S-MOS
SYSTEMS
A Seiko Epson Affiliate

SPC8110F0A Local Bus LCD/CRT VGA Controller

SPC8110F0A
TECHNICAL MANUAL

Issue Date: 98/01/27

Document No. X07G-Q-001-05

Copyright © 1996, 1998 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

CUSTOMER SUPPORT INFORMATION

Comprehensive Support Tools

S-MOS Systems provides to the system designer and computer OEM manufacturer a complete set of resources and tools for the development of VGA Graphics Systems.

Evaluation / Demonstration Board

- Assembled and fully tested graphics evaluation board with installation guide and schematics
- To borrow an evaluation board, please contact your local S-MOS sales representative or contact S-MOS at (408) 922-0200 Ext. 3440

VGA Chip Documentation

- Technical manual includes Data Sheet, Application Notes, and Programmer's Reference
- To remain on the mailing list for updates to this manual, please FAX the *Acknowledgment of Receipt* form to (408) 922-0238

Software

- Video BIOS
- OEM Utilities
- User Utilities
- Evaluation Software
- To download these programs, contact Application Engineering Support

Application Engineering Support

Engineering and Sales Support is provided by:

Head Office
S-MOS Systems Inc.
150 River Oaks Parkway
San Jose, CA 95134
Tel: (408) 922-0200
Fax: (408) 922-0238
http://www.smos.com

North Central USA
1450 East American Lane
Suite 1550
Schaumburg, Illinois 60173
Tel: (847) 517-7667
Fax: (847) 517-7601
Cell: (847) 558-6662
Pager: (888) 789-3471

Eastern USA
301 Edgewater Place
Suite 120
Wakefield, MA 01880
Tel: (781) 246-3600
Fax: (781) 246-5443

Southeast USA
4300 Six Forks Road
Suite 430
Raleigh, NC 27609
Tel: (919) 781-7667
Fax: (919) 781-6778

THIS PAGE LEFT BLANK

TABLE OF CONTENTS

INTRODUCTION
SPC8110F0A Data Sheet

SPECIFICATION
SPC8110F0A Hardware Functional Specification

PROGRAMMER'S REFERENCE
SPC8110F0A BIOS Functional Specification
SPC8110F0A Programming Notes and Examples

UTILITIES
8110CFG.EXE BIOS Configuration Utility
BOLD.EXE Display Utility
DISPLAY.COM Utility
HD.EXE Display Utility
LCDF8110.EXE Display Utility
LOADROM.EXE BIOS Utility
PLLSET.EXE Display Utility
PWRSAVE.EXE Power Save Utility
READCHIP.EXE Diagnostic Utility
REVERSE.COM Display Utility
SESHELL.EXE Shell Utility
SETFONT.EXE Display Utility
SHOWMODE.EXE Demonstration Program
SM.EXE Display Utility
SPRITEST.EXE Display Utility
VESADDC.EXE Display Utility
VESAPD.EXE Display Utility
VRTEXP.EXE Display Utility
WGS.EXE Windows Gray-Scale Utility

DRIVERS
SPC8110F0A Windows 3.1x Display Driver
SPC8110F0A Windows 95 / Windows NT 4.0 Display Driver
SPC8110F0A Windows NT 3.5x Display Driver
SPC8110F0A Autocad Display Driver

EVALUATION
SDU8110B0C Rev 1.0 VL-Bus Evaluation Board
SDU8110B0E Rev 1.0 PCI Bus Evaluation Board

APPLICATION NOTES
SPC8110F0A Power Consumption
PCB Layout Guidelines
SDU8110B0x PCI Boot Procedures
External Reference Component Values
Planar EL640.480-AA1 Panel Interface
Optrex DMF-50418NF-FW Panel Interface

THIS PAGE LEFT BLANK

January 1998

SPC8110F0A LOCAL BUS LCD/CRT VGA CONTROLLER

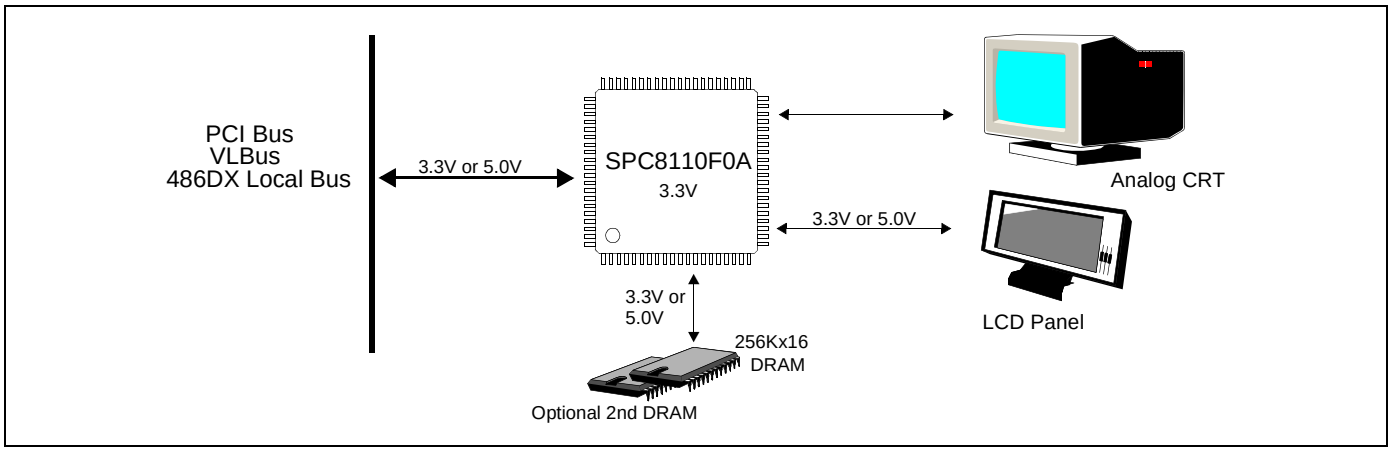
■ DESCRIPTION

The SPC8110 is a single chip, multi-function, mixed voltage LCD/CRT VGA controller. This controller features an integrated RAMDAC, PLL, Bit Block Transfer engine (BitBLT) and a CPU local bus interface. The SPC8110 accelerates the display of graphical user interface software on flat panel and analog CRT displays.

■ KEY FEATURES

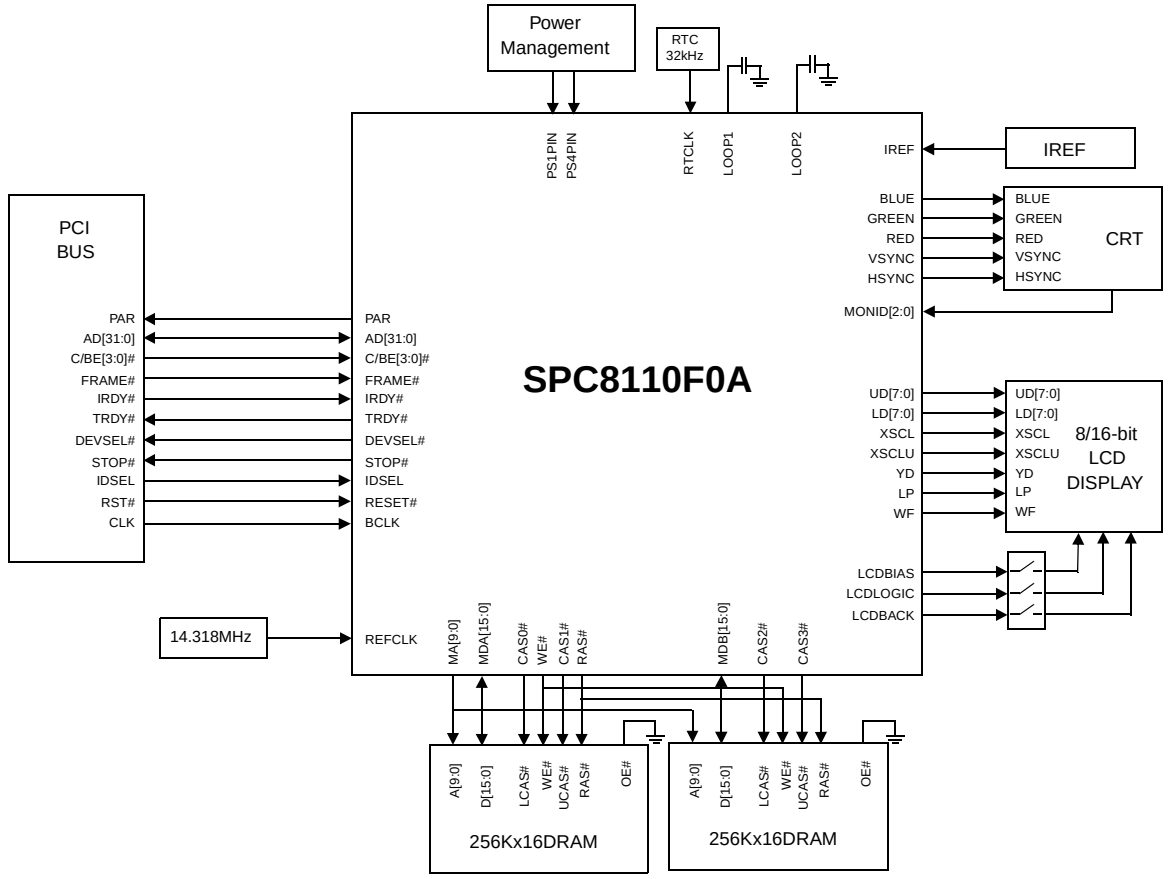
- Hardware VGA compatible
- 32-bit PCI, VL-Bus or 486DX local bus direct Interface
- 512KB or 1024KB display memory using one or two 256K x 16 self-refresh DRAM, respectively (CAS or WE controlled; symmetrical or asymmetrical addressing)
- Hardware Bit Block Transfer engine
- Hardware 64 x 64 pixel 2-bit cursor
- Hardware color expansion
- Linear mode addressing
- Internal PLL and clock generation
- Internal 256 x 18-bit RAMDAC
- Direct analog CRT drive
- 9/12/16/18-bit TFT LCD Panel support
- 4/8/16-bit passive LCD Panel support
- Multi-resolution LCD Panel interface: dual panel-dual drive, single panel-single drive
- 16, 32 & 64 LCD gray shades by Frame Rate Modulation (FRM) and dithering
- Maximum 256 simultaneous colors from a possible 256K colors on LCD or CRT
- Simultaneous LCD and analog CRT display
- Vertical centering and expansion
- LCD panel power sequencing
- Extensive hardware and software activated power save modes and status signals
- 3.3/5.0 mixed voltage operation
- 208 pin QFP8 S1 package

■ BLOCK DIAGRAM



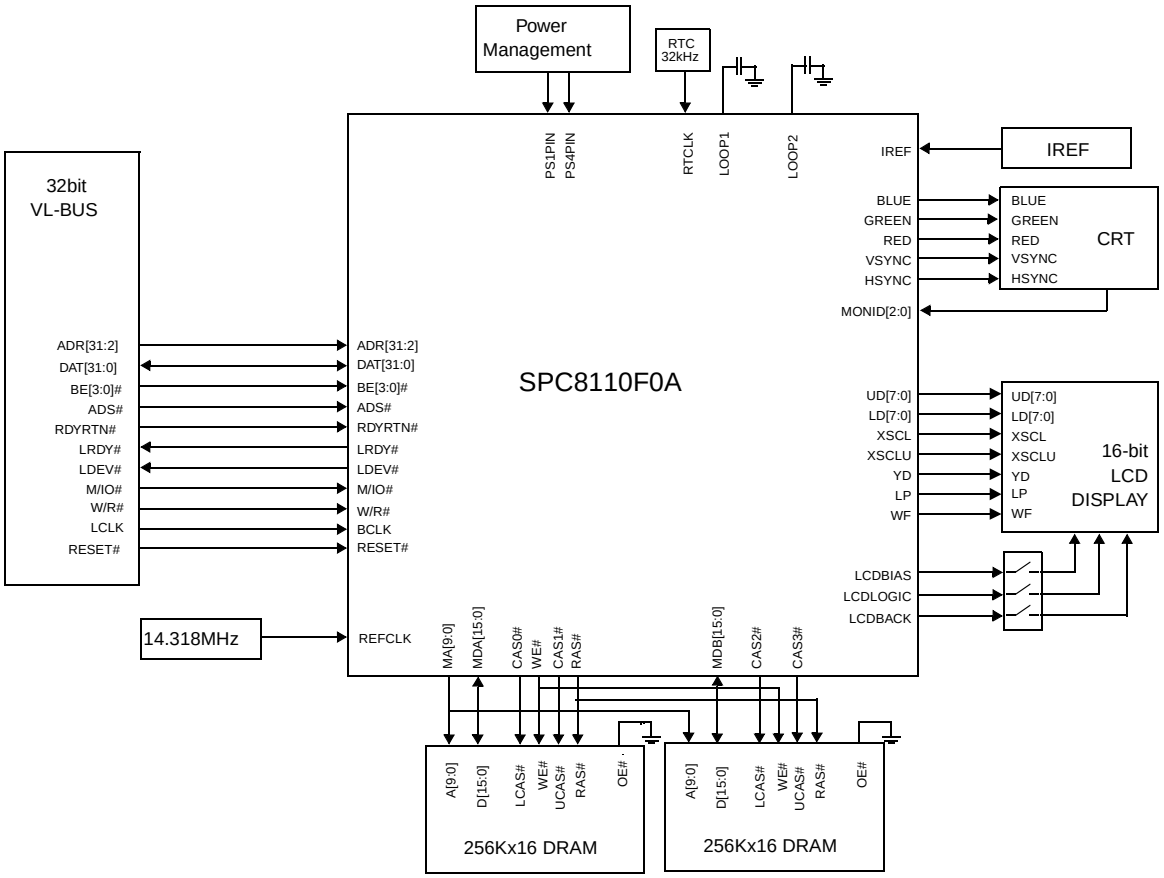
■ PCI BUS SYSTEM BLOCK DIAGRAM

Shown with 1024 KB memory option



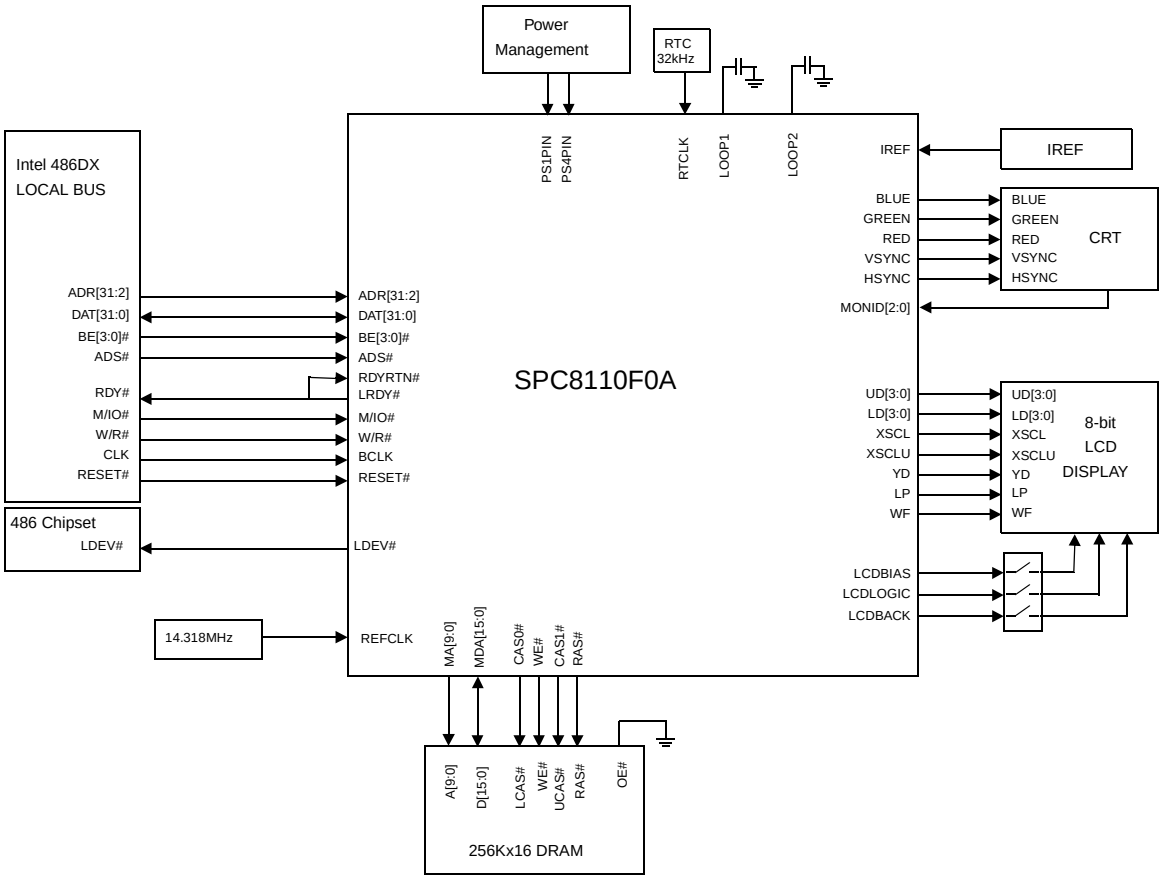
■ VL-BUS SYSTEM BLOCK DIAGRAM

Shown with 1024 KB memory option



■ 486DX-33 LOCAL BUS SYSTEM BLOCK DIAGRAM

Shown with 512 KB memory option



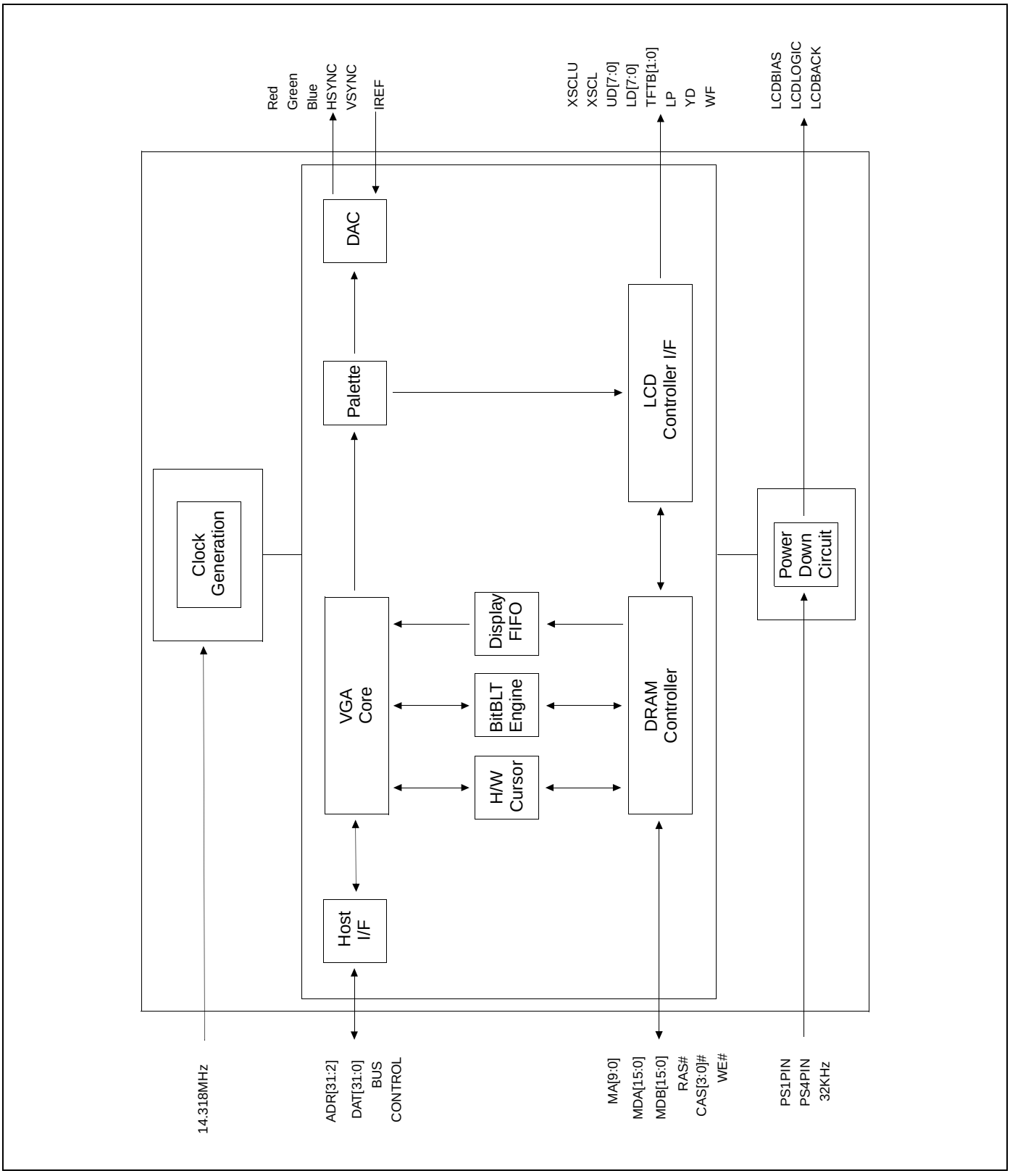
■ SPC8110 VIDEO MODES

Mode No. (Hex)	Pixels Addressable		Horizontal Pixels Displayed		Vertical Pixels Displayed	Font Size		Number of Shades			Nominal Frame Rate (Hz) for 1024KB RAM		
	Horizontal (characters)	Vertical (rows)	CRT	LCD		CRT	LCD	Mono LCD	Color LCD	CRT	CRT (typical)	Single Panel LCD	Dual Panel LCD
0	(40)	(25)	320	320	200	8x8	8x8	16	16	16	70	79	78
0+	(40)	(25)	320	320	350	8x14	8x14	16	16	16	70	79	78
0++	(40)	(25)	360	320	400	9x16	8x16	16	16	16	70	79	78
1	(40)	(25)	320	320	200	8x8	8x8	16	16	16	70	79	78
1+	(40)	(25)	320	320	350	8x14	8x14	16	16	16	70	79	78
1++	(40)	(25)	360	320	400	9x16	8x16	16	16	16	70	79	78
2	(80)	(25)	640	640	200	8x8	8x8	16	16	16	70	79	78
2+	(80)	(25)	640	640	350	8x14	8x14	16	16	16	70	79	78
2++	(80)	(25)	720	640	400	9x16	8x16	16	16	16	70	79	78
3	(80)	(25)	640	640	200	8x8	8x8	16	16	16	70	79	78
3+	(80)	(25)	640	640	350	8x14	8x14	16	16	16	70	79	78
3++	(80)	(25)	720	640	400	9x16	8x16	16	16	16	70	79	78
4	320	200	320	320	200	n/a	n/a	4	4	4	70	79	78
5	320	200	320	320	200	n/a	n/a	4	4	4	70	79	78
6	640	200	640	640	200	n/a	n/a	2	2	2	70	79	78
7	(80)	(25)	640	640	350	8x14	8x14	2	2	2	70	79	78
7+	(80)	(25)	720	640	400	9x16	8x16	2	2	2	70	79	78
0D	320	200	320	320	200	n/a	n/a	16	16	16	70	79	78
0E	640	200	640	640	200	n/a	n/a	16	16	16	70	79	78
0F	640	350	640	640	350	n/a	n/a	2	2	2	70	79	78
10	640	350	640	640	350	n/a	n/a	16	16	16	70	79	78
11	640	480	640	640	480	n/a	n/a	2	2	2	60	79	78
12	640	480	640	640	480	n/a	n/a	16	16	16	60	79	78
13	320	200	320	320	200	n/a	n/a	64	256	256	70	79	78
6A	800	600	800	800	600	n/a	n/a	16	16	16	72	60	72
100	640	400	640	640	400	n/a	n/a	64	256	256	70	79	78
101	640	480	640	640	480	n/a	n/a	64	256	256	60	79	78
102	800	600	800	800	600	n/a	n/a	16	16	16	72	60[2]	72
103	800	600	800	800	600	n/a	n/a	64	256	256	72	60[2]	72[1]
104	1024	768	1024	1024	768	n/a	n/a	16	16	16	60	n/a[2]	70
105	1024	768	1024	n/a	768	n/a	n/a	64	n/a	256	60[1]	n/a[2]	n/a
108	(80)	(60)	640	640	480	8x8	8x8	16	16	16	60	79	78
109	(132)	(25)	1056	n/a	350	8x14	n/a	n/a	n/a	16	70	n/a	n/a
10A	(132)	(43)	1056	n/a	350	8x8	n/a	n/a	n/a	16	70	n/a	n/a
10B	(132)	(50)	1056	n/a	400	8x8	n/a	n/a	n/a	16	70	n/a	n/a
10C	(132)	(60)	1056	n/a	480	8x8	n/a	n/a	n/a	16	60	n/a	n/a

1. Only available with 1024KB RAM.
2. Support for large TFT panels requires custom BIOS and drivers.

S-MOS SYSTEMS
A Seiko Epson Affiliate

■ FUNCTIONAL BLOCK DIAGRAM



S-MOS Systems, Inc. • 150 River Oaks Parkway • San Jose, California 95134 • Tel:(408) 922-0200 • Fax:(408) 922-0238

■ FUNCTIONAL BLOCK DESCRIPTION

Host Interface

The Host Interface can be programmed to accommodate any of the following three standards: Intel486 DX local bus interface, VL-Bus interface and PCI interface. It has a one-stage buffer for zero wait-state write operation.

Clock Generator

The Clock Generator contains two PLLs that are separately programmed to produce the memory and pixel clocks from a single clock source. The reference clock is typically 14.318 MHz.

VGA Core

The VGA Core contains the Sequencer, CRTC Controller, Graphics Controller, Attribute Controller, and the rest of the standard VGA circuitry.

Hardware Cursor

The Hardware Cursor generates a 64 x 64 x 2-bit hardware cursor or sprite that can be overlaid on the displayed image.

BitBLT Engine

The Bit Block Transfer Engine performs read, write, and move blits, solid fills, destination inversions, and pattern fills. It performs all data alignment and masking at the blit boundary and also performs color expansion to accelerate the writing of text images. The Bit Block Transfer Engine operates in both 4-bit planar mode and 8-bit linear (packed-pixel) mode.

Display FIFO

The Display FIFO is an 8 stage FIFO that is used to buffer the video data from display memory.

VGA Palette

The VGA Palette implements the standard 256-word x 18-bit VGA lookup table, plus 4-word x 18-bit entries for Hardware Sprites.

DAC

The DAC functions as a triple 6-bit 65 MHz DAC to drive the RGB outputs connected to the analog display.

LCD Interface

The LCD Interface contains frame rate modulation and dithering circuitry displaying a maximum of 64 shades of gray in monochrome LCD mode. In color LCD mode, it uses frame rate modulation (FRM) to display 256 out of a possible 4096 colors, and additional dithering techniques for a full 256K possible colors.

Power Save Logic

Power Save Logic implements all the power down features of the chip.

■ ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings

Symbol	Parameter	Rating	Units
V_{DD}	Supply Voltage	V_{SS} - 0.3 to 7.0	V
V_{IN}	Input Voltage	V_{SS} - 0.3 to V_{DD} + 0.3	V
V_{OUT}	Output Voltage	V_{SS} - 0.3 to V_{DD} + 0.3	V
T_{STG}	Storage Temperature	-65 to 150	°C
T_{SOL}	Solder Temperature/Time	260 for 10 sec. max at lead	°C

Recommended Operating Conditions

Symbol	Parameter	Condition	Min	Typ	Max	Units
HV_{DD}	Supply Voltage	V_{SS} = 0 V	4.5	5.0	5.5	V
LV_{DD}	Supply Voltage	V_{SS} = 0 V	3.0	3.3	3.6	V
V_{IN}	Input Voltage	V_{SS}	V_{SS}	--	V_{DD}	V
T_{OPR}	Operating Temperature		0	25	70	°C
I_{OPR}	Average Power Consumption			381.7		mW

Input Specifications

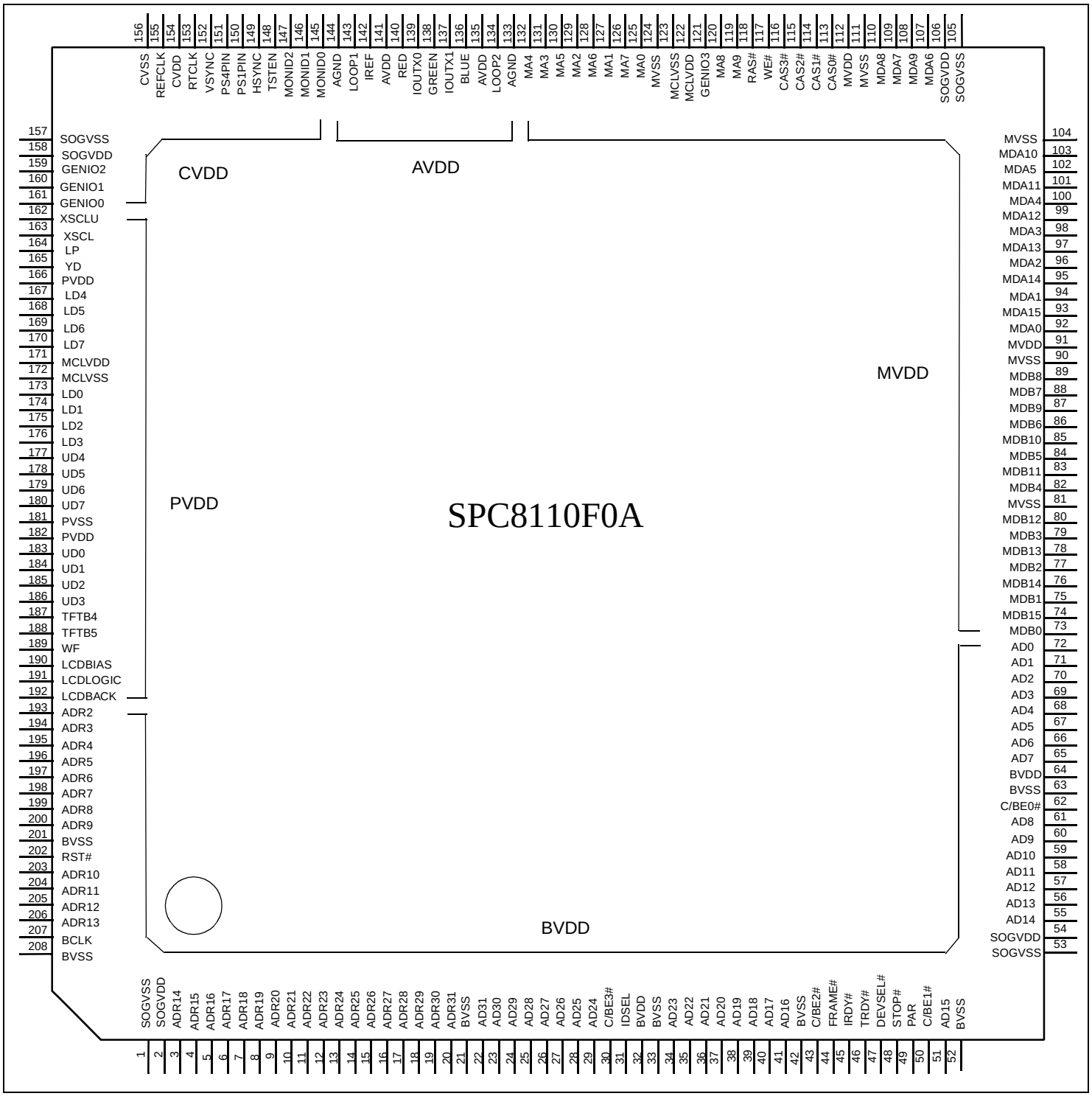
Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{IL}	Low Level Input Voltage	V_{DD} = 4.5V/3.0V	--	--	0.8	V
V_{IH}	High Level Input Voltage	V_{DD} = 5.5V/3.6V	2.0	--	--	V
HR_{PD}	Pull Down Resistance	V_{DD} = 5V	25	50	100	KΩ
LR_{PD}	Pull Down Resistance	V_{DD} = 3.3V	45	90	180	KΩ
V_{T+}	Positive-going Threshold CMOS Schmitt Trigger	V_{DD} = 5/3.3V			2.4	V
V_{T-}	Negative-going Threshold CMOS Schmitt Trigger	V_{DD} = 5/3.3V	0.6			V
HV_H	Hysteresis Voltage CMOS Schmitt Trigger	V_{DD} = 5/3.3V	0.1			V
C_{IN}	Clock Pin Capacitance	BCLK, RESET#, REFCLK, MONID[2:0], TSTEN, PS1PIN,PS4PIN, RTCLK		4		pF
C_{BID}	Input Pin Capacitance	all pins not listed in C_{IN}		10		pF
I_L	Input Leakage Current	V_{DD} = MAX $V_{IH} = V_{DD}$ $V_{IL} = V_{SS}$	-1		1	µA

Output Specifications

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{OL}	Low Level Output Voltage Type 2 Type 3 Type 4	V_{DD} = 5/3.3V I_{OL} = 6mA I_{OL} = 12mA I_{OL} = 24mA			V_{SS} + 0.3	V
V_{OH}	High Level Output Voltage Type 2 Type 3 Type 4	V_{DD} = 5/3.3V I_{OH} = -2mA I_{OH} = -4mA I_{OH} = -8mA	V_{DD}- 0.3			V
I_{OZ}	Off-state Leakage Current	V_{DD} = MAX $V_{IH} = V_{DD}$ $V_{IL} = V_{SS}$	-1		1	µA

S-MOS SYSTEMS
A Seiko Epson Affiliate

■ SPC8110 PIN OUT



Note: Pin names correspond to the PCI bus configuration.
Pin placement subject to change.

■ PIN DESCRIPTION

Key:	A	=	Analog	P	=	Power
	I	=	Input	SCH	=	Schmitt Trigger
	O	=	Output	PD	=	Pull Down
	I/O	=	Bidirectional			

■ CPU Interface

PCI Bus

Pin Name	Type	Output Type	Pin No.	Description
AD[31:0]	I/O	4	22-29, 34-41, 51, 55-61, 65-72	PCI multiplexed Address and Data Bus. These lines are driven by the chip only during read cycles, and are in a hi-Z state at all other times.
ADR[31:2]	I	--	20-3, 206-203, 200-193	Unused VL-Bus Address inputs. These pins should be tied high in PCI mode.
C/BE[3:0]#	I	--	30, 43, 50, 62	PCI Bus Command and Byte Enables.
IDSEL	I	--	31	PCI Bus Initialization Device Select.
STOP#	I/O	4	48	PCI Bus Stop.
FRAME#	I	--	44	PCI Bus Cycle Frame.
IRDY#	I	--	45	PCI Bus Initiator Ready.
PAR	O	4	49	Parity. This line is driven by the chip only during read cycles, and is in a hi-Z state at all other times. It is always hi-Z and should be tied high in VL Bus mode.
BCLK	I	--	207	PCI Bus Clock.
TRDY#	I/O	4	46	PCI Bus Target Ready.
DEVSEL#	O	4	47	PCI Bus Device Select.
RST#	I w/SCH	--	202	CPU Reset. The active low reset signal from the CPU clears all internal registers and forces all signals to their inactive state. On the rising edge of the RESET# the MDA[0...15] bus is latched in for configuration.

Intel486/VL-Bus

Pin Name	Type	Output Type	Pin No.	Description
ADR[31:2]	I	--	20-3, 206-203, 200-193	VL-Bus Address inputs. These inputs are unused and must be tied high in PCI Bus mode.
DAT[31:0]	I/O	4	22-29, 34-41, 51, 55-61, 65-72	VL-Bus Data inputs. These lines are driven by the chip only during read cycles, and are in a hi-Z state at all other times.
BE[3:0]#	I	--	30, 43, 50, 62	VL-Bus byte enables.
W/R#	I	--	31	VL-Bus Write or Read Status.
M/IO#	I	4	48	VL-Bus Memory or I/O Status.
ADS#	I	--	44	VL-Bus Address Data Strobe.
RDYRTN#	I	--	45	VL-Bus Ready Return.
BCLK	I	--	207	VL-Bus Local CPU Clock.
LRDY#	O	4	46	VL-Bus Local Ready.
LDEV#	O	4	47	VL-Bus Local Device.
RESET#	I w/SCH	--	202	CPU Reset. The active low reset signal from the CPU clears all internal registers and forces all signals to their inactive state. On the rising edge of the RESET# the MDA[0...15] bus is latched in for configuration.
PAR	O	4	49	Unused (Parity). This line is used in PCI mode only. It is hi-Z at all times in VL-Bus mode and should be tied high.

■ Video Memory Interface

Pin Name	Type	Output Type	Pin No.	Description
MA[9:0]	O	2	119, 120, 126, 128, 130, 132, 131, 129, 127, 125	Multiplexed row/column address bits for video display memory.
MDA[15:0]	I/O w/ PD	2	93, 95, 97, 99, 101, 103, 108, 110, 109, 107, 102, 100, 98, 96, 94, 92	Data bits for video display memory. The output drivers of these pins are placed into a high-impedance state when RESET# is low, or when MClkOFF is active. These pins are also used as configuration inputs. They have internal pull down resistors that have typical values of 50KΩ/100KΩ at 5V/3.3V respectively.
MDB[15:0]	I/O w/ PD	2	74, 76, 78, 80, 83, 85, 87, 89, 88, 86, 84, 82, 79, 77, 75, 73	Data bits for video display memory when 1024KB of memory is present. The output drivers of these pins are placed into a high-impedance state when RESET# is low, or when MClkOFF is active. The inputs have internal pull down resistors that have typical values of 50K Ω/100K Ω at 5 V/3.3 V respectively. They should be left open-circuit when only 512KB of memory is attached.
RAS#	O	2	118	DRAM Row Address Strobe.
CAS[0]# (CAS#)	O	2	113	DRAM Column Address Strobe for MDA[7:0], or Column Address Strobe for all bytes, as configured by an auxiliary register.
CAS[3:1]# (WE[3:1]#)	O	2	116-114	DRAM Column Address Strobes or Write Enable Strobes for {MDB[15:0],MDA[15:8]}, as configured by an auxiliary register. CAS[3:2]# (WE[3:2]#) are unused and should be left open-circuit when only 512KB of memory is present.
WE# (WE[0]#)	O	2	117	DRAM Write Enable Strobe for all bytes or Write Enable Strobe for MDA[7:0], as configured by an auxiliary register.

■ Clock Inputs

Pin Name	Type	Output Type	Pin No.	Description
REFCLK	I	--	155	This pin is the reference clock used by the internal PLLs to generate all the necessary clocks. This must be stable during full operation. REFCLK may shut down only after the chip is totally powered down. Input frequency is typically 14.318 MHz.
RTCLK	I	--	153	Real Time Clock. This input must be used to provide a low frequency clock for generating DRAM refresh. This clock must be approximately 32 kHz and 50% duty cycle.

■ CRT Interface

Pin Name	Type	Output Type	Pin No.	Description
RED, GREEN, BLUE	AO	--	140, 138, 136	Analog outputs from the Video DAC. Internal comparator for monitor sense is available on all three pins.
HSYNC	O	3	149	Horizontal Sync. This output is driven to indicate the horizontal retrace period. In CRT only mode, the polarity of this signal is controlled by register 3C2h, bit 6. In TFT or double scan mode this bit is active low to indicate 640 x 480 resolution. This pin is held low in LCD modes. This pin follows the DPMS standard during power down modes.
VSYNC	O	3	152	Vertical Sync. This output is driven to indicate the vertical retrace period. In CRT only mode, the polarity of this signal is controlled by register 3C3h, bit 7. In TFT or double scan mode this bit is active low to indicate 640 x 480 resolution. This pin is held low in LCD modes. This pin follows the DPMS standard during power down modes.
IREF	AI	--	142	3.3 mA current reference input for the Video DAC.
MONID[2:0]	I	--	147-145	Monitor ID bits. Connected to the CRT to identify the monitor type.

■ LCD Panel Interface

Pin Name	FPDI-1™ PIN Name[1]	Type	Output Type	Pin No.	Description
YD	FPFRAME	O	4	165	Vertical Scanning Start Pulse output. A logic 1 on this signal, sampled by the LCD module on the falling edge of LP, is used by the panel row drivers (Y drivers) to indicate the start of the vertical frame.
LP	FPLINE	O	4	164	Latch Pulse output. The falling edge of this signal is used to latch a row of display data in the LCD module's column driver shift registers and to turn on the row driver (Y driver) for that line.
XSCL	FPSHIFT	O	4	163	Shift Clock for LCD data. Display data is clocked out of the chip on the rising edge of this signal, to be shifted into the LCD panel module column drivers (X drivers) on each falling edge.
XSCLU	FPSHIFT2	O	4	162	Second Shift Clock for some color LCD displays.
UD[7:0] LD[7:0]	UD[7:0] LD[7:0]	I/O	4	180-177, 186-183, 170-167, 176-173	Panel display data bus. The data format depends on the specific panel connected. These pins are driven low when the LCD interface is disabled (e.g. CRT only mode). For 8-bit panels, UD[7:4] and LD[7:4] are driven low.
TFTB[5:4]	-	O	4	188-187	TFT display data bus for the two lsb of the color blue. For non-TFT panels, these two pins are driven low.
LCDBIAS	-	O	2	190	LCD power control for the LCD bias circuitry. Active (On) polarity is defined by the state of MDA[3] on the rising edge of RESET#.
LCDBACK	-	O	2	192	LCD power control for the LCD back light. Active (On) polarity is defined by the state of MDA[1] on the rising edge of RESET#.
LCDLOGIC	-	O	2	191	LCD power control for the LCD logic circuitry. Active (On) polarity is defined by the state of MDA[2] on the rising edge of RESET#.
WF	MOD	O	4	189	LCD Backplane Bias signal. Output toggling frequency is programmable in an auxiliary register. For TFT panels, this pin outputs the display enable signals.

1. VESA Flat Panel Display Interface Standard (FPDI-1™)

■ Miscellaneous

Pin Name	Type	Output Type	Pin No.	Description
TSTEN	I w/ PD	--	148	This pin, when high, sets the SPC8110F0A into either boundary pin SCAN or pin output drive test, depending on the state of PS1PIN. This input has an internal pull down resistor that has a typical value of 50K/100K Ω at 5 V/3.3 V respectively.
GENIO[3:0]	I/O w/ PD	2	121, 159, 160 161	General purpose I/O pins. Output state of each pin is programmable to control external devices.
LOOP1	I	--	143	Connects to Loop Filter Resister and Capacitor for PLL1, where suggested values are R=70 Ω and C=0.1 uf.
LOOP2	I	--	134	Connects to Loop Filter Resister and Capacitor for PLL2, where suggested values are R=70 Ω and C=0.1 uf.
IOUTX[1:0]	A	--	137, 139	Balanced current output for the DAC.

■ Power Save Mode Controls

Pin Name	Type	Output Type	Pin No.	Description
PS1PIN	I w/ SCH, PD	--	150	Pin used to initiate a power save mode 1. The polarity of this pin is programmable and has a default polarity of active low. This input has an internal pull down resistor that has a typical value of 50K/100K Ω at 5 V/3.3 V respectively. When TSTEN is active, the polarity of PS1PIN selects the test to be either boundary pin SCAN (PS1PIN = 1) or pin output drive test (PS1PIN = 0).
PS4PIN	I w/ SCH, PD	--	151	Pin used to initiate a power save mode 4. The polarity of this pin is programmable and has a default polarity of active low. This input has an internal pull down resistor that has a typical value of 50K/100K Ω at 5 V/3.3 V respectively.

■ Power Supply

Pin Name	Type	Pin No.	Description
SOGVDD	P	158, 106, 54, 2	V_{DD} supply for core logic. (3.3V only.)
MVDD	P	112, 91	V_{DD} supply for memory pins. (Either 3.3V or 5.0V.)
BVDD	P	32, 64	V_{DD} supply for bus interface pins. (Either 3.3V or 5.0V.)
CVDD	P	154	V_{DD} supply for control interface pins. (Either 3.3V or 5.0V.)
PVDD	P	166, 182	V_{DD} supply for panel interface pins. (Either 3.3V or 5.0V.)
AVDD	P	135, 141	Analog power supply. (3.3V only.)
MCLVDD	P	122, 171	VDD supply for MCL logic. (3.3V only.)
SOGVSS	P	157, 105, 53, 1	VSS supply for core logic.
MVSS	P	111, 124, 104, 90, 81	VSS supply for memory pins.
BVSS	P	201, 208, 21, 42, 52, 33, 63	VSS supply for bus interface pins.
CVSS	P	156	VSS supply for CRT interface pins.
PVSS	P	181	VSS supply for panel interface pins.
AGND	P	133, 144	Analog Ground.

Pin Name	Type	Pin No.	Description
MCLVSS	P	123, 172	VSS supply for MCL logic.

■ Intel486/VL-Bus to PCI Bus Pin Mapping

VL-Bus Name	PCI Name	Pin No.
RESET#	RST#	202
RDYRTN#	IRDY#	45
ADR[31:2]	--	20-3, 206-203, 200-193
W/R#	IDSEL	31
LDEV#	DEVSEL#	47
DAT[31:0]	AD[31:0]	22-29, 34-41, 51, 55-61, 65-72

VL-Bus Name	PCI Name	Pin No.
BCLK	BCLK	207
--	PAR	49
M/IO#	STOP#	48
ADS#	FRAME#	44
LRDY#	TRDY#	46
BE[3:0]#	C/BE[3:0]#	30, 43, 50, 62

■ Mixed Voltage Operation

Mixed Voltage	SOGVDD, MCLVDD, AVDD	BVDD	CVDD	PVDD	MVDD
Configuration 1	3.3 V	3.3 V	3.3 V	3.3 V	3.3 V
Configuration 2					5.0 V
Configuration 3				5.0 V	3.3 V
Configuration 4					5.0 V
Configuration 5			5.0 V	3.3 V	3.3 V
Configuration 6					5.0 V
Configuration 7				5.0 V	3.3 V
Configuration 8					5.0 V
Configuration 9		5.0 V	3.3 V	3.3 V	3.3 V
Configuration 10					5.0 V
Configuration 11				5.0 V	3.3 V
Configuration 12					5.0 V
Configuration 13			5.0 V	3.3 V	3.3 V
Configuration 14					5.0 V
Configuration 15				5.0 V	3.3 V
Configuration 16					5.0 V

■ Default Pin Mapping for the Various LCD and TFT Panel Options

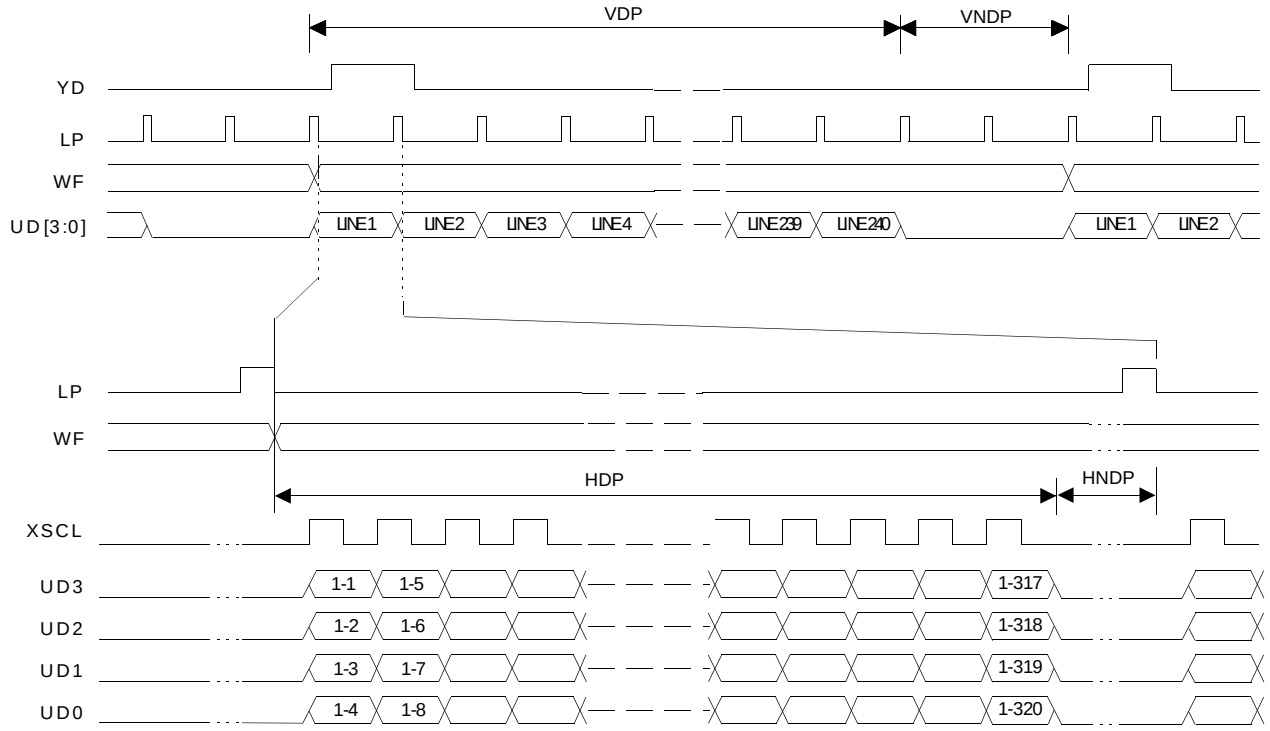
Pin Name	TFT				Color LCD				Mono LCD	
	18-Bit Color	9-Bit x 2 Color	12-Bit Color	9-Bit Color	16-Bit	8-Bit Half Panel	8-Bit	6-Bit	8-Bit	4-Bit
XSCLU	CLK	CLK	CLK	CLK			XSCLU			
XSCL					XSCL	XSCL	XSCL	XSCL	XSCL	XSCL
LP	HS	HS	HS	HS	LP	LP	LP	LP	LP	LP
YD	VS	VS	VS	VS	YD	YD	YD	YD	YD	YD
LD4	R0	R10			LD4				UD0	UD0
LD5	R1	R11			LD5				UD1	UD1
LD6	R2	R12	R0		LD6				UD2	UD2
LD7	R3	R00	R1	R0	LD7				UD3	UD3
LD0	R4	R01	R2	R1	LD0		LD0	LB	LD0	
LD1	R5	R02	R3	R2	LD1		LD1	LG	LD1	
LD2	G0	G10			LD2		LD2	LR	LD2	
LD3	G1	G11			LD3		LD3		LD3	
UD4	G2	G12	G0		UD4	UD4				
UD5	G3	G00	G1	G0	UD5	UD5				
UD6	G4	G01	G2	G1	UD6	UD6				
UD7	G5	G02	G3	G2	UD7	UD7				
UD0	B0	B10			UD0	UD0	UD0	UB		
UD1	B1	B11			UD1	UD1	UD1	UG		
UD2	B2	B12	B0		UD2	UD2	UD2	UR		
UD3	B3	B00	B1	B0	UD3	UD3	UD3			
TFTB4	B4	B01	B2	B1						
TFTB5	B5	B02	B3	B2						
WF	DE	DE	DE	DE	WF	WF	WF	WF/DE	WF	WF
LCDBIAS					(DISP)	(DISP)	(DISP)	(DISP)	(DISP)	(DISP)

■ Alternate Pin Mapping for the Various LCD and TFT Panel Options

Pin Name	TFT				Color LCD			Mono LCD	
	18-Bit Color	9-Bit x 2 Color	12-Bit Color	9-Bit Color	16-Bit	8-Bit Half Panel	8-Bit	8-Bit Single	8-Bit Dual
XSCLU	CLK	CLK	CLK	CLK			XSCLU		
XSCL					XSCL	XSCL	XSCL	XSCL	XSCL
LP	HS	HS	HS	HS	LP	LP	LP	LP	LP
YD	VS	VS	VS	VS	YD	YD	YD	YD	YD
LD4	R0	R10							
LD5	R1	R11							
LD6	R2	R12	R0		UD0	UD0	UD0		
LD7	R3	R00	R1	R0	UD1	UD1	UD1	D6	UD2
LD0	R4	R01	R2	R1	UD2	UD2	UD2	D7	UD3
LD1	R5	R02	R3	R2	UD3	UD3	UD3		
LD2	G0	G10			UD4	UD4			
LD3	G1	G11			UD5	UD5			
UD4	G2	G12	G0		LD4				
UD5	G3	G00	G1	G0	LD5			D3	LD3
UD6	G4	G01	G2	G1	LD6			D4	UD0
UD7	G5	G02	G3	G2	LD7			D5	UD1
UD0	B0	B10			UD6	UD6			
UD1	B1	B11			UD7	UD7			
UD2	B2	B12	B0		LD0		LD0		
UD3	B3	B00	B1	B0	LD1		LD1	D0	LD0
TFTB4	B4	B01	B2	B1	LD2		LD2	D1	LD1
TFTB5	B5	B02	B3	B2	LD3		LD3	D2	LD2
WF	DE	DE	DE	DE	WF	WF	WF	WF	WF
LCDBIAS					(DISP)	(DISP)	(DISP)	(DISP)	(DISP)

■ Monochrome Passive STN LCD Panel Interface

4-Bit Single Panel



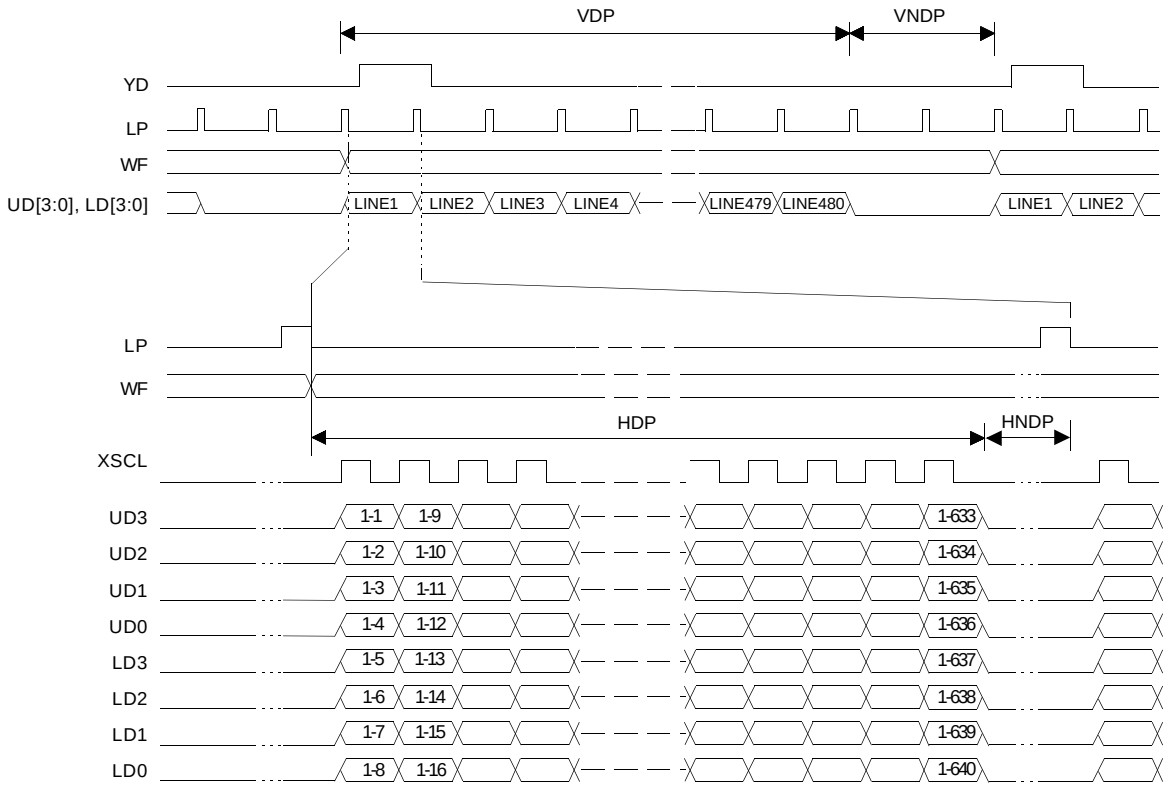
* diagram drawn with 2 LP vertical blank period

Example timing for a 320 x 240 panel

VDP = Vertical Display Period
VNDP = Vertical Non-Display Period
HDP = Horizontal Display Period
HNDP = Horizontal Non-Display Period

■ Monochrome Passive STN LCD Panel Interface

8-Bit Single Panel

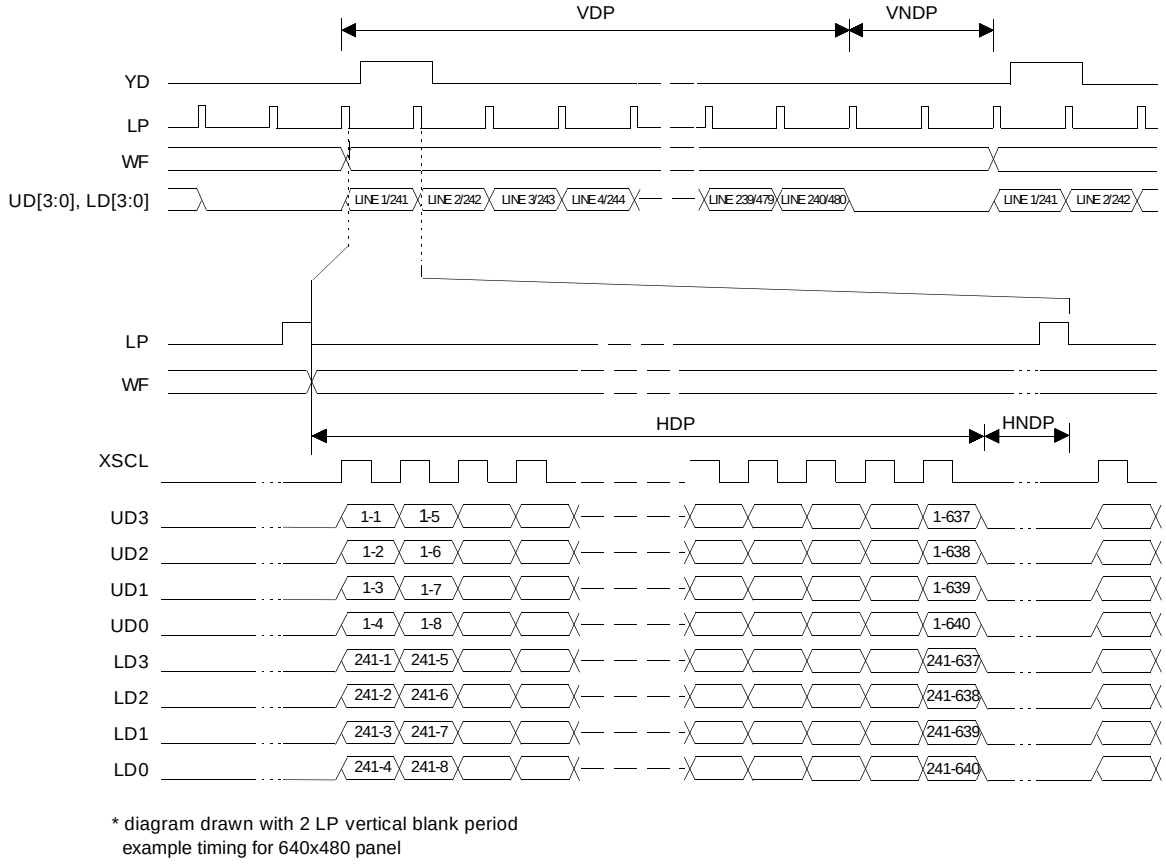


* diagram drawn with 2 LP vertical blank period
example timing for 640x480 panel

VDP = Vertical Display Period
VNDP = Vertical Non-Display Period
HDP = Horizontal Display Period
HNDP = Horizontal Non-Display Period

■ Monochrome Passive STN LCD Panel Interface

8-Bit Dual Panel

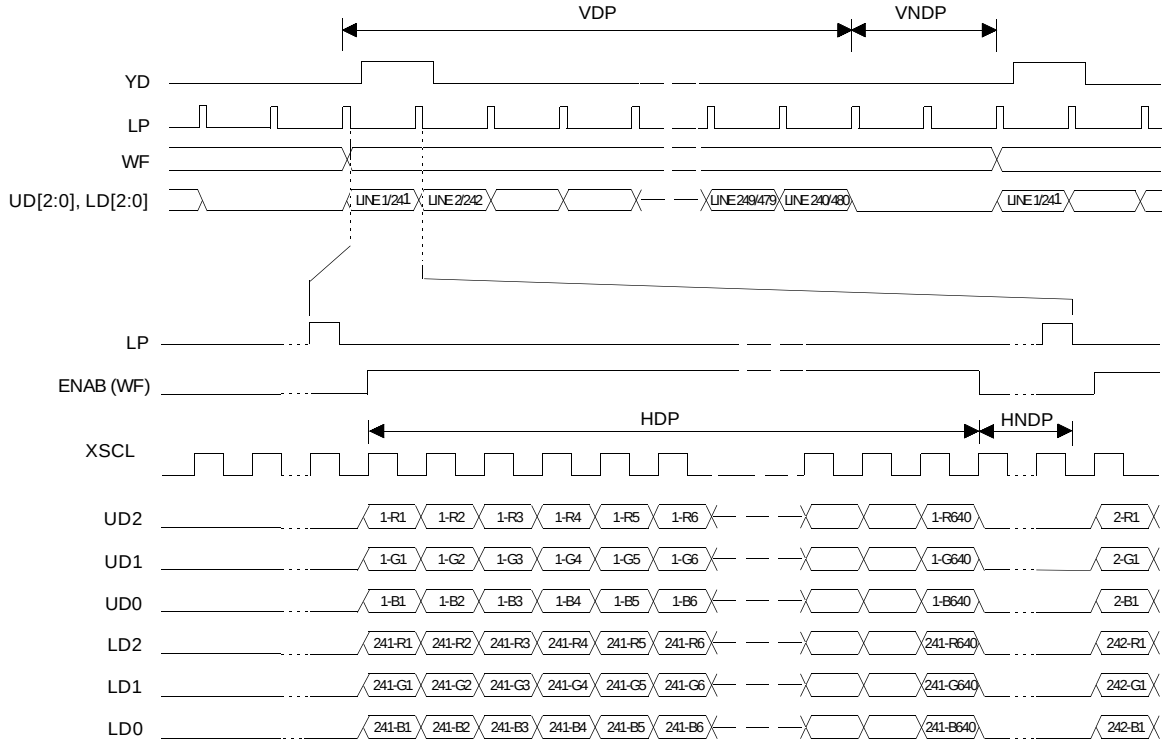


* diagram drawn with 2 LP vertical blank period
example timing for 640x480 panel

VDP = Vertical Display Period
VNDP = Vertical Non-Display Period
HDP = Horizontal Display Period
HNDP = Horizontal Non-Display Period

■ Color Passive STN LCD Panel Interface

6-Bit Dual Color Panel

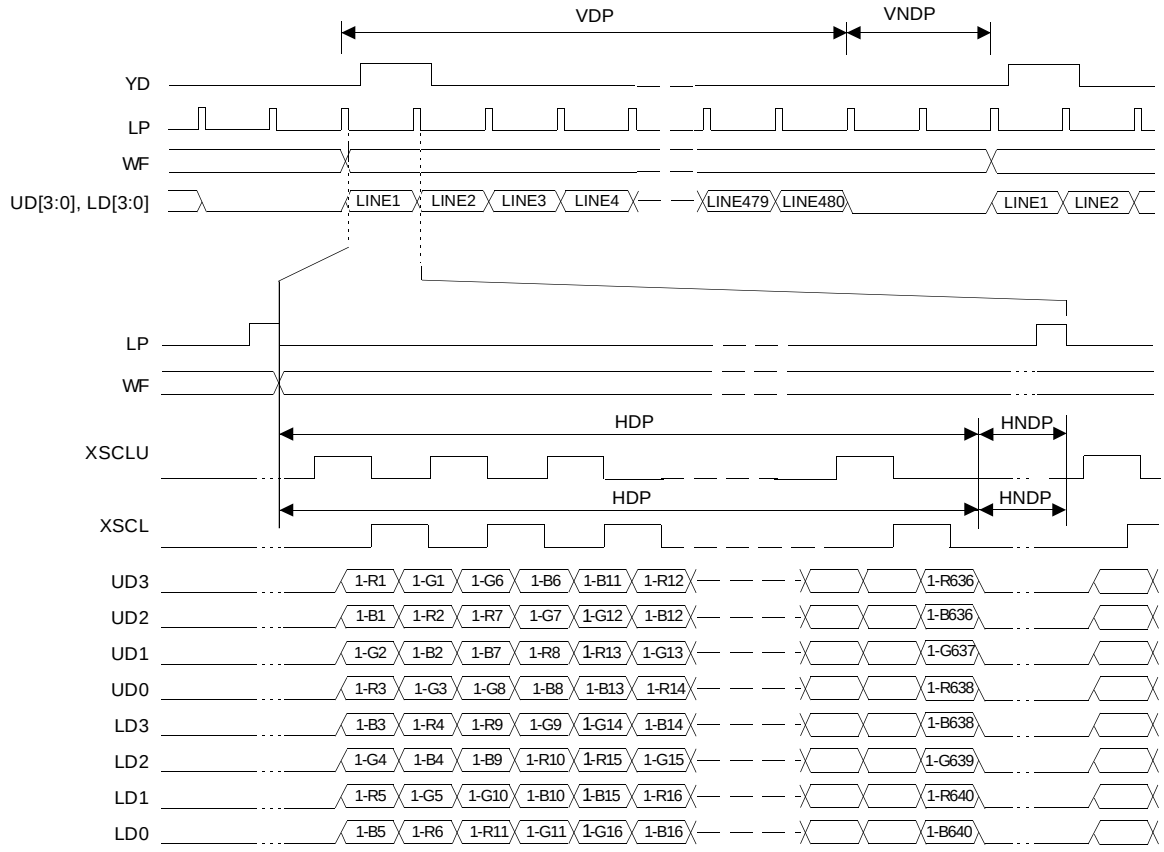


* diagram drawn with 2 LP vertical blank period
example timing for 640x480 panel

VDP = Vertical Display Period
VNDP = Vertical Non-Display Period
HDP = Horizontal Display Period
HNDP = Horizontal Non-Display Period

■ Color Passive STN LCD Panel Interface

8-Bit Single Color Panel

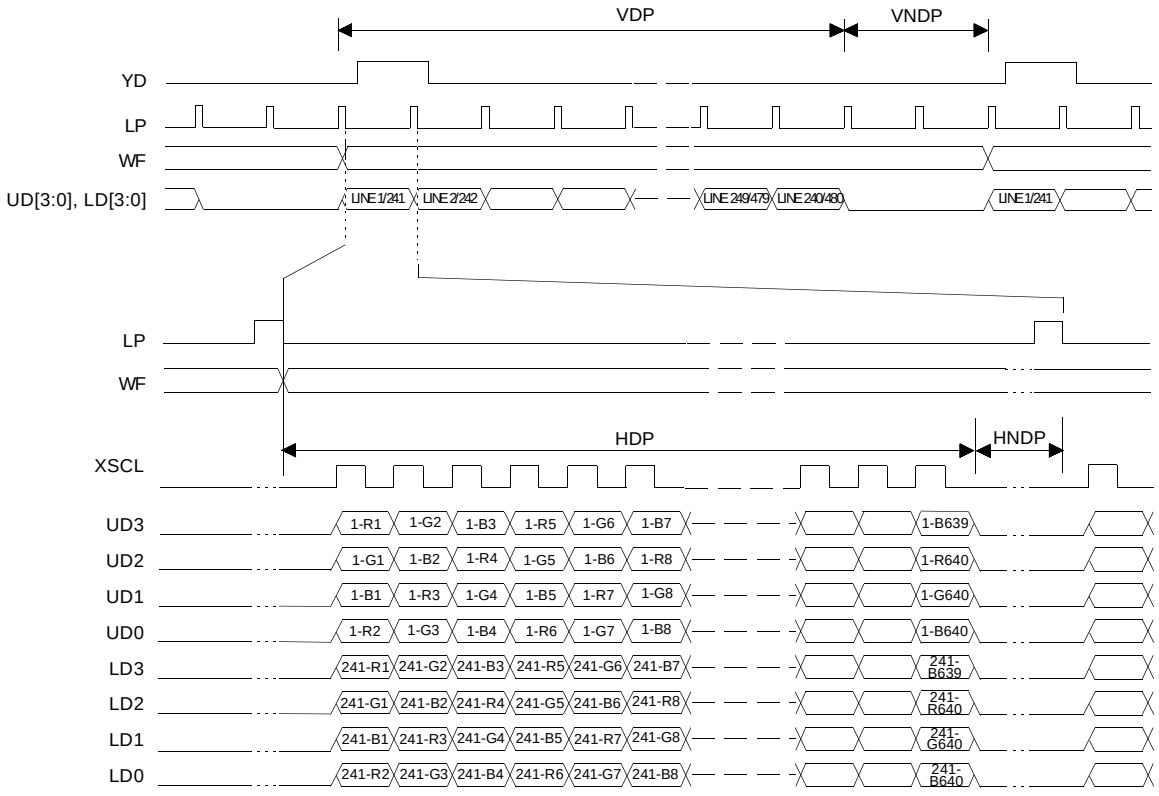


* diagram drawn with 2 LP vertical blank period
example timing for 640x480 panel

VDP = Vertical Display Period
VNDP = Vertical Non-Display Period
HDP = Horizontal Display Period
HND = Horizontal Non-Display Period

■ Color Passive STN LCD Panel Interface

8-Bit Dual Color Panel (One Shift Clock)

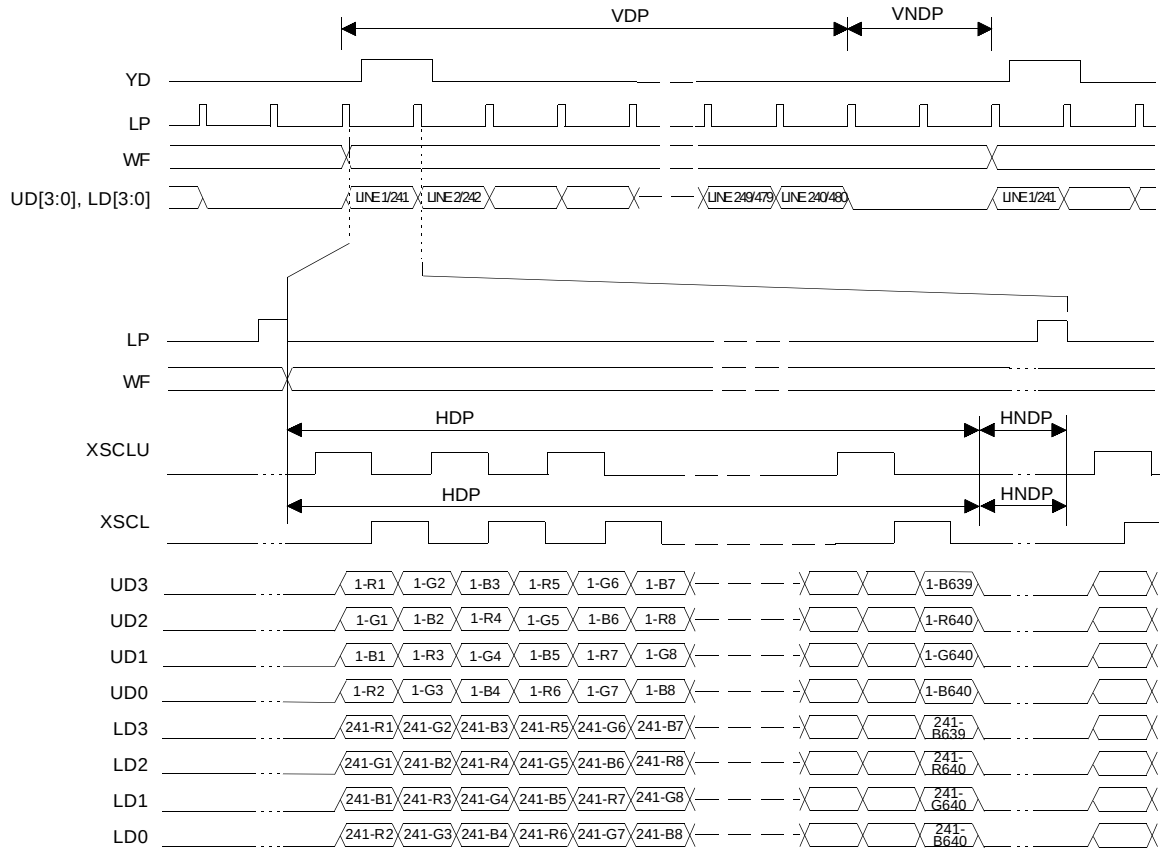


* diagram drawn with 2 LP vertical blank period
example timing for 640x480 panel

VDP = Vertical Display Period
VNDP = Vertical Non-Display Period
HD = Horizontal Display Period
HNDP = Horizontal Non-Display Period

■ Color Passive STN LCD Panel Interface

8-Bit Dual Color Panel (Two Shift Clocks)

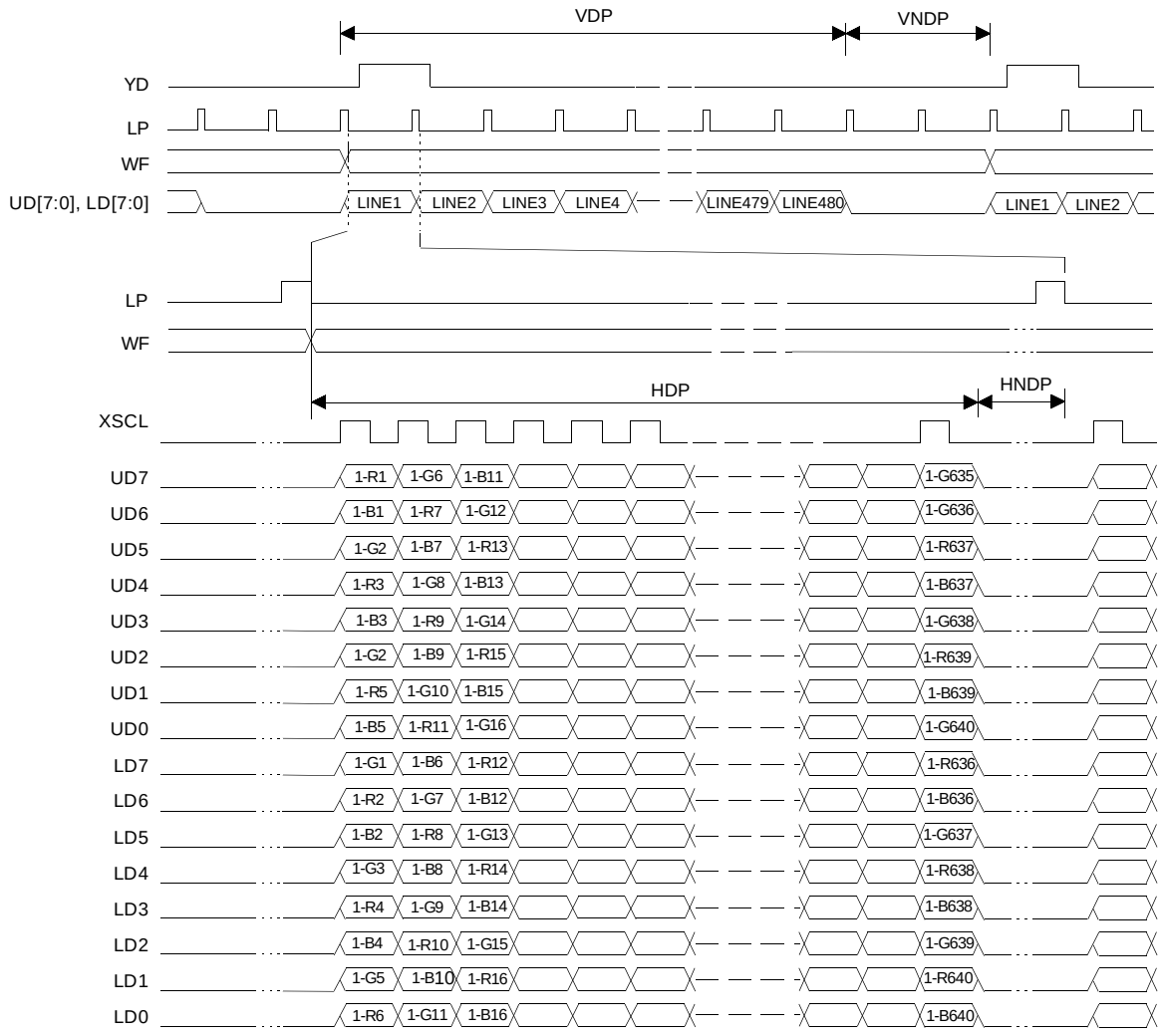


* diagram drawn with 2 LP vertical blank period
example timing for 640x480 panel

VDP = Vertical Display Period
VND = Vertical Non-Display Period
HDP = Horizontal Display Period
HNDP = Horizontal Non-Display Period

■ Color Passive STN LCD Panel Interface

16-Bit Single Color Panel

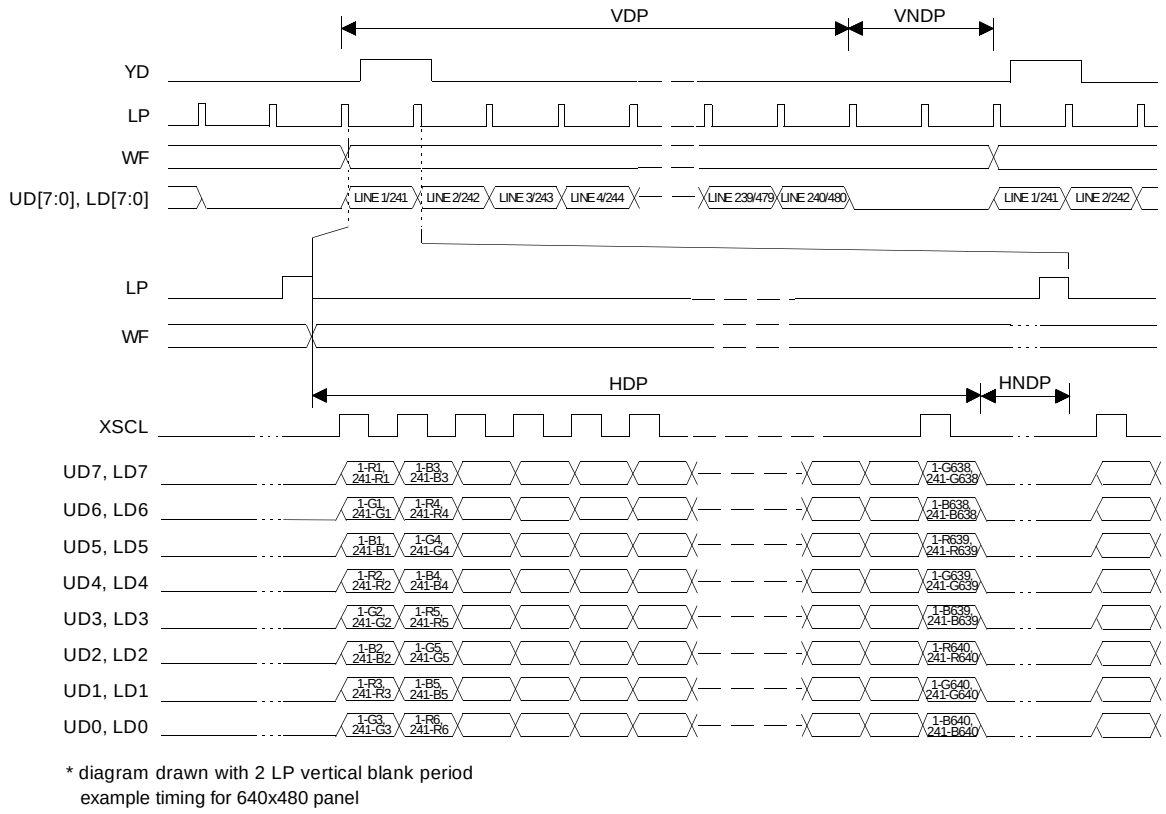


* diagram drawn with 2 LP vertical blank period
example timing for 640x480 panel

VDP = Vertical Display Period
VNDP = Vertical Non-Display Period
HDP = Horizontal Display Period
HNDP = Horizontal Non-Display Period

■ Color Passive STN LCD Panel Interface

16-Bit Dual Color Panel

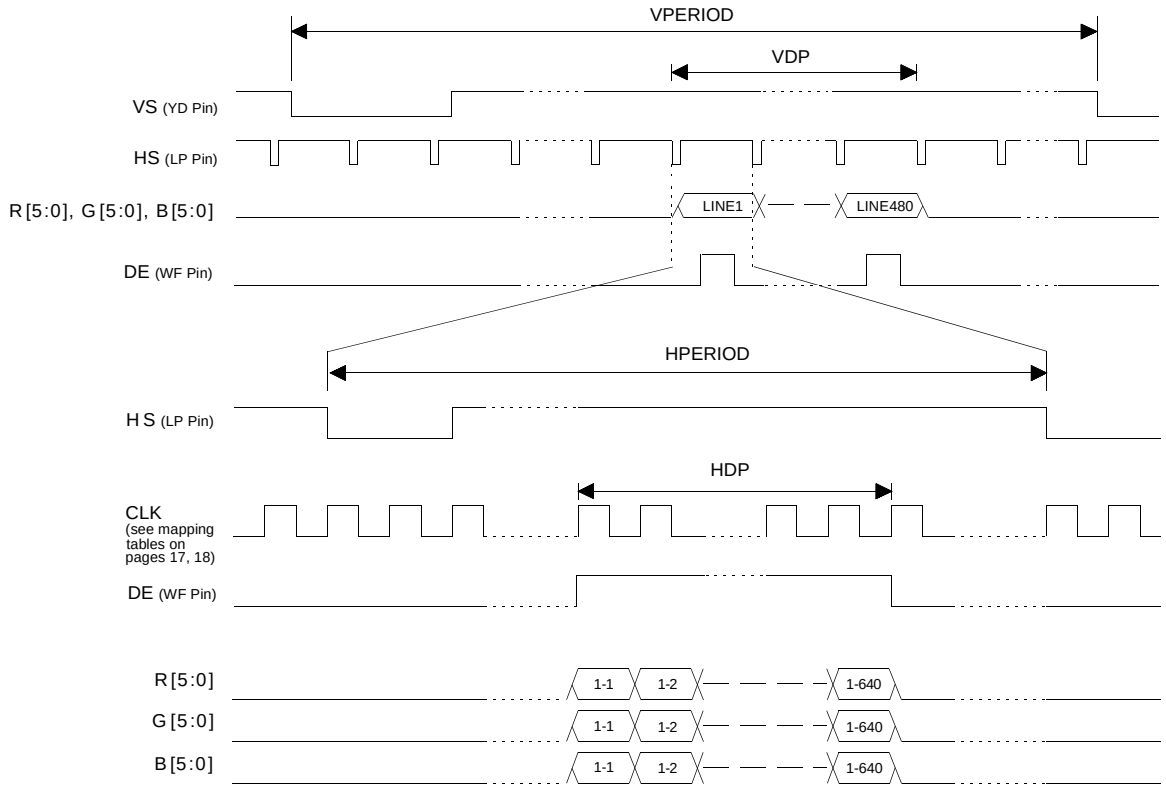


* diagram drawn with 2 LP vertical blank period
example timing for 640x480 panel

VDP = Vertical Display Period
VNDP = Vertical Non-Display Period
HDP = Horizontal Display Period
HNDP = Horizontal Non-Display Period

■ Active Matrix LCD Panel Interface

18-Bit Color TFT

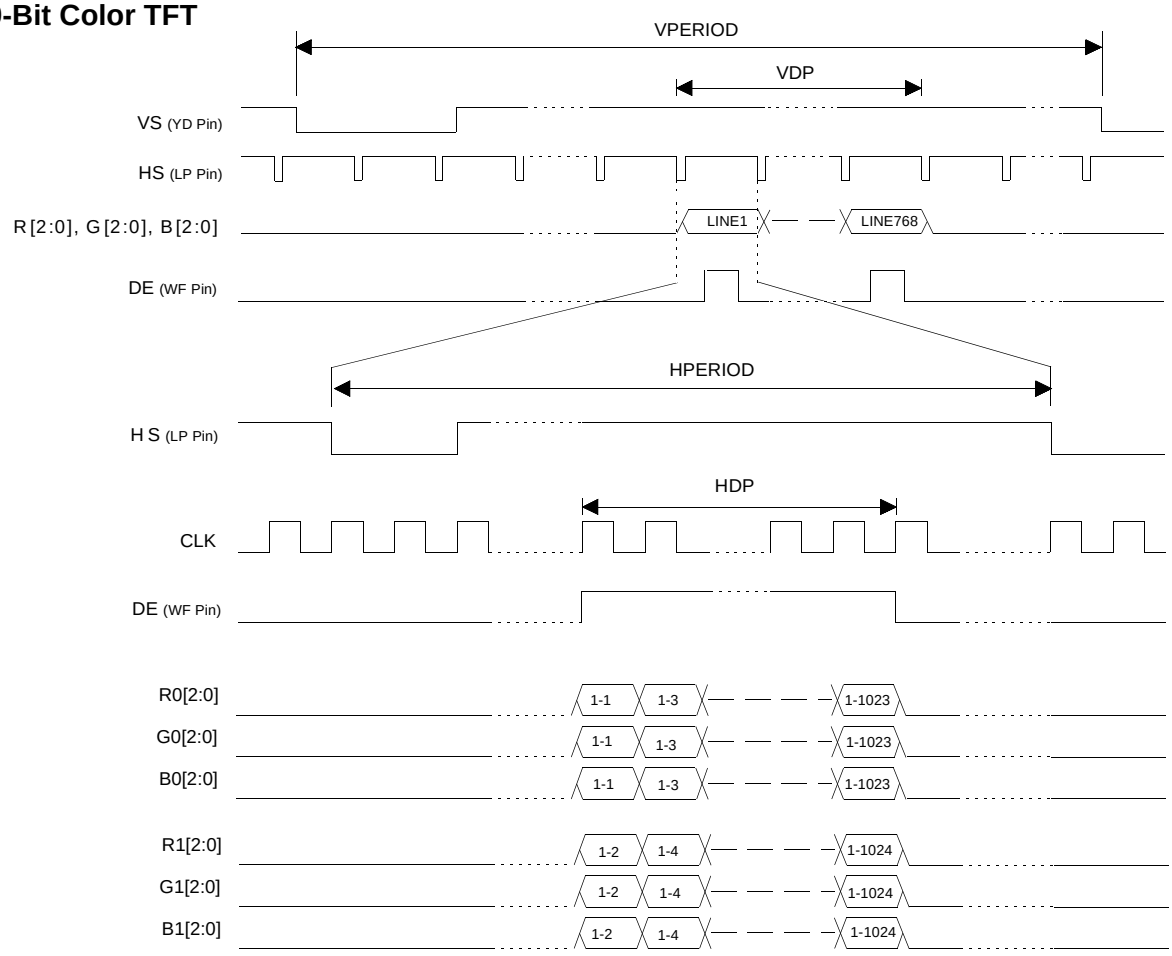


Note: DE (Data Enable) is used to indicate the first pixel
Example Timing for 640 x 480 panel

VPERIOD = Vertical Display Period
VDP = Vertical Display Period
HPERIOD = Horizontal Period
HDP = Horizontal Display Period

■ Active Matrix LCD Panel Interface

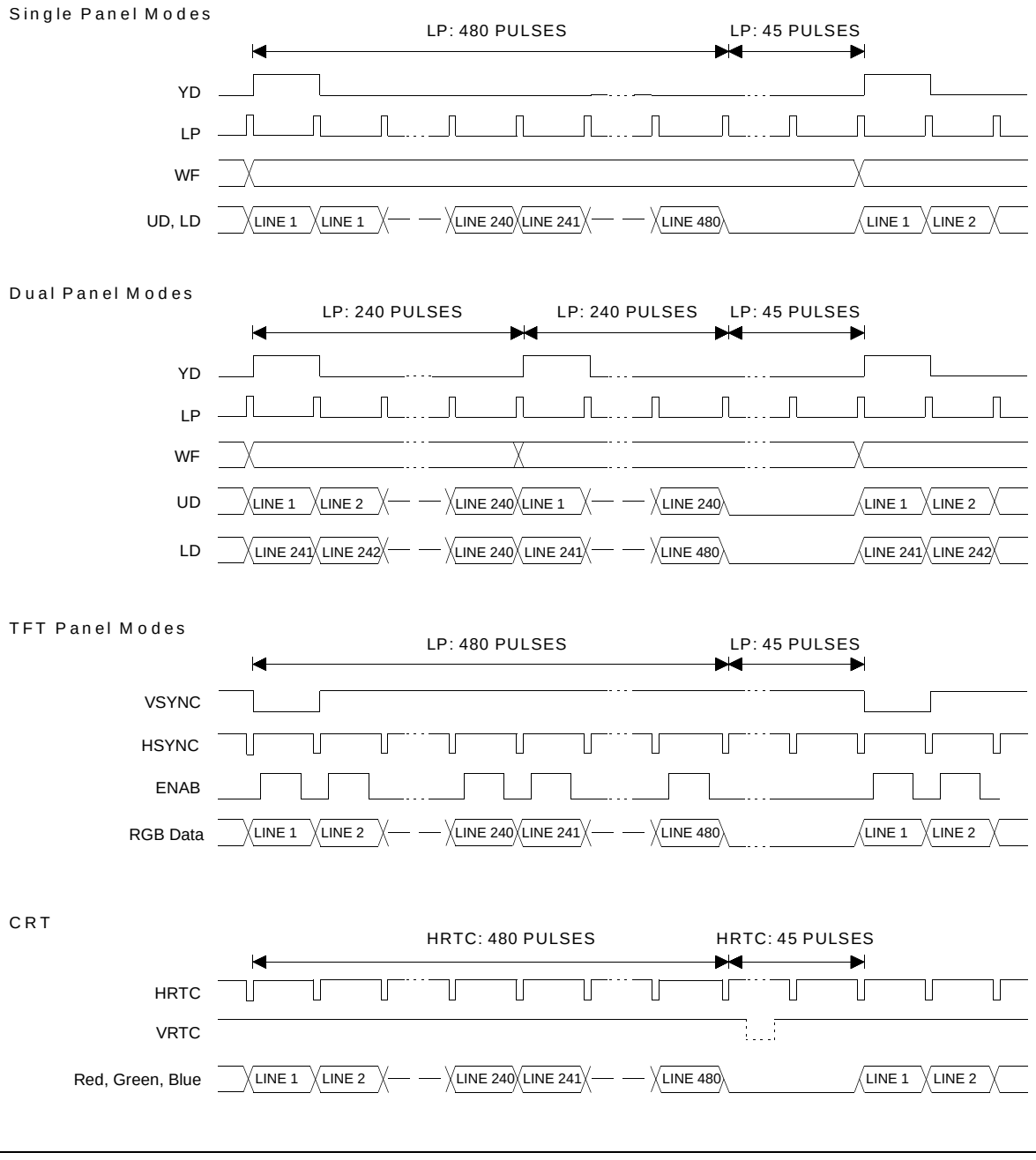
2 X 9-Bit Color TFT



Note: DE (Data Enable) is used to indicate the first pixel
example timing for 1024x768 panel

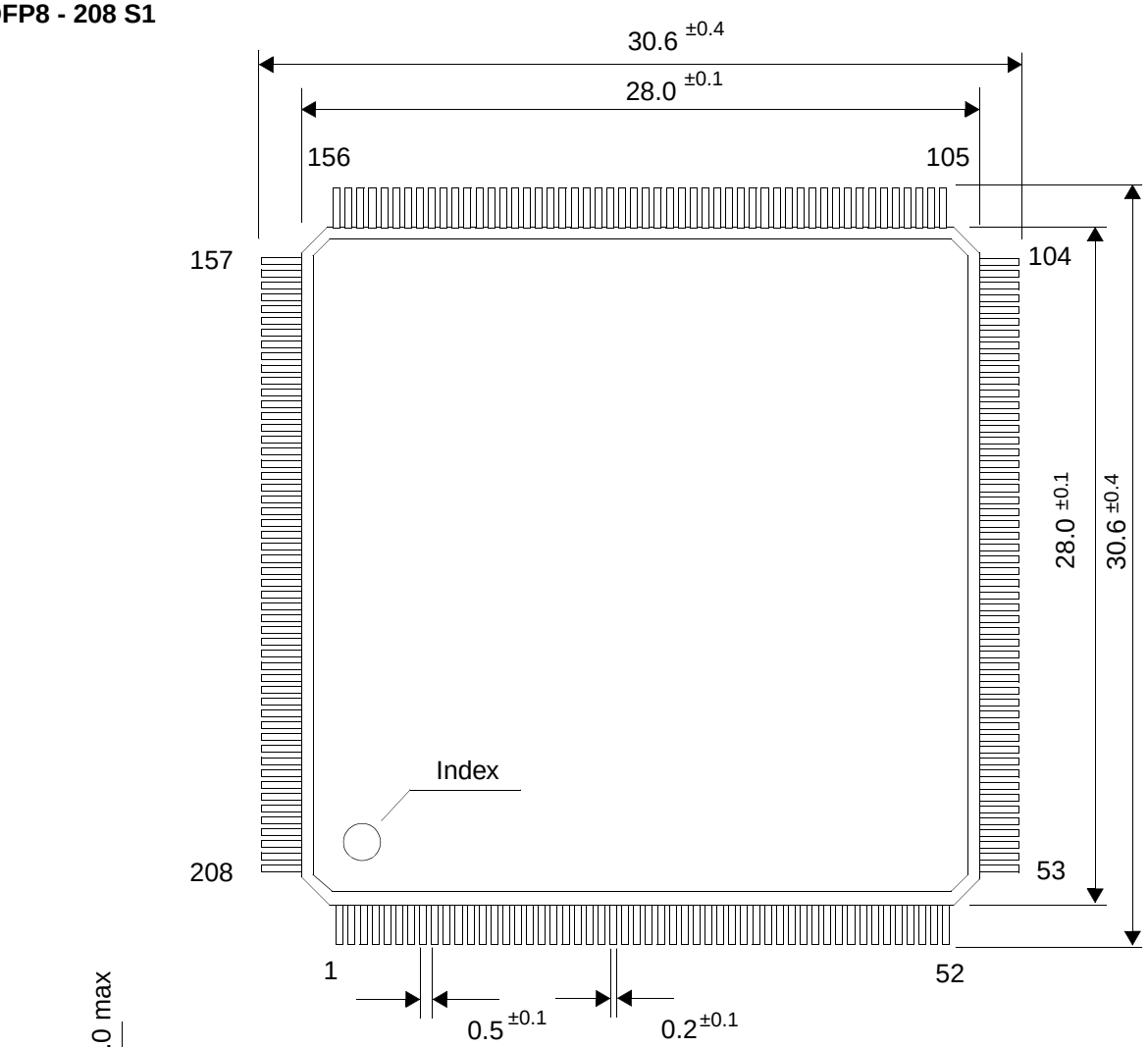
VPERIOD = Vertical Display Period
VDP = Vertical Display Period
HPERIOD = Horizontal Period
HDP = Horizontal Display Period

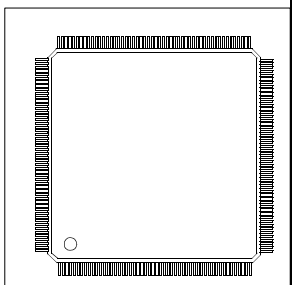
■ Simultaneous Display Interface



■ PACKAGE DIMENSIONS

QFP8 - 208 S1 **Unit: mm**





Actual Size

■ Comprehensive Support Tools

S-MOS Systems provides to the designer and manufacturer a complete set of resources and tools for the development of LCD Graphics Systems:

Evaluation/Demonstration Board

- Assembled and fully tested graphics evaluation board
- Schematic of evaluation/demo board
- Parts list
- Installation guide
- OEM utilities
- User utilities
- Evaluation software

Documentation

- Technical manuals
- Evaluation/demo board manual

Application Engineering Support

In order to help customers implement the LCD/ CRT Controller in their design, S-MOS offers training, technical support and the following services:

- Sales technical support
- Customer training
- Design assistance

CONTACT YOUR LOCAL SALES REPRESENTATIVE FOR THESE COMPREHENSIVE DESIGN TOOLS:

- SPC8110F0A TECHNICAL MANUAL
- SDU8110B0C VL BUS EVALUATION BOARD
- SDU8110B0E PCI BUS EVALUATION BOARD

Head Office
S-MOS Systems Inc.
150 River Oaks Parkway
San Jose, CA 95134
Tel: (408) 922-0200
Fax: (408) 922-0238
http://www.smos.com

North Central USA
1450 East American Lane
Suite 1550
Schaumburg, Illinois 60173
Tel: (847) 517-7667
Fax: (847) 517-7601
Cell: (847) 558-6662
Pager: (888) 789-3471

Eastern USA
301 Edgewater Place
Suite 120
Wakefield, MA 01880
Tel: (781) 246-3600
Fax: (781) 246-5443

Southeast USA
4300 Six Forks Rd.
Suite 430
Raleigh, NC 27609
Tel: (919) 781-7667
Fax: (919) 781-6778

S-MOS assumes no responsibility or liability for (1) any errors or inaccuracies contained in the information herein and (2) the use of the information or a portion thereof in any application, including any claim for (a) copyright or patent infringement or (b) direct, indirect, special or consequential damages. There are no warranties extended or granted by this document. The information herein is subject to change without notice from S-MOS.

