forces the
   hdrive_.AR    = reset;         " state machine into the idle
   lrdyoe.AR     = reset;         " state.
   lrdy_.AR      = reset;
   xstate.AR     = reset;

   hreq_.OE = 1;
   hdrive_.OE = 1;
   lrdy_.OE = lrdyoe & gateoe;
   xstate.OE = 1;

   hreq_.clk = lclk;
   hdrive_.clk = lclk;
   lrdy_.clk = lclk;
   xstate.clk = lclk;
   lrdyce.clk = lclk;

state_diagram sreg;

   State idle:
                IF (P9000WRITE) THEN wdelay;
                ELSE IF (P9000READ) THEN rreq;
                ELSE idle;

   State wdone:
                GOTO term;

   State wdelay:
                IF (hrdy) THEN wdone;
                ELSE wwait;

   State wwait:
                IF (hrdy) THEN wdone;
                ELSE wwait;

   State rreq:
                IF (hrdy) THEN rwait;
                ELSE rrqwait;
```

```
   State rrqwait:
                IF (hrdy) THEN rwait;
                ELSE rrqwait;

   State rwait:
                IF (hrdy) THEN rdone;
                ELSE rwait;

   State rdone:
                IF (rdyrtn) THEN term;
                ELSE rdata;

   State term:
                IF (P9000WRITE) THEN wdelay;
                ELSE IF (P9000READ) THEN rreq;
                ELSE idle;

   State rdata:
                IF (P9000WRITE) THEN wdelay;
                ELSE IF (P9000READ) THEN rreq;
                ELSE idle;

test_vectors                                                    " TEST STATE MACHINE
([lclk,reset_,adr ,dipsw, id2,ads_,m_io,w_r,hrdy_,rdy_rtn_] -> [hreq_,lrdy_,hdrive_,xstate])
 [C , 0 ,XADR_c,DIPSW_c, 0, X , X , X , X , X ] -> idle_c;" 001:test reset (Note:high speed
mode id2 off)
 [C , 1 ,XADR_c,DIPSW_c, 0, 1 , X , X , X , X ] -> idle_c;" 002:
 [C , 1 ,OADR_c,DIPSW_c, X, 0 , 1 , X , X , X ] -> idle_c;" 003:Non-P9000 access (addr mis-
match)
 [C , 1 ,PADR_c,DIPSW_c, X, 0 , 0 , X , X , X ] -> idle_c;" 004:Non-P9000 access (i/o)
 [C , 1 ,PADR_c,DIPSW_c, X, 0 , 1 , 1 , 0 , X ] -> wdelay_c;" 005:Non high-speed write
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , 0 , X ] -> wdone_c;" 006:
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , X , X ] -> term_c;" 007:
 [C , 1 ,PADR_c,DIPSW_c, X, 0 , 1 , 1 , 0 , X ] -> wdelay_c;" 008:Another Non high-speed write
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , 0 , X ] -> wdone_c;" 009:
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , X , X ] -> term_c;" 010:
 [C , 1 ,PADR_c,DIPSW_c, X, 0 , 1 , 1 , 1 , X ] -> wdelay_c;" 011:Write, P9000 not ready
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , 0 , X ] -> wdone_c;" 012:P9000 goes ready
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , X , X ] -> term_c;" 013:
 [C , 1 ,PADR_c,DIPSW_c, X, 0 , 1 , 1 , 1 , X ] -> wdelay_c;" 014:Write, P9000 not ready
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , 1 , X ] -> wwait_c;" 015:Still not ready
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , 1 , X ] -> wwait_c;" 016:Still not ready
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , 0 , X ] -> wdone_c;" 017:Finally ready
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , X , X ] -> term_c;" 018:
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , X , X ] -> idle_c;" 019:Back to idle
 [C , 1 ,PADR_c,DIPSW_c, X, 0 , 1 , 0 , X , X ] -> rreq_c;" 020:Read
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , 0 , X ] -> rwait_c;" 021:
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , 1 , X ] -> rwait_c;" 022:
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , 0 , X ] -> rdone_c;" 023:
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , 0 , 0 ] -> term_c;" 024:Same cycle rdyrtn
 [C , 1 ,PADR_c,DIPSW_c, X, 0 , 1 , 0 , X , X ] -> rreq_c;" 025:Read
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , 1 , X ] -> rrqwait_c;" 026:
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , 1 , X ] -> rrqwait_c;" 027:
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , 0 , X ] -> rwait_c;" 028:
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , 0 , X ] -> rdone_c;" 029:
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , X , 1 ] -> rdata_c;" 030:Delayed rdyrtn
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , X , X ] -> idle_c;" 031:
```