January 1998 © Copyright 1996, 1998 S-MOS Systems, Inc. Printed in U.S.A.

S-MOS
SYSTEMS
A Seiko Epson Affiliate

SPC8110F0A LOCAL BUS LCD/CRT VGA CONTROLLER

Hardware Functional Specification

Drawing Office No. X07-SP-001-18

Copyright © 1994, 1997 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

Table of Contents

1 INTRODUCTION 11
1.1 Scope 11
2 FEATURES 12
2.1 General 12
2.2 Technology 12
2.3 System 12
2.4 Compatibility 13
2.5 Display Support 13
2.6 Display Modes 14
3 OVERVIEW DESCRIPTION 15
3.1 Typical System Block Diagrams 15
3.2 Functional Block Diagram 17
3.3 Functional Block Description 18
3.3.1 Host Interface 18
3.3.2 Clock Generator 18
3.3.3 VGA Core 18
3.3.4 Hardware Cursor 18
3.3.5 BitBLT Engine 18
3.3.6 Display FIFO 18
3.3.7 VGA Palette 18
3.3.8 DAC 18
3.3.9 LCD Interface 18
3.3.10 Power Save Logic 18
4 PIN OUT DIAGRAM 19
5 PIN DESCRIPTIONS 20
5.1 Summary of Configuration Options 25
5.2 SPC8110F0A Pin Mapping 25
6 D.C. CHARACTERISTICS 28
7 A.C. CHARACTERISTICS 30
7.1 VL-Bus/Intel 486 DX Cycle Timing 30
7.2 VL-Bus/Intel 486 DX Input/Output Timing 30
7.3 PCI Bus Cycle Timing 32
7.4 PCI Local Bus Disconnect Timing 32
7.5 PCI Local Bus Input/Output Timing 33
7.6 DRAM A.C. Timing 34
7.6.1 Read/Write Timing 34
7.6.2 CAS Before RAS Refresh Timing 35
7.6.3 Self Refresh Timing 36
7.6.4 Power Down Mode CAS Before RAS Refresh Timing 36

7.7 Clock Input Requirements 37
7.8 LCD Panel Timing Diagrams 38
7.8.1 Single Monochrome 4-bit Panel Functional Timing 38
7.8.2 Single Monochrome 8-bit Panel Timing 40
7.8.3 Single Color 8-bit Panel Timing 42
7.8.4 Single Color 8-bit Panel Timing (Alternate Data Format) 44
7.8.5 Single Color 16-bit Panel Timing 46
7.8.6 Dual Monochrome 8-bit Panel Timing 48
7.8.7 Dual Color 6-bit Panel Timing 50
7.8.8 Dual Color 8-bit Panel Timing (One Shift Clock) 52
7.8.9 Dual Color 8-bit Panel Timing (Two Shift Clocks) 54
7.8.10 Dual Color 16-bit Panel Timing 56
7.8.11 TFT Panel Timing 58

8 REGISTER DESCRIPTIONS 62

8.1 Software Considerations 62
8.1.1 Background 62
8.1.2 Unsupported VGA Registers 62
8.2 VGA I/O Register Summary 63
8.2.1 General Registers 63
8.2.2 Palette Registers 64
8.2.3 Sequencer Registers 65
8.2.4 Graphics Controller Registers 67
8.2.5 Attributes Controller Registers 68
8.2.6 CRT Controller Registers - Standard Set (A-Set) 71
8.2.7 CRTC Registers - Extended Set: Sprite/HW Cursor Controls 77
8.2.8 B-Set Registers 81
8.3 Page Register 83
8.4 Auxiliary Registers 83
8.4.1 General Auxiliary Registers 84
8.4.2 Power Save Registers 96
8.4.3 Frame and Video Buffer Registers 105
8.4.4 PLL Control Registers 107
8.4.5 General I/O Registers 110
8.4.6 FIFO Level Registers 110
8.4.7 Auxiliary Enable Register 112
8.5 BitBLT Registers 112
8.5.1 BitBLT Data Register (Memory Mapped) 112
8.5.2 BitBLT Control Registers 112
8.6 PCI Configuration Registers 116

9 ARCHITECTURE DESCRIPTION 119
9.1 PLL 119
9.1.1 Functional Overview 119
9.2 Programming FIFO Level Request Registers 123
9.3 SPC8110 Display Support Specification 125
9.3.1 Displays Supported 125
9.3.2 CRT 125
9.3.3 Single Panel STN 125
9.3.4 Dual Panel STN 125
9.3.5 TFT Panel 126
9.3.6 Simultaneously Displayed CRT and Single Panel STN 126
9.3.7 Simultaneously Displayed CRT and Dual Panel STN 126
9.3.8 Simultaneously Displayed CRT and TFT 127
9.3.9 Vertical Timing for Simultaneous Display 127
9.4 DRAM Refresh 128
9.4.1 Functional Overview 128
9.5 Sprite/Hardware Cursor 129
9.5.1 Sprite Positioning 130
9.5.2 Sprite Memory Organization 131
9.6 BitBlt Specification 132
9.6.1 Overview 132
9.6.2 Features 132
9.7 Power Save Modes (PSM) 133
9.7.1 Functional Overview 133
9.7.2 Output Pin States Affected By Power Save Functions 135
10 MECHANICAL DATA 137
Appendix A TRADEMARKS 138

THIS PAGE LEFT BLANK

List of Tables

Table 5-1: CPU - PCI Bus Interface Pins (MDA[4] = 1 on rising edge of RESET#) 20
Table 5-2: CPU - Intel486/VL-Bus Interface Pins (MDA[4] = 0 on falling edge of RESET#) 21
Table 5-3: Video Memory Interface Pins 21
Table 5-4: Clock Input Pins 22
Table 5-5: CRT Interface Pins 22
Table 5-6: LCD Panel Interface Pins 23
Table 5-7: Miscellaneous Pins 23
Table 5-8: Power Save Mode Controls 24
Table 5-9: Power Supply Pins 24
Table 5-10: Configuration Options 25
Table 5-11: Intel486/VL-Bus to PCI Bus Pin Mapping 25
Table 5-12: Default Pin Assignments for Panel Options 26
Table 5-13: Alternate Pin Mapping for Panel Options 27
Table 6-1: Absolute Maximum Ratings 28
Table 6-2: Recommended Operating Conditions 28
Table 6-3: Input Specifications 28
Table 6-4: Output Specifications 29
Table 7-1: VL-Bus/Intel486 DX Input/Output Timing 31
Table 7-2: PCI Local Bus Input/Output Timing 33
Table 7-3: DRAM Write/Read Timing 35
Table 7-4: CAS Before RAS Refresh Timing 35
Table 7-5: Self Refresh Timing 36
Table 7-6: Power Down Mode CAS Before RAS Refresh Timing 36
Table 7-7: Clock Input Timing Requirements 37
Table 7-8: Single Monochrome 4-bit Panel A.C. Timing 39
Table 7-9: Single Monochrome 8-bit Panel A.C. Timing 41
Table 7-10: Single Color 8-bit Panel A.C. Timing 43
Table 7-11: Single Color 8-bit Panel A.C. Timing (Alternate Data Format) 45
Table 7-12: Single Color 16-bit Panel A.C. Timing 47
Table 7-13: Dual Monochrome 8-bit Panel A.C. Timing 49
Table 7-14: Dual Color 6-bit Panel A.C. Timing 51
Table 7-15: Dual Color 8-bit Panel A.C. Timing (One Shift Clock) 53
Table 7-16: Dual Color 8-bit Panel A.C. Timing (Two Shift Clocks) 55
Table 7-17: Dual Color 16-bit Panel A.C. Timing 57
Table 7-18: TFT A.C. Timing 61
Table 8-1: Unsupported VGA Registers 62
Table 8-2: Clock Select Options 63
Table 8-3: Horizontal Panning Register Display Shift 70
Table 8-4: Sprite/HW Cursor Palette Scheme Select 80
Table 8-5: LCD/CRT Enable Effects On VGA Registers 85
Table 8-6: Blink Select Bits 87
Table 8-7: LCD Panel Selection Examples 88
Table 8-8: TFT Panel Selection Examples 88
Table 8-9: DRAM Refresh Rate Selection 89
Table 8-10: MClk/BClk Wait State Selection 89
Table 8-11: WE Control Selection 90
Table 8-12: Interface Bus Type Select 95
Table 8-13: Time Out Select Value 96
Table 8-14: PSxPIN Active Polarity Selection 97
Table 8-15: Gray Scale Reduction Palette Mapping 98
Table 8-16: DPMS Effects On VSYNC and HSYNC 99

Table 8-17: PS Skew B Time Period 100
Table 8-18: PS Skew C Time Period 101
Table 8-19: PS Skew D Time Period 101
Table 8-20: PS Skew A Time Period 102
Table 8-21: LCD Panel Shut-Down Sequence Timing 103
Table 8-22: TFT Panel Shut-Down Sequence Timing 104
Table 8-23: Frame Start Boundary 105
Table 8-24: Selection Matrix of FIFO Registers 111
Table 8-25: BitBLT Operation Select 115
Table 9-1: Recommended PLL Numerator, Denominator, and Gain Select Values 120
Table 9-2: Recommended PLL Register Programmed Frequencies 120
Table 9-3: 32-bit Memory Data Width 121
Table 9-4: 16-bit Memory Data Width 121
Table 9-5: CRTC Scan Resolution Totals 122
Table 9-6: Values for C and K 123
Table 9-7: Hi and Low FIFO Level Request Register Values for Some Common Modes 124
Table 9-8: Supported Display Selection 125
Table 9-9: Sprite/HW Cursor Palette Schemes 129
Table 9-10: Power Save Mode Architecture 133
Table 9-11: Output Pin States Affected By Power Save Functions With CRT Enabled / LCD Disabled 135
Table 9-12: Output Pin States Affected By Power Save Functions With LCD Enabled / CRT Disabled 135

List of Figures

Figure 1: Typical System Block Diagram (PCI Bus - 1024KB Option) . . . 15
Figure 2: Typical System Block Diagram (VL-Bus - 1024KB Option) . . . 16
Figure 3: Typical System Block Diagram (Intel486 DX 33 Local Bus - 512KB Option) . . . 16
Figure 4: Functional Block Diagram . . . 17
Figure 5: Pinout Diagram . . . 19
Figure 6: VL-Bus/Intel 486 DX Timing . . . 30
Figure 7: VL-Bus/Intel 486 DX Input/Output Timing . . . 30
Figure 8: PCI Local Bus Timing . . . 32
Figure 9: Disconnect Cycles . . . 32
Figure 10: PCI Input/Output Timing . . . 33
Figure 11: DRAM Write/Read Timing . . . 34
Figure 12: CAS Before RAS Refresh . . . 35
Figure 13: Self Refresh Memory Control Output . . . 36
Figure 14: Delay Pulse CBR Refresh . . . 36
Figure 15: Clock Input Requirements . . . 37
Figure 16: Single Monochrome 4-bit Panel Timing . . . 38
Figure 17: Single Monochrome 4-bit Panel A.C. Timing . . . 39
Figure 18: Single Monochrome 8-bit Panel Timing . . . 40
Figure 19: Single Monochrome 8-bit Panel A.C. Timing . . . 41
Figure 20: Single Color 8-bit Panel Timing . . . 42
Figure 21: Single Color 8-bit Panel A.C. Timing . . . 43
Figure 22: Single Color 8-bit Panel Timing (Alternate Data Format) . . . 44
Figure 23: Single Color 8-bit Panel A.C. Timing (Alternate Data Format) . . . 45
Figure 24: Single Color 16-bit Panel Timing . . . 46
Figure 25: Single Color 16-bit Panel A.C. Timing . . . 47
Figure 26: Dual Monochrome 8-bit Panel Timing . . . 48
Figure 27: Dual Monochrome 8-bit Panel A.C. Timing . . . 49
Figure 28: Dual Color 6-bit Panel Timing . . . 50
Figure 29: Dual Color 6-bit Panel A.C. Timing . . . 51
Figure 30: Dual Color 8-bit Panel Timing (One Shift Clock) . . . 52
Figure 31: Dual Color 8-bit Panel A.C. Timing (One Shift Clock) . . . 53
Figure 32: Dual Color 8-bit Panel Timing (Two Shift Clocks) . . . 54
Figure 33: Dual Color 8-bit Panel A.C. Timing (Two Shift Clocks) . . . 55
Figure 34: Dual Color 16-bit Panel Timing . . . 56
Figure 35: Dual Color 16-bit Panel A.C. Timing . . . 57
Figure 36: 18-bit TFT Panel Timing . . . 58
Figure 37: 2 x 9-bit TFT Panel Timing . . . 59
Figure 38: TFT A.C. Timing . . . 60
Figure 39: LCD Panel Shut-Down Sequence . . . 103
Figure 40: TFT Panel Shut-Down Sequence . . . 104
Figure 41: PCI Target Device Minimum Assertion Time for TRDY# During Palette Snoop . . . 117
Figure 42: PLL Block Diagram . . . 119
Figure 43: Vertical Timing for Simultaneous Display . . . 127
Figure 44: Sprite/HW Cursor Display Memory . . . 129
Figure 45: Sprite Memory Organization . . . 131
Figure 46: Mechanical Drawing QFP8-208 . . . 137

THIS PAGE LEFT BLANK

1 INTRODUCTION

1.1 Scope

This is the Functional Specification for the SPC8110F0A LCD VGA Controller Chip. Included in this document are timing diagrams, AC and DC characteristics, register descriptions, and power management descriptions. This document is intended for two audiences; Video Subsystem Designers and Software Developers.

2 FEATURES

2.1 General

- supports 1 megabyte of Flat Addressing above the 1 megabyte address boundary
- bit block transfer (BitBLT) engine
- color expansion
- programmable 64 x 64 x 2-bit hardware sprite, or 64 x 64 x 2-bit hardware cursor
- 32-bit wide display pipeline
- asynchronous memory architecture for optimal memory performance
- two hardware and two software fully configurable power save modes

2.2 Technology

- low power CMOS
- 3.3V core operation
- internal voltage translators for configurable 3.3V or 5.0V I/O levels
- 208 pin QFP 28 x 28 mm surface mount package

2.3 System

- 32-bit PCI Bus
- 32-bit VESA VL-Bus™ or Intel486™ DX 33 MHz Local Bus
- internal 256 x18-bit RAMDAC
- half frame buffer architecture for dual panel dual drive LCD
- two internal fully programmable PLLs (Memory clock MClk & Pixel clock PCLk) driven by a single external reference clock
- max MClk = 40 MHz
- max PClk = 65 MHz
- 512KB or 1024KB display memory using one or two 256K x 16 DRAM respectively
- selectable DRAM interface configurations: CAS or WE controlled; symmetrical or asymmetrical addressing
- programmable DRAM refresh rate
- supports distributed self refresh DRAM
- 32 kHz 50% duty cycle clock required for power save and refresh circuitry
- board level connectivity test mode

2.4 Compatibility

- support for all standard VGA modes (0 to 13h) on LCD or CRT
- support for up to 1024x768 at 256 colors on CRT
- 16 shades of gray by Frame Rate Modulation (FRM) on monochrome LCD
- 32 or 64 shades of gray by combining Frame Rate Modulation (FRM) and dithering on monochrome LCD in 256 color modes
- maximum 256 simultaneous colors from a possible 256K colors on LCD or CRT
- hardware vertical expansion and centering on LCD

2.5 Display Support

- simultaneous display on both CRT and LCD/TFT
- direct analog VGA and multifrequency CRT displays supported
- supports the following panels:

  Single Monochrome 4-bit
  Single Monochrome 8-bit
  Single Color 8-bit
  Single Color 16-bit
  Dual Monochrome 8-bit
  Dual Color 6-bit
  Dual Color 8-bit
  Dual Color 16-bit
  TFT

2.6 Display Modes

Mode No. (Hex)	Pixels Addressable		Horizontal Pixels Displayed		Vertical Pixels Displayed	Font Size		Number of Shades			Nominal Frame Rate (Hz) for 1024KB RAM		
	Horizontal (characters)	Vertical (rows)	CRT	LCD		CRT	LCD	Mono LCD	Color LCD	CRT	CRT (typical)	Single Panel LCD	Dual Panel LCD
0	(40)	(25)	320	320	200	8x8	8x8	16	16	16	70	79	78
0+	(40)	(25)	320	320	350	8x14	8x14	16	16	16	70	79	78
0++	(40)	(25)	360	320	400	9x16	8x16	16	16	16	70	79	78
1	(40)	(25)	320	320	200	8x8	8x8	16	16	16	70	79	78
1+	(40)	(25)	320	320	350	8x14	8x14	16	16	16	70	79	78
1++	(40)	(25)	360	320	400	9x16	8x16	16	16	16	70	79	78
2	(80)	(25)	640	640	200	8x8	8x8	16	16	16	70	79	78
2+	(80)	(25)	640	640	350	8x14	8x14	16	16	16	70	79	78
2++	(80)	(25)	720	640	400	9x16	8x16	16	16	16	70	79	78
3	(80)	(25)	640	640	200	8x8	8x8	16	16	16	70	79	78
3+	(80)	(25)	640	640	350	8x14	8x14	16	16	16	70	79	78
3++	(80)	(25)	720	640	400	9x16	8x16	16	16	16	70	79	78
4	320	200	320	320	200	n/a	n/a	4	4	4	70	79	78
5	320	200	320	320	200	n/a	n/a	4	4	4	70	79	78
6	640	200	640	640	200	n/a	n/a	2	2	2	70	79	78
7	(80)	(25)	640	640	350	8x14	8x14	2	2	2	70	79	78
7+	(80)	(25)	720	640	400	9x16	8x16	2	2	2	70	79	78
0D	320	200	320	320	200	n/a	n/a	16	16	16	70	79	78
0E	640	200	640	640	200	n/a	n/a	16	16	16	70	79	78
0F	640	350	640	640	350	n/a	n/a	2	2	2	70	79	78
10	640	350	640	640	350	n/a	n/a	16	16	16	70	79	78
11	640	480	640	640	480	n/a	n/a	2	2	2	60	79	78
12	640	480	640	640	480	n/a	n/a	16	16	16	60	79	78
13	320	200	320	320	200	n/a	n/a	64	256	256	70	79	78
6A	800	600	800	800	600	n/a	n/a	16	16	16	72	60	72
100	640	400	640	640	400	n/a	n/a	64	256	256	70	79	78
101	640	480	640	640	480	n/a	n/a	64	256	256	60	79	78
102	800	600	800	800	600	n/a	n/a	16	16	16	72	60[2]	72
103	800	600	800	800	600	n/a	n/a	64	256	256	72	60[2]	72[1]
104	1024	768	1024	1024	768	n/a	n/a	16	16	16	60	n/a[2]	70
105	1024	768	1024	n/a	768	n/a	n/a	64	n/a	256	60[1]	n/a[2]	n/a
108	(80)	(60)	640	640	480	8x8	8x8	16	16	16	60	79	78
109	(132)	(25)	1056	n/a	350	8x14	n/a	n/a	n/a	16	70	n/a	n/a
10A	(132)	(43)	1056	n/a	350	8x8	n/a	n/a	n/a	16	70	n/a	n/a
10B	(132)	(50)	1056	n/a	400	8x8	n/a	n/a	n/a	16	70	n/a	n/a
10C	(132)	(60)	1056	n/a	480	8x8	n/a	n/a	n/a	16	60	n/a	n/a

[1] Only available with 1024KB RAM.
[2] Support for large TFT panels requires custom BIOS and drivers.

3 OVERVIEW DESCRIPTION

The SPC8110F0A is a single chip multi-function low voltage LCD/CRT VGA controller with an integrated RAMDAC and Liquid Crystal Display interface. With a built-in Hardware Cursor, a Bit Block Transfer Engine, two PLLs, and a CPU Local Bus Interface the SPC8110F0A accelerates the display of Graphical User Interface software on Analog CRT Displays and Single or Dual Panel Monochrome or Color LCD Displays.

Optimized for cost and power savings, the SPC8110F0A mixes 3.3 Volt 0.72 micron standard cell and gate array technology to achieve high density Palette RAM, DAC, PLL, and digital logic integration. Power consumption in 8-bit monochrome LCD modes is approximately 470 mW peak during extensive 32-bit 33 MHz Local Bus BitBLT operations, 170 mW in normal operation and 0.5 mW in standby with self-refresh DRAM.

3.1 Typical System Block Diagrams

The following figures show typical VL-Bus and PCI Bus system implementations with SPC8110F0A.

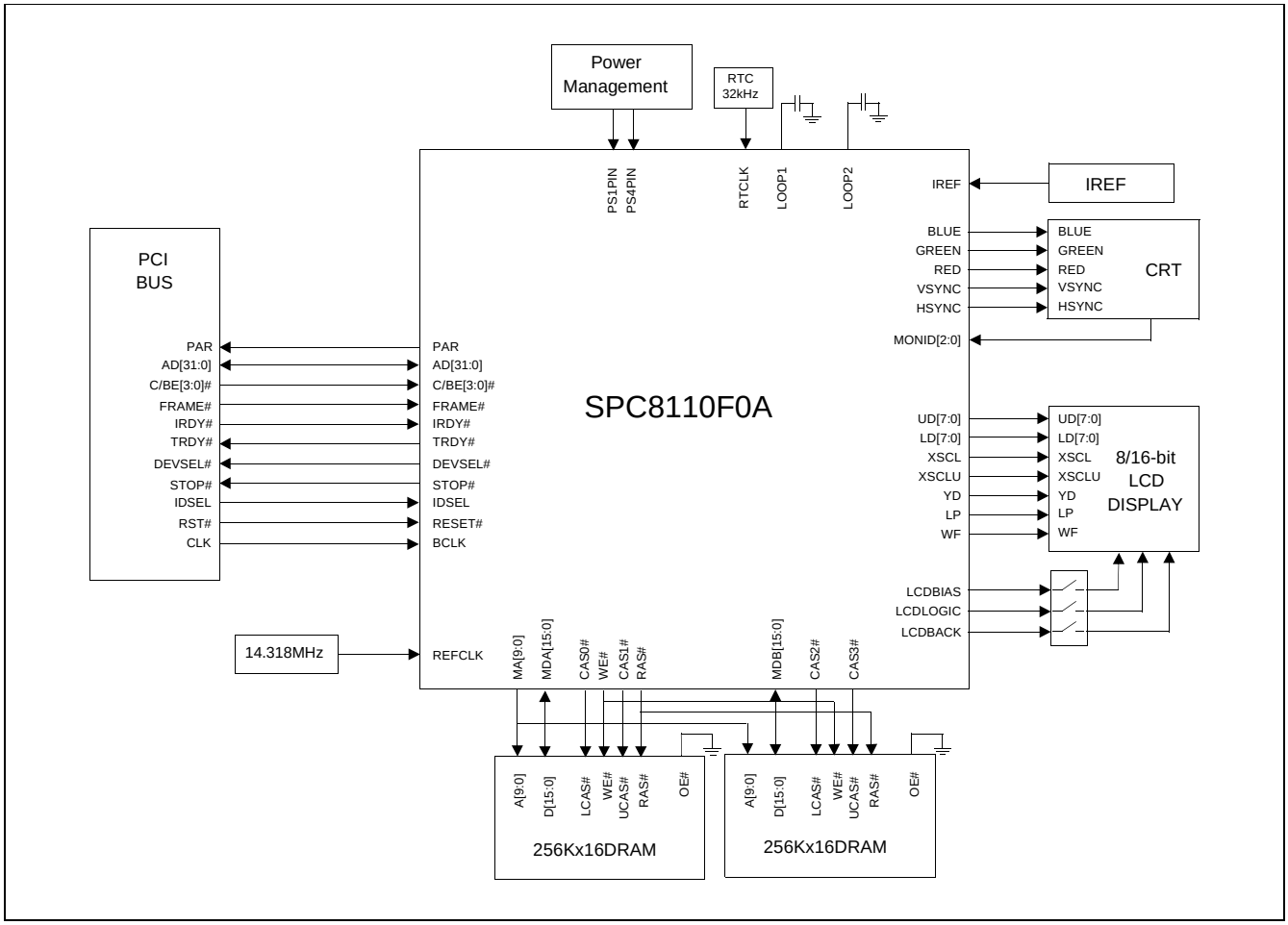


Figure 1: Typical System Block Diagram (PCI Bus - 1024KB Option)

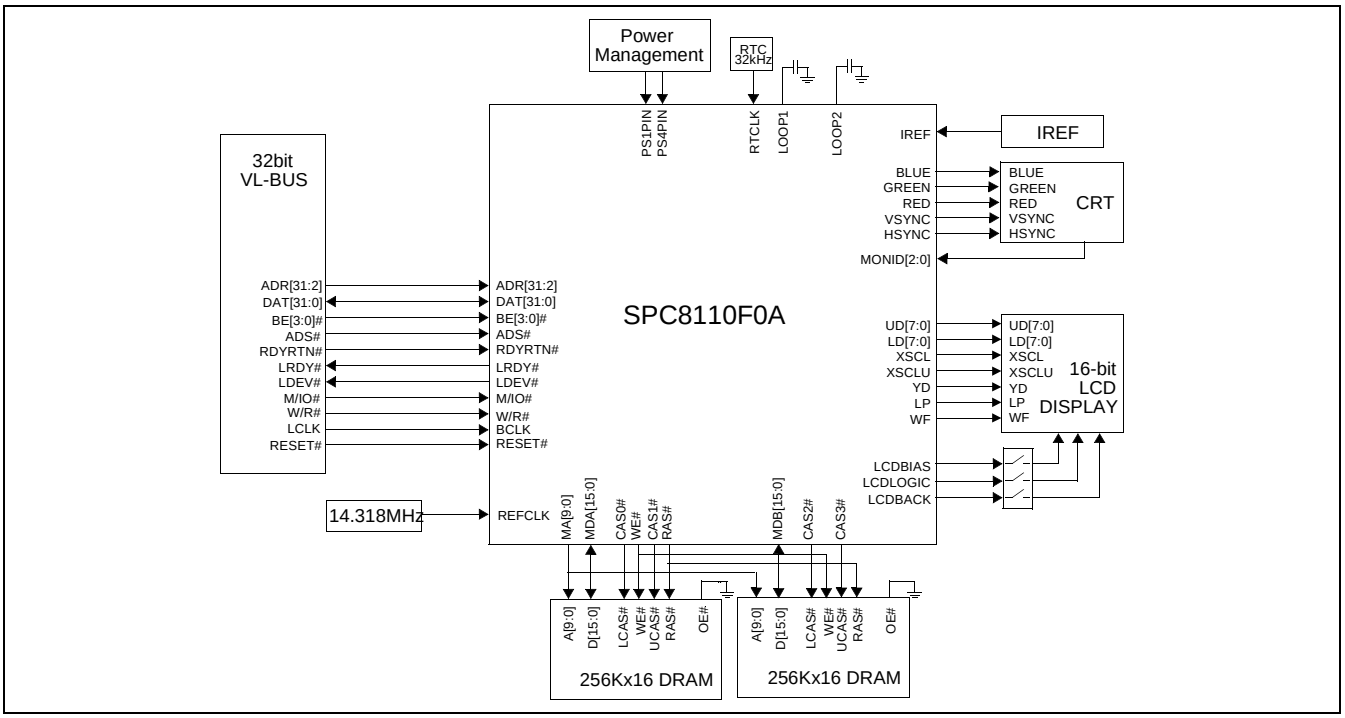


Figure 2: Typical System Block Diagram (VL-Bus - 1024KB Option)

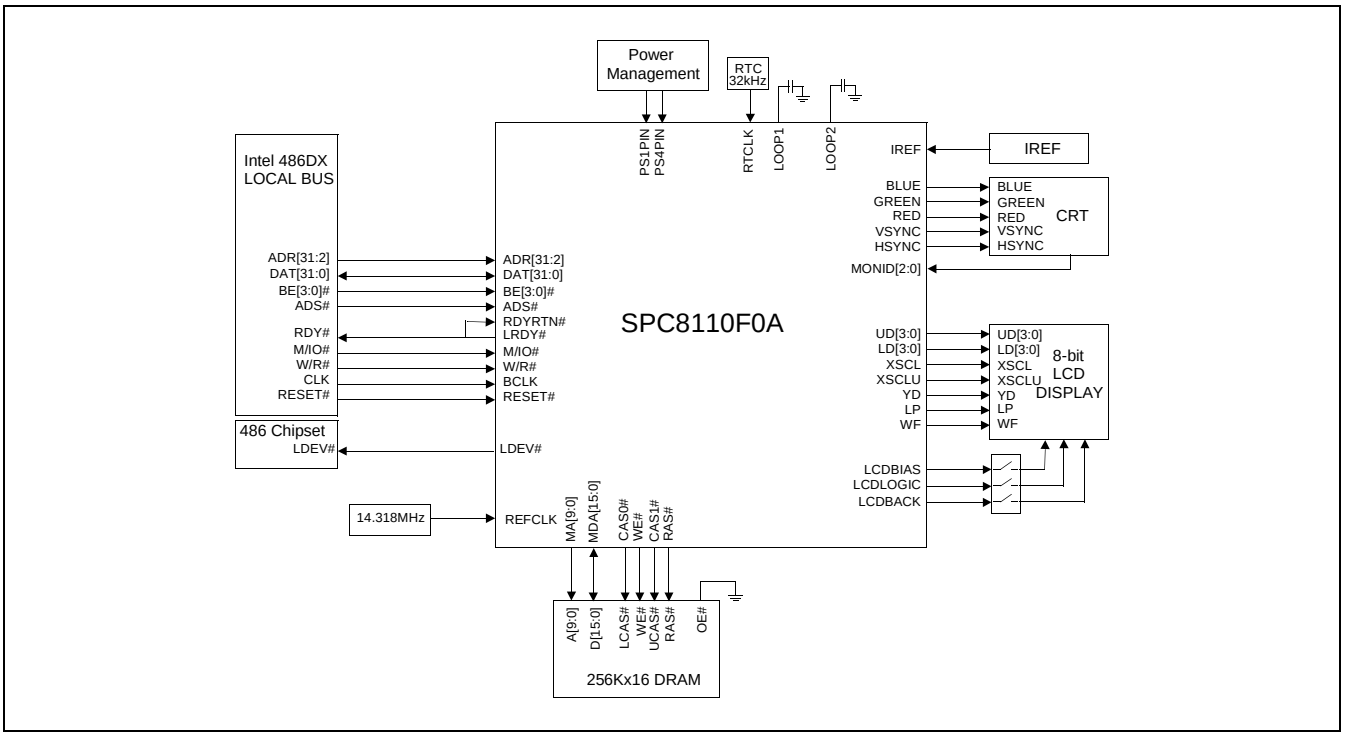


Figure 3: Typical System Block Diagram (Intel486 DX 33 Local Bus - 512KB Option)

3.2 Functional Block Diagram

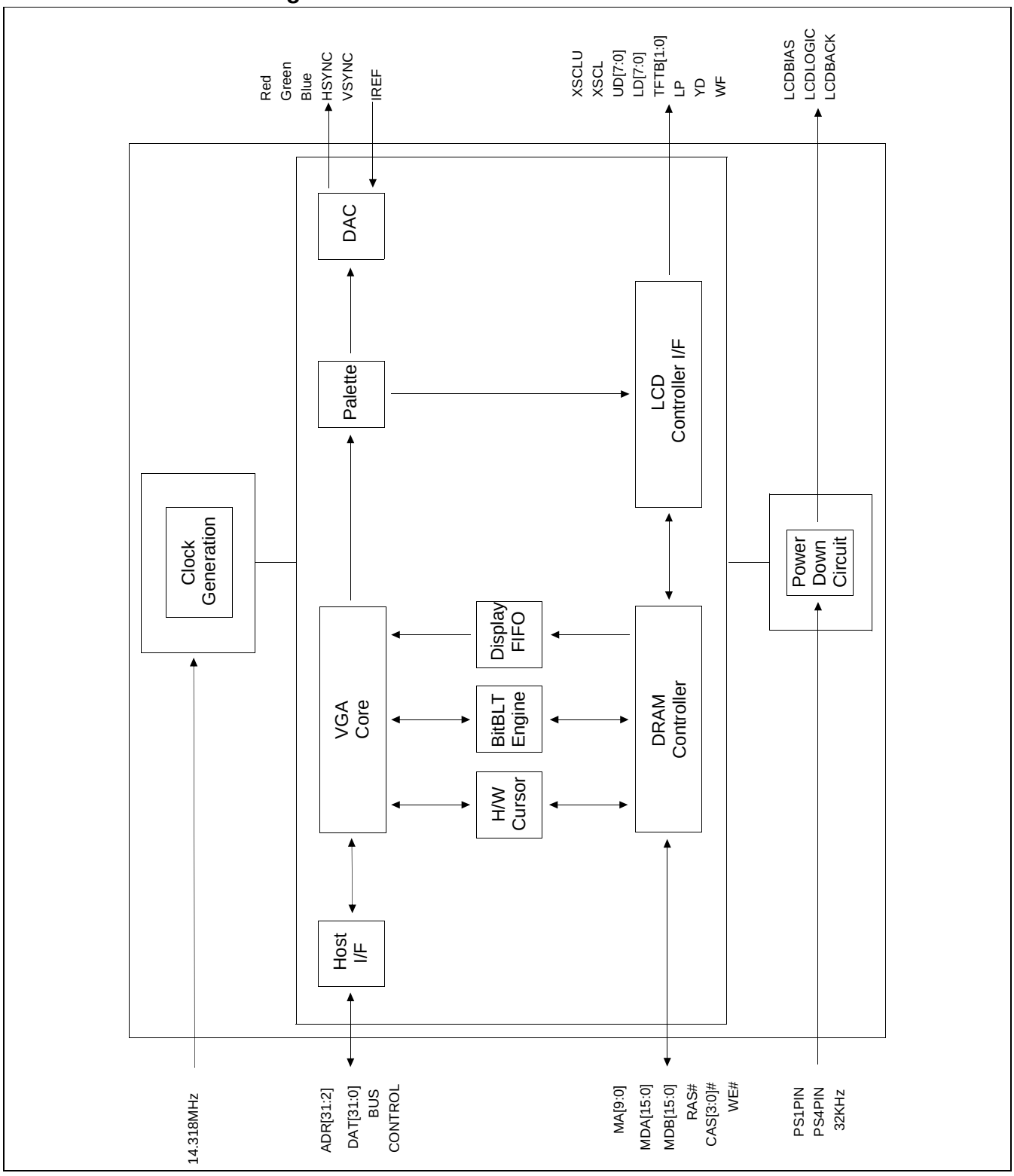


Figure 4: Functional Block Diagram

3.3 Functional Block Description

3.3.1 Host Interface

The Host Interface can be programmed to accommodate any of the following three standards: Intel486 DX local bus interface, VL-Bus interface and PCI interface. It has a one-stage buffer for zero wait-state write operation.

3.3.2 Clock Generator

The Clock Generator contains two PLLs that are separately programmed to produce the memory and pixel clocks from a single clock source. The reference clock is usually the 14.318 MHz bus clock.

3.3.3 VGA Core

The VGA Core contains the Sequencer, CRTC Controller, Graphics Controller, Attribute Controller, and the rest of the standard VGA circuitry.

3.3.4 Hardware Cursor

The Hardware Cursor generates a 64 x 64 x 2-bit hardware cursor or sprite that can be overlaid on the displayed image.

3.3.5 BitBLT Engine

The Bit Block Transfer Engine performs read, write, and move blits, solid fills, destination inversions, and pattern fills. It performs all data alignment and masking at the blit boundary. It also performs color expansion to accelerate the writing of text images. The Bit Block Transfer Engine operates in both 4-bit planar mode and 8-bit linear (packed-pixel) mode.

3.3.6 Display FIFO

The Display FIFO is an 8 stage FIFO that is used to buffer the video data from display memory.

3.3.7 VGA Palette

The VGA Palette implements the standard 256-word x 18-bit VGA lookup table, plus 4-word x 18-bit entries for Hardware Sprites.

3.3.8 DAC

The DAC functions as a triple 6-bit 65 MHz DAC to drive the RGB outputs connected to the analog display.

3.3.9 LCD Interface

The LCD Interface contains frame rate modulation and dithering circuitry for a maximum of 64 shades of gray in monochrome single and dual panel modes. In color LCD mode, it uses frame rate modulation to generate 4K colors, and additional dithering techniques for a full 256K colors.

3.3.10 Power Save Logic

Power Save Logic implements all the power down features of the chip.

4 PIN OUT DIAGRAM

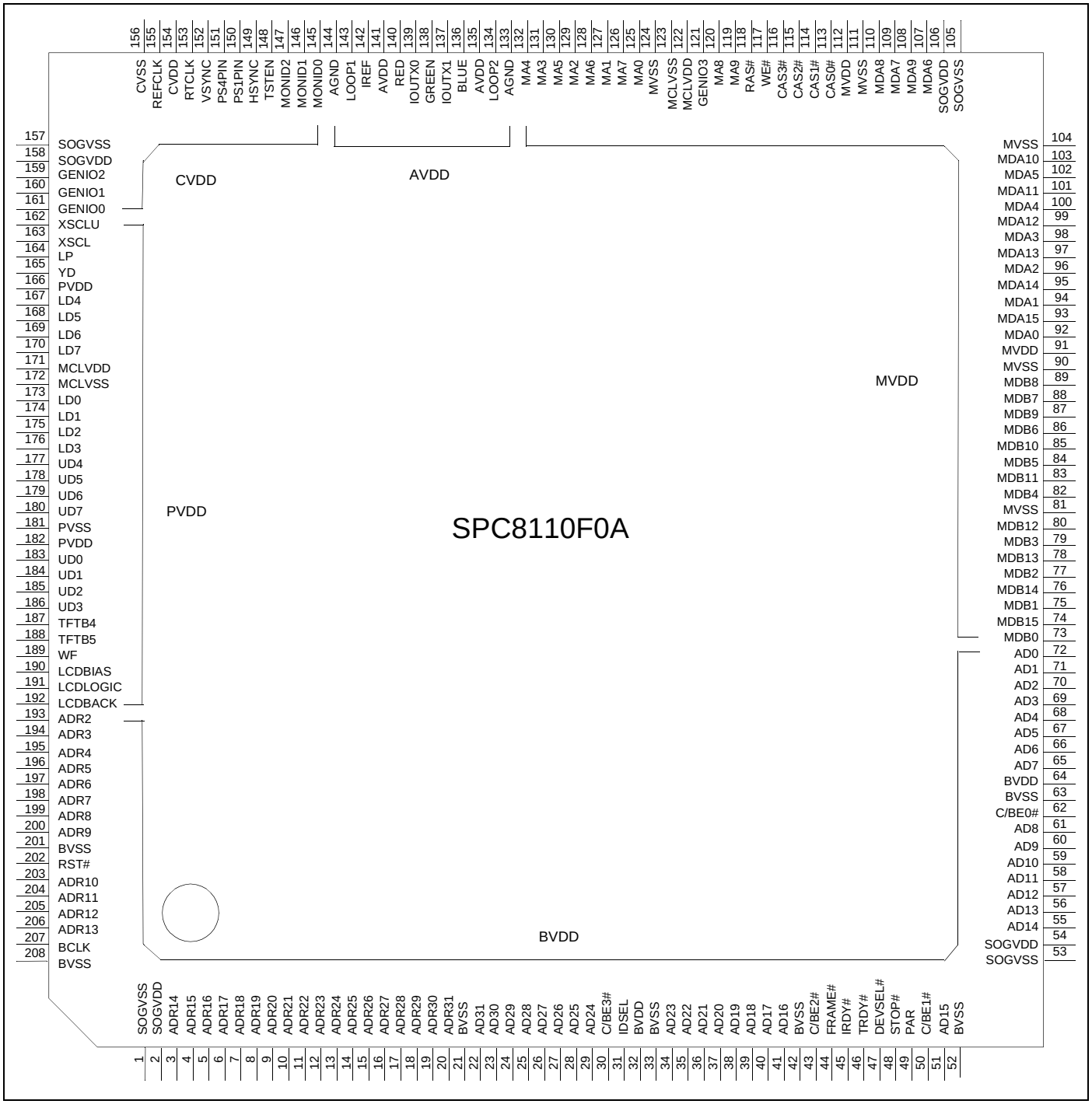


Figure 5: Pinout Diagram

Pinout placement subject to change. Some pins have dual use - default configuration is shown.
See PIN DESCRIPTIONS on page 20 for details.
Package type: QFP8-208 pin surface mount

5 PIN DESCRIPTIONS

Type Key

A	=	Analog
I	=	Input
O	=	Output
I/O	=	Bidirectional
P	=	Power
SCH	=	Schmitt Trigger
PD	=	Pull Down

See D.C. Characteristics, Output Specifications on page 29 for Output Type.

Table 5-1: CPU - PCI Bus Interface Pins (MDA[4] = 1 on rising edge of RESET#)

Pin Name	Type	Output Type	Pin No.	Description
AD[31:0]	I/O	4	22-29, 34-41, 51, 55-61, 65-72	PCI multiplexed Address and Data Bus. These lines are driven by the chip only during read cycles, and are in a hi-Z state at all other times.
ADR[31:2]	I	--	20-3, 206-203, 200-193	Unused VL-Bus Address inputs. These pins should be tied high in PCI mode.
C/BE[3:0]#	I	--	30, 43, 50, 62	PCI Bus Command and Byte Enables.
IDSEL	I	--	31	PCI Bus Initialization Device Select.
STOP#	I/O	4	48	PCI Bus Stop.
FRAME#	I	--	44	PCI Bus Cycle Frame.
IRDY#	I	--	45	PCI Bus Initiator Ready.
PAR	O	4	49	Parity. This line is driven by the chip only during read cycles, and is in a hi-Z state at all other times. It is always hi-Z and should be tied high in VL Bus mode.
BCLK	I	--	207	PCI Bus Clock.
TRDY#	I/O	4	46	PCI Bus Target Ready.
DEVSEL#	O	4	47	PCI Bus Device Select.
RST#	I w/SCH	--	202	CPU Reset. The active low reset signal from the CPU clears all internal registers and forces all signals to their inactive state. On the rising edge of the RESET# the MDA[0...15] bus is latched in for configuration.

Table 5-2: CPU - Intel486/VL-Bus Interface Pins (MDA[4] = 0 on falling edge of RESET#)

Pin Name	Type	Output Type	Pin No.	Description
ADR[31:2]	I	--	20-3, 206-203, 200-193	VL-Bus Address inputs. These inputs are unused and must be tied high in PCI Bus mode.
DAT[31:0]	I/O	4	22-29, 34-41, 51, 55-61, 65-72	VL-Bus Data inputs. These lines are driven by the chip only during read cycles, and are in a hi-Z state at all other times.
BE[3:0]#	I	--	30, 43, 50, 62	VL-Bus byte enables.
W/R#	I	--	31	VL-Bus Write or Read Status.
M/IO#	I	4	48	VL-Bus Memory or I/O Status.
ADS#	I	--	44	VL-Bus Address Data Strobe.
RDYRTN#	I	--	45	VL-Bus Ready Return.
BCLK	I	--	207	VL-Bus Local CPU Clock.
LRDY#	O	4	46	VL-Bus Local Ready.
LDEV#	O	4	47	VL-Bus Local Device.
RESET#	I w/SCH	--	202	CPU Reset. The active low reset signal from the CPU clears all internal registers and forces all signals to their inactive state. On the rising edge of the RESET# the MDA[0...15] bus is latched in for configuration.
PAR	O	4	49	Unused (Parity). This line is used in PCI mode only. It is hi-Z at all times in VL mode and should be tied high.

Table 5-3: Video Memory Interface Pins

Pin Name	Type	Output Type	Pin No.	Description
MA[9:0]	O	2	119, 120, 126, 128, 130, 132, 131, 129, 127, 125	Multiplexed row/column address bits for video display memory.
MDA[15:0]	I/O w/ PD	2	93, 95, 97, 99, 101, 103, 108, 110, 109, 107, 102, 100, 98, 96, 94, 92	Data bits for video display memory. The output drivers of these pins are placed into a high-impedance state when RESET# is low, or when MClkOFF is active. These pins are also used as configuration inputs. They have internal pull down resistors that have typical values of 50KΩ/100KΩ at 5V/3.3V respectively.
MDB[15:0]	I/O w/ PD	2	74, 76, 78, 80, 83, 85, 87, 89, 88, 86, 84, 82, 79, 77, 75, 73	Data bits for video display memory when 1024KB of memory is present. The output drivers of these pins are placed into a high-impedance state when RESET# is low, or when MClkOFF is active. The inputs have internal pull down resistors that have typical values of 50K Ω/100K Ω at 5 V/3.3 V respectively. They should be left open-circuit when only 512KB of memory is attached.
RAS#	O	2	118	DRAM Row Address Strobe
CAS[0]# (CAS#)	O	2	113	DRAM Column Address Strobe for MDA[7:0], or Column Address Strobe for all bytes, as configured by AUX [03h] bit 7.

Table 5-3: Video Memory Interface Pins (Continued)

Pin Name	Type	Output Type	Pin No.	Description
CAS[3:1]# (WE[3:1]#)	O	2	116-114	DRAM Column Address Strobes or Write Enable Strobes for {MDB[15:0],MDA[15:8]}, as configured by AUX[03h] bit 7. CAS[3:2]# (WE[3:2]#) are unused and should be left open-circuit when only 512KB of memory is present.
WE# (WE[0]#)	O	2	117	DRAM Write Enable Strobe for all bytes or Write Enable Strobe for MDA[7:0], as configured by AUX[03h] bit 7

Table 5-4: Clock Input Pins

Pin Name	Type	Output Type	Pin No.	Description
REFCLK	I	--	155	This pin is the reference clock used by the internal PLLs to generate all the necessary clocks. This must be stable during full operation. REFCLK may shut down only after the chip is totally powered down. Input frequency is typically 14.318 MHz. (See Clock Input Requirements on page 37).
RTCLK	I	--	153	Real Time Clock. This input must be used to provide a low frequency clock for generating refresh. This clock must be approximately 32 kHz and 50% duty cycle. (See Clock Input Requirements on page 37.)

Table 5-5: CRT Interface Pins

Pin Name	Type	Output Type	Pin No.	Description
RED, GREEN, BLUE	AO	--	140, 138, 136	Analog outputs from the Video DAC. Internal comparator for monitor sense is available on all three pins.
HSYNC	O	3	149	Horizontal Sync. This output is driven to indicate the horizontal retrace period. In CRT only mode, the polarity of this signal is controlled by register 3C2h, bit 6. In TFT or double scan mode this bit is active low to indicate 640 x 480 resolution. This pin is held low in LCD modes. This pin follows the DPMS standard during power down modes.
VSYNC	O	3	152	Vertical Sync. This output is driven to indicate the vertical retrace period. In CRT only mode, the polarity of this signal is controlled by register 3C3h, bit 7. In TFT or double scan mode this bit is active low to indicate 640 x 480 resolution. This pin is held low in LCD modes. This pin follows the DPMS standard during power down modes.
IREF	AI	--	142	Current reference input for the Video DAC.
MONID[2:0]	I	--	147-145	Monitor ID bits. Connected to the CRT to identify the monitor type.

Table 5-6: LCD Panel Interface Pins

Pin Name	Type	Output Type	Pin No.	Description
YD	O	4	165	Vertical Scanning Start Pulse output. A logic 1 on this signal, sampled by the LCD module on the falling edge of LP, is used by the panel row drivers (Y drivers) to indicate the start of the vertical frame.
LP	O	4	164	Latch Pulse output. The falling edge of this signal is used to latch a row of display data in the LCD module's column driver shift registers and to turn on the row driver (Y driver) for that line.
XSCL	O	4	163	Shift Clock for LCD data. Display data is clocked out of the chip on the rising edge of this signal, to be shifted into the LCD panel module column drivers (X drivers) on each falling edge.
XSCLU	O	4	162	Second Shift Clock for some color LCD displays.
UD[7:0] LD[7:0]	I/O	4	180-177, 186-183, 170-167, 176-173	Panel display data bus. The data format depends on the specific panel connected. These pins are driven low when the LCD interface is disabled (e.g. CRT only mode). For 8-bit panels, UD[7:4] and LD[7:4] are driven low.
TFTB[5:4]	O	4	188-187	TFT display data bus for the two lsb of the color blue. For non-TFT panels, these two pins are driven low.
LCDBIAS	O	2	190	LCD power control for the LCD bias circuitry. Active (On) polarity is defined by the state of MDA[3] on the rising edge of RESET#.
LCDBACK	O	2	192	LCD power control for the LCD back light. Active (On) polarity is defined by the state of MDA[1] on the rising edge of RESET#.
LCDLOGIC	O	2	191	LCD power control for the LCD logic circuitry. Active (On) polarity is defined by the state of MDA[2] on the rising edge of RESET#.
WF	O	4	189	LCD Backplane Bias signal. Output toggling frequency is programmable in an auxiliary register. For TFT panels, this pin outputs the display enable signals.

Table 5-7: Miscellaneous Pins

Pin Name	Type	Output Type	Pin No.	Description
TSTEN	I w/ PD	--	148	This pin, when high, sets the SPC8110F0A into either boundary pin SCAN or pin output drive test, depending on the state of PS1PIN. This input has an internal pull down resistor that has a typical value of 50K/100K Ω at 5 V/3.3 V respectively.
GENIO[3:0]	I/O w/ PD	2	121, 159, 160 161	General purpose I/O pins. Output state of each pin is programmable to control external devices. See AUX[34h].
LOOP1	I	--	143	Connects to Loop Filter Resister and Capacitor for PLL1, where suggested values are R=70 Ω and C=0.1 uf.
LOOP2	I	--	134	Connects to Loop Filter Resister and Capacitor for PLL2, where suggested values are R=70 Ω and C=0.1 uf.
IOUTX[1:0]	A	--	137, 139	Balanced current output for the DAC.

Table 5-8: Power Save Mode Controls

Pin Name	Type	Output Type	Pin No.	Description
PS1PIN	I w/ SCH, PD	--	150	Pin used to initiate a power save mode 1. The polarity of this pin is programmable and has a default polarity of active low. This input has an internal pull down resistor that has a typical value of 50K/100K Ω at 5 V/3.3 V respectively. When TSTEN is active, the polarity of PS1PIN selects the test to be either boundary pin SCAN (PS1PIN = 1) or pin output drive test (PS1PIN = 0)
PS4PIN	I w/ SCH, PD	--	151	Pin used to initiate a power save mode 4. The polarity of this pin is programmable and has a default polarity of active low. This input has an internal pull down resistor that has a typical value of 50K/100K Ω at 5 V/3.3 V respectively.

Table 5-9: Power Supply Pins

Pin Name	Type	Pin No.	Description
SOGVDD	P	158, 106, 54, 2	V_{DD} supply for core logic. (3.3V only.)
MVDD	P	112, 91	V_{DD} supply for memory pins. (Either 3.3V or 5.0V.)
BVDD	P	32, 64	V_{DD} supply for bus interface pins. (Either 3.3V or 5.0V.)
CVDD	P	154	V_{DD} supply for control interface pins. (Either 3.3V or 5.0V.)
PVDD	P	166, 182	V_{DD} supply for panel interface pins. (Either 3.3V or 5.0V.)
AVDD	P	135, 141	Analog power supply. (3.3V only.)
MCLVDD	P	122, 171	VDD supply for MCL logic. (3.3V only.)
SOGVSS	P	157, 105, 53, 1	VSS supply for core logic.
MVSS	P	111, 124, 104, 90, 81	VSS supply for memory pins.
BVSS	P	201, 208, 21, 42, 52, 33, 63	VSS supply for bus interface pins.
CVSS	P	156	VSS supply for control interface pins.
PVSS	P	181	VSS supply for panel interface pins.
AGND	P	133, 144	Analog Ground.
MCLVSS	P	123, 172	VSS supply for MCL logic.

5.1 Summary of Configuration Options

Table 5-10: Configuration Options

Pin Name	Rising Edge of RESET# Configuration
MDA[0]	Reserved.
MDA[1]	Latched into AUX[0Dh] bit 1 for LCDBACK Active Polarity.
MDA[2]	Latched into AUX[0Dh] bit 2 for LCDLOGIC Active Polarity.
MDA[3]	Latched into AUX[0Dh] bit 3 for LCDBIAS Active Polarity.
MDA[4]	Latched into AUX[0Dh] bit 4 for Local Bus Selected: PCI/VL.
MDA[5]	Reserved.
MDA[6]	Reserved.
MDA[7]	Reserved.
MDA[8:15]	Latched into AUX[0Ch] for software configuration only.

These inputs have internal pulldown resistors. The default configuration state is 0.

5.2 SPC8110F0A Pin Mapping

Table 5-11: Intel486/VL-Bus to PCI Bus Pin Mapping

VL-Bus Name	PCI Name	Pin No.	VL-Bus Name	PCI Name	Pin No.
RESET#	RST#	202	BCLK	BCLK	207
RDYRTN#	IRDY#	45	--	PAR	49
ADR[31:2]	--	20-3, 206-203, 200-193	M/IO#	STOP#	48
W/R#	IDSEL	31	ADS#	FRAME#	44
LDEV#	DEVSEL#	47	LRDY#	TRDY#	46
DAT[31:0]	AD[31:0]	22-29, 34-41, 51, 55-61, 65-72	BE[3:0]#	C/BE[3:0]#	30, 43, 50, 62

Default Pin Mapping for the Various LCD and TFT Panel Options

(AUX[07h]b2 = 0)

Table 5-12: Default Pin Assignments for Panel Options

Pin Name	TFT				Color LCD				Mono LCD	
	18-Bit Color	9-Bit x 2 Color	12-Bit Color	9-Bit Color	16-Bit	8-Bit Half Panel	8-Bit	6-Bit	8-Bit	4-Bit
XSCLU	CLK	CLK	CLK	CLK			XSCLU			
XSCL					XSCL	XSCL	XSCL	XSCL	XSCL	XSCL
LP	HS	HS	HS	HS	LP	LP	LP	LP	LP	LP
YD	VS	VS	VS	VS	YD	YD	YD	YD	YD	YD
LD4	R0	R10			LD4				UD0	UD0
LD5	R1	R11			LD5				UD1	UD1
LD6	R2	R12	R0		LD6				UD2	UD2
LD7	R3	R00	R1	R0	LD7				UD3	UD3
LD0	R4	R01	R2	R1	LD0		LD0	LB	LD0	
LD1	R5	R02	R3	R2	LD1		LD1	LG	LD1	
LD2	G0	G10			LD2		LD2	LR	LD2	
LD3	G1	G11			LD3		LD3		LD3	
UD4	G2	G12	G0		UD4	UD4				
UD5	G3	G00	G1	G0	UD5	UD5				
UD6	G4	G01	G2	G1	UD6	UD6				
UD7	G5	G02	G3	G2	UD7	UD7				
UD0	B0	B10			UD0	UD0	UD0	UB		
UD1	B1	B11			UD1	UD1	UD1	UG		
UD2	B2	B12	B0		UD2	UD2	UD2	UR		
UD3	B3	B00	B1	B0	UD3	UD3	UD3			
TFTB4	B4	B01	B2	B1						
TFTB5	B5	B02	B3	B2						
WF	DE	DE	DE	DE	WF	WF	WF	WF/DE	WF	WF
LCDBIAS					(DISP)	(DISP)	(DISP)	(DISP)	(DISP)	(DISP)

Alternate Pin Mapping for the Various LCD and TFT Panel Options

(AUX[07h]b2 = 1)

Table 5-13: Alternate Pin Mapping for Panel Options

Pin Name	TFT				Color LCD			Mono LCD	
	18-Bit Color	9-Bit x 2 Color	12-Bit Color	9-Bit Color	16-Bit	8-Bit Half Panel	8-Bit	8-Bit Single	8-Bit Dual
XSCLU	CLK	CLK	CLK	CLK			XSCLU		
XSCL					XSCL	XSCL	XSCL	XSCL	XSCL
LP	HS	HS	HS	HS	LP	LP	LP	LP	LP
YD	VS	VS	VS	VS	YD	YD	YD	YD	YD
LD4	R0	R10							
LD5	R1	R11							
LD6	R2	R12	R0		UD0	UD0	UD0		
LD7	R3	R00	R1	R0	UD1	UD1	UD1	D6	UD2
LD0	R4	R01	R2	R1	UD2	UD2	UD2	D7	UD3
LD1	R5	R02	R3	R2	UD3	UD3	UD3		
LD2	G0	G10			UD4	UD4			
LD3	G1	G11			UD5	UD5			
UD4	G2	G12	G0		LD4				
UD5	G3	G00	G1	G0	LD5			D3	LD3
UD6	G4	G01	G2	G1	LD6			D4	UD0
UD7	G5	G02	G3	G2	LD7			D5	UD1
UD0	B0	B10			UD6	UD6			
UD1	B1	B11			UD7	UD7			
UD2	B2	B12	B0		LD0		LD0		
UD3	B3	B00	B1	B0	LD1		LD1	D0	LD0
TFTB4	B4	B01	B2	B1	LD2		LD2	D1	LD1
TFTB5	B5	B02	B3	B2	LD3		LD3	D2	LD2
WF	DE	DE	DE	DE	WF	WF	WF	WF	WF
LCDBIAS					(DISP)	(DISP)	(DISP)	(DISP)	(DISP)

6 D.C. CHARACTERISTICS

Table 6-1: Absolute Maximum Ratings

Symbol	Parameter	Rating	Units
V_{DD}	Supply Voltage	V_{SS} - 0.3 to 7.0	V
V_{IN}	Input Voltage	V_{SS} - 0.3 to V_{DD} + 0.3	V
V_{OUT}	Output Voltage	V_{SS} - 0.3 to V_{DD} + 0.3	V
T_{STG}	Storage Temperature	-65 to 150	°C
T_{SOL}	Solder Temperature/Time	260 for 10 sec. max at lead	°C

Table 6-2: Recommended Operating Conditions

Symbol	Parameter	Condition	Min	Typ	Max	Units
HV_{DD}	Supply Voltage	V_{SS} = 0 V	4.5	5.0	5.5	V
LV_{DD}	Supply Voltage	V_{SS} = 0 V	3.0	3.3	3.6	V
V_{IN}	Input Voltage	V_{SS}	V_{SS}	--	V_{DD}	V
T_{OPR}	Operating Temperature		0	25	70	°C
I_{OPR}	Average Power Consumption			381.7		mW

Table 6-3: Input Specifications

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{IL}	Low Level Input Voltage	V_{DD} = 4.5V/3.0V	--	--	0.8	V
V_{IH}	High Level Input Voltage	V_{DD} = 5.5V/3.6V	2.0	--	--	V
HR_{PD}	Pull Down Resistance	V_{DD} = 5V	25	50	100	KΩ
LR_{PD}	Pull Down Resistance	V_{DD} = 3.3V	45	90	180	KΩ
V_{T+}	Positive-going Threshold CMOS Schmitt Trigger	V_{DD} = 5/3.3V			2.4	V
V_{T-}	Negative-going Threshold CMOS Schmitt Trigger	V_{DD} = 5/3.3V	0.6			V
HV_H	Hysteresis Voltage CMOS Schmitt Trigger	V_{DD} = 5/3.3V	0.1			V
C_{IN}	Clock Pin Capacitance	BCLK, RESET#, REFCLK, MONID[2:0], TSTEN, PS1PIN,PS4PIN, RTCLK		4		pF
C_{BID}	Input Pin Capacitance	all pins not listed in C_{IN}		10		pF
I_L	Input Leakage Current	V_{DD} = MAX V_{IH} = V_{DD} V_{IL} = V_{SS}	-1		1	µA

Table 6-4: Output Specifications

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{OL}	Low Level Output Voltage Type 2 Type 3 Type 4	V_{DD} = 5/3.3V I_{OL} = 6mA I_{OL} = 12mA I_{OL} = 24mA			V_{SS} + 0.3	V
V_{OH}	High Level Output Voltage Type 2 Type 3 Type 4	V_{DD} = 5/3.3V I_{OH} = -2mA I_{OH} = -4mA I_{OH} = -8mA	V_{DD}- 0.3			V
I_{OZ}	Off-state Leakage Current	V_{DD} = MAX V_{IH} = V_{DD} V_{IL} = V_{SS}	-1		1	µA

7 A.C. CHARACTERISTICS

7.1 VL-Bus/Intel 486 DX Cycle Timing

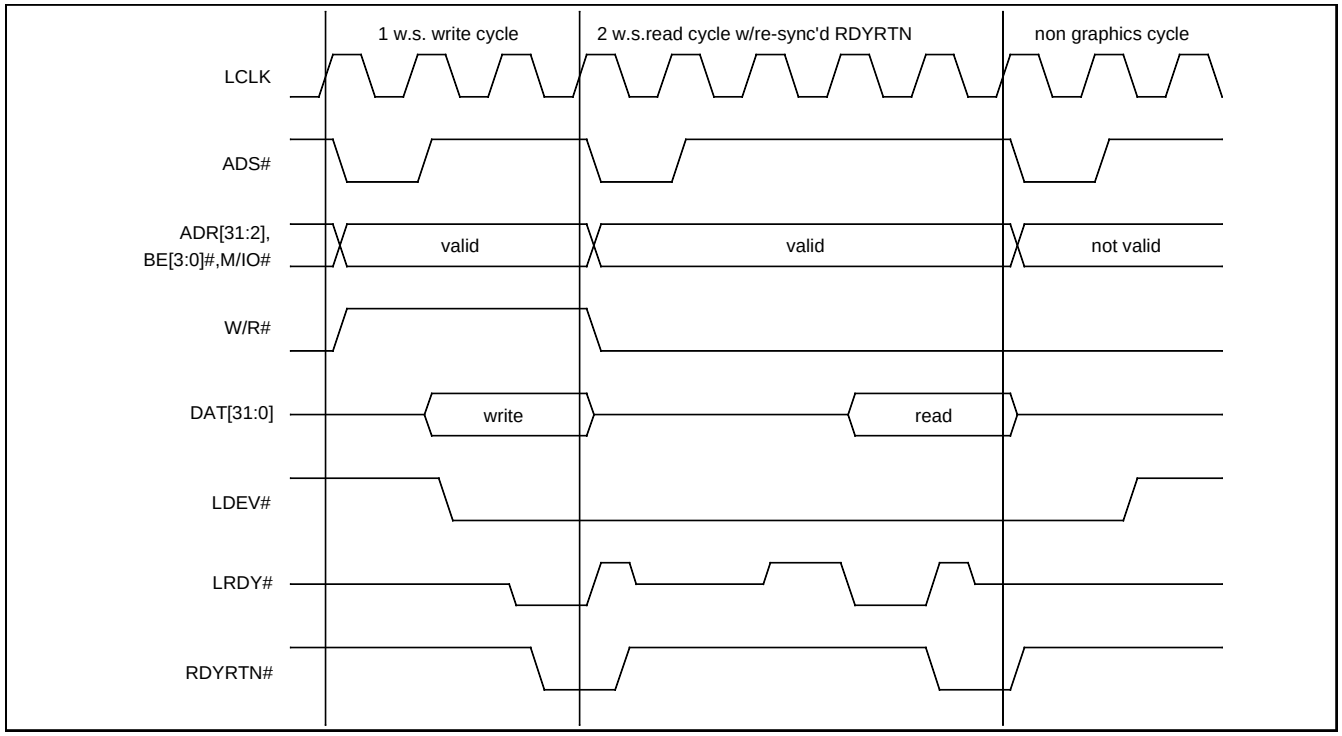


Figure 6: VL-Bus/Intel 486 DX Timing

7.2 VL-Bus/Intel 486 DX Input/Output Timing

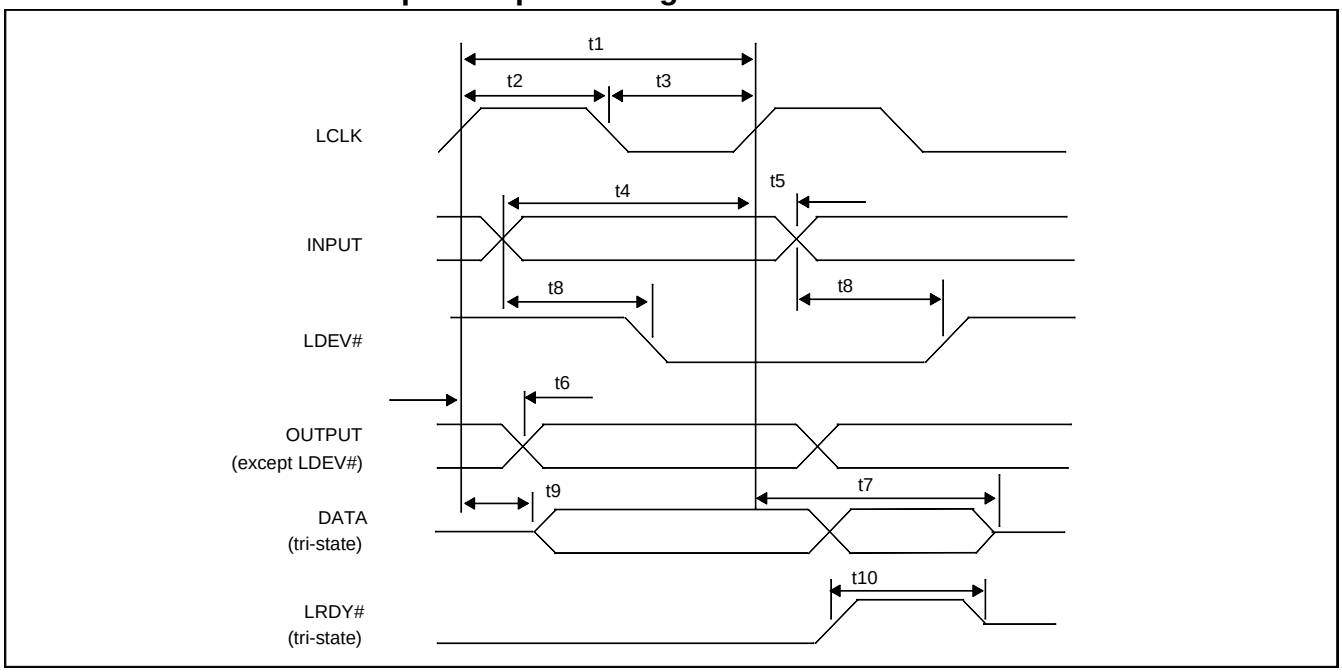


Figure 7: VL-Bus/Intel 486 DX Input/Output Timing

Table 7-1: VL-Bus/Intel486 DX Input/Output Timing

Symbol	Parameter	Min	Typ	Max	Units
t1	LCLK period	30			ns
t2	LCLK high period	0.4 * t1		0.6 * t1	ns
t3	LCLK low period	0.4 * t1		0.6 * t1	ns
t4	Input setup	7			ns
t5	Input hold	2			ns
t6	Signal valid delay LRDY# setup (100 pF), 3.3V LRDY# setup (100 pF), 5V LRDY# hold (100 pF), 3.3V LRDY# hold (100 pF), 5V DAT [31:0] setup (125 pF), 3.3V DAT [31:0] setup (125 pF), 5V DAT [31:0] hold (125 pF), 3.3V DAT [31:0] hold (125 pF), 5V	 3 3 3 3		 14 14 18 18 	ns
t7	Output active to tri-state delay LRDY# 3.3V LRDY# 5V DAT[31:0] 3.3V DAT[31:0] 5V			 30 30 20 20	ns
t8	LDEV# delay (3.3V and 5V)			20	ns
t9	Tri-state to active delay	20			ns
t10	LRDY# width	9.2			ns

7.3 PCI Bus Cycle Timing

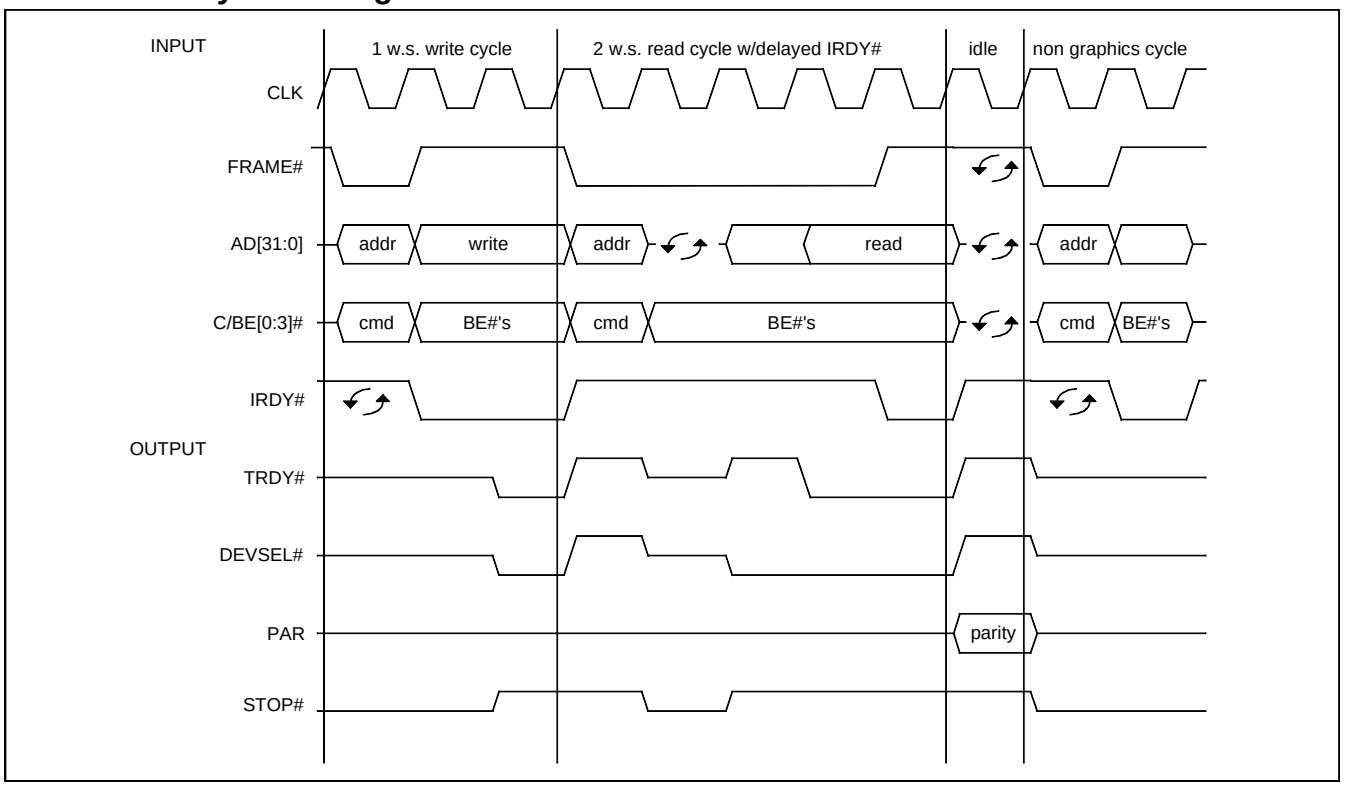


Figure 8: PCI Local Bus Timing

7.4 PCI Local Bus Disconnect Timing

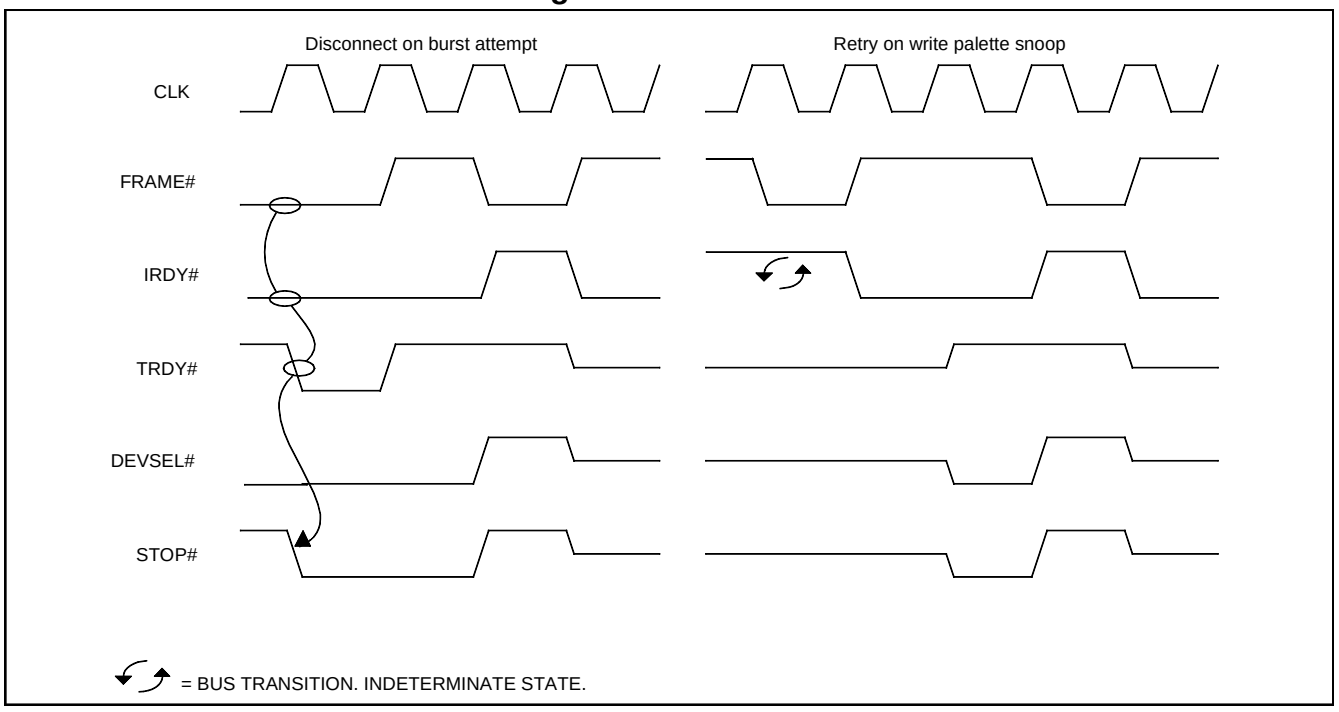


Figure 9: Disconnect Cycles

7.5 PCI Local Bus Input/Output Timing

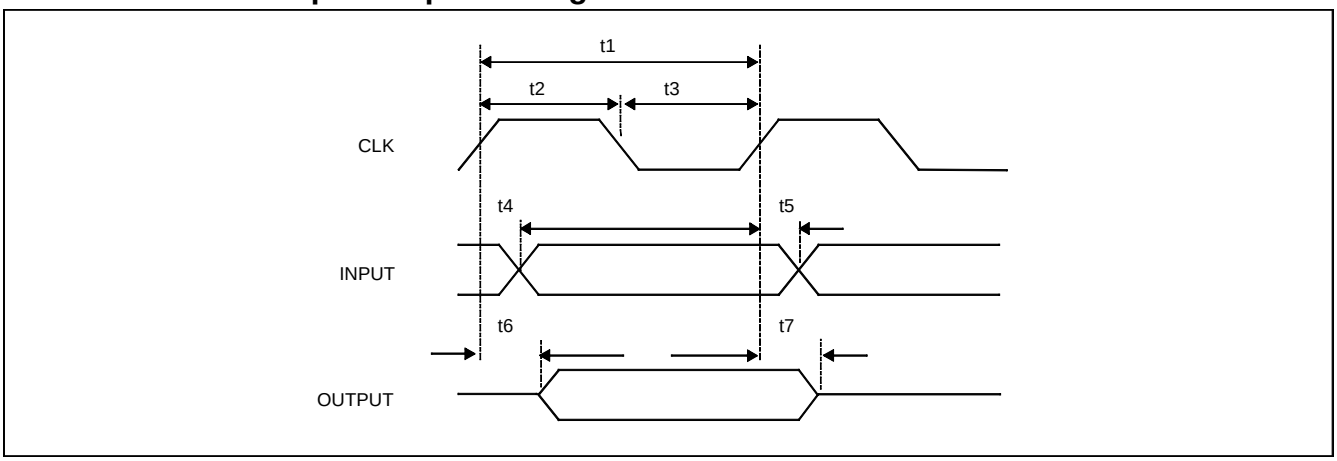


Figure 10: PCI Input/Output Timing

Table 7-2: PCI Local Bus Input/Output Timing

Symbol	Parameter	Min	Typ	Max	Units
t1	CLK period	30			ns
t2	CLK high period	0.4 * t1			ns
t3	CLK low period	0.4 * t1			ns
t4	Input setup	7			ns
t5	Input hold	0			ns
t6	Valid output delay setup (50 pF), 3.3V Valid output delay setup (50 pF), 5V Valid output delay setup (0 pF), 3.3V Valid output delay setup (0 pF), 5V	 2 2		11 11	ns
t7	Output active to float delay			28	ns

7.6 DRAM A.C. Timing

7.6.1 Read/Write Timing

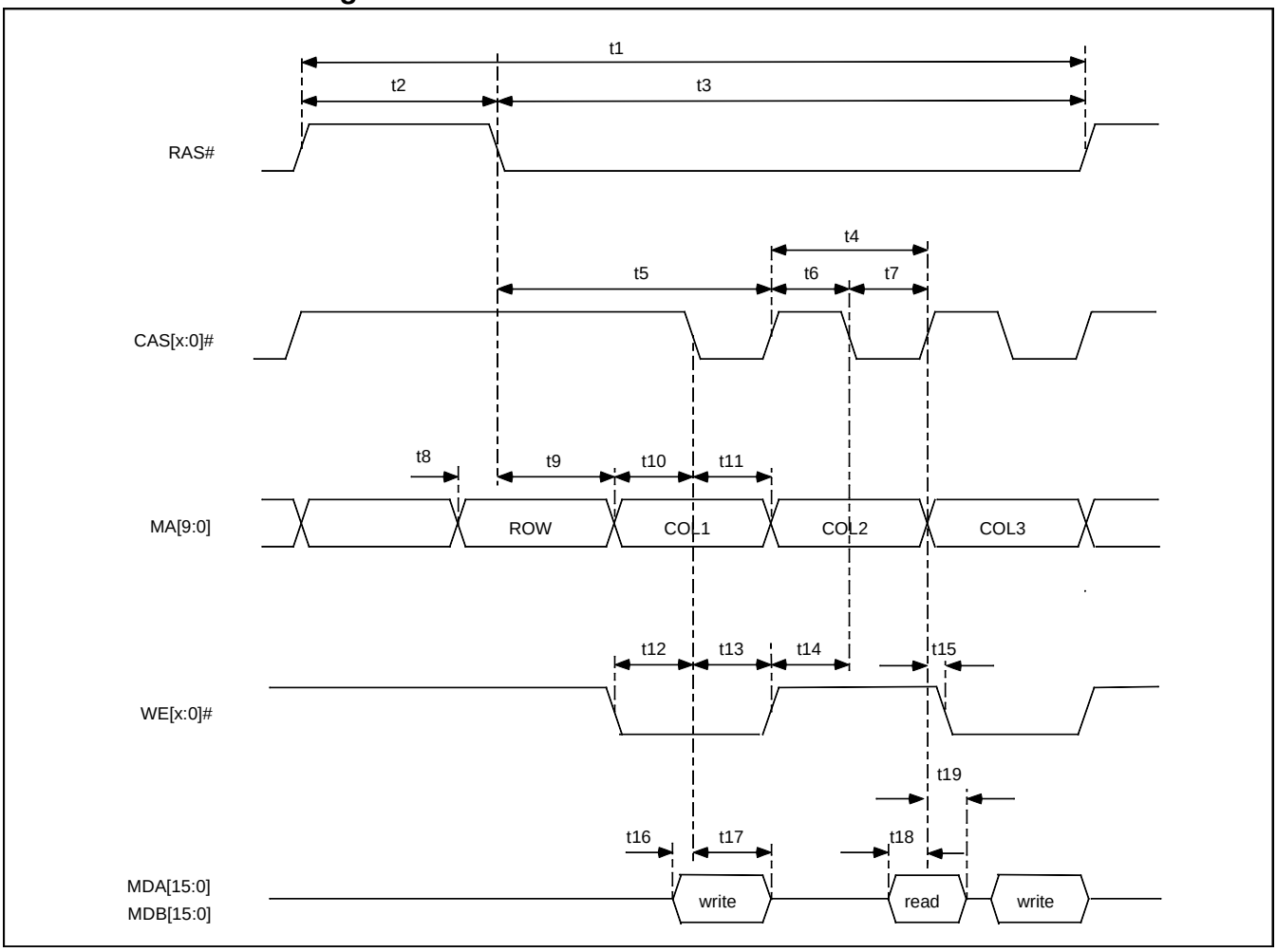


Figure 11: DRAM Write/Read Timing

Table 7-3: DRAM Write/Read Timing

Symbol	Parameter	Min	Typ	Max	Units
t1	Random read or write cycle time	6Tm			ns
t2	RAS# precharge time	2.5Tm - 2			ns
t3	RAS# pulse width	3.5Tm - 6			ns
t4	Fast page mode cycle time	2Tm - 2			ns
t5	CAS# hold time	3.5Tm - 5			ns
t6	Fast page mode CAS# precharge time	Tm - 6			ns
t7	CAS# pulse width	Tm - 2			ns
t8	Row address setup	0.5Tm - 2			ns
t9	Row address hold	1.5Tm - 10			ns
t10	Column address setup	Tm - 5			ns
t11	Column address hold	Tm - 10			ns
t12	Write command setup	0.5Tm - 2			ns
t13	Write command hold	1.5Tm - 10			ns
t14	Read command setup	0.5Tm - 2			ns
t15	Read command hold	1.5Tm - 10			ns
t16	Write data setup	1.0			ns
t17	Write data hold	Tm - 10			ns
t18	Read data setup	3.0			ns
t19	Read data hold	0			ns

7.6.2 CAS Before RAS Refresh Timing

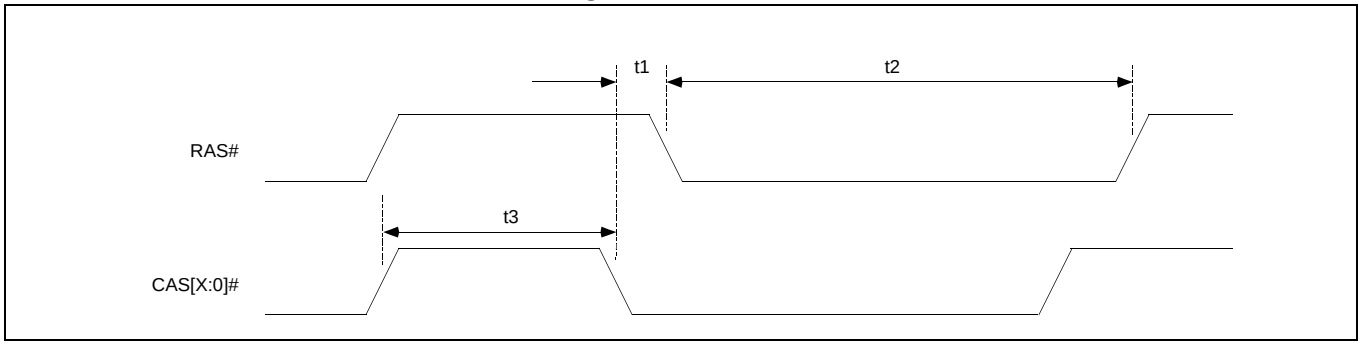


Figure 12: CAS Before RAS Refresh

Table 7-4: CAS Before RAS Refresh Timing

Symbol	Parameter	Min	Max	Units
t1	CAS# to RAS# delay	0.5Tm - 2		ns
t2	RAS# pulse width	3.5Tm - 5		ns
t3	CAS# pulse width	4Tm - 10		ns

7.6.3 Self Refresh Timing

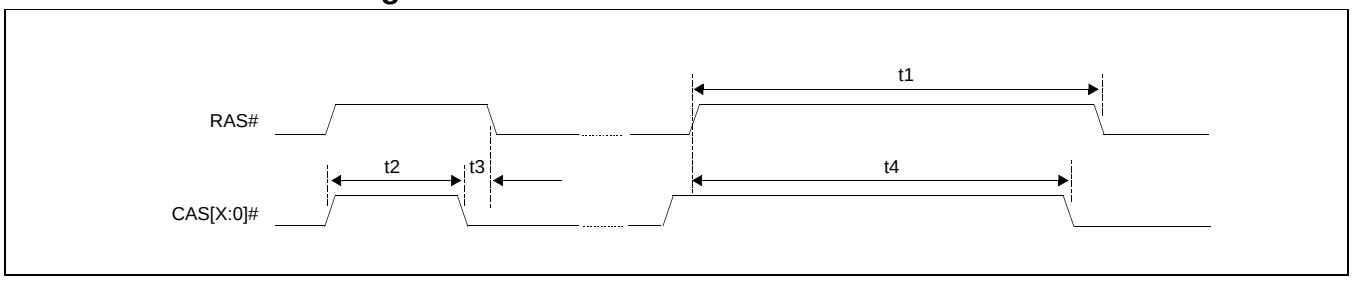


Figure 13: Self Refresh Memory Control Output

Table 7-5: Self Refresh Timing

Symbol	Parameter	Min	Max	Units
t1	RAS# precharge (leaving self refresh)	6.5Tm - 2		ns
t2	CAS# precharge	2Tm - 2		ns
t3	CAS# setup	10		ns
t4	CAS# precharge (leaving self refresh)	6Tm - 2		ns

7.6.4 Power Down Mode CAS Before RAS Refresh Timing

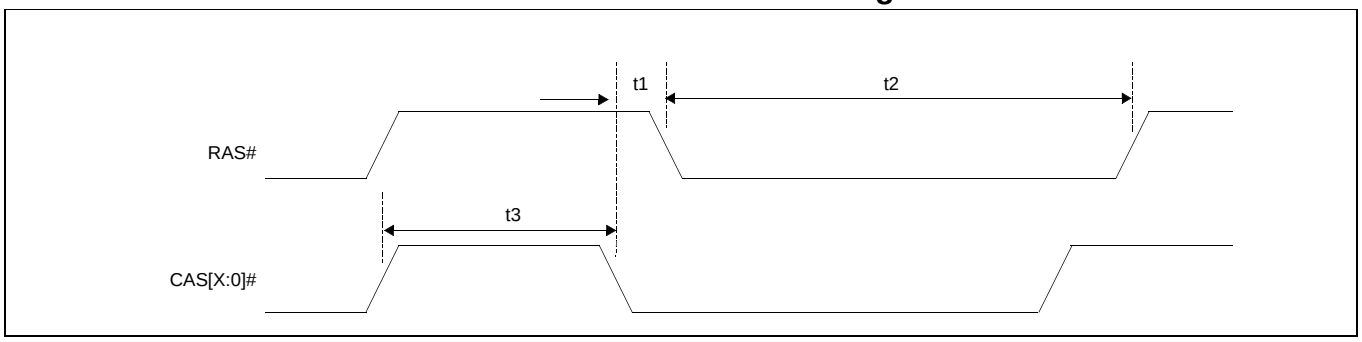


Figure 14: Delay Pulse CBR Refresh

Table 7-6: Power Down Mode CAS Before RAS Refresh Timing

Symbol	Parameter	Min	Max	Units
t1	CAS# to RAS# delay	10		ns
t2	RAS# pulse width	110		ns
t3	CAS# pulse width	110		ns

7.7 Clock Input Requirements

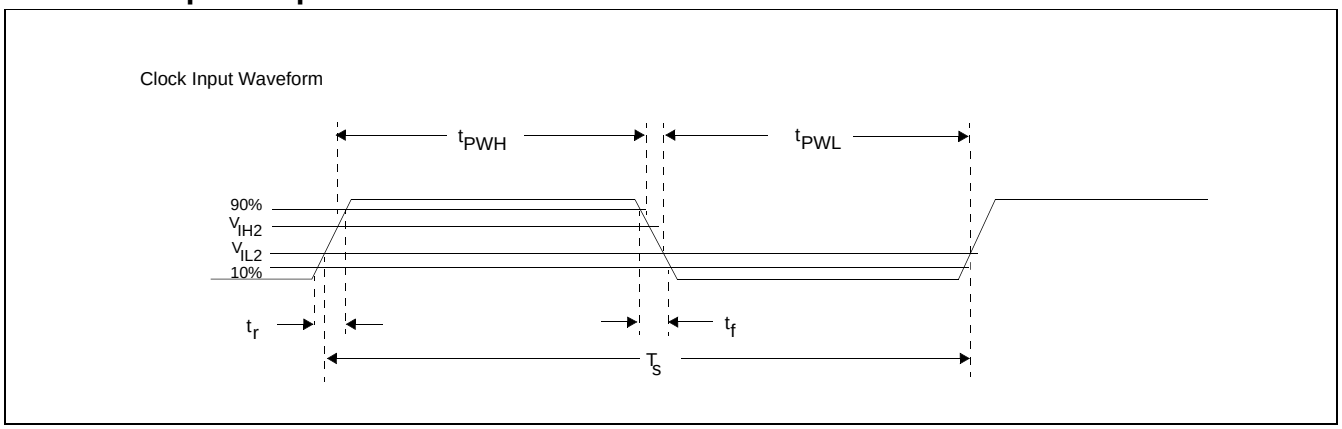


Figure 15: Clock Input Requirements

Table 7-7: Clock Input Timing Requirements

Symbol	Parameter	Min	Typ	Max	Units
T_S	Input Clock Period REFCLK		31.25		µs
	Input Clock Period RTCLK		69.8		ns
t_{PWH}	Input Clock Pulse Width High (REFCLK, RTCLK)	40		60	%Ts
t_{PWL}	Input Clock Pulse Width Low (REFCLK, RTCLK)	40		60	%Ts
t_f	Input Clock Fall Time (10% - 90%)		5		ns
t_r	Input Clock Rise Time (10% - 90%)		5		ns

7.8 LCD Panel Timing Diagrams

7.8.1 Single Monochrome 4-bit Panel Functional Timing

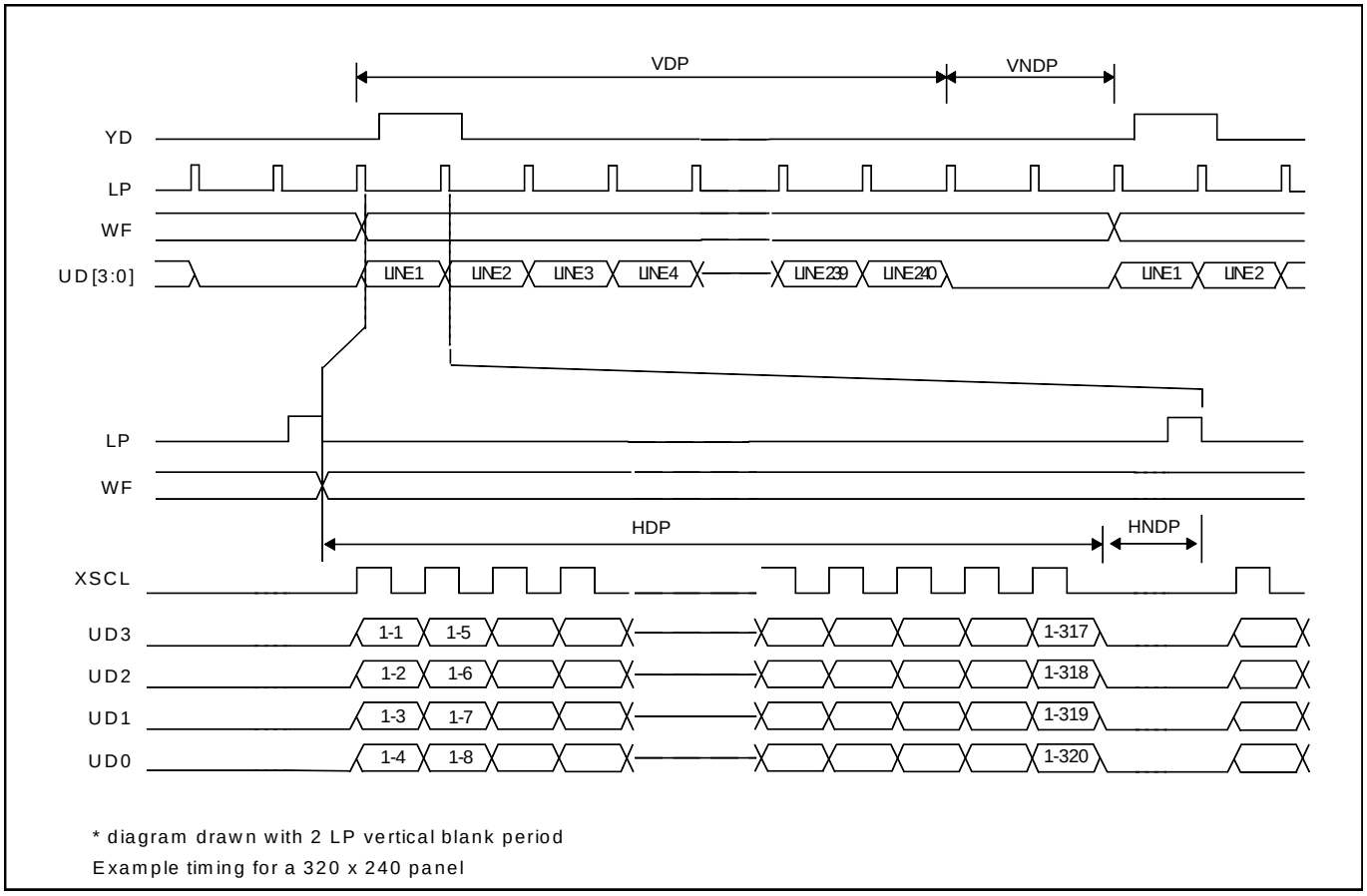


Figure 16: Single Monochrome 4-bit Panel Timing

VDP	=	Vertical Display Period:	[(CRTC B[12h]) * 4] T_{LP}
VNDP	=	Vertical Non-Display Period:	[{(CRTC B[07h] bits 5, 0), (CRTC B[06h])} + 1 - VDP] T_{LP}
HDP	=	Horizontal Display Period:	[(CRTC B[01h]) * 2] T_{XSCL}
HNDP	=	Horizontal Non-Display Period:	[(CRTC B[00h] bits 2-0) * 4] T_{XSCL}

Single Monochrome 4-bit Panel A.C. Timing

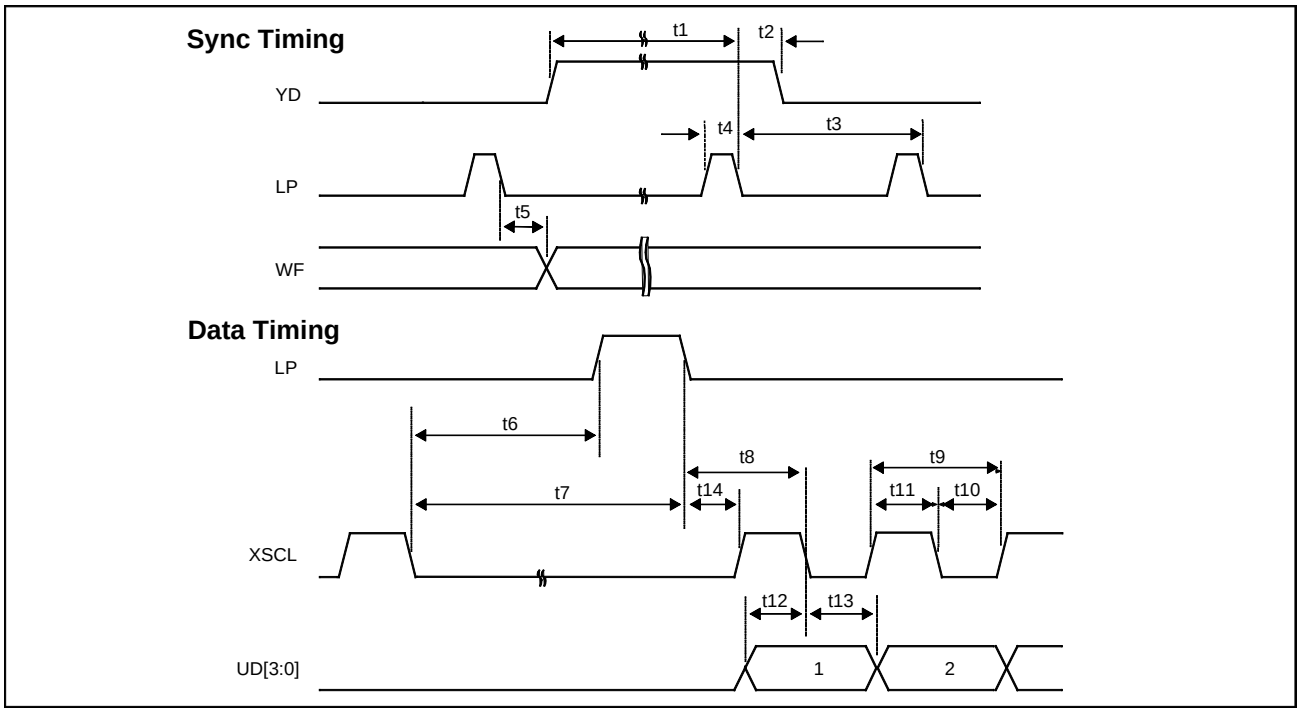


Figure 17: Single Monochrome 4-bit Panel A.C. Timing

Table 7-8: Single Monochrome 4-bit Panel A.C. Timing

Symbol	Parameter	Min	Typ	Max	Units
t1	YD setup to LP falling edge	HDP + HNDP - 10			Ts
t2	YD hold from LP falling edge	6			Ts
t3	LP period		HDP + HNDP		Ts
t4	LP pulse width	(LP_SEL + 1)			Ts
t5	WF delay from LP falling edge	0			ns
t6	XSCL falling edge to LP rising edge	(HNDP -10)			Ts
t7	XSCL falling edge to LP falling edge	(HNDP - 9+ LP_SEL)			Ts
t8	LP falling edge to XSCL falling edge	(13 - LP_SEL)			Ts
t9	XSCL period	4			Ts
t10	XSCL pulse width low	2			Ts
t11	XSCL pulse width high	2			Ts
t12	UD[3:0] setup to XSCL falling edge	2			Ts
t13	UD[3:0] hold to XSCL falling edge	2			Ts
t14	LP falling edge to XSCL rising edge	(11 - LP_SEL)			Ts

Ts	=	Pixel Clock Period:	35 ns typical
HDP	=	Horizontal Display Period:	[(CRTC B[01h]) * 8] T_S
HNDP	=	Horizontal Non-Display Period:	[(CRTC B[00h] bits 2-0) *16] T_S
LP_SEL[5:0]	=	LP Pulse Width - 1:	[(AUX[02h] bits 7-5)] T_S

7.8.2 Single Monochrome 8-bit Panel Timing

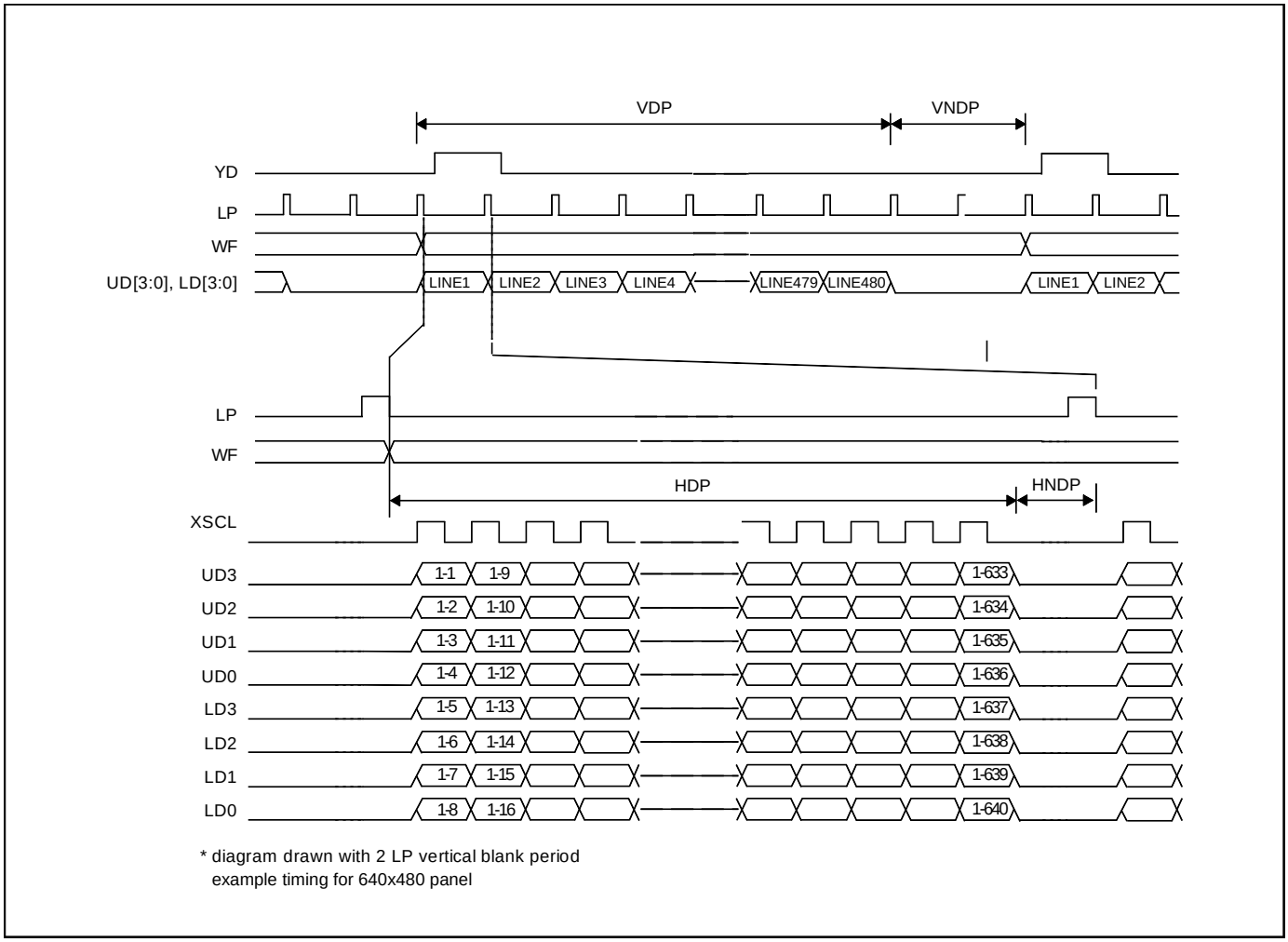


Figure 18: Single Monochrome 8-bit Panel Timing

VDP	=	Vertical Display Period:	[(CRTC B[12h]) * 4] T_{LP}
VNDP	=	Vertical Non-Display Period:	[{(CRTC B[07h] bits 5, 0), (CRTC B[06h])} + 1 - VDP] T_{LP}
HDP	=	Horizontal Display Period:	[(CRTC B[01h])] T_{XSCL}
HNDP	=	Horizontal Non-Display Period:	[(CRTC B[00h] bits 2-0) * 2] T_{XSCL}

Single Monochrome 8-bit Panel A.C. Timing

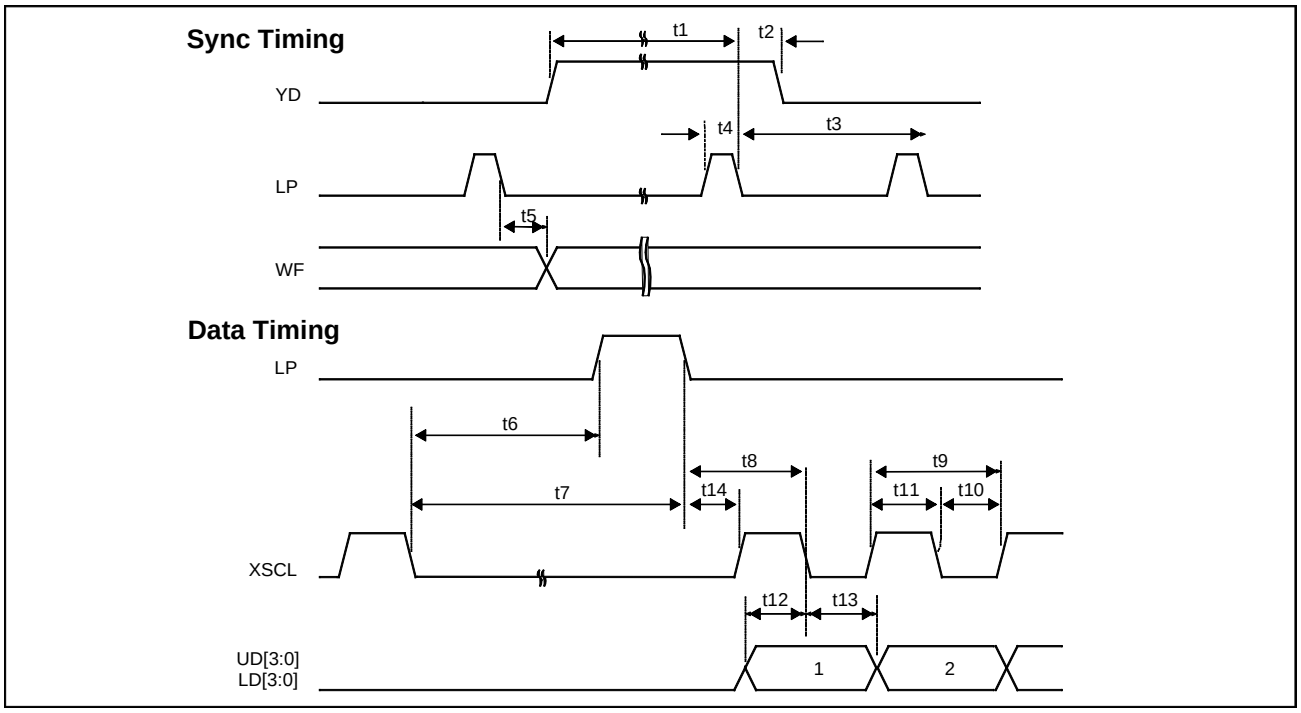


Figure 19: Single Monochrome 8-bit Panel A.C. Timing

Table 7-9: Single Monochrome 8-bit Panel A.C. Timing

Symbol	Parameter	Min	Typ	Max	Units
t1	YD setup to LP falling edge	HDP + HNDP - 10			Ts
t2	YD hold from LP falling edge	6			Ts
t3	LP period		HDP + HNDP		Ts
t4	LP pulse width	(LP_SEL + 1)			Ts
t5	WF delay from LP falling edge	0			ns
t6	XSCL falling edge to LP rising edge	(HNDP - 10)			Ts
t7	XSCL falling edge to LP falling edge	(HNDP - 7+ LP_SEL)			Ts
t8	LP falling edge to XSCL falling edge	(15 - LP_SEL)			Ts
t9	XSCL period	8			Ts
t10	XSCL pulse width low	4			Ts
t11	XSCL pulse width high	4			Ts
t12	UD[3:0], LD[3:0] setup to XSCL falling edge	4			Ts
t13	UD[3:0], LD[3:0] hold to XSCL falling edge	4			Ts
t14	LP falling edge to XSCL rising edge	(11 - LP_SEL)			Ts

Ts	=	Pixel Clock Period:	35 ns typical
HDP	=	Horizontal Display Period:	[(CRTC B[01h]) * 8] T_S
HNDP	=	Horizontal Non-Display Period:	[(CRTC B[00h] bits 2-0) *16] T_S
LP_SEL[5:0]	=	LP Pulse Width - 1:	[(AUX[02h] bits 7-5)] T_S

7.8.3 Single Color 8-bit Panel Timing

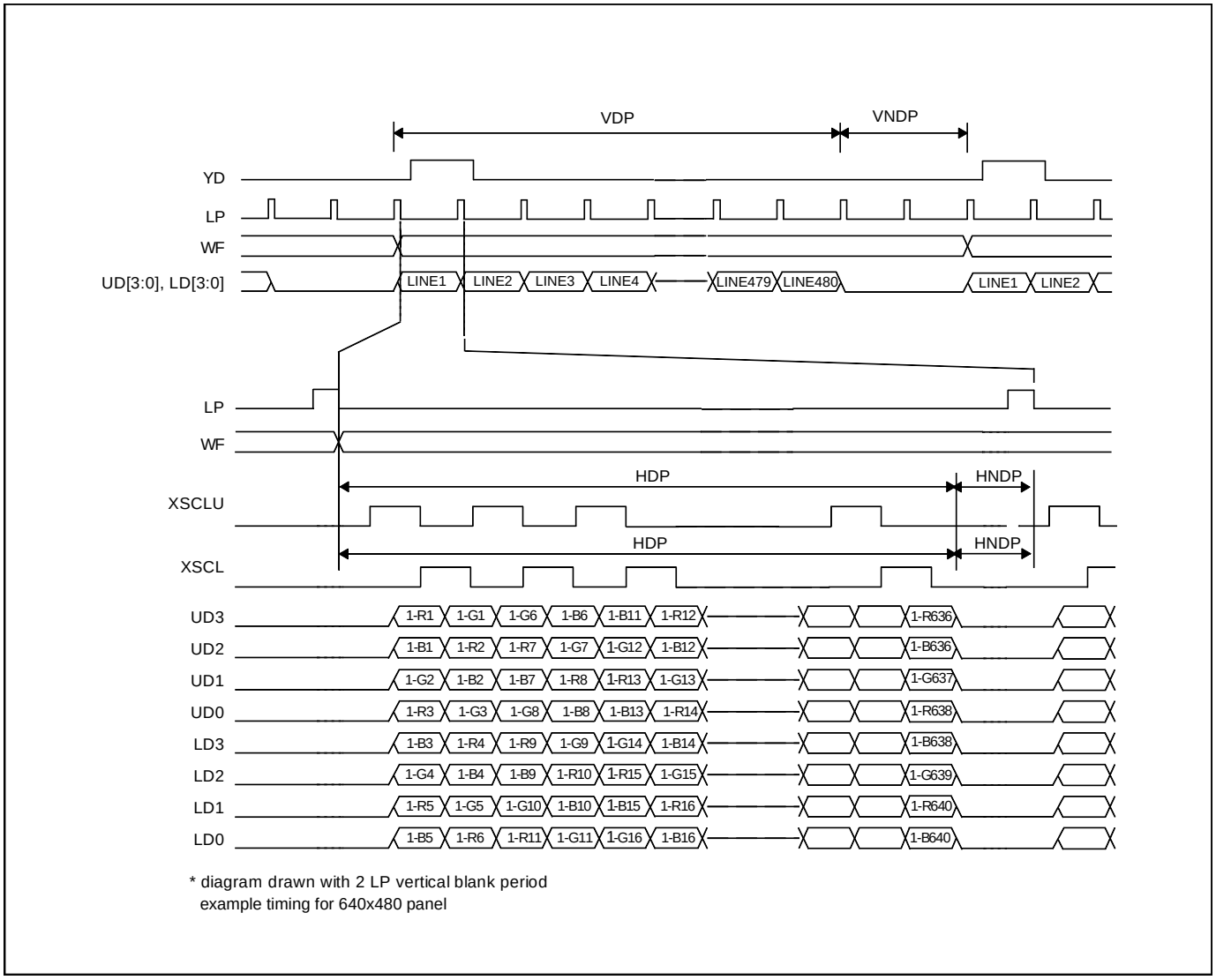


Figure 20: Single Color 8-bit Panel Timing

VDP	=	Vertical Display Period:	[(CRTC B[12h]) * 4] T_{LP}
VNDP	=	Vertical Non-Display Period:	[{(CRTC B[07h] bits 5, 0), (CRTC B[06h])} + 1 - VDP] T_{LP}
HDP	=	Horizontal Display Period:	[(CRTC B[01h]) * 3 / 2] T_{XSCL}, T_{XSCLU}
HNDP	=	Horizontal Non-Display Period:	[(CRTC B[00h] bits 2-0) * 3] T_{XSCL}, T_{XSCLU}

Single Color 8-bit Panel A.C. Timing

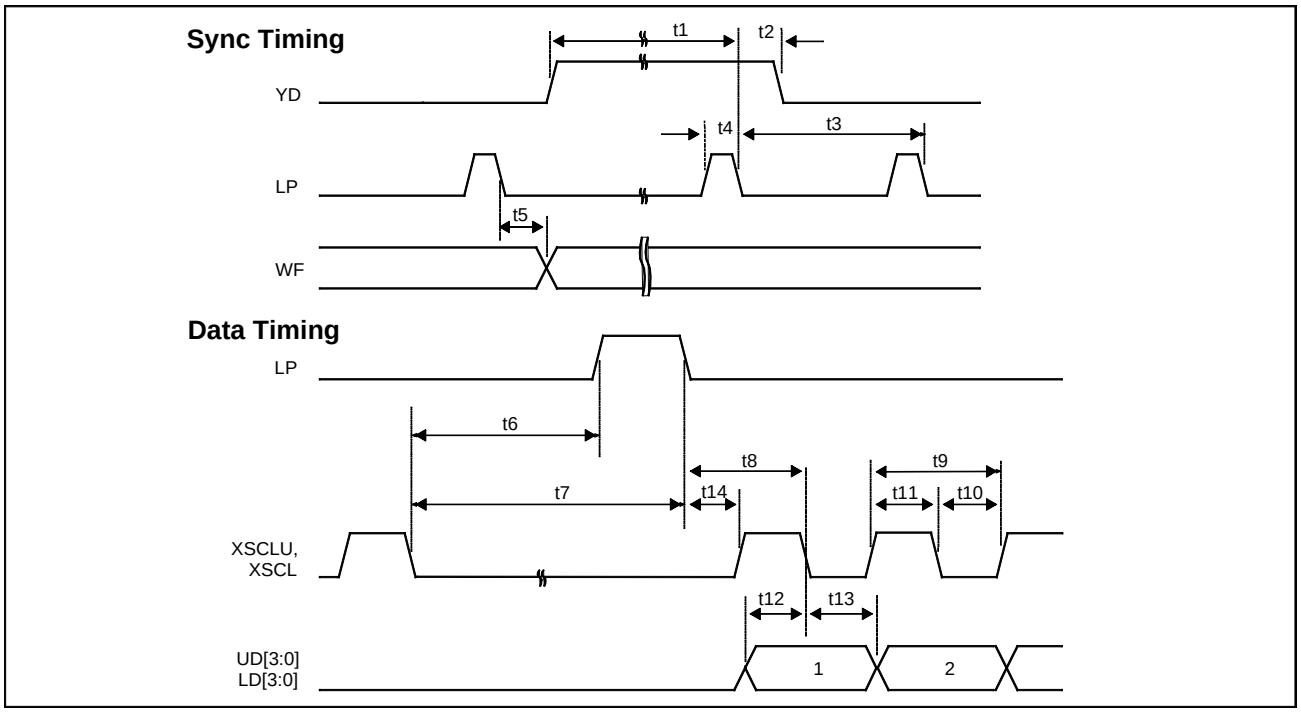


Figure 21: Single Color 8-bit Panel A.C. Timing

Table 7-10: Single Color 8-bit Panel A.C. Timing

Symbol	Parameter	Min	Typ	Max	Units
t1	YD setup to LP falling edge	HDP + HNDP - 10			Ts
t2	YD hold from LP falling edge	6			Ts
t3	LP period		HDP + HNDP		Ts
t4	LP pulse width	(LP_SEL + 1)			Ts
t5	WF delay from LP falling edge	0			ns
t6	XSCLU, XSCL falling edge to LP rising edge	(HNDP - 15)			Ts
t7	XSCLU, XSCL falling edge to LP falling edge	(HNDP - 14 + LP_SEL)			Ts
t8	LP falling edge to XSCLU, XSCL falling edge	(20 - LP_SEL)			Ts
t9	XSCLU, XSCL period	4			Ts
t10	XSCLU, XSCL pulse width low	2			Ts
t11	XSCLU, XSCL pulse width high	2			Ts
t12	UD[3:0], LD[3:0] setup to XSCLU, XSCL falling edge	1			Ts
t13	UD[3:0], LD[3:0] hold to XSCLU, XSCL falling edge	1			Ts
t14	LP falling edge to XSCLU, XSCL rising edge	(18 - LP_SEL)			Ts

Ts	=	Pixel Clock Period:	35 ns typical
HDP	=	Horizontal Display Period :	[(CRTC B[01h]) * 8] T_S
HNDP	=	Horizontal Non-Display Period:	[(CRTC B[00h] bits 2-0) *16] T_S
LP_SEL[5:0	=	LP Pulse Width - 1:	[(AUX[02h] bits 7-5)] T_S

7.8.4 Single Color 8-bit Panel Timing (Alternate Data Format)

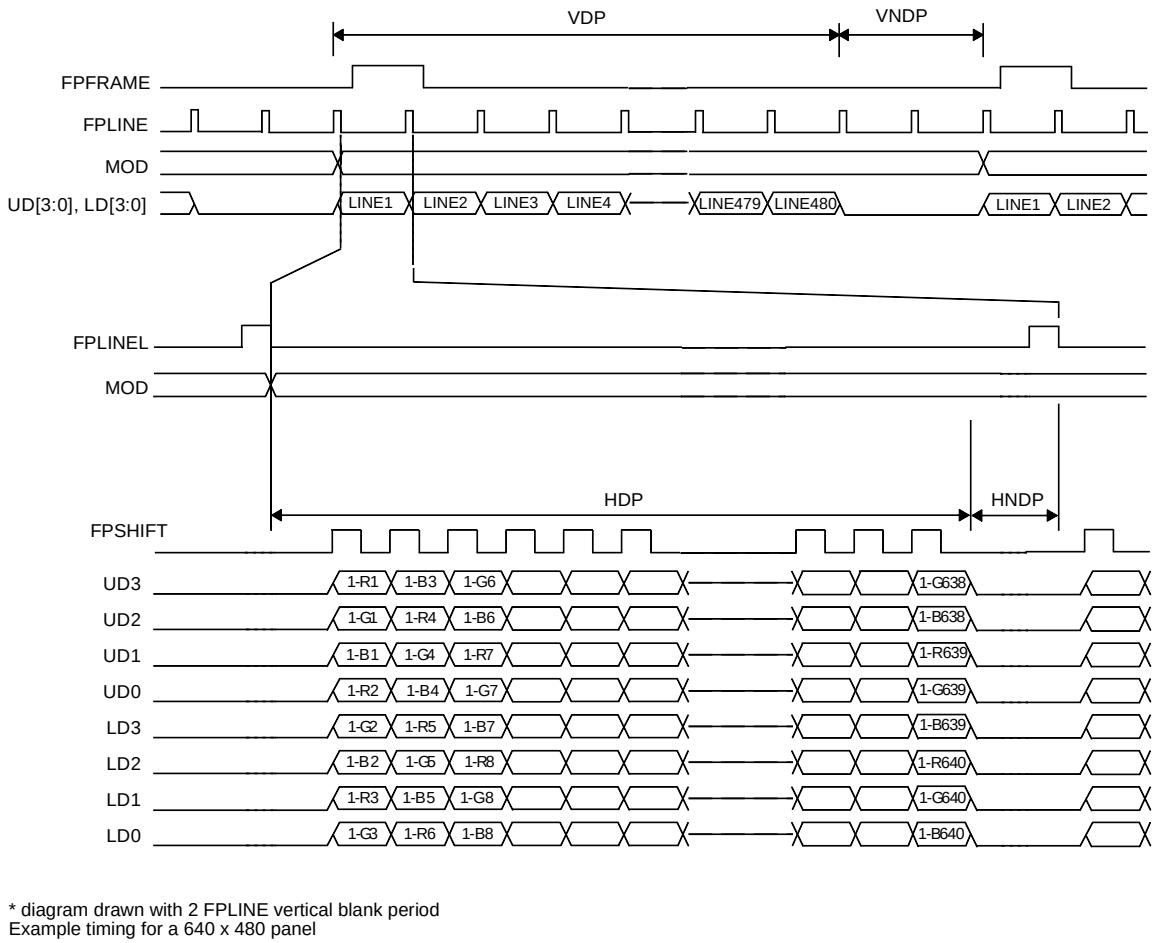


Figure 22: Single Color 8-bit Panel Timing (Alternate Data Format)

VDP	=	Vertical Display Period:	[(CRTC B[12h]) * 4] T_{LP}
VNDP	=	Vertical Non-Display Period:	[{(CRTC B[07h] bits 5, 0), (CRTC B[06h])} + 1 - VDP] T_{LP}
HDP	=	Horizontal Display Period:	[(CRTC B[01h]) * 3 / 2] T_{XSCL}, T_{XSCLU}
HNDP	=	Horizontal Non-Display Period:	[(CRTC B[00h] bits 2-0) * 3] T_{XSCL}, T_{XSCLU}

Single Color 8-bit Panel A.C. Timing (Alternate Data Format)

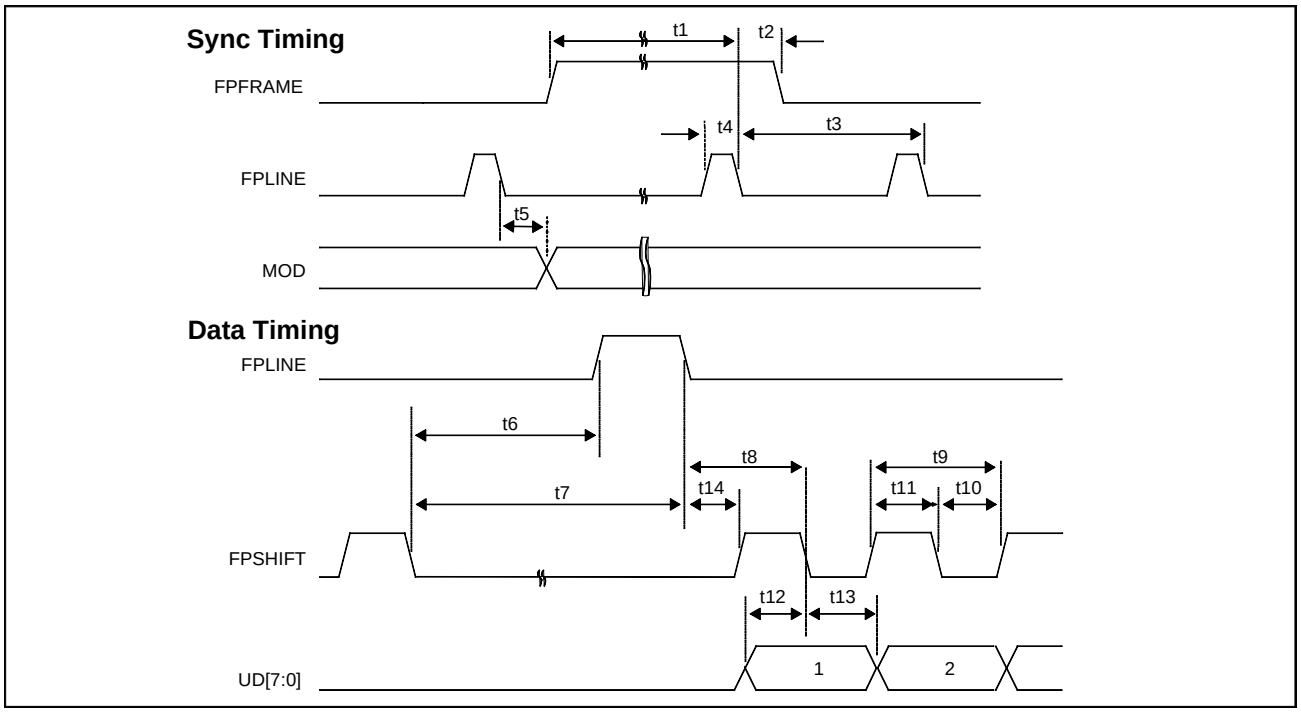


Figure 23: Single Color 8-bit Panel A.C. Timing (Alternate Data Format)

Table 7-11: Single Color 8-bit Panel A.C. Timing (Alternate Data Format)

Symbol	Parameter	Min	Typ	Max	Units
t1	FPFRAME setup to FPLINE falling edge	HDP + HNDP - 10			Ts
t2	FPFRAME hold from FPLINE falling edge	6			Ts
t3	FPLINE period		HDP + HNDP		Ts
t4	FPLINE pulse width	(LP_SEL + 1)			Ts
t5	MOD delay from FPLINE falling edge	0			ns
t6	FPSHIFT falling edge to FPLINE rising edge	(HNDP - 9)			Ts
t7	FPSHIFT falling edge to FPLINE falling edge	(HNDP - 8 + LP_SEL)			Ts
t8	FPLINE falling edge to FPSHIFT falling edge	(12 - LP_SEL)			Ts
t9	FPSHIFT period	2			Ts
t10	FPSHIFT pulse width low	1			Ts
t11	FPSHIFT pulse width high	1			Ts
t12	UD[3:0], LD[3:0] setup to FPSHIFT falling edge	1			Ts
t13	UD[3:0], LD[3:0] hold to FPSHIFT falling edge	1			Ts
t14	FPLINE falling edge to FPSHIFT rising edge	(11 - LP_SEL)			Ts

Ts	=	Pixel Clock Period:	35 ns typical
HDP	=	Horizontal Display Period:	[(CRTC B[01h]) * 8] T_S
HNDP	=	Horizontal Non-Display Period:	[(CRTC B[00h] bits 2-0) *16] T_S
LP_SEL[5:0]	=	LP Pulse Width - 1:	[(AUX[02h] bits 7-5)] T_S

7.8.5 Single Color 16-bit Panel Timing

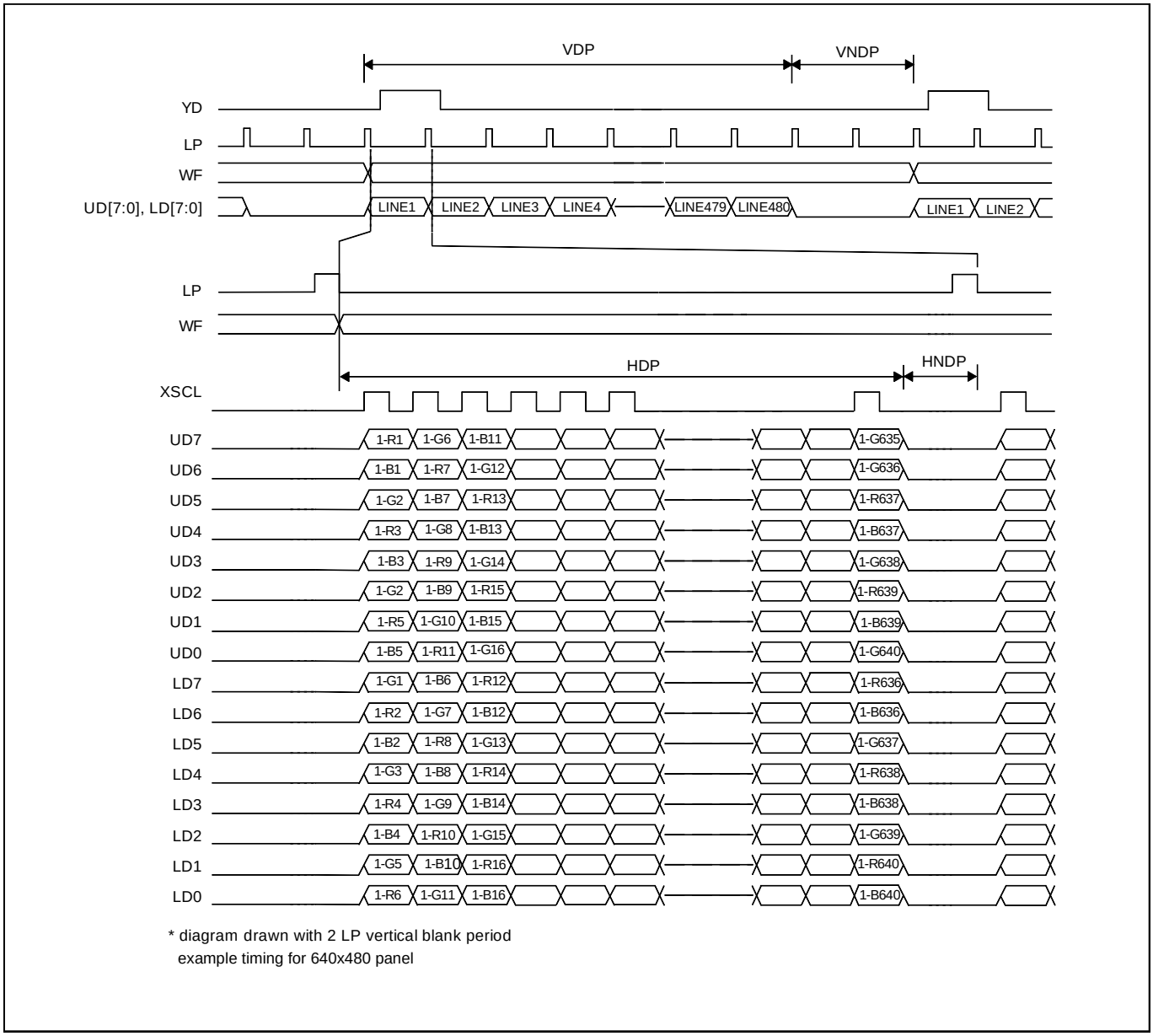


Figure 24: Single Color 16-bit Panel Timing

VDP	=	Vertical Display Period:	[(CRTC B[12h]) * 4] T_{LP}
VND	=	Vertical Non-Display Period:	[{(CRTC B[07h] bits 5, 0), (CRTC B[06h])} + 1 - VDP] T_{LP}
HDP	=	Horizontal Display Period:	[(CRTC B[01h]) * 3 / 2] T_{XSCL}, T_{XSCLU}
HNDP	=	Horizontal Non-Display Period:	[(CRTC B[00h] bits 2-0) * 3] T_{XSCL}, T_{XSCLU}

Single Color 16-bit Panel A.C. Timing

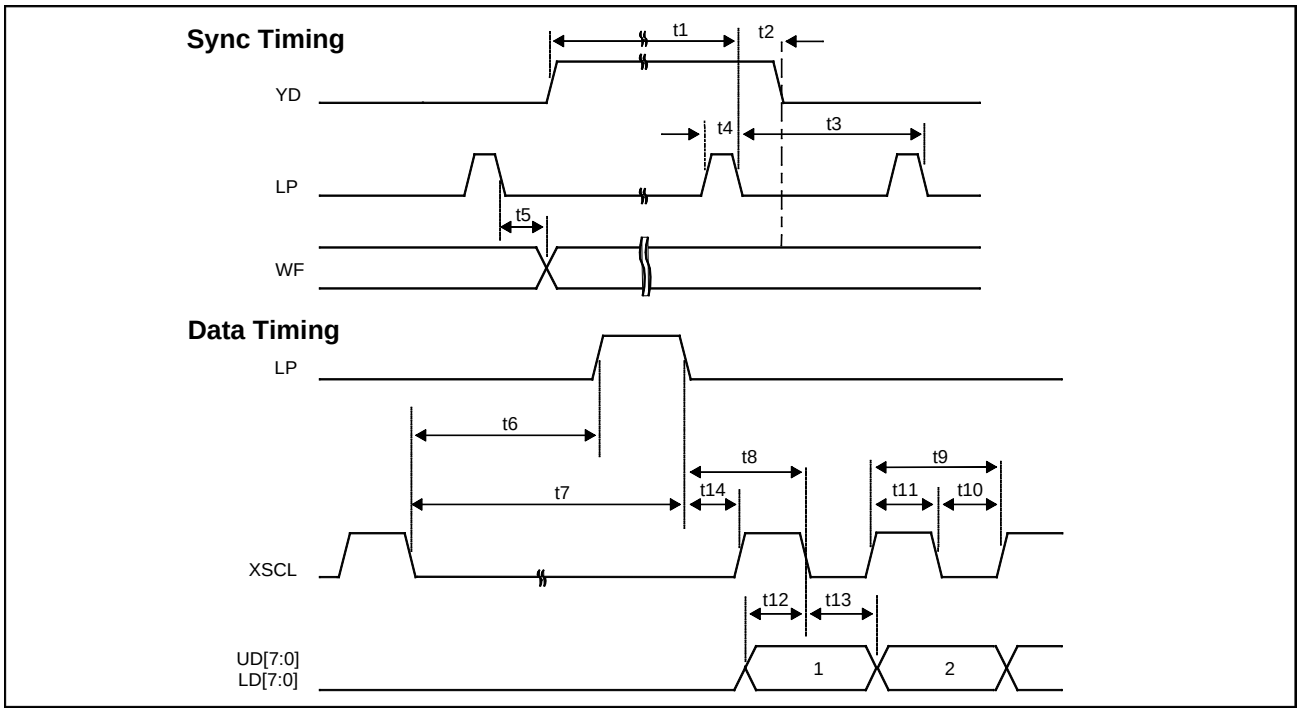


Figure 25: Single Color 16-bit Panel A.C. Timing

Table 7-12: Single Color 16-bit Panel A.C. Timing

Symbol	Parameter	Min	Typ	Max	Units
t1	YD setup to LP falling edge	HDP + HNDP - 10			Ts
t2	YD hold from LP falling edge	6			Ts
t3	LP period		HDP + HNDP		Ts
t4	LP pulse width	(LP_SEL + 1)			Ts
t5	WF delay from LP falling edge	0			ns
t6	XSCL falling edge to LP rising edge	(HNDP - 17)			Ts
t7	XSCL falling edge to LP falling edge	(HNDP - 16 + LP_SEL)			Ts
t8	LP falling edge to XSCL falling edge	(22 - LP_SEL)			Ts
t9	XSCL period	5			Ts
t10	XSCL pulse width low	3			Ts
t11	XSCL pulse width high	2			Ts
t12	UD[7:0], LD[7:0] setup to XSCL falling edge	3			Ts
t13	UD[7:0], LD[7:0] hold to XSCL falling edge	2			Ts
t14	LP falling edge to XSCL rising edge	(20 - LP_SEL)			Ts

Ts	=	Pixel Clock Period:	35 ns typical
HD	=	Horizontal Display Period:	[(CRTC B[01h]) * 8] T_S
HNDP	=	Horizontal Non-Display Period:	[(CRTC B[00h] bits 2-0) *16] T_S
LP_SEL[5:0]	=	LP Pulse Width - 1:	[(AUX[02h] bits 7-5)] T_S

7.8.6 Dual Monochrome 8-bit Panel Timing

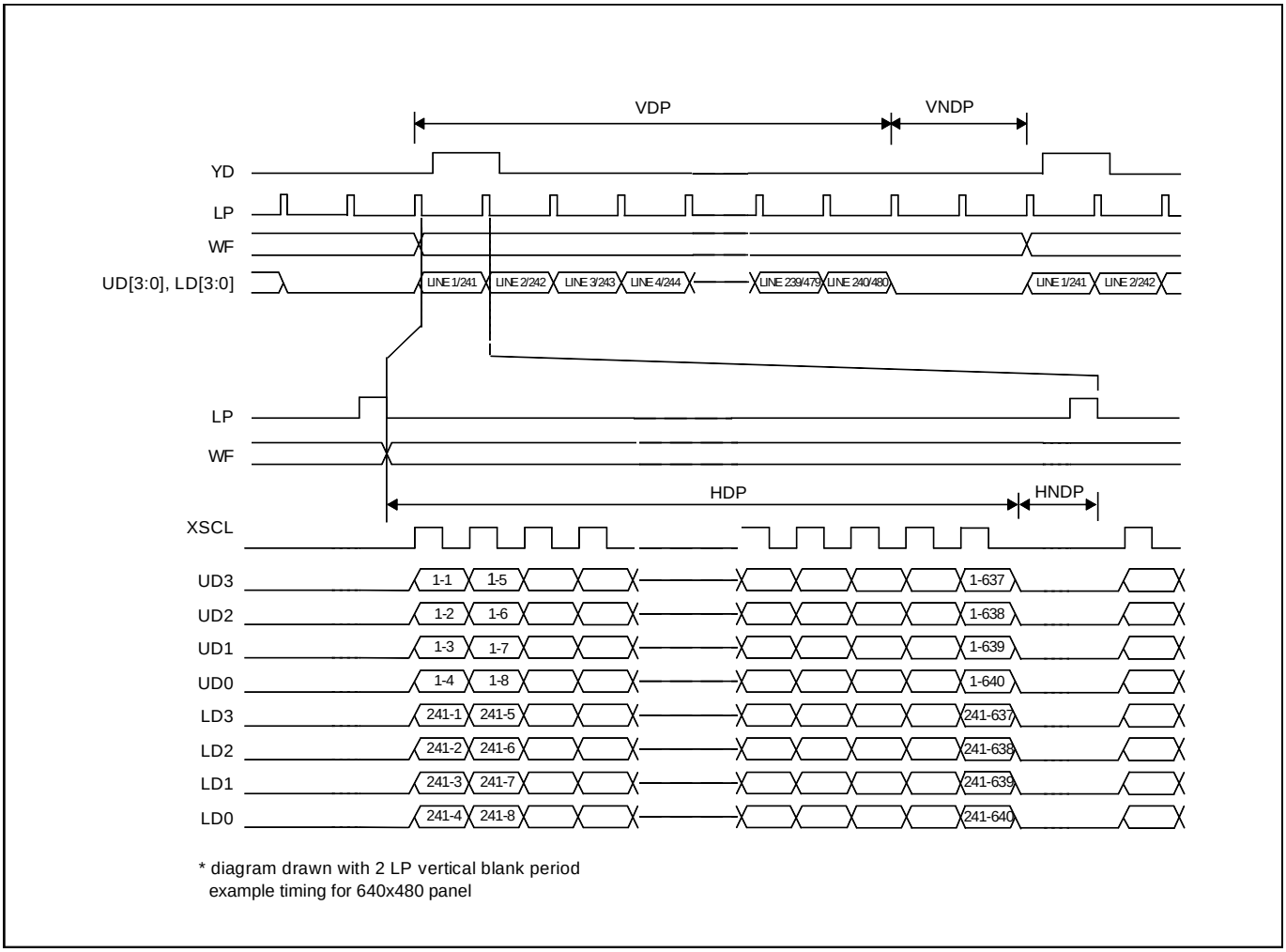


Figure 26: Dual Monochrome 8-bit Panel Timing

VDP	=	Vertical Display Period:	[(CRTC B[12h]) * 4] T_{LP}
VNDP	=	Vertical Non-Display Period:	[{(CRTC B[07h] bits 5, 0), (CRTC B[06h])} + 1 - VDP] T_{LP}
HDP	=	Horizontal Display Period:	[(CRTC B[01h]) * 2] T_{XSCL}
HNDP	=	Horizontal Non-Display Period:	[(CRTC B[00h] bits 2-0) * 4] T_{XSCL}

Dual Monochrome 8-bit Panel A.C. Timing

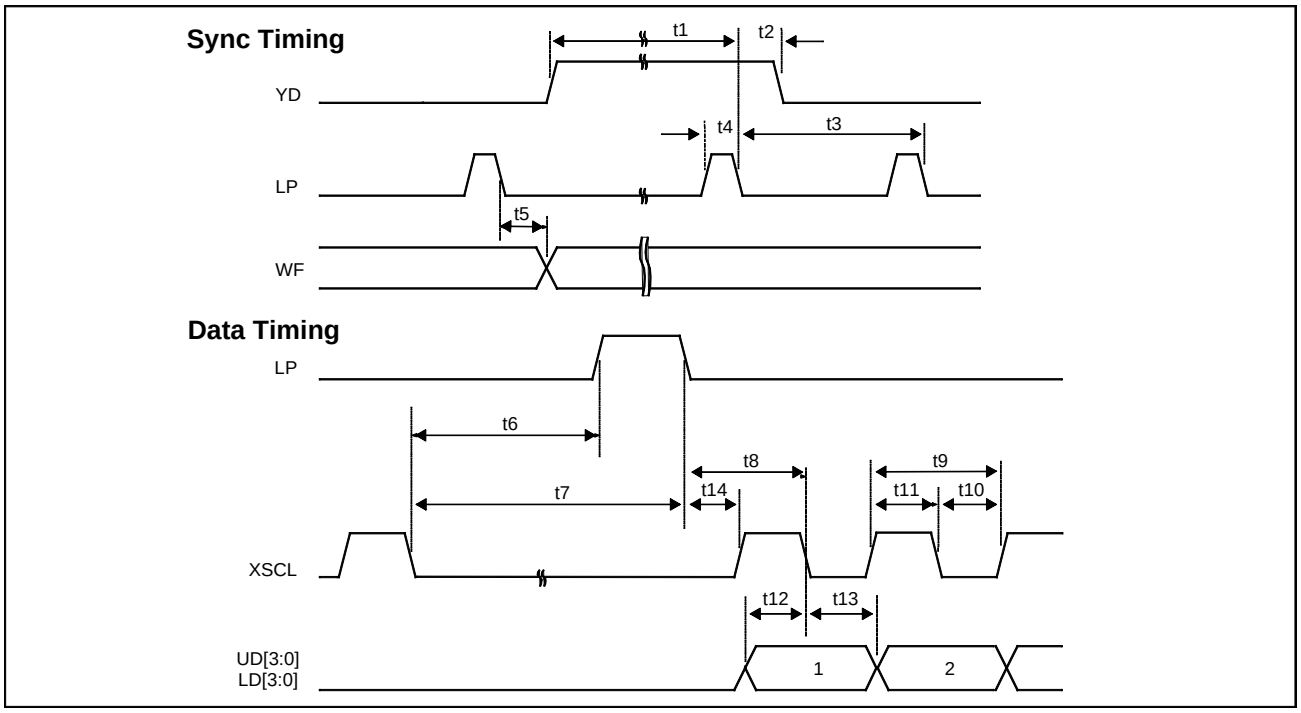


Figure 27: Dual Monochrome 8-bit Panel A.C. Timing

Table 7-13: Dual Monochrome 8-bit Panel A.C. Timing

Symbol	Parameter	Min	Typ	Max	Units
t1	YD setup to LP falling edge	HDP + HNDP - 10			Ts
t2	YD hold from LP falling edge	6			Ts
t3	LP period		HDP + HNDP		Ts
t4	LP pulse width	(LP_SEL + 1)			Ts
t5	WF delay from LP falling edge	0			ns
t6	XSCL falling edge to LP rising edge	(HNDP - 10)			Ts
t7	XSCL falling edge to LP falling edge	(HNDP - 9+ LP_SEL)			Ts
t8	LP falling edge to XSCL falling edge	(13 - LP_SEL)			Ts
t9	XSCL period	4			Ts
t10	XSCL pulse width low	2			Ts
t11	XSCL pulse width high	2			Ts
t12	UD[3:0], LD[3:0] setup to XSCL falling edge	2			Ts
t13	UD[3:0], LD[3:0] hold to XSCL falling edge	2			Ts
t14	LP falling edge to XSCL rising edge	(11 - LP_SEL)			Ts

Ts	=	Pixel Clock Period:	70 ns typical
HDP	=	Horizontal Display Period:	[(CRTC B[01h]) * 8] T_S
HNDP	=	Horizontal Non-Display Period:	[(CRTC B[00h] bits 2-0) *16] T_S
LP_SEL[5:0]	=	LP Pulse Width - 1:	[(AUX[02h] bits 7-5)] T_S

7.8.7 Dual Color 6-bit Panel Timing

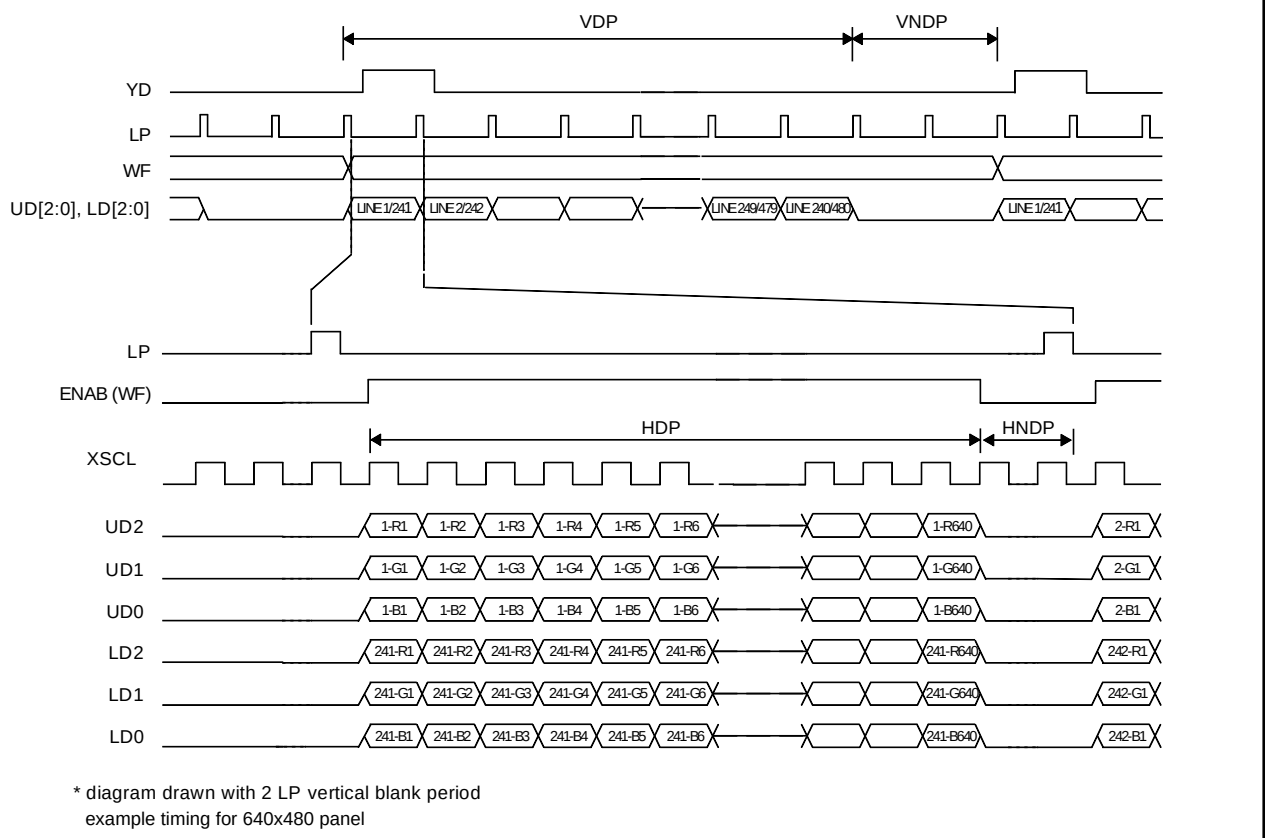


Figure 28: Dual Color 6-bit Panel Timing

VD	=	Vertical Display Period:	[(CRTC B[12h]) * 4] T_{LP}
VNDP	=	Vertical Non-Display Period:	[{(CRTC B[07h] bits 5, 0), (CRTC B[06h])} + 1 - VDP] T_{LP}
HDP	=	Horizontal Display Period:	[(CRTC B[01h]) * 8] T_{XSCL}
HND	=	Horizontal Non-Display Period:	[(CRTC B[00h] bits 2-0) *16] T_{XSCL}

Dual Color 6-bit Panel A.C. Timing (AUX[07h] bit 4 = 0, normal WF function)

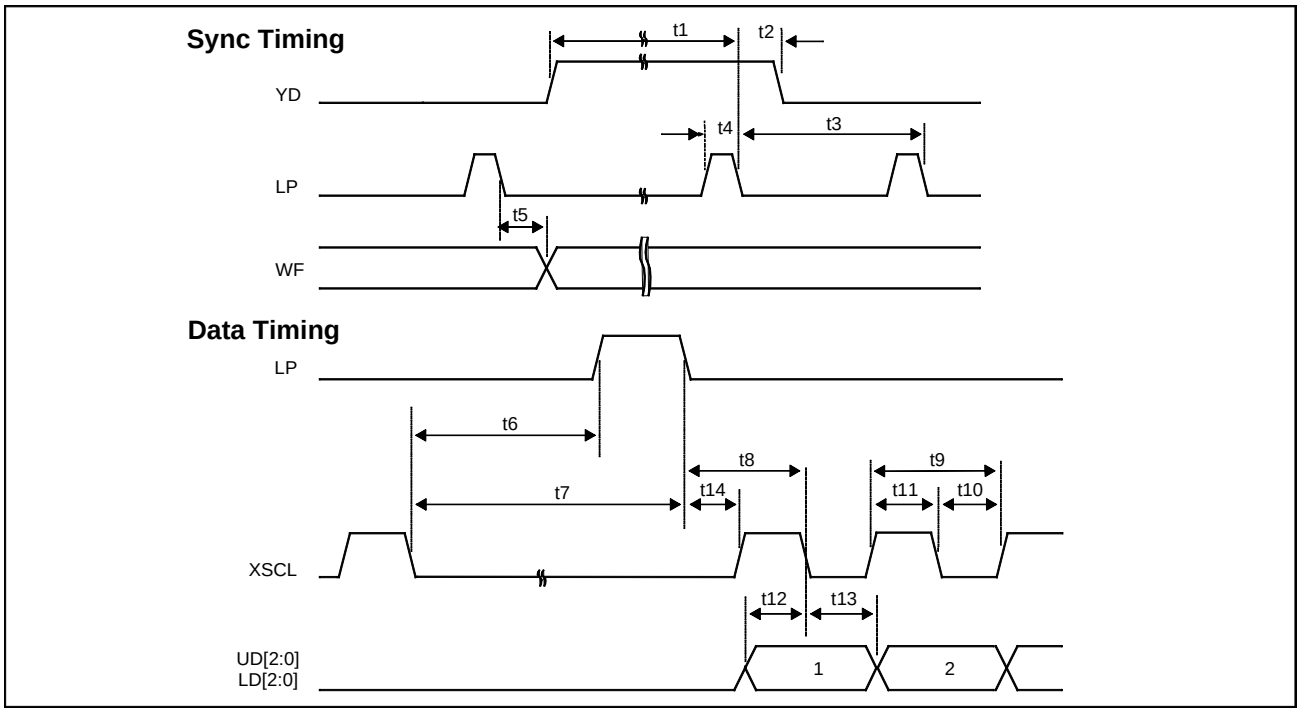


Figure 29: Dual Color 6-bit Panel A.C. Timing

Table 7-14: Dual Color 6-bit Panel A.C. Timing

Symbol	Parameter	Min	Typ	Max	Units
t1	YD setup to LP falling edge	HDP + HNDP - 10			Ts
t2	YD hold from LP falling edge	6			Ts
t3	LP period		HDP + HNDP		Ts
t4	LP pulse width	(LP_SEL + 1)			Ts
t5	WF delay from LP falling edge	0			ns
t6*	XSCL falling edge to LP rising edge	(HNDP -10.5)			Ts
t7*	XSCL falling edge to LP falling edge	(HNDP -9.5 + LP_SEL)			Ts
t8	LP falling edge to XSCL falling edge	(11.5 - LP_SEL)			Ts
t9	XSCL period	1			Ts
t10	XSCL pulse width low	0.5			Ts
t11	XSCL pulse width high	0.5			Ts
t12	UD[3:0], LD[3:0] setup to XSCL falling edge	0.5			Ts
t13	UD[3:0], LD[3:0] hold to XSCL falling edge	0.5			Ts
t14	LP falling edge to XSCL rising edge	(11 - LP_SEL)			Ts

Ts	=	Pixel Clock Period:	70 ns typical
HDP	=	Horizontal Display Period:	[(CRTC B[01h]) * 8] T_S
HNDP	=	Horizontal Non-Display Period:	[(CRTC B[00h] bits 2-0) *16] T_S
LP_SEL[5:0	=	LP Pulse Width - 1:	[(AUX[02h] bits 7-5)] T_S

7.8.8 Dual Color 8-bit Panel Timing (One Shift Clock)

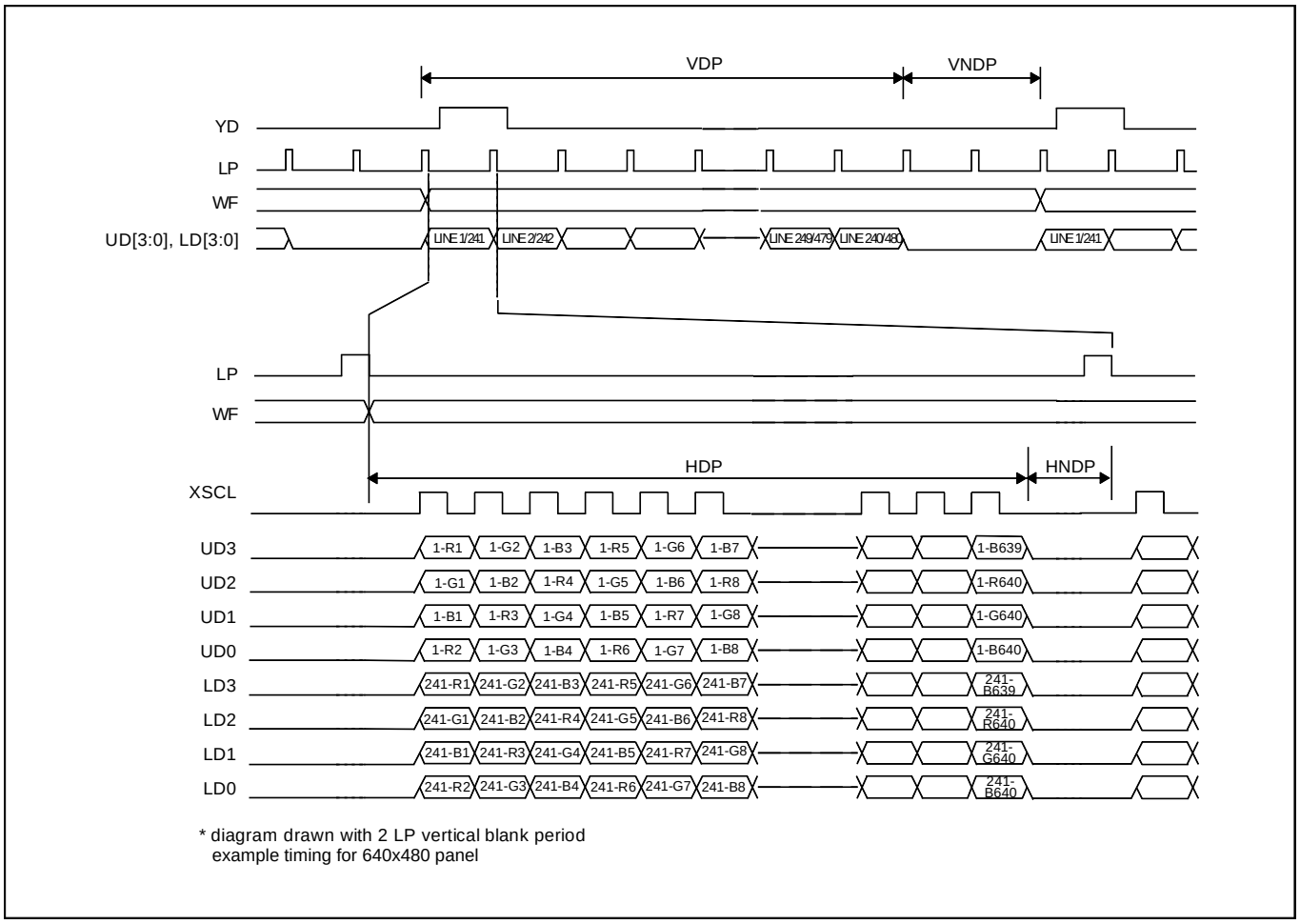


Figure 30: Dual Color 8-bit Panel Timing (One Shift Clock)

VDP	=	Vertical Display Period:	[(CRTC B[12h]) * 4] T_{LP}
VNDP	=	Vertical Non-Display Period:	[{(CRTC B[07h] bits 5, 0), (CRTC B[06h])} + 1 - VDP] T_{LP}
HDP	=	Horizontal Display Period:	[(CRTC B[01h]) * 6] T_{XSCL}
HNDP	=	Horizontal Non-Display Period:	[(CRTC B[00h] bits 2-0) * 12] T_{XSCL}

Dual Color 8-bit Panel A.C. Timing (One Shift Clock)

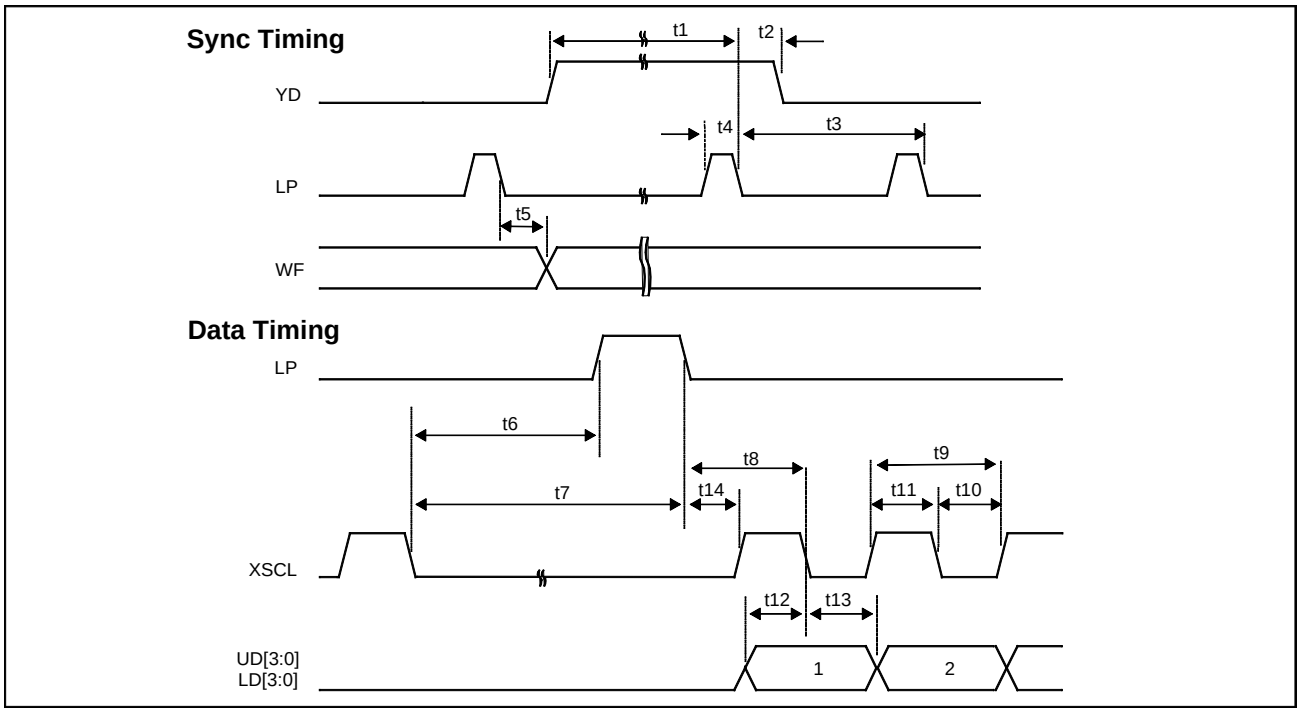


Figure 31: Dual Color 8-bit Panel A.C. Timing (One Shift Clock)

Table 7-15: Dual Color 8-bit Panel A.C. Timing (One Shift Clock)

Symbol	Parameter	Min	Typ	Max	Units
t1	YD setup to LP falling edge	HDP + HNDP - 10			Ts
t2	YD hold from LP falling edge	6			Ts
t3	LP period		HDP + HNDP		Ts
t4	LP pulse width	(LP_SEL + 1)			Ts
t5	WF delay from LP falling edge	0			ns
t6*	XSCL falling edge to LP rising edge	(HNDP - 9.5)			Ts
t7*	XSCL falling edge to LP falling edge	(HNDP - 8.5 + LP_SEL)			Ts
t8	LP falling edge to XSCL falling edge	(11.5 - LP_SEL)			Ts
t9	XSCL period	1			Ts
t10	XSCL pulse width low	0.5			Ts
t11	XSCL pulse width high	0.5			Ts
t12	UD[3:0], LD[3:0] setup to XSCL falling edge	0.5			Ts
t13	UD[3:0], LD[3:0] hold to XSCL falling edge	0.5			Ts
t14	LP falling edge to XSCL rising edge	11 - LP_SEL)			Ts

Ts	=	Pixel Clock Period:	70 ns typical
HDP	=	Horizontal Display Period:	[(CRTC B[01h]) * 8] T_S
HNDP	=	Horizontal Non-Display Period:	[(CRTC B[00h] bits 2-0) *16] T_S
LP_SEL[5:0]	=	LP Pulse Width - 1:	[(AUX[02h] bits 7-5)] T_S

7.8.9 Dual Color 8-bit Panel Timing (Two Shift Clocks)

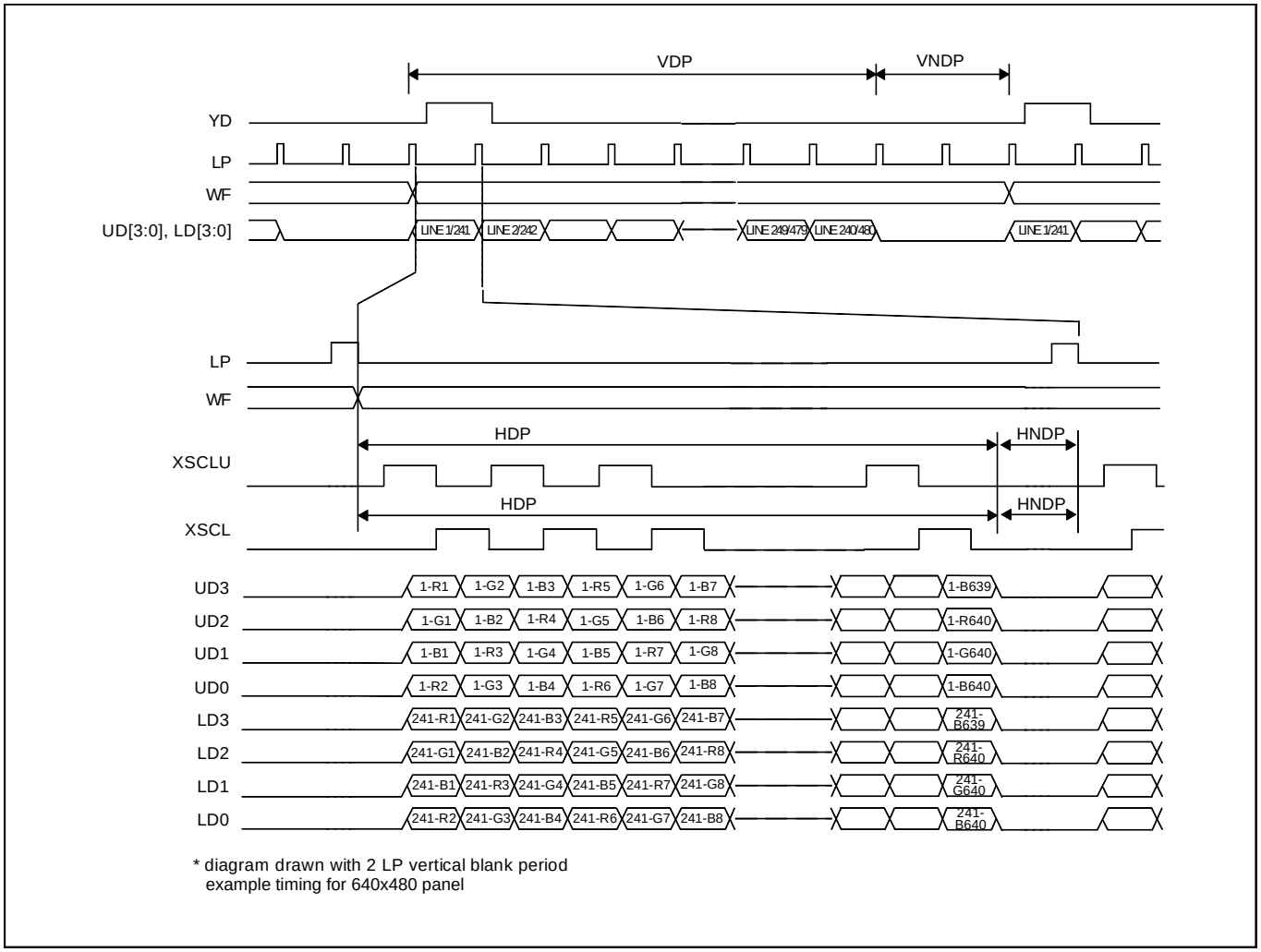


Figure 32: Dual Color 8-bit Panel Timing (Two Shift Clocks)

VDP = Vertical Display Period: [(CRTC B[12h]) * 4] T_{LP}

VNDP = Vertical Non-Display Period: [{(CRTC B[07h] bits 5, 0), (CRTC B[06h])} + 1 - VDP] T_{LP}

HDP = Horizontal Display Period: [(CRTC B[01h]) * 3] T_{XSCL}

HNDP = Horizontal Non-Display Period: [(CRTC B[00h] bits 2-0) * 6] T_{XSCL}

Dual Color 8-bit Panel A.C. Timing (Two Shift Clocks)

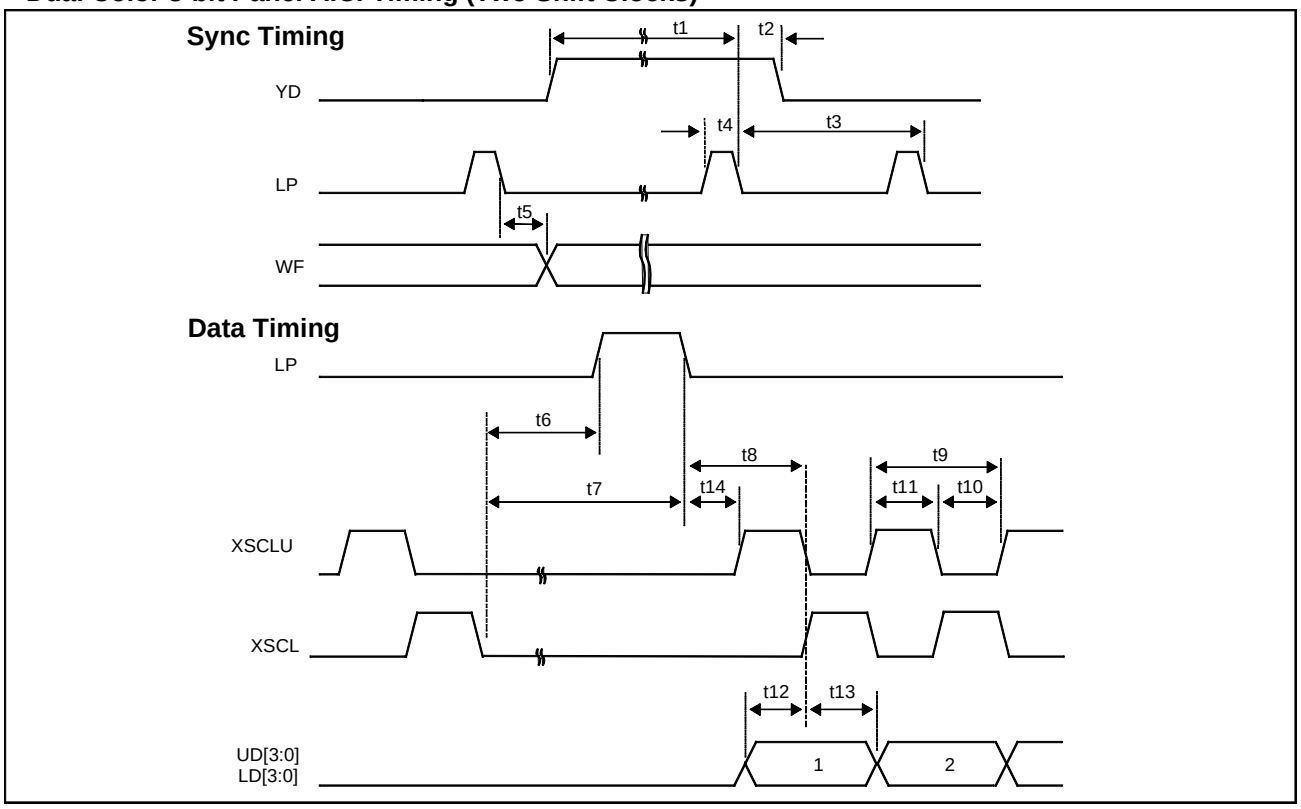


Figure 33: Dual Color 8-bit Panel A.C. Timing (Two Shift Clocks)

Table 7-16: Dual Color 8-bit Panel A.C. Timing (Two Shift Clocks)

Symbol	Parameter	Min	Typ	Max	Units
t1	YD setup to LP falling edge	HDP + HNDP - 10			Ts
t2	YD hold from LP falling edge	6			Ts
t3	LP period		HDP + HNDP		Ts
t4	LP pulse width	(LP_SEL + 1)			Ts
t5	WF delay from LP falling edge	0			ns
t6	XSCLU, XSCL falling edge to LP rising edge	(HNDP - 9.5)			Ts
t7	XSCLU, XSCL falling edge to LP falling edge	(HNDP - 8.5 + LP_SEL)			Ts
t8	LP falling edge to XSCL falling edge	(11.5 - LP_SEL)			Ts
t9	XSCLU, XSCL period	2			Ts
t10	XSCLU, XSCL pulse width low	1			Ts
t11	XSCLU, XSCL pulse width high	1			Ts
t12	UD[3:0], LD[3:0] setup to XSCL, XSCLU falling edge	0.5			Ts
t13	UD[3:0], LD[3:0] hold to XSCLU, XSCL falling edge	0.5			Ts
t14	LP falling edge to XSCLU, XSCL rising edge	(10.5 - LP_SEL)			Ts