```
test_vectors                                                    " TEST NEW ACTIVE TERMINA-
TION
([lclk,reset_,adr ,dipsw, id2,ads_,m_io,w_r,hrdy_,rdy_rtn_] -> [hreq_,lrdy_,hdrive_,xstate])
 [C , 0 ,XADR_c,DIPSW_c, 0, X , X , X , X , X ] -> idle_c;" 032:reset then test active termi-
nation
 [C , 1 ,XADR_c,DIPSW_c, 0, 1 , X , X , X , X ] -> idle_c;" 033:
 [C , 1 ,OADR_c,DIPSW_c, X, 0 , 1 , X , X , X ] -> idle_c;" 034:Non-P9000 access (addr mis-
match)
 [C , 1 ,PADR_c,DIPSW_c, X, 0 , 0 , X , X , X ] -> idle_c;" 035:Non-P9000 access (i/o)
 [C , 1 ,PADR_c,DIPSW_c, X, 0 , 1 , 1 , 0 , X ] -> wdelay_c;" 036:Non high-speed write
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , 0 , X ] -> wdone_c;" 037:
 [1 , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , X , X ] -> term_a;" 038: lrdy driven high (phase 0)
 [0 , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , X , X ] -> term_c;" 039: lrdy tristated (phase 1)
 [C , 1 ,PADR_c,DIPSW_c, X, 0 , 1 , 1 , 0 , X ] -> wdelay_c;" 040:Another Non high-speed write
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , 0 , X ] -> wdone_c;" 041:
 [1 , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , X , X ] -> term_a;" 042: (active term)
 [0 , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , X , X ] -> term_c;" 043:
 [C , 1 ,PADR_c,DIPSW_c, X, 0 , 1 , 1 , 1 , X ] -> wdelay_c;" 044:Write, P9000 not ready
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , 0 , X ] -> wdone_c;" 045:P9000 goes ready
 [1 , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , X , X ] -> term_a;" 046: (active term)
 [0 , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , X , X ] -> term_c;" 047:
 [C , 1 ,PADR_c,DIPSW_c, X, 0 , 1 , 1 , 1 , X ] -> wdelay_c;" 048:Write, P9000 not ready
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , 1 , X ] -> wwait_c;" 049:Still not ready
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , 1 , X ] -> wwait_c;" 050:Still not ready
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , 0 , X ] -> wdone_c;" 051:Finally ready
 [1 , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , X , X ] -> term_a;" 052: (active term)
 [0 , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , X , X ] -> term_c;" 053:
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , X , X ] -> idle_c;" 054:Back to idle
 [C , 1 ,PADR_c,DIPSW_c, X, 0 , 1 , 0 , X , X ] -> rreq_c;" 055:Read
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , 0 , X ] -> rwait_c;" 056:
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , 1 , X ] -> rwait_c;" 057:
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , 0 , X ] -> rdone_c;" 058:
 [1 , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , 0 , 0 ] -> term_a;" 059:Same cycle rdyrtn (active
term)
 [0 , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , 0 , 0 ] -> term_c;" 060:Same cycle rdyrtn
 [C , 1 ,PADR_c,DIPSW_c, X, 0 , 1 , 0 , X , X ] -> rreq_c;" 061:Read
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , 1 , X ] -> rrqwait_c;" 062:
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , 1 , X ] -> rrqwait_c;" 063:
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , 0 , X ] -> rwait_c;" 064:
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , 0 , X ] -> rdone_c;" 065:
 [1 , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , X , 1 ] -> rdata_a;" 066:Delayed rdyrtn (active term)
 [0 , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , X , 1 ] -> rdata_c;" 067:Delayed rdyrtn
 [C , 1 ,XADR_c,DIPSW_c, X, 1 , X , X , X , X ] -> idle_c;" 068:

test_vectors
([m_io,adr,dipsw] -> [ldev_])                                   " TEST LDEV GENERATION
 [ 0 ,PADR_c,DIPSW_c] -> [1];                                   " 069:i/o not memory
 [ 1 ,OADR_c,DIPSW_c] -> [1];                                   " 070:address mismatch
 [ 1 ,PADR_c,DIPSW_c] -> [0];                                   " 071:All systems go!

end LOCALSM;
```

10.5.2 REG3CX - P16R4 at U23

```
module REG3CX;
" Options '-reduce bypin fixed';
flag '-r2';
```

```
title 'Register 3C2 and 3C5.';
reg3cx device 'P16R4';

" This PAL implements the 3C2/3C5 registers external to the
" W5x86. These are used to hold the ICD2061A frequency
" select bits, and to hold the P9000 enable and sync
" polarity bits.

" History
" =======
" 2/3/93
" Created file for combo board.

"power
 VCC pin 20;
 GND pin 10;

" inputs
 isareset pin 9;

 miscwr_ pin 1;" Used as a clock
 sa1 pin 2;
 sa0 pin 3;

 bd2 pin 4;
 bd3 pin 5;
 bd4 pin 6;
 bd5 pin 7;

" outputs
   p9syncpolaritypin 17;
 p9videoenable pin 16;

 sel_d pin 15;
 sel_clk pin 14;

   isareset_    pin 12;

" CONSTANTS

 T,F,X,C,Z = 1,0,.X.,.C.,.Z.;

equations

" Register 3C5 holds the video enable and sync polarity bits.
" The address is decoded via the sa1 address bit and the
" miscwr- signal used to clock this register.

p9syncpolarity := !sa1 & bd5
                # sa1 & p9syncpolarity;

p9videoenable := !sa1 & bd4
    # sa1 & p9videoenable;

" Register 3C2 holds the frequency select bits. The address
" is decoded via the sa0 address bit and the miscwr- signal.

sel_d := !sa0 & bd3
```

```
 # sa0 & sel_d;

sel_clk := !sa0 & bd2
 # sa0 & sel_clk;