Ts	=	Pixel Clock Period:	70 ns typical
HDP	=	Horizontal Display Period:	[(CRTC B[01h]) * 8] T_S
HNDP	=	Horizontal Non-Display Period:	[(CRTC B[00h] bits 2-0) *16] T_S
LP_SEL[5:0]	=	LP Pulse Width - 1:	[(AUX[02h] bits 7-5)] T_S

7.8.10 Dual Color 16-bit Panel Timing

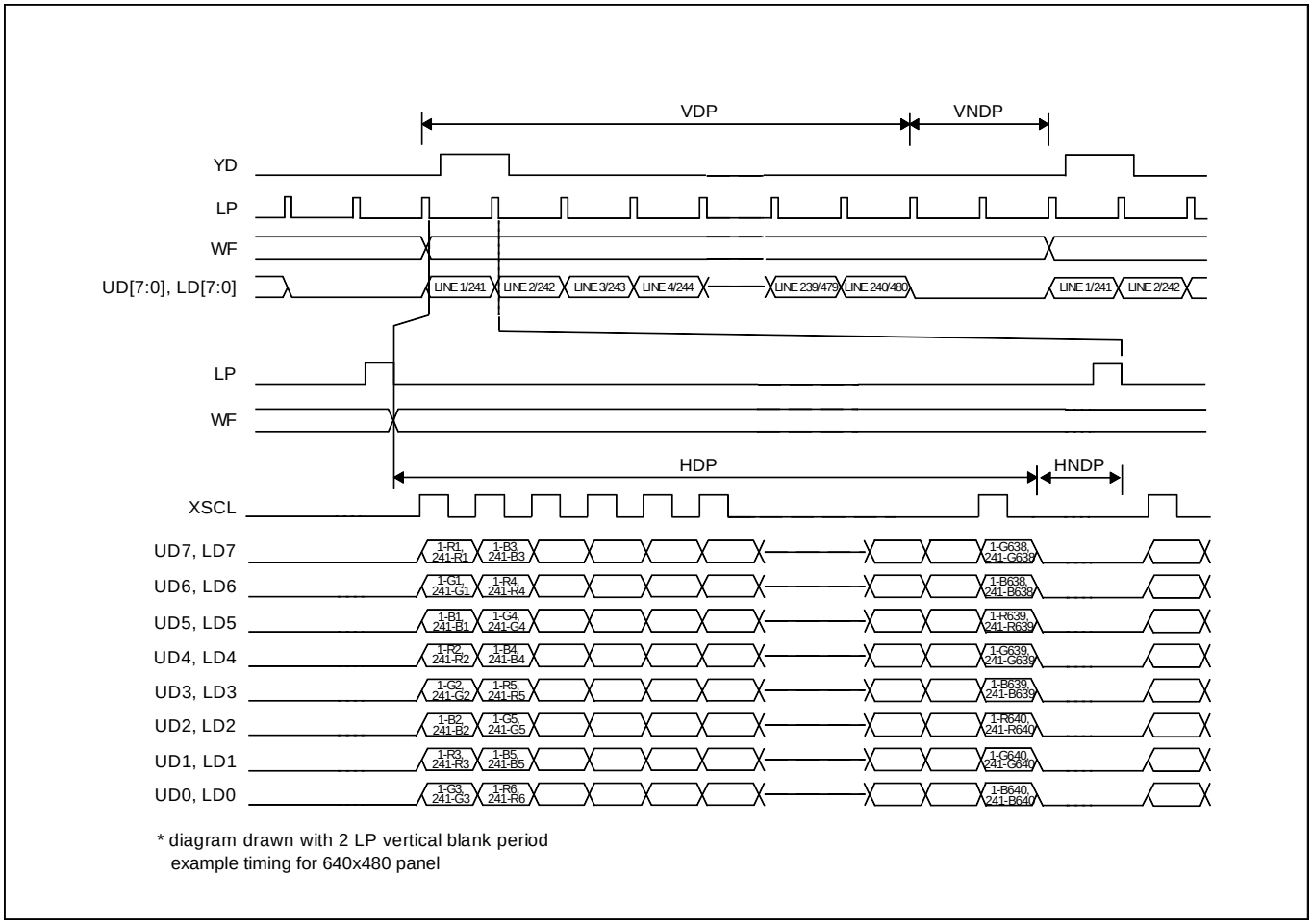


Figure 34: Dual Color 16-bit Panel Timing

VDP	=	Vertical Display Period:	[(CRTC B[12h]) * 4] T_{LP}
VNDP	=	Vertical Non-Display Period:	[{(CRTC B[07h] bits 5, 0), (CRTC B[06h])} + 1 - VDP] T_{LP}
HDP	=	Horizontal Display Period:	[(CRTC B[01h]) * 3] T_{XSCL}
HNDP	=	Horizontal Non-Display Period:	[(CRTC B[00h] bits 2-0) * 6] T_{XSCL}

Dual Color 16-bit Panel A.C. Timing

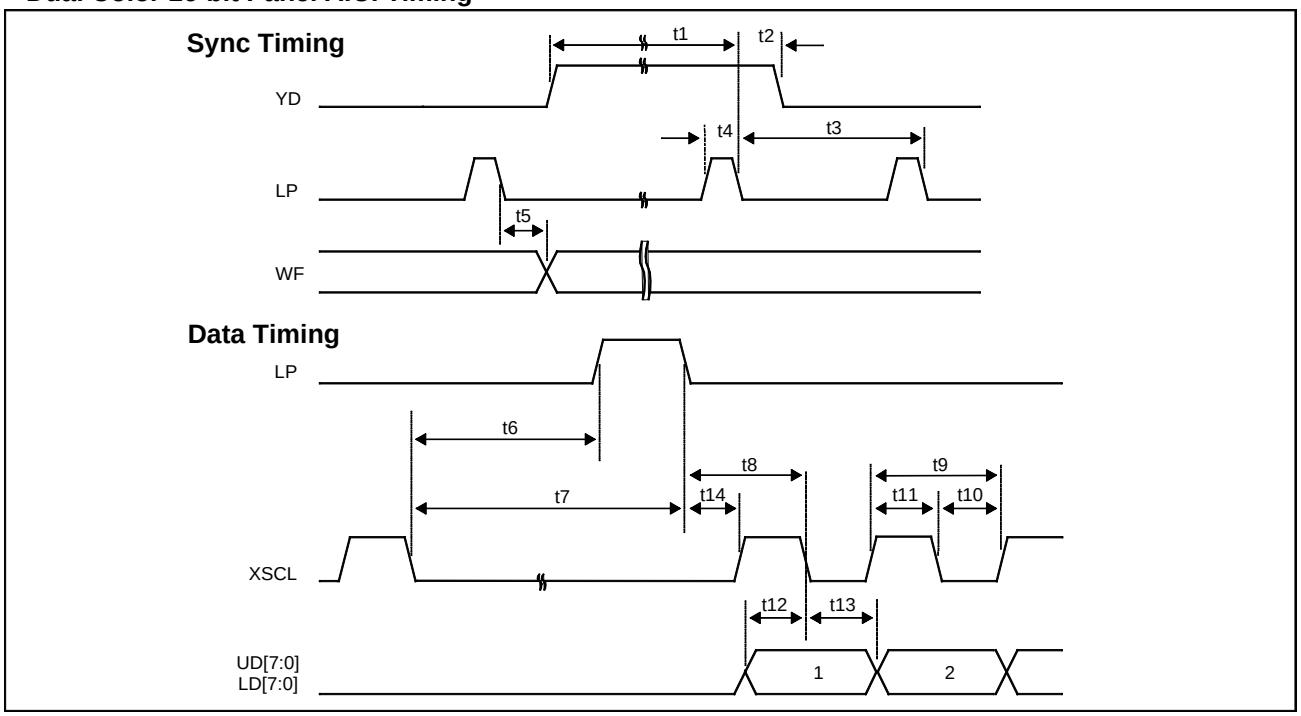


Figure 35: Dual Color 16-bit Panel A.C. Timing

Table 7-17: Dual Color 16-bit Panel A.C. Timing

Symbol	Parameter	Min	Typ	Max	Units
t1	YD setup to LP falling edge	HDP + HNDP - 10			Ts
t2	YD hold from LP falling edge	6			Ts
t3	LP period		HDP + HNDP		Ts
t4	LP pulse width	(LP_SEL + 1)			Ts
t5	WF delay from LP falling edge	0			ns
t6	XSCL falling edge to LP rising edge	(HNDP - 9)			Ts
t7	XSCL falling edge to LP falling edge	(HNDP - 8 + LP_SEL)			Ts
t8	LP falling edge to XSCL falling edge	(12 - LP_SEL)			Ts
t9	XSCL period	2			Ts
t10	XSCL pulse width low	1			Ts
t11	XSCL pulse width high	1			Ts
t12	UD[7:0], LD[7:0] setup to XSCL falling edge	1			Ts
t13	UD[7:0], LD[7:0] hold to XSCL falling edge	1			Ts
t14	LP falling edge to XSCL rising edge	(11 - LP_SEL)			Ts

Ts	=	Pixel Clock Period:	70 ns typical
HDP	=	Horizontal Display Period:	[(CRTC B[01h]) * 8] T_S
HNDP	=	Horizontal Non-Display Period:	[(CRTC B[00h] bits 2-0) *16] T_S
LP_SEL[5:0]	=	LP Pulse Width - 1:	[(AUX[02h] bits 7-5)] T_S

7.8.11 TFT Panel Timing

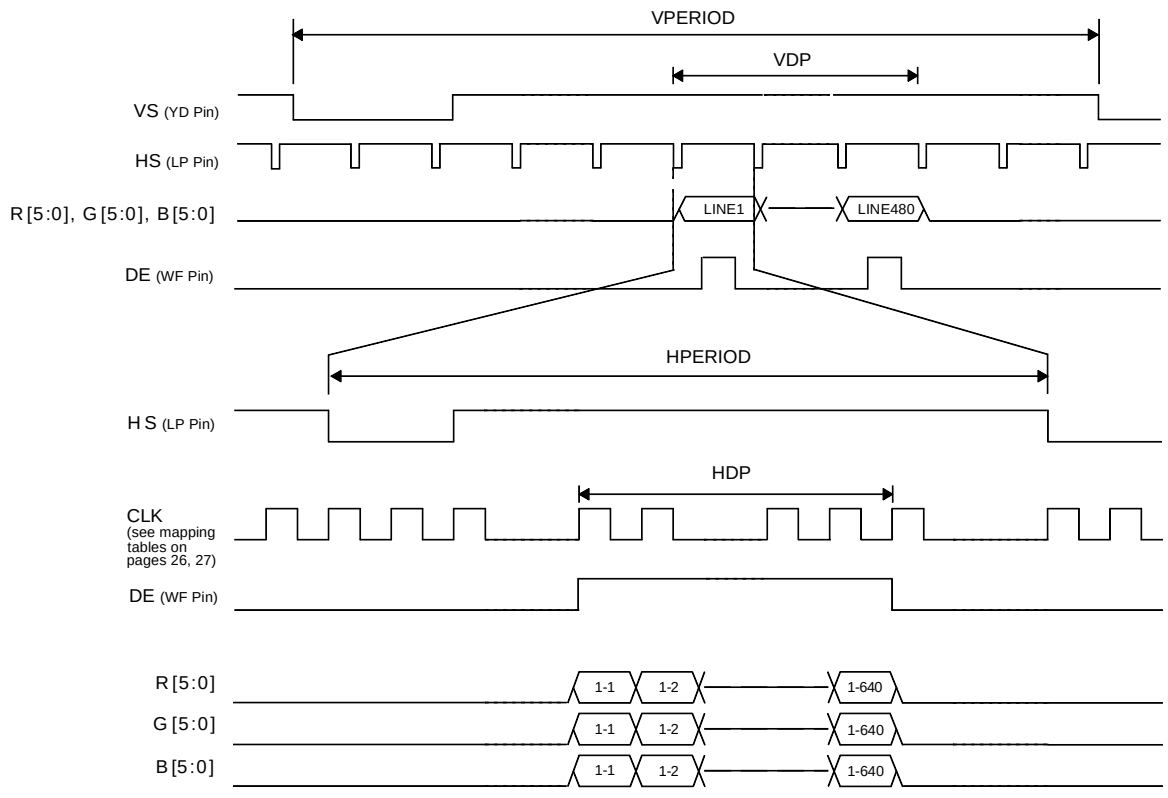
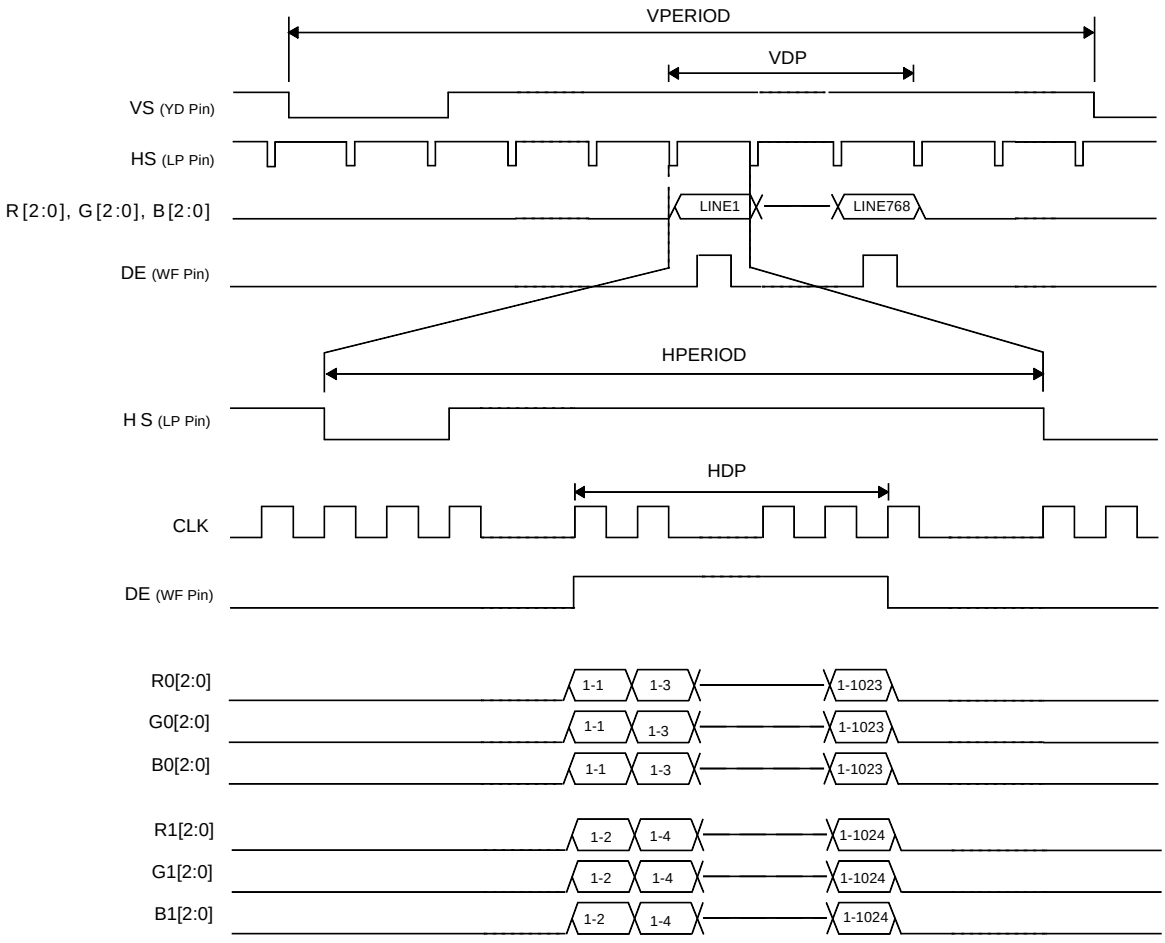


Figure 36: 18-bit TFT Panel Timing

VPERIOD	=	Vertical Display Period:	[{(CRTC B[07h] bits 5, 0), (CRTC B[06h])} + 1] T_{HS}
VDP	=	Vertical Display Period:	[(CRTC B[12h]) * 4] T_{HS}
HPERIOD	=	Horizontal Period:	[(CRTC [00h] + 5) * 8] T_{CLK}
HDP	=	Horizontal Display Period:	[(CRTC B[01h]) * 8] T_{CLK}



Note: DE (Data Enable) is used to indicate the first pixel
example timing for 1024x768 panel

Figure 37: 2 x 9-bit TFT Panel Timing

VPERIOD	=	Vertical Display Period:	[{(CRTC B[07h] bits 5, 0), (CRTC B[06h])} + 1] T_{HS}
VDP	=	Vertical Display Period:	[(CRTC B[12h]) * 4] T_{HS}
HPERIOD	=	Horizontal Period:	[(CRTC [00h] + 5) * 8] (T_{CLK}/2)
HDP	=	Horizontal Display Period:	[(CRTC B[01h]) * 8] (T_{CLK}/2)

TFT A.C. Timing

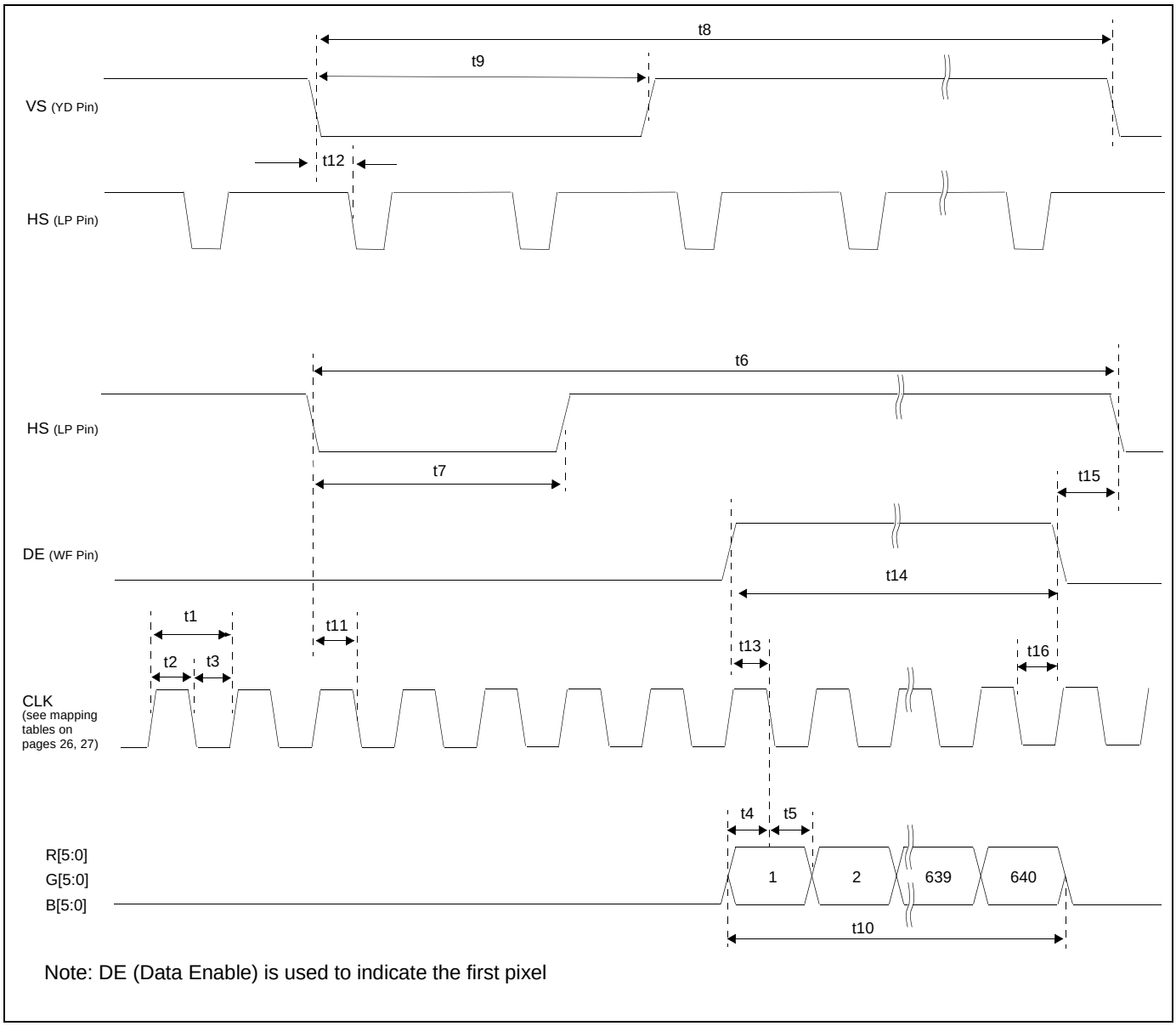


Figure 38: TFT A.C. Timing

Table 7-18: TFT A.C. Timing

Symbol	Parameter	Min	Typ	Max	Units
t1	CLK period	Ts			
t2	CLK pulse width high	0.5Ts			
t3	CLK pulse width low	0.5Ts			
t4	data setup to CLK falling edge	0.5Ts			
t5	data hold from CLK falling edge	0.5Ts			
t6	HS cycle time		note 2		
t7	HS pulse width low		note 3		
t8	VS cycle time		note 4		lines
t9	VS pulse width low		note 5		lines
t10	horizontal display period		note 6		
t11	HS setup to CLK falling edge	0.5Ts			
t12	VS falling edge to HS falling edge phase difference	0.5Ts			
t13	DE to CLK falling edge setup time	0.5Ts			
t14	DE pulse width		note 7		
t15	DE falling edge to HS falling edge	note 8		note 9	
t16	DE hold from CLK falling edge	0.5Ts			

Note

1. Ts = CLK (clock period)
2. $t6_{typ}$ = [(CRTC [00h] + 5) * 8] Ts
3. $t7_{typ}$ = [(CRTC [05h] bits 4-0) - (CRTC [04h] bits 4-0)] * 8 Ts
4. $t8_{typ}$ = [{(CRTC B[07h] bits 5, 0), (CRTC B[06h])} + 1] T_{HS}
5. $t9_{typ}$ = [(CRTC B[11h] bits 3-0) - (CRTC B[10h] bits 3-0)] lines (If result is negative, add 16)
6. $t10_{typ}$ = [(CRTC B[01h]) * 8] Ts
7. $t14_{typ}$ = [(CRTC B[01h]) * 8] Ts
8. $t15_{min}$ = {[(CRTC [04h]) - (CRTC B[01h]) * 8] - 15} Ts
9. $t15_{max}$ = {[(CRTC [04h]) - (CRTC B[01h]) * 8] - 13} Ts

If using formulas to calculate the timing for 9x2 TFT, replace Ts above with Ts/2 in formulas 2, 3, 6, 7, 8, 9.

8 REGISTER DESCRIPTIONS

8.1 Software Considerations

This section describes deviations of the SPC8110F0A from the VGA standard register specification.

8.1.1 Background

There are two ways in which the SPC8110F0A VGA register set differs from the VGA standard.

First, there is a second set of CRTC timing registers called the B-set which is used to control the timing in LCD and simultaneous display modes. The standard A-set timing registers are used in CRT mode.

Second, there are some functions provided in the VGA register set that are never used in common VGA applications - the register bits for these functions have been removed in order to optimize the design for VGA operation on a 640 x 480 dot LCD panel. In their places are read/write registers bits that have no effect.

In addition to the VGA registers, there are the auxiliary registers, the BitBLT registers, Sprite/Hardware Cursor registers, and the memory paging register, all of which are necessary to support the features of the SPC8110F0A.

8.1.2 Unsupported VGA Registers

The following table lists all the standard VGA registers bits that have had their functionality removed.

Table 8-1: Unsupported VGA Registers

Register	Address	Index	Bit	Description	Status
Misc	3C2h(W), 3CCh(R)		1	Enable DRAM	Dummy
Input Status 0	3C2h		5, 6	Feature Inputs	R as 0,0
Input Status 1	3DAh		1, 2	Light Pen	R as 0,1
			4, 5	Diagnostic	R as 0,0
Sequencer	3C4h, 3C5h	00h	0, 1	Sequencer Reset	Dummy in LCD mode
		01h	1	Bandwidth	Dummy
		01h	2	Shiftload	Dummy
		01h	4	Shift by 4	Dummy
		04h	0	Alpha Select	R as 0
		04h	1	Memory Size	R as 1
Attributes Controller	3C0h, 3C1h	10h	6	Pel Width	Dummy
		12h	4, 5	Display Status	Dummy
CRT Controller	3D4h, 3D5h 3B4h, 3B5h	03h	5, 6	Horizontal Skew Bits	Dummy
		03h	7	Read Select	Dummy
		0Bh	5, 6	Cursor Skew	Dummy
		11h	6	Refresh	Dummy
		17h	2	HRTC by 2	Dummy
		17h	3	Count by 2	Dummy
		17h	7	Retrace Enable	R as 1

Where: Dummy: R/W only. No other effect in hardware.
Deleted: Location completely removed. Bit is now reserved.
R as x: Bit changed from R/W status to R status, with x defining the readback state.

8.2 VGA I/O Register Summary

This section summarizes the VGA I/O registers of SPC8110F0A. Note that the functionality of a subset of the IBM VGA standard registers is supported, with an additional set of Auxiliary Registers and CRTC Registers containing SPC8110F0A specific functions. Only details of the register functions which are not part of the IBM VGA standard definition are given below.

Note

Unless otherwise noted, all read/write register bits are cleared to 0 after a reset and all register bits marked as “Reserved” have no effect on hardware and read-only as 0.

8.2.1 General Registers

Miscellaneous Output Register **3C2h W 3CCh R**							
VRTC Polarity	HRTC Polarity	Odd/Even Page Bit	Reserved	Clock Select Bit 1	Clock Select Bit 0	Dummy Enable DRAM	3Dx/3Bx Select

bit 1 Dummy Enable DRAM
Read/write only. No other effect in hardware.

bits 2-3 Clock Select Bits[1:0]
These bits are used to select which PLL1 timing registers are used to generate the active pixel clock. PLL1 has three sets of timing registers associated with it. Each set has a numerator and a denominator which define the frequency output of PLL1. For normal operation Set 0 should be programmed to 25.175 MHz, and Set 1 should be programmed to 28.6 MHz.

Table 8-2: Clock Select Options

Clock Select		**Clock register selected**
Bit 1	**Bit 0**	
0	0	PLL1 Set 0
0	1	PLL1 Set 1
1	x	PLL1 Set 2

Note that for LCD or simultaneous display operation, the setting of the above bits is ignored. For more information see Programming PLL1 (Pixel Clock) on page 120.

Input Status Register 0 **3C2h R**							
CRTC Vertical Interrupt Status	Feature Input Bit 1 RO as 0	Feature Input Bit 0 RO as 0	DAC Sense Bit	Reserved	Reserved	Reserved	Reserved

bit 4 DAC Sense Bit
For LCD mode, this bit reads back a zero.
When in CRT or Simultaneous Display mode, this bit is the output of the DAC’s internal analog comparator that detects the presence of an attached CRT display.

bits 5-6 Feature Input Bits [1:0]
Read-only as 0.

bit 7 CRTC Vertical Interrupt Status
This read-only bit shows the status of the vertical retrace interrupt. When read as a 1 a vertical interrupt is pending, when read as 0, vertical interrupt is cleared.

Input Status Register 1 **3DAh R**							
Reserved	Reserved	Diagnostic Bit1 RO as 0	Diagnostic Bit 0 RO as 0	Vertical Retrace Status	Light Pen Switch RO as 1	Light Pen Latch RO as 0	Display Enable Status

bit 1 Light Pen Latch
Read-only as 0.

bit 2 Light Pen Switch
Read-only as 1.

Note

the readback combination of the above two bits signifies that no light pen is attached.

bits 4-5 Diagnostic Bits [1:0]
Read-only as 00.

Video Subsystem Enable Register **3C3h RW**							
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Video Subsystem Enable

8.2.2 Palette Registers

The palette registers 3C6h - 3C9h are identical to the VGA palette interface. When the sprite palette enable bit is set to one (AUX [00h] b7), access to 3C6h - 3C9h is re-directed to the Sprite/HW Cursor palette. Since the Sprite/HW Cursor only has four palette entries, only the bits [1:0] of the read and write address registers 3C7h and 3C8h are significant.

Only 8-bit I/O access is permitted when reading or writing to the palette registers. All 32-bit accesses are ignored (e.g., a 32-bit write to 3C8h will not be decoded) and 16-bit accesses produce unpredictable results.

When Pixel clock is not stable or is shut down, access to all 4 palette registers is ignored. When coming out of reset or a power down mode where Pixel clock is shut down, software must check PLL1LOCK status (AUX [28h] b0) before attempting any access to these registers.

Palette Table Pixel Mask Register **3C6h RW**							
Palette Pixel Mask Bit 7	Palette Pixel Mask Bit 6	Palette Pixel Mask Bit 5	Palette Pixel Mask Bit 4	Palette Pixel Mask Bit 3	Palette Pixel Mask Bit 2	Palette Pixel Mask Bit 1	Palette Pixel Mask Bit 0

bits 0-7 Palette Pixel Mask bits[7:0]
This masking register will also affect the palette selection of the Sprite/HW Cursor.

Palette Table Status Register **3C7h R**							
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Read Write Mode Status Bit 1	Read Write Mode Status Bit 0

bits 0-1 Read Write Mode Status Bits [1:0]
Directly after a write to the Palette Table Read Address Register, the Read/Write Mode Status bits 1-0 are both set to 1. Directly after a write to the Palette Table Write Address Register, both these bits are set to 0. This also applies to Sprite/HW Cursor palette accesses.

Palette Read Address Register **3C7h W**							
Palette Read Address Bit 7	Palette Read Address Bit 6	Palette Read Address Bit 5	Palette Read Address Bit 4	Palette Read Address Bit 3	Palette Read Address Bit 2	Palette Read Address Bit 1	Palette Read Address Bit 0

bits 0-7 Palette Read Address Bits [7:0]
When sprite palette enable is set (AUX [00h] b7), bits [7:2] are ignored.

Palette Write Address Register **3C8h RW**							
Palette Write Address Bit 7	Palette Write Address Bit 6	Palette Write Address Bit 5	Palette Write Address Bit 4	Palette Write Address Bit 3	Palette Write Address Bit 2	Palette Write Address Bit 1	Palette Write Address Bit 0

bits 0-7 Palette Write Address Bits [7:0]
When sprite palette enable is set (AUX [00h] b7), bits [7:2] are ignored.

Palette Data Register **3C9h RW**							
Reserved	Reserved	Palette Data Bit 5	Palette Data Bit 4	Palette Data Bit 3	Palette Data Bit 2	Palette Data Bit 1	Palette Data Bit 0

8.2.3 Sequencer Registers

Sequencer Index/Data Register **3C4h RW 3C5h RW**							
Reserved	Reserved	Reserved	Reserved	Reserved	Sequencer Index Bit 2	Sequencer Index Bit 1	Sequencer Index Bit 0

bits 0-2 The Sequencer Index bits select which of the Sequencer Data registers is available at address 3C5h.

bits 3-7 Reserved bits read back as 0.

Reset Register **00h RW**							
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Sequencer Reset	Sequencer Reset

Clocking Mode Register **01h RW**							
Reserved	Reserved	Screen Off	Dummy Shift by 4	Dotclock Divide by 2	Dummy Shift Load	Dummy Bandwidth	8/9 Dot Select

bit 0 8/9 Dot Select
For LCD and simultaneous display modes: No effect on character clock.
For CRT modes, this bit selects 8 or 9 dot mode.

bit 1 Dummy CPU Bandwidth
Read/write only. No other effect in hardware.

bit 2 Dummy Shift-Load
Read/write only. No other effect in hardware.

bit 3 Dotclock Divide By 2
This bit “divides” the dotclock by two when set to a 1.

bit 4 Dummy Shift-4
Read/write only. No other effect in hardware.

Map Mask Register **02h RW**							
Reserved	Reserved	Reserved	Reserved	Plane 3 Write Enable	Plane 2 Write Enable	Plane 1 Write Enable	Plane 0 Write Enable

Character Map Select Register **03h RW**							
Reserved	Reserved	Map A Select Bit 2	Map B Select Bit 2	Map A Select Bit 1	Map A Select Bit 0	Map B Select Bit 1	Map B Select Bit 0

Memory Mode Register **04h RW**							
Reserved	Reserved	Reserved	Reserved	Chain 4	Odd/Even Map Select	256K Memory Size RO as 1	Reserved

bit 1 256K Memory Size
Read-only as 1.

8.2.4 Graphics Controller Registers

Graphics Controller Index/Data Register **3CEh RW 3CF RW**							
Reserved	Reserved	Reserved	Reserved	Graphics Controller Index Bit 3	Graphics Controller Index Bit 2	Graphics Controller Index Bit 1	Graphics Controller Index Bit 0

Set/Reset Register **00h RW**							
Reserved	Reserved	Reserved	Reserved	Set/Reset Plane 3	Set/Reset Plane 2	Set/Reset Plane 1	Set/Reset Plane 0

Enable Set/Reset Register **01h RW**							
Reserved	Reserved	Reserved	Reserved	Enable Set/Reset Plane 3	Enable Set/Reset Plane 2	Enable Set/Reset Plane 1	Enable Set/Reset Plane 0

Color Compare Register **02h RW**							
Reserved	Reserved	Reserved	Reserved	Color Compare Bit 3	Color Compare Bit 2	Color Compare Bit 1	Color Compare Bit 0

Data Rotate Register **03h RW**							
Reserved	Reserved	Reserved	Logic Function Select Bit 1	Logic Function Select Bit 0	Data Rotate Count Bit 2	Data Rotate Count Bit 1	Data Rotate Count Bit 0

Read Map Select Register **04h RW**							
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Read Plane Select Bit 1	Read Plane Select Bit 0

Mode Register **05h RW**							
Reserved	256 Color Select	Shift Register Interleave	Odd/Even Plane Select	Read Mode Select	Reserved	Write Mode Select Bit 1	Write Mode Select Bit 0

bits 5-6 Shift Register Interleave, 256 Color select
These two bits control the graphics mode display shift registers.

Miscellaneous Register **06h RW**							
Reserved	Reserved	Reserved	Reserved	Display Memory Map Bit 1	Display Memory Map Bit 0	Odd Even Chain Select	Graphics Mode

Color Don't Care Register **07h RW**							
Reserved	Reserved	Reserved	Reserved	Compare Plane Select Bit 3	Compare Plane Select Bit 2	Compare Plane Select Bit 1	Compare Plane Select Bit 0

Bit Mask Register **08h RW**							
Graphics Data Write Mask Bit 7	Graphics Data Write Mask Bit 6	Graphics Data Write Mask Bit 5	Graphics Data Write Mask Bit 4	Graphics Data Write Mask Bit 3	Graphics Data Write Mask Bit 2	Graphics Data Write Mask Bit 1	Graphics Data Write Mask Bit 0

8.2.5 Attributes Controller Registers

Attributes Controller Index/Data Register **3C0h RW 3C0h W 3C1h R**							
Reserved	Reserved	EGA Palette Enable	Attributes Controller Index Bit 4	Attributes Controller Index Bit 3	Attributes Controller Index Bit 2	Attributes Controller Index Bit 1	Attributes Controller Index Bit 0

EGA Palette Registers **00h - 0Fh RW**							
Reserved	Reserved	Secondary Red	Secondary Green	Secondary Blue	Primary Red	Primary Green	Primary Blue

Mode Control Register 10h RW							
EGA Palette Bits 4, 5 Control	Dummy Pel Width	Pixel Pan Compat	Reserved	Blink/ Intensity	Line Graphics Char Code Enable	Mono Mode Select	Graphics Mode Select

bit 2 Line Graphics Character Code Enable
In LCD Active modes the character clock is forced to 8 dots, so the 9th dot is never displayed.

bit 6 Dummy Pel Width
Read/write only. No other effect in hardware.

Overscan Color Register 11h RW							
Overscan Color Bit 7	Overscan Color Bit 6	Overscan Color Bit 5	Overscan Color Bit 4	Overscan Color Bit 3	Overscan Color Bit 2	Overscan Color Bit 1	Overscan Color Bit 0

Color Plane Enable Register 12h RW							
Reserved	Reserved	Dummy Display Status Bit 1	Dummy Display Status Bit 0	Enable Plane 3	Enable Plane 2	Enable Plane 1	Enable Plane 0

bits 4-5 Dummy Display Status MUX Bits [1:0].
Read/write only. No other effect in hardware.

Horizontal Panning Register 13h RW							
Reserved	Reserved	Reserved	Reserved	Horizontal Pan Count Bit 3	Horizontal Pan Count Bit 2	Horizontal Pan Count Bit 1	Horizontal Pan Count Bit 0

bits 0-3 Horizontal Panning Register
This register sets the number of pixels the display is shifted left horizontally:

Table 8-3: Horizontal Panning Register Display Shift

Value	9 Dot Modes	8 Dot Modes	Mode 13h	HR256 Mode
0	1	0	0	0
1	2	1	-	1
2	3	2	1	2
3	4	3	-	3
4	5	4	2	4
5	6	5	-	4
6	7	6	3	4
7	8	7	-	4
8	0	-1	-	4
9	0	-1	-	4
A	0	-1	-	4
B	0	-1	-	4
C	0	-1	-	4
D	0	-1	-	4
E	0	-1	-	4
F	0	-1	-	4

Note

When LCD Enable is true, the character clock is forced to 8 dots, BUT the horizontal panning logic still operates in 9 dot mode if the 8/9 dot select register is set to 9 dots.

Color Select Register 14h RW							
Reserved	Reserved	Reserved	Reserved	Color Select Bit 3	Color Select Bit 2	Color Select Bit 1	Color Select Bit 0

bits 0-3 Color Select Bits [3:0]
In 256 Color modes these color select registers have no effect.

bits 4-7 Reserved bits read back as 0.

8.2.6 CRT Controller Registers - Standard Set (A-Set)

CRT Controller Index/Data Register **3B4h/3D4h RW**		**3B5h/3D5h RW**					
Reserved	Reserved	CRT Controller Index Bit 5	CRT Controller Index Bit 4	CRT Controller Index Bit 3	CRT Controller Index Bit 2	CRT Controller Index Bit 1	CRT Controller Index Bit 0

Horizontal Total Register **00h RW if Aux (index 00h) bit 0 = 0**							
Horizontal Total Bit 7	Horizontal Total Bit 6	Horizontal Total Bit 5	Horizontal Total Bit 4	Horizontal Total Bit 3	Horizontal Total Bit 2	Horizontal Total Bit 1	Horizontal Total Bit 0

bits 0-7 Horizontal Total Bits [7:0]
For passive LCD-only modes - Read/write only. No other effect in hardware.
For CRT and simultaneous display modes, normal VGA functionality.
For TFT display modes, this register must be set ≥ CRTC B[01] Horizontal Panel Size.

Horizontal Display Enable End Register **01h RW if Aux (index 00h) bit 0 = 0**							
Horizontal Disp Enable End Bit 7	Horizontal Disp Enable End Bit 6	Horizontal Disp Enable End Bit 5	Horizontal Disp Enable End Bit 4	Horizontal Disp Enable End Bit 3	Horizontal Disp Enable End Bit 2	Horizontal Disp Enable End Bit 1	Horizontal Disp Enable End Bit 0

Horizontal Blanking Start Register **02h RW**							
Horizontal Blanking Start Bit 7	Horizontal Blanking Start Bit 6	Horizontal Blanking Start Bit 5	Horizontal Blanking Start Bit 4	Horizontal Blanking Start Bit 3	Horizontal Blanking Start Bit 2	Horizontal Blanking Start Bit 1	Horizontal Blanking Start Bit 0

bits 0-7 Horizontal Blanking Start Bits [7:0]
For passive LCD-only modes - Read/write only. No other effect in hardware.
For CRT, TFT and simultaneous display modes, normal VGA functionality.

Horizontal Blanking End Register
03h RW

Dummy Read Select	Dummy Horizontal Skew Bit 1	Dummy Horizontal Skew Bit 0	Horizontal Blanking End Bit 4	Horizontal Blanking End Bit 3	Horizontal Blanking End Bit 2	Horizontal Blanking End Bit 1	Horizontal Blanking End Bit 0

bits 0-4 Horizontal Blanking End Bits [4:0]
For passive LCD-only modes - Read/write only. No other effect in hardware.
For CRT, TFT and simultaneous display modes, normal VGA functionality.

bits 5-6 Dummy Horizontal Skew Bits [1:0]
Read/write only. No other effect in hardware.

bit 7 Dummy Read Select
Read/write only. No other effect in hardware.

Horizontal Retrace Start Register
04h RW

Horizontal Retrace Start Bit 7	Horizontal Retrace Start Bit 6	Horizontal Retrace Start Bit 5	Horizontal Retrace Start Bit 4	Horizontal Retrace Start Bit 3	Horizontal Retrace Start Bit 2	Horizontal Retrace Start Bit 1	Horizontal Retrace Start Bit 0

bits 0-7 Horizontal Retrace Start Bits [7:0]
For passive LCD-only modes - Read/write only. No other effect in hardware.
For CRT, TFT and simultaneous display modes, normal VGA functionality.

Horizontal Retrace End Register
05h RW

Horizontal Blank End Bit 5	Horizontal Retrace Delay Bit 1	Horizontal Retrace Delay Bit 0	Horizontal Retrace End Bit 4	Horizontal Retrace End Bit 3	Horizontal Retrace End Bit 2	Horizontal Retrace End Bit 1	Horizontal Retrace End Bit 0

bits 0-4 Horizontal Retrace End Bits [4:0]
For passive LCD-only modes - Read/write only. No other effect in hardware.
For CRT, TFT and simultaneous display modes, normal VGA functionality.

Vertical Total End Register 0
06h RW if Aux (index 00h) bit 0 =

Vertical Total End Bit 7	Vertical Total End Bit 6	Vertical Total End Bit 5	Vertical Total End Bit 4	Vertical Total End Bit 3	Vertical Total End Bit 2	Vertical Total End Bit 1	Vertical Total End Bit 0

bits 0-7 Vertical Total End Bits [7:0]
For all LCD and simultaneous display modes - Read/write only. No other effect in hardware.
For CRT modes, normal VGA functionality.

CRTC Overflow Register **07h RW if AUX[00h] b0 = 0**							
Vertical Retrace Start Bit 9	Vertical Display Enable End Position Bit 9	Vertical Total End Bit 9	Line Compare Bit 8	Vertical Blank Start Bit 8	Vertical Retrace Start Bit 8	Vertical Display Enable End Position Bit 8	Vertical Total End Bit 8

bits 0, 5 Vertical Total End Bit [9:8]
For all LCD and simultaneous display modes - Read/write only. No other effect in hardware.
For CRT modes - normal VGA functionality.

bits 1, 6 Vertical Display Enable End Bit [9:8]

bits 2, 7 Vertical Retrace Start Bits [9:8]
For all LCD and simultaneous display modes - Read/write only. No other effect in hardware.
For CRT modes - normal VGA functionality.

bit 3 Vertical Blank Start Bit 8
For LCD and simultaneous display modes - Read/write only. No other effect in hardware.
For CRT modes - normal VGA functionality.

bit 4 Line Compare Bit 8

Preset Row Scan Register **08h RW**							
Reserved	Byte Pan Bit 1	Byte Pan Bit 0	Preset Row Scan Bit 4	Preset Row Scan Bit 3	Preset Row Scan Bit 2	Preset Row Scan Bit 1	Preset Row Scan Bit 0

Maximum Scan Line Register **09h RW**							
Line Doubling Enable	Line Compare Bit 9	Vertical Blank Start Bit 9	Max Scan Line Bit 4	Max Scan Line Bit 3	Max Scan Line Bit 2	Max Scan Line Bit 1	Max Scan Line Bit 0

bit 5 Vertical Blank Start Bit 9
For all LCD and simultaneous display modes - Read/write only. No other effect in hardware.
For CRT modes, normal VGA functionality.

bit 6 Line Compare Bit 9

Cursor Start Register **0Ah RW**							
Reserved	Reserved	Cursor Disable	Cursor Start Row Bit 4	Cursor Start Row Bit 3	Cursor Start Row Bit 2	Cursor Start Row Bit 1	Cursor Start Row Bit 0

Cursor End Register **0Bh RW**							
Reserved	Dummy Cursor Skew Bit 1	Dummy Cursor Skew Bit 0	Cursor End Row Bit 4	Cursor End Row Bit 3	Cursor End Row Bit 2	Cursor End Row Bit 1	Cursor End Row Bit 0

bits 5-6 Dummy Cursor Skew Bits [1:0]
Read/write only. No other effect in hardware.

Start Address High Register **0Ch RW**							
Start Address High Bit 7	Start Address High Bit 6	Start Address High Bit 5	Start Address High Bit 4	Start Address High Bit 3	Start Address High Bit 2	Start Address High Bit 1	Start Address High Bit 0

Start Address Low Register **0Dh RW**							
Start Address Low Bit 7	Start Address Low Bit 6	Start Address Low Bit 5	Start Address Low Bit 4	Start Address Low Bit 3	Start Address Low Bit 2	Start Address Low Bit 1	Start Address Low Bit 0

Cursor Position High Register **0Eh RW**							
Cursor Position High Bit 15	Cursor Position High Bit 14	Cursor Position High Bit 13	Cursor Position High Bit 12	Cursor Position High Bit 11	Cursor Position High Bit 10	Cursor Position High Bit 9	Cursor Position High Bit 8

Cursor Position Low Register **0Fh RW**							
Cursor Position Low Bit 7	Cursor Position Low Bit 6	Cursor Position Low Bit 5	Cursor Position Low Bit 4	Cursor Position Low Bit 3	Cursor Position Low Bit 2	Cursor Position Low Bit 1	Cursor Position Low Bit 0

Vertical Retrace Start Register **10h RW if AUX[00h] b0 = 0**							
Vertical Retrace Start Bit 7	Vertical Retrace Start Bit 6	Vertical Retrace Start Bit 5	Vertical Retrace Start Bit 4	Vertical Retrace Start Bit 3	Vertical Retrace Start Bit 2	Vertical Retrace Start Bit 1	Vertical Retrace Start Bit 0

bits 0-7 Vertical Retrace Start Bits [7:0]
For LCD and simultaneous display modes- Read/write only. No other effect in hardware.
For CRT modes, normal VGA functionality.

Vertical Retrace End Register **11h RW if AUX[00h] b0 = 0**							
Protect CRTC Registers 0 to 7	Dummy 3/5 Refresh Cycles	Vertical Interrupt Disable	Vertical Interrupt Clear	Vertical Retrace End Bit 3	Vertical Retrace End Bit 2	Vertical Retrace End Bit 1	Vertical Retrace End Bit 0

bits 0-3 Vertical Retrace End Bits [3:0]
For all LCD and simultaneous display modes - Read/write only. No other effect in hardware.
For CRT modes, normal VGA functionality.

bit 4 Vertical Interrupt Clear
This register will clear a pending “Vertical Interrupt” when set to a 0. This bit should be set to a 1 to allow Vertical interrupts to occur.

bit 5 Vertical Interrupt Disable
This register is set to 1 to disable any vertical interrupts. To allow vertical interrupts to occur, it should be set to 0.

Note

The 8110 does not drive any system interrupt lines. The “Vertical Interrupt” logic is still implemented in case some software uses this logic by polling the interrupt status register bit.

bit 6 Dummy 3/5 Refresh Cycles
Read/Write Only. No other effect in hardware.

bit 7 Protect CRTC Registers 0 to 7
When set to 1, this bit protects the following registers and bits from being written: index 00, 01, 02, 03, 04, 05, 06 - all bits, and index 07 bits 7, 6, 5, 3, 2, 1, 0 (excluding bit 4).

Vertical Display Enable End Register **12h RW if AUX[00h] b0 = 0**							
Vertical Display Enable End Position Bit 7	Vertical Display Enable End Position Bit 6	Vertical Display Enable End Position Bit 5	Vertical Display Enable End Position Bit 4	Vertical Display Enable End Position Bit 3	Vertical Display Enable End Position Bit 2	Vertical Display Enable End Position Bit 1	Vertical Display Enable End Position Bit 0

Offset Register **13h RW**							
Offset Bit 7	Offset Bit 6	Offset Bit 5	Offset Bit 4	Offset Bit 3	Offset Bit 2	Offset Bit 1	Offset Bit 0

Underline Location Register **14h RW**							
Reserved	Double Word Select	Dummy Count-by-4	Underline Row Scan Bit 4	Underline Row Scan Bit 3	Underline Row Scan Bit 2	Underline Row Scan Bit 1	Underline Row Scan Bit 0

bit 5 Dummy Count-by-4
Read/write only. No other effect in hardware.

Vertical Blanking Start Register **15h RW if AUX[00h] b0 = 0**							
Vertical Blank Start Bit 7	Vertical Blank Start Bit 6	Vertical Blank Start Bit 5	Vertical Blank Start Bit 4	Vertical Blank Start Bit 3	Vertical Blank Start Bit 2	Vertical Blank Start Bit 1	Vertical Blank Start Bit 0

bits 0-7 Vertical Blank Start Bits [7:0]
For all LCD and simultaneous display modes - Read/write only. No other effect in hardware.
For CRT modes, normal VGA functionality.

Vertical Blanking End Register **16h RW if AUX[00h] b0 = 0**							
Vertical Blank End Bit 7	Vertical Blank End Bit 6	Vertical Blank End Bit 5	Vertical Blank End Bit 4	Vertical Blank End Bit 3	Vertical Blank End Bit 2	Vertical Blank End Bit 1	Vertical Blank End Bit 0

bits 0-7 Vertical Blank End Bits [7:0]
For all LCD and simultaneous display modes - Read/write only. No other effect in hardware.
For CRT modes, normal VGA functionality.

Mode Control Register **17h RW**							
Retrace Enable RO as 1	Word/Byte Mode Select	MA0 Select MA13/15	Reserved	Dummy Count-by-2	Dummy HRTC/2	MA14/ Row Scan Select	Compatibility Mode Select

bit 2 Dummy HRTC/2 Select
Read/write only. No other effect in hardware.

bit 3 Dummy Count-by-2
Read/write only. No other effect in hardware.

bit 7 Retrace Enable
Read-only as 1. Retraces always enabled.

Line Compare Register 18h RW							
Line Compare Bit 7	Line Compare Bit 6	Line Compare Bit 5	Line Compare Bit 4	Line Compare Bit 3	Line Compare Bit 2	Line Compare Bit 1	Line Compare Bit 0

Attribute Controller Addr/Data Flip-Flop Status Register 24h RO							
Index/Data	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved

8.2.7 CRTC Registers - Extended Set: Sprite/HW Cursor Controls

The Sprite/Hardware Cursor registers are accessed via the CRTC index/data registers (3B4h/3D4h and 3B5h/3D5h), at index 30h - 3Fh.

Sprite/HW Cursor X Position High Register 30h RW							
Reserved	Reserved	Reserved	Reserved	Reserved	Sprite/HW Cursor X Position Bit 10	Sprite/HW Cursor X Position Bit 9	Sprite/HW Cursor X Position Bit 8

bits 0-2 Sprite/HW Cursor X Position Bits [10:8]
This register contains the most significant three bits of the current horizontal left-most displayed pixel position of the Sprite/HW Cursor. A write to this register will latch in both Sprite/HW Cursor X Position Low and High registers.

Sprite/HW Cursor X Position Low Register 31h RW							
Sprite/HW Cursor X Position Bit 7	Sprite/HW Cursor X Position Bit 6	Sprite/HW Cursor X Position Bit 5	Sprite/HW Cursor X Position Bit 4	Sprite/HW Cursor X Position Bit 3	Sprite/HW Cursor X Position Bit 2	Sprite/HW Cursor X Position Bit 1	Sprite/HW Cursor X Position Bit 0

bits 0-7 Sprite/HW Cursor X Position Bits [7:0]
This register contains the least significant eight bits of the current horizontal left-most displayed pixel position of the Sprite/HW Cursor. For changes to the Sprite/HW Cursor X position to take effect this register must be written followed by a write to the Sprite/HW Cursor X Position High Register.

Sprite/HW Cursor Y Position High Register
32h RW

Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Sprite/HW Cursor Y Position Bit 9	Sprite/HW Cursor Y Position Bit 8

bits 0-1 Sprite/HW Cursor Y Position Bits [9:8]
This register contains the most significant two bits of the current horizontal upper-most displayed pixel position of the Sprite/HW Cursor. A write to this register will latch in both Sprite/HW Cursor Y Position Low and High registers.

Sprite/HW Cursor Y Position Low Register
33h RW

Sprite/HW Cursor Y Position Bit 7	Sprite/HW Cursor Y Position Bit 6	Sprite/HW Cursor Y Position Bit 5	Sprite/HW Cursor Y Position Bit 4	Sprite/HW Cursor Y Position Bit 3	Sprite/HW Cursor Y Position Bit 2	Sprite/HW Cursor Y Position Bit 1	Sprite/HW Cursor Y Position Bit 0

bits 0-7 Sprite/HW Cursor Y Position Bits [7:0]
This register contains the least significant eight bits of the current horizontal upper-most displayed pixel position of the Sprite/HW Cursor. For changes to the Sprite/HW Cursor Y Position to take effect this register must be written followed by a write to the Sprite/HW Cursor Y Position High Register.

X Display Start Column Register
34h RW

Reserved	Reserved	X Display Start Column Bit 5	X Display Start Column Bit 4	X Display Start Column Bit 3	X Display Start Column Bit 2	X Display Start Column Bit 1	X Display Start Column Bit 0

bits 0-5 X Display Start Column Bits [5:0]
This register specifies the first displayed column of the Sprite/HW Cursor. When this register is set to 0, there is no clipping of the left side of the Sprite/HW Cursor image.

Y Display Start Row Register
35h RW

Reserved	Reserved	Y Display Start Row Bit 5	Y Display Start Row Bit 4	Y Display Start Row Bit 3	Y Display Start Row Bit 2	Y Display Start Row Bit 1	Y Display Start Row Bit 0

bits 0-5 Y Display Start Row Bits[5:0]
This register specifies the first displayed row of the Sprite/HW Cursor. When this register is set to 0, there is no clipping of the top of the Sprite/HW Cursor image.

Sprite/HW Cursor Select High Register **36h RW**							
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Sprite/HW Cursor Select Bit 9	Sprite/HW Cursor Select Bit 8

bits 0-1 Sprite/HW Cursor Select Bits [9:8]
These are the most significant bits of the address location of the top left hand point of the Sprite/HW Cursor pattern stored in memory

Sprite/HW Cursor Select Low Register **37h RW**							
Sprite/HW Cursor Select Bit 7	Sprite/HW Cursor Select Bit 6	Sprite/HW Cursor Select Bit 5	Sprite/HW Cursor Select Bit 4	Sprite/HW Cursor Select Bit 3	Sprite/HW Cursor Select Bit 2	Sprite/HW Cursor Select Bit 1	Sprite/HW Cursor Select Bit 0

bits 0-7 Sprite/HW Cursor Select Bits [7:0]
These are the least significant 8 bits of the address location of the top left hand point of the HW Cursor pattern stored in memory. The complete 18 bit memory address for the Sprite/HW Cursor is formed as follows:

MA[17:8] Sprite/HW Cursor Select [9:0]
MA[7:2] Row [5:0]
MA[1:0] Column [1:0]

The row bits represent the current row of the Sprite/HW Cursor being fetched (0-63). The column bits represent the current column of the cursor being fetched. Each row consists of four columns. Each column contains 32 bits or 16 pixels. A single fetch from the 32-bit wide video memory obtains one column of one row of the cursor.

Sprite/HW Cursor Function Register **38h RW**							
Reserved	Reserved	Reserved	Reserved	Vertical Doubling Enable	Horizontal Doubling Enable	Sprite/HW Cursor Palette Select Bit 1	Sprite/HW Cursor Palette Select Bit 0

bits 0-1 Sprite/HW Cursor Palette Scheme Select Bits [1:0]
When these bits are set to 0, the Sprite/HW Cursor bits select one of the four Sprite/HW Cursor palette entries as the value of the currently displayed pixel. When these bits are set to 1, the Sprite/HW Cursor bit values 1 through 3 select Sprite/HW Cursor palette entries 1 through 3 respectively and Sprite/HW Cursor bit value 0 selects transparency, setting the current pixel to the value of the displayed image behind the Sprite/HW Cursor pixel. When these bits are set to 2, the Sprite/HW Cursor bit values 0 through 1 select Sprite/HW Cursor palette entries 0 through 1 respectively; Sprite/HW Cursor bit value 2 selects transparency, setting the current pixel to the value of the displayed image behind the Sprite/HW Cursor pixel; Sprite/HW Cursor bit value 3 selects inversion, setting the current pixel to the inverted value of the displayed image behind the Sprite/HW Cursor pixel. A Sprite/HW Cursor Palette Scheme Select value of 3 is reserved. See Sprite/Hardware Cursor on page 129 for details.

Table 8-4: Sprite/HW Cursor Palette Scheme Select

Pixel		**Sprite/HW Cursor Pixel Displayed**		
Bit 1	**Bit 0**	**CRTC[38h] b1:0 = 00b**	**CRTC[38h] b1:0 = 01b**	**CRTC[38h] b1:0 = 10b**
0	0	Sprite/HW Cursor Palette Entry 0	Screen Image	Sprite/HW Cursor Palette Entry 0
0	1	Sprite/HW Cursor Palette Entry 1	Sprite/HW Cursor Palette Entry 1	Sprite/HW Cursor Palette Entry 1
1	0	Sprite/HW Cursor Palette Entry 2	Sprite/HW Cursor Palette Entry 2	Screen Image
1	1	Sprite/HW Cursor Palette Entry 3	Sprite/HW Cursor Palette Entry 3	Inverted Screen Image

bit 2 Horizontal Doubling Enable
When this bit is set to 1, the displayed image of the Sprite/HW Cursor is doubled in the horizontal dimension, providing the Sprite/HW Cursor function is enabled. When this bit is set to 0, the Sprite/HW Cursor is not doubled. If the Sprite/HW Cursor function is not enabled, this bit has no effect.

bit 3 Vertical Doubling Enable
When this bit is set to 1, the displayed image of the Sprite/HW Cursor is doubled in the vertical dimension, providing the Sprite/HW Cursor function is enabled. When this bit is set to 0, the Sprite/HW Cursor is not doubled. If the Sprite/HW Cursor function is not enabled, this bit has no effect.

8.2.8 B-Set Registers

The B-set registers are accessed in CRTC register space by setting AUX[00h] b0, LCD B Registers Program Enable to 1. These registers are overlaid with A set registers and are only accessible if AUX[00h] b0 = 1.

B Horizontal Panel Timing Register **00h RW if AUX[00h] b0 = 1**							
Reserved	Reserved	LCD HSync Position Select Bit 2	LCD HSync Position Select Bit 1	LCD HSync Position Select Bit 0	LCD Non-Display Width Bit 2	LCD Non-Display Width Bit 1	LCD Non-Display Width Bit 0

bits 0-2 LCD Non-Display Width Bits [2:0]
Sets the width of the horizontal non-display period (HNDP) in 16 PClk increments when in LCD only modes:
Width HNDP = (LCD Non-Display Width + 1) *16 T_{PClk}

bits 3-5 LCD HSync Position Select Bits [2:0]
Sets the position of the HSync signal within the horizontal non-display period (HNDP) when in LCD only modes. A position is a 16 T_{PClk} increment in the HNDP. The first position is 0 and the last position is set by the LCD Non-Display Width bits [2:0].

B Horizontal Panel Size Register **01h RW if AUX[00h] b0 = 1**							
Horizontal Size Set B Bit 7	Horizontal Size Set B Bit 6	Horizontal Size Set B Bit 5	Horizontal Size Set B Bit 4	Horizontal Size Set B Bit 3	Horizontal Size Set B Bit 2	Horizontal Size Set B Bit 1	Horizontal Size Set B Bit 0

bits 0-7 Horizontal Size Set B Bits[7:0]
Specifies the horizontal size of the LCD panel in the number of 8-dot characters. When using standard IBM video modes, this register must be set to 50h or greater. For example; a 640 dot panel is programmed with the value 640/8 = 80 => 50h. If using non-IBM video modes this register can reflect the actual panel size (i.e., 320 dot/8 = 40 =>28h).

See Sub-VGA Resolution Application Note, X07-AN-009-01, for details.

This register is ignored if LCD mode is disabled.

B Vertical Total Register **06h RW if AUX[00h] b0 = 1**							
Vertical Total Set B Bit 7	Vertical Total Set B Bit 6	Vertical Total Set B Bit 5	Vertical Total Set B Bit 4	Vertical Total Set B Bit 3	Vertical Total Set B Bit 2	Vertical Total Set B Bit 1	Vertical Total Set B Bit 0

bits 0-7 Vertical Total Set B Bits [7:0]
These 8 LSBs are part of the 10 bit value that sets the Vertical Total while in any LCD Active display mode.

Program:
- 10-bit value of (full panel size + non-display period - 1) in Single Panel mode, e.g., 1E1h for 480 panel with 2 line non-display period (1/482 duty-cycle).
- 10-bit value of (half panel size + non-display period - 1) in Dual Panel mode, e.g. 0F1h for 1/242 duty-cycle, 0F3h for 1/244 duty-cycle.

B Overflow Register **07h RW if AUX[00h] b0 = 1**							
Vertical Retrace Start Set B Bit 9	Reserved	Vertical Total Set B Bit 9	Dummy R/W	Dummy R/W	Vertical Retrace Start Set B Bit 8	Reserved	Vertical Total Set B Bit 8

bits 0, 5 Vertical Total B Bits [9:8]

bits 2, 7 Vertical Retrace Start Set B Bits [9:8]

bits 3- 4 Dummy Bits
Read/Write only. These bits have no other effect in hardware.

B Vertical Retrace Start Register **10h RW if AUX[00h] b0 = 1**							
Vertical Retrace Start Set B Bit 7	Vertical Retrace Start Set B Bit 6	Vertical Retrace Start Set B Bit 5	Vertical Retrace Start Set B Bit 4	Vertical Retrace Start Set B Bit 3	Vertical Retrace Start Set B Bit 2	Vertical Retrace Start Set B Bit 1	Vertical Retrace Start Set B Bit 0

bits 0-7 Vertical Retrace Start Set B Bits [7:0]
These are the LSBs of a 10 bit value.
For TFT and simultaneous display modes - these bits are used to start the Vertical Retrace signal.
For CRT modes, these bits are ignored.

B Vertical Retrace End Register **11h RW if AUX[00h] b0 = 1**							
Reserved	Reserved	Reserved	Reserved	Vertical Retrace End Set B Bit 3	Vertical Retrace End Set B Bit 2	Vertical Retrace End Set B Bit 1	Vertical Retrace End Set B Bit 0

bits 0-3 Vertical Retrace End Set B Bits [3:0]
For TFT and simultaneous display modes - these bits are used to end the Vertical Retrace signal.
For CRT modes, these bits are ignored.

B Vertical Size Register **12h RW if AUX[00h] b0 = 1**							
Vertical Size Set B Bit 7	Vertical Size Set B Bit 6	Vertical Size Set B Bit 5	Vertical Size Set B Bit 4	Vertical Size Set B Bit 3	Vertical Size Set B Bit 2	Vertical Size Set B Bit 1	Vertical Size Set B Bit 0

bits 0-7 Vertical Size Set B Bits [7:0]
This register contains the 8 most significant bits of the 9-bit Number Of Lines in half the panel. In other words, this register contains the number of lines in the panel divided by four. This register is programmed the same for single and dual panels.

Example; for a 480 line panel this register is programmed with the 8 msb's of the 9-bit binary representation of 480/ 2 = 240 = 011110000b , so this register is programmed with 01111000b or 78h.

8.3 Page Register

This register is always read/write. This register is ignored unless the following conditions are met:

1. The Flat Addressing Enable bit (AUX[05h] b1) = 0.
2. Either the Memory Paging Enable bit (AUX[05h] b0) = 1 or the Graphics Controller Bypass Enable bit (AUX[05h] b4) = 1.

Page Select Register							
3CDh RW							
Read Page Bit 3	Read Page Bit 2	Read Page Bit 1	Read Page Bit 0	Write Page Bit 3	Write Page Bit 2	Write Page Bit 1	Write Page Bit 0

bits 0-3 Write Page Bits [3:0]
These four bits are used to relocate the VGA address space A0000h - AFFFFh to point to a 64KB write aperture within a 1MB memory space for all memory write operations.

bits 4-7 Read Page Bits [3:0]
These four bits are used to relocate the VGA address space A0000h - AFFFFh to point to a 64KB read aperture within a 1MB memory space for all memory read operations.

8.4 Auxiliary Registers

Auxiliary Index/Data Register							
3DEh RW 3DFh RW							
Auxiliary Index Bit 7	Auxiliary Index Bit 6	Auxiliary Index Bit 5	Auxiliary Index Bit 4	Auxiliary Index Bit 3	Auxiliary Index Bit 2	Auxiliary Index Bit 1	Auxiliary Index Bit 0

8.4.1 General Auxiliary Registers

Extended Function Register 0 **00h RW**							
Sprite Palette Enable	CRT Enable	LCD Enable	Extended MAC Enable	Palette Bypass Select	VL-Bus Snoop Disable	Test Mode Enable	LCD B-Set Registers Program Enable

bit 0 LCD B-Set Registers Program Enable
Selects between the A-set and B-set CRTC registers for read/write at the CRTC I/O address. When this bit = 0 the A-set registers are selected. When this bit = 1 then the B-set registers are selected.

bit 1 Test Mode Enable
For normal operation this bit must be set to 0. This bit is intended for diagnostic testing.

bit 2 VL-Bus Palette Snoop Disable
This bit disables the palette snoop function on the VL-Bus. When this bit = 1, all writes to the palette will be claimed by the on-chip VGA palette.

bit 3 Palette Bypass Select
This bit causes the VGA Palette to be bypassed. When this bit = 0, data from the VGA Palette is used as input to the LCD Interface module. When this bit = 1, data from the Attribute Controller bypasses the VGA Palette and is used as input to the LCD Interface module. This bit is intended for diagnostic testing.

bit 4 Extended Memory Address Counter (MAC) Enable
When this bit = 0 then only 256 Kbytes of video buffer are displayable; the Memory Address Counter wraps around at the 256Kbyte boundary. When this bit = 1 then 1 Mbyte of video buffer is displayable; the Memory Address Counter is extended by two bits, and the CRTC Start Address (CRTC registers 0Ch and 0Dh) is extended by AUX[05h] bits 3,2.

bits 5-6 LCD Enable
CRT Enable
Specifies the display type. See the following table.

Depending on the condition of the enable bits, the functionality of some VGA registers will change to dummy R/W registers that have no effect on hardware:

Table 8-5: LCD/CRT Enable Effects On VGA Registers

CRT Enable	LCD Enable	Active display	Registers Changed to Dummy R/W		
			Register	Index	Bit(s)
0	0	All displays off			
0	1	LCD Only	CRTC	00h	all
				02h	all
				03h	all
				04h	all
				05h	all
				06h	all
				07h	7, 5, 3, 2, 0
				09h	5
				10h	all
				11h	3, 2, 1, 0
				15h	all
				16h	all
			SEQ	01h	0
			AC	10h	2
				13h	3
1	0	CRT Only			
1	1	Simultaneous Display	CRTC	06h	all
				07h	7, 5 3, 2, 0
				09h	5
				10h	all
				11h	3, 2, 1, 0
				15h	all
			SEQ	01h	0
			AC	10h	2
				13h	3

When simultaneous display is active, all power save functions are disabled.

bit 7 Sprite Palette Enable
When this bit = 1, all access to the VGA palette will be interpreted as access to the sprite palette. The 6 MSBs of the palette address are ignored during sprite palette accesses.

Panel Configuration Register **01h RW**							
Blink Select Bit 1	Blink Select Bit 0	Block Text Cursor Override	Green/NTSC Weighting Select	Inverse Graphics Enable	Inverse Text Enable	Auto-Center Enable	Reserved

bit 0 Reserved - must be read back as 0.

bit 1 Auto-Centering Enable
This bit is used to control the auto-centering function which allows display modes with less than 480 lines to be vertically centered on the LCD panel. All the following conditions must be true for Auto-Centering enable to occur:

1) Auto-Centering Enable = 1
2) Vertical Expansion is inactive: see AUX[07h] b1,0
4) LCD Enable = 1, i.e. either LCD or simultaneous mode: see AUX[00h] b5

If all the conditions are met, then for all modes hardware will adjust the vertical position of the first line of the display so that the image is centered vertically on the panel.

bit 2 Inverse Text Enable
When this bit is 0, then normal text mode display attributes are enabled. When this bit is 1, and the SPC8110F0A is in text mode, then inverse video is displayed on the LCD display; the inversion takes effect on the output of the VGA palette. This bit has no effect on the image displayed on the CRT display.

bit 3 Inverse Graphics Enable
When this bit is 0, then normal graphics mode display attributes are enabled. When this bit is 1, and the SPC8110F0A is in graphics mode, then inverse video is displayed on the LCD display; the inversion takes effect on the output of the VGA palette. This bit has no effect on the image displayed on the CRT display.

bit 4 Green-Only/NTSC Gray-Scale Weighting Select
This bit is used to select one of two possible gray-scale weighting functions to be applied to the 18-bit Palette RGB data in Monochrome LCD mode. The weighting circuitry operates on the 18-bit RGB data output of the palette and converts it to 6-bit monochrome data before it is sent to the LCD Interface block. When this bit is set to 0, RGB data values are mapped to gray values using NTSC weighting. When this bit is set to 1, only the green component of the RGB data is used as the gray value and if only LCD is enabled (i.e. not simultaneous display), the display cycle palette access to the blue and red entries are shut down to conserve power.

bit 5 Block Text Cursor Override
This bit is used to select a more visible text cursor. When this bit = 1, CRTC[0Ah] and CRTC[0Bh] are ignored and the complete character box at the text cursor position alternates between normal and inverse video at the rate specified by the Blink Select bits (see below). When this bit = 0 the cursor is as per the VGA standard.

bits 6-7 Blink Select Bits [1:0]
These bits are used to select the blink rate of the cursor and text in text modes and graphics pixels in graphics modes, if blink is enabled. The blink rate is selected according to the following table. The VGA standard rates are active when these bits are both zero (first entry in the table).

Table 8-6: Blink Select Bits

Blink Select Bit 1	Blink Select Bit 0	Cursor Frequency	Text/Graphics Frequency
0	0	4 Hz	2 Hz
0	1	2 Hz	1 Hz
1	0	1 Hz	0.5 Hz
1	1	reserved - do not use this setting	

LCD Interface Register
02h RW

LP Width Select Bit 2	LP Width Select Bit 1	LP Width Select Bit 0	TFT Select	Panel Bus Size Select Bit 1	Panel Bus Size Select Bit 0	Color/Mono Panel Select	Single/Dual Panel Select

bit 0 Single/Dual Panel Select
The Single/Dual Panel bit is used to configure the chip timing for the correct LCD panel type. When b0 = 1, single panel mode is enabled, and when b0 = 0 dual panel mode is enabled.

bit 1 Color/Mono Panel Select
This bit is set to 0 for a monochrome panel and set to 1 for color panel support. See table below.

bits 2-3 Panel Bus Size Select Bits[1:0]
This bit selects the panel data bus size. See table below.

bit 4 TFT Select
When this bit is set to 1 TFT mode is selected.

bits 5-7 LP Width Select Bits[2:0]
These bits are used to adjust the LP latch pulse where width of LP = LP width Select[2:0] + 1 in Pixel Clocks

The following table shows how to set these bits for use with a specific LCD panel type:

Table 8-7: LCD Panel Selection Examples

LCD Panel Mode	Panel Bus Size		Color/ Mono	Single/ Dual	Half Size Panel	Example Panel(s)
	Bit 1	Bit 0				
16-bit dual color	1	1	1	0	0	Sharp LM64C15P Sharp LM64C08P Kyocera KCL6448HSTT
8-bit dual color (one shift clock)	0	0	1	0	0	
8-bit dual color (two shift clocks)	1	0	1	0	0	
6-bit dual color	0	1	1	0	0	
16-bit single color	1	1	1	1	0	Sanyo LM-CK53-22NEZ
8-bit single color	1	0	1	1	0	Sharp LM64C05P
8-bit single color half-panel	1	1	1	1	1	Optrex DMF-50268NCU-FW
8-bit dual monochrome	1	0	0	0	0	Sharp LM64P
8-bit single monochrome	1	0	0	1	0	
4-bit single monochrome	0	0	0	1	0	Sharp LM32P10 Sharp LM32008F

Half Size Panel comes from AUX[07h] b5.

The following table shows how to set these bits for use with a specific TFT panel type:

Table 8-8: TFT Panel Selection Examples

TFT Panel Mode	Color/ Mono	Panel Bus Size		Modulation	Example Panel(s)
		Bit 1	Bit 0		
18-bit color	1	1	1	0	Sharp LQ10D311
12-bit color	1	1	0	0	Hitachi TX26D52VC1CAA
9-bit color	1	0	1	0	Sharp LD5452 Sharp LQ10DH15 Hitachi TX26D51VC1CAA
9-bit color, modulated	1	0	1	1	same as 9-bit color above[1]

[1] AUX[02] b0 must be set to 1 (single panel select).

Memory Config Register 03h RW							
WE Control	Single DRAM Select	Self Refresh Support	DRAM Select	Wait State Sel Bit 1	Wait State Sel Bit 0	Refresh Rate Bit 1	Refresh Rate Bit 0

bits 0-1 Refresh Rate Bits[1:0]
These two bits are used to specify the refresh rate of the DRAM. These bits apply to normal CAS before RAS (CBR) as well as power down mode CBR refresh cycles. With RTCLK having a frequency of 32 kHz, the table below maps out the DRAM refresh specifications to the register bit value.

Table 8-9: DRAM Refresh Rate Selection

Refresh Rate		**Symmetrical DRAM (512 cycles)**	**Asymmetrical DRAM (1024 cycles)**
Bit 1	**Bit 0**		
0	0	8 ms	16 ms
0	1	64 ms	128 ms
1	0	128 ms	256 ms
1	1	256 ms	512 ms

bits 2-3 MClk/BClk Wait State Select [1:0]
These bits are used to select the number of wait states for handshaking between the memory clock (MClk) circuitry and the bus clock (BClk) circuitry.

Table 8-10: MClk/BClk Wait State Selection

Wait State Select		**Restrictions**	**Details**
Bit 1	**Bit 0**		
0	0	none	Lowest performance. Puts no restrictions on clock frequencies.
0	1	$T_M + 5.2\ ns \leq 2T_B$	Medium performance. Can be used if BClk <= 33MHz and MClk >= 20MHz.
1	0	$T_M + 5.2\ ns \leq T_B$	Highest performance. Can be used if BClk <= 33MHz and MClk >= 40MHz.
1	1	Reserved	Do not use this setting.

Where:
T_M = the period of MClk
T_B = the period of BClk

bit 4 DRAM Select
This bit selects asymmetrical DRAM support. When this bit is not set the 18-bit memory address is multiplexed into a 9-bit Row Address and a 9-bit Column Address which is placed on MA[8:0]. When this bit = 1, the 18-bit memory address is multiplexed into a 8-bit Row Address which is placed on MA[7:0] and a 10-bit Column Address which is placed on MA[9:0].

bit5 Self Refresh Support
This bit enables the MClk Off power down function (see Power Save Registers on page 96) to use the self refresh option. When this bit = 1, it indicates that the DRAM has CAS before RAS (CBR) self refresh capabilities. CBR self refresh capable DRAM is defined as a DRAM that does not require burst refresh cycles before entering and after leaving a self refresh mode.

bit 6 Single DRAM Select
When this bit = 0, then 1024Kbytes of DRAM is supported on MA[9:0], MDB[15:0], MDA[15:0], RAS#, CAS[3:0]#, WE#. When this bit = 1, then 512Kbytes of DRAM is supported on MA[9:0], MDA[15:0], RAS#, CAS[1:0]#, WE#. In Single DRAM mode, MDB[15:0], CAS[3:2]# should be open-circuited.

bit 7 WE Control
When this bit = 1, DRAM with a single CAS# and two WE# control lines are supported. If this bit = 0, DRAM with two CAS# and a single WE# control lines are supported.

Table 8-11: WE Control Selection

	Pin Name				
WE Control	**WE#**	**CAS0#**	**CAS1#**	**CAS2#**	**CAS3#**
0	WE#	CAS0#	CAS1#	CAS2#	CAS3#
1	WE0#	CAS#	WE1#	WE2#	WE3#

General Storage Register 0
04h RW

General Storage Bit 7	General Storage Bit 6	General Storage Bit 5	General Storage Bit 4	General Storage Bit 3	General Storage Bit 2	General Storage Bit 1	General Storage Bit 0

bits 0-7 General Storage Bits [7:0]
The General Storage Register 0 can be used to provide 8 bits of read/write temporary storage. These bits have no effect on hardware.

Extended Function Register 1
05h RW

High Res 256 Color Mode Enable	Fast Text Mode	Sprite/HW Cursor Enable	Graphics Controller Bypass Enable	Start Address SA17	Start Address SA16	Flat Buffer Addressing Enable	Memory Paging Enable

bit 0 Memory Paging Enable
This bit enables the Memory Paging function. Address space A0000h - AFFFFh is used as a relocatable 64KB memory read aperture and a separate relocatable 64KB memory write aperture into the full 512KB/1MB video buffer. I/O register 3CDh is used to select the location of the read and write apertures. If Flat Buffer Addressing Enable (see below) is set to 1, then this bit has no effect.

bit 1 Flat Buffer Addressing Enable
This bit enables Flat Addressing mode. If this bit = 1 then the whole video buffer is available in a contiguous 1 Mbyte memory address space. In PCI Bus mode, the location of this address space is defined in the configuration space. In VL-Bus mode, the location of this address space is found in the Video Buffer Location Register (AUX[1Ah] and AUX[1Bh]). This bit overrides the Memory Paging Enable bit.

bits 2-3 Start Address SA[17:16]
These bits are an extension to the CRTC Start Address registers (CRTC[0Ch] and CRTC[0Dh]). They can be used to set the display start address to anywhere in the 1MB address space. These bits are ignored (effectively zeroed) unless the Extended MAC Enable bit (AUX[00h] bit 4) = 1.

bit 4 Graphics Controller Bypass Enable
When this bit = 1, all CPU accesses to video memory bypass the read and write mode logic of the VGA graphics controller. Data presented on the host interface during CPU writes are presented to the memory without any manipulation. Data presented to the host interface during CPU reads come directly from the DRAM data bus.

bit 5 Sprite/HW Cursor Enable
This bit enables the sprite and hardware cursor functions. When this bit = 0, the sprite/hardware cursor function is disabled. When this bit = 1, the sprite/hardware cursor is enabled.

bit 6 Fast Text Mode
When this bit = 0 then standard VGA text mode addressing is implemented for the font (character generator) table stored in plane 2 of the video buffer, i.e. the font addressing is the following concatenation:
{MapSelect[2:0],CharacterCode[7:0],CharacterScan[4:0]}.
When this bit = 1 then a non-VGA text mode addressing scheme is implemented. The address location of the character code is relocated to provide efficient page mode memory access which allows high resolution text modes to be displayed. The font addressing is re-arranged into the following concatenation:
{MapSelect[2:0],CharacterScan[4:0],CharacterCode[7:0]}.

bit 7 High Resolution 256 Color Mode Enable
This bit is set to enable the high resolution 256 Color circuitry. This bit must be selected for all non-VGA standard 256 color modes. This bit overrides any text or graphics mode setting.
High Resolution 256 color mode uses the video buffer in packed pixel format, i.e., physical memory is contiguous. The following must be set for HR256 Mode:

1) HR256 Mode Select is enabled
2) Word/Byte Mode Select is set to Byte Mode, CRTC[17h] b6 = 1
3) Doubleword Mode is disabled, CRTC[14h] b6 = 0

If Flat Buffer Addressing is selected (AUX[05h] b1) then the whole 1MB video buffer is available to the CPU in a contiguous linear address starting at the position specified in the Video Buffer Location registers (AUX[1Ah] and AUX[1Bh]) for VL-Bus. The PCI address can be obtained from the configuration space.

If Memory Paging, AUX[05h] b0, is selected then the 1MB is accessed through two 64KB windows each addressed at A0000h to AFFFFh. One window is Read-only, the other Write Only. Page Register 3CDh is used to position these windows within the video buffer.

LCD WF Register **06h RW**							
Reserved	Reserved	WF Count Bit 5 or TFT Modulation Enable	WF Count Bit 4 or TFT Modulation Pattern Bit 1	WF Count Bit 3 or TFT Modulation Pattern Bit 0	WF Count Bit 2 or TFT Modulation Pattern Invert	WF Count Bit 1	WF Count Bit 0

Note

The function of bits 5:2 of this register depend upon the state of the TFT Select Bit (AUX[02] b4). If the TFT Select signal = 0, then the WF Count options apply. If the TFT Select signal = 1, then the TFT modulation options apply.

bits 0-5 WF Count Bits [5:0] (If TFT Select Bit = 0).
These bits are used to adjust the WF output signal period. The binary value stored in these bits represents the number of (LP pulses - 1) between toggles of the WF output. The power up reset value of these bits is 20h. A value of 0 programmed in these bits causes the WF output to toggle every frame. Values of 01h - 3Fh programmed in these bits result in WF toggling every 1+n LP pulses.

bit 2 TFT Modulation Pattern Invert (If TFT Select Bit = 1).
TFT modulation functions by rapidly switching between two 3 bit values (for each of red, green and blue pixels), in order to produce an intermediate color or grey shade value. When a modulation pattern other than 50%/50% of the two pixel values is used, this bit can be used to reverse the weighting of the two pixel values.

bits 3-4 TFT Modulation Pattern Bits [1:0] (If TFT Select Bit = 1).
These bits are used to select the TFT modulation pattern. The pattern determines the relative duration of the two values which are switched between rapidly to produce an intermediate value. The relative duration of the two values is as follows:

00b: 1/1
01b: 5/6
10b: 2/3
11b: 1/2

bit 5 TFT Modulation Enable (If TFT Select Bit = 1).
This bit turns on the TFT modulation option. Note that this feature is only available for 9-bit color and 3-bit monochrome TFT panels. If the TFT Modulation Enable Bit = 0, then 512 colors are possible. If the TFT Modulation Enable Bit = 1, then 4096 colors are possible. The single panel select bit, AUX[02h] b0, must also be set = 1.

LCD Display Register 07h RW							
32/64 Gray Select	256 Color Dithering Disable	LCD Half Size Panel	LCD Shift Clock Enable/ WF Select	9-bit x 2 TFT Select	Alternate LCD Pin Mapping Select	Graphics Vertical Expansion Enable	Text Vertical Expansion Enable

bit 0 Text Vertical Expansion Enable
This bit is used to enable vertical expansion of 400 line text modes on the LCD display, using selective line duplication. If this bit is 0, then vertical expansion of text modes is disabled. If this bit = 1, then 400 line text modes are vertically expanded to 475 lines. All the following conditions must be true for text expansion to occur:

1) Text Vertical Expansion Enable = 1
2) 16 point font is set, i.e. the 5 least significant bits of CRTC[09h] = 0Fh
3) 400 line mode is set, i.e. CRTC[12h] and associated overflow bits = 18Fh
4) text mode is set, i.e. GC[06h] b0 = 0.
5) LCD Enable = 1

Note that if these conditions are met, then the Auto Centering Enable bit, AUX[03h] b6, is ignored and auto centering is disabled.

bit 1 Graphics Vertical Expansion Enable
This bit is used to enable vertical expansion of 400 line graphics modes on the LCD display, using selective line duplication. If this bit = 0, then vertical expansion of graphics modes is disabled. If this bit = 1, then 400 line graphics modes are vertically expanded to fill the screen. This method of expansion will become active for text modes if Text Vertical Expansion is enabled and 16-point font is not set. All the following conditions must be true for graphics vertical expansion to occur:

1) Graphics Vertical Expansion Enable = 1
2) 400 line mode set, i.e. CRTC[12h] and associated overflow bits = 18Fh
3) either
graphics mode set, i.e., GC[06h] b0 = 1,
or
text mode set and Text Vertical Expansion enabled and 16 point font is not set
4) LCD Enable = 1

Note that if all conditions for graphics vertical expansion are met, then the Auto Centering Enable bit, AUX[03h] b6, is ignored and auto centering is disabled.

bit 2 Alternate LCD Pin Mapping Select
When this bit = 0, the LCD pin mapping follows the table Default Pin Mapping for the Various LCD and TFT Panel Options on page 26. When this bit = 1, the LCD pin mapping follows the table Alternate Pin Mapping for the Various LCD and TFT Panel Options on page 27.

bit 3 9-bit x 2 TFT Select
This bit selects the 9-bit x 2 TFT data format. See Figure 37: 2 x 9-bit TFT Panel Timing on page 59.

bit 4 LCD Shift Clock Enable/WF Select
This bit is used to select the function of the WF pin when a 6-bit dual panel color LCD is used. If this bit = 0, then the WF pin has its normal function. If the bit = 1, then the WF pin functions as a shift clock enable for the LCD panel.

bit 5 LCD Half Size Panel
This bit is used for LCD panels that operate as if they were the top half (8 bits) of a 16-bit dual panel color LCD.

bit 6 256 Color Dither Disable
This bit is used to control the dithering logic for the LCD display. When this bit = 0, dithering is enabled for 256 color mode (GC[05h] b6 = 1 or AUX[05h] b7 = 1) and disabled for other modes - this is the normal setting to provide 64 gray shades in 256 color mode. When this bit = 1, then dithering is disabled for 256 color mode. This bit has no effect on an image displayed on a CRT display.

bit 7 256 Color 32/64 Gray Select
This bit allows the selection of 32 or 64 gray shades on the monochrome LCD panel when in 256 color mode (GC[05h] b6 = 1 or AUX[05h] b7 = 1). When this bit = 0, 64 gray shades are displayed in 256 color mode (16 gray levels by FRM + dithering). When this bit = 1, then 32 gray shades are displayed in 256 color mode (16 gray levels by FRM + dithering). This bit has no effect on an image displayed on a CRT display.

Primary Revision Code Register
08h RO

Primary Revision Code Bit 2 RO	Primary Revision Code Bit 1 RO	Primary Revision Code Bit 0 RO	Reserved	Reserved	MONID2	MONID1	MONID0

bits 0-2 Monitor ID[2:0]
These three bits reflect the present state of the input pins MONID[2:0]. These pins are used to determine the type of monitor connected to the SPC8110F0A.

bits 5-7 Primary Revision Code
The Primary Revision Code are read-only bits permanently set to 1. The current revision code of the chip is a combination of the primary and secondary revision code values. The secondary revision code bits are contained in AUX[0Fh].

General Storage Register 1
0Ah RW

General Storage Bit 7	General Storage Bit 6	General Storage Bit 5	General Storage Bit 4	General Storage Bit 3	General Storage Bit 2	General Storage Bit 1	General Storage Bit 0

bits 0-7 General Storage Bits [7:0]
This register can be used to provide 8 bits of temporary read/write storage; they have no effect on hardware.

General Storage Register 2
0Bh RW

General Storage Bit 7	General Storage Bit 6	General Storage Bit 5	General Storage Bit 4	General Storage Bit 3	General Storage Bit 2	General Storage Bit 1	General Storage Bit 0

bits 0-7 General Storage Bits [7:0]
This register provides 8 bits of temporary read/write storage; they have no effect on hardware.

Configuration Readback Register 0 0Ch RO							
MDA15 Status on Reset RO	MDA14 Status on Reset RO	MDA13 Status on Reset RO	MDA12 Status on Reset RO	MDA11 Status on Reset RO	MDA10 Status on Reset RO	MDA9 Status on Reset RO	MDA8 Status on Reset RO

bits 0-7 MDA[15:8] Status on Reset
These are read-only bits which may be used for inputting power-up reset information to be read by software. On the rising edge of the RESET# input, the logic states on the MDA[15:8] pins are latched into the chip and then may be read in this register. MDA[15:8] are internally pulled low (see sections 5.1 and 5.2). These bits have no effect in hardware.

Configuration Readback Register 1 0Dh RW							
MDA7 Reserved RO	MDA6 Reserved RO	MDA5 Reserved	MDA4 Bus Select RO	MDA3 LCDBIAS Polarity RO	MDA2 LCDLOGIC Polarity RO	MDA1 LCDBACK Polarity RO	MDA0 Reserved

The state of pins MDA[7:0] are latched into this register on the rising edge of RESET# (see sections 5.1 and 5.2). MDA[7:0] are internally pulled low. The states of these register bits have effects on hardware operation. The latched default values of bits 5 and 0 can be overwritten by an IO write to this register.

bit 0 MDA0 Reserved
On the rising edge of RESET#, the logic value on the MDA0 pin is latched. This pin should be left unconnected, and this register bit should always be set to 0.

bits 1-3 MDA1: LCDBACK Polarity
MDA2: LCDLOGIC Polarity
MDA3: LCDBIAS Polarity
Bits 1-3 respectively select the active (On) polarity of the output pins: LCDBACK, LCDLOGIC and LCDBIAS. For example, if bit 1 is set to a logic 1 then LCDBACK will be driven high when active and low when inactive (powered down).

bits 4 MDA4: Bus Select
This read-only bit selects the CPU Bus type. On the rising edge of the RESET# input, the logic state on the MDA4 pin is latched into the chip and then may be read in this register. Internal pulldown resistors on this input pin ensure that if nothing is connected externally to this input, then this register bit will read 0. The following table shows the bus selection

Table 8-12: Interface Bus Type Select

MDA4 Bus Select	Bus Type
0	VL-32/Intel486 LB
1	PCI Bus

bit 5 MDA5: Reserved
On the rising edge of RESET#, the logic value on the MDA pins is latched. This pin should be left unconnected, and this register bit should always be set to 0.

bits 6-7 MDA[7:6] Reserved
On the rising edge of RESET#, the logic value on the MDA pins is latched. This pin should be left unconnected.