" Generate an active low reset from the active high isa reset.
isareset_ = !isareset;

test_vectors
([miscwr_,sa1,sa0,bd2,bd3,bd4,bd5]->[sel_clk,sel_d,p9videoenable,p9syncpolarity])
 [ 0 , X , X , X , X , X , X ]->[ X , X , X , X ];
 [ C , 0 , 1 , X , X , 0 , 0 ]->[ X , X , 0 , 0 ]; " write to 3c5
 [ C , 1 , 0 , 0 , 0 , X , X ]->[ 0 , 0 , 0 , 0 ]; " write to 3c2
 [ 0 , X , X , X , X , X , X ]->[ 0 , 0 , 0 , 0 ]; " hold
 [ C , 0 , 1 , X , X , 0 , 1 ]->[ 0 , 0 , 0 , 1 ]; " write to 3c5
 [ 0 , X , X , X , X , X , X ]->[ 0 , 0 , 0 , 1 ]; " hold
 [ C , 0 , 1 , X , X , 1 , 0 ]->[ 0 , 0 , 1 , 0 ]; " write to 3c5
 [ 0 , X , X , X , X , X , X ]->[ 0 , 0 , 1 , 0 ]; " hold
 [ C , 0 , 1 , X , X , 1 , 1 ]->[ 0 , 0 , 1 , 1 ]; " write to 3c5
 [ 0 , X , X , X , X , X , X ]->[ 0 , 0 , 1 , 1 ]; " hold
 [ C , 1 , 0 , 0 , 1 , X , X ]->[ 0 , 1 , 1 , 1 ]; " write to 3c2
 [ 0 , X , X , X , X , X , X ]->[ 0 , 1 , 1 , 1 ]; " hold
 [ C , 1 , 0 , 1 , 0 , X , X ]->[ 1 , 0 , 1 , 1 ]; " write to 3c2
 [ 0 , X , X , X , X , X , X ]->[ 1 , 0 , 1 , 1 ]; " hold
 [ C , 1 , 0 , 1 , 1 , X , X ]->[ 1 , 1 , 1 , 1 ]; " write to 3c2
 [ 0 , X , X , X , X , X , X ]->[ 1 , 1 , 1 , 1 ]; " hold

test_vectors
([isareset]->[isareset_])
 [ 0 ]->[ 1 ];
 [ 1 ]->[ 0 ];

end REG3CX;
```

10.5.3 VIDEO - P16R6 at U12

```
module VIDEO;
" Options '-reduce bypin fixed';
flag '-r2';
title 'Video Clock Generator.';
video device 'P16R6';

" This PAL generates serial clocks and enables for the P9000 frame
" buffer. It also muxes the appropriate load clock and sync to the
" Brooktree 485 RAMDAC, and muxes the appropriate sync signals to
" the video output conector.

" History
" =======
" 2/3/93
" Created file for the combo board.

"power
 VCC pin 20;
 GND pin 10;
```

```
" inputs
   clk                    pin 1; " dacload gets fed back to this pin
   sclk                   pin 2;
   vidoutclk     pin 4;

   p9000enable   pin 3;

   vgablank_     pin 6; vgablank macro {!vgablank_};
   p9000cblank_  pin 5; p9000cblank macro {!p9000cblank_};

   oe_                    pin 11;

" outputs
   " Registered outputs
   sc0                    pin 16;
   sc1                    pin 15;
   se0_                   pin 14; se0 macro {!se0_};
   se1_                   pin 13; se1 macro {!se1_};

   delp9cblank_ pin 17;
   ddelp9cblank_pin 18;

   " Combinatorial outputs
   dacload                pin 19;
   dacblank_     pin 12;

" Logical value/clock definitions
 T,F,X,C,Z = 1,0,.X.,.C.,.Z.;

equations
   " Multiplex the appropriate clock to the RAMDAC load signal.
   " This was put in this PAL since there shouldn't be too much
   " delay in this clock path. (Notice the vidoutclk is inverted,
   " this implements a digital delay line.)

   dacload = p9000enable & sclk
             # !p9000enable & !vidoutclk;

   " Multiplex the appropriate blank to the RAMDAC.

   dacblank_ = p9000enable & ddelp9cblank_
               # !p9000enable & vgablank_;

   " Generate the delayed blank which is sent to the RAMDAC when
   " the P9000 is enabled. The blank must be delayed to keep it
   " in sync with the serial clocks and enables.

   delp9cblank_ := p9000cblank_;
   ddelp9cblank_ := delp9cblank_;

   " Generate the serial clocks and enables for the P9000
   " framebuffer. (This could also be done some DFF's and a couple
   " of AND gates.)