Secondary Revision Code Register 0Fh RO							
Product Code Bit 4	Product Code Bit 3	Product Code Bit 2	Product Code Bit 1	Product Code Bit 0	Revision Bit 2	Revision Bit 1	Revision Bit 0

bits 0-7 Secondary Revision Code Bits [7:0]
The Secondary Revision Code consists of a 5-bit Product Code and a 4-bit Revision Code (of which bit 3 is located at AUX [20] bit 5). The Product Code for the SPC8110F0A is 10100b. For the ES version of the SPC8110F0A the Revision Code is 1010b.

8.4.2 Power Save Registers

Power Save (PS) Pin and Timer Register 10h RW							
PS4PIN Polarity	PS1PIN Polarity	PS4PIN Status RO	PS1PIN Status RO	Timer Select	Time Out Bit 2	Time Out Bit 1	Time Out Bit 0

bits 0-2 Time Out Bit[0:2]
These three bits are used by the internal timer to define the period of inactivity before the timer enters a power save mode (PSM). With RTCLK having a frequency of 32 kHz, the table below maps out the period of inactivity to the register bit value.

Table 8-13: Time Out Select Value

Time Out [2:0]			**Time (approx.) (RTCLK=32 kHz)**
Bit 2	**Bit 1**	**Bit 0**	
0	0	0	1/2 sec
0	0	1	1 sec
0	1	0	4 sec
0	1	1	8 sec
1	0	0	16 sec
1	0	1	1 min
1	1	0	4 min
1	1	1	16 min

bit 3 Timer Select
When this bit = 0, PS1PIN is used to initialize PSM1.

When this bit = 1, the internal power-down timer is used as the initiator for PSM1. PS1PIN is used as an activity pin. Activity is defined as activity on PS1PIN, or CPU accesses to video controller memory and/or I/O space. Inactivity for a period equal to the time out period will initiate PSM1.

bits 4-5 PS1PIN Status
PS4PIN Status
These read-only bits reflect the state of external pins PS1PIN and PS4PIN. A logic 1 indicates that the pin is in an active state; the active polarity of the pin is selected by the bits below.

bits 6-7 PS1PIN Polarity
PS4PIN Polarity
These two bits select the active polarities of the PS1PIN and PS4PIN input pins.

Table 8-14: PSxPIN Active Polarity Selection

PSxPIN Polarity Register	Pin State	PSxPIN Status Register
0	0	1
0	1	0
1	0	0
1	1	1

Note

Refer to Power Save Modes (PSM) on page 133.

Power Save (PS) FunctionX Register, X= 1...4 **11h-14h RW**							
MClk Off	PClk Off	LCDBACK Off	DPMS Bit 1	DPMS Bit 0	Display Off	PClk Slow	Reduce Gray Scale

bit 0 Reduce Gray Scale
Valid for LCD only. The pixel path leading to the VGA palette is redirected to a gray scale reduction palette, bypassing the VGA palette entirely. Depending whether the Sprite/HW Cursor is currently being drawn, the gray-scale reduction palette takes pixel data from either the attribute controller block or the sprite/cursor block. It transforms the data into black, white or 50% gray and sends it to the LCD gray scale interface. Full CPU access to the VGA palette is maintained. The attributes controller, the LCD gray scale interface and the RAM palette are partially shut down by this function. Mapping of the gray scale reduction palette is tabled below:

Table 8-15: Gray Scale Reduction Palette Mapping

Pixel Source	Sprite/ Cursor Data	Display Data	Gray-Scale Reduction Output
Attribute	don't care	00h	black
		FFh	white
		01h-FEh	50% gray
Sprite 0	00b	don't care	black
	01b	don't care	white
	10b	don't care	50% gray
	11b	don't care	50% gray
Sprite 1	00b	00h	black
		FFh	white
		01h-FEh	50% gray
	01b	don't care	white
	10b	don't care	50% gray
	11b	don't care	50% gray
Cursor	00b	don't care	black
	01b	don't care	white
	10b	00h	black
		FFh	white
		01h-FEh	50% gray
	11b	00h	white
		FFh	black
		01h-FEh	50% gray

bit 1 PClk Slow

This function will select the Set1 PLL1 registers that are programmed with values that would lower the PClk frequency. This function will also select the appropriate slow dot fifo level (see FIFO Level Register section). The power consumption decreases in proportion to the frequency. The OEM must ensure that the reduction of the frame rate frequency is within the operating limits of the panel, and that the image displayed on the panel has an acceptable level of flicker. When coming out of a PSM, the PLL switches back to the original registers. Note that the PLL will take some time (on the order of milliseconds) for it to lock into a new frequency. This bit has no effect when CRT is enabled.

bit 2 Display Off

All circuits that contribute to the displaying of an image are shut down. The display FIFO, hardware cursor, frame buffer and the LCD gray-scale circuits become static. The VGA palette is partially shut down, while maintaining full CPU access. If CRT is active, the internal DAC goes into power down mode and the Red, Green, Blue signals are set to GND level. If the LCD is active, the panel is shut down with the panel power down sequence. The LCD data bus is set low.

When leaving a PSM, all affected circuits are re-activated. If the LCD is enabled, the panel is turned on with the panel power up sequence.

Sync signals to the CRT are not affected by this function.

bits 3-4 DPMS Bits [1:0]
Sets the VESA Display Power Management Signal protocol. This function affects the HSYNC and VSYNC signals connected to the CRT.

Table 8-16: DPMS Effects On VSYNC and HSYNC

DPMS		State	VSYNC	HSYNC
Bit 1	Bit 0			
0	0	ON	active	active
0	1	Stand-By	active	low
1	0	Suspend	low	active
1	1	OFF	low	low

VESA requires that the analog outputs Red, Green and Blue do not contain any picture information whenever DPMS is activated. In order to comply with this requirement, the Display Off function (see above) must be used in conjunction with the DPMS function.

bit 5 LCDBack Off
This function switches the LCDBACK pin to an inactive state to shut down the external LCD backlight. The active polarity of the LCDBACK is defined by a configuration bit on the memory data bus, latched in on the rising edge of RESET#.

bit 6 PClk Off (Pixel Clock)
This function is nearly identical to the Display Off function with two exceptions: circuitry that generates sync signals for the CRT are shut down and the PLL that supplies the pixel clock is shut down. Note that although the PLL is shut down, the associated registers are still R/W.

Note

This function will shut down all circuits that generate CRT sync signals. If a CRT is attached, the power management software must make sure that the CRT is either DPMS compatible or is capable of operating without any sync signals.

bit 7 MClk Off (Memory Clock)
This function has two options, depending whether self refresh DRAM support has been selected in the Memory Config Register. Note that although the PLL is shut down, the associated registers are still R/W.

Self Refresh Supported
Entering a PSM with this option will place the DRAM in self refresh mode and shut down PLL2 which drives MClk. When exiting the PSM, PLL2 is turned on. Once PLL2 is in a lock condition, an internal reset of critical state machines occurs. The DRAM is then brought out of self refresh and the normal refresh circuitry takes over. The system designer must ensure that RTCLK is stable and operating by the time the DRAM comes out of self refresh.

Self Refresh Not Supported
PLL2 is shut down and the refresh logic is switched over to a CAS before RAS pulse generating circuit driven by both edges of a 32 kHz, 50% duty cycle RTCLK (see section 7.7 on page 37). When exiting the PSM PLL2 is immediately turned on. When PLL2 has achieved a lock condition (order of milliseconds), an internal reset of critical state machines occurs. The normal refresh circuitry then resumes operation.

Any attempt to access memory by the CPU is ignored when this mode is active.

Power Save (PS) Enable Register **15h RW**							
PSM4 Enable	PSM3 Enable	PSM2 Enable	PSM1 Enable	PS Skew B Bit 1	PS Skew B Bit 0	PSM4 BusIF Off	PSM1 BusIF Off

bit 0 PSM1 BusIF Off
This is an extension to the PSM1 function register. When PSM1 is active all bus cycles except RESET# are ignored and the bus output buffers are tri-stated. LDEV# is forced high (in-active) in VL-Bus mode.

bit 1 PSM4 BusIF Off
This is an extension to the PSM4 function register. When PSM4 is active all bus cycles except RESET# are ignored and the bus output buffers are tri-stated. LDEV# is forced high (in-active) in VL-Bus mode.

bits 2-3 PS Skew B Bits [1:0]
These bits define the time period, t2, in frames (see Figure 39: LCD Panel Shut-Down Sequence on page 103) between LCD Output signals going inactive on a power off sequence.

Table 8-17: PS Skew B Time Period

PS Skew B		**t2 Frames**
Bit 1	**Bit 0**	
0	0	0
0	1	1
1	0	2
1	1	3

Note

Refer to the LCD panel specification for correct setting.

bits 4-7 PSMX Enable, X = 1, 2, 3, 4
Master switch for each PSM. If set, the PSM may be activated by its initiator, otherwise the PSM is completely disabled.

Power Save (PS) Skew Register **16h RW**							
PS Skew D Bit 2	PS Skew D Bit 1	PS Skew D Bit 0	PS Skew C Bit 1	PS Skew C Bit 0	PS Skew A Bit 2	PS Skew A Bit 1	PS Skew A Bit 0

Note

The PS Skew registers A, B, C and D should be written once during the power on intitialization sequence. They should not be subsequently modified.

bits 0-2 PS Skew A Bits[2:0]
These bits select the number of frames, t1, between LCDBIAS going inactive to LCD Output signals going inactive during the power down sequence. See PS Skew A High Register, AUX[17h], for table description.

bits 3-4 PS Skew C Bits[1:0]
These bits select the number of frames, t3, between LCDLOGIC going active to LCD Output signals going active during a power up sequence.

Table 8-18: PS Skew C Time Period

PS Skew C		t3 Frames
Bit 1	Bit 0	
0	0	0
0	1	1
1	0	2
1	1	3

Note

Refer to the LCD panel specification for correct setting.

bits 5-7 PS Skew D Bits[2:0]
These bits select the number of frames, t4, between LCD Output signals going active to LCDBIAS going active during a power up sequence.

Table 8-19: PS Skew D Time Period

PS Skew D			t4 Frames
Bit 2	Bit 1	Bit 0	
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Note

Refer to the LCD panel specification for correct setting.

Power Save (PS) Skew A High Register 17h RW							
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	PS Skew A Bit 4	PS Skew A Bit 3

bits 0-1 PS Skew A Bits[4:3]
These bits, together with the PS Skew Register bits, select the number of frames, t1, between LCDBIAS going inactive to LCD Output signals going inactive during a power down sequence

Table 8-20: PS Skew A Time Period

PS Skew A					**t1 Frames**
Bit 4	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	
0	0	0	0	0	1
0	0	0	0	1	2
0	0	0	1	0	3
0	0	0	1	1	4
0	0	1	0	0	5
0	0	1	0	1	6
0	0	1	1	0	7
0	0	1	1	1	8
etc.					
1	1	1	1	1	32

Note

This signal is intended to enable/disable the LCD panel BIAS voltage. The value programmed into this register is determined by the time required by the BIAS power supply to change from ON voltage to OFF voltage.

Panel Power Sequencing

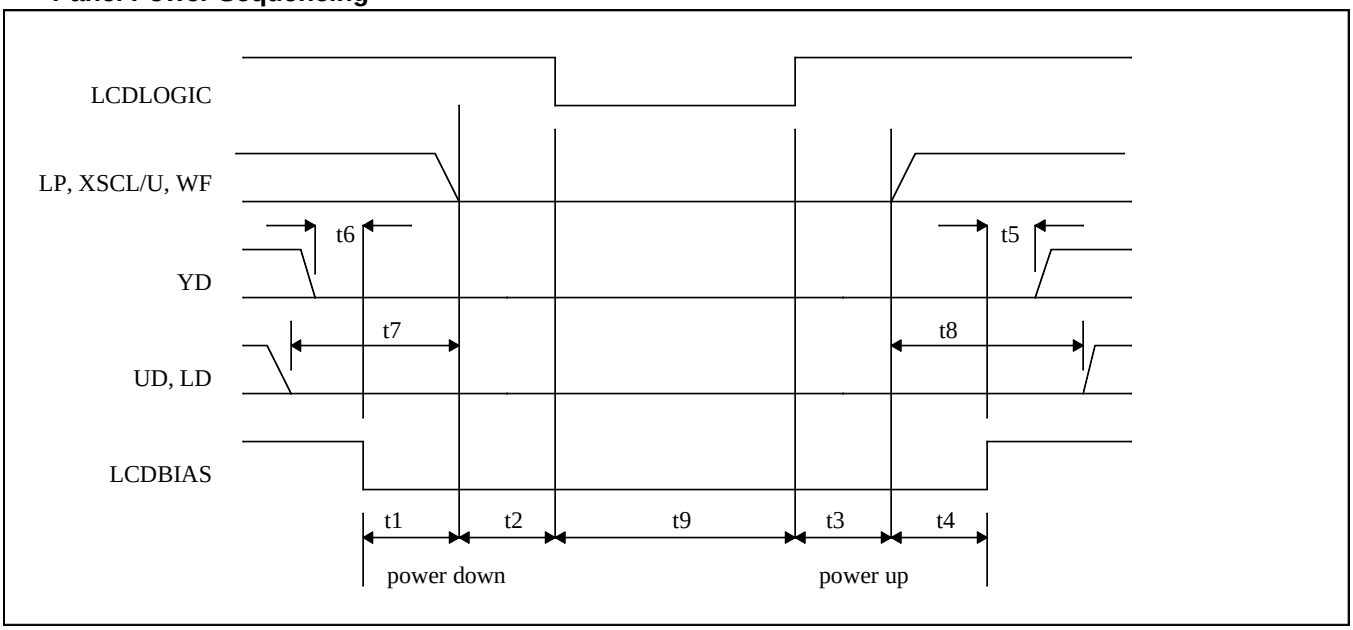


Figure 39: LCD Panel Shut-Down Sequence

Table 8-21: LCD Panel Shut-Down Sequence Timing

Symbol	Parameter	Value
t1	LCDBIAS inactive to Control Signals inactive	PSSkew A Register +1 frames
t2	Control Signals inactive to LCDLOGIC inactive	PSSkew B Register frames
t3	LCDLOGIC active to Control Signals active	PSSkew C Register frames
t4	Control Signals active to LCDBIAS active	PSSkew D Register frames
t5	LCDBIAS active to YD active	1 frame
t6	YD inactive to LCDBIAS inactive	1 frame - 1 line
t7	Data inactive to Control Signals Inactive	0ns min
t8	Control Signals active to Data active	0ns min
t9	Time between power cycles	controlled by software

Note

1. The active polarities of the LCDBIAS and LCDLOGIC pins are defined by configuration bits on the memory data bus, latched in on the rising edge of RESET#. In the diagram above, the LCDLOGIC and LCDBIAS signals are drawn as active high.

2. LCDBIAS, LCDLOGIC and LCDBACK (not shown) signals are held to their inactive states during reset.

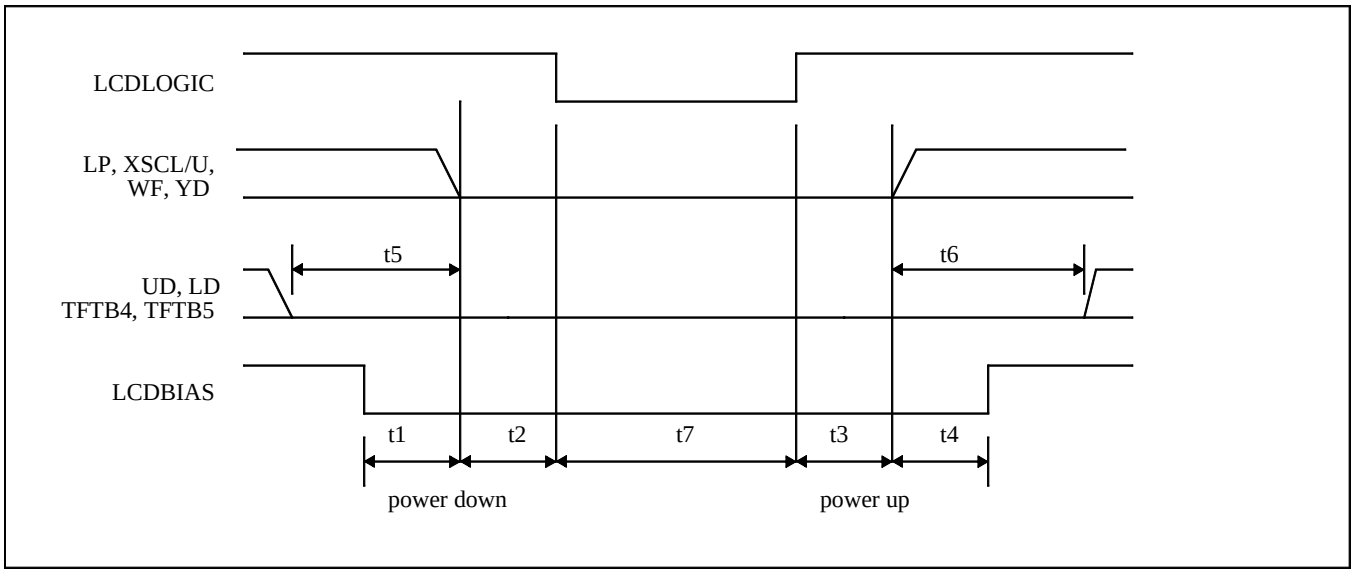


Figure 40: TFT Panel Shut-Down Sequence

Table 8-22: TFT Panel Shut-Down Sequence Timing

Symbol	Parameter	Value
t1	LCDBIAS inactive to Control Signals inactive	PSSkew A Register +1 frames
t2	Control Signals inactive to LCDLOGIC inactive	PSSkew B Register frames
t3	LCDLOGIC active to Control Signals active	PSSkew C Register frames
t4	Control Signals active to LCDBIAS active	PSSkew D Register frames
t5	Data inactive to Control Signals Inactive	0ns min
t6	Control Signals active to Data active	0ns min
t7	Time between power cycles	controlled by software

Note

1. The active polarities of the LCDBIAS and LCDLOGIC pins are defined by configuration bits on the memory data bus, latched in on the rising edge of RESET#. In the diagram above, the LCDLOGIC and LCDBIAS signals are drawn as active high.

2. LCDBIAS, LCDLOGIC and LCDBACK (not shown) signals are held to their inactive states during reset.

8.4.3 Frame and Video Buffer Registers

Video Buffer Location High Register 1Ah RW							
Reserved	Reserved	Reserved	Reserved	A31	A30	A29	A28

bits 0-3 Video Buffer Location A[31:28]
Only used during Flat Addressing Mode on the VL-Bus. See Video Buffer Location Low Register.

Video Buffer Location Low Register 1Bh RW							
A27	A26	A25	A24	A23	A22	A21	A20

bits 0-7 Video Buffer Location A[27:20]
Used only during Flat Addressing Mode on the VL-Bus. The Video Buffer Location Registers are used to position the Video Buffer when the chip is connected to a CPU with a 32-bit address bus. This address space must be above the 1MB boundary, and cannot be shared by any other devices. The video buffer location on the PCI Bus is specified in the PCI configuration space.

Frame Start Boundary Register 1Ch RW							
Reserved	Reserved	Reserved	Reserved	Frame Start Boundary Bit 3	Frame Start Boundary Bit 2	Frame Start Boundary Bit 1	Frame Start Boundary Bit 0

bits 0-3 Frame Start Boundary Bits [3:0]
Used only when the display is a dual LCD. This register defines the 64K boundary where the half frame information for the LCD display is to be stored. For VGA applications, the first four 64K boundaries are reserved for normal VGA display memory and should not be used. See Frame Length Low Register for further restrictions on programming these bits.

Table 8-23: Frame Start Boundary

Bits[3:0]	Memory	Bits[3:0]	Memory (1MB only)
0	0K-64K	8	512K-576K
1	64K-128K	9	576K-640K
2	128K-192K	A	640K-704K
3	192K-256K	B	704K-768K
4	256K-320K	C	768K-832K
5	320K-384K	D	832K-896K
6	384K-448K	E	896K-960K
7	448K-512K	F	960K-1024K

Frame Length High Register **1Dh RW**							
Frame Length Bit 15	Frame Length Bit 14	Frame Length Bit 13	Frame Length Bit 12	Frame Length Bit 11	Frame Length Bit 10	Frame Length Bit 9	Frame Length Bit 8

bits 0-7 Frame Length Bits [15:8]
See Frame Length Low Register.

Frame Length Low Register **1Eh RW**							
Frame Length Bit 7	Frame Length Bit 6	Frame Length Bit 5	Frame Length Bit 4	Frame Length Bit 3	Frame Length Bit 2	Frame Length Bit 1	Frame Length Bit 0

bits 0-7 Frame Length Bits [7:0]
Used only when the display is a dual LCD. The value programmed into these two registers determines the size of the half frame buffer.
Formula:

$$Length = \left(\frac{HxW}{2} \times Colors\right) / 32 - (2 \times Colors) - 1$$

where Colors = 1 for mono panel
3 for color panel
H = height of panel in pixels
W = width of panel in pixels

If Frame Length b14 = 1, Frame Start Boundary b0 must be set to 0. Use Frame Start Boundary b3 to specify the 128KB region occupied by the half frame buffer. If Frame Length b15 = 1, Frame Start Boundary b1:0 must be set to 0. Use Frame Start Boundary b3:2 to specify the 256KB region occupied by the half frame buffer.

Frame Buffer Disable Register **1Fh RW**							
Frame Buffer Disable	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved

bits 0-5 Reserved bits read back as 0

bit 7 Frame Buffer Disable
This bit is used to disable the half frame buffer. This is done to support some color dual panels in simultaneous display mode.

8.4.4 PLL Control Registers

For the following pair of registers sets, the value for the denominator must be written first, followed by the numerator. The value of the pairs do not come into effect until after a write to the numerator.

PLL1 Set 0 Numerator Register 20h RW							
Gain Select Bit 1	Gain Select Bit 0	Revision (read only) Bit 3	Reserved	PLL1 Set 0 Numerator Bit 3	PLL1 Set 0 Numerator Bit 2	PLL1 Set 0 Numerator Bit 1	PLL1 Set 0 Numerator Bit 0

bits 0-3 PLL1 Set 0 Numerator Bits [3:0]
Numerator register for the PLL first register set of the pixel clock.

bit 4 Reserved
This reserved bit must be set to 0

bit 5 Revision Bit 3
This is bit 3 of SPC8110F0A Revision Code (see *0Fh RO on page 96*). This bit must be set to 0 if written, however, it always returns a 1 when read.

bits 6-7 Gain Select Bits [1:0]
PLL 1 Set 0 Gain Select (see *Table 9-1:, "Recommended PLL Numerator, Denominator, and Gain Select Values," on page 120* for programming details).

PLL1 Set 0 Denominator Register 21h RW							
Reserved	Reserved	Reserved	Reserved	PLL1 Set 0 Denominator Bit 3	PLL1 Set 0 Denominator Bit 2	PLL1 Set 0 Denominator Bit 1	PLL1 Set 0 Denominator Bit 0

bits0-3 PLL1 Set 0 Denominator Bits [3:0]
Denominator bits for the PLL first register set of the pixel clock.

bits 4-7 Reserved
These reserved bits must be set to 0

PLL1 Set 1 Numerator Register 22h RW							
Gain Select Bit 1	Gain Select Bit 0	Reserved	Reserved	PLL1 Set 1 Numerator Bit 3	PLL1 Set 1 Numerator Bit 2	PLL1 Set 1 Numerator Bit 1	PLL1 Set 1 Numerator Bit 0

bits 0-3 PLL1 Set 1 Numerator Bits [3:0]
Numerator register for the PLL second register set of the pixel clock.

bits 4-5 Reserved
These reserved bits must be set to 0

bits 6-7 Gain Select Bits [1:0]
PLL 1 Set 1 Gain Select (see *Table 9-1:, "Recommended PLL Numerator, Denominator, and Gain Select Values," on page 120* for programming details).

PLL1 Set 1 Denominator Register
23h RW

Reserved	Reserved	Reserved	Reserved	PLL1 Set 1 Denominator Bit 3	PLL1 Set 1 Denominator Bit 2	PLL1 Set 1 Denominator Bit 1	PLL1 Set 1 Denominator Bit 0

bits 0-3 PLL1 Set 1 Denominator Bits [3:0]
Denominator register for the PLL second register set of the pixel clock.

bits 4-7 Reserved
These reserved bits must be set to 0

PLL1 Set 2 Numerator Register
24h RW

Gain Select Bit 1	Gain Select Bit 0	Reserved	Reserved	PLL1 Set 2 Numerator Bit 3	PLL1 Set 2 Numerator Bit 2	PLL1 Set 2 Numerator Bit 1	PLL1 Set 2 Numerator Bit 0

bits 0-3 PLL1 Set 2 Numerator Bits [3:0]
Numerator register for the PLL third register set of the pixel clock.

bits 4-5 Reserved
These reserved bits must be set to 0

bits 6-7 Gain Select Bits [1:0]
PLL 1 Set 2 Gain Select (see *Table 9-1:, "Recommended PLL Numerator, Denominator, and Gain Select Values," on page 120* for programming details).

PLL1 Set 2 Denominator Register
25h RW

Reserved	Reserved	Reserved	Reserved	PLL1 Set 2 Denominator Bit 3	PLL1 Set 2 Denominator Bit 2	PLL1 Set 2 Denominator Bit 1	PLL1 Set 2 Denominator Bit 0

bits 0-3 PLL1 Set 2 Denominator Bits [3:0]
Denominator register for the PLL third register set of the pixel clock.

bits 4-7 Reserved
These reserved bits must be set to 0

PLL2 Numerator Register
26h RW

Gain Select Bit 1	Gain Select Bit 0	Reserved	Reserved	PLL2 Numerator Bit 3	PLL2 Numerator Bit 2	PLL2 Numerator Bit 1	PLL2 Numerator Bit 0

bits 0-3 PLL2 Numerator Bits [3:0]
Numerator register for the memory clock PLL.

bits 4-5 Reserved
These reserved bits must be set to 0

bits 6-7 Gain Select Bits [1:0]
PLL 2 Gain Select (see *Table 9-1:, "Recommended PLL Numerator, Denominator, and Gain Select Values," on page 120* for programming details).

PLL2 Denominator Register 27h RW							
Reserved	Reserved	Reserved	Reserved	PLL2 Denominator Bit 3	PLL2 Denominator Bit 2	PLL2 Denominator Bit 1	PLL2 Denominator Bit 0

bits 0-3 PLL2 Denominator Bits [3:0]
Denominator register for the memory clock PLL.

bits 4-7 Reserved
These reserved bits must be set to 0

PLL Status Register 28h RW							
Reserved	Reserved	Reserved	Reserved	PLLx Lock Override	PLL Out Select	PLL2 Lock	PLL1 Lock

bits 0-1 PLL1 Lock
PLL2 Lock
An I/O read from these bits gives the effective lock status of each respective PLL. An I/O write to these bits is stored and, depending on the status of PLLx Lock Override (below), either is ignored or becomes the effective lock status of the respective PLL.

bit 2 PLL Out Select
When set, this bit will select the output of PLL1 (PClk) to be output at the GENIO1 pin, and the output of PLL2 (MClk) to be output at the GENIO3 pin. Note that GENIO1,3 must be configured as outputs.

bit 3 PLLx Lock Override
If this bit is 0 then the effective lock status of PLLx is determined by internal circuitry. If this bit is 1 then the effective lock status of PLLx is determined by the value of the last I/O write to PLLx Lock (see bits 0,1).

8.4.5 General I/O Registers

General Input Output Pin Register 34h RW							
GENIO[3] Output Enable	GENIO[2] Output Enable	GENIO[1] Output Enable	GENIO[0] Output Enable	GENIO[3] State	GENIO[2] State	GENIO[1] State	GENIO[0] State

bits 0-3 GENIO Bits [3:0] state
Reading: When reading these bits, the return value reflects the state of the respective GENIO pins.
Writing: Writing to these bits will set the output state of the corresponding GENIO pins. The GENIO Output Enable bits for the corresponding pins must be enabled. If the PLL Out Select bit is set in the PLL Status register, then the GENIO3 and GENIO1 state bits will have no effect.

bits 4-7 GENIO Bits [3:0] Output Enable
A zero value will set the corresponding GENIO pin to an input pin. A value of one will set the corresponding GENIO pin to an output pin.

8.4.6 FIFO Level Registers

Note

See “Programming FIFO Level Request Registers” on page 123.

VGA Graphics FIFO Level Register 38h RW							
Reserved	VGA Graphics FIFO High Level Bit 2	VGA Graphics FIFO High Level Bit 1	VGA Graphics FIFO High Level Bit 0	Reserved	VGA Graphics FIFO Low Level Bit 2	VGA Graphics FIFO Low Level Bit 1	VGA Graphics FIFO Low Level Bit 0

bits 0-2 VGA Graphics Mode FIFO Low Level Bits[2:0]
Low level FIFO marker used during standard VGA graphics mode.

bits 4-6 VGA Graphics Mode FIFO High Level Bits[2:0]
High level FIFO marker used during standard VGA graphics mode.

VGA Text FIFO Level Register 39h RW							
Reserved	VGA Text FIFO High Level Bit 2	VGA Text FIFO High Level Bit 1	VGA Text FIFO High Level Bit 0	Reserved	VGA Text FIFO Low Level Bit 2	VGA Text FIFO Low Level Bit 1	VGA Text FIFO Low Level Bit 0

bits 0-2 VGA Text Mode FIFO Low Level Bits[2:0]
Low level FIFO marker used during standard VGA text mode.

bits 4-6 VGA Text Mode FIFO High Level Bits[2:0]
High level FIFO marker used during standard VGA text mode.

Slow Dot Graphics FIFO Level Register 3Ah RW							
Reserved	Slow Dot Graphics FIFO High Level Bit 2	Slow Dot Graphics FIFO High Level Bit 1	Slow Dot Graphics FIFO High Level Bit 0	Reserved	Slow Dot Graphics FIFO Low Level Bit 2	Slow Dot Graphics FIFO Low Level Bit 1	Slow Dot Graphics FIFO Low Level Bit 0

bits 0-2 Slow Dot Graphics FIFO Low Level Bits[2:0]
Low level FIFO marker used during VGA slow dot graphics mode.

bits 4-6 Slow Dot Graphics FIFO High Level Bits[2:0]
High level FIFO marker used during VGA slow dot graphics mode.

Slow DotText FIFO Level Register 3Bh RW							
Reserved	Slow Dot Text FIFO High Level Bit 2	Slow Dot Text FIFO High Level Bit 1	Slow Dot Text FIFO High Level Bit 0	Reserved	Slow Dot Text FIFO Low Level Bit 2	Slow Dot Text FIFO Low Level Bit 1	Slow Dot Text FIFO Low Level Bit 0

bits 0-2 Slow Dot Text Mode FIFO Low Level Bits[2:0]
Low level FIFO marker used during VGA slow dot text mode.

bits 4-6 Slow Dot Text Mode FIFO High Level Bits[2:0]
High level FIFO marker used during VGA slow dot text mode.

Enhanced FIFO Level Register 3Ch RW							
Reserved	Enhanced FIFO High Level Bit 2	Enhanced FIFO High Level Bit 1	Enhanced FIFO High Level Bit 0	Reserved	Enhanced FIFO Low Level Bit 2	Enhanced FIFO Low Level Bit 1	Enhanced FIFO Low Level Bit 0

bits 0-2 Enhanced FIFO Low Level Bits[2:0]
Low level FIFO marker. See comment below.

bits 4-6 Enhanced FIFO High Level Bits[2:0]
High level FIFO marker. See comment below.

This register is selected when the Misc. Clock Select bit 1 = 1 and the PClkSlow power save function is not active.

Table 8-24: Selection Matrix of FIFO Registers

Misc. Clock Select Bit 1 = 1 and PClkSlow Not Active	**Graphics Mode AC[10h] b0**	**Slow Dot SEQ[01h] b3 or PClkSlow**	**Selected FIFO Register**
0	0	0	VGA Text
0	0	1	SD Text
0	1	0	VGA Graphics
0	1	1	SD Graphics
1	x	x	Enhanced

8.4.7 Auxiliary Enable Register

Auxiliary Enable Register **DEh RO**							
Auxiliary Enable Code Bit 7	Auxiliary Enable Code Bit 6	Auxiliary Enable Code Bit 5	Auxiliary Enable Code Bit 4	Auxiliary Enable Code Bit 3	Auxiliary Enable Code Bit 2	Auxiliary Enable Code Bit 1	Auxiliary Enable Code Bit 0

bits 0-7 Auxiliary Enable Code Bits[7:0] RO
Read back 1Ah.

8.5 BitBLT Registers

8.5.1 BitBLT Data Register (Memory Mapped)

This is a 32-bit memory mapped register that is used to transfer data to/from the CPU.

BitBLT CPU Data when 3D0h Bit 7 = 1 **Memory Address Axxxxh**							
Data Bit 31	Data Bit 30	Data Bit 29	Data Bit 28	Data Bit 27	Data Bit 26	Data Bit 25	Data Bit 24
Data Bit 23	Data Bit 22	Data Bit 21	Data Bit 20	Data Bit 19	Data Bit 18	Data Bit 17	Data Bit 16
Data Bit 15	Data Bit 14	Data Bit 13	Data Bit 12	Data Bit 11	Data Bit 10	Data Bit 9	Data Bit 8
Data Bit 7	Data Bit 6	Data Bit 5	Data Bit 4	Data Bit 3	Data Bit 2	Data Bit 1	Data Bit 0

bits 0-31 CPU BitBLT Data Bits [31:0]
For normal addressing mode, any read or write to memory location Axxxxh will be considered a BitBLT access whenever 3D0h bit 7 = 1.
When Flat Buffer Addressing is enabled (AUX[05h] b1 = 1), any read or write to a valid video memory location will be considered a BitBLT access whenever 3D0h bit 7 =1.

8.5.2 BitBLT Control Registers

The BitBLT control registers are 32 bits wide and reside in I/O address space 3D0h - 3D3h. 3D0h contains a 3-bit index register that selects the 24-bit indexed data register at 3D1h - 3D3h.

BitBLT Index Data Register **3D0h RW 3D1h-3D3h RW**							
BitBLT Status	Reserved	Reserved	Reserved	Reserved	BitBLT Index Bit 2	BitBLT Index Bit 1	BitBLT Index Bit 0

bits 0-2 BitBLT Index Bits [2:0]
Index value for 32-bit I/O space at 3D1h-3D3h.

bit 7 BitBLT Status
The BitBLT operation is started when this bit = 1. It is automatically reset when the BitBLT operation is finished. Software should use this bit to detect the end of a BitBLT operation. If software resets this bit while a BitBLT is in progress the BitBLT circuitry is reset and the operation is prematurely terminated.

BitBLT Source Address Register **00h RW**							
Reserved	Reserved	Reserved	Source Address Bit 20	Source Address Bit 19	Source Address Bit 18	Source Address Bit 17	Source Address Bit 16
Source Address Bit 15	Source Address Bit 14	Source Address Bit 13	Source Address Bit 12	Source Address Bit 11	Source Address Bit 10	Source Address Bit 9	Source Address Bit 8
Source Address Bit 7	Source Address Bit 6	Source Address Bit 5	Source Address Bit 4	Source Address Bit 3	Source Address Bit 2	Source Address Bit 1	Source Address Bit 0

bits 0-20 Source Address Bits [20:0]
This is the 21-bit source start address. This will be the upper left hand corner for a BitBLT in the positive direction, or the lower right hand corner for a BitBLT in the negative direction. In planar mode the three low order bits define the pixel position within the byte.

BitBLT Destination Address High Register **01h RW**							
Reserved	Reserved	Reserved	Destination Address Bit 20	Destination Address Bit 19	Destination Address Bit 18	Destination Address Bit 17	Destination Address Bit 16
Destination Address Bit 15	Destination Address Bit 14	Destination Address Bit 13	Destination Address Bit 12	Destination Address Bit 11	Destination Address Bit 10	Destination Address Bit 9	Destination Address Bit 8
Destination Address Bit 7	Destination Address Bit 6	Destination Address Bit 5	Destination Address Bit 4	Destination Address Bit 3	Destination Address Bit 2	Destination Address Bit 1	Destination Address Bit 0

bits 0-20 Destination Address Bits [20:0]
This is the 21-bit destination start address. This will be the upper left hand corner for a BitBLT in the positive direction, or the lower right hand corner for a BitBLT in the negative direction. In planar mode the three low order bits define the pixel position within the byte.

BitBLT Height and Background Color Register **02h RW**							
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Height Bit 9	Height Bit 8
Height Bit 7	Height Bit 6	Height Bit 5	Height Bit 4	Height Bit 3	Height Bit 2	Height Bit 1	Height Bit 0
Background Color Bit 7	Background Color Bit 6	Background Color Bit 5	Background Color Bit 4	Background Color Bit 3	Background Color Bit 2	Background Color Bit 1	Background Color Bit 0

bits 0-7 Background Color Bits [7:0]
This is the background color value used in color expansion. This is a 4-bit register in planar mode and a 8-bit register in packed pixel mode.

bits 8-17 Height Bits [9:0]
Height in lines of the BitBLT rectangle. This register is programmed with the actual height minus one. In the case of a linear BitBLT the best performance is achieved by programming the rectangle to be as wide as possible.

BitBLT Width and Foreground Color Register **03h RW**							
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Width Bit 9	Width Bit 8
Width Bit 7	Width Bit 6	Width Bit 5	Width Bit 4	Width Bit 3	Width Bit 2	Width Bit 1	Width Bit 0
Foreground Color Bit 7	Foreground Color Bit 6	Foreground Color Bit 5	Foreground Color Bit 4	Foreground Color Bit 3	Foreground Color Bit 2	Foreground Color Bit 1	Foreground Color Bit 0

bits 0-7 Foreground Color Bits[7:0]
This is the foreground color value used in color expansion. This is a 4-bit register in planar mode and a 8-bit register in packed pixel mode.

bits 8-17 Width Bits[9:0]
Width in pixels of the BitBLT rectangle. This register is programmed with the actual width minus one. In the case of a linear BitBLT the best performance is achieved by programming the rectangle to be as wide as possible.

BitBLT Description Register 04h							
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
Direction	Destination Linear Select	Source Linear Select	Reserved	Reserved	BitBLT Operation Select Bit 2	BitBLT Operation Select Bit 1	BitBLT Operation Select Bit 0

bit 0-2 BitBLT Operation Select Bits [2:0]

Table 8-25: BitBLT Operation Select

BitBLT Operation Select			**BitBLT Operation**
Bit 2	**Bit 1**	**Bit 0**	
0	0	0	Write BitBLT
0	0	1	Write BitBLT with Color Expansion
0	1	0	Solid fill
0	1	1	Move BitBLT
1	0	0	Pattern fill
1	0	1	Read BitBLT
1	1	0	Destination invert BitBLT
1	1	1	Reserved

bit 5 Source Linear Select
Selects the source to be stored in a contiguous linear memory location when high. When this bit is low, selects the source to be a rectangular region of memory; the Offset register of CRTC[13h] is used to determine the address offset from the start of one line to the next.

bit 6 Destination Linear Select
Selects the destination to be stored in a contiguous linear chunk of memory when high. When this bit is low, selects the destination to be a rectangular region of memory; the Offset register of CRTC[13h] is used to determine the address offset from the start of one line to the next.

bit 7 Direction
When this bit is zero, the BitBLT operation is in the positive direction (i.e., from top left to bottom right). When set to one, the BitBLT operation is in the negative direction (i.e., from bottom right to top left).

BitBLT Function Register **05h RW**							
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
Reserved	Reserved	Auto Update Select	Auto Start Select	Reserved	Reserved	Transparent Background Select	Translate Select

bit 0 Translate Select
4 bit/pixel video information is automatically translated from packed-pixel format from the host to planar format for the video buffer and vice versa. This bit is ignored unless either a read or write BitBLT is selected (BitBLTOpSel = 000b or 101b).

bit 1 Transparent Background Select
Selects Color Expansion with transparency. If selected, incoming monochrome data selects the foreground Color register when high and leaves the destination data untouched when low. This bit is ignored unless a write BitBLT with Color Expansion is selected (BitBLTOpSel = 001b).

bit 4 Auto Start Select
Enables the automatic start feature. The BitBLT is started as soon as the Width register is written. This is useful in conjunction with the automatic update feature for text out situations.

bit 5 Auto Update Select
Enables automatic destination update. The destination address register is automatically updated at completion of the BitBLT to point to the position to the immediate right of the last BitBLT rectangle. This is useful in a text out situation.

8.6 PCI Configuration Registers

Vender ID Register [15:0] **00h RO**							
Vender ID Bit 15	Vender ID Bit 14	Vender ID Bit 13	Vender ID Bit 12	Vender ID Bit 11	Vender ID Bit 10	Vender ID Bit 9	Vender ID Bit 8
Vender ID Bit 7	Vender ID Bit 6	Vender ID Bit 5	Vender ID Bit 4	Vender ID Bit 3	Vender ID Bit 2	Vender ID Bit 1	Vender ID Bit 0

bits 0-15 Vender ID [15:0]
Hard coded to 10F4h.

Device ID Register [31:16] **00h RO**							
Device ID Bit 15	Device ID Bit 14	Device ID Bit 13	Device ID Bit 12	Device ID Bit 11	Device ID Bit 10	Device ID Bit 9	Device ID Bit 8
Device ID Bit 7	Device ID Bit 6	Device ID Bit 5	Device ID Bit 4	Device ID Bit 3	Device ID Bit 2	Device ID Bit 1	Device ID Bit 0

bits 0-15 Device ID [15:0]
Hard coded to 8110h.

Command Register [15:0] **04h RW**							
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
Reserved	Reserved	VGA Snoop Enable	Reserved	Reserved	Reserved	Enable Memory Access	Enable I/O Access

bit 0 Enable IO Access
A value of 1 for this bit enables I/O access. A value of 0 disables all I/O access attempts.

bit 1 Enable Memory Access
A value of 1 for this bit enables valid memory access. A value of 0 disables all memory access attempts.

bit 5 VGA Snoop Enable
When this bit is enabled the PCI bus will not be claimed whenever a write access is made to the palette. When reset, this bit causes all palette accesses to be treated the same as all other accesses.

The PCI snooping circuitry has the following restrictions:

1 The PCI target designated to claim palette access must be a slow or subtractive address decoder. This is to prevent bus collisions of the DEVSEL# line in the situation where the SPC8110F0A attempts a retry or claims the bus during reads.

2 The PCI target that claims the palette access must not assert TRDY# until at least two BClks after IRDY# is sampled low. See diagram below.

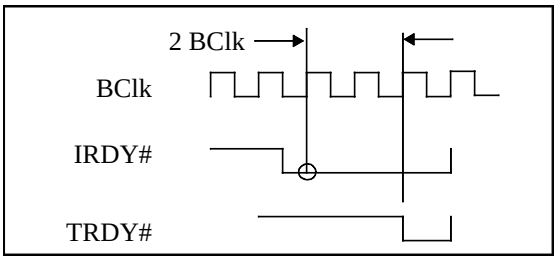


Figure 41: PCI Target Device Minimum Assertion Time for TRDY# During Palette Snoop

Status Register [31:16] **04h RW**							
Reserved	Reserved	Reserved	Reserved	Target Abort	DEVSEL Timing Bit 1 RO as 0	DEVSEL Timing Bit 0 RO as 1	Reserved
Fast Back-to-Back RO as 1	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved

bit 7 Fast Back-to-Back Capable
Hard coded to 1, indicating fast back-to-back transactions are permitted.

bits 9-10 DEVSEL Timing [1:0]
These bits are hard coded to 01, indicating medium speed decode.

bit 11 Target Abort
This bit is set by the SPC8110F0A whenever a target abort is performed. This bit can only be reset by writing a 1 to this location.

Revision ID Register [7:0] **08h RO**							
Revision ID Bit 7	Revision ID Bit 6	Revision ID Bit 5	Revision ID Bit 4	Revision ID Bit 3	Revision ID Bit 2	Revision ID Bit 1	Revision ID Bit 0

bits 0-7 Revision ID [7:0]
Bits [7:3] are set to 0. Bits [2:0] are the same as the Secondary Revision register AUX[0Fh] b2:0, which is currently set to 1.

Class Code Register [31:8] **08h RO**							
Class Code Bit 31 RO as 0	Class Code Bit 30 RO as 0	Class Code Bit 29 RO as 0	Class Code Bit 28 RO as 0	Class Code Bit 27 RO as 0	Class Code Bit 26 RO as 0	Class Code Bit 25 RO as 1	Class Code Bit 24 RO as 1
Class Code Bit 23 RO as 0	Class Code Bit 22 RO as 0	Class Code Bit 21 RO as 0	Class Code Bit 20 RO as 0	Class Code Bit 19 RO as 0	Class Code Bit 18 RO as 0	Class Code Bit 17 RO as 0	Class Code Bit 16 RO as 0
Class Code Bit 15 RO as 0	Class Code Bit 14 RO as 0	Class Code Bit 13 RO as 0	Class Code Bit 12 RO as 0	Class Code Bit 11 RO as 0	Class Code Bit 10 RO as 0	Class Code Bit 9 RO as 0	Class Code Bit 8 RO as 0

bits 0-23 Class Code [31:8]
This register is hard coded to 030000h, which translates to VGA compatible display controller.

Base Address Register [31:0] **10h RW**							
Base Address Bit 31	Base Address Bit 30	Base Address Bit 29	Base Address Bit 28	Base Address Bit 27	Base Address Bit 26	Base Address Bit 25	Base Address Bit 24
Base Address Bit 23	Base Address Bit 22	Base Address Bit 21	Base Address Bit 20	Base Address Bit 19 RO as 0	Base Address Bit 18 RO as 0	Base Address Bit 17 RO as 0	Base Address Bit 16 RO as 0
Base Address Bit 15 RO as 0	Base Address Bit 14 RO as 0	Base Address Bit 13 RO as 0	Base Address Bit 12 RO as 0	Base Address Bit 11 RO as 0	Base Address Bit 10 RO as 0	Base Address Bit 9 RO as 0	Base Address Bit 8 RO as 0
Base Address Bit 7 RO as 0	Base Address Bit 6 RO as 0	Base Address Bit 5 RO as 0	Base Address Bit 4 RO as 0	Base Address Bit 3 RO as 0	Base Address Bit 2 RO as 0	Base Address Bit 1 RO as 0	Base Address Bit 0 RO as 0

bits 0-19 Base Address [19:0]
These bits are read only, and are set to zero.

bits 20-31 Base Address [31:20]
These bits represent the upper twelve bits of the video memory address location when the flat buffer AUX[05h] b1 = 1.

9 ARCHITECTURE DESCRIPTION

9.1 PLL

9.1.1 Functional Overview

The PLL is a feedback analog device that synthesizes a square wave clock to drive the internal circuits. There are essentially five sections within the PLL: two clock dividers, a phase detector, a charge pump and a voltage controlled oscillator (VCO). The clock as seen by the internal circuits comes from the VCO.

An external reference clock, REFCLK, is divided by one of the clock dividers. The clock coming out of the VCO is divided by the other clock divider. The two divided clocks are phase compared in the phase comparator. The difference in frequency is communicated to the charge pump, which, depending on the polarity and magnitude of the difference, will incrementally increase or decrease a current that is fed to an external loop capacitor. The VCO sees the voltage across the external capacitor and generates a frequency proportional to the voltage. The feedback of the device is complete when the output of the VCO is divided and is phase compared to the divided down reference clock.

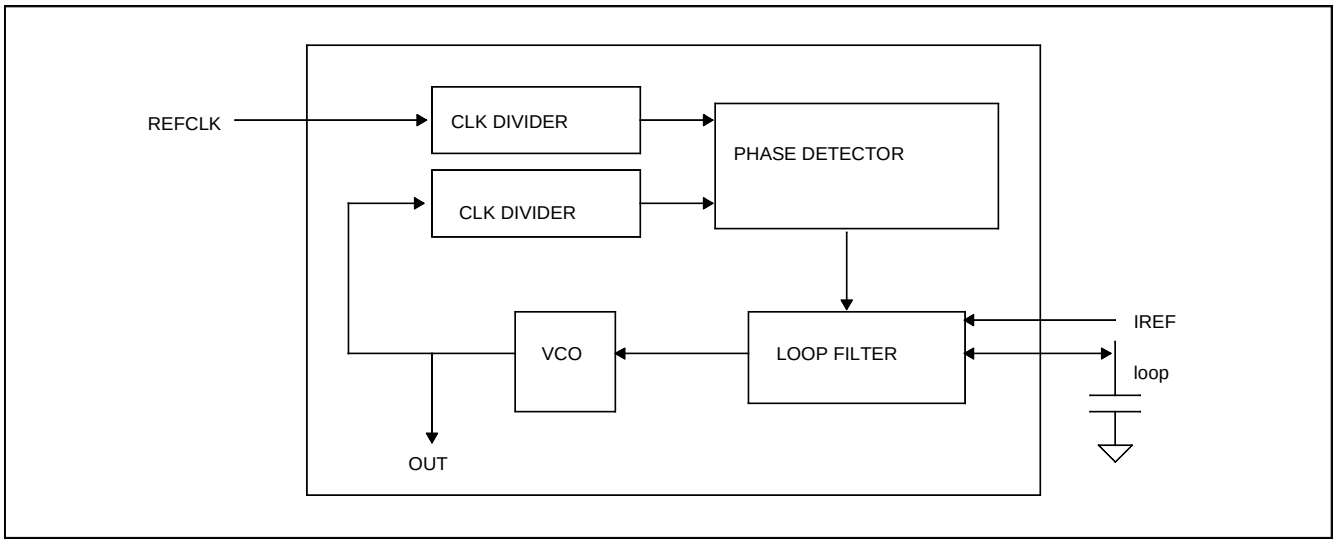


Figure 42: PLL Block Diagram

By changing the clock divide down values, the PLL is capable of producing a very large range of frequencies from a single clock source. Two registers, numerator and denominator, are required to define the output frequency. The relationship between these two registers, the output frequency and the reference frequency is as follows:

$$f_{out} = f_{reference} \times \frac{(numerator)}{(denominator)}$$

In the SPC8110F0A there are two PLLs, each one fully independent of the other in operation. Each has its own loop capacitor and current reference, but both share the same reference clock. PLL1 is the device that drives the pixel clock. PLL2 is the device that drives the memory clock.

Note

Gain select bit 0 should always be set to 1 for $f_{out} \leq 51$ MHz.
Gain select bit 1 should always be set to 1 for $f_{out} \leq 31$ MHz.

Table 9-1: Recommended PLL Numerator, Denominator, and Gain Select Values

F_{out} (f_{ref}=14.32 MHz)	Numerator (Hex)	Denominator (Hex)	Gain Select Bit 1	Gain Select Bit 0
14.3	1	1	1	1
25.1	7	4	1	1
28.6	2	1	1	1
32.2	9	4	0	1
35.8	A	4	0	1
39.4	B	4	0	1
50.1	7	2	0	1
65	9	2	0	0

Programming PLL1 (Pixel Clock)

The pixel clock has three identical sets of registers. Each set completely defines the operating conditions of the PLL for a frequency. The three sets are required for several reasons, depending on the display mode. The table below suggests the frequencies that should be programmed into the registers, based on the current display mode. The control column specifies the register that will select which set is to control the PLL.

Table 9-2: Recommended PLL Register Programmed Frequencies

Display Mode	Set 0 (MHz)	Set 1 (MHz)	Set 2 (MHz)	Control
S/VGA CRT	25.175	28.6	max. 65	MOR b3:2
simultaneous display	25.175	not used	not used	Locked Out
Dual LCD	14.161	9.5	not used	Power Save Block
Single LCD	28.6	19	not used	Power Save Block
TFT	25.175	19	not used	Power Save Block

The following two tables show the maximum PCLK and therefore the maximum frame-rate that can be used for any given display mode and still guarantee correct FIFO operation. The actual FIFO values must still be calculated using the formula found in the section titled “Programming the PLL and FIFO” of the Programming Notes and Examples manual (X07-AN-004-0x).

Table 9-3: 32-bit Memory Data Width

Video Mode is in ( )		CRT / TFT			Single Panel			Dual Color			Dual Monochrome		
		640x480	800x600	1024x768	640x480	800x600	1024x768	640x480	800x600	1024x768	640x480	800x600	1024x768
VGA Slow Text (0,1)	**PCLK (max)**	65 MHz			33 MHz			33 MHz			33 MHz		
	Frame Rate	161	97	62	93	61	Not Supported	186	122	76	186	122	76
VGA Slow Graphics (4,5)	**PCLK (max)**	65 MHz			33 MHz			33 MHz			33 MHz		
	Frame Rate	161	97	62	93	61	Not Supported	186	122	76	186	122	76
VGA Text (2,3,7)	**PCLK (max)**	35.56 MHz			33 MHz			24.62 MHz			29.09 MHz		
	Frame Rate	89	53	Not Supported	93	61	Not Supported	138	91	57	164	107	67
VGA Graphics (10,11,12, 13)	**PCLK (max)**	65 MHz			33 MHz			33 MHz			33 MHz		
	Frame Rate	161	97	62	93	61	Not Supported	186	122	76	186	122	76
256 Color	**PCLK (max)**	65 MHz			33 MHz			32 MHz			33 MHz		
	Frame Rate	161 (101)	97 (103)	62 (105)	93 (101)	61 (103)	Not Supported	180 (101)	118 (103)	74 (105)	186 (101)	122 (103)	76 (105)
Enhanced Text	**PCLK (max)**	53.33 MHz			33 MHz			32 MHz			33 MHz		
	Frame Rate	133 (10A)	80 (10B)	51 (10C)	93 (10A)	61 (10B)	Not Supported	180 (10A)	118 (10B)	74 (10C)	186 (10A)	122 (10B)	76 (10C)

Table 9-4: 16-bit Memory Data Width

Video Mode is in ( )		CRT / TFT			Single Panel			Dual Color			Dual Monochrome		
		640x480	800x600	1024x768	640x480	800x600	1024x768	640x480	800x600	1024x768	640x480	800x600	1024x768
VGA Slow Text (0,1)	**PCLK max**	65 MHz			33 MHz			27.83 MHz			33 MHz		
	Frame Rate	161	97	62	93	61	Not Supported	156	103	64	186	122	76
VGA Slow Graphics (4,5)	**PCLK max**	65 MHz			33 MHz			33 MHz			33 MHz		
	Frame Rate	161	97	62	93	61	Not Supported	186	122	76	186	122	76
VGA Text (2,3,7)	**PCLK max**	35.56 MHz			33[1] MHz			20.00 MHz			26.67 MHz		
	Frame Rate	89	53	Not Supported	93	61	Not Supported	112	74	Not Supported	150	99	62
VGA Graphics (10,11,12, 13)	**PCLK max**	65 MHz			33 MHz			26.67 MHz			33 MHz		
	Frame Rate	161	97	62	93	61	Not Supported	150	99	62	186	122	76
256 Color	**PCLK max**	38.10 MHz			33 MHz			18.82 MHz			24.62 MHz		
	Frame Rate	95 (101)	57 (103)	Not Supported	93 (101)	61 (103)	Not Supported	106 (101)	69 (103)	Not Supported	138 (101)	91 (103)	57 (105)
Enhanced Text	**PCLK max**	53.33 MHz			33 MHz			24.62 MHz			33 MHz		
	Frame Rate	133 (10A)	80 (10B)	51 (10C)	93 (10A)	61 (10B)	Not Supported	138 (10A)	91 (10B)	57 (10C)	186 (10A)	122 (10B)	76 (10C)

Where:

$$FrameRate = \frac{PCLK_{max}}{\{(hdp + hndp) \times (vdp + vndp)\}}$$

hdp: horizontal display period
hndp: horizontal non-display period
vdp: vertical display period
vndp: vertical non-display period

Note

The above values are set as per VESA requirements.

The frame-rate calculations above are based on the following CRTC values;

Table 9-5: CRTC Scan Resolution Totals

Image Resolution	CRT / TFT	Single Passive	Dual Passive
640x480	765x525	736x482	736x241
800 x 600	1005x666	896x602	896x301
1024x768	1309x806	1120x770	1120x385

The horizontal total {CRTC A[00] + 5} * 8 = hdp + hndp for CRT mode.
The vertical total {CRTC A[07] bits 5,0 + CRTC A[06]} + 2 = vdp + vndp for CRT mode.
The horizontal total for LCD modes = {{CRTC B[01]} * 8} + {{CRTC B [00] bits 0,1,2} + 1}* 16.
The vertical total for LCD modes = {CRTC B [07] bits 5,0 + CRTC B[06]} + 1.
The PCLK from the tables combined with the physical memory size will determine whether the mode is supported.
Memory Size = hdp * vdp * (bytes per pixel).

Bytes-per-pixel is determined by the color depth (i.e., 256 color modes require 8 bits-per-pixel, whereas 16 color modes only require 4 bits-per-pixel, thus requiring half the memory).

Notes

1. Dual panel operation requires using the Half Frame Buffer. Because of this, dual panel operation is limited by physical memory size. The Half Frame Buffer requires 64K, 128K, 256K (640x480, 800x600,1024x768) memory therefore limiting resolution based on physical memory size. The frame rates listed above do not reflect whether there is enough memory for that specific mode.
2. Disabling the Half Frame Buffer, although does work, may reduce contrast to an unacceptable level.
3. TFT panels do not exhibit any visual quality increase at frame-rates higher than 50Hz, therefore they can be run at lower frequencies thus reducing EMI problems.
4. To determine the resultant XSCL (LCD Shift Clock) from the above calculated PCLK, you must use the value given for XSCL period for that specific display type (i.e., 2Ts = 2 * 1/PCLK). The XSCL frequency can not be calculated as there is a non-uniform relationship between PCLK and XSCL when in color panel modes.
5. It is recommended to set the PCLK frequency based on panel frame-rate recommendations.

Programming PLL2 (Memory Clock)

Unlike PLL1, PLL2 has only one set of registers. Programming the memory clock is entirely dependent upon the DRAM specifications. To determine a valid memory clock frequency, the five equations below must be satisfied.

6 x MClk period	≥	Random read/write cycle time
2 x MClk period	≥	Fast page mode cycle time
3.4 x MClk period	≥	CAS hold time
2.4 x MClk period	≥	RAS precharge time
Frequency of MClk	≤	40 MHz

If possible, one should match the MClk period to the page mode cycle time equation. This would make the memory controller be at or near optimal operating capacity.

9.2 Programming FIFO Level Request Registers

Low Request level is the depth of the FIFO where four uninterrupted page mode memory accesses can be achieved without overwriting unread data from the previous memory access. This level represents the earliest time the FIFO can request a memory request.

High Request level is the depth of the FIFO where the time taken to read to the end of the FIFO plus the first level of the next memory access is equal to the time it takes to fill the first level of the FIFO. This level represents the very latest time the FIFO can request memory access.

The Following tables tabulate the values for each FIFO level register based on pixel clock frequency, memory clock frequency and display mode.

Where: Tp = period of PClk in nanoseconds
Tm = period of MClk in nanoseconds

$$PixelPerLevel = 32/BitsPerPixel$$

$$LowRequestRegister \leq \left[\frac{C \times Tm}{Tp \times PixPerLevel}\right] + 4$$

$$HighRequestRegister \geq \left[\frac{K \times Tm + 43.3ns}{Tp \times PixPerLevel}\right] - 1$$

Table 9-6: Values for C and K

			VGA slow text	VGA slow gfx	VGA text 8 dot	VGA text 9 dot	VGA Graphics	Enh text	HR256
Pixels per Level			16	16	8	9	8	8	4
512 KB	C		15.45	15.45	15.45	15.45	15.45	15.45	15.45
	K	CRT	29.5	19.5	29.5	29.5	19.5	29.5	19.5
		Dual Mono	33.5	23.5	33.5	33.5	23.5	33.5	23.5
		Dual Color	49.5	39.5	49.5	49.5	39.5	49.5	39.5
1024 KB	C		15.45	7.45	15.45	15.45	7.45	15.45	7.45
	K	CRT	27.5	15.5	27.5	27.5	15.5	27.5	15.5
		Dual Mono	29.5	17.5	29.5	29.5	17.5	29.5	17.5
		Dual Color	37.5	25.5	37.5	37.5	25.5	37.5	25.5

Table 9-7: Hi and Low FIFO Level Request Register Values for Some Common Modes

				CRT				Dual Mono[1]				Dual Color[1]				Simul Dual Mono[2]				Simul Dual Color[2]				Single[3]			
1024K DRAM			**MClk=**	39.4		32.9		39.4		32.9		39.4		32.9		39.4		32.9		39.4		32.9		39.4		32.9	
Mode	**Description**	**Resolution**	**PClk**	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L
0,1	slow text	320x400	28.6	1	4	1	4	0	4	0	4	0	4	1	4	1	4	1	4	1	4	1	4	1	4	1	4
2,3,7	VGA text 8 dot	640x400	25.2	2	5	2	5	1	4	1	4	1	4	2	4	2	5	2	5	3	5	3	5	2	5	3	5
	VGA text 9 dot	720x400	28.6	2	5	2	5																				
108	Enhanced text	640x480 Text	25.2	2	5	2	5	1	4	1	4	1	4	2	4	2	5	2	5	3	5	3	5	2	5	3	5
109-10c		1056x350 Text	39.4	3	5	4	6																				
4,5,d	slow gfx	320x400	25.2	0	4	0	4	0	4	0	4	0	4	0	4	0	4	0	4	1	4	1	4	0	4	0	4
6,e,f,10-13	VGA gfx	640x480	25.2	1	4	1	4	0	4	1	4	1	4	1	4	1	4	1	4	2	4	2	4	1	4	1	4
6a, 102		800x600x16	35.8	1	4	2	5																				
			39.4	2	4	2	5																				
			50.1	2	5	3	5																				
104		1024x768x16	64.4	3	5	4	5																				
100, 101	HR256	640x480x256	25.2	2	5	3	5	1	4	2	4	2	4	2	4	3	5	3	5	4	5	5	5	3	5	3	5
103		800x600x256	35.8	3	5	4	6																				
			39.4	4	5	5	6																				
			50.1	5	6	6	6																				
105		1024x768x256	64.4	7	7	8	7																				
512K DRAM																											
0,1	slow text	320x400	28.6	1	4	1	4	0	4	0	4	1	4	1	4	1	4	1	4	2	4	2	4	1	4	1	4
2,3,7	VGA text 8 dot	640x400	25.2	2	5	2	5	1	4	1	4	2	4	2	4	2	5	3	5	4	5	4	5	2	5	3	5
	VGA text 9 dot	720x400	28.6	2	5	2	5																				
108	Enhanced text	640x480 Text	25.2	2	5	2	5	1	4	1	4	2	4	2	4	2	5	3	5	4	5	4	5	2	5	3	5
109-10c		1056x350 Text	39.4	3	5	4	6																				
4,5,d	slow gfx	320x400	25.2	0	4	0	4	0	4	0	4	0	4	1	4	1	4	1	4	1	4	1	4	0	4	1	4
6,e,f,10-13	VGA gfx	640x480	25.2	1	5	1	5	1	4	1	4	1	4	2	4	2	5	2	5	3	5	3	5	1	5	2	5
6a, 102		800x600x16	35.8	2	5	2	6																				
			39.4	2	5	3	6																				
			50.1	3	6	3	6																				
104		1024x768x16	64.4	4	7	5	7																				
100, 101	HR256	640x480x256	25.2	3	6	3	6	2	5	2	5	3	5	4	5	4	6	4	6	6	6	7	6	3	6	4	7
103		800x600x256	35.8	4	7	5	8																				

= modes which are not possible

[1] PClk = 14 MHz for all modes
[2] PClk = 25 MHz for all modes
[3] PClk = 28 MHz for all modes

Note

Other modes are possible. The formulae in section 9.2 on page 123 must be used to calculate the FIFO numbers for modes not shown above.

9.3 SPC8110 Display Support Specification

This section describes the display support of the SPC8110F0A.

9.3.1 Displays Supported

The following display timings are supported by the SPC8110F0A:

Table 9-8: Supported Display Selection

Timing Mode	CRT Enable Aux[00]b6	LCD Enable Aux[00]b5	TFT Select Aux[02]b4	Single/ Dual Aux[02]b1
CRT	1	0	X	X
Single Panel STN	0	1	0	0
Dual Panel STN	0	1	0	1
TFT Panel	0	1	1	X[1]
Simultaneously Displayed CRT and Single-Panel STN	1	1	0	0
Simultaneously Displayed CRT and Dual-Panel STN	1	1	0	1
Simultaneously Displayed CRT and TFT	1	1	1	X[1]

[1] must be 1 for modulated panel

9.3.2 CRT

All timings and polarities are set by standard VGA-register set.

9.3.3 Single Panel STN

1. The character width is fixed to 8 dots. 9-dot text modes are internally fixed to 8 dots. Horizontal timing is set by the Horizontal Panel Size Register (CRTC B[01h]), and the LcdNonDisplayWidth and LcdHSyncPosition bits (both in CRTC B[00h]). HDE End is still set by the standard CRTC A[01h].
2. Vertical timing is set by the B-set registers (3D4h/3D5h when AUX[00h] b0 = 1). The vertical panel size and duty cycle are programmed using AUX[12h] and [06h] respectively; see the register descriptions for details. 400-line modes can be automatically expanded to 480 lines. For any other vertical resolution less than 480 lines, the image either starts at the top of the display or, if AutoCentering is set, is vertically centered; unused lines are filled with overscan. Vertical resolutions greater than 480 are started at the top and truncated at the bottom.
3. PClk frequency is selected by PLL1 Set 0 registers (AUX[20h] and [21h]), i.e., 3C2h bits3,2 are ignored. The BIOS normally programs these registers to 28.636MHz.

9.3.4 Dual Panel STN

1. Same as Single Panel STN on page 125.
2. Same as Single Panel STN on page 125.
3. PClk frequency is selected by PLL1 Set 0 registers (AUX[20h] and [21h]), i.e., 3C2h bits3, 2 are ignored. The BIOS normally programs these registers to 14.318MHz.

9.3.5 TFT Panel

1. The character width is fixed to 8 dots. 9-dot text modes are internally fixed to 8 dots. Horizontal timing is set by the A-set registers and is the same as in CRT mode. The Horizontal Panel Size Register (CRTC B[01h]) is used to generate the DisplayEnable signal.
2. Vertical timing is set by the B-set registers (3D4h/3D5h when AUX[00h] b0 = 1). The vertical timing should be set to the same as CRT 480-line modes. 400-line modes can be automatically expanded to 480 lines. For any other vertical resolution less than 480 lines, the image either starts at the top of the display or, if AutoCentering is set, is vertically centered; unused lines are filled with overscan. Vertical resolutions greater than 480 are started at the top and truncated at the bottom. The polarities of HRTC and VRTC are fixed to negative pulses to indicate 480-line resolution to the CRT.

   Note

   Some TFT panels can do their own vertical centering of 350- and 400-line modes using the HSYNC and VSYNC to indicate modes. The SPC8110F0A does not use these capabilities as it does not change the sync polarities. Instead it has a single TFT mode that satisfies all TFT panels and allows the use of its own vertical centering and expansion techniques.

3. PClk frequency is selected by PLL1 Set 0 registers (AUX[20h] and [21h]), i.e., 3C2h bits3,2 are ignored. The BIOS normally programs these registers to 25.175MHz - this is the same as the Simultaneous Display mode requirement.
4. If the TFT panel resolution is greater than 640x480, care must be taken to ensure that the CRTC A[00] Horizontal Total is ≥ CRTC B[01] Horizontal Panel size.

9.3.6 Simultaneously Displayed CRT and Single Panel STN

1. The character width is fixed to 8 dots. 9-dot text modes are internally fixed to 8 dots. Horizontal timing is set by the standard A-set CRTC registers. The horizontal resolution should always be 640 dots. 720-dot text modes are internally fixed to 640. For any horizontal resolution less than 640 dots, the image starts at the left of the displays and the remaining space at the right is blanked.
2. The vertical resolution is fixed at 480 lines. 400-line modes are automatically expanded to 480 lines. For any vertical resolution less than 480 lines, the image either starts at the top of the display or, if AutoCentering is set, is vertically centered; unused lines are blanked. Vertical resolutions greater than 480 are truncated. Vertical timing is set by the non-standard B-set CRTC registers (3D4h/3D5h when AUX[00h] b0 = 1). The polarities of HRTC and VRTC are fixed to negative pulses to indicate 480-line resolution to the CRT.
3. The dot clock is fixed at 25.175MHz for a 60Hz screen refresh rate on the CRT. Single panel LCD's are fixed to the timings of the PS/2 display in 640x480 dot mode at 25.175MHz (i.e., mode 12h). The LCD Bias circuitry must detect the presence of simultaneous display mode and adjust the contrast accordingly; this is the responsibility of the System Designer.

9.3.7 Simultaneously Displayed CRT and Dual Panel STN

1. Same as Simultaneously Displayed CRT and Single Panel STN on page 126.
2. Same as Simultaneously Displayed CRT and Single Panel STN on page 126.
3. Same as Simultaneously Displayed CRT and Single Panel STN on page 126.
4. Dual panel LCD's must run at twice the rate of the CRT in simultaneous display mode. This averages out to 120Hz (they run at ~75Hz in normal operation). For dual-panel LCD's an extra VRTC pulse is generated at the mid-point of the CRT image. This means that the vertical scan length of the dual panel is not constant but alternates between 240 and 285 lines (45 lines is the length of the CRT non-display period).

9.3.8 Simultaneously Displayed CRT and TFT

1. Same as TFT Panel on page 126.
2. Same as TFT Panel on page 126, except that unused lines are blanked instead of filled with overscan color.
3. Same as TFT Panel on page 126.

9.3.9 Vertical Timing for Simultaneous Display

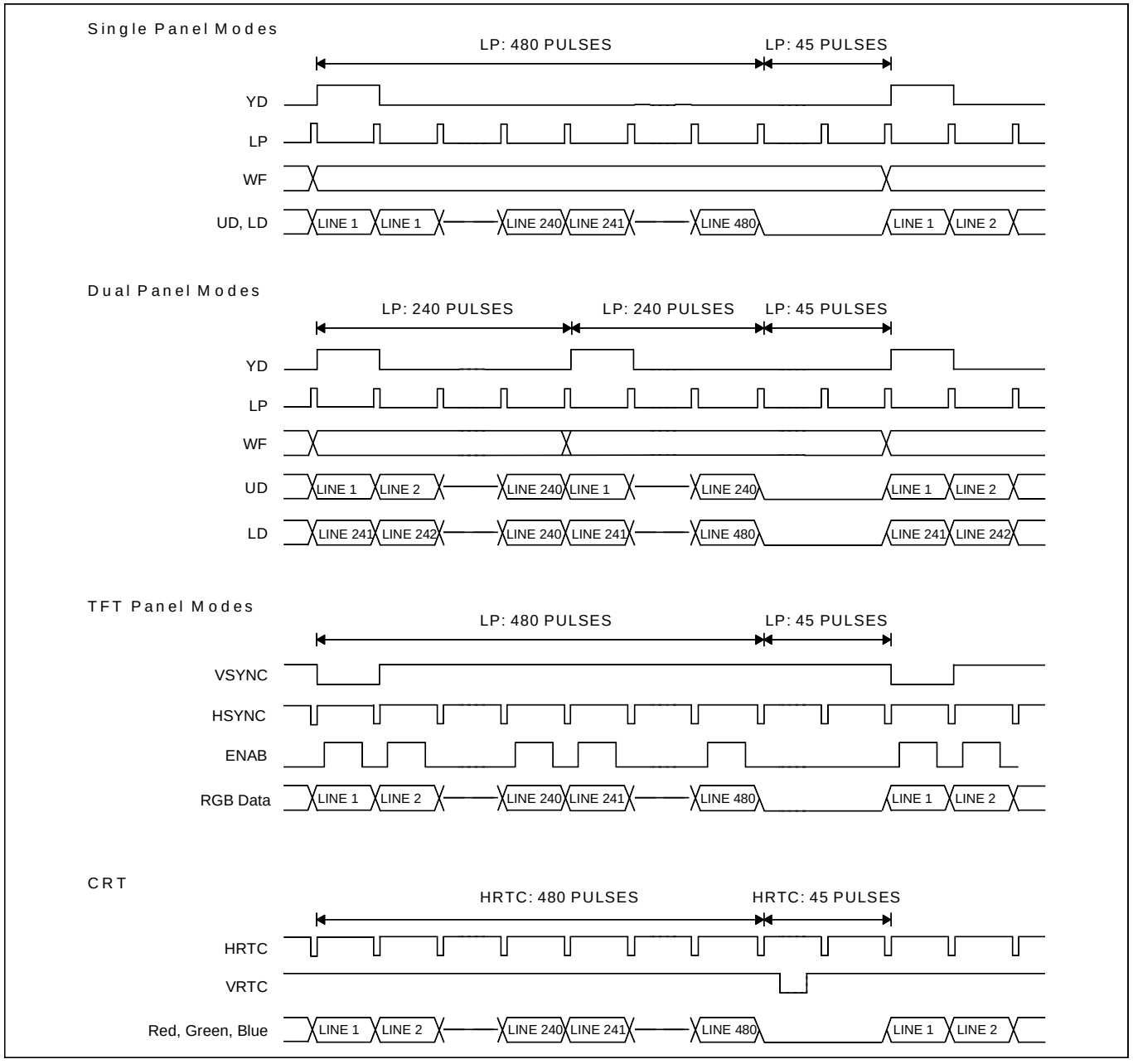


Figure 43: Vertical Timing for Simultaneous Display

9.4 DRAM Refresh

9.4.1 Functional Overview

The DRAM refresh circuitry is based on the 32 kHz clock coming in through the RTCLK pin. The refresh circuit is completely independent of the pixel clock and memory clock frequency. In the Memory Config Register, the Refresh Rate bits determine how many times RTCLK is divided down so that the frequency of the divided down clock equals one half the frequency of the DRAM refresh rate. During normal operation, the divided down RTCLK is synchronized to MCLK. Both edges of the synchronized clock are then used to signal the start of a CAS before RAS (CBR) refresh cycle. In the case where MClk is shut down during a power down mode, the normal refresh circuitry is replaced with one of two separate circuits: CBR self refresh or delay pulse CBR refresh.

CBR Self Refresh

The DRAM can only enter self refresh mode when a PSM with the MClk off function is activated and the Self Refresh Support bit is selected in the Memory Config Register. When entering self refresh, the memory interface block first completes any cycles that are pending. It then sets the DRAM into a CAS before RAS self refresh mode. When coming out of self refresh, the refresh duties are simply transferred back to the normal CBR cycles.

Power Down Mode CBR Refresh

This power down refresh cycle must be used by DRAMs that do not support CBR self refresh. The power down mode CBR refresh is only used when a PSM with the MClk off function is activated and the Self Refresh Support bit in the Memory Config Register is not selected. When entering the PSM, the memory interface block first completes any cycles that are pending. The control of the RAS and CAS output buffers are then released to the power down mode CBR refresh circuit. The circuit takes each edge of the divided down RTCLK and generates a pulse of 175 ns wide to be used as the active RAS# and CAS# pulse of a CBR refresh cycle. When coming out of the PSM, the power down mode CBR circuit will complete its last CBR cycle before passing the control of the RAS# and CAS# lines back to the memory interface block.

9.5 Sprite/Hardware Cursor

This section provides a functional description of the SPC8110F0A Sprite/HW Cursor. The SPC8110F0A cursor hardware will display a single four color (2-bit) overlay image at any location on the screen, allowing clipping of the overlay image. The image vertical and horizontal dimensions can be programmed independently, allowing sizes from 64 x 64 dots to 1 x 1 dot. Each dot of the image can be one pixel, two horizontal pixels, two vertical pixels or a group of 4 pixels on the display, allowing one cursor image to operate in all VGA display modes. The size of the image dots are controlled by bits 2 and 3 of CRTC[38h]. One of three palette schemes for the two-bit pixels can be selected by the Sprite/HW Cursor bit-map select bits, bits 1 and 0 of CRTC[38h]. The palette schemes, shown below, select either one of four palette entries, one of three palette entries and the value of the screen image behind the Sprite/HW Cursor or one of two palette entries and the value or the inverted value of the screen image behind the Sprite/HW Cursor.

Table 9-9: Sprite/HW Cursor Palette Schemes

Pixel		Sprite/HW Cursor Pixel Displayed		
Bit 1	**Bit 0**	**CRTC[38h] b1:0 = 00b**	**CRTC[38h] b1:0 = 01b**	**CRTC[38h] b1:0 = 10b**
0	0	Sprite/HW Cursor Palette Entry 0	Screen Image	Sprite/HW Cursor Palette Entry 0
0	1	Sprite/HW Cursor Palette Entry 1	Sprite/HW Cursor Palette Entry 1	Sprite/HW Cursor Palette Entry 1
1	0	Sprite/HW Cursor Palette Entry 2	Sprite/HW Cursor Palette Entry 2	Screen Image
1	1	Sprite/HW Cursor Palette Entry 3	Sprite/HW Cursor Palette Entry 3	Inverted Screen Image

The Sprite/HW Cursor palette entries 0 to 3 use the overlay portion of the Palette and are accessed via the Palette Register I/O addresses (3C7h-3C9h) when AUX[00h] b7=1. The pattern of the Sprite/HW Cursor is stored in an off screen area of the display memory. The starting address of the Sprite/HW Cursor patterns are stored in the Sprite/HW Cursor Select Register, CRTC[36h] and CRTC[37h]. The uppermost address bits, CRTC[36h] b1:0, should not be set to 00 in order to avoid conflict with existing programs that use the VGA display space. The option to use this space is included in order to allow complete flexibility. With the upper 768KB free from VGA compatibility issues, this allows up to 768 Sprite/HW Cursors to be stored in video memory (for 1024KB video memory).

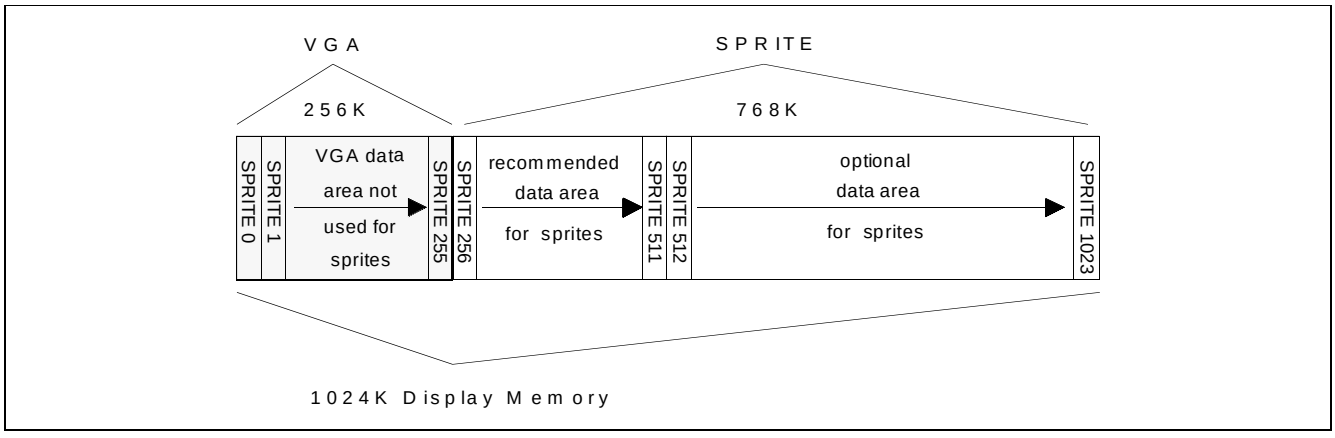


Figure 44: Sprite/HW Cursor Display Memory

9.5.1 Sprite Positioning

Positioning of the Sprite/HW Cursor is programmed with Sprite/HW Cursor X,Y Position Register in conjunction with Sprite/HW Cursor X, Y Start Column, Row which permit partial Sprite/HW Cursor images on all sides of the display. The X coordinate has a range of values from 0 to 2047 and the Y coordinate has a range of values from 0 to 1023.

The Sprite/HW Cursor Enable bit, the Sprite/HW Cursor X & Y Position Register, the Sprite/HW Cursor Address Register, and the X & Y Display Start Column & Row Register values are internally latched following the display of the current Sprite/HW Cursor to prevent any visual anomalies. Anomalies may occur in the case where the registers are being modified while the controller is reading in the value. The Sprite/HW Cursor Palette Registers are not latched internally. Any modification to these registers should be done either during the vertical non-display period or when the Sprite/HW Cursor circuitry is disabled. To prevent visual anomalies the Sprite/HW Cursor X Low and High Position Register values are not internally latched from the time the Sprite/HW Cursor X Low Register is written until after the Sprite/HW Cursor X High Register is written. Similarly, the Sprite/HW Cursor Y Low and High Position Register values are not internally latched from the time the Sprite/HW Cursor Y Low Register is written until after the Sprite/HW Cursor Y High Register is written.

In order for changes to either the Sprite/HW Cursor X or Sprite/HW Cursor Y Position Registers to affect the display the least significant bits must be written first followed by a write to the most significant bits.

9.5.2 Sprite Memory Organization

The following figure shows memory as accessed in Graphics Controller Bypass mode. Each sprite pixel is composed of 2 bits, PB and PA. The most significant bit being PB.

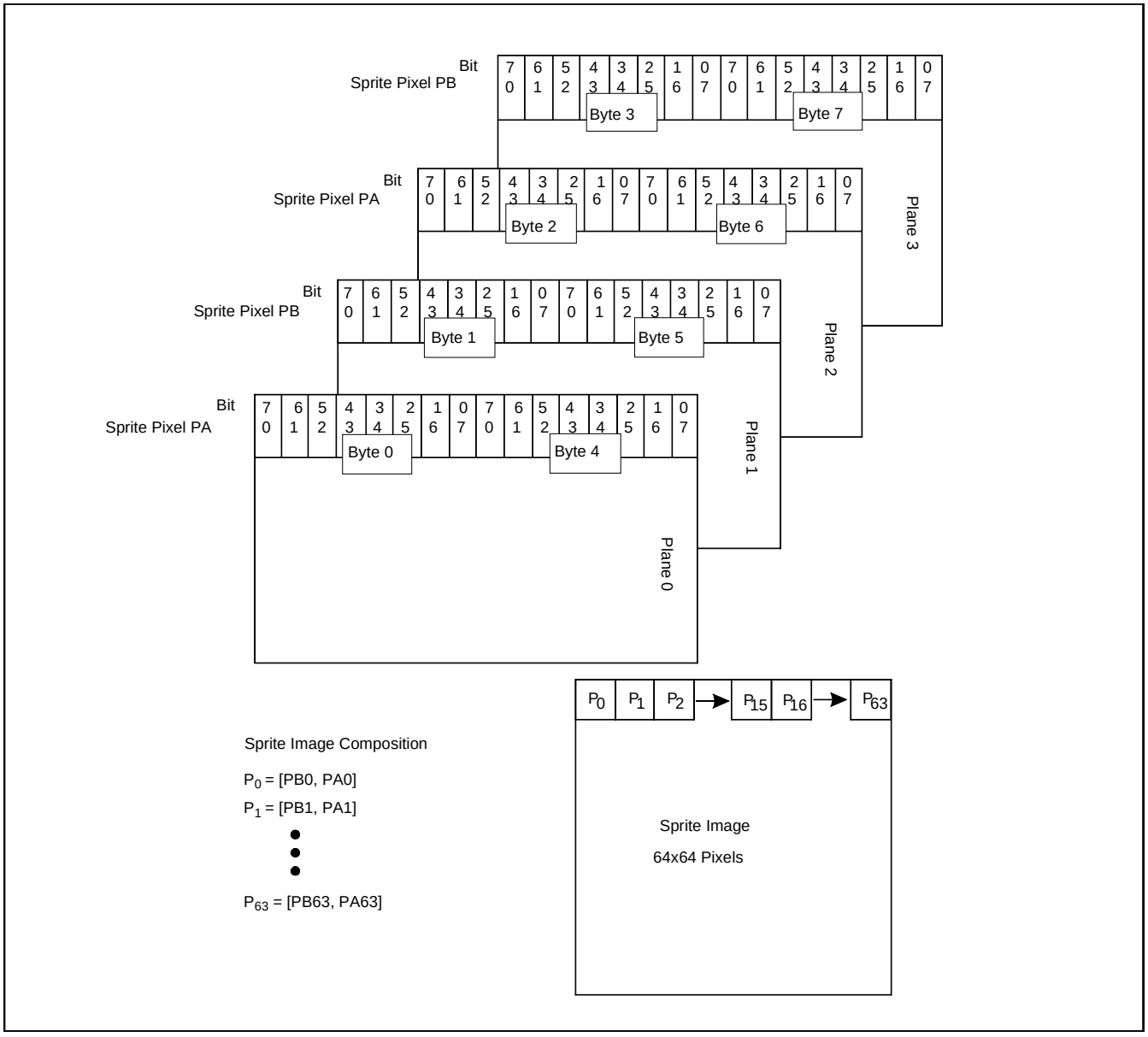


Figure 45: Sprite Memory Organization

9.6 BitBlt Specification

9.6.1 Overview

The BitBlt Engine is designed to increase performance of Bit Block Transfers. It does this by off-loading work from the CPU, reducing traffic on the CPU bus and improving the efficiency of the Display Memory Interface.

It is designed to work in both planar and packed pixel graphics modes. Data alignment between the source and destination is handled automatically.

The BitBLTter has several features for improving text-out operations including Color Expansion and Automatic Update (shift right to next letter position). Also supported are transparency, solid fills, patterns and Destination Invert.

9.6.2 Features

Color Expansion

Monochrome data from the host bus is translated into foreground/background Color data before being stored in the video buffer. Each bit of the host data represents a pixel in the image; the most significant bit of the host data represents the leftmost pixel. In 4 bit/pixel planar modes, the 8 lsb of the 32-bit Data Register are expanded to eight 4-bit pixels; the upper 24 bits of the register are discarded. In 8-bit/pixel linear modes, the 4 lsb of the 32-bit Data Register are expanded to four 8-bit pixels; the upper 28 bits of the register are discarded. A logical one selects the foreground color for the pixel, while a logical zero selects the background color. The foreground and background colors are selected by programming the Foreground and Background Color Registers.

Transparency

Same as Color Expansion except that a logical zero in the host data selects the destination data. This can be used to overlay an image in the foreground color on top of the existing image.

Translation

16-color video information is automatically translated from packed-pixel format from the host to planar format for the video buffer and vice-versa. The amount of data read/written by the host for one line of the BitBLT must be a multiple of double-words even if the last double-word contains undisplayed (dummy) data; note that the amount of data transferred is the same as that required by a normal planar mode transaction. This feature is designed to off-load work from the CPU.

Solid Fills

The specified BitBLT area is filled with a solid color from the Foreground Color Register.

Patterns

An 8 x 8 pixel color pattern is replicated over the whole BitBLT area. The source of the pattern must be a 64 pixel color bitmap stored in video memory. The bitmap is stored linearly as per the selected mode (planar or packed pixel), i.e. a contiguous 32 byte block in planar mode and a contiguous 64 byte block in packed pel mode. These blocks must start on logical block boundaries; the high order bits of the Source Address Register are used to determine the location of the block to be used and the low order bits are used to select which pixel in the 8 x 8 pattern is placed at the top left corner of the BitBLT. Note that only BitBLTs in the positive direction are supported for patterns.

Destination Invert

The destination data is logically inverted, i.e. each bit of video memory data for the BitBLT area is indiscriminately inverted. The effect this has on the viewed image is dependant on how the palettes are programmed.

Automatic Update

The DestAddress register is automatically updated at the end of a BitBLT to point to the immediate right of this BitBLT. This is useful when writing text. Note that this only works in the x-positive direction.

Automatic Start

Used in association with Automatic Update when writing text. The next BitBLT is automatically started as soon as the Width register is written.

9.7 Power Save Modes (PSM)

9.7.1 Functional Overview

The SPC8110F0A has four independent power save modes (PSM). Each PSM has the following associated with it: an enable, an initiator, and a function register. The enable is a bit that permits activation of a PSM. The initiator is the means by which the PSM is activated. The function register selects what actions are carried out by the PSM.

A hierarchy arbitrates between co-active PSM; a PSM with a higher priority will completely override one with a lower number.

The following table summarizes the power save mode architecture:

Table 9-10: Power Save Mode Architecture

Name	Priority	Enable	Initiator	Function Reg
PSM 1	1 (lowest)	PSM 1 Enable	PS1PIN or Timer [1]	PS Function1
PSM 2	2	PSM 2 Enable	PS2SW	PS Function2
PSM 3	3	PSM 3 Enable	PS3SW	PS Function3
PSM 4	4 (highest)	PSM 4 Enable	PS4PIN	PS Function4

[1] The timer initiator shares PSM 1 with PS1PIN initiator. A select bit in the PS Pin and Timer Register is provided to select either the timer or the PS1PIN pin as the initiator for PSM 1.

Enable

The enable bit, provided to all PSMs, is a master switch for its associated PSM. If the enable bit is not set, the related PSM can not be activated. For simultaneous display modes, all PSMs are disabled.

Initiators

When an initiator becomes active and has the highest precedence of all active initiators, its associated PSM is activated. The PSM remains active until one of three events occur: the initiator becomes inactive, the PSM's enable bit is turned off, or a PSM of higher precedence becomes active. In the case where an initiator of higher precedence goes from active to inactive, the remaining active initiator with the highest level of precedence will have its PSM activated. There are three types of initiators: hardware, software, or timer.

The hardware initiators, PS1PIN and PS4PIN, are inputs from the external pins of the same name. When the external pin is pulled to an active state, the initiator becomes active. Each pin has an active status bit for software monitoring. The active polarity of the pins are defined by two polarity bits found in the PS Pin and Timer Register (AUX[10h]). PS1PIN is also used as an activity sensor when PSM 1 is configured to the timer option.

For the software initiators, PS2SW and PS3SW, the state of the initiators follows the state of the enable bits. Setting the enable bit directly activates the PSM, assuming a PSM of higher precedence is not already active.

The timer monitors the CPU bus and the PS1PIN pin. When the period programmed into the PS Pin and Timer Register has passed without any activity, the timer initiator becomes active and the power save mode is activated. If activity is detected at anytime, the power save mode is deactivated, the timer and initiator resets and the process repeats. Activity is defined as an edge on the PS1PIN pin, or an access to either the VGA memory or VGA I/O ports.

Function Register

The function register determines which actions are carried out by the PSM. All four PSMs have their own 8-bit function register, where each register has identical functions. In addition, PSM 1 (with PS1PIN selected as the initiator) and PSM 4 have the additional function of shutting down the bus interface completely. Each function is discussed in the register section.

9.7.2 Output Pin States Affected By Power Save Functions

Table 9-11: Output Pin States Affected By Power Save Functions With CRT Enabled / LCD Disabled

CRT Enabled, LCD Disabled		Pins									
		CRT			LCD				Memory		CPU
		RED, GREEN BLUE	HRTC	VRTC	TFTB[1:0] UD[7:0], LD[7:0]	YD, LP, WF XSCLL, XSCLU	LCDBACK	LCDLOGIC LCDBIAS	MDA[15:0] MDB[15:0]	RAS#,WE[x:0]# CAS[x:0]#	local bus outputs
Power Save Fct Note 1	Normal	active	active	active	low	low	off	off	active	active	active
	Display Off	0.0V							CPU	CPU	
	DPMS 01		low	active							
	DPMS 10		active	low							
	DPMS 11		low	low							
	LCDBACK										
	PClkOff	0.0V	low	low					CPU	CPU	
	MClkOff								ts	PDR	I/O
	HostI/F										inactive

Table 9-12: Output Pin States Affected By Power Save Functions With LCD Enabled / CRT Disabled

LCD Enabled, CRT Disabled		Pins									
		CRT			LCD				Memory		CPU
		RED, GREEN BLUE	HRTC	VRTC	TFTB[1:0] UD[7:0], LD[7:0]	YD, LP, WF XSCLL, XSCLU	LCDBACK	LCDLOGIC LCDBIAS	MDA[15:0] MDB[15:0]	RAS#,WE[x:0]# CAS[x:0]#	local bus outputs
Power Save Fct Note 1	Normal	0.0V	low	low	active	active	on	on	active	active	active
	Display Off				low	low		off	CPU	CPU	
	DPMS 01										
	DPMS 10										
	DPMS 11										
	LCDBACK						off				
	PClkOff				low	low		off	CPU	CPU	
	MClkOff								ts	PDR	I/O
	HostI/F										inactive

= function does not affect state of output pins

Note

Reduce Gray-scale and PClk Slow functions do not affect the state of the output pins.

Table Terminology

active:	normal operation.
inactive:	Output pin state can either be low or tri-state, depending on the setting of the register bit LCD Data PSM Tri-State Enable.
CPU:	Memory access from the CPU or BitBLT. No memory access for display.
I/O:	Bus cycle decoded for I/O access only. All attempts to access memory are ignored.
low:	Low level CMOS output.
on:	On state, definable in a configuration register
off:	Off state, definable in a configuration register
PDR:	Power down refresh. Can either be a CAS before RAS refresh cycle generated from the 32 kHz clock or a self refresh cycle
ts:	Tri-state.

10 MECHANICAL DATA

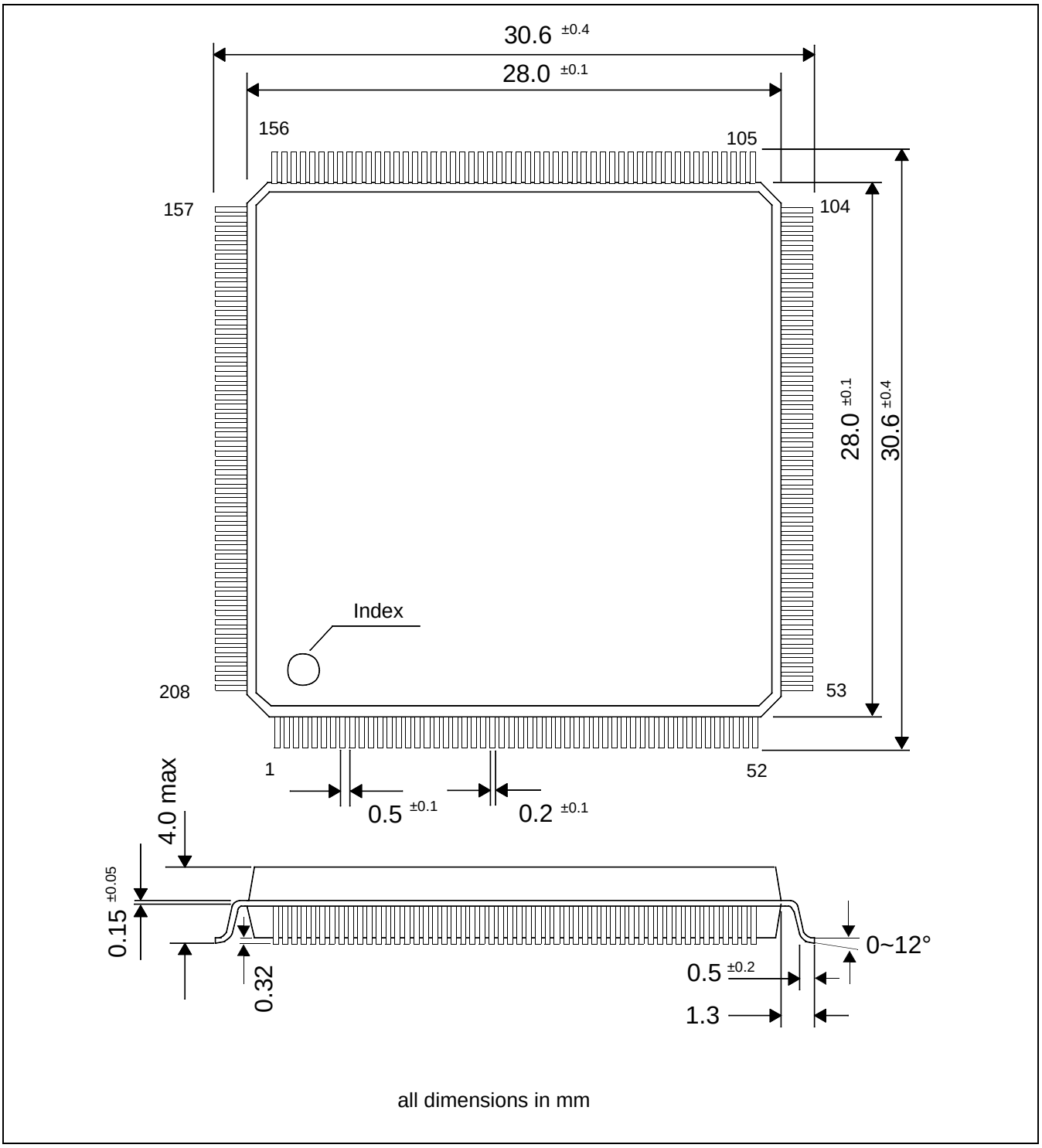


Figure 46: Mechanical Drawing QFP8-208

Appendix A TRADEMARKS

Intel486 is a trademark of the Intel Corporation.

VL-Bus and VESA are trademarks of the Video Electronics Standards Association.

IBM is a trademark of the IBM Corporation.

Tri-state is a trademark of National Semiconductor.

S-MOS
SYSTEMS
A Seiko Epson Affiliate

SPC8110F0A Color VGA LCD/CRT Controller

BIOS Functional Specification

Drawing Office No. X07-SP-003-05

Copyright © 1993, 1997 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

TABLE OF CONTENTS

11 INTRODUCTION 7
1.1 Scope and Objectives 7
2 FEATURES 8
2.1 Technology 8
2.2 System and System Compatibility 8
2.3 Display support 8
2.4 Video Modes 9
2.4.1 LCD Display Modes 9
2.4.2 CRT Display Modes 10
3 OVERVIEW DESCRIPTION 11
3.1 Main BIOS 11
3.2 VESA Extensions 11
3.3 Sollex Extensions 11
4 MAIN BIOS FUNCTION SUMMARY 12
4.1 Supported BIOS Functions 12
5 VESA FUNCTION SUMMARY 25
5.1 Status Information 25
5.2 VESA Functions 25
6 SOLLEX FUNCTION SUMMARY 34
6.1 Sollex Status Information 34
6.2 Sollex Reserved Bits 34
6.3 Sollex Functions 34
7 PHYSICAL LAYOUT OF THE BIOS 41
7.1 Structure 41
7.1.1 VL-Bus 41
7.1.2 PCI Bus 41
7.2 IBM Notice, Copyright Notice 41
8 SMALL FORM FACTOR PANEL SUPPORT ISSUES 42
8.1 Special Mode Support 42
8.1.1 Viewing Area 42
8.1.2 Software Compatibility 43
8.1.3 Programming and Driver Considerations 43
Appendix A BIOS CONFIGURATION INPUTS 44
A.1 Panel Type 44
A.2 Refresh Select 45
A.3 DRAM Speed Select 45
Appendix B BIOS USE OF INTERNAL SCRATCH BITS 46
Appendix C INITIALIZATION 47

Appendix D SPECIAL CASES 75
Appendix E BIOS DEFAULT CONFIGURATION OPTIONS 76
Appendix F VESA INFORMATION 77
F.1 VESA Structures 77
F.2 VESA Mode Support 78
F.2.1 VESA Graphics Modes 78
F.2.2 VESA Text Modes 78
F.2.3 VESA High Color Graphics Modes 79
Appendix G SIMULTANEOUS DISPLAY ISSUES 80
Appendix H BIOS HEADER INFORMATION 81
Appendix I PCI HEADER INFORMATION 82

LIST OF TABLES

Table 4-1 Information Block Offset 2D 22
Table 4-2 Information Block Offset 32 23
Table 6-1 Sub-Function 00h: Set Adapter 35
Table 6-2 Sub-Function 01h: Get Adapter 35
Table 6-3 Sub-Function 00h: Set Display Output 36
Table 6-4 Sub-Function 01h: Get Display Output. 36
Table 6-5 Display Attached 37
Table 6-6 Chip Specific Function 40

THIS PAGE LEFT BLANK

1 INTRODUCTION

1.1 Scope and Objectives

This is the Functional Specification for the SPC8110F0A BIOS. This document is a preliminary specification which summarizes the functions, structures and characteristics of the SPC8110F0A BIOS. It is intended for use by persons familiar with VGA BIOS functions and describes differences between the SPC8110F0A and a standard VGA BIOS.

In addition this manual documents some of the behavioral characteristics and structures within the SPC8110F0A BIOS.

This document is written in several sections including:

- Features
- Video Modes
- Main BIOS Function Summary
- VESA Function Summary
- SOLLEX Function Summary
- Physical Layout of the BIOS
- A Series of SPC8110F0A Specific Summaries

2 FEATURES

2.1 Technology

- 32 KB EPROM maximum size
- Microsoft MASM 6.11 compatible source
- 80386 instruction minimum

2.2 System and System Compatibility

- VL-Bus or PCI BIOS architecture
- support for VESA VBE Core Functions Version 1.2 (BIOS versions lower than 1.2)
- support for VESA VBE Core Functions Version 2.0 (BIOS versions 1.2 or higher)
- programmable power-save modes
- 3C3h video enable register supported
- support for all standard IBM defined VGA modes, including modes 07h and 0Fh
- selectable gray-scaling
- normal/reverse and autoswitch text/graphics of display polarity support
- autocenter support
- selectable cursor blink rate and cursor shape
- scan line expansion in text and graphics modes
- simultaneous display (dual display - CRT and LCD) support in all 480 (or less) scan line modes
- supports VESA Display Power Management Signalling (DPMS) with BIOS version 1.1 or higher
- supports VESA Display Data Channel (DDC level 2B) with BIOS version 1.2 or higher

2.3 Display support

- supports a variety of single/dual LCD panels of various resolutions from 320 x 200 to 640 x 480, based on MDA line inputs
- supports analog CRT/Hi-Resolution VESA DPMS/DDC compatible monitors with BIOS versions higher than 1.2

2.4 Video Modes

2.4.1 LCD Display Modes

Mode No.	Mode Type	Font	Resolution	Characters	Colors[a]	Memory Segment	Simultaneous Display Support
0	T	8 x 8	320 x 200	40 x 25	16	B800	Yes
0+	T	8 x 14	320 x 350	40 x 25	16	B800	Yes
0++	T	8 x 16	320 x 400	40 x 25	16	B800	Yes
1	T	8 x 8	320 x 200	40 x 25	16	B800	Yes
1+	T	8 x 14	320 x 350	40 x 25	16	B800	Yes
1++	T	8 x 16	320 x 400	40 x 25	16	B800	Yes
2	T	8 x 8	640 x 200	80 x 25	16	B800	Yes
2+	T	8 x 14	640 x 350	80 x 25	16	B800	Yes
2++	T	8 x 16	640 x 400	80 x 25	16	B800	Yes
3	T	8 x 8	640 x 200	80 x 25	16	B800	Yes
3+	T	8 x 14	640 x 350	80 x 25	16	B800	Yes
3++	T	8 x 16	640 x 400	80 x 25	16	B800	Yes
4	G	N/A	320 x 200	N/A	4	B800	Yes
5	G	N/A	320 x 200	N/A	4	B800	Yes
6	G	N/A	640 x 200	N/A	2	B800	Yes
7	T	8 x 14	640 x 350	80 x 25	2	B000	Yes
7+	T	8 x 16	640 x 400	80 x 25	2	B000	Yes
8-C Invalid modes							
0D	G	N/A	320 x 200	N/A	16	A000	Yes
0E	G	N/A	640 x 200	N/A	16	A000	Yes
0F	G	N/A	640 x 350	N/A	2	A000	Yes
10	G	N/A	640 x 350	N/A	16	A000	Yes
11	G	N/A	640 x 480	N/A	2	A000	Yes
12	G	N/A	640 x 480	N/A	16	A000	Yes
13	G	N/A	320 x 200	N/A	256[b]	A000	Yes
41	G	N/A	640 x 200	N/A	16	A000	Yes
42	G	N/A	640 x 240	N/A	16	A000	Yes
43	G	N/A	640 x 256	N/A	16	A000	Yes
44	G	N/A	640 x 320	N/A	16	A000	Yes
45	G	N/A	640 x 400	N/A	16	A000	Yes
53	G	N/A	320 x 240	N/A	256[b]	A000	Yes
6A	G	N/A	800 x 600	N/A	16	A000	No
100	G	N/A	640 x 400	N/A	256[b]	A000	Yes
101	G	N/A	640 x 480	N/A	256[b]	A000	Yes
102	G	N/A	800 x 600	N/A	16	A000	No
103	G	N/A	800 x 600	N/A	256[b]	A000	No
104	G	N/A	1024 x 768	N/A	16	A000	No
105[c]	G	N/A	1024 x 768	N/A	256[b]	A000	No
108	T	8 x 8	640 x 480	80 x 60	16	B800	Yes

[a] Monochrome panels display gray shades.
[b] Colors mapped to 64 shades of gray by hardware on monochrome panels.
[c] Requires 1024 K bytes of video memory and TFT panel.

2.4.2 CRT Display Modes

Mode No.	Mode Type	Font	Resolution	Characters	Colors	Memory Segment	Simultaneous Display Support
0	T	8 x 8	320 x 200	40 x 25	16	B800	Yes
0+	T	8 x 14	320 x 350	40 x 25	16	B800	Yes
0++	T	9 x 16	360 x 400	40 x 25	16	B800	Yes
1	T	8 x 8	320 x 200	40 x 25	16	B800	Yes
1+	T	8 x 14	320 x 350	40 x 25	16	B800	Yes
1++	T	9 x 16	360 x 400	40 x 25	16	B800	Yes
2	T	8 x 8	640 x 200	80 x 25	16	B800	Yes
2+	T	8 x 14	640 x 350	80 x 25	16	B800	Yes
2++	T	9 x 16	720 x 400	80 x 25	16	B800	Yes
3	T	8 x 8	640 x 200	80 x 25	16	B800	Yes
3+	T	8 x 14	640 x 350	80 x 25	16	B800	Yes
3++	T	9 x 16	720 x 400	80 x 25	16	B800	Yes
4	G	N/A	320 x 200	N/A	4	B800	Yes
5	G	N/A	320 x 200	N/A	4	B800	Yes
6	G	N/A	640 x 200	N/A	2	B800	Yes
7	T	9 x 14	720 x 350	80 x 25	2	B000	Yes
7+	T	9 x 16	720 x 400	80 x 25	2	B000	Yes
8 ~ C Invalid modes							
0D	G	N/A	320 x 200	N/A	16	A000	Yes
0E	G	N/A	640 x 200	N/A	16	A000	Yes
0F	G	N/A	640 x 350	N/A	2	A000	Yes
10	G	N/A	640 x 350	N/A	16	A000	Yes
11	G	N/A	640 x 480	N/A	2	A000	Yes
12	G	N/A	640 x 480	N/A	16	A000	Yes
13	G	N/A	320 x 200	N/A	256	A000	Yes
41	G	N/A	640 x 200	N/A	16	A000	Yes
42	G	N/A	640 x 240	N/A	16	A000	Yes
43	G	N/A	640 x 256	N/A	16	A000	Yes
44	G	N/A	640 x 320	N/A	16	A000	Yes
45	G	N/A	640 x 400	N/A	16	A000	Yes
53	G	N/A	320 x 240	N/A	256	A000	Yes
6A	G	N/A	800 x 600	N/A	16	A000	No
100	G	N/A	640 x 400	N/A	256	A000	Yes
101	G	N/A	640 x 480	N/A	256	A000	Yes
102	G	N/A	800 x 600	N/A	16	A000	No
103	G	N/A	800 x 600	N/A	256	A000	No
104	G	N/A	1024 x 768	N/A	16	A000	No
105[a]	G	N/A	1024 x 768	N/A	256	A000	No
108	T	8 x 8	640 x 480	80 x 60	16	B800	Yes
109	T	8 x 16	1056 x 400	132 x 25	16	B800	No
10A	T	8 x 8	1056 x 350	132 x 43	16	B800	No
10B	T	8 x 8	1056 x 400	132 x 50	16	B800	No
10C	T	8 x 8	1056 x 480	132 x 60	16	B800	No

[a] Requires 1024 K bytes of video memory

3 OVERVIEW DESCRIPTION

The SPC8110F0A is a single chip mixed voltage LCD/CRT video controller based on VGA architecture and optimized for driving an LCD panel display. VGA standard mode functionality is supported using standard IBM VGA parameters. Hardware Bit Block Transfer, hardware cursor/sprite enhancements provide software with the ability to speed up Windows or Windows-like applications.

The target market for this device is notebook computers, or other specialized products. The ability to run all VGA software on an LCD panel display is the major design consideration, therefore the BIOS must perform the same functions that the IBM VGA BIOS performs. In addition, the BIOS is extended to take advantage of the hardware enhancements in the SPC8110F0A.

The BIOS is divided into 3 major sections: the main BIOS, the VESA Extensions and the Sollex Extensions. It requires 32 KB of EPROM space decoded by the system board.

3.1 Main BIOS

The main BIOS contains the core VGA compatible information. It requires 24 KB of the total 32 KB of EPROM space. This main BIOS is responsible for the initialization of the chip and for performing the IBM compatible function calls. It contains the Video Parameter Tables and the Character Tables.

3.2 VESA Extensions

The VESA Extensions are found in the last 8 KB of the 32 KB. This contains the VESA compatible functions as defined by the Video Electronics Standards Association. BIOS versions 1.0 to 1.2 will contain VESA BIOS Extension Standard compatibility for VBE 1.2. BIOS versions greater than 1.2 will contain VBE 1.2, VBE/PM 1.0, and VBE/DDC 1.0 compatibility. BIOS versions greater than 1.2 will contain VBE Core Functions 2.0, VBE/PM 1.0, and VBE/DDC 1.0 compatibility. More information on VESA and VESA Functions can be obtained by contacting the Video Electronics Standards Association headquartered in San Jose, CA.

3.3 Sollex Extensions

The Sollex Extensions are defined by Seiko Epson Corporation to augment the functionality of the BIOS to include panel and power down functions. This also resides in the last 8 KB. More information on Sollex can be found in the *SOLLEX Specification* Drawing Office No. S03-SP-001-xx.

4 MAIN BIOS FUNCTION SUMMARY

4.1 Supported BIOS Functions

These functions are the IBM defined functions that are supported on all VGA compatible products. These functions will be supported in the SPC8110F0A BIOS with the noted exceptions. To call these functions:

MOV AH, function number

MOV other register, other parameters

INT 10H

Function 00h - Set Video Mode

```
Input:    AH=00h    Set Video Mode
          AL        Video mode (bit 7 set prevents VRAM clear)
Return:   n/a
```

Function 01h - Set Cursor Type

```
Input:    AH=01h    Set Cursor Type
          CH        Cursor start scan
          CL        Cursor end scan
Return:   n/a
```

Function 02h - Set Cursor Position

```
Input:    AH=02h    Set Cursor Position
          BH        Page number
          DL        Column (0-x)
          DH        Row (0-x)
Return:   n/a
Destroyed:AX,SI
```

Function 03h - Read Cursor Position

```
Input:    AH=03h    Read Cursor Position
          BH        Page number
Return:   CX        Current cursor mode
          DX        Current cursor position
```

Function 04h - Read Lightpen Position (Unsupported On VGA)

```
Input:    AH=04h    Read Lightpen Position
Return:   AH        Lightpen status (0=none, 1=active)
                    If AH=1 then:
                              BX - Pixel column
                              CX - Scan line
                              DX - Character row/column
```

Function 05h - Select Active Display Page

```
Input:   AH=05h   Select Active Display Page
         AL       New page number
Return:  n/a
```

Function 06h - Scroll Active Page Up

```
Input:   AH=06h   Scroll Active Page Up
         AL       Lines to scroll (0=blank window)
         BH       New line(s) attribute
         CX       Top-left corner of scroll window
         DX       Bottom-right corner of scroll window
Return:  n/a
Destroyed:AX, SI, DI, (and DS if text modes)
```

Function 07h - Scroll Active Page Down

```
Input:   AH=07h   Scroll Active Page Down
         AL       Lines to scroll (0=blank window)
         BH       New line(s) attribute
         CX       Top-left corner of scroll window
         DX       Bottom-right corner of scroll window
Return:  n/a
```

Function 08h - Read Character/attribute at Cursor Position

```
Input:   AH=08h   Read Character/attribute at Cursor Position
         BH       Page number
Return:  AL       Character read
         AH       Attribute read
Destroyed:AX, SI, DI, (and DS if text modes)
```

Function 09h - Write Character/attribute at Cursor Position

```
Input:   AH=09h   Write Character/attribute at Cursor Position
         AL       Character to write
         BL       Character attribute/color (b7 set for XOR)
         BH       Page number (Background color in Mode 13)
         CX       Character count
Return:  n/a
Destroyed:AX, SI, DI, (and DS if text modes)
```

Function 0Ah - Write Character Only at Cursor Position

```
Input:   AH=0Ah   Write Character only at Cursor Position
         AL       Character to write
         BH       Page number
         CX       Character count
Return:  n/a
Destroyed: AX, SI, DI, (and DS if text modes)
```

Function 0Bh - Set Color Palette

```
Input:   AH=0Bh   Set Color Palette
         BH=0     (selects background color)
                  BL=0-Fh   Background color
         BH=1     (selects palette)
                  BL=0     (Green, Red, Brown)
                  BL=1     (Cyan, Magenta, White)
Return:  n/a
```

Function 0Ch - Write Dot

```
Input:   AH=0Ch   Write Dot
         AL       Color (b7 set for XOR)
         BH       Page number
         CX       Column
         DX       Row
Return:  n/a
```

Function 0Dh - Read Dot

```
Input:   AH=0Dh   Read Dot
         BH       Page number
         CX       Column
         DX       Row
Return:  AL       Dot color
```

Function 0Eh - Write TTY Character to Active Page

```
Input:   AH=0Eh   Write TTY Character to Active Page
         AL       Character (CR, LF, BS, and BELL accepted)
         BL       Color in graphics mode
Return:  n/a
```

Function 0Fh - Get Current Video State

```
Input:   AH=0Fh   Get Current Video State
Return:  AL       Current video mode
         AH       Number of columns
         BH       Current page number
```

Function 10h - Palette Functions

```
Input:   AH=10h   Palette Functions
         AL=00    Set palette register
         BL       Palette register
         BH       Value to be set
Return:  n/a

Input:   AH=10h   Palette Functions
         AL=01    Set overscan register
         BH       Value to be set
Return:  n/a

Input:   AH=10h   Palette Functions
         AL=02    Set all palette registers and overscan
         ES:DX    Pointer to 17-byte table
Return:  n/a

Input:   AH=10h   Palette Functions
         AL=03    Toggle intensity/blink bit
         BL       1=Blink, 0=Intensity
Return:  n/a

Input:   AH=10h   Palette Functions
         AL=07    Get palette register
         BL       Palette register
Return:  BH       Palette register value

Input:   AH=10h   Palette Functions
         AL=08    Get overscan register
Return:  BH       Overscan register value

Input:   AH=10h   Palette Functions
         AL=09    Get all palette registers and overscan
         ES:DX    Pointer to 17-byte table
Return:  n/a

Input:   AH=10h   Palette Functions
         AL=10h   Set DAC color register
         BX       Color register
         DH:CH:CL Red, Green, Blue data
Return:  n/a
```

```
Input:   AH=10h   Palette Functions
         AL=12h   Set block of DAC registers
         BX       Start color register
         CX       Number of registers
         ES:DX    Pointer to RGB table
Return:  n/a

Input:   AH=10h   Palette Functions
         AL=13h   Select color page
         BL       Paging function (0-1)
                  00       Select paging mode
                  01       Select page
         BH       If BL=0 (0=4 of 64, 1=16 of 16)
                  If BL=1 (Page number 0-3, 0-15)
Return:  n/a

Input:   AH=10h   Palette Functions
         AL=15h   Get DAC color register
         BX       Color register
Return:  DH       Red value
         CH       Green value
         CL       Blue value

Input:   AH=10h   Palette Functions
         AL=17h   Get block of DAC registers
         BX       Start color register
         CX       Number of registers
         ES:DX    Pointer to RGB table
Return:  n/a

Input:   AH=10h   Palette Functions
         AL=18h   Set PEL Mask (VGA Undocumented)
         BL       PEL Mask to write
Return:  n/a

Input:   AH=10h   Palette Functions
         AL=19h   Get PEL Mask (VGA Undocumented!)
         BX       Returned PEL Mask value
Return:  n/a

Input:   AH=10h   Palette Functions
         AL=1Ah   Get current color page
         BL       Returned paging mode
         BH       Returned page number
Return:  n/a
```

```
Input:   AH=10h   Palette Functions
         AL=1Bh   Convert all DAC registers to gray-scale
         BX       Start color register
         CX       Number of registers
Return:  n/a
```

Note
Overscan Subfunction 01 incorrectly puts the data into the parameter save area offset + 11h to be consistent with IBM code.

Function 11h - Character Generator Control

```
         AL               Character generator function (0-30h)
                          0x - Alpha load (x=0-4)
                          1x - Alpha load, recalculated (x=0-2, 4)
                          2x - Graphics load (x=0-4)
                          30 - Return information

Input:   AH=11h   Character generator function
         AL=00 or AL=10h (user alpha load):
         BL       Block to load
         BH       Points
         CX       Character count
         DX       Character offset
         ES:BP    Font table pointer
Return:  n/a

Input:   AH=11h   Character generator function
         AL=01 or AL=11h (ROM 8x14 set):
         BL       Block to load
Return:  n/a

Input:   AH=11h   Character generator function
         AL=02 or AL=12h (ROM 8x8 set):
         BL       Block to load
Return:  n/a

Input:   AH=11h   Character generator function
         AL=04 or AL=14h (ROM 8x16 set):
         BL       Block to load
Return:  n/a
```

```
Input:   AH=11h   Character generator function
         AL=03    (set active block):
         BL       Value for sequencer register 3
Return:  n/a

Input:   AH=11h   Character generator function
         AL=20h    (user graphics characters):
         ES:BP    Font table pointer (chars 128-255)
Return:  n/a

Input:   AH=11h   Character generator function
         AL=21h    (user graphics load):
         BL       Rows select (0=user, 1=14, 2=25, 3=43)
         CX       Points
         DL       Rows Input: BL=0
         ES:BP    Font table pointer
Return:  n/a

Input:   AH=11h   Character generator function
         AL=22h    (ROM 8x14 set):
         BL       Rows select
Return:  n/a

Input:   AH=11h   Character generator function
         AL=23h    (ROM 8x8 set):
         BL       Rows select
Return:  n/a

Input:   AH=11h   Character generator function
         AL=24h    (ROM 8x16 set):
         BL       Rows select
Return:  n/a
```

```
Input:   AH=11h   Character generator function
         AL=30h    (return information):
         BH       Function request (0-7)
                  0 - Get INT 1F pointer
                  1 - Get INT 43 pointer
                  2 - Get ROM 8x14 pointer
                  3 - Get ROM 8x8 pointer
                  4 - Get ROM 8x8 pointer (128-255)
                  5 - Get 9x14 fudge table pointer
                  6 - Get ROM 8x16 pointer
                  7 - Get 9x16 fudge table pointer
                  (fudge font never used, but
                  point size returned should be valid)
Return:  CX       Points
         DL       Rows
         ES:BP    Table pointer
```

Function 12h - Miscellaneous Functions

```
Input:   AH=12h   Miscellaneous Function
         BL=10h   Return EGA information
Return:  BL       Memory (0=64K, 1=128K, 2=192K, 3=256K)
         BH       0 = color mode active, 1 = mono mode active
         CL       Switch settings
         CH       Feature bits

Input:   AH=12h   Miscellaneous Function
         BL=20h   Select EGA print screen routine
Return:  n/a

Input:   AH=12h   Miscellaneous Function
         BL=30h   Set alpha mode scan count
         AL       0=200, 1=350, 2=400 scans
Return:  AL=12h

Input:   AH=12h   Miscellaneous Function
         BL=31h   Palette load on mode set
         AL       0=enable, 1=disable
Return:  AL=12h

Input:   AH=12h   Miscellaneous Function
         BL=32h   Video control
         AL       0=enable, 1=disable
Return:  AL=12h
```

```
Input:   AH=12h   Miscellaneous Function
         BL=33h   gray-scale summing
         AL       0=enable, 1=disable
Return:  AL=12h

Input:   AH=12h   Miscellaneous Function
         BL=34h   Cursor emulation control
         AL       0=enable, 1=disable
Return:  AL=12h

Input:   AH=12h   Miscellaneous Function
         BL=35h   Display control
         AL       Function request
                  00 - Adapter off (initial)
                  01 - Planar on (initial)
                  02 - Active off
                  03 - Inactive on
                  ES:DX    Pointer to 128-byte buffer
Return:  n/a

Input:   AH=12h   Miscellaneous Function
         BL=36h   Video data control
         AL       0=enable, 1=disable
Return:  AL=12h
```

Function 13h - Write String Functions

```
Input:   AH=13h   Write String function
         AL       String function
                  0 - BL=attribute, string=char, char,...
                  1 - BL=attribute, string=char, char,... cursor moved
                  2 - String=char, attr, char, attr...
                  3 - String=char, attr, char, attr... cursor moved
         BL       Attribute (if AL=0 or 1)
         BH       Page number
         CX       Character count
         DX       Start cursor position
         ES:BP    String pointer
Return:  n/a
```

Note
If any scrolling occurs, the active page will be scrolled, not the requested page! This “feature” is also present in IBM's code and it has been determined that it is better to reproduce this in our code for compatibility.

Function 14h To Function 19h - Null Functions

Functions 14h to 19h are reserved by VGA definition. Any requests of these functions will just simply return and nothing happens.

```
Input:   n/a
Return:  n/a
```

Function 1Ah - Read/write Display Combination Code

```
Input:   AH=1Ah   Read/write Display Combination Code
         AL       0=read, 1=write
         BL       Active DCC (r/w)
         BH       Alternate DCC (r/w)
Return:  AL=1Ah
         BX       Set as above
```

Function 1Bh - Return Functionality/state Information

```
Input:   AH=1Bh   Return Functionality/State Information
         BX       Implementation type (must be 00)
         ES:DI    Pointer to buffer
Return:  AL=1Bh
         ES:DI    Contains information
```

The format of the information block is as follows:

Offset	Type	Description
00	DD	Pointer to static functionality table
04	DB	CRT video mode----------+
05	DW	CRT columns |
07	DW	VRAM page length |
09	DW	VRAM start address |
0B 8h x	DW	Cursor row/column for 8 pages +These are values
1B	DW	Cursor type |directly copied
1D	DB	Active page |from low memory.
1E	DW	CRTC address |
20	DB	Port 3D8 data |
21	DB	Port 3D9 data |
22	DB	Number of rows |
23	DW	Point size ----------+
25	DB	Active DCC
26	DB	Auxiliary DCC
27	DW	Number of colors this mode
29	DB	Number of pages this mode
2A	DB	Number of scans this mode (0=200 1=350 2=400 3=480)
2B	DB	Primary character block
2C	DB	Secondary character block
2D	DB	Misc. state info (see table below)
2E 3h x	DB	Reserved (set to zero)
31	DB	VRAM size (0=64K 1=128K 2=192K 3=256K)
32	DB	Save pointer state info (see table below)
33 Dh x	DB	Reserved (set to zero)

Table 4-1 Information Block Offset 2D

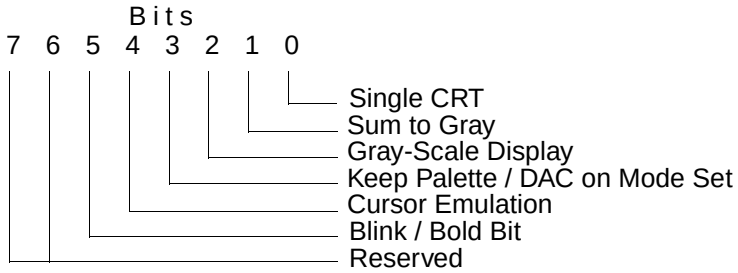
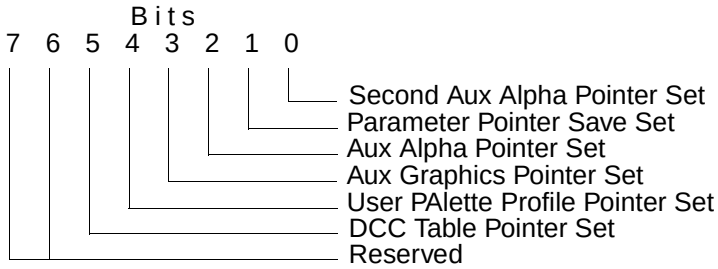


Table 4-2 Information Block Offset 32



Function 1Ch - Save/restore Video State

```
Input:   AH=1Ch   Save/Restore Video State
         AL       Function request (0-2)
                  00 - Return size of buffer
                  01 - Save video state
                  02 - Restore video state
         ES:BX    Pointer to buffer
         CX       Bit map of request
                  001 - Video h/w status
                  010 - Video data arena
                  100 - Video DAC and color registers
Return:  AL=1Ch
         BX       Buffer size if AL input =00
```

Note
While emulating IBM code this function does not save the latch data or color select register 14h of the Attributes Controller. Because it doesn't get saved or restored, we don't touch it at all.

The format of the save buffer is as follows:

Offsets

```
00h   DW       Offset to the Video Hardware Status buffer
02h   DW       Offset to the Video Data Arena buffer
04h   DW       Offset to the DAC and Color Registers buffer
06h 0Dh x DW   Reserved (uninitialized)
```

Video Hardware Status

```
00h   DB       Sequencer (3C4) index
01h   DB       CRTC (3D4) index
02h   DB       Graphics controller (3CE) index
03h   DB       Attributes controller (3C0) index
04h   DB       Feature control register (3DA) data
05h 4h x DB    Sequencer data
09h   DB       Misc. output register (3C2) data
0Ah 19h x DB   CRTC data
23h 14h x DB   Attributes controller data
37h 9h x DB    Graphics controller data
40h   DW       CRTC base port
42h 4h x DB    Latch data
```

Video Data Arena

```
00h   DB       EquipLow AND 30h
01h   DB       CRTMode
02h   DW       CRTColumns
04h   DW       VRAMPageLength
06h   DW       VRAMStartAddr
08h  8h x DW   CursorRowCol
18h   DW       CursorType
1Ah   DB       ActivePage
1Bh   DW       CRTCAddr
1Dh   DB       Port3D8Data
1Eh   DB       Port3D9Data
1Fh   DB       MaxRow
0h    DW       Points
22h   DB       EGAInfo
23h   DB       DIPInfo
24h   DB       VGAInfo
25h   DB       DCCIndex
26h   DD       SavePtr
2Ah   DD       Vec05
2Eh   DD       Vec1D
32h   DD       Vec1F
36h   DD       Vec43
```

DAC and Color Registers

```
00h   DB             DAC state (3C7) AND 01h (1=read mode, 0=write mode)
01h   DB             DAC (3C8) index (adjusted)
02h   DB             Pel mask (3C6) data
03h 300h x DB        DAC RGB data
```

5 VESA FUNCTION SUMMARY

These functions are defined by the Video Electronics Standards Association. They cover issues of inquiry on chip capability, and available modes. The following shows the SPC8110F0A implementation of these functions. The SPC8110F0A BIOS versions 1.x will support VESA VBE 1.2. BIOS versions 1.2 and greater to be specified later, will support VBE 1.2, VBE/DDC 1.0 and VBE/PM 1.1. BIOS versions 2.0 and greater will additionally support VBE 2.0. In the future, when VESA passes the VBE/FP extensions, the SOLLEX extensions found in Section 6 will be eliminated from the BIOS.

5.1 Status Information

Every function returns status information in the AX register. The format of the status word is as follows:

```
AL ==    4Fh:     Function is supported
AL !=    4Fh:     Function is not supported
AH ==    00h:     Function call successful
AH ==    01h:     Function call failed
AH ==    02h:     Function not supported in current hardware configura-
                  tion
AH ==    03h:     Function call invalid in current video mode
```

5.2 VESA Functions

Function 00h - Return VBE Controller Information

```
 Input:   AX       = 4F00h  Return VBE Controller Information
          ES:DI    =        Pointer to buffer in which to place VbeInfoBlock
                            structure (VbeSignature should be set to 'VBE2' when
                            function is called to indicate VBE 2.0 information is
                            desired and the information block is 512 bytes in
                            size.)
 Return:  AX       =        VBE Return Status
```

Note
All other registers are preserved.

The VBE 1.2 compatible information block has the following structure:

```
VGAInfoBlock struc
    VESASignature     db    'VESA'       ; 4 signature bytes
    VESAVersion       dw    102h         ; VESA version number
    OEMStringPtr      dd    ?            ; Pointer to OEM string
    Capabilities      dd    ?            ; capabilities of the video environment
    VideoModePtr      dd    ?            ; pointer to supported VESA modes
    TotalMemory       dw    ?            ; Number of 64kb memory blocks on board
    Reserved          db    236 dup (0) ; Remainder of VGAInfoBlock
VGAInfoBlock ends
```

The VBE 2.0 information block has the following structure:

```
VbeInfoBlock struc
VbeSignature         db    'VESA'      ; VBE Signature
VbeVersion           dw    0200h       ; VBE Version
OemStringPtr         dd    ?           ; Pointer to OEM String
Capabilities         db    4 dup (?)   ; Capabilities of graphics cont.
VideoModePtr         dd    ?           ; Pointer to Video Mode List
TotalMemory          dw    ?           ; Number of 64kb memory blocks
                                       ; Added for VBE 2.0
OemSoftwareRev       dw    ?           ; VBE implementation Software revision
OemVendorNamePtr     dd    ?           ; Pointer to Vendor Name String
OemProductNamePtr    dd    ?           ; Pointer to Product Name String
OemProductRevPtr     dd    ?           ; Pointer to Product Revision String
Reserved             db    222 dup (?) ; Reserved for VBE implementation scratch
                                       ; area
OemData              db    256 dup (?) ; Data Area for OEM Strings
VbeInfoBlock ends
```

Function 01h - Return VBE Mode Information

```
Input:   AX       = 4F01h  Return VBE mode information
         CX       =        Mode number
         ES:DI    =        Pointer to ModeInfoBlock structure

Return:  AX       =        VBE Return Status
```

Note
All other registers are preserved.

The mode information block has the following structure:

```
ModeInfoBlock   struc
; Mandatory information for all VBE revisions
ModeAttributes       dw    ?           ; mode attributes
WinAAttributes       db    ?           ; window A attributes
WinBAttributes       db    ?           ; window B attributes
WinGranularity       dw    ?           ; window granularity
WinSize              dw    ?           ; window size
WinASegment          dw    ?           ; window A start segment
WinBSegment          dw    ?           ; window B start segment
WinFuncPtr           dd    ?           ; pointer to window function
BytesPerScanLine     dw    ?           ; bytes per scan line

; Mandatory information for VBE 1.2 and above
XResolution          dw    ?           ; horizontal resolution in pixels or
                                         characters
YResolution          dw    ?           ; vertical resolution in pixels or characters
XCharSize            db    ?           ; character cell width in pixels
YCharSize            db    ?           ; character cell height in pixels
NumberOfPlanes       db    ?           ; number of memory planes
BitsPerPixel         db    ?           ; bits per pixel
```

```
NumberOfBanks         db    ?           ; number of banks
MemoryModel           db    ?           ; memory model type
BankSize              db    ?           ; bank size in KB
NumberOfImagePages    db    ?           ; number of images
Reserved              db    1           ; reserved for page function

; Direct Color fields (required for direct/6 and YUV/7 memory models)
RedMaskSize           db    ?           ; size of direct color red mask in bits
RedFieldPosition      db    ?           ; bit position of lsb of red mask
GreenMaskSize         db    ?           ; size of direct color green mask in bits
GreenFieldPosition    db    ?           ; bit position of lsb of green mask
BlueMaskSize          db    ?           ; size of direct color blue mask in bits
BlueFieldPosition     db    ?           ; bit position of lsb of blue mask
RsvdMaskSize          db    ?           ; size of direct color reserved mask in bits
RsvdFieldPosition     db    ?           ; bit position of lsb of reserved mask
DirectColorModeInfo   db    ?           ; direct color mode attributes

; Mandatory information for VBE 2.0 and above
PhysBasePtr           dd    ?           ; physical address for flat memory
                                          frame buffer
OffScreenMemOffset    dd    ?           ; pointer to start of off screen memory
OffScreenMemSize      dw    ?           ; amount of off screen memory in 1k units
Reserved              db    206 dup (?)  ; remainder of ModeInfoBlock
ModeInfoBlock ends
```

Function 02h - Set VBE Mode

```
Input:   AX       = 4F02h  Set VBE Mode
         BX       =        Desired Mode to set
                           D0-D8 = Mode number
                           D9-D13 = Reserved (must be 0)
                           D14 = 0 Use windowed frame buffer model
                               = 1 Use linear/flat frame buffer model
                           D15 = 0 Clear display memory
                               = 1 Don't clear display memory
Return:  AX       =        VBE Return Status
```

Note
All other registers are preserved.

Function 03h - Return current VBE Mode

```
Input:   AX       = 4F03h  Return current VBE Mode
Return:  AX       =        VBE Return Status
         BX       =        Current VBE mode
                           D0-D13 = Mode number
                           D14 = 0 Windowed frame buffer model
                               = 1 Linear/flat frame buffer model
                           D15 = 0 Memory cleared at last mode set
                               = 1 Memory not cleared at last mode set
```

Note
All other registers are preserved.

Function 04h - Save/Restore state

```
Input:   AX       =        4F04h Save/Restore state
         DL       =        00h Return save/restore state buffer size
                  =        01h Save state
                  =        02h Restore state
         CX       =        Requested states
                  D0 =     Save/restore controller hardware state
                  D1=      Save/restore BIOS data state
                  D2=      Save/restore DAC state
                  D3=      Save/restore Register state
         ES:BX    =        Pointer to buffer (if DL <> 00h)
Return:  AX       =        VBE Return Status
         BX       =        Number of 64-byte blocks to hold the state buffer (if
                           DL=00h)
```

Note
All other registers are preserved.

Function 05h - Display Window Control

```
Input:   AX       = 4F05h  VBE Display Window Control
         BH       = 00h    Set memory window
                  = 01h    Get memory window
         BL       =        Window number
                  = 00h    Window A
                  = 01h    Window B
         DX       =        Window number in video memory in window granularity
                           units (Set Memory Window only)
Return:  AX       =        VBE Return Status
         DX       =        Window number in window granularity units (Get Memory
                           Window only)
```

Function 06h - Set/Get Logical Scan Line Length

Input:	AX	= 4F06h	VBE Set/Get Logical Scan Line Length
	BL	= 00h	Set Scan Line Length in Pixels
		= 01h	Get Scan Line Length
		= 02h	Set Scan Line Length in Bytes
		= 03h	Get Maximum Scan Line Length
	CX	=	If BL=00h Desired Width in Pixels If BL=02h Desired Width in Bytes (Ignored for Get Functions)
Return:	AX	=	VBE Return Status
	BX	=	Bytes Per Scan Line
	CX	=	Actual Pixels Per Scan Line (truncated to nearest complete pixel)
	DX	=	Maximum Number of Scan Lines

Function 07h - Set/Get Display Start

Input:	AX	= 4F07h	VBE Set/Get Display Start Control
	BH	= 00h	Reserved and must be 00h
	BL	= 00h	Set Display Start
		= 01h	Get Display Start
		= 80h	Set Display Start during Vertical Retrace
	CX	=	First Displayed Pixel In Scan Line (Set Display Start only)
	DX	=	First Displayed Scan Line (Set Display Start only)
Return:	AX	=	VBE Return Status
	BH	=	00h Reserved and will be 0 (Get Display Start only)
	CX	=	First Displayed Pixel In Scan Line (Get Display Start only
	DX	=	First Displayed Scan Line (Get Display Start only)

Function 08h - Set/Get DAC Palette Format

Input:	AX	= 4F08h	VBE Set/Get Palette Format
	BL	= 00h	Set DAC Palette Format
		= 01h	Get DAC Palette Format
	BH	=	Desired bits of color per primary (Set DAC Palette Format only)
Return:	AX	=	VBE Return Status
	BH	=	Current number of bits of color per primary

Function 09h - Set/Get Palette Data

```
Input:   AX       = 4F09h  VBE Load/Unload Palette Data
         BL       = 00h    Set Palette Data
                  = 01h    Get Palette Data
                  = 02h    Set Secondary Palette Data
                  = 03h    Get Secondary Palette Data
                  = 80h    Set Palette Data during Vertical Retrace with Blank
                           Bit on
         CX       =        Number of palette registers to update
         DX       =        First palette register to update
         ES:DI    =        Table of palette values (see below for format)
Return:  AX       =        VBE Return Status
```

Format of Palette Values: Alignment byte, Red byte, Green byte, Blue byte

Function 0Ah - Return VBE Protected Mode Interface

```
Input:   AX       = 4F0Bh  VBE 2.0 Protected Mode Interface
         BL       = 00h    Return protected mode table
Return:  AX       =        Status
         ES       =        Real Mode Segment of Table
         DI       =        Offset of Table
         CX       =        Length of Table including protected mode code
                           (for copying purposes)
```

The format of the table is as follows:

```
ES:DI + 00h       Word Offset in table of Protected mode code for
                  Function 5 for Set Window Call
ES:DI + 02h       Word Offset in table of Protected mode code for
                  Function 7 for set Display Start
ES:DI + 04h       Word Offset in table of Protected mode code for
                  Function 9 for set Primary Palette data
ES:DI + 06h       Word Offset in table of Ports and Memory Locations
                  that the application may need I/O privilege for.
                  (Optional: if unsupported this must be 0000h)
                  (See Sub-table for format)
ES:DI + ?         Variable remainder of Table including Code
```

Function 10h - Display Power Management Extensions
(valid for BIOS version 1.1 or greater)

The VESA VBE sub-Function 10h is used to implement the VBE/PM services. The VBE / PM services are defined as follows:

Sub-Function 00h - Report VBE / PM Capabilities

```
Input:   AH       = 4Fh    VESA Extension.
         AL       = 10h    VBE/PM Services.
         BL       = 00h    Report VBE/PM Capabilities.
         CX       = 00h    Controller unit number (00 = primary controller).
         ES:DI             Null pointer, must be 0000:0000h in version 1.0.
                           Reserved for future use.
Return:  AX       =        Status.
         BH       =        Power saving state signals supported by the
                           controller. 1 = supported, 2 = not supported
                           bit 0 STANDBY
                           bit 1 SUSPEND
                           bit 2 OFF
                           bit 3 REDUCED ON
                           bits 4-7 reserved for future power control of the
                                    display controller or other related
                                    circuits.
         BL       =        VBE/PM version number (0001 0000b for this version).
                           bits 0-3 minor version number
                           bits 4-7 major version number
         CX       =        Unchanged
         ES:DI             Unchanged
All other registers may be destroyed.
```

Sub-Function 01h - Set Display Power State

```
Input:   AH       = 4Fh    VESA Extension.
         AL       = 10h    VBE/PM Services.
         BL       = 01h    Set Display Power State.
         BH       = 00h    ON
                  = 01h    STANDBY
                  = 02h    SUSPEND
                  = 04h    OFF
                  = 08h    REDUCED ON
                  All other BH values are currently undefined and are reserved
                  for future power control of the display controller.
         CX       = 00h    Controller unit number (00 = primary controller).
Return:  AX       =        Status.
         BH       =        Unchanged
         CX       =        Unchanged
All other registers may be destroyed.
```

Sub-Function 02h - Get Display Power State

```
Input:   AH       = 4Fh    VESA Extension.
         AL       = 10h    VBE/PM Services.
         BL       = 02h    Get Display Power State.
         CX       = 00h    Controller unit number (00 = primary controller).
Return:  AX       =        Status.
         BH                Power state currently requested by the controller.
                  = 00h    ON
                  = 01h    STANDBY
                  = 02h    SUSPEND
                  = 04h    OFF
                  = 08h    REDUCED ON
                  All other BH values are reserved and may be used to signal
                  other power saving states in future revisions of VBE/PM. For
                  future compatibility, applications written for VBE/PM 1.0
                  should ignore the value of bits 4 to 7.
         CX       =        Unchanged
All other registers may be destroyed.
```

Function 15h - Display Identification Extensions
(valid for BIOS version 1.2 or greater)

The VESA VBE sub-Function 15h is used to implement the VBE / DDC services. The VBE / DDC services are defined as follows:

Sub-Function 00h - Report VBE / DDC Capabilities

```
Input:   AH       = 4Fh    VESA Extension.
         AL       = 15h    VBE/DDC Services.
         BL       = 00h    Report VBE/DDC Capabilities.
         CX       = 00h    Controller unit number (00 = primary controller).
         ES:DI             Null pointer, must be 0000:0000h in version 1.0.
                           Reserved for future use.
Return:  AX       =        Status
         BH       =        Approximate time, in seconds, rounded up, to transfer
                           one EDID block (128 bytes).
         BL       =        DDC level supported.
                           bit 0 = 0 DDC1 not supported
                                 = 1 DDC1 supported
                           bit 1 = 0 DDC2 not supported
                                 = 1 DDC2 supported
                           bit 2 = 0 screen not blanked during data transfer
                                 = 1 screen blanked during data transfer
         CX       =        Unchanged
         ES:DI             Unchanged
All other registers may be destroyed.
```

Sub-Function 01h - Read EDID

Input:	AH	= 4Fh	VESA Extension.
	AL	= 15h	VBE/DDC Services.
	BL	= 01h	Read EDID.
	CX	= 00h	Controller unit number (00 = primary controller).
	DX	= 00h	EDID block number. Zero is the only valid value in version 1.0.
	ES:DI		Pointer to the area in which the EDID block (128 bytes) shall be returned.
Return:	AX	=	Status
	BH	=	Unchanged
	CX	=	Unchanged
	ES:DI		Pointer to the area in which the EDID block (128 bytes) is returned.

All other registers may be destroyed.

Sub-Function 02h - Read VDIF Block

Input:	AH	= 4Fh	VESA Extension.
	AL	= 15h	VBE/DDC Services.
	BL	= 02h	Read VDIF block.
	CX	= 00h	Controller unit number (00 = primary controller).
	DX	= 00h	VDIF block number (64 byte block).
	ES:DI		Pointer to the area in which the VDIF block (64 bytes) shall be returned.
Return:	AX	=	Status
	BH	=	Unchanged
	CX	=	Unchanged
	ES:DI		Pointer to the area in which the VDIF block (64 bytes) is returned.

All other registers may be destroyed.

6 SOLLEX FUNCTION SUMMARY

These functions are defined by Seiko Epson as a generic interface for functions not covered by the Video Electronics Standards Association or IBM's standard video BIOS. This specification is soon to be outdated, as VESA is now in the process of developing Flat Panel, Power Management and Display Data Channel specifications. With version 2.x of the SPC8110 BIOS, the Sollex functions will be removed, to be replaced by the VESA specifications. The following is incomplete since much of the functions that Sollex was designed for have been removed. Only functions 00h, 01h, 02h, 04h, 05h, 07h, 08h, 0Ah and FFh are valid for the SPC8110F0A. Only function 04h, 0Ah and FFh have been marked up for SPC8110F0A specific implementation details.

6.1 Sollex Status Information

Every function returns status information in the AX register. The format of the status words is as follows:

```
AL ==    7Fh:     Function is supported
AL !=    7Fh:     Function is not supported
AH ==    00h:     Function call successful
AH ==    01h:     Function call fails
```

6.2 Sollex Reserved Bits

All reserved bit returns will return 0 by default, unless otherwise noted.

6.3 Sollex Functions

Function 00h - Return Extensions Info

```
Input:   AH=7Fh   SOLLEX Support
         AL=00h   Return Extensions Info
Return:  AX=      017Fh    (Function call fails)
                           No longer valid for SPC8110F0A
```

Function 01h - Adapter Control

```
Input:   AH=7Fh   SOLLEX Support
         AL=01h   Adapter Control
         BL=00h   Set Adapter
         CX       Adapter Request
Return:  AX       Status

Input:   AH=7Fh   SOLLEX Support
         AL=01h   Adapter Control
         BL=01h   Get Adapter
Return:  AX       Status
         BX       Adapter type
         DX       Display type

Input:   AH=7Fh   SOLLEX Support
```

```
          AL=01h   Adapter Control
          BL=02h   Return Adapter Support
          CX       Adapter Request
Return:   AX       Status
```

Table 6-1 Sub-Function 00h: Set Adapter

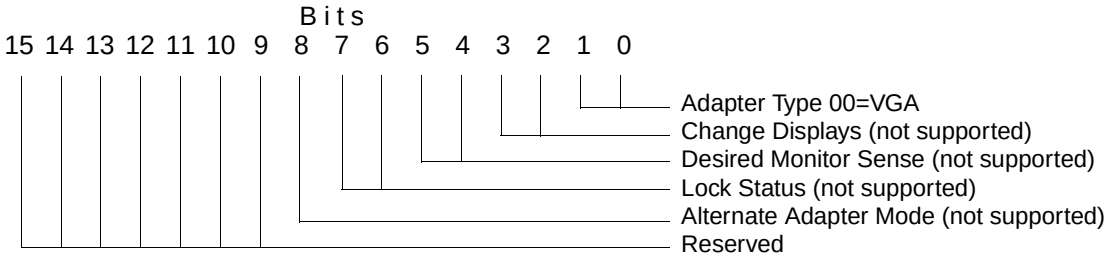
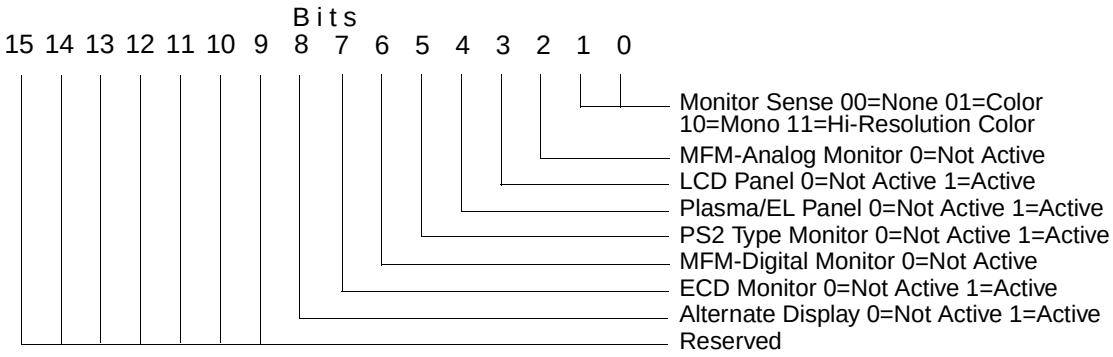


Table 6-2 Sub-Function 01h: Get Adapter



The Monitor Sense return is based on MONID lines 0 - 2, which produces a total of 8 combinations. The following table shows the interpretation of these 8 combinations into 4.

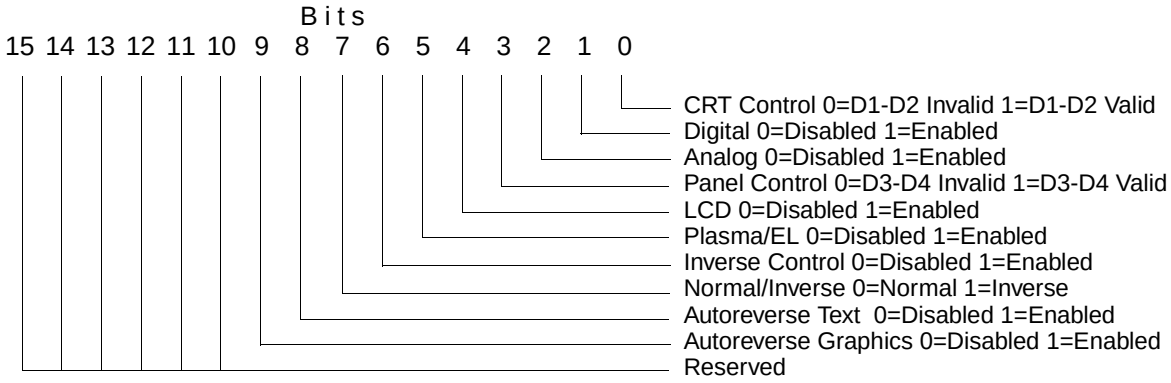
MONID	Monitor Sense
0 0 0b	B1=1 B0=0
0 0 1b	B1=1 B0=0
0 1 0b	B1=1 B0=1
0 1 1b	B1=1 B0=1
1 0 0b	B1=1 B0=0
1 0 1b	B1=1 B0=0
1 1 0b	B1=0 B0=1
1 1 1b	B1=0 B0=0

Sub-Function 02h uses the same Adapter Request format as Sub-Function 00h to determine whether the requested adapter setting could be successfully set in the current environment.

Function 02h - Display Output Control

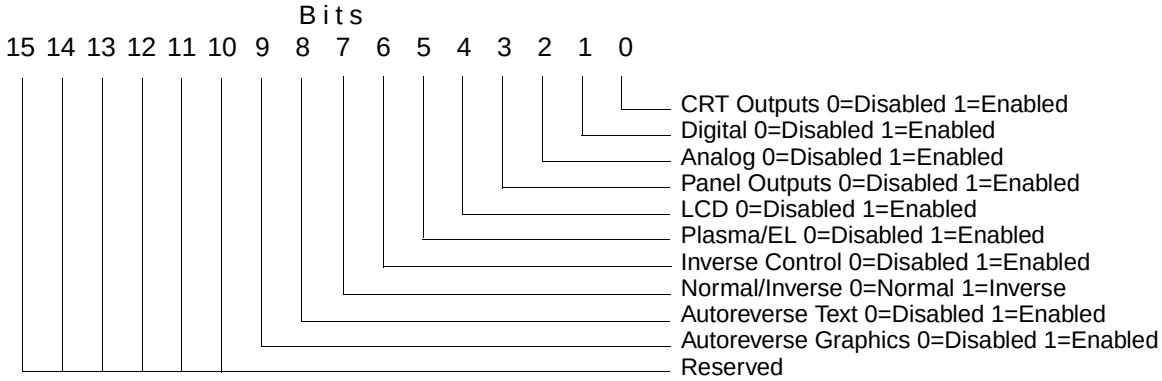
```
Input:   AH=7Fh   SOLLEX Support
         AL=02h   Display Output Control
         BL=00h   Set Display Output
         CX       Display Setting
Return:  AX       Status

Input:   AH=7Fh   SOLLEX Support
         AL=02h   Display Output Control
         BL=01h   Get Display Output
Return:  AX       Status
         BX       Display Output
         CX       Displays attached
```

Table 6-3 Sub-Function 00h: Set Display Output



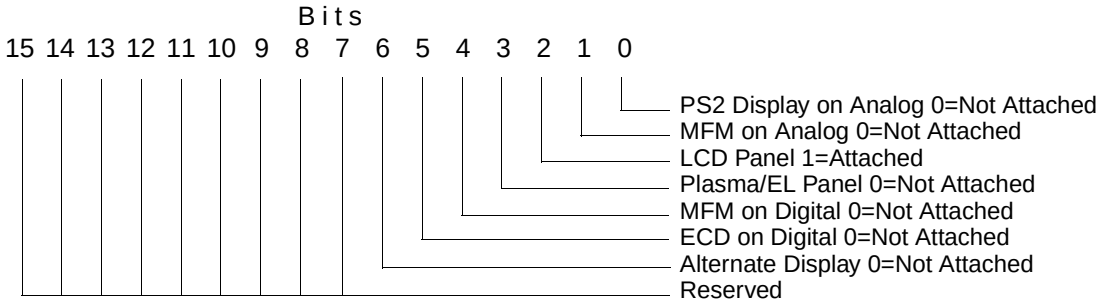
Note
use Set Display Output to set CRT, LCD or none, but not multiple outputs. Use chip specific function FFh to set simultaneous display.

Table 6-4 Sub-Function 01h: Get Display Output



The CX register contains the following bit mask that describes the attached displays:

Table 6-5 Display Attached



Function 03h - Video Support Control

```
Input:   AH=7Fh   SOLLEX Support
         AL=03h   Video Support Control
         BL=xx    Return Support Info
         CX       n/a
Return:  AX=017fh (function call fails)
                  No longer valid for SPC8110F0A
```

Function 04h - Power Control

```
Input:   AH=7Fh   SOLLEX Support
         AL=04h   Power Control
         BL=00h   Set Power State
         CX       Power State (0 to Maximum State)
Return:  AX       Status

Input:   AH=7Fh   SOLLEX Support
         AL=04h   Power Control
         BL=01h   Get Power State
Return:  AX       Status
         CX       Power State
         DX       Maximum State

Input:   AH=7Fh   SOLLEX Support (UNSUPPORTED in SPC8110F0A)
         AL=04h   Power Control
         BL=02h   Set Time Out Reset
         CX       Time Out Reset (0 to Maximum Time Out Reset)
Return:  AX       Status AL=7Fh, AH=01h

Input:   AH=7Fh   SOLLEX Support (UNSUPPORTED in SPC8110F0A)
         AL=04h   Power Control
         BL=03h   Get Time Out Reset
Return:  AX       Status AL=7Fh, AH=01h
```

Sub-Function 00h: Set Power State

Set Power State according to table below:

Sub-Function 01h: Get Power State

Returns Power State according to table below:

State	0	1	2	3	4
VESA Name	On	Stand By	Suspend	Off	Reduced On
Reduced Gray Scale	yes	yes	yes	yes	yes
Slow PCLK	no	no	no	no	no
Display Off	no	yes	yes	yes	no
DPMS State	on	stand by	suspend	off	on
Backlight Off	no	yes	yes	yes	no
PCLK Off	no	no	no	yes	no
MCLK Off	no	no	yes	yes	no
Relative Power Saving	none	low	medium	high	very low

Note
these are the BIOS defaults and may be configured by the OEM

Function 05h - Load Register

```
Input:   AH=7Fh   SOLLEX Support
         AL=05h   Load Register
         ES:DI    Pointer to table (format of table is available in Sollex
                  Specification)
Return:  AX=017Fh (function call fails)
                  No longer valid for SPS8110F0B
```

Function 06h - Multiple Font Control

```
Input:   AH=7Fh   SOLLEX Support
         AL=06h   Multiple Font Control
Return:  AX=017Fh (function call fails)
                  No longer valid for SPS8110F0B
```

Function 07h - Fill Video RAM

```
Input:   AH=7Fh   SOLLEX Support
         AL=07h   Fill Video RAM
         BL       Amount of Video RAM to Fill
                  00h=all currently addressable memory without a page switch
                  01h=all currently addressable memory including a page switch
                  (currently addressable memory is mode dependent)
         CX       Pattern to write (text mode)
                  Color to write (graphics mode)
Return:  AX=017Fh (function call fails)
```

```
                  No longer valid for SPS8110F0B
```

Function 08h - Autocenter Control

```
Input:   AH=7Fh   SOLLEX Support
         AL=08h   Autocenter control
         BL=00h   Set Autocenter control
         CX       0000h Disable Autocenter
                  0001h Enable Autocenter
Return:  AX       Status

Input:   AH=7Fh   SOLLEX Support
         AL=08h   Autocenter control
         BL=01h   Get Autocenter state
Return:  AX       Status
         BL       Autocenter control status
                  00h=Disabled
                  01h=Enabled
```

Function 09h - Lookup Table Control

```
Input:   AH=7Fh   SOLLEX Support
         AL=09h   Lookup Table Control
Return:  AX=017Fh (function call fails)
                  No longer valid for SPS8110F0B
```

Function 0Ah - Non-Standard Font Control

(SPC8110F0A uses 19pt.font for h/w text expansion*)

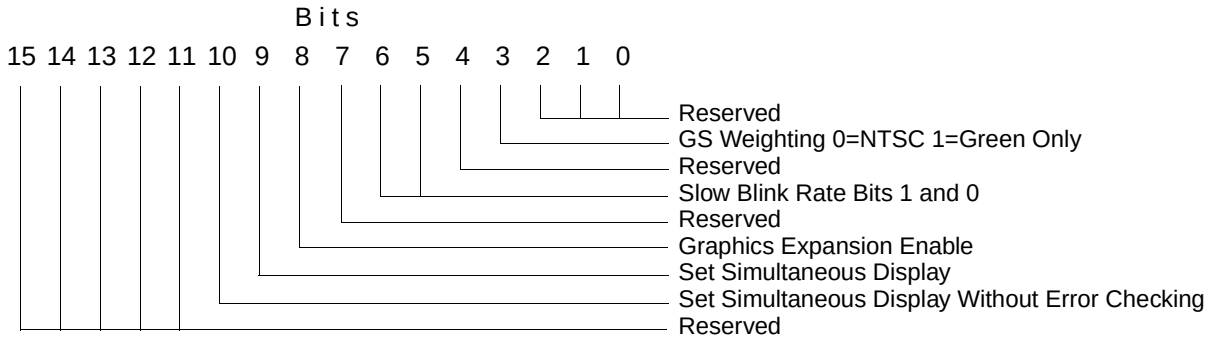
```
Input:   AH=7Fh   SOLLEX Support
         AL=0Ah   Non-Standard Font Control
         BL=00h   Set Non-Standard Font Control state
         CL       Font Width
         CH       Font Height (if CX=0, it will set normal system
                  font to be default)
Return:  AX       Status

Input:   AH=7Fh   SOLLEX Support
         AL=0Ah   Non-Standard Font Control
         BL=01h   Get Non-Standard Font Control state
Return:  AX=      Status
         BL=      Font Width
         BH=      Font Height (normal system font will return proper
                  values not 0 as in the set)
         ES:DI    pointer to table of available fonts
                  (format width, height, width, height ... 00, 00)
```

* use 19 point for set to enable Hardware Text Expansion

Function FFh - Chip Specific Function

```
Input:   AH=7Fh   SOLLEX Support
         AL=FFh   Chip Specific Function
         BL=00h   Set
         CX       Requested State
         DH       Graphic expansion line number (if D8 of CX is 1)
Return:  AX       Status

Input:   AH=7Fh   SOLLEX Support
         AL=FFh   Chip Specific Function
         BL=01h   Get
Return:  AX=      Status
         BH=      Chip Revision Product Code
         BL=      Chip Revision
         CX=      Current State
         DH=      Graphic expansion line number (if D8 of CX is 1)
```

Table 6-6 Chip Specific Function



The Slow Blink Rate is set as follows:

Slow Blink Rate		Cursor Frequency	Text/Graphics Frequency
Bit 1	Bit 0		
0	0	4 Hz	2 Hz
0	1	2 Hz	1 Hz
1	0	1 Hz	0.5 Hz
1	1	0 Hz	0 Hz

7 PHYSICAL LAYOUT OF THE BIOS

7.1 Structure

The BIOS will exist in a 32 KB section of ROM located at C000h or any other configurable location. Segment references within the code will self define at initialization time, however an initial BIOS segment must be configured into the ROM image.

7.1.1 VL-Bus

The layout of this segment is as follows:

Initialization Routine	Function Dispatch	Function Calls	Video Parameter Table	Character Fonts 8, 14, 16, & 18 or 19 if needed	Extensions Initialization	Extensions Dispatch	Extensions Function Calls	Extensions Parameter Tables

7.1.2 PCI Bus

The layout of this segment is as follows:

Initialization Routine	PCI Header	Function Dispatch	Function Calls	Video Parameter Table	Character Fonts 8, 14, 16, & 18 or 19 if needed	Extensions Initialization	Extensions Dispatch	Extensions Function Calls	Extensions Parameter Tables

7.2 IBM Notice, Copyright Notice

Some software expects the word 'IBM' to be located at C000:001E to determine if it is running on an EGA or better. We have initially put the word 'IBM' at this location, but, if the BIOS is relocated to another segment, E000 for example, software that does check at C000:001E will not work. There is very little chance of running across current software with this test.

The BIOS will also contain the string “Copyright (c) Seiko Epson Corp. 1987, 1996. All rights reserved” in two locations. One of these must stay in the code, the other is part of the header that is displayed on power-up. The power-up string is configurable and can be overwritten by the OEM.

8 SMALL FORM FACTOR PANEL SUPPORT ISSUES

Running VGA software on a small panel poses a couple of problems with viewing area and software compatibility. These issues are discussed below. In addition, to aid in software development for embedded systems (typically small panel sizes), the SPC8110F0A BIOS has special non-standard mode numbers which match up well with the common vertical panel sizes.

8.1 Special Mode Support

To support a sub-VGA panel (or any panel), the panel type is selected by using the configuration MD inputs . The pin polarities serve as an index into a list of supported panels (see section on configuration inputs for the list). (Note: the BIOS can also be custom configured for a given panel size rather than use the MD line inputs.) Once a non-VGA panel size has been selected, the BIOS will attempt to display VGA modes on this panel by changing standard VGA registers (and some non-VGA registers). To further enhance small panel support, the BIOS also has custom planar graphics modes targeted for use with the most common vertical resolutions of small panels. They are as follows:

Resolution	Mode No.
200	41H
240	42H
256	43H
320	44H
400	45H

These modes are particularly useful for embedded system design where VGA standard modes are not required and the OEM desires a mode customized for the panel size. If the mode is set (using a standard IBM mode set), the resulting resolution is 640 pixels wide by the number of scan lines indicated above. The OEM application can then trim back the CRTC offset register to a horizontal width appropriate for the application. For example, when selecting mode 42h for use with a 320x240 panel, the mode will be set with a horizontal offset value of 640 pixels or 28h in CRTC[13h]. To adjust to a line length of 320 pixels, the application may need to program the value to 14h.

These modes are similar in functionality to standard mode 12h planar graphics modes, and use all standard VGA programming techniques (write/read modes, plane enables, etc.) and memory locations.

8.1.1 Viewing Area

The major problem with small form factor panels, is that IBM defined its VGA modes around two basic resolutions: 640 x 480 and 720 x 400. To get the information presented in these modes onto the smaller panels, the BIOS incorporates some workarounds to display this data. These methods differ for Text and Graphics modes. Unfortunately, the BIOS workarounds do have their limitations.

Text Modes

Standard VGA text mode (mode 3) typically displays 80 x 25 characters on a monitor with a 720x400 resolution. To display the same information on a small panel, the graphics chip truncates the 720 text modes to 640 by using an 8-dot clock instead of the standard 9-dot clock; unfortunately for horizontal resolutions there is nothing else we can do to get more information on the screen, therefore if a panel is smaller than 640 dots wide we will truncate information off the right side of the screen. In the vertical direction the BIOS has many more options. For panel sizes less than 400 lines, the BIOS uses the 8 pixel high characters instead of the regular 16 pixel high characters, so that at least 25 lines can be visible on the screen. For all other panel sizes, the BIOS still uses the normal 16 pixel high font.

Graphics Modes

As with text modes, the horizontal display will be clipped, but, in the vertical direction, the BIOS uses different techniques. For panels less than 400 lines, the BIOS changes the modes which normally double 200 lines to 400 scan lines, to be an undoubled 200 lines, therefore on panels 200 to 400 lines, all information will be displayed. For graphics modes that are a full 480 scan lines, the BIOS cannot undouble the mode, so they will be clipped in the vertical direction as well.

8.1.2 Software Compatibility

The major concern when modifying standard mode values (including font and offset information) is that commercially available software sometimes does not compensate for these changes. Some VGA aware text mode software expects a standard VGA font to be present (16 pixel high) and may not expect this to change. (Programs that modify the text cursor, or program their own font probably will not work correctly.) In addition, most DOS software expects text mode to be 80 characters wide and if the Horizontal Offset is modified (in CRTC[13h]) , the display will get a “barber pole” effect instead of being clipped, so caution should be taken when changing the Horizontal Offset for standard VGA modes.

8.1.3 Programming and Driver Considerations

Because the special modes listed above will act like Mode 12h (planar graphics) in memory planes, colors and memory addresses, it is very easy for OEMs to develop software in Mode 12h on a standard VGA, and port to this new environment. Applications should modify the offset in the standard Mode 12h simulated environment to be the same as with the custom environment, then when you are ready to port to your new platform, change the mode set from Mode 12h to a VESA Mode 4xh function call.

Appendix A BIOS CONFIGURATION INPUTS

A.1 Panel Type

There are 32 possible panels types supported by the SPC8110F0A. The panel type configuration inputs come in on MDA9-12 and are read back in Auxiliary Register Index 0Ch bits 1-4.

Pin Name	Configuration Pin Functionality
MDA8	B0 of panel table
MDA9	B1 of panel table
MDA10	B2 of panel table
MDA11	B3 of panel table
MDA12	B4 of panel table

Panel Configuration Bits

AUX[0Ch]					Resolution	Mode	Bits	Comments
Bit 4	Bit 3	Bit 2	Bit 1	Bit 0				
1	1	1	1	1	640x480	Dual	8	Monochrome
1	1	1	1	0	640x480	Single	8	Monochrome
1	1	1	0	1	640x200	Single	4	Monochrome
1	1	1	0	0	320x240	Single	4	Monochrome
1	1	0	1	1	640x480	Dual	8	Color STN / 2 shift clocks
1	1	0	1	0	640x480	Single	8	Color STN
1	1	0	0	1	640x480	Dual	16	Color STN
1	1	0	0	0	640x480	Single	16	Color STN
1	0	1	1	1	640x480	Dual	6	Color (InFocus)
1	0	1	1	0	640x480	TFT	18	TFT
1	0	1	0	1	640x480	TFT	12	TFT
1	0	1	0	0	640x480	TFT	9	TFT
1	0	0	1	1	640x480	TFT	9	MODULATED
1	0	0	1	0	Half Size	Single	8	Optrex panel only
1	0	0	0	1	640x480	Dual	8	Color STN / 1 shift clock
1	0	0	0	0	OEM Config			
0	1	1	1	1	800x600	Dual	16	Color STN
0	1	1	1	0	1024x768	TFT	18	TFT
0	1	1	0	1	480x320	Single	4	
0	1	1	0	0	Reserved (Configurable by OEM)			
through								
0	0	0	0	0				

All panels default to 78 Hz frame rate.

A.2 Refresh Select

Pin Name	Pin No.	Configuration Pin Functionality
MDA13	97	Refresh select bit 0
MDA14	95	Refresh select bit 1

MDA14 and MDA13 are Refresh Select. MDA14 is programmed into Aux[03]b1, MDA13 is programmed into Aux[03]b0 except when they are 11, if they are programmed 11, 00 is put in Aux[03]b1,b0 and b5 is set to a 1 to signify Self Refresh DRAM is present.

A.3 DRAM Speed Select

Pin Name	Pin No.	Configuration Pin Functionality
MDA15	93	Symmetrical/Asymmetrical DRAM

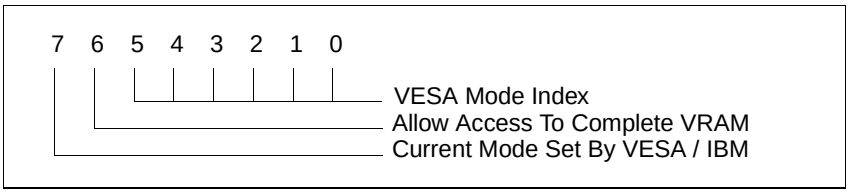
MDA15 is used to select Symmetrical or Asymmetrical DRAM. 1=Symetrical, 0=Assymetrical, this value is inverted and programmed into Aux[03]b7 and Aux[03]b4.

Appendix B BIOS USE OF INTERNAL SCRATCH BITS

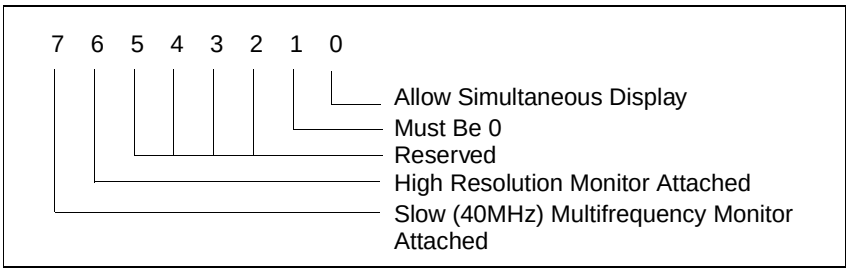
The BIOS uses scratch bits, which have no equivalent in a standard IBM compatible BIOS, to store information. Changing a bit state has no immediate effect on the video state as scratch bits are typically processed at a mode set.

Note
these scratch registers should not be programmed directly, using SOLLEX calls is recommended instead, as bit locations may change without notice

Auxiliary Register 4

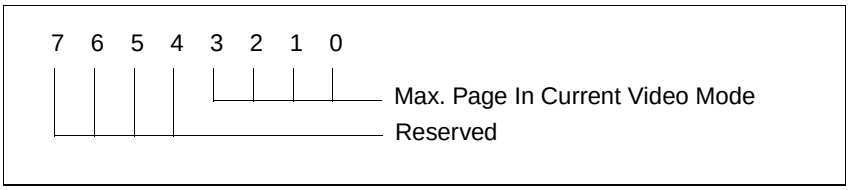


Auxiliary Register A



Where:		
Bit 7	**Bit 6**	**Attached Display Type**
0	0	Standard IBM Only
0	1	High Resolution Monitor > 800 x 600
1	0	Standard IBM Only
1	1	High Resolution Monitor But Slow ≤ 800 x 600 @ 40MHz

Auxiliary Register B



VESA Mode Index

This is an index into a table of possible VESA modes. If all bits are zero, no VESA mode is currently active. The remaining seven combinations are indexes into a table of supported VESA modes. This table has the following values: 00 = no VESA mode, 01 = first VESA mode as returned by function 0, 02 = second VESA mode as returned by function 0, 03 = third VESA mode, etc. as returned by function 0.