   sc0 := !sc0 & !p9000cblank;
   sc1 := sc0;
   se0_ := !sc0;
   se1_ := se0_;
```

```
test_vectors
([clk,p9000cblank_] -> [sc0, sc1, se0_, se1_])
 [ C , 0 ] -> [ X , X , X , X ]; " 1
 [ C , 0 ] -> [ X , X , X , X ]; " 2
 [ C , 0 ] -> [ X , X , X , X ]; " 3
 [ C , 0 ] -> [ X , X , X , X ]; " 4
 [ C , 0 ] -> [ 0 , 0 , 1 , 1 ]; " 5
 [ C , 1 ] -> [ 1 , 0 , 1 , 1 ]; " 6
 [ C , 1 ] -> [ 0 , 1 , 0 , 1 ]; " 7
 [ C , 1 ] -> [ 1 , 0 , 1 , 0 ]; " 8
 [ C , 1 ] -> [ 0 , 1 , 0 , 1 ]; " 9
 [ C , 1 ] -> [ 1 , 0 , 1 , 0 ]; " 10
 [ C , 1 ] -> [ 0 , 1 , 0 , 1 ]; " 11
 [ C , 1 ] -> [ 1 , 0 , 1 , 0 ]; " 12

test_vectors
([clk,p9000cblank_,vgablank_,p9000enable]->[delp9cblank_,ddelp9cblank_, dacblank_])
 [ C , 0 , X , 1 ]->[ 0 , X , X ];
 [ C , 0 , X , 1 ]->[ 0 , 0 , 0 ];
 [ C , 0 , X , 1 ]->[ 0 , 0 , 0 ];
 [ 0 , 1 , X , 1 ]->[ 0 , 0 , 0 ];
 [ C , 1 , X , 1 ]->[ 1 , 0 , 0 ];
 [ C , 1 , X , 1 ]->[ 1 , 1 , 1 ];
 [ 0 , 1 , X , 1 ]->[ 1 , 1 , 1 ];
 [ 0 , 0 , X , 1 ]->[ 1 , 1 , 1 ];
 [ C , 0 , X , 1 ]->[ 0 , 1 , 1 ];
 [ C , 0 , X , 1 ]->[ 0 , 0 , 0 ];
 [ 0 , X , 0 , 0 ]->[ X , X , 0 ];
 [ 0 , X , 1 , 0 ]->[ X , X , 1 ];

test_vectors
([sclk,vidoutclk,p9000enable]->[dacload])
 [ 0 , 0 , 0 ]->[ 1 ];
 [ 0 , 1 , 0 ]->[ 0 ];
 [ 1 , 0 , 0 ]->[ 1 ];
 [ 1 , 1 , 0 ]->[ 0 ];
 [ 0 , 0 , 1 ]->[ 0 ];
 [ 0 , 1 , 1 ]->[ 0 ];
 [ 1 , 0 , 1 ]->[ 1 ];
 [ 1 , 1 , 1 ]->[ 1 ];
end VIDEO;
```

10.5.4 RAMDAC - P16L8 at U27

```
module RAMDAC;
" Options '-reduce bypin fixed';
flag '-r2';
title 'Ramdac ISA Bus Interface.';
ramdac device 'P16L8';

" This PAL generates the dacrd/dacwr control signals for the Bt485 RAMDAC
" used on the P9000 VESA LocalBus card. It also generates the buffer control
" signal for the external ISA bus transceivers.
"
"power
        VCC       pin 20;
        GND       pin 10;
```

```
" inputs

        sa0pin 1;
        sa1     pin 2;
        sa2     pin 3;
        sa3     pin 4;
        sa4     pin 5;
        sa5     pin 6;
        sa6     pin 7;
        sa7     pin 8;
        sa8     pin 9;
        sa9     pin 11;
    sa = [sa9,sa8,sa7,sa6,sa5,sa4,sa3,sa2,sa1,sa0];

    aen                     pin 13;
    iowc_                   pin 14; iowc macro {!iowc_};
    iorc_                   pin 15; iorc macro {!iorc_};
    busout_                 pin 16; busout macro {!busout_};

" outputs
    dacwr_                  pin 19; dacwr macro {!dacwr_};
    dacrd_                  pin 18; dacrd macro {!dacrd_};
    isabusout_              pin 12; isabusout macro {!isabusout_};

" Logical value/clock definitions
        T,F,X,C,Z = 1,0,.X.,.C.,.Z.;

equations
    dacwr = !aen & iowc & ((sa == ^h3c6) #
                           (sa == ^h3c7) #
                           (sa == ^h3c8) #
                           (sa == ^h3c9));

    dacrd = !aen & iorc & ((sa == ^h3c6) #
                           (sa == ^h3c7) #
                           (sa == ^h3c8) #
                           (sa == ^h3c9));
    " We have to enable the ISA tranceivers when reads are made from the RAMDAC.
    " Since the 5186 won't know when all the reads happen we have to squeeze in
    " this extra term.

    isabusout = busout # dacrd;