Appendix C INITIALIZATION

```
;--------------------------------------------------------------------------
;--------------------------------------------------------------------------
;--------------------------------------------------------------------------
;       InitVideo - The system bios will perform a far call to this
;       routine during the POST.  InitVideo is only invoked once per
;       IPL and is not used during normal VGA operation.  This
;       routine will program the VGA as per the switch settings
;       and will install all vectors and data areas.
;
;
;--------------------------------------------------------------------------
;--------------------------------------------------------------------------
;--------------------------------------------------------------------------

        G11     EQU     1100 0000b      ; Gain bits
                ORG     100h

InitVideo PROC  FAR
        PUSH    BP                      ; Added for V2.2
        MOV     AX,1201h                ; Setup function 12
        MOV     BL,32h                  ; Video control
        INT     10h                     ; Turn off active video system
        XOR     AX,AX                   ; Zero AX
        MOV     DS,AX                   ; Set DS to segment 0
        MOV     ES,AX                   ; Set ES to segment 0
        ASSUME  DS:INT_SEGMENT,ES:INT_SEGMENT

        ; Clear several SPC auxiliary registers and states.

        LEA     SI,InitTheWorldTbl
        CALL    BlowRegisters           ; Init everything

        ; We have just modified the Memory Clock.
        ; It would be unsafe to access memory for a few miliseconds,
        ; until the memory clock is stable. Checking for PLL lock
        ; won't be of any help, as it would signal locked state during
        ; the unstable period. So we simply waste some time at this
        ; point....

;;;     CALL    SPCWaitForPLLsLocked

        MOV     AX,200*10               ; Setup 200mS wait
        CALL    Delay100us              ; Waste time
```

```
        CALL    SetupDisplay            ; Crack MD lines...
        CALL    SetupVGA                ; Setup for VGA operation, mode set
        CALL    ShowID                  ; Identify BIOS on display
        CALL    InitExt

        POP     BP                      ; Added for V2.2
        XOR     AX,AX                   ; Set Z clr, C clr (Epson AT fix)
        RET
InitVideo ENDP
        PAGE


;-----------------------------------------------------------------------------
;
;
;-----------------------------------------------------------------------------

InitTheWorldTbl :
        DW      PW+03C3h,1

        DW      PW+03C2h,67h            ; Default card to 3Dx, 28Mhz Clock
        DW      03DEh                   ; Program the Aux. registers

        DB       27H,00000100b          ; PLL2 Denominator, 40MHz clock
        DB       26H,01001011b          ; PLL2 Numerator

        ; preserve the LCD bit if warm boot

;;      DB       00h,00000000b

        ; registers 1,2 determined via MD lines...

;;      DB       01H,01000010b          ; autocenter, 2Hz blink
;;      DB       02H,00000000b          ; LCD interface register

        ; some bits of register 3 determined via MD lines...

        DB       03H,00001000b          ; Memory Config Register (Wait state=Turbo)
        DB       04H,00000000b          ; clear scratch pad
        DB       05H,00000000b          ; disable memory paging, HIREZ256 etc.
```

```
        ; registers 6,7 determined at run time via MD lines...

;;      DB      06H,00000000b           ; LCD WF register
;;      DB      07H,00000000b           ; LCD Display register

;;      DB      08H,00000000b           ; RO
;;      DB      09H,00000000b           ; N/A
        DB      0AH,01000000b           ; initialize scratch pad as MFM 45MHz
        DB      0BH,00000000b           ; clear scratch pad

;;      DB      0CH,00000000b           ; treat as RO !
;;      DB      0DH,00000000b           ; RO
;;      DB      0EH,00000000b           ; N/A
;;      DB      0FH,00000000b           ; RO

        ; set default values for PD registers

        DB      10H,00000000b           ; polarity PS1 & PS4 low, no timer
        DB      11H,00000011b           ; PS 1: GreyScale, Slow Clock,H&V sync active
        DB      12H,00000011b           ; PS 2: GreyScale, Slow Clock,H&V sync active
        DB      13H,10110111b           ; PS 3: MClk Off,LCDBack off,
                                        ; DPMS Suspend,GS,Display off,
        DB      14H,10110111b           ; PS 4: MClk Off,LCDBack off,DPMS
                                        ; Suspend,GS,Display off,

        ; All skew values are based on MD lines!

        DB      15H,00001100b           ; PS Disable PS1&PS4, BUS I/F off PS4,skew XX
;;      DB      16H,01001111b           ; PS Skew Register
;;      DB      17H,00000001b           ; PS SkewA b3=1

;;      DB      18H,00000000b           ; N/A
;;      DB      19H,00000000b           ; N/A
        DB      1AH,00000000b           ; Video Buffer High
        DB      1BH,00100000b           ; Video Buffer Low

        ; 1C..1E are determined at run time via MD lines

;;      DB      1CH,00000000b           ; Frame Start (Dual LCD only)
;;      DB      1DH,00000000b           ; Frame Length High
;;      DB      1EH,00000000b           ; Frame Length Low

        DB      1FH,00000000b           ; Frame Buffer enable + High Request
```

```
        ; 20..29 are determined at run time via MD lines

        DB      21H,00000001b           ; PLL1 Set0 Denominator
        DB      20H,00000001b OR G11    ; PLL1 Set0 Numerator
        DB      23H,00000001b           ; PLL1 Set1 Denominator
        DB      22H,00000001b OR G11    ; PLL1 Set1 Numerator

;;      DB      24H,00000000b           ; PLL1 Set2 Numerator
;;      DB      25H,00000000b           ; PLL1 Set2 Denominator
;;      DB      27H,00000100b           ; PLL2 already done above
;;      DB      26H,00001011b           ; PLL2 already done above
        DB      28H,00000000b           ; clear any PLL Lock override

        ;; 2a..33 are N/A

        DB      34H,00000000b           ; disable GENIO outputs

        ; FIFOs determined via MD lines...

;;      DB      38H,00000000b           ; FIFO
;;      DB      39H,00000000b           ; FIFO
;;      DB      3AH,00000000b           ; FIFO
;;      DB      3BH,00000000b           ; FIFO
;;      DB      3CH,00000000b           ; FIFO

        DB      -1                      ; end of Aux registers
        DW      03CEh
        DB      06h,00000100b           ; set A000 addressing
        DB      -1
        DW      -1                      ; end of the register table

        PAGE


PLL0DenCRT      DB      04h,(07h OR G11),01h,(02h OR G11)
FOSC1000        EQU     14318
```

```
;-------------------------------------------------------------------------
;
; Pointers to panel info tables. Table selected by dip-switches.
;
;-------------------------------------------------------------------------
SetLCDTypeTbl:
        DW      Tbl640x480D8M      ; 0  0  0  0  0 640x480 (Dual,8)
        DW      Tbl640x480S8M      ; 0  0  0  0  1 640x480 (Single,8)
        DW      Tbl640x200S4M      ; 0  0  0  1  0 640x200 (Single,4)
        DW      Tbl320x240S4M      ; 0  0  0  1  1 320x240 (Single,4)
        DW      Tbl640x480D8C2SC   ; 0  0  1  0  0 640x480 (Dual,8,2shft clk)
        DW      Tbl640x480S8C      ; 0  0  1  0  1 640x480 (Single,8)
        DW      Tbl640x480D16C     ; 0  0  1  1  0 640x480 (Dual,16)
        DW      Tbl640x480S16C     ; 0  0  1  1  1 640x480 (Single,16)
        DW      Tbl640x480D6InF    ; 0  1  0  0  0 640x480 (Dual,6,InFocus)
        DW      Tbl640x480TFT18    ; 0  1  0  0  1 640x480 (TFT,18)
        DW      Tbl640x480TFT12    ; 0  1  0  1  0 640x480 (TFT,12)
        DW      Tbl640x480TFT09    ; 0  1  0  1  1 640x480 (TFT,09)
        DW      Tbl640x480TFT09Mod ; 0  1  1  0  0 640x480 (TFT,09,Modulated)
        DW      Tbl320x240S8OPT    ; 0  1  1  0  1 320x240 (Single,8,Optrex)
        DW      Tbl640x480D8C      ; 0  1  1  0  0 640x480 (Dual,8,Colour,1shft clk)
        DW      TblOEM             ; 0  1  1  1  1 OEM_1
        DW      Tbl800x600D16C     ; 1  0  0  0  0 800x600 (Dual,16)
        DW      Tbl1024x768TFT     ; 1  0  0  0  1 1024x768(TFT,18)
        DW      Tbl480x320S4M      ; 1  0  0  1  0 480x320 (Single,4)
        DW      TblReserved3       ; 1  0  0  1  1 320x240 (Single,4)
        DW      TblReserved4       ; 1  0  1  0  0 640x480 (Dual,8,2shft clk)
        DW      TblReserved5       ; 1  0  1  0  1 640x480 (Single,8)
        DW      TblReserved6       ; 1  0  1  1  0 640x480 (Dual,16)
        DW      TblReserved7       ; 1  0  1  1  1 640x480 (Single,16)
        DW      TblReserved8       ; 1  1  0  0  0 640x480 (Dual,6,InFocus)
        DW      TblReserved9       ; 1  1  0  0  1 640x480 (TFT,18)
        DW      TblReservedA       ; 1  1  0  1  0 640x480 (TFT,12)
        DW      TblReservedB       ; 1  1  0  1  1 640x480 (TFT,09)
        DW      TblReservedC       ; 1  1  1  0  0 640x480 (TFT,09,Modulated)
        DW      TblReservedD       ; 1  1  1  0  1 320x240 (Single,8,Optrex)
        DW      TblReservedE       ; 1  1  1  0  0 640x480 (Dual,8,Colour,1shft clk)
        DW      TblReservedF       ; 1  1  1  1  1 OEM_1
```

```
;-------------------------------------------------------------------------
;
; Panel info tables. Based o DIP switches we assume a certain panel attached.
; All relevant registers for the panel are within one PST structure.
; These values are configurable using an external configuration utility.
;
;-------------------------------------------------------------------------

PST     STRUCT
        AUX_1           DB      ?
        AUX_2           DB      ?
        AUX_6           DB      ?
        AUX_7           DB      ?
        AUX_16          DB      ?
        AUX_17          DB      ?
        AUX_15          DB      ?
        CRTCB_0         DB      ?
        CRTCB_1         DB      ?
        CRTCB_6         DB      ?
        CRTCB_7         DB      ?
        CRTCB_10        DB      ?
        CRTCB_11        DB      ?
        CRTCB_12        DB      ?
        PLL0Den         DB      ?
        PLL0Num         DB      ?
        PLL1Den         DB      ?
        PLL1Num         DB      ?
PST     ENDS

aAUXIndeces     DB      1,2,6,7,16h,17h,15h
aCRTCBIndeces   DB      0,1,6,7,10h,11h,12h

;                        A1  A2        A6  A7  A16  A17 A15 C0  C1    C6   C7  C10
C11 C12  Den0    Num0   Den1   Num1
Tbl640x480D8M      PST
<42h,01001000b,20h,00h,04Fh,01h,0Ch,1Eh,640/8,0F1h,00h,0EBh,0Dh,480/4,01h,01h OR
G11,03h,02h OR G11>
Tbl640x480S8M      PST
<42h,01001001b,20h,00h,04Fh,01h,0Ch,1Eh,640/8,0E1h,01h,0EBh,0Dh,480/4,01h,02h OR
G11,03h,04h OR G11>
Tbl640x200S4M      PST
<42h,01000001b,20h,00h,04Fh,01h,0Ch,1Eh,640/8,0C9h,00h,0EBh,0Dh,200/4,0Ch,0Ah OR
G11,0Ch,05h OR G11>
Tbl320x240S4M      PST
<42h,01000001b,20h,00h,04Fh,01h,0Ch,1Eh,320/8,0F1h,00h,0EBh,0Dh,240/4,01h,01h OR
G11,02h,01h OR G11>
Tbl640x480D8C2SC   PST
<42h,01001010b,20h,00h,04Fh,01h,0Ch,1Eh,640/8,0F1h,00h,0EBh,0Dh,480/4,01h,01h OR
G11,03h,02h OR G11>
Tbl640x480S8C      PST
<42h,01001011b,20h,00h,04Fh,01h,0Ch,1Eh,640/8,0E1h,01h,0EBh,0Dh,480/4,01h,02h OR
```

```
G11,03h,04h OR G11>
Tbl640x480D16C     PST
<42h,01001110b,20h,00h,04Fh,01h,0Ch,1Eh,640/8,0F1h,00h,0EBh,0Dh,480/4,01h,01h OR
G11,03h,02h OR G11>
Tbl640x480S16C     PST
<42h,01001111b,20h,00h,04Fh,01h,0Ch,1Eh,640/8,0E1h,01h,0EBh,0Dh,480/4,01h,02h OR
G11,03h,04h OR G11>
Tbl640x480D6InF    PST
<42h,01000110b,20h,00h,04Fh,01h,0Ch,1Eh,640/8,0F1h,00h,0EBh,0Dh,480/4,01h,01h OR
G11,03h,02h OR G11>
Tbl640x480TFT18    PST
<42h,01011111b,00h,00h,000h,00h,00h,1Eh,640/8,00Ch,24h,0EBh,0Dh,480/4,04h,07h OR
G11,01h,02h OR G11>
Tbl640x480TFT12    PST
<42h,01011011b,00h,00h,000h,00h,00h,1Eh,640/8,00Ch,24h,0EBh,0Dh,480/4,04h,07h OR
G11,01h,02h OR G11>
Tbl640x480TFT09    PST
<42h,01010111b,00h,00h,000h,00h,00h,1Eh,640/8,00Ch,24h,0EBh,0Dh,480/4,04h,07h OR
G11,01h,02h OR G11>
Tbl640x480TFT09Mod PST
<42h,01010111b,20h,00h,000h,00h,00h,1Eh,640/8,00Ch,24h,0EBh,0Dh,480/4,04h,07h OR
G11,01h,02h OR G11>
Tbl320x240S8OPT    PST
<42h,01001111b,20h,20h,04Fh,01h,0Ch,1Eh,320/8,0F1h,00h,0EBh,0Dh,240/4,01h,02h OR
G11,03h,04h OR G11>
Tbl640x480D8C      PST
<42h,01000010b,20h,00h,04Fh,01h,0Ch,1Eh,640/8,0F1h,00h,0EBh,0Dh,480/4,01h,01h OR
G11,03h,02h OR G11>
TblOEM             PST
<42h,01001000b,20h,00h,04Fh,01h,0Ch,1Eh,640/8,0F1h,00h,0EBh,0Dh,480/4,01h,01h OR
G11,03h,02h OR G11>
Tbl800x600D16C     PST
<42h,01001110b,20h,00h,04Fh,01h,0Ch,1Eh,800/8,02Dh,01h,0EBh,0Dh,600/4,04h,07h OR
G11,01h,02h OR G11>
Tbl1024x768TFT     PST
<42h,01011111b,00h,00h,000h,00h,00h,1Eh,1024/8,25h,0A5h,03h,09h,768/4,04h,07h OR
G11,01h,02h OR G11>
Tbl480x320S4M      PST
<42h,01000001b,20h,00h,04Fh,01h,0Ch,1Eh,480/8,041h,01h,0EBh,0Dh,320/4,03h,04h OR
G11,06h,04h OR G11>
TblReserved3       PST
<42h,01000001b,20h,00h,04Fh,01h,0Ch,1Eh,320/8,0F1h,00h,0EBh,0Dh,240/4,01h,02h OR
G11,03h,04h OR G11>
TblReserved4       PST
<42h,01001010b,20h,00h,04Fh,01h,0Ch,1Eh,640/8,0F1h,00h,0EBh,0Dh,480/4,01h,01h OR
G11,03h,02h OR G11>
TblReserved5       PST
<42h,01001011b,20h,00h,04Fh,01h,0Ch,1Eh,640/8,0E1h,01h,0EBh,0Dh,480/4,01h,02h OR
G11,03h,04h OR G11>
TblReserved6       PST
<42h,01001110b,20h,00h,04Fh,01h,0Ch,1Eh,640/8,0F1h,00h,0EBh,0Dh,480/4,01h,01h OR
G11,03h,02h OR G11>
TblReserved7       PST
<42h,01001111b,20h,00h,04Fh,01h,0Ch,1Eh,640/8,0E1h,01h,0EBh,0Dh,480/4,01h,02h OR
```

```
G11,03h,04h OR G11>
TblReserved8       PST
<42h,01000110b,20h,00h,04Fh,01h,0Ch,1Eh,640/8,0F1h,00h,0EBh,0Dh,480/4,01h,01h OR
G11,03h,02h OR G11>
TblReserved9       PST
<42h,01011111b,00h,00h,000h,00h,00h,1Eh,640/8,00Ch,24h,0EBh,0Dh,480/4,04h,07h OR
G11,01h,02h OR G11>
TblReservedA       PST
<42h,01011011b,00h,00h,000h,00h,00h,1Eh,640/8,00Ch,24h,0EBh,0Dh,480/4,04h,07h OR
G11,01h,02h OR G11>
TblReservedB       PST
<42h,01010111b,00h,00h,000h,00h,00h,1Eh,640/8,00Ch,24h,0EBh,0Dh,480/4,04h,07h OR
G11,01h,02h OR G11>
TblReservedC       PST
<42h,01010111b,20h,00h,000h,00h,00h,1Eh,640/8,00Ch,24h,0EBh,0Dh,480/4,04h,07h OR
G11,01h,02h OR G11>
TblReservedD       PST
<42h,01001111b,20h,20h,04Fh,01h,0Ch,1Eh,320/8,0F1h,00h,0EBh,0Dh,240/4,01h,02h OR
G11,03h,04h OR G11>
TblReservedE       PST
<42h,01000010b,20h,00h,04Fh,01h,0Ch,1Eh,640/8,0F1h,00h,0EBh,0Dh,480/4,01h,01h OR
G11,03h,02h OR G11>
TblReservedF       PST
<42h,01001000b,20h,00h,04Fh,01h,0Ch,1Eh,640/8,0F1h,00h,0EBh,0Dh,480/4,01h,01h OR
G11,03h,02h OR G11>

;--------------------------------------------------------------------------------
;       SetupDisplay - Setup CRT/LCD display operation.
;
;       Entry:  n/a
;       Exit:   n/a
;       Uses:   AX,BX,CX,DX,SI,DI
;--------------------------------------------------------------------------------

;--------------------------------------------------------------------------------
;               MD15 MD14 MD13    AUX[3]
;               -------------------------
;               0    0    0     10010000b
;               0    0    1     10010001b
;               0    1    0     10010010b
;               0    1    1     10110000b
;               1    0    0     00000000b
;               1    0    1     00000001b
;               1    1    0     00000010b
;               1    1    1     00100000b
;--------------------------------------------------------------------------------

MemCfgTbl      DB  10010000b,10010001b,10010010b,10110000b
               DB  00000000b,00000001b,00000010b,00100000b

SetupDisplay:
```

```
        ; to prevent "DC lines" on LCD panel after warm boot, we have to
        ; turn off the panel using a regular pd sequence...

        GETAUX  00h
        AND     AL,20h                  ; clear everything but the LCD bit
        PUTAUX  00h,AL
        CALL    SPCPanelOff             ; this will clear LCD on bit in AUX[0]

        GETAUX  0Ch                     ; read the MD lines
        MOV     BX,AX                   ; save MD lines in BX

        ; crack MD lines and program AUX[3]

        GETAUX  03h
        AND     AL,00001100b            ; preserve turbo bits
        SHR     BX,5                    ; position for indexing
        AND     BX,00000111b
        ADD     BX,OFFSET MemCfgTbl
        OR      AL,CS:[BX]              ; mask in the rest of the memory bits
        PUTAUX  03h,AL
        CALL    Is1MegRamPresent        ; Q : 1 MegDRAM?
        JE      SHORT @F                ;   Y: all done
        OR      AL,01000000b            ;   N: set Single DRAM bit
        PUTAUX  03h,AL
@@:
        ; now program the panel parameters

        CALL    SPCLoadPanelRegs

        ; Try to enable CRT, if fails enable LCD

        CALL    SPCCRTOn
        JNC     SHORT SetupDisplayX
        CALL    SPCPanelOn

SetupDisplayX:

        ; program default sprite palette

        CALL    SPCDoSpriteDAC
        RET
        PAGE
```

```
;--------------------------------------------------------------------------
;
; Detect if 1M or 512k of video memory is present.
; The algorithm is simple: write a pattern and read it back.
; For 512k systems we will read an error, as the AUX[3] register is NOT
; configured for single VRAM chip (yet). Also, to defeat any caching
; we perform more than one read/write.
;
; Note: any memory error will be interpreted as 512k of memory!
;
;--------------------------------------------------------------------------
Is1MegRamPresent:
        PUSH    AX
        PUSH    CX
        PUSH    DX
        PUSH    ES
        MOV     AX,0A000h
        MOV     ES,AX
        MOV     DX,3CDh

        PUTAUX  05h,11h                 ; bypass graphic controller, allow banking

        XOR     AX,AX
        OUT     DX,AL                   ; select page 0 for r/w
        MOV     DI,2
        MOV     CX,1000h
@@:
        STOSW   ES:[DI]                 ; fill page 0 with pattern
        ADD     DI,2
        INC     AX
        LOOP    @B

        MOV     DI,2
        XOR     AX,AX
        MOV     CX,1000h
@@:
        CMP     AX,ES:[DI]
        JNZ     Is1MegRamPresentX       ; 'Z' cleared
        ADD     DI,4
        INC     AX
        LOOP    @B
        XOR     AX,AX                   ; force 'Z' flag set
```

```
Is1MegRamPresentX:
        PUTAUX  05h,00h                 ; disable gr. controller bypass
        POP     ES
        POP     DX
        POP     CX
        POP     AX
        RET


;-------------------------------------------------------------------------
;       SPCLoadPanelRegs - Programs all registers for a given panel type.
;
;       Uses: Nothing
;       Note: Called only once, at initialization.
;             Panel is OFF at the time of this call.
;
;-------------------------------------------------------------------------
SPCLoadPanelRegs:
        PUSHA
        CALL    GetPanelType            ; CS:SI table proper

        ; for dual panels program the half frame buffer

        TEST    CS:[SI.PST.AUX_2],1     ; Q: is panel dual?
        JNZ     SHORT SPCFrameBufDone   ;    N: split

        ; Slow dot clock modes on panels smaller than 640 pixels may
        ; not show up correctly. Panels smaller than 640 pixels will
        ; be programmed to 640 pixels width. (There's a bug in 8110
        ; if Width(B-set) < Width(A-set). Width(A-set) is always = 640 for
        ; slow dot clock modes)

        MOV     AL,640/8
        MOV     AH,CS:[SI.PST.CRTCB_1]
        CMP     AH,AL
        JA      SHORT @F
        MOV     AH,AL
@@:
        MOV     AL,CS:[SI.PST.CRTCB_12] ; AL = Height/4

        MUL     AH                      ; AX = (Height/4)*(Width/8)=(H*W)/32
        SHR     AX,1                    ; AX = (HxW)/64

        ; do mono/colour adjustments

        TEST    CS:[SI.PST.AUX_2],2     ; Q: colour panel
        JZ      SHORT SPCMono           ;    N: do mono adjustments
        MOV     BX,AX
```

```
        ADD     AX,AX
        ADD     AX,BX                   ; AX = 3*(H*W)/64
        SUB     AX,7                    ; AX = 3*(H*W)/64-7
        JMP     SHORT SPCDoFrameLength
SPCMono:
        SUB     AX,3                    ; AX = (H*W)/64-3
SPCDoFrameLength:
        MOV     BX,AX
        PUTAUX  1Dh,BH
        PUTAUX  1Eh,BL
        MOV     AL,0Fh                  ; assume page 15 and 1Mb of memory
        CALL    SPCIs1Meg
        JE      SHORT @F
        SHR     AX,1                    ; make it page 7 for 512k of memory
@@:
        ; adjust the frame buffer position based on frame
        ; buffer length bits b15,b14

        TEST    BH,40h
        JZ      SHORT @F
        AND     AL,11111110b
@@:
        TEST    BH,80h
        JZ      SHORT @F
        AND     AL,11111100b
@@:
        PUTAUX  1Ch,AL                  ; frame start at the very end of mem.

SPCFrameBufDone:
        MOV     CX,6                    ; do 6 AUX registers
        LEA     DI,aAUXIndeces
        CLD
DoAUXRegs:
        LODS    CS:CS_BYTE              ; get value for AUX
        PUTAUX  CS:[DI],AL              ; program AUX reg
        INC     DI
        LOOP    DoAUXRegs

        ; special case AUX[15h] : modify only the skew bits

        GETAUX  15h
        AND     AL,11110011b
        MOV     AH,AL
        LODS    CS:CS_BYTE
        OR      AL,AH
        PUTAUX  15h,AL
```

```
        GETAUX  00h
        PUSH    AX                      ; save AUX[0]
        OR      AL,1
        PUTAUX  00h,AL                  ; enable CRTC_B set
        MOV     CX,7                    ; do 7 CRTC_B registers
        LEA     BX,aCRTCBIndeces
        CALL    LoadCRTCTable

        ; If a panel x-size is less then 640 pixels,
        ; program the CRTCB[1] with 640 pixels. This is to
        ; take care of a bug if panel size <= HDE. The smallest
        ; HDE is 40 for any IBM mode.

        MOV     AL,01h
        CALL    SPCReadCRT
        CMP     AL,640/8
        JAE     SHORT @F
        MOV     AX,0150h                ; write CRTCB[1] = 50h (80 decimal)
        CALL    SPCWriteCRT
@@:
        POP     AX                      ; get saved AUX[0]
        PUTAUX  00h,AL                  ; restore it

        POPA
        RET


;-------------------------------------------------------------------------
;       SPCLoadSingleScanRegs - Programs CRTCB[6,7] with panel values.
;
;       NOTE: for dual panels, we have to enable frame buffer!
;
;       Uses: AX,SI,!DX
;-------------------------------------------------------------------------
SPCLoadSingleScanRegs:
        PUSH    DX
        CALL    GetPanelType            ; CS:SI table proper

        GETAUX  00h
        MOV     DX,AX                   ; save current AUX[0] value
        OR      AL,1

        CLI
        PUTAUX  00h,AL                  ; enable CRTC_B set
        MOV     AL,CS:[SI.PST.CRTCB_6]
        MOV     AH,06h
        CALL    SPCWriteCRT             ; CRTCB[6] = AL
        MOV     AL,CS:[SI.PST.CRTCB_7]
```

```
        MOV     AH,07h
        CALL    SPCWriteCRT             ; CRTCB[7] = AL
        PUTAUX  00h,DL                  ; restore AUX[0]
        STI

        ; for dual panels, we have to enable frame buffer!

        GETAUX  02h
        TEST    AL,1b                   ; single panel?
        JNZ     SHORT @F                ; yes, skip enabling of frame buffer
        GETAUX  1Fh                     ; al = AUX[1Fh]
        AND     AL,7Fh                  ; enable frame buffer
        PUTAUX  1Fh,AL                  ; clear the disable flag
@@:
        POP     DX
        RET

;-------------------------------------------------------------------------------
;       SPCLoadSimultaneousDisplayRegs - Programs CRTCB[6,7] with simultaneous
        display values.
;
;       NOTE: for dual panels, we have to disable frame buffer!
;
;       Uses: AX,BX,!DX
;-------------------------------------------------------------------------------
SPCLoadSimultaneousDisplayRegs:
        PUSH    DX

        GETAUX  00h
        MOV     DX,AX
        OR      AL,1

        ; program CRTCB[6,7] with simultaneous display values

        CLI
        PUTAUX  00h,AL                  ; enable CRTC B-set
        MOV     AX,060Ch
        CALL    SPCWriteCRT             ; CRTCB[6] = 0Ch
        MOV     AX,0724h
        CALL    SPCWriteCRT             ; CRTCB[7] = 24h
        PUTAUX  00h,DL
        STI
```

```
        ; for dual panels, we have to disable frame buffer!

        GETAUX  02h
        TEST    AL,1b                   ; single panel?
        JNZ     SHORT @F                ; yes, skip disabling of frame buffer
        GETAUX  1Fh                     ; al = AUX[1Fh]
        OR      AL,80h                  ; disable frame buffer
        PUTAUX  1Fh,AL                  ; clear the disable flag
@@:
        POP     DX
        RET

;-------------------------------------------------------------------------
;       SPCCRTOn - Activates CRT circuitry
;
;       Output: CY = 1 if failed (no monitor)
;
;-------------------------------------------------------------------------
SPCCRTOn:
        GETAUX  08h                     ; see if any monitor attached
        AND     AL,0111b                ; keep Monito ID bits only
        CMP     AL,0111b                ; any Monitor ID?
        JE      SHORT SPCCRTOnFail      ; ...no, return failure

        GETAUX  00h
        TEST    AL,40h                  ; already on....
        JNZ     SHORT SPCCRTOnX

        OR      AL,40h                  ; select CRT
        PUTAUX  00h,AL

        ; program the PLLs and FIFO registers....

        CALL    SPCDoDisplay
SPCCRTOnX:
        CLC
        RET

SPCCRTOnFail:
        STC
        RET
        PAGE
```

```
;-------------------------------------------------------------------------------
;       SPCCRTOff - De-Activates CRT circuitry
;
;-------------------------------------------------------------------------------

SPCCRTOff:
        GETAUX  00h
        AND     AL,NOT 40h              ; turn off CRT
        PUTAUX  00h,AL
        TEST    AL,20h                  ; is LCD active ?
        JZ      SHORT @F                ; nop, all done

        ; we might have been in simultaneous display, running LCD with CRT values.
        ; Now it is a good time to get proper LCD values.
        ; Program the PLLs and FIFO registers for panels

        CALL    SPCPanelOn
@@:
        RET
        PAGE


;-------------------------------------------------------------------------------
;       SPCCPanelOn - Activates panel circuitry
;
;       Uses: AX,BX,!SI
;
;       NOTE: If CRT is active, we do not touch PLLs and FIFOs.
;             We always program half frame buffer for dual panels.
;             If CRT is NOT active, we clear the frame buffer disable
;             bit in AUX[1Fh]
;-------------------------------------------------------------------------------
SPCPanelOn:
        PUSH    SI

        GETAUX  00h
        OR      AL,20h                          ; select panel
        PUTAUX  00h,AL
        TEST    AL,40h                          ; is CRT active ?
        JZ      SHORT SPCPanelNoCRT             ; ...no, do some maintenance
        CALL    SPCLoadSimultaneousDisplayRegs
        JMP     SHORT SPCPanelOnX               ; keep the CRT registers

SPCPanelNoCRT:
```

```
        ; program the PLLs and FIFO registers....

        CALL    SPCDoDisplay
        CALL    SPCLoadSingleScanRegs   ; program CRTCB[6,7] with panel values

SPCPanelOnX:
        POP     SI
        RET
        PAGE

;-------------------------------------------------------------------------
;       SPCCPanelOff - De-Activates panel circuitry
;
;       Uses: AX
;
;       NOTE: Simple turning off LCD bit in AUX[0] won't do!
;
;-------------------------------------------------------------------------
SPCPanelOff:
        CALL    SPCIsLCDActive
        JNE     SHORT SPCPanelOffX
        PUSH    BX
        PUSH    DX
        CALL    SPCPdDisplayOff         ; DL = AUX[0]
        AND     DL,NOT 20h              ; de-select panel
        CALL    SPCPdDisplayOn
        POP     DX
        POP     BX
SPCPanelOffX:
        RET
        PAGE

;-------------------------------------------------------------------------
;
;
;
;-------------------------------------------------------------------------
SPCPdDisplayOff:
        GETAUX  12h                     ; software PD reg. 2
        MOV     BL,AL                   ; save AUX[12h] in BL
        MOV     AL,04h                  ; force Display Off
        PUTAUX  12h,AL

        GETAUX  00h
        MOV     DL,AL
        AND     AL,NOT 40h
        PUTAUX  00h,AL                  ; turn off CRT so PD can kick in
        GETAUX  15h
        MOV     BH,AL                   ; save AUX[15h] in BH
```

```
        OR      AL,20h                  ; set PD MODE 2, now Display is Off
        PUTAUX  15h,AL
        MOV     AX,500*10               ; Setup 500mS wait
        CALL    Delay100us              ; Waste time for PD to kick in
        RET

;-------------------------------------------------------------------------
;
;
;
;-------------------------------------------------------------------------
SPCPdDisplayOn:
        PUTAUX  00h,DL
        PUTAUX  15h,BH                  ; restore PD engage bit
        PUTAUX  12h,BL                  ; restore PD reg. 2
        RET

;-------------------------------------------------------------------------
;       SPCDoDisplay - Crack the current environment and program
;               the FIFO and PLL registers accordingly.
;               Called from SPCPanelOn,SPCCRTOn. See the SPC8110
;               technical reference manual for FIFO formulae.
;
;
;       Uses: AX,!BX,!SI
;-------------------------------------------------------------------------

                ; SlowTxt SlowGfx Text8 Text9 Planar TextE HR256

PxPerLevel      DW    16,   16,     8,    9,     8,    8,    4
C512            DW  1545, 1545,  1545, 1545,  1545, 1545, 1545
C1024           DW  1545,  745,  1545, 1545,   745, 1545,  745

K512CRT         DW  2950, 1950,  2950, 2950,  1950, 2950, 1950
K512DualMono    DW  3350, 2350,  3350, 3350,  2350, 3350, 2350
K512DualColor   DW  4950, 3950,  4950, 4950,  3950, 4950, 3950
K512SIZE EQU $-K512CRT
K1024CRT        DW  2750, 1550,  2750, 2750,  1550, 2750, 1550
K1024DualMono   DW  2950, 1750,  2950, 2950,  1750, 2950, 1750
K1024DualColor  DW  3750, 2550,  3750, 3750,  2550, 3750, 2550

SPCDoDisplay:
        PUSH    DI
        PUSH    SI
        PUSH    BX
        PUSH    CX
```

```
        ; 1. program the PLL set 0 & 1

        LEA     SI,PLL0DenCRT           ; assume CRT Values
        CALL    SPCIsCRTActive          ; check the assumption
        JE      SHORT @F                ; yes, CRT values OK
        CALL    GetPanelType            ; no, get panel specific clock values
        LEA     SI,[SI.PST.PLL0Den]
@@:
        LODS    CS:CS_BYTE
        MOV     CH,AL
        PUTAUX  21h,AL
        LODS    CS:CS_BYTE
        MOV     CL,AL
        PUTAUX  20h,AL
        LODS    CS:CS_BYTE
        PUTAUX  23h,AL
        LODS    CS:CS_BYTE
        PUTAUX  22h,AL

        ; 2. program the FIFOs (AUX[38h,39h,3Ah,3Bh])

        LEA     DI,K512CRT
        CALL    SPCGetCurrentDisplay
        CMP     BX,4                    ; CRT,TFT or single panel
        JBE     SHORT @F
        LEA     DI,K512DualMono
        CMP     BX,6
        JE      SHORT @F
        LEA     DI,K512DualColor
@@:
        LEA     SI,C512                 ; assume 512k
        CALL    SPCIs1Meg
        JNE     SHORT @F
        LEA     SI,C1024
        ADD     DI,K512SIZE
@@:
        ; VGA slow text

        XOR     BX,BX
        CALL    CalcFIFOHiLow
        PUTAUX  3Bh,AL

        ; VGA slow graphics

        MOV     BX,1*2
        CALL    CalcFIFOHiLow
        PUTAUX  3Ah,AL
```

```
        MOV     BX,2*2
        CALL    CalcFIFOHiLow
        PUTAUX  39h,AL

        MOV     BX,4*2
        CALL    CalcFIFOHiLow
        PUTAUX  38h,AL

        ; If we are currently in VESA mode, we have to re-program
        ; also AUX[3ch] and PLL1 set 2. (External clock).

        CALL    SPCValidateBiosMode
        JNC     SHORT NotExternalClock  ; external clock not used

        EXTERN  FindCurrentResTable:NEAR
        CALL    FindCurrentResTable
        PUSH    DS
        PUSH    CS
        POP     DS

        MOV     DI,SI
;;;     MOV     AL,CS:[DI].wSMOSClockNumDen
        EXTERN  SetPLL:NEAR
        CALL    SetPLL                  ; set the external clock & Fifos
        POP     DS

NotExternalClock:

        POP     CX
        POP     BX
        POP     SI
        POP     DI
        RET


       ; AH = den,AL = num+GAIN bits
       ; BX = Current display (0,2,4,6,8)
       ; CX = 8 (VGA Planar)
       ;    = 10 (Enh text)
       ;    = 12 (Hirez256)

SPCSetExtClock:
        PUSH    DI
        PUSH    SI
        PUSH    BX
        PUSH    CX
        PUSH    DX
```

```
        ; 1. program the PLL set 2

        MOV     DX,AX                   ; save num+den in DX
        PUTAUX  25h,DH
        PUTAUX  24h,DL

        ; 2. program the FIFO (AUX[3C])

        LEA     DI,K512CRT
        CMP     BX,4                    ; CRT,TFT or single panel
        JBE     SHORT @F
        LEA     DI,K512DualMono
        CMP     BX,6
        JE      SHORT @F
        LEA     DI,K512DualColor
@@:
        LEA     SI,C512                 ; assume 512k
        CALL    SPCIs1Meg
        JNE     SHORT @F
        LEA     SI,C1024
        ADD     DI,K512SIZE
@@:
        MOV     BX,CX
        MOV     CX,DX
        CALL    CalcFIFOHiLow
        PUTAUX  3Ch,AL

        POP     DX
        POP     CX
        POP     BX
        POP     SI
        POP     DI
        RET
```

```
;-------------------------------------------------------------------------------
;       CalcFIFOHiLow - Crack the current environment and program
;               the FIFO and PLL registers accordingly.
;               Called from SPCPanelOn,SPCCRTOn.
;
;       Entry: SI : "C" table
;              DI : "K" table
;              BX : index into above tables
;              CH : PCLK denomunator
;              CL : PCLK numerator
;       Exit:  AL : calculated FIFO value using the above
;
;       Uses:  AX,BX,DX,!CX,!SI,!DI
;-------------------------------------------------------------------------------
CalcFIFOHiLow:
        PUSH    CX
        AND     CL,00001111b            ;keep numerator bits only
        MOV     EDX,FOSC1000
        MOVZX   EAX,CL
        MUL     EDX                     ;eax = num*FOSC1000
        PUSH    EAX

        MOV     EDX,1000000
        MOVZX   EAX,CH
        MUL     EDX                     ;eax = den*1000000
        POP     ECX
        DIV     ECX                     ;eax = (den*1000000)/(num*FISC1000)
                                        ;eax = Tp
        ADD     BX,OFFSET PxPerLevel
        MOV     CX,CS:[BX]
        SUB     BX,OFFSET PxPerLevel
        MOVZX   ECX,CX
        MUL     ECX                     ;eax = Tp*PPL
        PUSH    EAX
```

```
  MOV     ECX,FOSC1000
  GETAUX  26h                     ;mclk num
  AND     AL,00001111b            ;keep 4 bits only
  MOVZX   EAX,AL
  MUL     ECX                     ;eax = num*FOSC1000
  MOV     ECX,EAX

  MOV     EDX,1000000
  GETAUX  27h                     ;mclk den
  MOVZX   EAX,AL
  MUL     EDX                     ; eax = den*1000000
  DIV     ECX                     ; eax = (den*1000000)/(num*FISC1000)
                                  ; eax = Tm
  POP     ECX                     ; ecx = Tp*PPL
  PUSH    ECX                     ; leave it on the stack
  PUSH    EAX                     ; eax = Tm

  MOV     DX,CS:[SI+BX]           ; dx = C100
  MOVZX   EDX,DX
  MUL     EDX                     ; eax = C100*Tm
  DIV     ECX                     ; eax = (C100*Tm)/(Tp*PPL)

  XOR     EDX,EDX
  MOV     ECX,100
DIV    ECX                  ; eax = (c*Tm)/(Tp*PPL)
  ADD     AX,4                    ; AL Low request reg

  MOV     DX,CS:[DI+BX]           ; dx = K100
  MOVZX   EDX,DX
  MOV     BX,AX                   ; BX = Low request reg value

  POP     EAX                     ; eax = Tm
  MUL     EDX                     ; eax = Tm*K100
  ADD     EAX,4330
  ADC     EDX,0
  POP     ECX                     ; ecx = Tp*PPL
  DIV     ECX                     ; eax = (Tm*K100+43.3*100)/(Tp*PPL)
  ADD     EAX,99                  ; round up
  XOR     EDX,EDX
  MOV     ECX,100
  DIV     ECX
  SUB     AX,1                    ; AL = High request reg value
  SHL     AL,4
  AND     BL,0Fh
  OR      AL,BL
  POP     CX
  RET
```

```
;-----------------------------------------------------------------------------
;       SPCGetCurrentDisplay - Crack the current environment
;
;       EXIT:  BX = 0 if CRT
;              BX = 2 if TFT
;              BX = 4 if Single LCD
;              BX = 6 if Dual LCD(mono)
;              BX = 8 if Dual LCD(color)
;-----------------------------------------------------------------------------
SPCGetCurrentDisplay:
        CALL    SPCIsCRTActive
        JNE     SHORT SPC_NotCRT
        XOR     BX,BX
        JMP     SHORT SPC_OK

SPC_NotCRT:

        ; check if TFT panel attached...

        GETAUX  02h                     ; AL = AUX[02h]
        TEST    AL,10h
        JZ      SHORT SPC_NotTFT
        MOV     BX,02
        JMP     SHORT SPC_OK

SPC_NotTFT:

        ; check if single LCD panel attached...

;;;     GETAUX  02h                     ; AL = AUX[02h]
        TEST    AL,1h
        JZ      SHORT SPC_NotSingle
        MOV     BX,04
        JMP     SHORT SPC_OK

SPC_NotSingle:

        ; not CRT,TFT,Single panel, well it must be dual panel

        MOV     BX,06
;;;     GETAUX  02h                     ; done above
        TEST    AL,2h                   ; color?
        JZ      SHORT SPC_OK            ; ...no
        MOV     BX,08                   ; ...yes
;;;     JMP     SHORT SPCDo_OK

SPC_OK:
        RET
```

```
;-------------------------------------------------------------------------
;       SPCDoSpriteDAC:
;             Programs sprite palette registers.
;
;       Entry:  n/a
;       Uses:   AX,CX,DX,SI,ES
;-------------------------------------------------------------------------

; Default grey scaled CRT sprite RamDac values. These correspond to
; Default LCD grey values of 0,5,10,15.

                DW      8000h,4 ; GS DAC (GS)
DefSpriteDAC    DB      0,5 SHL 2,10 SHL 2,03Fh

SPCDoSpriteDAC:
        GETAUX  00h
        MOV     BX,AX                   ; save AUX_0 in BL
        OR      AL,80h
        PUTAUX  00h,AL                  ; enable sprite palette load
        PUSH    CS
        POP     ES
        LEA     SI,DefSpriteDAC
        CALL    ProgDACRegs
        PUTAUX  00h,BL                  ; restore AUX_0
        RET

;-------------------------------------------------------------------------
;       SPCIsCRTActive - Test if CRT display active.
;
;       Entry:  n/a
;       Exit:   Z flg - Set if CRT active else clear
;       Uses:   nothing
;-------------------------------------------------------------------------
SPCIsCRTActive:
        PUSH    AX
        GETAUX  00h
        TEST    AL,01000000b            ; CRT enabled?
        JZ      SHORT NotActive         ; circuitry disabled (AL != FFh)

        GETAUX  08h                     ; see if any monitor attached
        AND     AL,0111b                ; keep Monitor ID bits only
        XOR     AL,0111b
        JZ      SHORT NotActive         ; AL = 0 ( != 0FFh )
        MOV     AL,0FFh
NotActive:
        INC     AL
        POP     AX
        RET
```

```
;-------------------------------------------------------------------------
;       SPCIsLCDActive - Test if LCD display active.
;
;       Entry:  n/a
;       Exit:   Z flg - Set if LCD active else clear
;       Uses:   nothing
;-------------------------------------------------------------------------
SPCIsLCDActive:
        PUSH    AX
        GETAUX  00h
        TEST    AL,00100000b            ; LCD enabled?
        JZ      SHORT LCDNotActive      ; circuitry disabled (AL != FFh)
        MOV     AL,0FFh
LCDNotActive:
        INC     AL
        POP     AX
        RET

;-------------------------------------------------------------------------
;       SPCIs1Meg - Test if 1 Mb of memory installed
;
;       Entry:  n/a
;       Exit:   Z flg - Set if 1 Mb installed
;       Uses:   nothing
;-------------------------------------------------------------------------
SPCIs1Meg:
        PUSH    AX
        GETAUX  03h
        TEST    AL,01000000b            ; single DRAM?
        POP     AX
        RET
```

```
;-------------------------------------------------------------------------------
;       GetPanelType
;
;       Entry:  N/A
;       Exit:   CS:SI Proper Table Info
;
;-------------------------------------------------------------------------------
GetPanelType PROC NEAR
        PUSH    AX
        GETAUX  0Ch                     ; read the MD lines
        NOT     AX
        AND     AX,01Fh
        LEA     SI,SetLCDTypeTbl
        SHL     AX,1
        ADD     SI,AX
        MOV     SI,CS:[SI]
        POP     AX
        RET
GetPanelType ENDP

;-------------------------------------------------------------------------------
;       SPCGetLCDLines
;
;       Entry:  N/A
;       Exit:   BX = VERTICAL PANEL SIZE in lines
;       Uses:   N/A
;       Note:   We cannot acces the B-set directly, as Windows gets confused.
;               So we do it via the panel table.
;-------------------------------------------------------------------------------
SPCGetLCDLines PROC NEAR
        PUSH    SI
        CALL    GetPanelType
        XOR     BX,BX
        MOV     BL,CS:[SI.PST.CRTCB_12]
        SHL     BX,2
        POP     SI
        RET
SPCGetLCDLines ENDP
```

```
;-------------------------------------------------------------------------
;       BlowRegisters - Initialize registers from table
;
;       Entry:  CS:SI - Pointer to register table
;
;       Exit:   n/a
;       Uses:   AX,DX,SI
;-------------------------------------------------------------------------
BlowRegisters:
        LODS    CS_WORD                 ; Read next control word
        CMP     AX,-1                   ; End of table?
        JE      SHORT BRX               ; Exit if so

        MOV     DX,AX                   ; Get port in DX
        TEST    DH,80h                  ; Test if port write?
        JNZ     SHORT BR_PW             ; Jump if so
BR2:
        LODS    CS_WORD                 ; Read index,data
        CMP     AL,-1                   ; End of run
        JNE     SHORT @F                ; Jump back if so
        DEC     SI
        JMP     SHORT BlowRegisters     ; Back for more
@@:
        OUT     DX,AX                   ; Write the data
        JMP     SHORT BR2               ; Back for more
BR_PW:
        LODS    CS_WORD                 ; Read data
        AND     DH,7Fh                  ; Mask off port write bit
        OUT     DX,AL                   ; Write the data
        JMP     SHORT BlowRegisters     ; Back for more
BRX:
        RET
```

Appendix D SPECIAL CASES

1. Driver configured for High Resolution and only LCD available on mode set.

   This is the responsibility of the driver to recognize the failure of the mode set call. Using VESA standard calls, the driver can determine if the mode is available. If the mode is not available at that given time, then the driver must either fail, set a lower mode and pan/scroll, or set a lower mode and deal with it. (e.g. Windows driver should pan/scroll; Autocad driver will set lower resolution mode.)

2. Driver configured for High Resolution and Simultaneous Display mode set.

   The BIOS will turn off the LCD panel until a return to a supported mode occurs. Auxiliary Register A bit 0 will be set to 1.

3. Request to set Simultaneous Display while in High Resolution mode.

   The BIOS will turn off the LCD panel until a return to a supported mode occurs. Auxiliary Register A bit 0 will be set to 1.

4. Request to switch display to CRT with none attached.

   Fail function request.

5. Request to switch displays to LCD when in High Resolution mode.

   Fail function request.

6. Monitor is disconnected, LCD resume state.

   The BIOS will resume on the display that it powered down on.

7. Monitor connected hot.

   Nothing should happen, since the monitor presence check only occurs on a mode change or a display selection request.

8. Monitor selection at boot time.

   If monitor is attached, it will default to the monitor, if the monitor not attached, it will assume color monitor modes and enable the LCD.

9. Hardware Cursor/Sprite and mode set.

   The BIOS disables the hardware cursor on mode sets as it is the responsibility of the mouse driver to re-initialize the cursor. The BIOS also programs the hardware cursor overlay colors to four shades of gray ranging from black to bright white.

10. For the Mass Production version of the SPC8110F0A the BIOS will modify all save/restore state functions to write a zero to bit 5 of AUX[20h]. In addition, when programming bit 5 of AUX[20h] in all functions the BIOS will ensure this bit is masked off prior to writing.

Appendix E BIOS DEFAULT CONFIGURATION OPTIONS

The following are some of the many customizable features within the BIOS:

- Sign on Message
  The power up displayed message can be altered or removed using the supplied 8110CFG program.
- Auto-Center
  The default state of Auto-Center can be set.
- Normal/Reverse/AutoReverse Text/Autoreverse Graphics
  The default screen color polarity for monochrome panels can be set.
- Gray-scale Weighting Default Algorithm (Green only/NTSC)
  The default gray-scale weighting can be set.
- Text/Graphics Expansion default
  The default state of expansion can be set.
- Default Segment (either C000h/E000h/F000h)
  The default BIOS segment can be set.
- Slow Cursor Enable
  The default cursor blink speed can be set.
- Default Cursor Shape
  The default cursor shape can be set.
- Panel Frame rate
  Custom panel frame rates can be changed.
- DAC ISA/PS/2
  The default DAC programming, IBM ISA format or IBM PS/2 format can be chosen.
- WF Timings
  WF timings can be altered.
- Custom Panel Type (Vert. size, 4/8 bit I/F, single/dual)
  Custom panel types and sizes can be added.
- Mode 7/F on color
  Support for Mode 7/F when running as a color VGA can be enabled, or like the IBM ISA VGA adapter, it can be disabled.
- Power Save State
  The default power save states can be changed, and DPMS added/removed.

Appendix F VESA INFORMATION

F.1 VESA Structures

The VBE 2.0 information block has the following structure:

```
VbeInfoBlock struc

VbeSignature         db    'VESA'      ; VBE Signature
VbeVersion           dw    0200h       ; VBE Version
OemStringPtr         dd    ?           ; Pointer to OEM String
Capabilities         db    4 dup (?)   ; Capabilities of graphics cont.
VideoModePtr         dd    ?           ; Pointer to Video Mode List
TotalMemory          dw    ?           ; Number of 64kb memory blocks
                                       ; Added for VBE 2.0
OemSoftwareRev       dw    ?           ; VBE implementation Software revision
OemVendorNamePtr     dd    ?           ; Pointer to Vendor Name String
OemProductNamePtr    dd    ?           ; Pointer to Product Name String
OemProductRevPtr     dd    ?           ; Pointer to Product Revision String
Reserved             db    222 dup (?) ; Reserved for VBE implementation scratch
                                       ; area
OemData              db    256 dup (?) ; Data Area for OEM Strings

VbeInfoBlock ends
OEMString            db    'Seiko Epson VGA VESA BIOS'

OEMVendorName        db    'Seiko Epson Corp.'
OEMProductName       db    'SPC8110'
OEMProductRev        db    '2.0'
```

The VBE 1.2 information block has the following structure:

```
VGAInfoBlock struc
    VESASignature    db    'VESA'      ; 4 signature bytes
    VESAVersion      dw    0102h       ; VESA version number
    OEMStringPtr     dd    ?           ; Pointer to OEM string
    Capabilities     dd    1           ; capabilities of the video environment
    VideoModePtr     dd    ?           ; pointer to supported VESA modes
    TotalMemory      dw    4           ; Number of 64kb memory blocks on board
    Reserved         db    236 dup (0) ; Remainder of VGAInfoBlock
VGAInfoBlock ends
OEMString            db    'Seiko Epson VGA VESA BIOS'

ModeInfoBlock   struc ;(mode 0108h)
    ModeAttributes   dw    000Fh       ; mode attributes
    WinAAttributes   db    00h         ; window A attributes
    WinBAttributes   db    00h         ; window B attributes
    WinGranularity   dw    40h         ; window granularity
    WinSize          dw    40h         ; window size
    WinASegment      dw    0b800h      ; window A start segment
    WinBSegment      dw    00h         ; window B start segment
```

```
    WinFuncPtr        dd    00h          ; pointer to window function
    BytesPerScanLine  dw    A0h          ; bytes per scan line
    XResolution       dw    80h          ; horizontal resolution
    YResolution       dw    60h          ; vertical resolution
    XCharSize         db    08h          ; character cell width
    YCharSize         db    08h          ; character cell height
    NumberOfPlanes    db    03h          ; number of memory planes
    BitsPerPixel      db    04h          ; bits per pixel
    NumberOfBanks     db    01h          ; number of banks
    MemoryModel       db    00h          ; memory model type
    BankSize          db    08h          ; bank size in kb
    NumberOfImagePagesdb    02h          ; No. of Additional Images (0 base)
    Reserved          db    1            ; reserved for page function
    Reserved          db    225 dup (?) ; remainder of ModeInfoBlock
ModeInfoBlock   ends
```

F.2 VESA Mode Support

F.2.1 VESA Graphics Modes

15-bit Mode Number	7-bit Mode Number	Resolution	Colors	Supported	TTY Functions Supported
0100h	-	640x400	256	Yes	Yes
0101h	-	640x480	256	Yes	Yes
0102h	6Ah	800x600	16	Yes	Yes
0103h	-	800x600	256	Yes	Yes
0104h	-	1024x768	16	Yes	Yes
0105h	-	1024x768	256	Yes	Yes
0106h	-	1280x1024	16	No	No
0107h	-	1280x1024	256	No	No

F.2.2 VESA Text Modes

15-bit Mode Number	7-bit Mode Number	Columns	Rows	Supported	TTY Functions Supported
0108h	-	80	60	Yes	Yes
0109h	-	132	25	Yes	Yes
010Ah	-	132	43	Yes	Yes
010Bh	-	132	50	Yes	Yes
010Ch	-	132	60	Yes	Yes

F.2.3 VESA High Color Graphics Modes

15-bit Mode Number	7-bit Mode Number	Resolution	Colors	Supported	TTY Functions Supported
010Dh	-	320x200	32K (1:5:5:5:)	No	No
010Eh	-	320x200	64K (5:6:5)	No	No
010Fh	-	320x200	16.8M (8:8:8)	No	No
0110h	-	640x480	32K (1:5:5:5:)	No	No
0111h	-	640x480	64K (5:6:5)	No	No
0112h	-	640x480	16.8M (8:8:8)	No	No
0113h	-	800x600	32K (1:5:5:5:)	No	No
0114h	-	800x600	64K (5:6:5)	No	No
0115h	-	800x600	16.8M (8:8:8)	No	No
0116h	-	1024x768	32K (1:5:5:5:)	No	No
0117h	-	1024x768	64K (5:6:5)	No	No
0118h	-	1024x768	16.8M (8:8:8)	No	No
0119h	-	1280x1024	32K (1:5:5:5:)	No	No
011Ah	-	1280x1024	64K (5:6:5)	No	No
011Bh	-	1280x1024	16.8M (8:8:8)	No	No

Appendix G SIMULTANEOUS DISPLAY ISSUES

One of the features of the SPC8110F0A is that it is capable of simultaneous display. The following are the BIOS limitations for this display mode.

- Simultaneous display will only be supported fully for valid LCD panel modes. Modes beyond the ability of the LCD panel will not display the same on the CRT and LCD.
- Simultaneous display on a dual panel LCD will run at double the normal frame rate for the LCD. The CRT will run at the same clock rate as the LCD and will most likely require a multi-sync monitor to be attached.
- Autocenter and Stretch will be hardware suppressed during simultaneous display, they will continue to readback as though they were on.
- All power save modes will continue to work in simultaneous display.
- A request for simultaneous display while in a mode that is not supported on the LCD will result in the flag to support simultaneous display to be turned on, but the LCD will be turned off and the CRT will become active.
- If the allow simultaneous display bit is active and the LCD is off, if the mode requested is supported on LCD, both the CRT and LCD will be enabled.

The basic rule for simultaneous display:

- A simultaneous display mode set will always succeed except when the mode is flagged (for whatever reason) as unavailable to simultaneous display or no CRT exists.

Appendix H BIOS HEADER INFORMATION

The Seiko Epson VGA BIOS has a Header Structure as follows. This header allows for flexibility in configuring the default information. By modifying strings or configuration information either contained in the header or pointed to by the header, it is possible to change Signon strings, default panel tables, DAC tables etc. If the BIOS does get configured by an external configuration program, the CONFIGURATION WORD at offset 0014h should be modified accordingly.

```
                                Offset
                                ------
                DB      055h,0AAh,0     ; 0000 PC ROM flag bytes
                JMP     NEAR PTR Init   ; 0003 Link to startup init code
                DW      HEX_VERSION     ; 0006 BIOS version in hex
                DW      SaveTblVGA      ; 0008 Address of Save tables
                DW      VparmTblVGA     ; 000A Address of Video Parameter tables
                DW      DacTables       ; 000C Address of DacTables
                DW      IDText          ; 000E Address of signon message
                DW      InitTheWorldTbl ; 0010 Initialize everything table
                DW      SetLCDTypeTbl   ; 0012 Panel sizes tables

                DW      ??              ; 0014 CONFIGURATION WORD
                                        ;      Bit 0  = 1:DEBUG Bios
                                        ;      Bit 1  = 1:Disable modes 7 & F
                                        ;      Bit 2  = 1:Disable DDC support
                                        ;      Bit 14 = 1:Re-configured panels
                                        ;      Bit 15 = 1:Re-configured BIOS
                DW      ??              ; 0016 Address of debug panel info if
                                        ;      DEBUG BIOS otherwise RESERVED
                DW      ??              ; 0018 Address of PCI Data Structure
                                        ;      if PCI BIOS otherwise RESERVED
                DW      MiscCfgTables   ; 001A Additional config tables
                DB      0               ; 001C Reserved

;               C001Dh = vvvvv Some rude software expects this to work!
SPCHeaderText   DB      ' IBM COMPATIBLE ',CHIP_NAME
                DB      ' VGA BIOS (V',BIOS_VERSION,SUBVER,')'
CopyrightText   DB      'Copyright (c) Seiko Epson Corp. 1987, 1996.  All rights
reserved.',0
```

Appendix I PCI HEADER INFORMATION

The PCI BIOS Header Information has the following structure:

```
VENDOR_ID      EQU     10F4h     ; SPC8110 embedded
DEVICE_ID      EQU     8110h     ; SPC8110 embedded

      align 4                    ; must be DWORD aligned
PCIDataStructure:
      db      'PCIR'             ; Offset 00: Signature
      dw      VENDOR_ID          ; Offset 04: Vendor Identification
      dw      DEVICE_ID          ; Offset 06: Device Identification
      dw      0                  ; Offset 08: Pointer to VPD (none)
      dw      18h                ; Offset 0A: Size of this PCI Data Structure
      db      0                  ; Offset 0C: PCI Data Structure rev.(0)
      db      00,00,03           ; Offset 0D: Class code VGA
      dw      8000h/512          ; Offset 10: Image length in 512 blocks (32k)
      dw      HEX_VERSION        ; Offset 12: Bios revision
      db      0                  ; Offset 14: Code type x86
      db      80h                ; Offset 15: Last image
      dw      0                  ; Offset 16: Reserved
.errnz $ - PCIDataStructure-18h; must be same as offset 0C
```

S-MOS
SYSTEMS
A Seiko Epson Affiliate

SPC8110F0A LOCAL BUS LCD/CRT VGA CONTROLLER

Programming Notes and Examples

Drawing Office No. X07-AN-004-04

Copyright © 1996, 1997 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

The sample code contained within this document may be used without restriction.

THIS PAGE LEFT BLANK

Table of Contents

INITIALIZING THE SPC8110 WITHOUT A BIOS 5
Initializing The SPC8110 In Mode 03h Without A BIOS 5
Initializing The SPC8110 In Mode 12h Without A BIOS 31
Initializing The SPC8110 In Mode 101h Without A BIOS 48
LCD.DAT 65
USING THE SPC8110 BitBLT ACCELERATOR 66
SPC8110 BitBLT Types 66
The BitBLT Registers 67
Programming The BitBLT Engine 70
Programming Notes 70
Program Examples 76
BLT-SAMP.C - Sample Programming Code for the SPC8110 76
Assembler Support Routines for BLT-SAMP.C 107
PROGRAMMING THE SPRITE / HARDWARE CURSOR 113
Image Loading and Hardware Enabling 114
Setting the Sprite Position 117
PROGRAMMING THE PLL AND FIFO 119
How To Program The PLL Clock 119
How To Program The FIFO 120
Recommended Procedure For Non-VGA Compatible Systems 120
Recommended Procedure For VGA Compatible Systems 121
MANUAL SETTING OF POWER STATES ON THE SPC8110 127
Comments 127
Example code 128
IDENTIFYING THE SPC8110 130
ID BIT LIMITATION 132
ENABLING AND DISABLING THE SPC8110 133
USING THE SPC8110 DISPLAY MEMORY 134
Paging 134
Linear Mapping 134
Using VESA BIOS Extensions 134
Special Notes About Hardware Cursor And Half Frame Buffer 134
Banking Sample Code 135
Linear Mapping Sample Code 136
FRAME BUFFER FOR DUAL DRIVE PANELS 138
640x480 2-BITS PER PIXEL (BPP) GRAPHICS MODE PROCEDURE 139

THIS PAGE LEFT BLANK

INITIALIZING THE SPC8110 WITHOUT A BIOS

Initializing The SPC8110 In Mode 03h Without A BIOS

The purpose of this document is to demonstrate how to initialize the SPC8110 into a mode similar to IBM's Standard VGA mode 3 by supplying sample code. The steps involved in this are:

- Initializing the SPC8110, programing all necessary registers and printing a string on the CRT or LCD.
- Setting up the SPC8110 panel specific registers.
- Setting up the Standard VGA registers for Mode 3 (with the exception of only loading an 8 point font versus the 16 point font).
- Loading the font into Memory (required for text mode only).
- Initializing the RAMDAC values. The overlay RAMDAC (sprite) is not initialized.
- Setting up the attribute registers.
- Initializing Video Memory. This program assumes 80ns DRAMs with self refresh. If there are DRAM problems, modify values for AUX[03]. If using 100ns DRAMS, program the PLL2 to 33Mhz.

The resulting mode is not BIOS supported, so no DOS text will work, however, writing to the memory location B800:0000 values 41 07 will put a white letter A in the top left corner.

```
;
;-----------------------------------------------------------------------------
;
; Instructions: using MASM 6.11
;
;  - Make your changes
;  - Assemble       --  masm MODE3.asm;
;  - Link          --  link MODE3;
;  - Make a .COM file  --  exe2bin MODE3.exe MODE3.com
;
;
;-----------------------------------------------------------------------------

.286
.MODEL TINY

PciSupported  EQU       1           ; comment off this line to skip PCI enable
PW            EQU       8000h       ; Port write (RunTable)
G11           EQU       11000000b
VENDOR_ID     EQU       10F4h
DEVICE_ID     EQU       8110h
```

```
GETAUX      MACRO       reg         ; Read a SPCxxxx AUX register
            mov         al,reg
            call        SPCReadAuxReg
            ENDM


PUTAUX      MACRO       reg,data    ; Write a SPCxxxx AUX register
            mov         al,data
            mov         ah,reg
            call        SPCWriteAuxReg
            ENDM


.CODE

CS_BYTE     LABEL       BYTE        ; Local byte reference
CS_WORD     LABEL       WORD        ; Local word reference

            ORG         100h
MODE3       PROC        NEAR
            cld                     ; Set the direction flag.
            mov         ax, cs
            mov         ds, ax      ;
            mov         es, ax      ;
            cli                     ; Interupts off while changing stack.
            mov         ss, ax      ;
            mov         sp, 0FFFFh  ;
            sti                     ; Stack change done: restore interupts.

            call        Step1       ; Enable SPC8110 chip.

            ; at this point, all the registers are accessible, we need to
            ; program Auxilliary registers according to panel size, memory,
            ; and some preferences...

            call        Step2       ; Setup for the panel and enviroment.

            ; At this point all non-standard registers are fully initialized.
            ; The following is standard VGA stuff that would run without any
            ; change on any IBM-compatible VGA chip.

            call        Step3       ; Setup the standard VGA registers.
            call        Step4       ; Load the font.
            call        Step5       ; Blow the DAC with mode 3 colors.
            call        Step6       ; Setup the attribute registers.
            call        Step7       ; Clear VRAM

            ;  You should now be running as MODE 3 without BIOS support.
```

```
        mov         si,offset Message
        call        TextOut

        mov         ax, 4C00h           ; DOS - Exit with Return Code.
        int         21h


Message:
        DB 'MODE3.COM : programming SPC8110 registers for 8x8 font text mode
without BIOS.      ',0

;-------------------------------------------------------------------------
;
;  STEP1 - Before ANY ports can be accessed, the SPC8110 chip must be
;          enabled. (For PCI systems, the Memory & IO decode must be
;          enabled prior to writing to 3c3h)
;
;-------------------------------------------------------------------------

Step1:
        IFDEF       PciSupported
        ECHO        PCI Support Enabled
        call        SPCPCIEnable
        ENDIF

        mov         dx, 3c3h
        mov         al, 1
        out         dx, al
        ret


;-----------------------------------------------------------------------------
;
;  STEP2 - This sets the chip up for the correct panel and for
;          a proper working environment.
;
;
;-----------------------------------------------------------------------------

Step2:
        mov         si, OFFSET InitTheWorldTbl
        call        RunTable

        ; We have just programmed the memory clock to 40MHz.
        ; Accessing video memory is not safe for a few miliseconds,
        ; as the PLL is not stable right away. We simply wait
        ; about 300ms (slight overkill).

        mov         al, 300/8       ; Setup 300mS wait
        call        Delay
        call        SetupDisplay
        ret
```

```
InitTheWorldTbl:
        DW   PW+03C2h,63h     ; default card to 3Dx, 25Mhz Clock
        DW   03DEh            ; program the Aux. registers

        DB   00h,00000000b    ; clear register

        DB   01H,01000010b    ; autocenter, 2Hz blink
        DB   02H,01001000b    ; 8bit mono dual LCD panel
        DB   03H,00101000b    ; Self refresh DRAM (Wait state=Turbo)
        DB   04H,00000000b    ; clear scratch pad
        DB   05H,00000000b    ; disable memory paging, HIREZ256 etc.
        DB   06H,00100000b    ; LCD WF count register = 20h
        DB   07H,00000000b    ; LCD Display register

;;;     DB   08H,00000000b    ; RO
        DB   0AH,00000000b    ; clear scratch pad
        DB   0BH,00000000b    ; clear scratch pad
;;;     DB   0CH,00000000b    ; treat as RO !
;;;     DB   0DH,00000000b    ; RO
;;;     DB   0FH,00000000b    ; RO

        ; set default values for PD registers

        DB   10H,00000000b    ; polarity PS1 & PS4 low, no timer
        DB   11H,00000011b    ; PS 1: GreyScale, Slow Clock,H&V sync active
        DB   12H,00000011b    ; PS 2: GreyScale, Slow Clock,H&V sync active
        DB   13H,10110111b    ; PS 3:MClk Off,LCDBack off,DPMS Suspend,GS,DisplayOff,
        DB   14H,10110111b    ; PS 4:MClk Off,LCDBack off,DPMS Suspend,GS,DisplayOff,
        DB   15H,00001100b    ; PS Disable PS1&PS4, skew XX
        DB   16H,01001111b    ; PS Skew Register
        DB   17H,00000001b    ; PS SkewA b3=1

        DB   1AH,00000000b    ; Video Buffer High
        DB   1BH,00000000b    ; Video Buffer Low

        ; Frame buffer in the last page, programmed for mono dual 640x480
        ; panels

        DB   1CH,00001111b    ; Frame Start (Dual LCD only)
        DB   1DH,00010010b    ; Frame Length High
        DB   1EH,10111101b    ; Frame Length Low

        DB   1FH,00000000b    ; Frame Buffer enable + High Request

        ; PLL1 set 0&1 determined based on LCD, done elsewhere

;;;     DB   20H,00000000b    ; PLL1 Set0 Numerator
;;;     DB   21H,00000000b    ; PLL1 Set0 Denominator
;;;     DB   22H,00000000b    ; PLL1 Set1 Numerator
```

```
;;;  DB  23H,00000000b    ; PLL1 Set1 Denominator

     ; PLL1 set 2 not used here

     DB  24H,00000000b    ; PLL1 Set2 Numerator
     DB  25H,00000000b    ; PLL1 Set2 Denominator

     ; PLL2 programmed for 80ns DRAM (40MHz Memory clock)

     DB  27H,00000100b    ; PLL2 Denominator (must be first)
     DB  26H,00001011b    ; PLL2 Numerator

     DB  28H,00000000b    ; clear any lock override

     ; 2a..33 are N/A

     DB  34H,00000000b    ; disable GENIO outputs

     ; FIFOs determined based on LCD...

     DB  38H,14h          ; FIFO
     DB  39H,25h          ; FIFO
     DB  3AH,04h          ; FIFO
     DB  3BH,14h          ; FIFO
;;;  DB  3CH,00000000b    ; FIFO

     DB  -1               ; end of Aux registers
     DW  -1               ; end of the register table

;-------------------------------------------------------------------------
;    SetupDisplay - Setup LCD display operation.
;
;    Entry:  n/a
;    Exit:   n/a
;    Uses:   AX,BX,CX,DX,SI,DI
;-------------------------------------------------------------------------
SetupDisplay:

     ; program the panel parameters
     GETAUX   00h
     or       al, 20h     ; select panel
     PUTAUX   00h, al

     call     SPCLoadPanelRegs

setupdisplayx:
     ret
```

```
;-------------------------------------------------------------------------
;
;  STEP3 - Use tables to program the standard VGA registers.
;
;-------------------------------------------------------------------------

Step3:
            mov         si, offset VGAregs
            call        RunTable
            ret

VGAregs:
            DW          3C4h                ; Sequencer.
            DB          00h,00h
            DB          01h,01h             ; 8 dot clock
            DB          02h,03h
            DB          03h,00h
            DB          04h,02h
            DB          00h,03h
            DB          -1

            DW          3D4H                ; CRTC
            DB          11H,0CH             ; enable CRTC regs 0..7
            DB          00H,05Fh
            DB          01H,04Fh
            DB          02H,050h
            DB          03H,082h
            DB          04H,055h
            DB          05H,081h
            DB          06H,0BFh
            DB          07H,01Fh
            DB          08H,000h
            DB          09H,0C7h            ; font height = 8
            DB          0AH,006h
            DB          0BH,007h
            DB          0CH,000h
            DB          0DH,000h
            DB          0EH,000h
            DB          0FH,000h
            DB          10H,09Ch
            DB          11H,08Eh
            DB          12H,08Fh
            DB          13H,028h
            DB          14H,01Fh
            DB          15H,096h
            DB          16H,0B9h
            DB          17H,0A3h
            DB          18H,0FFh
            DB          -1
```

```
        DW              3CEH                ; Graphics.
        DB              00H,000h
        DB              01H,000h
        DB              02H,000h
        DB              03H,000h
        DB              04H,000h
        DB              05H,010h
        DB              06H,00Eh
        DB              07H,000h
        DB              08H,0FFh
        DB              -1
        DW              -1

;-------------------------------------------------------------------------
;
;    STEP4 - Load Font. Load font into plane 2.
;
;-------------------------------------------------------------------------

Step4:
          mov           si, offset OpenFontMapTbl
          call          RunTable                ; estabilish planar addressing
                                                ; at A000h, enable plane 2 for write

          mov           si, offset CharSet8     ; 8x8 font offset
          mov           ax, 0A000h              ; Set ES to Vram segment
          mov           es, ax
          xor           di, di
          mov           ax, 256                 ; Set character count
@@:
          mov           cx, 4                   ; Set point count (4 WORDS)
          rep           movsw                   ; Transfer this character

          add           di, 32-8                ; Next map entry
          dec           ax                      ; Count character
          jnz           @B                      ; Do all characters

          mov           si, offset CloseFontMapTbl
          call          RunTable                ; estabilish text addressing again
          ret

OpenFontMapTbl:
          DW            3C4h                    ; Sequencer.
          DB            02h,04h
          DB            04h,07h
          DB            -1
          DW            3CEH                    ; Graphics controller
          DB            05H,00h
          DB            06H,04h
          DB            -1
          DW            -1                      ; End of table
```

```
CloseFontMapTbl:
            DW          3C4h                    ; Sequencer.
            DB          02h,03h
            DB          04h,02h
            DB          -1
            DW          3CEH                    ; Graphics controller
            DB          05H,10h
            DB          06H,0Eh
            DB          -1
            DW          -1                      ; End of table


CharSet8:
            DB          000h,000h,000h,000h,000h,000h,000h,000h ; 0
            DB          07Eh,081h,0A5h,081h,0BDh,099h,081h,07Eh ; 1
            DB          07Eh,0FFh,0DBh,0FFh,0C3h,0E7h,0FFh,07Eh ; 2
            DB          06Ch,0FEh,0FEh,0FEh,07Ch,038h,010h,000h ; 3
            DB          010h,038h,07Ch,0FEh,07Ch,038h,010h,000h ; 4
            DB          038h,07Ch,038h,0FEh,0FEh,07Ch,038h,07Ch ; 5
            DB          010h,010h,038h,07Ch,0FEh,07Ch,038h,07Ch ; 6
            DB          000h,000h,018h,03Ch,03Ch,018h,000h,000h ; 7
            DB          0FFh,0FFh,0E7h,0C3h,0C3h,0E7h,0FFh,0FFh ; 8
            DB          000h,03Ch,066h,042h,042h,066h,03Ch,000h ; 9
            DB          0FFh,0C3h,099h,0BDh,0BDh,099h,0C3h,0FFh ; 10
            DB          00Fh,007h,00Fh,07Dh,0CCh,0CCh,0CCh,078h ; 11
            DB          03Ch,066h,066h,066h,03Ch,018h,07Eh,018h ; 12
            DB          03Fh,033h,03Fh,030h,030h,070h,0F0h,0E0h ; 13
            DB          07Fh,063h,07Fh,063h,063h,067h,0E6h,0C0h ; 14
            DB          099h,05Ah,03Ch,0E7h,0E7h,03Ch,05Ah,099h ; 15
            DB          080h,0E0h,0F8h,0FEh,0F8h,0E0h,080h,000h ; 16
            DB          002h,00Eh,03Eh,0FEh,03Eh,00Eh,002h,000h ; 17
            DB          018h,03Ch,07Eh,018h,018h,07Eh,03Ch,018h ; 18
            DB          066h,066h,066h,066h,066h,000h,066h,000h ; 19
            DB          07Fh,0DBh,0DBh,07Bh,01Bh,01Bh,01Bh,000h ; 20
            DB          03Eh,063h,038h,06Ch,06Ch,038h,0CCh,078h ; 21
            DB          000h,000h,000h,000h,07Eh,07Eh,07Eh,000h ; 22
            DB          018h,03Ch,07Eh,018h,07Eh,03Ch,018h,0FFh ; 23
            DB          018h,03Ch,07Eh,018h,018h,018h,018h,000h ; 24
            DB          018h,018h,018h,018h,07Eh,03Ch,018h,000h ; 25
            DB          000h,018h,00Ch,0FEh,00Ch,018h,000h,000h ; 26
            DB          000h,030h,060h,0FEh,060h,030h,000h,000h ; 27
            DB          000h,000h,0C0h,0C0h,0C0h,0FEh,000h,000h ; 28
            DB          000h,024h,066h,0FFh,066h,024h,000h,000h ; 29
            DB          000h,018h,03Ch,07Eh,0FFh,0FFh,000h,000h ; 30
            DB          000h,0FFh,0FFh,07Eh,03Ch,018h,000h,000h ; 31
            DB          000h,000h,000h,000h,000h,000h,000h,000h ;
            DB          030h,078h,078h,030h,030h,000h,030h,000h ; !
            DB          06Ch,06Ch,06Ch,000h,000h,000h,000h,000h ; "
            DB          06Ch,06Ch,0FEh,06Ch,0FEh,06Ch,06Ch,000h ; #
```

```
DB          030h,07Ch,0C0h,078h,00Ch,0F8h,030h,000h ; $
DB          000h,0C6h,0CCh,018h,030h,066h,0C6h,000h ; %
DB          038h,06Ch,038h,076h,0DCh,0CCh,076h,000h ; &
DB          060h,060h,0C0h,000h,000h,000h,000h,000h ; '
DB          018h,030h,060h,060h,060h,030h,018h,000h ; (
DB          060h,030h,018h,018h,018h,030h,060h,000h ; )
DB          000h,066h,03Ch,0FFh,03Ch,066h,000h,000h ; *
DB          000h,030h,030h,0FCh,030h,030h,000h,000h ; +
DB          000h,000h,000h,000h,000h,030h,030h,060h ; ,
DB          000h,000h,000h,0FCh,000h,000h,000h,000h ; -
DB          000h,000h,000h,000h,000h,030h,030h,000h ; .
DB          006h,00Ch,018h,030h,060h,0C0h,080h,000h ; /
DB          07Ch,0C6h,0CEh,0DEh,0F6h,0E6h,07Ch,000h ; 0
DB          030h,070h,030h,030h,030h,030h,0FCh,000h ; 1
DB          078h,0CCh,00Ch,038h,060h,0CCh,0FCh,000h ; 2
DB          078h,0CCh,00Ch,038h,00Ch,0CCh,078h,000h ; 3
DB          01Ch,03Ch,06Ch,0CCh,0FEh,00Ch,01Eh,000h ; 4
DB          0FCh,0C0h,0F8h,00Ch,00Ch,0CCh,078h,000h ; 5
DB          038h,060h,0C0h,0F8h,0CCh,0CCh,078h,000h ; 6
DB          0FCh,0CCh,00Ch,018h,030h,030h,030h,000h ; 7
DB          078h,0CCh,0CCh,078h,0CCh,0CCh,078h,000h ; 8
DB          078h,0CCh,0CCh,07Ch,00Ch,018h,070h,000h ; 9
DB          000h,030h,030h,000h,000h,030h,030h,000h ; :
DB          000h,030h,030h,000h,000h,030h,030h,060h ; ;
DB          018h,030h,060h,0C0h,060h,030h,018h,000h ; <
DB          000h,000h,0FCh,000h,000h,0FCh,000h,000h ; =
DB          060h,030h,018h,00Ch,018h,030h,060h,000h ; >
DB          078h,0CCh,00Ch,018h,030h,000h,030h,000h ; ?
DB          07Ch,0C6h,0DEh,0DEh,0DEh,0C0h,078h,000h ; @
DB          030h,078h,0CCh,0CCh,0FCh,0CCh,0CCh,000h ; A
DB          0FCh,066h,066h,07Ch,066h,066h,0FCh,000h ; B
DB          03Ch,066h,0C0h,0C0h,0C0h,066h,03Ch,000h ; C
DB          0F8h,06Ch,066h,066h,066h,06Ch,0F8h,000h ; D
DB          0FEh,062h,068h,078h,068h,062h,0FEh,000h ; E
DB          0FEh,062h,068h,078h,068h,060h,0F0h,000h ; F
DB          03Ch,066h,0C0h,0C0h,0CEh,066h,03Eh,000h ; G
DB          0CCh,0CCh,0CCh,0FCh,0CCh,0CCh,0CCh,000h ; H
DB          078h,030h,030h,030h,030h,030h,078h,000h ; I
DB          01Eh,00Ch,00Ch,00Ch,0CCh,0CCh,078h,000h ; J
DB          0E6h,066h,06Ch,078h,06Ch,066h,0E6h,000h ; K
DB          0F0h,060h,060h,060h,062h,066h,0FEh,000h ; L
DB          0C6h,0EEh,0FEh,0FEh,0D6h,0C6h,0C6h,000h ; M
DB          0C6h,0E6h,0F6h,0DEh,0CEh,0C6h,0C6h,000h ; N
DB          038h,06Ch,0C6h,0C6h,0C6h,06Ch,038h,000h ; O
DB          0FCh,066h,066h,07Ch,060h,060h,0F0h,000h ; P
DB          078h,0CCh,0CCh,0CCh,0DCh,078h,01Ch,000h ; Q
DB          0FCh,066h,066h,07Ch,06Ch,066h,0E6h,000h ; R
DB          078h,0CCh,0E0h,070h,01Ch,0CCh,078h,000h ; S
DB          0FCh,0B4h,030h,030h,030h,030h,078h,000h ; T
```

```
DB          0CCh,0CCh,0CCh,0CCh,0CCh,0CCh,0FCh,000h ; U
DB          0CCh,0CCh,0CCh,0CCh,0CCh,078h,030h,000h ; V
DB          0C6h,0C6h,0C6h,0D6h,0FEh,0EEh,0C6h,000h ; W
DB          0C6h,0C6h,06Ch,038h,038h,06Ch,0C6h,000h ; X
DB          0CCh,0CCh,0CCh,078h,030h,030h,078h,000h ; Y
DB          0FEh,0C6h,08Ch,018h,032h,066h,0FEh,000h ; Z
DB          078h,060h,060h,060h,060h,060h,078h,000h ; [
DB          0C0h,060h,030h,018h,00Ch,006h,002h,000h ; \
DB          078h,018h,018h,018h,018h,018h,078h,000h ; ]
DB          010h,038h,06Ch,0C6h,000h,000h,000h,000h ; ^
DB          000h,000h,000h,000h,000h,000h,000h,0FFh ; _
DB          030h,030h,018h,000h,000h,000h,000h,000h ; ‘
DB          000h,000h,078h,00Ch,07Ch,0CCh,076h,000h ; a
DB          0E0h,060h,060h,07Ch,066h,066h,0DCh,000h ; b
DB          000h,000h,078h,0CCh,0C0h,0CCh,078h,000h ; c
DB          01Ch,00Ch,00Ch,07Ch,0CCh,0CCh,076h,000h ; d
DB          000h,000h,078h,0CCh,0FCh,0C0h,078h,000h ; e
DB          038h,06Ch,060h,0F0h,060h,060h,0F0h,000h ; f
DB          000h,000h,076h,0CCh,0CCh,07Ch,00Ch,0F8h ; g
DB          0E0h,060h,06Ch,076h,066h,066h,0E6h,000h ; h
DB          030h,000h,070h,030h,030h,030h,078h,000h ; i
DB          00Ch,000h,00Ch,00Ch,00Ch,0CCh,0CCh,078h ; j
DB          0E0h,060h,066h,06Ch,078h,06Ch,0E6h,000h ; k
DB          070h,030h,030h,030h,030h,030h,078h,000h ; l
DB          000h,000h,0CCh,0FEh,0FEh,0D6h,0C6h,000h ; m
DB          000h,000h,0F8h,0CCh,0CCh,0CCh,0CCh,000h ; n
DB          000h,000h,078h,0CCh,0CCh,0CCh,078h,000h ; o
DB          000h,000h,0DCh,066h,066h,07Ch,060h,0F0h ; p
DB          000h,000h,076h,0CCh,0CCh,07Ch,00Ch,01Eh ; q
DB          000h,000h,0DCh,076h,066h,060h,0F0h,000h ; r
DB          000h,000h,07Ch,0C0h,078h,00Ch,0F8h,000h ; s
DB          010h,030h,07Ch,030h,030h,034h,018h,000h ; t
DB          000h,000h,0CCh,0CCh,0CCh,0CCh,076h,000h ; u
DB          000h,000h,0CCh,0CCh,0CCh,078h,030h,000h ; v
DB          000h,000h,0C6h,0D6h,0FEh,0FEh,06Ch,000h ; w
DB          000h,000h,0C6h,06Ch,038h,06Ch,0C6h,000h ; x
DB          000h,000h,0CCh,0CCh,0CCh,07Ch,00Ch,0F8h ; y
DB          000h,000h,0FCh,098h,030h,064h,0FCh,000h ; z
DB          01Ch,030h,030h,0E0h,030h,030h,01Ch,000h ; {
DB          018h,018h,018h,000h,018h,018h,018h,000h ; |
DB          0E0h,030h,030h,01Ch,030h,030h,0E0h,000h ; }
DB          076h,0DCh,000h,000h,000h,000h,000h,000h ; ~
DB          000h,010h,038h,06Ch,0C6h,0C6h,0FEh,000h ; 127

; EXTENDED character set

DB          078h,0CCh,0C0h,0CCh,078h,018h,00Ch,078h ; 128
DB          000h,0CCh,000h,0CCh,0CCh,0CCh,07Eh,000h ; 129
DB          01Ch,000h,078h,0CCh,0FCh,0C0h,078h,000h ; 130
```

```
DB          07Eh,0C3h,03Ch,006h,03Eh,066h,03Fh,000h ; 131
DB          0CCh,000h,078h,00Ch,07Ch,0CCh,07Eh,000h ; 132
DB          0E0h,000h,078h,00Ch,07Ch,0CCh,07Eh,000h ; 133
DB          030h,030h,078h,00Ch,07Ch,0CCh,07Eh,000h ; 134
DB          000h,000h,078h,0C0h,0C0h,078h,00Ch,038h ; 135
DB          07Eh,0C3h,03Ch,066h,07Eh,060h,03Ch,000h ; 136
DB          0CCh,000h,078h,0CCh,0FCh,0C0h,078h,000h ; 137
DB          0E0h,000h,078h,0CCh,0FCh,0C0h,078h,000h ; 138
DB          0CCh,000h,070h,030h,030h,030h,078h,000h ; 139
DB          07Ch,0C6h,038h,018h,018h,018h,03Ch,000h ; 140
DB          0E0h,000h,070h,030h,030h,030h,078h,000h ; 141
DB          0C6h,038h,06Ch,0C6h,0FEh,0C6h,0C6h,000h ; 142
DB          030h,030h,000h,078h,0CCh,0FCh,0CCh,000h ; 143
DB          01Ch,000h,0FCh,060h,078h,060h,0FCh,000h ; 144
DB          000h,000h,07Fh,00Ch,07Fh,0CCh,07Fh,000h ; 145
DB          03Eh,06Ch,0CCh,0FEh,0CCh,0CCh,0CEh,000h ; 146
DB          078h,0CCh,000h,078h,0CCh,0CCh,078h,000h ; 147
DB          000h,0CCh,000h,078h,0CCh,0CCh,078h,000h ; 148
DB          000h,0E0h,000h,078h,0CCh,0CCh,078h,000h ; 149
DB          078h,0CCh,000h,0CCh,0CCh,0CCh,07Eh,000h ; 150
DB          000h,0E0h,000h,0CCh,0CCh,0CCh,07Eh,000h ; 151
DB          000h,0CCh,000h,0CCh,0CCh,07Ch,00Ch,0F8h ; 152
DB          0C3h,018h,03Ch,066h,066h,03Ch,018h,000h ; 153
DB          0CCh,000h,0CCh,0CCh,0CCh,0CCh,078h,000h ; 154
DB          018h,018h,07Eh,0C0h,0C0h,07Eh,018h,018h ; 155
DB          038h,06Ch,064h,0F0h,060h,0E6h,0FCh,000h ; 156
DB          0CCh,0CCh,078h,0FCh,030h,0FCh,030h,030h ; 157
DB          0F8h,0CCh,0CCh,0FAh,0C6h,0CFh,0C6h,0C7h ; 158
DB          00Eh,01Bh,018h,03Ch,018h,018h,0D8h,070h ; 159
DB          01Ch,000h,078h,00Ch,07Ch,0CCh,07Eh,000h ; 160
DB          038h,000h,070h,030h,030h,030h,078h,000h ; 161
DB          000h,01Ch,000h,078h,0CCh,0CCh,078h,000h ; 162
DB          000h,01Ch,000h,0CCh,0CCh,0CCh,07Eh,000h ; 163
DB          000h,0F8h,000h,0F8h,0CCh,0CCh,0CCh,000h ; 164
DB          0FCh,000h,0CCh,0ECh,0FCh,0DCh,0CCh,000h ; 165
DB          03Ch,06Ch,06Ch,03Eh,000h,07Eh,000h,000h ; 166
DB          038h,06Ch,06Ch,038h,000h,07Ch,000h,000h ; 167
DB          030h,000h,030h,060h,0C0h,0CCh,078h,000h ; 168
DB          000h,000h,000h,0FCh,0C0h,0C0h,000h,000h ; 169
DB          000h,000h,000h,0FCh,00Ch,00Ch,000h,000h ; 170
DB          0C3h,0C6h,0CCh,0DEh,033h,066h,0CCh,00Fh ; 171
DB          0C3h,0C6h,0CCh,0DBh,037h,06Fh,0CFh,003h ; 172
DB          018h,018h,000h,018h,018h,018h,018h,000h ; 173
DB          000h,033h,066h,0CCh,066h,033h,000h,000h ; 174
DB          000h,0CCh,066h,033h,066h,0CCh,000h,000h ; 175
DB          022h,088h,022h,088h,022h,088h,022h,088h ; 176
DB          055h,0AAh,055h,0AAh,055h,0AAh,055h,0AAh ; 177
DB          0DBh,077h,0DBh,0EEh,0DBh,077h,0DBh,0EEh ; 178
DB          018h,018h,018h,018h,018h,018h,018h,018h ; 179
```

```
DB          018h,018h,018h,018h,0F8h,018h,018h,018h ; 180
DB          018h,018h,0F8h,018h,0F8h,018h,018h,018h ; 181
DB          036h,036h,036h,036h,0F6h,036h,036h,036h ; 182
DB          000h,000h,000h,000h,0FEh,036h,036h,036h ; 183
DB          000h,000h,0F8h,018h,0F8h,018h,018h,018h ; 184
DB          036h,036h,0F6h,006h,0F6h,036h,036h,036h ; 185
DB          036h,036h,036h,036h,036h,036h,036h,036h ; 186
DB          000h,000h,0FEh,006h,0F6h,036h,036h,036h ; 187
DB          036h,036h,0F6h,006h,0FEh,000h,000h,000h ; 188
DB          036h,036h,036h,036h,0FEh,000h,000h,000h ; 189
DB          018h,018h,0F8h,018h,0F8h,000h,000h,000h ; 190
DB          000h,000h,000h,000h,0F8h,018h,018h,018h ; 191
DB          018h,018h,018h,018h,01Fh,000h,000h,000h ; 192
DB          018h,018h,018h,018h,0FFh,000h,000h,000h ; 193
DB          000h,000h,000h,000h,0FFh,018h,018h,018h ; 194
DB          018h,018h,018h,018h,01Fh,018h,018h,018h ; 195
DB          000h,000h,000h,000h,0FFh,000h,000h,000h ; 196
DB          018h,018h,018h,018h,0FFh,018h,018h,018h ; 197
DB          018h,018h,01Fh,018h,01Fh,018h,018h,018h ; 198
DB          036h,036h,036h,036h,037h,036h,036h,036h ; 199
DB          036h,036h,037h,030h,03Fh,000h,000h,000h ; 200
DB          000h,000h,03Fh,030h,037h,036h,036h,036h ; 201
DB          036h,036h,0F7h,000h,0FFh,000h,000h,000h ; 202
DB          000h,000h,0FFh,000h,0F7h,036h,036h,036h ; 203
DB          036h,036h,037h,030h,037h,036h,036h,036h ; 204
DB          000h,000h,0FFh,000h,0FFh,000h,000h,000h ; 205
DB          036h,036h,0F7h,000h,0F7h,036h,036h,036h ; 206
DB          018h,018h,0FFh,000h,0FFh,000h,000h,000h ; 207
DB          036h,036h,036h,036h,0FFh,000h,000h,000h ; 208
DB          000h,000h,0FFh,000h,0FFh,018h,018h,018h ; 209
DB          000h,000h,000h,000h,0FFh,036h,036h,036h ; 210
DB          036h,036h,036h,036h,03Fh,000h,000h,000h ; 211
DB          018h,018h,01Fh,018h,01Fh,000h,000h,000h ; 212
DB          000h,000h,01Fh,018h,01Fh,018h,018h,018h ; 213
DB          000h,000h,000h,000h,03Fh,036h,036h,036h ; 214
DB          036h,036h,036h,036h,0FFh,036h,036h,036h ; 215
DB          018h,018h,0FFh,018h,0FFh,018h,018h,018h ; 216
DB          018h,018h,018h,018h,0F8h,000h,000h,000h ; 217
DB          000h,000h,000h,000h,01Fh,018h,018h,018h ; 218
DB          0FFh,0FFh,0FFh,0FFh,0FFh,0FFh,0FFh,0FFh ; 219
DB          000h,000h,000h,000h,0FFh,0FFh,0FFh,0FFh ; 220
DB          0F0h,0F0h,0F0h,0F0h,0F0h,0F0h,0F0h,0F0h ; 221
DB          00Fh,00Fh,00Fh,00Fh,00Fh,00Fh,00Fh,00Fh ; 222
DB          0FFh,0FFh,0FFh,0FFh,000h,000h,000h,000h ; 223
DB          000h,000h,076h,0DCh,0C8h,0DCh,076h,000h ; 224
DB          000h,078h,0CCh,0F8h,0CCh,0F8h,0C0h,0C0h ; 225
DB          000h,0FCh,0CCh,0C0h,0C0h,0C0h,0C0h,000h ; 226
DB          000h,0FEh,06Ch,06Ch,06Ch,06Ch,06Ch,000h ; 227
DB          0FCh,0CCh,060h,030h,060h,0CCh,0FCh,000h ; 228
```

```
        DB          000h,000h,07Eh,0D8h,0D8h,0D8h,070h,000h ; 229
        DB          000h,066h,066h,066h,066h,07Ch,060h,0C0h ; 230
        DB          000h,076h,0DCh,018h,018h,018h,018h,000h ; 231
        DB          0FCh,030h,078h,0CCh,0CCh,078h,030h,0FCh ; 232
        DB          038h,06Ch,0C6h,0FEh,0C6h,06Ch,038h,000h ; 233
        DB          038h,06Ch,0C6h,0C6h,06Ch,06Ch,0EEh,000h ; 234
        DB          01Ch,030h,018h,07Ch,0CCh,0CCh,078h,000h ; 235
        DB          000h,000h,07Eh,0DBh,0DBh,07Eh,000h,000h ; 236
        DB          006h,00Ch,07Eh,0DBh,0DBh,07Eh,060h,0C0h ; 237
        DB          038h,060h,0C0h,0F8h,0C0h,060h,038h,000h ; 238
        DB          078h,0CCh,0CCh,0CCh,0CCh,0CCh,0CCh,000h ; 239
        DB          000h,0FCh,000h,0FCh,000h,0FCh,000h,000h ; 240
        DB          030h,030h,0FCh,030h,030h,000h,0FCh,000h ; 241
        DB          060h,030h,018h,030h,060h,000h,0FCh,000h ; 242
        DB          018h,030h,060h,030h,018h,000h,0FCh,000h ; 243
        DB          00Eh,01Bh,01Bh,018h,018h,018h,018h,018h ; 244
        DB          018h,018h,018h,018h,018h,0D8h,0D8h,070h ; 245
        DB          030h,030h,000h,0FCh,000h,030h,030h,000h ; 246
        DB          000h,076h,0DCh,000h,076h,0DCh,000h,000h ; 247
        DB          038h,06Ch,06Ch,038h,000h,000h,000h,000h ; 248
        DB          000h,000h,000h,018h,018h,000h,000h,000h ; 249
        DB          000h,000h,000h,000h,018h,000h,000h,000h ; 250
        DB          00Fh,00Ch,00Ch,00Ch,0ECh,06Ch,03Ch,01Ch ; 251
        DB          078h,06Ch,06Ch,06Ch,06Ch,000h,000h,000h ; 252
        DB          070h,018h,030h,060h,078h,000h,000h,000h ; 253
        DB          000h,000h,03Ch,03Ch,03Ch,03Ch,000h,000h ; 254
        DB          000h,000h,000h,000h,000h,000h,000h,000h ; 255



;-------------------------------------------------------------------------
;
;    STEP5 - program the DAC.
;
;-------------------------------------------------------------------------

Step5:
        mov         al, 0FFh                ; Turn on PEL mask.
        mov         dx, 03c6h               ;
        out         dx, al                  ;

        mov         al, 0                   ; Set data index to zero.
        mov         dx, 3c8h                ;
        out         dx, al                  ;

        mov         dx, 3C9h                ; Set DAC write index
        mov         cx, 3 * 256             ; CX = # bytes (RGB * 256).
        mov         si, offset NewDAC       ; SI -> new DAC data.
```

```
DACloop:
            lodsb                               ; Get the byte.
            out         dx, al                  ; Set the PEL data.
            loop        DACloop                 ; Continue till CX == 0.

            ret

NewDAC      DB          00h,00h,00h,00h,00h,2ah,00h,2ah,00h,00h,2ah,2ah
            DB          2ah,00h,00h,2ah,00h,2ah,2ah,2ah,00h,2ah,2ah,2ah
            DB          00h,00h,15h,00h,00h,3fh,00h,2ah,15h,00h,2ah,3fh
            DB          2ah,00h,15h,2ah,00h,3fh,2ah,2ah,15h,2ah,2ah,3fh
            DB          00h,15h,00h,00h,15h,2ah,00h,3fh,00h,00h,3fh,2ah
            DB          2ah,15h,00h,2ah,15h,2ah,2ah,3fh,00h,2ah,3fh,2ah
            DB          00h,15h,15h,00h,15h,3fh,00h,3fh,15h,00h,3fh,3fh
            DB          2ah,15h,15h,2ah,15h,3fh,2ah,3fh,15h,2ah,3fh,3fh
            DB          15h,00h,00h,15h,00h,2ah,15h,2ah,00h,15h,2ah,2ah
            DB          3fh,00h,00h,3fh,00h,2ah,3fh,2ah,00h,3fh,2ah,2ah
            DB          15h,00h,15h,15h,00h,3fh,15h,2ah,15h,15h,2ah,3fh
            DB          3fh,00h,15h,3fh,00h,3fh,3fh,2ah,15h,3fh,2ah,3fh
            DB          15h,15h,00h,15h,15h,2ah,15h,3fh,00h,15h,3fh,2ah
            DB          3fh,15h,00h,3fh,15h,2ah,3fh,3fh,00h,3fh,3fh,2ah
            DB          15h,15h,15h,15h,15h,3fh,15h,3fh,15h,15h,3fh,3fh
            DB          3fh,15h,15h,3fh,15h,3fh,3fh,3fh,15h,3fh,3fh,3fh
            DB          00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h
```

```
        DB          00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h
        DB          00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h
        DB          00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h
        DB          00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h
        DB          00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h
        DB          00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h
        DB          00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h
        DB          00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h
        DB          00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h
        DB          00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h
        DB          00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h
        DB          00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h
        DB          00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h
        DB          00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h
        DB          00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h
        DB          00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h
        DB          00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h
        DB          00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h
        DB          00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h
        DB          00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h
        DB          00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h
        DB          00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h


;-----------------------------------------------------------------------------
;
;  STEP6 - Program Attribute controller.
;
;  Reset Index/Data Flip Flop of Attribute controller.
;  Write attributes information.
;
;-----------------------------------------------------------------------------

Step6:
        mov         dx, 3DAh                ; Set attributes register to index.
        in          al, dx                  ;

        mov         si, offset AttribTable
        mov         dx, 3C0h                ;
        mov         cx, 15h                 ; Write to 15h registers.
        mov         bx, 0h                  ; Use BX as the index counter.
@@:
        mov         ax, bx                  ; Set the next index.
        out         dx, al                  ;
        inc         bx                      ;
        lodsb                               ; Read the next byte.
        out         dx, al                  ; Write the next data.
        loop        @B
```

```
            mov         dx, 3DAh            ; Set attributes register to index.
            in          al, dx

            mov         al, 20h
            mov         dx, 3C0h
            out         dx, al
            ret


AttribTable:
            db          00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h
            db          38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh
            db          0ch, 00h, 0Fh, 00h, 00h


;-------------------------------------------------------------------------
;
;    STEP7 - Clear Vram.
;
;-------------------------------------------------------------------------

Step7:
            mov         ax, 0B800h
            mov         es, ax
            mov         cx, 4000            ; 16k Words
            xor         di, di
            mov         ax, 720h            ; normal attribute, ASCII space
            cld
            rep         stosw
            ret



; ADDITIONAL SUPPORT ROUTINES
;----------------------------

; RunTable
; This routine interprets the configuration tables that we have
; presented above.  PW is a mask bit that causes a branch from
; index/data to a simple port write of data (ie. if High bit on
; Port value is set, then branch to simple port write BR3)


br0:
            dec         si                  ; Undo last byte overrun

RunTable:
            lodsw                           ; Read next control word
            cmp         ax, -1              ; End of table?
```

```
        je          brx                 ; Exit if so

        mov         dx, ax              ; Get port in DX
        and         dh, 7fh             ; Mask off port write bit
        test        ah, 80h             ; Test if port write?
        jnz         br3                 ; Jump if so
br2:
        lodsw                           ; Read index,data
        cmp         al, -1              ; End of run
        je          br0                 ; Jump back if so
        out         dx, ax              ; Write the data
        jmp         br2                 ; Back for more
br3:
        lodsw                           ; Read data
        out         dx, al              ; Write the data
        jmp         RunTable            ; Back for more
brx:
        ret


;-------------------------------------------------------------------------
;       ReadAuxReg - Read Auxiliary Register
;
;       Entry:  AL - Register to read
;       Exit:   AL - Data read
;       Uses:   AL
;-------------------------------------------------------------------------
SPCReadAuxReg:
        push        dx                  ; Save DX
        mov         dx, 3de             ; Setup auxiliary register
        out         dx, a               ; Write index
        inc         dx                  ; Adjust to data register
        in          al, dx              ; Read data
        pop         dx                  ; Restore DX
        ret
```

```
;-------------------------------------------------------------------------------
;       WriteAuxReg - Write Auxiliary Register
;
;       Entry:  AL - Data to write
;               AH - Register to write
;       Exit:   AX - Unchanged
;       Uses:   AX
;-------------------------------------------------------------------------------
SPCWriteAuxReg:
            push        dx                  ; Save DX
            xchg        al, ah              ; Set AL=reg, AH=data
            mov         dx, 3deh            ; Setup auxiliary register
            out         dx, ax              ; Write data to reg
            xchg        al, a               ; Restore for caller
            pop         dx                  ; Restore DX
            ret


;-------------------------------------------------------------------------------
;       SPCWaitForPLLsLocked
;
;       Note: Called only at initialization time and after each power-up
;
;-------------------------------------------------------------------------------
SPCWaitForPLLsLocked:
            xor         cx, cx              ; prevent looping forever
@@:
            GETAUX      028h                ; get current PLL state
            and         al, 03h             ; keep lock bits only
            cmp         al, 03h             ; both PLLs locked?
            loopnz      @B
            ret


TextOut:
            xor         di,di
            mov         ax,0b800h
            mov         es,ax               ; es:di points to video buffer
@@:
            lodsb                           ; get a character
            test        al,al               ; end of string?
            jz          TextOutX            ; ..yes, quit
            stosb                           ; ..no, copy character into video buffer
            inc         di                  ; skip attribute
            jmp         @B                  ; do them all
TextOutX:
            ret
```

```
;-------------------------------------------------------------------------
;
;           Delay - Waste time in mS/8 ticks.
;
;-------------------------------------------------------------------------
Delay:
         mov         ah, al             ; Save beep time in AH
         in          al, 61h            ; Read system port
         and         al, 11111101b      ; Turn off speaker
         out         61h, al            ; Do it
         mov         al, ah             ; Setup wait
         call        DelayLoop          ; Waste time
         ret

DelayLoop:
         push        cx
         mov         cl, al             ; Copy mS/8 to CL

         ; Initialize timer 2 at 1000Hz.

         mov         al, 10110110b      ; Timer 2, LSB/MSB, mode 3, binary
         out         43h, al            ; Setup timer 2
         mov         ax, 1193           ; 1.1931816MHz/1000Hz=1193
         out         42h, al            ; Write LSB
         mov         al, ah             ; Get MSB
         out         42h, al            ; Write MSB

         xor         ch, ch             ; Convert to word
         shl         cx, 3              ; Multiply by 8
         jcxz        Delayx             ; Split if no time to Delay

         in          al, 61h            ; Read system port
         mov         ah, al             ; Save it
         or          al, 00000001b      ; Ensure OUT2 bit readable
         out         61h, al            ; Do it
Delay3:
         in          al, 61h            ; Get system info
         test        al, 00100000b      ; Mask timer 2 bit
         jnz         Delay3             ; Wait for trailing edge
Delay4:
         in          al, 61h            ; Get system info
         test        al, 00100000b      ; Mask timer 2 bit
         jz          Delay4             ; Wait for rising edge
         loop        Delay3             ; Do all 1mS cycles

         mov         al, ah             ; Get system port value
         out         61h, al            ; Reset original value
```

```
Delayx:
         pop         cx
         ret


MODE3    ENDP


;-------------------------------------------------------------------------------
;
; Panel info tables. All relevant registers for the panel are within one PST
; structure.
;
;-------------------------------------------------------------------------------
PST      STRUCT
         AUX_1           DB      ?
         AUX_2           DB      ?
         AUX_6           DB      ?
         AUX_7           DB      ?
         AUX_16          DB      ?
         AUX_17          DB      ?
         AUX_15          DB      ?
         CRTCB_0         DB      ?
         CRTCB_1         DB      ?
         CRTCB_6         DB      ?
         CRTCB_7         DB      ?
         CRTCB_10        DB      ?
         CRTCB_11        DB      ?
         CRTCB_12        DB      ?
         PLL0Den         DB      ?
         PLL0Num         DB      ?
         PLL1Den         DB      ?
         PLL1Num         DB      ?
PST      ENDS

aAUXIndeces     DB      1,2,6,7,16h,17h,15h
aCRTCBIndeces   DB      0,1,6,7,10h,11h,12h
```

```
;-------------------------------------------------------------------------
;       SPCLoadPanelRegs - Programs all registers for a given panel type.
;
;       Uses: Nothing
;       Note: Called only once, at initialization.
;             Panel is OFF at the time of this call.
;
;-------------------------------------------------------------------------
SPCLoadPanelRegs:
         PUSHA

         LEA         SI,LcdPanelTypeTbl

         ; for dual panels program the half frame buffer

         TEST        CS:[SI.PST.AUX_2],1    ; Q: is panel dual?
         JNZ         SHORT SPCFrameBufDone  ; N: split

         ;ECHO       SPC8110.ASM : Small panel bug fix1 ...

         ; Slow dot clock modes on panels smaller than 640 pixels may
         ; not show up correctly. Panels smaller than 640 pixels will
         ; be programmed to 640 pixels width. (There's a bug in 8110
         ; if Width(B-set) < Width(A-set). Width(A-set) is always = 640 for
         ; slow dot clock modes)

         MOV         AL,640/8
         MOV         AH,CS:[SI.PST.CRTCB_1]
         CMP         AH,AL
         JA          SHORT @F
         MOV         AH,AL
@@:
         MOV         AL,CS:[SI.PST.CRTCB_12]; AL = Height/4

         MUL         AH                     ; AX = (Height/4)*(Width/8)=(H*W)/32
         SHR         AX,1                   ; AX = (HxW)/64

         ; do mono/color adjustments

         TEST        CS:[SI.PST.AUX_2],2    ; Q: color panel
         JZ          SHORT SPCMono          ; N: do mono adjustments
         MOV         BX,AX
         ADD         AX,AX
         ADD         AX,BX                  ; AX = 3*(H*W)/64
         SUB         AX,7                   ; AX = 3*(H*W)/64-7
         JMP         SHORT SPCDoFrameLength
SPCMono:
         SUB         AX,3                   ; AX = (H*W)/64-3
```

```
SPCDoFrameLength:
            MOV         BX,AX
            PUTAUX      1Dh,BH
            PUTAUX      1Eh,BL
            MOV         AL,0Fh              ; assume page 15 and 1Mb of memory
            CALL        SPCIs1Meg
            JE          SHORT @F
            SHR         AX,1                ; make it page 7 for 512k of memory
@@:
            ; adjust the frame buffer position based on frame
            ; buffer length bits b15,b14

            TEST        BH,40h
            JZ          SHORT @F
            AND         AL,11111110b
@@:
            TEST        BH,80h
            JZ          SHORT @F
            AND         AL,11111100b
@@:
            PUTAUX      1Ch,AL              ; frame start at the very end of mem.

SPCFrameBufDone:
            MOV         CX,6                ; do 6 AUX registers
            LEA         DI,aAUXIndeces
            CLD
DoAUXRegs:
            LODS        CS:CS_BYTE          ; get value for AUX

            PUTAUX      CS:[DI],AL          ; program AUX reg
            INC         DI
            LOOP        DoAUXRegs

            ; special case AUX[15h] : modify only the skew bits

            GETAUX      15h
            AND         AL,11110011b
            MOV         AH,AL
            LODS        CS:CS_BYTE
            OR          AL,AH
            PUTAUX      15h,AL

            GETAUX      00h
            PUSH        AX                  ; save AUX[0]
            OR          AL,1
            PUTAUX      00h,AL              ; enable CRTC_B set
            MOV         CX,7                ; do 7 CRTC_B registers
            LEA         BX,aCRTCBIndeces
            CALL        LoadCRTCTable
```

```
        ; If a panel x-size is less then 640 pixels,
        ; program the CRTCB[1] with 640 pixels. This is to
        ; take care of a bug if panel size <= HDE. The smallest
        ; HDE is 40 for any IBM mode.

        ;ECHO     SPC8110.ASM : Small panel bug fix2 ...

        MOV         AL,01h
        CALL        SPCReadCRTC
        CMP         AL,640/8
        JAE         SHORT @F
        MOV         AX,0150h              ; write CRTCB[1] = 50h (80 decimal)
        CALL        SPCWriteCRTC
@@:
        POP         AX                    ; get saved AUX[0]
        PUTAUX      00h,AL                ; restore it

        POPA

        ;program    PLL
        PUSH        SI
        LEA         SI,LcdPanelTypeTbl
        LEA         SI,[SI.PST.PLL0Den]
        LODS        CS:CS_BYTE
        MOV         CH,AL
        PUTAUX      21h,AL
        LODS        CS:CS_BYTE
        MO          CL,AL
        PUTAUX      20h,AL
        LODS        CS:CS_BYTE
        PUTAUX      23h,AL
        LODS        CS:CS_BYTE
        PUTAUX      22h,AL
        POP         SI

        RET
```

```
;-------------------------------------------------------------------------
;       SPCPCIEnable - Find and enable PCI chip
;
        Entry:  Nothing
;       Exit:   Nothing
;       Uses:   Nothing
;-------------------------------------------------------------------------
SPCPCIEnable PROC NEAR
        PUSHA
        CALL          SPCFind8110         ; yes, try to find our PCI chip
        JC            PCIError            ; could not find, do nothing
        CALL          SPCPCIWriteReg4     ; enable IO & mem decoding
PCIError:
        POPA
        RET

SPCPCIEnable ENDP


;-------------------------------------------------------------------------
;       SPCIs1Meg - Test if 1 Mb of memory installed
;
;       Entry:  Nothing
;       Exit:   Z flg - Set if 1 Mb installed
;       Uses:   Nothing
;-------------------------------------------------------------------------
SPCIs1Meg:
         PUSH         AX
         GETAUX       03h
         TEST         AL,01000000b            ; single DRAM?
         POP          AX
         RET


;-------------------------------------------------------------------------
;       LoadCRTCTable - Load CRTC table
;
;       Entry:  BX - Offset of aCRTCBIndeces
                CX - Number of registers
;       Exit:   AL - Data to write
;               AH - Register to write
;       Uses:   AX, BX, CS
;-------------------------------------------------------------------------
LoadCRTCTable PROC NEAR
@@:      LODS         CS:CS_BYTE          get value for CRTC in AL
         MOV          AH,CS:[BX]          get index in AH
         CALL         SPCWriteCRTC
         INC          BX
```

```
LOOP     @B
         RET
LoadCRTCTable ENDP


;-------------------------------------------------------------------------------
;       SPCReadCRTC - Read CRTC Register
;
;       Entry:  AL - register to read
;       Exit:   AL - data read
;       Uses:   AL
;-------------------------------------------------------------------------------
SPCReadCRTC:
         PUSH        DX                  ; Save DX
         PUSH        AX                  ; Save AX
         MOV         AX,3D4H             ; CRTC at 3D4H
         MOV         DX,AX
         POP         AX                  ; Restore AX
         OUT         DX,AL               ; Write index
         INC         DX                  ; Adjust to data register
         IN          AL,DX               ; Read data
         POP         DX                  ; Restore DX
         RET


;-------------------------------------------------------------------------------
;       SPCWriteCRTC - Write CRTC Register
;
;       Entry:  AL - Data to write
;               AH - Register to write
;       Exit:   AX - Unchanged
;       Uses:   AX
;-------------------------------------------------------------------------------
SPCWriteCRTC:
         PUSH        DX                  ; Save DX
         XCHG        AL,AH               ; Set AL=reg, AH=data
         PUSH        AX                  ; Save AX
         MOV         AX,3D4H             ; CRTC at 3D4H
         MO          DX,AX
         POP         AX                  ; Restore AX
         OUT         DX,AX               ; Write data to reg
         XCHG        AL,AH               ; Restore for caller
         POP         DX                  ; Restore DX
         RET
```

```
;-------------------------------------------------------------------------
;       SPCFind8110 - Find SPC8110 Chip
;
;       Entry:  Nothing
;       Exit:   BH - bus number
;               BL - devnum
;               C Flag - set if chip not found
;       Uses:   AX, CX, DX, SI
;-------------------------------------------------------------------------
SPCFind     8110        PROC NEAR
            MOV         AH,0B1H
            MOV         AL,2
            XOR         SI,SI
            MOV         CX,DEVICE_ID        ; CX = DEVID
            MOV         DX,VENDOR_ID        ; DX = VENDOR ID
            INT         1AH
            JC          SHORT ERRDEVICE
            TEST        AH,AH
            JNZ         SHORT ERRDEVICE
            CLC
            RET
ERRDEVICE:
            STC
            RET
SPCFind8110             ENDP


;-------------------------------------------------------------------------
;     SPCPCIWriteReg4 - Enable memory and IO for PCI system
;       Entry : BH - bus number
;               BL - devnum
;       Exit:   Nothing
;       Uses:   AX, CX, DI
;-------------------------------------------------------------------------
SPCPCIWriteReg4         PROC NEAR
            MOV         AX,0B10BH
            MOV         DI,4                    ; offset
            MOV         CX,3                    ; R[4] = 3 (decode memory & IO)
            INT         1AH
            RET
SPCPCIWriteReg4 ENDP

INCLUDE lcd.dat
; (See LCD.DAT on page 65)

            END         MODE3
```