test_vectors
([sa     , aen, iowc_, iorc_, busout_] -> [dacwr_, dacrd_, isabusout_])
 [ ^h000,  0 ,  0   ,  0   ,   1    ] -> [  1   ,   1   ,      1     ]; " Ignored zero address
 [ ^h3c6,  0 ,  0   ,  1   ,   1    ] -> [  0   ,   1   ,      1     ]; " All RAMDAC writes
 [ ^h3c7,  0 ,  0   ,  1   ,   1    ] -> [  0   ,   1   ,      1     ];
 [ ^h3c8,  0 ,  0   ,  1   ,   1    ] -> [  0   ,   1   ,      1     ];
 [ ^h3c9,  0 ,  0   ,  1   ,   1    ] -> [  0   ,   1   ,      1     ];
 [ ^h3c6,  0 ,  1   ,  0   ,   1    ] -> [  1   ,   0   ,      0     ]; " All RAMDAC reads
 [ ^h3c7,  0 ,  1   ,  0   ,   1    ] -> [  1   ,   0   ,      0     ];
 [ ^h3c8,  0 ,  1   ,  0   ,   1    ] -> [  1   ,   0   ,      0     ];
 [ ^h3c9,  0 ,  1   ,  0   ,   1    ] -> [  1   ,   0   ,      0     ];
 [ ^h000,  1 ,  1   ,  1   ,   0    ] -> [  1   ,   1   ,      0     ]; " Other read from 5x86
                                                                        end RAMDAC;
```

10.6 Switch Settings

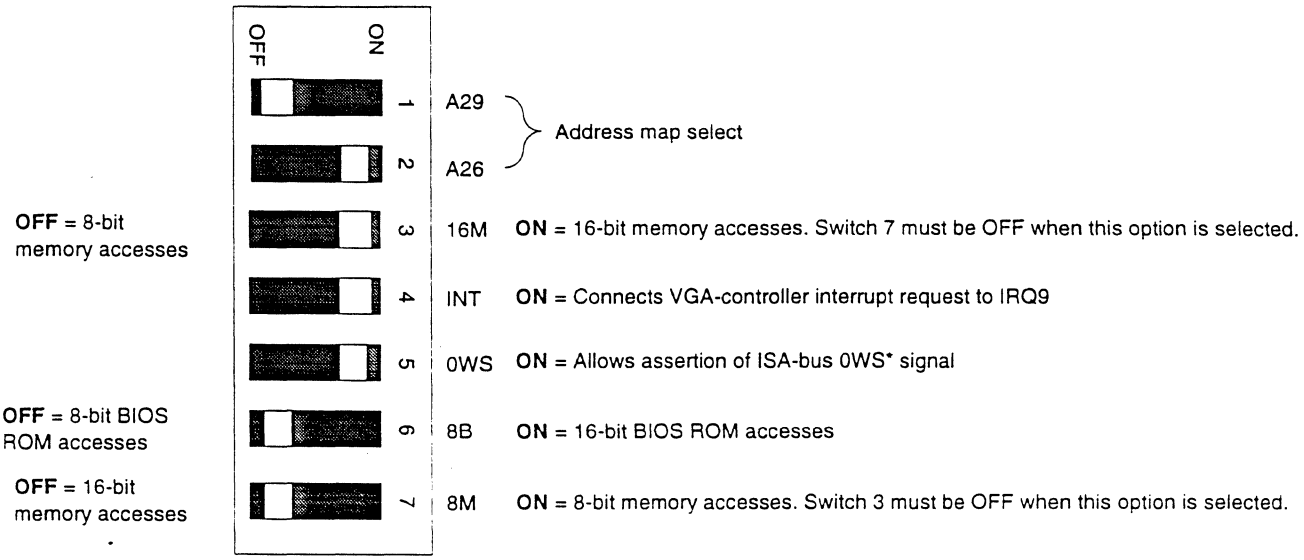


Figure 10-11. DIP Switch Functions and VESA® Default Settings

Table 10-5. Memory-Width Settings, Typical Uses

8M	8B	16M	Frame Buffer	BIOS ROM
OFF	OFF	ON	16 bits	8 bits
OFF	ON	ON	16 bits	16 bits
ON	OFF	OFF	8 bits	8 bits
ON	ON	OFF	8 bits	8 bits

Table 10-6. Address Mapping

A29	A26	Base Address, VESA Low
OFF	OFF	E000 0000
OFF	ON	C000 0000
ON	OFF	8000 0000
ON	ON	0400 0000

The CL-PX2085 board uses the VESA-Low address range (BusType=VESALow in P9000RES.DAT).

11. VAFC™ VIDEO CARD DESIGN CONSIDERATIONS

Previous application notes provide details of VAFC graphics card design. This application note briefly describes the other side of the connector — the VAFC video card.

11.1 The VAFC™ Port, Viewed from the Video Card

In this application, the VAFC port receives a video data stream from the VRAM serial outputs. The port can be treated as an extension of the interface between the CL-PX2085 and the CL-PX2070, with the CL-PX2070 as controller.

Figure 11-1 shows the VAFC interface. Two VAFC signals are not actively used. VRDY, which drives VWE, is tied to VCC. EVIDEO* is typically tied to ground, although in more complex systems EVIDEO* can be driven to control the data-flow direction across the connector, allowing use in a VAFC 'memory-attach' design. The CL-PX2070 monitors signals HSYNC, VSYNC, and BLANK (passed through the CL-PX2085 by means of BLANKOUT*) generated by the graphics controller, and CL-PX2085 signals PCO, DCLK,and GRDY to synchronize overlays and timing.

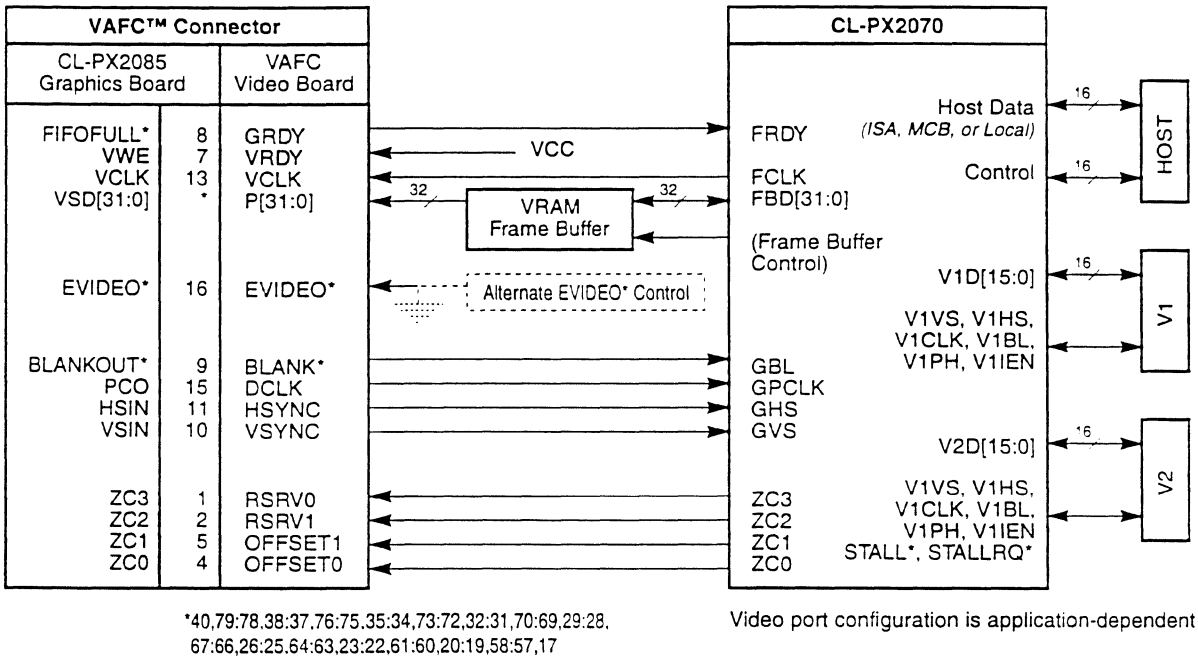


Figure 11-1. CL-PX2070 Video Card for Use with CL-PX2085-Based VAFC™ Graphics Card

The CL-PX2070 accepts and processes video data in realtime from three different sources:

- Host Interface — may be configured as ISA, MicroChannel®, or local hardware interface
- Video Port 1 (V1) — 16-bit bidirectional port with bidirectional control signals
  - — may be configured as input-only, output-only, or duplex
  - — block transfer mode to accommodate compression/decompression devices
- Video Port 2 (V2) — identical to V1 with the addition of 'handshaking' signals STALL* and STALLRQ*

These three flexible interfaces can be configured in a variety of application-specific ways. The interface treatment is the primary differentiating factor among VAFC video cards. For more information, contact a Cirrus Logic sales office and request the CL-PX2070 technical reference manual. Refer to the application note entitled "Video-Port Interface Techniques".

Possible applications of a CL-PX2070-based VAFC video card include:

- JPEG-based video editing adapter
- Px64 videoteleconferencing card
- Video phone for use over standard voice line.
- MPEG decoder from CD-ROM
- Time-based corrector/frame synchronizer

11.2 Adapting for Use with a VAFC™ Video Card

The VAFC port can be treated as an extension of the original CL-PX2070/'85 interface. An existing design incorporating these devices can be upgraded to VAFC capability by modifying the board layout to include the VAFC port. The board can then be partially populated to suit the application, thereby reducing materials cost. The resulting board can be used in the following ways:

- To use the existing design with a reduced-cost VAFC video card: install the connector, but not the CL-PX2085.
- To use the design as a stand-alone video/graphics card: install the CL-PX2085, but not the connector.

Figure 11-2 shows a block diagram of the CL-PX2070/'80 Development System with a VAFC port added. Refer to the CL-PX2070/'80 Development System documentation and schematics for more information. The development system is available as Cirrus Logic part number CL-PXK2070/'80-DEV. Optional video modules are available as part number CL-PXK7191-DEV.

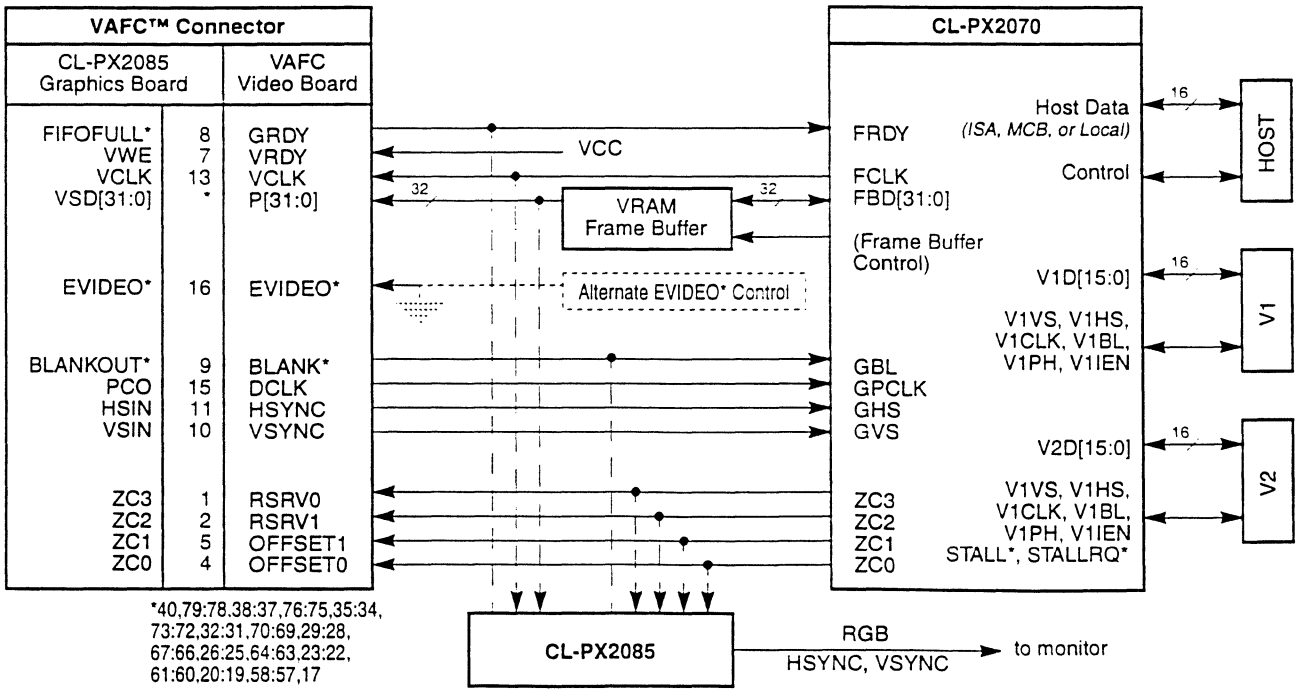


Figure 11-2. Existing CL-PX2070/'85 Design With VAFC™ Port

PART IV: REFERENCE

Section Page

Appendix A. CL-PX2080/CL-PX2085 Differences

This appendix summarizes the differences between the CL-PX2085 and the CL-PX2080.

CL-PX2085 Feature	Description
Clock Selection	The CL-PX2085 contains an internal clock doubler that extends the pixel-clock frequency to 135 MHz, simplifying the design of systems capable of driving 1280 x 1024 displays at high refresh rates. The clock doubler is enabled using register GCR[2CLK].
VSVI Port	Video port that complies with the VESA Advanced Video Interface (VAVI). See Section 4.9.4 on page 74 for additional information.
VAFC Output Mode	The CL-PX2085 outputs VGA[7:0] or GSD[31:0] data on pins VSD[31:0]. The VGA[7:0]/GSD[31:0] data sampled on the rising edge of LCLK is output through VSD[31:0]. The format of the GSD[31:0] data is unchanged from the input.
Default Power-Up Conditions	When the CL-PX2085 powers up, VAFC VESA Advanced Feature Connector[OE*] is reset to '0'; VGA[7:0] is driven out on VSD[7:0], with VSD[31:8] in output mode. EVIDEO* is asserted by an external device when data is to be driven into VSD[31:0].
VAFC Input Mode	The CL-PX2085 accepts input data on VSD[31:0]. VAFC VESA Advanced Feature Connector[VM] is defined to support VAVI 16- and 32-bit modes. VAFC VESA Advanced Feature Connector[V*] defines the VAVI mode. When VAFC VESA Advanced Feature Connector[V*] is '0', the device is in standard VAVI mode, and VAFC VESA Advanced Feature Connector[VM] is used to select the interface mode of the VAVI connector. Offset pins are not supported. In 32-bit 2x mode, the CL-PX2085 interpolates by inserting appropriate zoom codes for a 2x zoom after the FIFO.
64 x 64 Hardware Cursor	In addition to the CL-PX2080's 32 x 32-pixel, 2-plane hardware cursor, the CL-PX2085 has a 64 x 64 hardware cursor mechanism. The 64 x 64 cursor is identical to the 32 x 32 cursor in all areas of operation, except for the 64 x 64 size and the array-access method. Only one of the two hardware cursor mechanisms can be enabled at a time. The cursor size is configured using register GCR[CU64].
32 x 32 Cursor RAM Access	The 32 x 32 cursor RAM uses an 8-bit address to access a 128 x 8 x 2-bit RAM array, as described in the *CL-PX2080 Data Book*.
64 x 64 Cursor RAM Access	The 64 x 64 cursor operates in the same way as the 32 x 32 cursor — two-bit planes allow the color and transparency of each pixel in the cursor to be individually controlled. The three operational modes are unchanged from the CL-PX2080. The RAM-array size for the 64 x 64 cursor is 512 x 8 x 2. Since this size requires a 10-bit address, two additional planes are provided in GCR[A9,A8]. In the 32 x 32 cursor, A7 selects the bit plane to access. In the 64 x 64 cursor, GCR[A9] selects the bit plane.
Register Access Modes	The new register-access mode 3 (11 binary on pins BS[1:0]) allows more convenient access of the 64 x 64 cursor RAM. It uses only RS[3:0] and a block index in GCR[BLK] to address the entire extended register set. GCR, which is at the same physical I/O and index address as GSR (a read-only register), can be accessed by writing a '1' to ASC[GCRE].

INDEX

Symbols

Numerics

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

V

X

Y

Z

INDEX OF CONTROL REGISTERS

Numerics

A

B

C

D

E

G

G

H

I

L

M

P

R

S

T

V

W

X

Y

Direct Sales Offices

Domestic

N. CALIFORNIA
Fremont
TEL: 510/623-8300
FAX: 510/252-6020

Sacramento
TEL: 916/933-4200
FAX: 916/933-4211

S. CALIFORNIA
Tustin
TEL: 714/573-9911
FAX: 714/573-4665

Thousand Oaks
TEL: 805/371-5381
FAX: 805/371-5382

NORTHWESTERN AREA
Portland, OR
TEL: 503/620-5547
FAX: 503/624-5665

ROCKY MOUNTAIN AREA
Denver, CO
TEL: 303/786-9696
FAX: 303/786-9695

SOUTH CENTRAL AREA
Austin, TX
TEL: 512/255-0080
FAX: 512/255-0733

Dallas, TX
TEL: 214/252-6698
FAX: 214/252-5681

Houston, TX
TEL: 713/379-5772
FAX: 713/379-4341

CENTRAL AREA
Chicago, IL
TEL: 708/981-6950
FAX: 708/981-6846

NORTHEASTERN AREA
Andover, MA
TEL: 508/474-9300
FAX: 508/474-9149

Boston, MA
TEL: 617/721-1439
FAX: 617/721-4509

Iselin, NJ
TEL: 908/632-2771
FAX: 908/632-2914

Philadelphia, PA
TEL: 215/625-0781
FAX: 215/625-0731

SOUTHEASTERN AREA
Atlanta, GA
TEL: 404/623-4653
FAX: 404/497-0414

Boca Raton, FL
TEL: 407/241-5777
FAX: 407/241-7990

Raleigh, NC
TEL: 919/481-9610
FAX: 919/481-9640

International

GERMANY
Herrsching
TEL: 49/8152-40084
FAX: 49/8152-40077

FRANCE
Rosny sous bois
TEL: 33/1-48-122812
FAX: 33/1-48-122810

HONG KONG
Tsimshatsui
TEL: 852/376-0801
FAX: 852/375-1202

JAPAN
Tokyo
TEL: 81/3-3340-9111
FAX: 81/3-3340-9120

KOREA
Seoul
TEL: 82/2-565-8561
FAX: 82/2-565-8565

SINGAPORE
TEL: 65/353-2122
FAX: 65/353-2166

TAIWAN
Taipei
TEL: 886/2-718-4533
FAX: 886/2-718-4526

UNITED KINGDOM
Hertfordshire, England
TEL: 44/0727-872424
FAX: 44/0727-875919

The Company

Headquartered in Fremont, California, Cirrus Logic, Inc., develops innovative architectures for analog and digital system functions. The Company implements those architectures in proprietary integrated circuits and related software for applications that include user interface and multimedia (graphics, audio, and video), mass storage, communications, and data acquisition.

Key markets for Cirrus Logic's products include desktop and portable computing, workstations, telecommunications, and consumer electronics.

The Cirrus Logic formula combines innovative architectures in silicon with system design expertise. We deliver complete solutions — chips, software, evaluation boards, and manufacturing kits — on-time, to help you win in the marketplace.

Cirrus Logic's manufacturing strategy, unique in the semiconductor industry, employs a full manufacturing infrastructure to ensure maximum product quality, availability, and value for our customers.

Talk to our systems and applications specialists; see how you can benefit from a new kind of semiconductor company.

© Copyright, Cirrus Logic, Inc., 1994

Advance product information describes products that are in development and subject to developmental changes. Cirrus Logic, Inc., has made best efforts to ensure that the information contained in this document is accurate and reliable. However, the information is subject to change without notice. No responsibility is assumed by Cirrus Logic, Inc., for the use of this information, nor for infringements of patents or other rights of third parties. This document is the property of Cirrus Logic, Inc., and implies no license under patents, copyrights, or trade secrets. No part of this publication may be copied, reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photographic, or otherwise, or used as the basis for manufacture or sale of any items without the prior written consent of Cirrus Logic, Inc. Cirrus Logic, AutoMap, Fair Share, FeatureChips, Good Data, MotionVideo, MVA, SimulSCAN, S/LA, SoftTarget, UXART, Vision Port, and WavePort are trademarks of Cirrus Logic, Inc. Other trademarks in this document belong to their respective companies. Cirrus Logic, Inc., products are covered by the following U.S. patents: 4,293,783; Re. 31,287; 4,763,332; 4,777,635; 4,839,896; 4,931,946; 4,975,828; 4,979,173; 5,032,981; 5,122,783; 5,131,015; 5,140,595; 5,157,618; 5,179,292; 5,185,602; 5,220,295; 5,280,488; 5,287,241; 5,291,499; 5,293,159; 5,293,474; 5,297,184; 5,298,915; 5,300,835; 5,311,460; 5,313,224; 5,327,128; 5,329,554; 5,351,231. Additional patents pending.

CIRRUS LOGIC, Inc., **3100 West Warren Ave.** **TEL: 510/623-8300**
Fremont, CA 94538 **FAX: 510/252-6020**