Initializing The SPC8110 In Mode 12h Without A BIOS

The purpose of this document is to demonstrate how to initialize the SPC8110 into a mode similar to IBM's Standard VGA mode 12h by supplying sample code. The steps involved in this are:

- Enabling the SPC8110
- Setting up the SPC8110 panel specific registers
- Setting up the Standard VGA registers for Mode 12h
- Setting up the attribute registers
- Initializing the RAMDAC values. The overlay RAMDAC (sprite) is not initialized.
- Clearing Video Memory

This program assumes 512k of 80ns DRAM. If there are DRAM problems, modify the values for AUX[03]. If using 100ns DRAMS, program the PLL2 to 33Mhz.

The resulting mode is not BIOS supported, so no DOS text will work, however, routines developed to write pixels without the BIOS into mode 12h video memory will work correctly.

```
;-------------------------------------------------------------------------
;
; Build instructions: using MASM 6.11
;
;  - Make your changes
;  - Assemble          --  masm MODE12.asm;
;  - Link              --  link MODE12;
;  - Make a .COM file  --  exe2bin MODE12.exe MODE12.com
;
;-------------------------------------------------------------------------

.286
.MODEL TINY

PciSupported    EQU     1       ; comment off this line to skip PCI enable
PW              EQU     8000h   ; Port write (RunTable)
G11             EQU     11000000b
VENDOR_ID       EQU     10F4h
DEVICE_ID       EQU     8110h

GETAUX  MACRO   reg             ; Read a SPCxxxx AUX register
        mov     al,reg
        call    SPCReadAuxReg
        ENDM


PUTAUX  MACRO   reg,data        ; Write a SPCxxxx AUX register
        mov     al,data
        mov     ah,reg
        call    SPCWriteAuxReg
        ENDM
```

```
.CODE

CS_BYTE         LABEL   BYTE    ; Local byte reference
CS_WORD         LABEL   WORD    ; Local word reference

        ORG     100h
MODE12  PROC    NEAR
        cld                     ; Set the direction flag.
        mov     ax, cs          ; Ensure that DS and ES point to this sement.
        mov     ds, ax          ;
        mov     es, ax          ;
        cli                     ; Interupts off while changing the stack.
        mov     ss, ax          ;
        mov     sp, 0FFFFh      ;
        sti                     ; Stack change done - restore interupts.

        call    Step1           ; Enable SPC8110 chip.

        ; at this point, all the registers are accessible, we need to
        ; program Auxilliary registers according to panel size, memory,
        ; and some preferences...

        call    Step2           ; Setup for the panel and enviroment.

        ; At this point all non-standard registers are fully initialized
        ; and LCD is activated. The following is standard VGA stuff
        ; that would run without any change on any IBM-compatible VGA chip.

        call    Step3           ; Setup the standard VGA registers.
        call    Step4           ; Setup the attribute registers.
        call    Step5           ; Blow the DAC with mode 12 colors.
        call    Step6           ; show color stripes...

        ;  You should now be running as MODE 12 without BIOS support.

        mov     ax, 4C00h       ; DOS - Exit with Return Code.
        int     21h



;-------------------------------------------------------------------------
;
;  STEP1 - Before ANY ports can be accessed, the SPC8110 chip must be
;          enabled.
;
;-------------------------------------------------------------------------

Step1:
            IFDEF   PciSupported
            ECHO    PCI Support Enabled
            call    SPCPCIEnable
            ENDIF
       mov     dx,3c3h
```

```
        mov     al,1
        out     dx,al
        ret


;-------------------------------------------------------------------------
;
;  STEP2 - This sets the chip up for the correct panel and for
;          a proper working environment.
;
;
;-------------------------------------------------------------------------

Step2:
        mov     si, OFFSET InitTheWorldTbl
        call    RunTable
        call    SPCWaitForPLLsLocked
        call    SetupDisplay
        ret


InitTheWorldTbl :
        DW       PW+03C2h,67h   ; default card to 3Dx, 28Mhz Clock
        DW       03DEh          ; program the Aux. registers

        ; turn on the LCD as the very end...

;;      DB       00h,00000000b  ; clear register

        DB       01H,01000010b  ; autocenter, 2Hz blink
        DB       02H,01001000b  ; 8bit mono dual LCD panel
        DB       03H,01001000b  ; Single DRAM, (Wait state=Turbo), 8ms symetrical
        DB       04H,00000000b  ; clear scratch pad
        DB       05H,00000000b  ; disable memory paging, HIREZ256 etc.
        DB       06H,00100000b  ; LCD WF count register = 20h
        DB       07H,00000000b  ; LCD Display register

;;      DB       08H,00000000b  ; RO
        DB       0AH,00000000b  ; clear scratch pad
        DB       0BH,00000000b  ; clear scratch pad
;;      DB       0CH,00000000b  ; treat as RO !
;;      DB       0DH,00000000b  ; RO
;;      DB       0FH,00000000b  ; RO

        ; set default values for PD registers

        DB       10H,00000000b  ; polarity PS1 & PS4 low, no timer
        DB       11H,00000011b  ; PS 1: GreyScale, Slow Clock,H&V sync active
        DB       12H,00000011b  ; PS 2: GreyScale, Slow Clock,H&V sync active
        DB       13H,10110111b  ; PS 3: MClk Off,LCDBack off,DPMS
```

```
Suspend,GS,DisplayOff,
        DB      14H,10110111b  ; PS 4: MClk Off,LCDBack off,DPMS
Suspend,GS,DisplayOff,
        DB      15H,00001100b  ; PS Disable PS1&PS4,skew XX
        DB      16H,01001111b  ; PS Skew Register
        DB      17H,00000001b  ; PS SkewA b3=1

        DB      1AH,00000000b  ; video Buffer High
        DB      1BH,00000000b  ; video Buffer Low

        ; Frame buffer in the last page, programmed for mono dual 640x480
        ; panels

        DB      1CH,00001111b  ; Frame Start (Dual LCD only)
        DB      1DH,00010010b  ; Frame Length High
        DB      1EH,10111101b  ; Frame Length Low
        DB      1FH,00000000b  ; Frame Buffer enable + High Request


        ; PLL1 set 0 programmed to 14.318 MHz, done elsewhere

        ;DB      21H,1h        ; PLL1 Set0 Denominator
        ;DB      20H,1h        ; PLL1 Set0 Numerator

        ; PLL1 set 1 programmed to 9.55 MHz, done elsewhere

        ;DB      23H,03h       ; PLL1 Set1 Denominator
        ;DB      22H,02h       ; PLL1 Set1 Numerator

        ; PLL2 programmed for 80ns DRAM (40MHz Memory clock)

        DB      27H,04h        ; PLL2 Denominator (must be first)
        DB      26H,0bh        ; PLL2 Numerator

        DB      28H,00000000b  ; clear any PLL override bits

        ;; 2a..33 are N/A

        DB      34H,00000000b  ; disable GENIO outputs

        DB      38H,04h        ; FIFO
        DB      39H,14h        ; FIFO
        DB      3AH,04h        ; FIFO
        DB      3BH,04h        ; FIFO
        DB      3CH,00h        ; FIFO

        DB      -1             ; end of Aux registers
        DW      -1             ; end of the register table
```

```
;-------------------------------------------------------------------------------
;       SetupDisplay - Setup LCD display operation.
;
;       Entry:  n/a
;       Exit:   n/a
;       Uses:   AX,BX,CX,DX,SI,DI
;-------------------------------------------------------------------------------
SetupDisplay:

        CALL    SPCLoadPanelRegs

        PUTAUX  00h,20h         ; turn the panel on

SetupDisplayX:
        RET


;-------------------------------------------------------------------------------
;
;  STEP3 - Use tables to program the standard VGA registers.
;
;-------------------------------------------------------------------------------

Step3:
        mov     si, offset VGAregs
        call    RunTable
        ret

VGAregs:
        DW      PW+3C2h,0E3h      ; Misc. Output.

        DW       3C4h             ; Sequencer.
        DB       00h,00h
        DB       01h,01h
        DB       02h,0fh
        DB       03h,00h
        DB       04h,06h
        DB       00h,03h
        DB       -1

        DW       3D4h             ; CRTC
        DB       11h,0Ch
        DB       00h,05Fh
        DB       01h,04Fh
        DB       02h,050h
        DB       03h,082h
        DB       04h,054h
        DB       05h,080h
        DB       06h,00Bh
        DB       07h,03Eh
        DB       08h,000h
        DB       09h,040h
        DB       0Ah,000h
```

```
DB      0Bh,000h
DB      0Ch,000h
DB      0Dh,000h
DB      0Eh,000h
DB      0Fh,000h
DB      10h,0EAh
DB      11h,08Ch
DB      12h,0DFh
DB      13h,028h
DB      14h,000h
DB      15h,0E7h
DB      16h,004h
DB      17h,0E3h
DB      18h,0FFh

; The following are sprite specific registers.
; Note the low byte values are written before high byte values.
; ( writing the high byte latches both low+high byte )

DB      38h,00h
DB      37h,00h
DB      36h,00h
DB      35h,00h
DB      34h,00h
DB      33h,00h
DB      32h,00h
DB      31h,00h
DB      30h,00h
DB      -1

DW      3CEh                    ; Graphics.
DB      00h,000h
DB      01h,000h
DB      02h,000h
DB      03h,000h
DB      04h,000h
DB      05h,000h
DB      06h,005h
DB      07h,00Fh
DB      08h,0FFh
DB      -1
DW      -1
```

```
;-------------------------------------------------------------------------
;
;  STEP4 - Program Attribute controller.
;
;  Reset Index/Data Flip Flop of Attribute controller.
;  Write attributes information.
;
;-------------------------------------------------------------------------

Step4:
        mov     dx, 3DAh        ; Set attributes register to index.
        in      al, dx          ;

        mov     si, offset AttribTable
        mov     dx, 3C0h        ;
        mov     cx, 14h         ; Write to 14h registers.
        mov     bx, 0h          ; Use BX as the index counter.
@@:
        mov     ax, bx          ; Set the next index.
        out     dx, al          ;
        inc     bx              ;
        lodsb                   ; Read the next byte.
        out     dx, al          ; Write the next data.
        loop    @B

        mov     al, 20h         ; al = enable palette bit
        mov     dx, 3C0h
        out     dx, al          ; enable palette
        ret

AttribTable:
        db      00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h
        db      38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh
        db      01h, 00h, 0Fh, 00h, 00h

;-------------------------------------------------------------------------
;
;    STEP5 - program the DAC.
;
;-------------------------------------------------------------------------

Step5:
        mov     al, 0FFh          ; Turn on PEL mask.
        mov     dx, 03c6h         ;
        out     dx, al            ;

        xor     al, al            ; Set data index to zero.
        mov     dx, 3c8h          ;
        out     dx, al            ;

        mov     dx, 3C9h          ; Set DAC write index
        mov     cx, 3 * 256       ; CX = # bytes (RGB * 256).
        mov     si, offset NewDAC ; SI -> new DAC data.
```

```
DACloop:
        lodsb                   ; Get the byte.
        out     dx, al          ; Set the PEL data.
        loop    DACloop         ; Continue till CX == 0.

        ret

NewDAC  DB      00h, 00h, 00h, 00h, 00h, 2ah, 00h, 2ah, 00h, 00h, 2ah, 2ah
        DB      2ah, 00h, 00h, 2ah, 00h, 2ah, 2ah, 2ah, 00h, 2ah, 2ah, 2ah
        DB      00h, 00h, 15h, 00h, 00h, 3fh, 00h, 2ah, 15h, 00h, 2ah, 3fh
        DB      2ah, 00h, 15h, 2ah, 00h, 3fh, 2ah, 2ah, 15h, 2ah, 2ah, 3fh
        DB      00h, 15h, 00h, 00h, 15h, 2ah, 00h, 3fh, 00h, 00h, 3fh, 2ah
        DB      2ah, 15h, 00h, 2ah, 15h, 2ah, 2ah, 3fh, 00h, 2ah, 3fh, 2ah
        DB      00h, 15h, 15h, 00h, 15h, 3fh, 00h, 3fh, 15h, 00h, 3fh, 3fh
        DB      2ah, 15h, 15h, 2ah, 15h, 3fh, 2ah, 3fh, 15h, 2ah, 3fh, 3fh
        DB      15h, 00h, 00h, 15h, 00h, 2ah, 15h, 2ah, 00h, 15h, 2ah, 2ah
        DB      3fh, 00h, 00h, 3fh, 00h, 2ah, 3fh, 2ah, 00h, 3fh, 2ah, 2ah
        DB      15h, 00h, 15h, 15h, 00h, 3fh, 15h, 2ah, 15h, 15h, 2ah, 3fh
        DB      3fh, 00h, 15h, 3fh, 00h, 3fh, 3fh, 2ah, 15h, 3fh, 2ah, 3fh
        DB      15h, 15h, 00h, 15h, 15h, 2ah, 15h, 3fh, 00h, 15h, 3fh, 2ah
        DB      3fh, 15h, 00h, 3fh, 15h, 2ah, 3fh, 3fh, 00h, 3fh, 3fh, 2ah
        DB      15h, 15h, 15h, 15h, 15h, 3fh, 15h, 3fh, 15h, 15h, 3fh, 3fh
        DB      3fh, 15h, 15h, 3fh, 15h, 3fh, 3fh, 3fh, 15h, 3fh, 3fh, 3fh
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
```

```
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
        DB      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h

;-------------------------------------------------------------------------
;
;    STEP6 - just to get something on the
;            screen to show we are alive....
;
;-------------------------------------------------------------------------

Step6:
        cld
        mov     ax, 0A000h
        mov     es, ax

        mov     dx, 3ceh
        mov     ax, 0F01h
        out     dx, ax

        mov     bx, 16
        xor     di, di
        xor     ax, ax
@@:
        out     dx, ax
        mov     cx, 80*(480/16) ; one line = 80 bytes
        rep     stosb
        inc     ah              ; bump color
        dec     bx
        jnz     @B

        mov     dx, 3ceh
        mov     ax, 0001h
        out     dx, ax

        ret
```

```
; ADDITIONAL SUPPORT ROUTINES
;---------------------------
;-------------------------------------------------------------------------
;       RunTable - Initialize registers from table
;
;       Entry:  DS:SI - Pointer to register table
;
;       Exit:   n/a
;       Uses:   AX,DX,SI
;-------------------------------------------------------------------------

BR0:
        DEC     SI                      ; Undo last byte overrun

RunTable:
        LODSW                           ; Read next control word
        CMP     AX,-1                   ; End of table?
        JE      BRX                     ; Exit if so

        MOV     DX,AX                   ; Get port in DX
        AND     DH,7Fh                  ; Mask off port write bit
        TEST    AH,80h                  ; Test if port write?
        JNZ     BR3                     ; Jump if so
BR2:
        LODSW                           ; Read index,data
        CMP     AL,-1                   ; End of run
        JE      BR0                     ; Jump back if so
        OUT     DX,AX                   ; Write the data
        JMP     BR2                     ; Back for more
BR3:
        LODSW                           ; Read data
        OUT     DX,AL                   ; Write the data
        JMP     RunTable                ; Back for more
BRX:
        RET


;-------------------------------------------------------------------------
;       ReadAuxReg - Read Auxiliary Register
;
;       Entry:  AL - Register to read
;       Exit:   AL - Data read
;       Uses:   AL
;-------------------------------------------------------------------------
SPCReadAuxReg:
        PUSH    DX                      ; Save DX
        MOV     DX,3DEh                 ; Setup auxiliary register
        OUT     DX,AL                   ; Write index
        INC     DX                      ; Adjust to data register
        IN      AL,DX                   ; Read data
        POP     DX                      ; Restore DX
        RET
```

```
;-------------------------------------------------------------------------------
;       WriteAuxReg - Write Auxiliary Register
;
;       Entry:  AL - Data to write
;               AH - Register to write
;       Exit:   AX - Unchanged
;       Uses:   AX
;-------------------------------------------------------------------------------
SPCWriteAuxReg:
        PUSH    DX                      ; Save DX
        XCHG    AL,AH                   ; Set AL=reg, AH=data
        MOV     DX,3DEh                 ; Setup auxiliary register
        OUT     DX,AX                   ; Write data to reg
        XCHG    AL,AH                   ; Restore for caller
        POP     DX                      ; Restore DX
        RET


;-------------------------------------------------------------------------------
;       SPCWaitForPLLsLocked
;
;       Note: Called only at initialization time and after each power-up
;
;-------------------------------------------------------------------------------
SPCWaitForPLLsLocked:
        XOR     CX,CX                   ; prevent looping forever
@@:
        GETAUX  028h                    ; get current PLL state
        AND     AL,03h                  ; keep lock bits only
        CMP     AL,03h                  ; both PLLs locked?
        LOOPNZ  @B
        RET

MODE12  ENDP
```

```
;-------------------------------------------------------------------------
;
; Panel info tables. All relevant registers for the panel are within one PST
; structure.
;
;-------------------------------------------------------------------------
PST     STRUCT
        AUX_1           DB      ?
        AUX_2           DB      ?
        AUX_6           DB      ?
        AUX_7           DB      ?
        AUX_16          DB      ?
        AUX_17          DB      ?
        AUX_15          DB      ?
        CRTCB_0         DB      ?
        CRTCB_1         DB      ?
        CRTCB_6         DB      ?
        CRTCB_7         DB      ?
        CRTCB_10        DB      ?
        CRTCB_11        DB      ?
        CRTCB_12        DB      ?
        PLL0Den         DB      ?
        PLL0Num         DB      ?
        PLL1Den         DB      ?
        PLL1Num         DB      ?
PST     ENDS

aAUXIndeces     DB      1,2,6,7,16h,17h,15h
aCRTCBIndeces   DB      0,1,6,7,10h,11h,12h


;-------------------------------------------------------------------------
;       SPCLoadPanelRegs - Programs all registers for a given panel type.
;
;       Uses: Nothing
;       Note: Called only once, at initialization.
;             Panel is OFF at the time of this call.
;
;-------------------------------------------------------------------------
SPCLoadPanelRegs:
        PUSHA

        LEA     SI,LcdPanelTypeTbl

push si
                ;051297
        LEA     SI,[SI.PST.PLL0Den]
        LODS    CS:CS_BYTE
        MOV     CH,AL
        PUTAUX  21h,AL
        LODS    CS:CS_BYTE
        MOV     CL,AL
        PUTAUX  20h,AL
```

```
        LODS    CS:CS_BYTE
        PUTAUX  23h,AL
        LODS    CS:CS_BYTE
        PUTAUX  22h,AL
pop si

        ; for dual panels program the half frame buffer

        TEST    CS:[SI.PST.AUX_2],1     ; Q: is panel dual?
        JNZ     SHORT SPCFrameBufDone   ;    N: split

        ;ECHO   SPC8110.ASM : Small panel bug fix1 ...

        ; Slow dot clock modes on panels smaller than 640 pixels may
        ; not show up correctly. Panels smaller than 640 pixels will
        ; be programmed to 640 pixels width. (There's a bug in 8110
        ; if Width(B-set) < Width(A-set). Width(A-set) is always = 640 for
        ; slow dot clock modes)

        MOV     AL,640/8
        MOV     AH,CS:[SI.PST.CRTCB_1]
        CMP     AH,AL
        JA      SHORT @F
        MOV     AH,AL
@@:
        MOV     AL,CS:[SI.PST.CRTCB_12] ; AL = Height/4

        MUL     AH                      ; AX = (Height/4)*(Width/8)=(H*W)/32
        SHR     AX,1                    ; AX = (HxW)/64

        ; do mono/color adjustments

        TEST    CS:[SI.PST.AUX_2],2     ; Q: color panel
        JZ      SHORT SPCMono           ;    N: do mono adjustments
        MOV     BX,AX
        ADD     AX,AX
        ADD     AX,BX                   ; AX = 3*(H*W)/64
        SUB     AX,7                    ; AX = 3*(H*W)/64-7
        JMP     SHORT SPCDoFrameLength
SPCMono:
        SUB     AX,3                    ; AX = (H*W)/64-3
SPCDoFrameLength:
        MOV     BX,AX
        PUTAUX  1Dh,BH
        PUTAUX  1Eh,BL
        MOV     AL,0Fh                  ; assume page 15 and 1Mb of memory
        CALL    SPCIs1Meg
        JE      SHORT @F
        SHR     AX,1                    ; make it page 7 for 512k of memory
```

```
@@:
        ; adjust the frame buffer position based on frame
        ; buffer length bits b15,b14

        TEST    BH,40h
        JZ      SHORT @F
        AND     AL,11111110b
@@:
        TEST    BH,80h
        JZ      SHORT @F
        AND     AL,11111100b
@@:
        PUTAUX  1Ch,AL                  ; frame start at the very end of mem.

SPCFrameBufDone:
        MOV     CX,6                    ; do 6 AUX registers
        LEA     DI,aAUXIndeces
        CLD
DoAUXRegs:
        LODS    CS:CS_BYTE              ; get value for AUX

        PUTAUX  CS:[DI],AL              ; program AUX reg
        INC     DI
        LOOP    DoAUXRegs

        ; special case AUX[15h] : modify only the skew bits

        GETAUX  15h
        AND     AL,11110011b
        MOV     AH,AL
        LODS    CS:CS_BYTE
        OR      AL,AH
        PUTAUX  15h,AL

        GETAUX  00h
        PUSH    AX                      ; save AUX[0]
        OR      AL,1
        PUTAUX  00h,AL                  ; enable CRTC_B set
        MOV     CX,7                    ; do 7 CRTC_B registers
        LEA     BX,aCRTCBIndeces
        CALL    LoadCRTCTable

        ; If a panel x-size is less then 640 pixels,
        ; program the CRTCB[1] with 640 pixels. This is to
        ; take care of a bug if panel size <= HDE. The smallest
        ; HDE is 40 for any IBM mode.

        ;ECHO     SPC8110.ASM : Small panel bug fix2 ...

        MOV     AL,01h
        CALL    SPCReadCRTC
        CMP     AL,640/8
        JAE     SHORT @F
```

```
        MOV     AX,0150h                ; write CRTCB[1] = 50h (80 decimal)
        CALL    SPCWriteCRTC
@@:
        POP     AX                      ; get saved AUX[0]
        PUTAUX  00h,AL                  ; restore it

        POPA
        RET


;-------------------------------------------------------------------------
;       SPCPCIEnable - Find and enable PCI chip
;
;       Entry:  Nothing
;       Exit:   Nothing
;       Uses:   Nothing
;-------------------------------------------------------------------------
SPCPCIEnable PROC NEAR
        PUSHA
        CALL    SPCFind8110             ; yes, try to find our PCI chip
        JC      PCIError                ; could not find, do nothing
        CALL    SPCPCIWriteReg4         ; enable IO & mem decoding
PCIError:
        POPA
        RET

SPCPCIEnable ENDP


;-------------------------------------------------------------------------
;       SPCIs1Meg - Test if 1 Mb of memory installed
;
;       Entry:  Nothing
;       Exit:   Z flg - Set if 1 Mb installed
;       Uses:   Nothing
;-------------------------------------------------------------------------
SPCIs1Meg:
        PUSH    AX
        GETAUX  03h
        TEST    AL,01000000b            ; single DRAM?
        POP     AX
        RET
```

```
;-------------------------------------------------------------------------------
;       LoadCRTCTable - Load CRTC table
;
;       Entry:  BX - Offset of aCRTCBIndeces
;               CX - Number of registers
;       Exit:   AL - Data to write
;               AH - Register to write
;       Uses:   AX, BX, CS
;-------------------------------------------------------------------------------
LoadCRTCTable PROC NEAR
@@:         LODS        CS:CS_BYTE  ; get value for CRTC in AL
            MOV         AH,CS:[BX]  ; get index in AH
            CALL        SPCWriteCRTC
            INC         BX
            LOOP        @B
            RET
LoadCRTCTable ENDP


;-------------------------------------------------------------------------------
;       SPCReadCRTC - Read CRT Register
;
;       Entry:  AL - Register to read
;       Exit:   AL - Data read
;       Uses:   AL
;-------------------------------------------------------------------------------
SPCReadCRTC:
            PUSH    DX              ; Save DX
            PUSH    AX              ; Save AX
            MOV     AX,3D4H         ; CRTC at 3D4H
            MOV     DX,AX
            POP     AX              ; Restore AX
            OUT     DX,AL           ; Write index
            INC     DX              ; Adjust to data register
            IN      AL,DX           ; Read data
            POP     DX              ; Restore DX
            RET


;-------------------------------------------------------------------------------
;       SPCWriteCRTC - Write CRTC Register
;
;       Entry:  AL - Data to write
;               AH - Register to write
;       Exit:   AX - Unchanged
;       Uses:   AX
;-------------------------------------------------------------------------------
SPCWriteCRTC:
            PUSH    DX              ; Save DX
            XCHG    AL,AH           ; Set AL=reg, AH=data
            PUSH    AX              ; Save AX
            MOV     AX,3D4H         ; CRTC at 3D4H
            MOV     DX,AX
```

```
            POP     AX                  ; Restore AX
            OUT     DX,AX               ; Write data to reg
            XCHG    AL,AH               ; Restore for caller
            POP     DX                  ; Restore DX
            RET

;-------------------------------------------------------------------------------
;       SPCFind8110 - Find SPC8110 Chip
;
;       Entry:  Nothing
;       Exit:   BH - bus number
;               BL - devnum
;               C Flag - set if chip not found
;       Uses:   AX, CX, DX, SI
;-------------------------------------------------------------------------------
SPCFind8110     PROC NEAR
        MOV     AH,0B1H
        MOV     AL,2
        XOR     SI,SI
        MOV     CX,DEVICE_ID            ; CX = DEVID
        MOV     DX,VENDOR_ID            ; DX = VENDOR ID
        INT     1AH
        JC      SHORT ERRDEVICE
        TEST    AH,AH
        JNZ     SHORT ERRDEVICE
        CLC
        RET
ERRDEVICE:
        STC
        RET
SPCFind8110     ENDP

;-------------------------------------------------------------------------------
;       SPCPCIWriteReg4 - Enable memoey and IO for PCI system
;
;       Entry : BH - bus number
;               BL - devnum
;       Exit:   Nothing
;       Uses:   AX, CX, DI
;
;-------------------------------------------------------------------------------
SPCPCIWriteReg4 PROC NEAR
        MOV     AX,0B10BH
        MOV     DI,4                    ; offset
        MOV     CX,3                    ; R[4] = 3 (decode memory & IO)
        INT     1AH
        RET
SPCPCIWriteReg4 ENDP
INCLUDE LCD.DAT
;(See LCD.DAT on page 65)

        END     MODE12
```

Initializing The SPC8110 In Mode 101h Without A BIOS

The purpose of this document is to demonstrate how to initialize the SPC8110 into a mode similar to VESA SVGA mode 101h (640x480x256 colors) by supplying sample code. The steps involved are:

- Enabling the SPC8110
- Setting up the SPC8110 panel specific registers
- Setting up the Standard VGA registers for Mode 101h
- Setting up the attribute registers
- Initializing the RAMDAC values to values used for Mode 13h/101h
- Initializing Video Memory
- It assumes 512K bytes of DRAM (80ns, symmetrical, single chip)

Note
The value of AUX[03h] may need adjustment for different types of DRAM

The resulting mode is not BIOS supported, so no DOS text will work, however, routines developed to write pixels without the BIOS into mode 101h video memory will work correctly. For more information on paging for this mode, please refer to the SPC8110 Hardware Functional Specification, or refer to USING THE SPC8110 DISPLAY MEMORY on page 134.

```
;-------------------------------------------------------------------------
;
; Build instructions: using MASM 6.11
;
;  - Make your changes
;  - Assemble          --  masm MODE101.asm;
;  - Link              --  link MODE101;
;  - Make a .COM file  --  exe2bin MODE101.exe MODE101.com
;
;;-------------------------------------------------------------------------

.286
.MODEL TINY
PciSupported    EQU     1          ; comment off this line to skip PCI enable
PW              EQU     8000h      ; Port write (RunTable)
G11             EQU     11000000b
VENDOR_ID       EQU     10F4h
DEVICE_ID       EQU     8110h

GETAUX          MACRO   reg         ; Read a SPCxxxx AUX register
                mov     al,reg
                call    SPCReadAuxReg
        ENDM


PUTAUX  MACRO   reg,data            ; Write a SPCxxxx AUX register
        mov     al,data
        mov     ah,reg
        call    SPCWriteAuxReg
        ENDM
```

```
.CODE
CS_BYTE         LABEL   BYTE    ; Local byte reference

        ORG     100h
MODE101 PROC    NEAR
        cld                     ; Set the direction flag.
        mov     ax, cs          ; Ensure that DS and ES point to this sement.
        mov     ds, ax          ;
        mov     es, ax          ;
        cli                     ; Interupts off while changing the stack.
        mov     ss, ax          ;
        mov     sp, 0FFFFh      ;
        sti                     ; Stack change done - restore interupts.

        call    Step1           ; Enable the SPC8110 chip.

        ; at this point, all the registers are accessible, we need to
        ; program Auxilliary registers according to panel size, memory,
        ; and some preferences...

        call    Step2           ; Setup for the panel and enviroment.

        ; At this point all non-standard registers are fully initialized
        ; and LCD is activated. The following is standard VGA stuff
        ; that would run without any change on any IBM-compatible VGA chip.

        call    Step3           ; Setup the standard VGA registers.
        call    Step4           ; Blow the DAC with mode 13h colors.
        call    Step5           ; Setup the attribute registers.

        ; You should now be running as MODE 101 without BIOS support.

        call    Step6           ; show something on the panel

        ; terminate the program

        mov     ax, 4C00h       ; DOS - Exit with Return Code.
        int     21h
```

```
;-----------------------------------------------------------------------
;
;  STEP1 - Before ANY ports can be accessed, the SPC8110 chip must be
;          enabled. (For PCI systems, the Memory & IO decode must be
;          enabled prior to writing to 3c3h)
;
;-----------------------------------------------------------------------

Step1:
         IFDEF    PciSupported
         ECHO     PCI Support Enabled
         call     SPCPCIEnable
         ENDIF
         mov     dx, 3c3h
         mov     al, 1
         out     dx, al
         ret

;-----------------------------------------------------------------------
;
;  STEP2 - This sets the chip up for the correct panel and for
;          a proper working environment.
;
;;----------------------------------------------------------------------

Step2:
        mov     si, OFFSET InitTheWorldTbl
        call    RunTable
        call    SPCWaitForPLLsLocked
        call    SetupDisplay
        ret

InitTheWorldTbl :
        DW       PW+03C2h,067h  ; default card to 3Dx
        DW       03DEh          ; program the Aux. registers

;;      DB       00h,00110000b  ; program as the last thing

        DB       01H,01000010b  ; autocenter, 2Hz blink
        DB       02H,01001110b  ; 16bit color dual LCD panel
        DB       03H,01001000b  ; single DRAM (Wait state=Turbo)
        DB       04H,00000000b  ; clear scratch pad
        DB       05H,10010001b  ; memory paging, HIREZ256 etc.
        DB       06H,00100000b  ; LCD WF count register = 20h
        DB       07H,00000000b  ; LCD Display register

;;      DB       08H,00000000b  ; RO
        DB       0AH,00000000b  ; clear scratch pad
        DB       0BH,00000000b  ; clear scratch pad
;;      DB       0CH,00000000b  ; RO
;;      DB       0DH,00000000b  ; RO
;;      DB       0FH,00000000b  ; RO
```

```
; set default values for PD registers

DB       10H,00000000b  ; polarity PS1 & PS4 low, no timer
DB       11H,00000011b  ; PS 1: GreyScale, Slow Clock,H&V sync active
DB       12H,00000011b  ; PS 2: GreyScale, Slow Clock,H&V sync active
DB       13H,10110111b  ; PS 3: MClk Off,LCDBack off,DPMS Suspend,
                        ;       GS,DisplayOff,
DB       14H,10110111b  ; PS 4: MClk Off,LCDBack off,DPMS Suspend,
                        ;       GS,DisplayOff,
DB       15H,00001100b  ; PS Disable PS1&PS4, skew XX
DB       16H,01001111b  ; PS Skew Register
DB       17H,00000001b  ; PS SkewA b3=1

DB       1AH,00000000b  ; Video Buffer High
DB       1BH,00000000b  ; Video Buffer Low

; Frame buffer in the last page, programmed for color dual 800x600
; panels

DB       1CH,00001111b  ; Frame Start
DB       1DH,00111000b  ; Frame Length High
DB       1EH,00111001b  ; Frame Length Low
DB       1FH,00000000b  ; Frame Buffer enable

; PLL1 set 0,1,2 : note the order of the registers is reversed

;DB      21H,00000001b  ; PLL1 Set0 Denominator
;DB      20H,00000001b  ; PLL1 Set0 Numerator
;DB      23H,00000011b  ; PLL1 Set1 Denominator
;DB      22H,00000010b  ; PLL1 Set1 Numerator
DB       25H,00000001b  ; PLL1 Set2 Denominator
DB       24H,00000001b  ; PLL1 Set2 Numerator

; PLL2 programmed for 80ns DRAM (40MHz Memory clock)

DB       27H,00000100b  ; PLL2 Denominator (must be first)
DB       26H,00001011b  ; PLL2 Numerator
DB       28H,00000000b  ; no PLL Status override

;; 2a..33 are N/A

DB       34H,00000000b  ; disable GENIO outputs

; FIFOs ...

DB       38H,00010100b  ; FIFO
DB       39H,00010100b  ; FIFO
DB       3AH,00000100b  ; FIFO
DB       3BH,00000100b  ; FIFO
DB       3CH,00100100b  ; FIFO for mode 101

DB       -1             ; end of Aux registers
DW      PW+03CDh,00h    ; set page 0
```

```
        DW      -1              ; end of the register table

;-------------------------------------------------------------------------
;       SetupDisplay - Setup LCD display operation.
;
;       Entry:  n/a
;       Exit:   n/a
;       Uses:   AX,BX,CX,DX,SI,DI
;-------------------------------------------------------------------------

SetupDisplay:

        call    SPCWaitForPLLsLocked
        call    SPCLoadPanelRegs

        PUTAUX  00h,30h         ; Turn on the LCD circuitry

        ret


;-----------------------------------------------------------------------
;
;  STEP3 - Use tables to program the standard VGA registers.
;
;-----------------------------------------------------------------------

Step3:
        mov     si, offset VGAregs
        call    RunTable
        ret

VGAregs:
        DW      PW+3C2h,0EBh    ; Misc. Output.

        DW       3C4h           ; Sequencer.
        DB       00h, 00h
        DB       01h, 01h
        DB       02h, 0Fh
        DB       03h, 00h
        DB       04h, 0Eh
        DB       00h, 03h
        DB       -1

        DW       3D4H           ; CRTC
        DB       11H, 08H       ; enable CRTC regs 0..7
        DB       00H, 05fh
        DB       01H, 04fh
        DB       02H, 050h
        DB       03H, 082h
        DB       04H, 054h
        DB       05H, 080h
        DB       06H, 00bh
        DB       07H, 03eh
```

```
DB      08H, 000h
DB      09H, 040h
DB      0AH, 000h
DB      0BH, 000h
DB      0CH, 000h
DB      0DH, 000h
DB      0EH, 000h
DB      0FH, 000h
DB      10H, 0eah
DB      11H, 08ch
DB      12H, 0dfh
DB      13H, 050h
DB      14H, 040h
DB      15H, 0e7h
DB      16H, 004h
DB      17H, 0A3h
DB      18H, 0FFh

DB      38H, 00h
DB      37H, 00h
DB      36H, 00h
DB      35H, 00h
DB      34H, 00h
DB      33H, 00h
DB      32H, 00h
DB      31H, 00h
DB      30H, 00h
DB      -1

DW      3CEH            ; Graphics controller.
DB      00H, 000h
DB      01H, 000h
DB      02H, 000h
DB      03H, 000h
DB      04H, 000h
DB      05H, 040h
DB      06H, 005h
DB      07H, 00Fh
DB      08H, 0FFh
DB      -1
DW      -1
```

```
;-----------------------------------------------------------------------
;
;    STEP4 - program the DAC.
;
;-----------------------------------------------------------------------

Step4:
        mov     al, 0FFh          ; Turn on PEL mask.
        mov     dx, 03c6h         ;
        out     dx, al            ;

        mov     al, 0             ; Set data index to zero.
        mov     dx, 3c8h          ;
        out     dx, al            ;

        mov     dx, 3C9h          ; Set DAC write index
        mov     cx, 3 * 256       ; CX = # bytes (RGB * 256).
        mov     si, offset NewDAC ; SI -> new DAC data.
DACloop:
        lodsb                    ; Get the byte.
        out      dx, al          ; Set the PEL data.
        loop     DACloop         ; Continue till CX == 0.

        ret


NewDAC  DB      00h,00h,00h,00h,00h,2ah,00h,2ah,00h,00h,2ah,2ah
        DB      2ah,00h,00h,2ah,00h,2ah,2ah,15h,00h,2ah,2ah,2ah
        DB      15h,15h,15h,15h,15h,3fh,15h,3fh,15h,15h,3fh,3fh
        DB      3fh,15h,15h,3fh,15h,3fh,3fh,3fh,15h,3fh,3fh,3fh
        DB      00h,00h,00h,05h,05h,05h,08h,08h,08h,0bh,0bh,0bh
        DB      0eh,0eh,0eh,11h,11h,11h,14h,14h,14h,18h,18h,18h
        DB      1ch,1ch,1ch,20h,20h,20h,24h,24h,24h,28h,28h,28h
        DB      2dh,2dh,2dh,32h,32h,32h,38h,38h,38h,3fh,3fh,3fh
        DB      00h,00h,3fh,10h,00h,3fh,1fh,00h,3fh,2fh,00h,3fh
        DB      3fh,00h,3fh,3fh,00h,2fh,3fh,00h,1fh,3fh,00h,10h
        DB      3fh,00h,00h,3fh,10h,00h,3fh,1fh,00h,3fh,2fh,00h
        DB      3fh,3fh,00h,2fh,3fh,00h,1fh,3fh,00h,10h,3fh,00h
        DB      00h,3fh,00h,00h,3fh,10h,00h,3fh,1fh,00h,3fh,2fh
        DB      00h,3fh,3fh,00h,2fh,3fh,00h,1fh,3fh,00h,10h,3fh
        DB      1fh,1fh,3fh,27h,1fh,3fh,2fh,1fh,3fh,37h,1fh,3fh
        DB      3fh,1fh,3fh,3fh,1fh,37h,3fh,1fh,2fh,3fh,1fh,27h
        DB      3fh,1fh,1fh,3fh,27h,1fh,3fh,2fh,1fh,3fh,37h,1fh
        DB      3fh,3fh,1fh,37h,3fh,1fh,2fh,3fh,1fh,27h,3fh,1fh
        DB      1fh,3fh,1fh,1fh,3fh,27h,1fh,3fh,2fh,1fh,3fh,37h
        DB      1fh,3fh,3fh,1fh,37h,3fh,1fh,2fh,3fh,1fh,27h,3fh
        DB      2dh,2dh,3fh,31h,2dh,3fh,36h,2dh,3fh,3ah,2dh,3fh
        DB      3fh,2dh,3fh,3fh,2dh,3ah,3fh,2dh,36h,3fh,2dh,31h
        DB      3fh,2dh,2dh,3fh,31h,2dh,3fh,36h,2dh,3fh,3ah,2dh
        DB      3fh,3fh,2dh,3ah,3fh,2dh,36h,3fh,2dh,31h,3fh,2dh
        DB      2dh,3fh,2dh,2dh,3fh,31h,2dh,3fh,36h,2dh,3fh,3ah
        DB      2dh,3fh,3fh,2dh,3ah,3fh,2dh,36h,3fh,2dh,31h,3fh
        DB      00h,00h,1ch,07h,00h,1ch,0eh,00h,1ch,15h,00h,1ch
```

```
DB      1ch,00h,1ch,1ch,00h,15h,1ch,00h,0eh,1ch,00h,07h
DB      1ch,00h,00h,1ch,07h,00h,1ch,0eh,00h,1ch,15h,00h
DB      1ch,1ch,00h,15h,1ch,00h,0eh,1ch,00h,07h,1ch,00h
DB      00h,1ch,00h,00h,1ch,07h,00h,1ch,0eh,00h,1ch,15h
DB      00h,1ch,1ch,00h,15h,1ch,00h,0eh,1ch,00h,07h,1ch
DB      0eh,0eh,1ch,11h,0eh,1ch,15h,0eh,1ch,18h,0eh,1ch
DB      1ch,0eh,1ch,1ch,0eh,18h,1ch,0eh,15h,1ch,0eh,11h
DB      1ch,0eh,0eh,1ch,11h,0eh,1ch,15h,0eh,1ch,18h,0eh
DB      1ch,1ch,0eh,18h,1ch,0eh,15h,1ch,0eh,11h,1ch,0eh
DB      0eh,1ch,0eh,0eh,1ch,11h,0eh,1ch,15h,0eh,1ch,18h
DB      0eh,1ch,1ch,0eh,18h,1ch,0eh,15h,1ch,0eh,11h,1ch
DB      14h,14h,1ch,16h,14h,1ch,18h,14h,1ch,1ah,14h,1ch
DB      1ch,14h,1ch,1ch,14h,1ah,1ch,14h,18h,1ch,14h,16h
DB      1ch,14h,14h,1ch,16h,14h,1ch,18h,14h,1ch,1ah,14h
DB      1ch,1ch,14h,1ah,1ch,14h,18h,1ch,14h,16h,1ch,14h
DB      14h,1ch,14h,14h,1ch,16h,14h,1ch,18h,14h,1ch,1ah
DB      14h,1ch,1ch,14h,1ah,1ch,14h,18h,1ch,14h,16h,1ch
DB      00h,00h,10h,04h,00h,10h,08h,00h,10h,0ch,00h,10h
DB      10h,00h,10h,10h,00h,0ch,10h,00h,08h,10h,00h,04h
DB      10h,00h,00h,10h,04h,00h,10h,08h,00h,10h,0ch,00h
DB      10h,10h,00h,0ch,10h,00h,08h,10h,00h,04h,10h,00h
DB      00h,10h,00h,00h,10h,04h,00h,10h,08h,00h,10h,0ch
DB      00h,10h,10h,00h,0ch,10h,00h,08h,10h,00h,04h,10h
DB      08h,08h,10h,0ah,08h,10h,0ch,08h,10h,0eh,08h,10h
DB      10h,08h,10h,10h,08h,0eh,10h,08h,0ch,10h,08h,0ah
DB      10h,08h,08h,10h,0ah,08h,10h,0ch,08h,10h,0eh,08h
DB      10h,10h,08h,0eh,10h,08h,0ch,10h,08h,0ah,10h,08h
DB      08h,10h,08h,08h,10h,0ah,08h,10h,0ch,08h,10h,0eh
DB      08h,10h,10h,08h,0eh,10h,08h,0ch,10h,08h,0ah,10h
DB      0bh,0bh,10h,0ch,0bh,10h,0dh,0bh,10h,0fh,0bh,10h
DB      10h,0bh,10h,10h,0bh,0fh,10h,0bh,0dh,10h,0bh,0ch
DB      10h,0bh,0bh,10h,0ch,0bh,10h,0dh,0bh,10h,0fh,0bh
DB      10h,10h,0bh,0fh,10h,0bh,0dh,10h,0bh,0ch,10h,0bh
DB      0bh,10h,0bh,0bh,10h,0ch,0bh,10h,0dh,0bh,10h,0fh
DB      0bh,10h,10h,0bh,0fh,10h,0bh,0dh,10h,0bh,0ch,10h
DB      00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h
DB      00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h,00h
```

```
;-----------------------------------------------------------------------
;
;  STEP5 - Program Attribute controller.
;
;  Reset Index/Data Flip Flop of Attribute controller.
;  Write attributes information.
;
;-----------------------------------------------------------------------

Step5:
        mov     dx, 3DAh        ; Set attributes register to index.
        in      al, dx          ;

        mov     si, offset AttribTable
        mov     dx, 3C0h        ;
        mov     cx, 15h         ; Write to 15h registers.
        mov     bx, 0h          ; Use BX as the index counter.
@@:
        mov     ax, bx          ; Set the next index.
        out     dx, al          ;
        inc     bx              ;
        lodsb                   ; Read the next byte.
        out     dx, al          ; Write the next data.
        loop    @B

        mov     dx, 3DAh        ; Set attributes register to index.
        in      al, dx          ;

        mov     al, 20h
        mov     dx, 3C0h
        out     dx, al
        ret


AttribTable:
        db      00h, 01h, 02h, 03h, 04h, 05h, 06h, 07h
        db      08h, 09h, 0Ah, 0Bh, 0Ch, 0Dh, 0Eh, 0Fh
        db      41h, 00h, 0Fh, 00h, 00h


;-----------------------------------------------------------------------
;
;    STEP6 - Show different banks..., just to get something on the
;            screen to show we are alive....
;
;-----------------------------------------------------------------------

Step6:
        cld
        mov     ax, 0A000h
        mov     es, ax

        mov     bl, 0h          ; bl = bank counter
```

```
        mov     dx, 3cdh        ; dx points to bank register
@@:
        mov     al, bl
        out     dx, al          ; set bank
        mov     cx, 8000h       ; 1 bank = 32k Words
        xor     di, di
        mov     ah, al          ; ah = al = color
        rep     stosw           ; fill on bank with color
        add     bl, 11h         ; next bank?
        jnc     @B

        mov     al,0
        out     dx,al           ; set bank 0 again

        ret



; ADDITIONAL SUPPORT ROUTINES
;----------------------------

; RunTable
; This routine interprets the configuration tables that we have
; presented above.  PW is a mask bit that causes a branch from
; index/data to a simple port write of data (ie. if High bit on
; Port value is set, then branch to simple port write BR3)


br0:
        dec     si                      ; Undo last byte overrun

RunTable:
        lodsw                           ; Read next control word
        cmp     ax, -1                  ; End of table?
        je      brx                     ; Exit if so

        mov     dx, ax                  ; Get port in DX
        and     dh, 7fh                 ; Mask off port write bit
        test    ah, 80h                 ; Test if port write?
        jnz     br3                     ; Jump if so
br2:
        lodsw                           ; Read index,data
        cmp     al, -1                  ; End of run
        je      br0                     ; Jump back if so
        out     dx, ax                  ; Write the data
        jmp     br2                     ; Back for more
br3:
        lodsw                           ; Read data
        out     dx, al                  ; Write the data
        jmp     RunTable                ; Back for more
brx:
        ret
```

```
;-------------------------------------------------------------------------
;       ReadAuxReg - Read Auxiliary Register
;
;       Entry:  AL - Register to read
;       Exit:   AL - Data read
;       Uses:   AL
;-------------------------------------------------------------------------
SPCReadAuxReg:
        push    dx                      ; Save DX
        mov     dx, 3deh                ; Setup auxiliary register
        out     dx, al                  ; Write index
        inc     dx                      ; Adjust to data register
        in      al, dx                  ; Read data
        pop     dx                      ; Restore DX
        ret


;-------------------------------------------------------------------------
;       WriteAuxReg - Write Auxiliary Register
;
;       Entry:  AL - Data to write
;               AH - Register to write
;       Exit:   AX - Unchanged
;       Uses:   AX
;-------------------------------------------------------------------------
SPCWriteAuxReg:
        push    dx                      ; Save DX
        xchg    al, ah                  ; Set AL=reg, AH=data
        mov     dx, 3deh                ; Setup auxiliary register
        out     dx, ax                  ; Write data to reg
        xchg    al, ah                  ; Restore for caller
        pop     dx                      ; Restore DX
        ret


;-------------------------------------------------------------------------
;       SPCWaitForPLLsLocked
;
;       Note: Called only at initialization time and after each power-up
;
;-------------------------------------------------------------------------
SPCWaitForPLLsLocked:
        xor     cx, cx                  ; prevent looping forever
@@:
        GETAUX  028h                    ; get current PLL state
        and     al, 03h                 ; keep lock bits only
        cmp     al, 03h                 ; both PLLs locked?
        loopnz  @B
        ret

MODE101 ENDP
```

```
;-------------------------------------------------------------------------------
;; Panel info tables. All relevant registers for the panel are within one PST
; structure.
;
;-------------------------------------------------------------------------------
PST     STRUCT
        AUX_1           DB      ?
        AUX_2           DB      ?
        AUX_6           DB      ?
        AUX_7           DB      ?
        AUX_16          DB      ?
        AUX_17          DB      ?
        AUX_15          DB      ?
        CRTCB_0         DB      ?
        CRTCB_1         DB      ?
        CRTCB_6         DB      ?
        CRTCB_7         DB      ?
        CRTCB_10        DB      ?
        CRTCB_11        DB      ?
        CRTCB_12        DB      ?
        PLL0Den         DB      ?
        PLL0Num         DB      ?
        PLL1Den         DB      ?
        PLL1Num         DB      ?
PST     ENDS

aAUXIndeces     DB      1,2,6,7,16h,17h,15h
aCRTCBIndeces   DB      0,1,6,7,10h,11h,12h

;-------------------------------------------------------------------------------
;       SPCLoadPanelRegs - Programs all registers for a given panel type.
;
;       Uses: Nothing
;       Note: Called only once, at initialization.
;             Panel is OFF at the time of this call.
;
;-------------------------------------------------------------------------------
SPCLoadPanelRegs:
        PUSHA
        LEA     SI,LcdPanelTypeTbl
        PUSH    SI
        LEA     SI,[SI.PST.PLL0Den]
        LODS    CS:CS_BYTE
        MOV     CH,AL
        PUTAUX  21h,AL
        LODS    CS:CS_BYTE
        MOV     CL,AL
        PUTAUX  20h,AL
        LODS    CS:CS_BYTE
        PUTAUX  23h,AL
        LODS    CS:CS_BYTE
        PUTAUX  22h,AL
        POP     SI
```

```
        ; for dual panels program the half frame buffer

        TEST    CS:[SI.PST.AUX_2],1     ; Q: is panel dual?
        JNZ     SHORT SPCFrameBufDone   ;    N: split

        ;ECHO     SPC8110.ASM : Small panel bug fix1 ...

        ; Slow dot clock modes on panels smaller than 640 pixels may
        ; not show up correctly. Panels smaller than 640 pixels will
        ; be programmed to 640 pixels width. (There's a bug in 8110
        ; if Width(B-set) < Width(A-set). Width(A-set) is always = 640 for
        ; slow dot clock modes)

        MOV     AL,640/8
        MOV     AH,CS:[SI.PST.CRTCB_1]
        CMP     AH,AL
        JA      SHORT @F
        MOV     AH,AL
@@:
        MOV     AL,CS:[SI.PST.CRTCB_12] ; AL = Height/4

        MUL     AH                      ; AX = (Height/4)*(Width/8)=(H*W)/32
        SHR     AX,1                    ; AX = (HxW)/64

        ; do mono/color adjustments

        TEST    CS:[SI.PST.AUX_2],2     ; Q: color panel
        JZ      SHORT SPCMono           ;    N: do mono adjustments
        MOV     BX,AX
        ADD     AX,AX
        ADD     AX,BX                   ; AX = 3*(H*W)/64
        SUB     AX,7                    ; AX = 3*(H*W)/64-7
        JMP     SHORT SPCDoFrameLength
SPCMono:
        SUB     AX,3                    ; AX = (H*W)/64-3
SPCDoFrameLength:
        MOV     BX,AX
        PUTAUX  1Dh,BH
        PUTAUX  1Eh,BL
        MOV     AL,0Fh                  ; assume page 15 and 1Mb of memory
        CALL    SPCIs1Meg
        JE      SHORT @F
        SHR     AX,1                    ; make it page 7 for 512k of memory
@@:
        ; adjust the frame buffer position based on frame
        ; buffer length bits b15,b14

        TEST    BH,40h
        JZ      SHORT @F
        AND     AL,11111110b
```

```
@@:
        TEST    BH,80h
        JZ      SHORT @F
        AND     AL,11111100b
@@:
        PUTAUX  1Ch,AL                  ; frame start at the very end of mem.

SPCFrameBufDone:
        MOV     CX,6                    ; do 6 AUX registers
        LEA     DI,aAUXIndeces
        CLD

DoAUXRegs:
        LODS    CS:CS_BYTE              ; get value for AUX

        PUTAUX  CS:[DI],AL              ; program AUX reg
        INC     DI
        LOOP    DoAUXRegs

        ; special case AUX[15h] : modify only the skew bits

        GETAUX  15h
        AND     AL,11110011b
        MOV     AH,AL
        LODS    CS:CS_BYTE
        OR      AL,AH
        PUTAUX  15h,AL

        GETAUX  00h
        PUSH    AX                      ; save AUX[0]
        OR      AL,1
        PUTAUX  00h,AL                  ; enable CRTC_B set
        MOV     CX,7                    ; do 7 CRTC_B registers
        LEA     BX,aCRTCBIndeces
        CALL    LoadCRTCTable

        ; If a panel x-size is less then 640 pixels,
        ; program the CRTCB[1] with 640 pixels. This is to
        ; take care of a bug if panel size <= HDE. The smallest
        ; HDE is 40 for any IBM mode.

        ;ECHO     SPC8110.ASM : Small panel bug fix2 ...

        MOV     AL,01h
        CALL    SPCReadCRTC
        CMP     AL,640/8
        JAE     SHORT @F
        MOV     AX,0150h                ; write CRTCB[1] = 50h (80 decimal)
        CALL    SPCWriteCRTC
```

```
@@:
        POP     AX                      ; get saved AUX[0]
        PUTAUX  00h,AL                  ; restore it

        POPA
        RET

;-------------------------------------------------------------------------
;       SPCPCIEnable - Find and enable PCI chip
;
;       Entry:  Nothing
;       Exit:   Nothing
;       Uses:   Nothing
;-------------------------------------------------------------------------
SPCPCIEnable PROC NEAR
            PUSHA
        CALL    SPCFind8110             ; yes, try to find our PCI chip
        JC      PCIError                ; could not find, do nothing
        CALL    SPCPCIWriteReg4         ; enable IO & mem decoding
PCIError:
        POPA
        RET

SPCPCIEnable ENDP

;-------------------------------------------------------------------------
;       SPCIs1Meg - Test if 1 Mb of memory installed
;
;       Entry:  Nothing
;       Exit:   Z flg - Set if 1 Mb installed
;       Uses:   Nothing
;-------------------------------------------------------------------------
SPCIs1Meg:
        PUSH    AX
        GETAUX  03h
        TEST    AL,01000000b          ; single DRAM?
        POP     AX
        RET

;-------------------------------------------------------------------------
;       LoadCRTCTable - Load CRTC table
;
;       Entry:  BX - Offset of aCRTCBIndeces
;               CX - Number of registers
;       Exit:   AL - Data to write
;               AH - Register to write
;       Uses:   AX, BX, CS
;-------------------------------------------------------------------------
LoadCRTCTable PROC NEAR
@@:     LODS    CS:CS_BYTE              ; get value for CRTC in AL
        MOV     AH,CS:[BX]              ; get index in AH
        CALL    SPCWriteCRTC
        INC     BX
```

```
        LOOP    @B
        RET
LoadCRTCTable ENDP

;-------------------------------------------------------------------------
;       SPCReadCRTC - Read CRTC Register
;
;       Entry:  AL - Register to read
;       Exit:   AL - Data read
;       Uses:   AL
;-------------------------------------------------------------------------
SPCReadCRTC:
        PUSH    DX                  ; Save DX
        PUSH    AX                  ; Save AX
        MOV     AX,3D4H             ; CRTC at 3D4H
        MOV     DX,AX
        POP     AX                  ; Restore AX
        OUT     DX,AL               ; Write index
        INC     DX                  ; Adjust to data register
        IN      AL,DX               ; Read data
        POP     DX                  ; Restore DX
        RET

;-------------------------------------------------------------------------
;       SPCWriteCRTC - Write CRTC Register
;
;       Entry:  AL - Data to write
;               AH - Register to write
;       Exit:   AX - Unchanged
;       Uses:   AX
;-------------------------------------------------------------------------
SPCWriteCRTC:
        PUSH     DX               ; Save DX
        XCHG     AL,AH            ; Set AL=reg, AH=data
        PUSH     AX               ; Save AX
        MOV      AX,3D4H          ; CRTC at 3D4H
        MOV      DX,AX
        POP      AX               ; Restore AX
        OUT      DX,AX            ; Write data to reg
        XCHG     AL,AH            ; Restore for caller
        POP      DX               ; Restore DX
        RET
```

```
;-------------------------------------------------------------------------------
;       SPCFind8110 - Find SPC8110 Chip
;
;       Entry:  Nothing
;       Exit:   BH - bus number
;               BL - devnum
;               C Flag - set if chip not found
;       Uses:   AX, CX, DX, SI
;-------------------------------------------------------------------------------
SPCFind8110     PROC NEAR
        MOV     AH,0B1H
        MOV     AL,2
        XOR     SI,SI
        MOV     CX,DEVICE_ID          ; CX = DEVID
        MOV     DX,VENDOR_ID          ; DX = VENDOR ID
        INT     1AH
        JC      SHORT ERRDEVICE
        TEST    AH,AH
        JNZ     SHORT ERRDEVICE
        CLC
        RET
ERRDEVICE:
        STC
        RET
SPCFind8110     ENDP


;-------------------------------------------------------------------------------
;       SPCPCIWriteReg4 - Enable memory and IO for PCI system
;       Entry : BH - bus number
;               BL - devnum
;       Exit:   Nothing
;       Uses:   AX, CX, DI
;
;-------------------------------------------------------------------------------
SPCPCIWriteReg4 PROC NEAR
        MOV     AX,0B10BH
        MOV     DI,4                    ; offset
        MOV     CX,3                    ; R[4] = 3 (decode memory & IO)
        INT     1AH
        RET
SPCPCIWriteReg4 ENDP

INCLUDE lcd.dat
;(See LCD.DAT on page 65)

        END     MODE101
```

LCD.DAT

```
;-------------------------------------------------------------------------------
;;
; Copyright (c) Seiko Epson Corp. 1997. All rights reserved.
;
;-------------------------------------------------------------------------------
;
; This table supports various panels. To support a specific panel, remove
; the semicolon (uncomment) in front of the line.
; e.g. if we want to support 640x200 single 4-bit monochrome panel,
;      1. change ";Tbl640x200S4M..." to "Tbl640x200S4M...".
;      2. add a semicolon in front of any other line that is not commented.
;         i.e. only one entry is allowed in this table.
;
;-------------------------------------------------------------------------------
;
; For 640x480 Dual 8-Bit Monochrome panel, use Tbl640x480D8M
; For 640x480 Dual 8-Bit Color panel, use Tbl640x480D8C
; For 640x480 Single 8-Bit Monochrome panel, use Tbl640x480S8M
; For 640x200 Single 4-Bit Monochrome panel, use Tbl640x200S4M
; For 320x240 Single 4-Bit Monochrome panel, use Tbl320x240S4M
; For 640x480 Dual 8-Bit Color (with Two Shift Clock) panel, use Tbl640x480D8C2SC
; For 640x480 Single 8-Bit Color panel, use Tbl640x480S8C
; For 640x480 Dual 16-Bit Color panel, use Tbl640x480D16C
; For 640x480 Single 16-Bit Color panel, use Tbl640x480S16C
; For 640x480 TFT 18-Bit panel, use Tbl640x480TFT18
; For 640x480 TFT 12-Bit panel, use Tbl640x480TFT12
; For 640x480 TFT 9-Bit panel, use Tbl640x480TFT09
; For 320x240 Single 8-Bit Color panel, use Tbl320x240S8C
; For 800x600 Dual 16-Bit Color panel, use Tbl800x600D16C
; For 480x320 Single 4-Bit Monochrome panel, use Tbl480x320S4M
;
;-------------------------------------------------------------------------------

LcdPanelTypeTbl:

;                     A1    A2     A6  A7  A16  A17 A15 C0   C1    C6   C7  C10 C11  C12   Den0   Num0     Den1  Num1
;Tbl640x480D8M    PST <42h,01001000b,20h,00h,04Fh,01h,0Ch,1Eh,640/8,0F1h,00h,0EBh,0Dh,480/4,01h,01h OR G11,03h,02h OR G11>
;Tbl640x480D8C    PST <42h,01000010b,20h,00h,04Fh,01h,0Ch,1Eh,640/8,0F1h,00h,0EBh,0Dh,480/4,01h,01h OR G11,03h,02h OR G11>
;Tbl640x480S8M    PST <42h,01001001b,20h,00h,04Fh,01h,0Ch,1Eh,640/8,0E1h,01h,0EBh,0Dh,480/4,01h,02h OR G11,03h,04h OR G11>
;Tbl640x200S4M    PST <42h,01000001b,20h,00h,04Fh,01h,0Ch,1Eh,640/8,0C9h,00h,0EBh,0Dh,200/4,0Ch,0Ah OR G11,0Ch,05h OR G11>
;Tbl320x240S4M    PST <42h,01000001b,20h,00h,04Fh,01h,0Ch,1Eh,320/8,0F1h,00h,0EBh,0Dh,240/4,01h,01h OR G11,02h,01h OR G11>
;Tbl640x480D8C2SC PST <42h,01001010b,20h,00h,04Fh,01h,0Ch,1Eh,640/8,0F1h,00h,0EBh,0Dh,480/4,01h,01h OR G11,03h,02h OR G11>
;Tbl640x480S8C    PST <42h,01001011b,20h,00h,04Fh,01h,0Ch,1Eh,640/8,0E1h,01h,0EBh,0Dh,480/4,01h,02h OR G11,03h,04h OR G11>
;Tbl640x480D16C   PST <42h,01001110b,20h,00h,04Fh,01h,0Ch,1Eh,640/8,0F1h,00h,0EBh,0Dh,480/4,01h,01h OR G11,03h,02h OR G11>
;Tbl640x480S16C   PST <42h,01001111b,20h,00h,04Fh,01h,0Ch,1Eh,640/8,0E1h,01h,0EBh,0Dh,480/4,01h,02h OR G11,03h,04h OR G11>
;Tbl640x480TFT18  PST <42h,01011111b,00h,00h,000h,00h,00h,1Eh,640/8,00Ch,24h,0EBh,0Dh,480/4,04h,07h OR G11,01h,02h OR G11>
;Tbl640x480TFT12  PST <42h,01011011b,00h,00h,000h,00h,00h,1Eh,640/8,00Ch,24h,0EBh,0Dh,480/4,04h,07h OR G11,01h,02h OR G11>
;Tbl640x480TFT09  PST <42h,01010111b,00h,00h,000h,00h,00h,1Eh,640/8,00Ch,24h,0EBh,0Dh,480/4,04h,07h OR G11,01h,02h OR G11>
;Tbl320x240S8C    PST <42h,01001111b,20h,20h,04Fh,01h,0Ch,1Eh,320/8,0F1h,00h,0EBh,0Dh,240/4,01h,02h OR G11,03h,04h OR G11>
;Tbl800x600D16C   PST <42h,01001110b,20h,00h,04Fh,01h,0Ch,1Eh,800/8,02Dh,01h,0EBh,0Dh,600/4,04h,07h OR G11,01h,02h OR G11>
;Tbl480x320S4M    PST <42h,01000001b,20h,00h,04Fh,01h,0Ch,1Eh,480/8,041h,01h,0EBh,0Dh,320/4,03h,04h OR G11,06h,04h OR G11>
```

USING THE SPC8110 BitBLT ACCELERATOR

The term BitBLT is an acronym for Bit Block Transfer. During a BitBLT operation data is the transferred from one memory location (source) to another memory location (destination). With current graphical user interfaces (GUIs) this term generally refers to the transfer of bitmap images to or from video memory. The resulting bitmap image may be formed by up to three items or operands:

- the source data
- an optional pattern
- the current destination data

The operands are combined using logical AND, OR, XOR and NOT operations. The combining process is called a Raster Operation or ROP for short.

The bulk of the BitBLTs which take place in everyday GUI usage consist of a limited number of the total possible ROPs. The BitBLTaccelerator (engine) in the SPC8110 was designed to increase the speed of several of the most common GUI operations by off-loading work from the CPU, reducing traffic on the system bus and improving the efficiency of the display memory interface. It is designed to work in both planar and packed pixel graphical modes.

The following sections describe the BitBLTs supported by the SPC8110, the BitBLT control registers and programming notes. The final section contains sample BitBLT programming code.

SPC8110 BitBLT Types

The SPC8110 supports seven fundamental BitBLT operations.

Four of the seven operations (solid fill, move, pattern fill and destination invert) are self completing. Once they are started they complete on their own and do not require data transfers with the CPU.

Three BitBLT operations (write, color expand and read) require that data be either written to, or read from video memory. In each case the data must be transferred 32-bits (one DWORD) at a time.

The following is a brief description of the BLT types:

Write BLT

This BitBLT fills a specified area of video memory with CPU supplied data. A typical use would be to copy a bitmap image from system memory to video memory.

Color Expand BLT

This operation is similar to the Write BLT. The difference is that a bit set to “1” in the source data becomes a pixel of foreground color. A source bit set to “0” is converted to a pixel of background color. This function is useful for increasing the speed of writing text in graphical modes.

This BLT operation includes several options to further enhanced its text handling capabilities. As with the Write BLT, all data sent to the BLT engine must be 32-bit writes. Not all of the 32-bits are expanded, in planer display modes the eight least significant bits are expanded to fill eight pixels. In packed pixel display modes the four least significant bits are expanded to fill four pixels.

Solid Fill BLT

Solid Fill BLT is used to rapidly fill a given area of video memory with a solid color. This operation can be used to paint large screen areas or to set areas of video memory to a given value.

Move BLT

The Move BLT is a video memory to video memory copy. This operation has two intended purposes: 1) unattended video memory to video memory copies and 2) The ability to efficiently save a visible bitmap to off-screen video memory.

The move may be from one rectangular area to another. To temporarily save a portion of the visible display memory the source or destination may be specified to be linear rather than rectangular. This configuration allows each line of the move BLT area to butt up to the previous line rather than require a complete row of address space for each line.

Pattern Fill BLT

This BitBLT fills a specified rectangular area of display memory with a pattern. The fill pattern is an array of pixels stored in off-screen memory. The pattern area is limited to an eight by eight pixel array and must be loaded to off-screen memory prior to the BLT starting.

The pattern may stored at any video memory location in a consecutive array of pixels. As a pattern is defined to be eight pixels square this works out to being thirty-two consecutive bytes for planar display modes and sixty-four consecutive pixels for packed pixel modes.

Read BLT

This BitBLT increases the speed of transferring data from display memory to system memory. This operation is the compliment of a Write BitBLT.

Invert BLT

This BitBLT operation inverts the colors within the specified rectangular area of display memory. The inversion is performed by NOTing each of the pixels in the specified area. For example if the data pattern in video memory is 55h it will become AAh.

The BitBLT Registers

The BitBLT control registers on the SPC8110 are located at I/O ports 3D0h through 3D3h. Port 3D0h serves both as an index register and as the BitBLT status register. Setting the index bits in 3D0h to values 0 through 5 accesses the remaining registers addressed at I/O ports 3D1h - 3D3h. A thirty-two bit write may be used to write both the index and data registers simultaneously or each of the I/O ports may be written with eight bits of data in turn. In order to read the data registers an 8-bit write at address 3D0h to set the index followed by one or more reads at address 3D1h - 3D3h is required.

The following tables summarize the SPC8110 BitBLT control registers.

BitBLT Index Register						
3D0h	BitBLT Status	n/a	n/a	n/a	n/a	BitBLT Register Index

BitBLT Status - Set this bit to “1” to start a BitBLT operation. Upon completion of the BLT the bit returns to “0”. Software can prematurely end a BitBLT by resetting this bit.

BitBLT Register Index - selects each of the BitBLT Data Registers.

Register 0: BitBLT Source Address								
3D3h	n/a	n/a	n/a	b20	b19	b18	b17	b16
3D2h	b15	b14	b13	b12	b11	b10	b9	b8
3D1h	b7	b6	b5	b4	b3	b2	b1	b0

Bits b20 - b0 of this register specifies the pixel location of the BLT source for Move, Pattern Fill and Read BLTs. Source phase alignment is implicitly specified by this value during Write and Color Expand BLTs. For a discussion of alignment see Phase Alignment on page 72.

The source address is the upper left corner pixel of the BitBLT rectangle for incrementing BitBLT operations, and the lower right corner of the BitBLT rectangle for decrementing Move BLTs.

e.g. To specify the source address of the pixel located at row 99, column 23 (in mode 12h or mode 101h).
Src Address = 99 * 640 + 23. As we are specifying a pixel location this calculation works for both packed pixel and planer modes.

Register 1: BitBLT Destination Address								
3D3h	n/a	n/a	n/a	b20	b19	b18	b17	b16
3D2h	b15	b14	b13	b12	b11	b10	b9	b8
3D1h	b7	b6	b5	b4	b3	b2	b1	b0

Bits b20 - b0 specify the destination pixel for all BitBLT operations except Reads. For Read BLTs this register implicitly defines the destination phase alignment. For a discussion of alignment see Phase Alignment on page 72.

The destination address is the upper left corner of the BLT rectangle for incrementing Move BLTs and the lower right corner of the BitBLT rectangle for decrementing BLTs.

e.g. To specify a destination address of the pixel located at row 220, column 100 (mode 12h or mode 101h). Dst Address = 220 * 640 + 100. As we are specifying a pixel location this calculation works for both packed pixel and planer modes.

Register 2: BitBLT Height and Background Color								
3D3h	n/a	n/a	n/a	n/a	n/a	n/a	h9	h8
3D2h	h7	h6	h5	h4	h3	h2	h1	b0
3D1h	c7	c6	c5	c4	c3	c2	c1	c0

Bits h9 - h0 specify the actual height (in rows) less one for the BLT. All BitBLT operations require a height to be set.

e.g. A value of 199 written to the height instructs the blter to do a 200 line high BitBLT.

Bits c7 - c0 specify the background color for Color Expand BLTs. The background color specifies the color to draw a pixel with when a “0” is found in the color expand data. If the Transparent Background option is selected then this value is ignored. In planar display modes only the lower 4 bits are significant. In packed pixel display modes all 8 bits are significant.

Register 3: BitBLT Width and Foreground Color								
3D3h	n/a	n/a	n/a	n/a	n/a	n/a	w9	w8
3D2h	w7	w6	w5	w4	w3	w2	w1	w0
3D1h	c7	c6	c5	c4	c3	c2	c1	c0

Bits w9 - w0 specify the actual width of the BitBLT (in columns) less one. All BLTs require a width to be specified.

e.g. A value of 83 would be written for a BitBLT width of 84 pixels.

Bits c7 - c0 specify the foreground color for Color Expand and Solid Fill BLTs. The FG (foreground) Color bits specify the color to draw a pixel with when a “1” is found in the data when performing a Color Expand BitBLT. The FG Color is the color used for the fill when performing a solid fill BitBLT. In planar display modes only the lower 4 bits are significant. In packed pixel display modes all 8 bits are significant.

Register 4: BitBLT Description Register								
3D3h	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
3D2h	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
3D1h	Direction dec/inc	Dst Linear Select	Src Linear Select	n/a	n/a	BitBLT Operation		

With the exception of the BitBlt Operation field this register controls Move BLT options.

Direction determines which direction the BLT address counters will step. Setting this bit to a “1” will result in a decrement BitBLT, set to a “0” results in an increment BLT. This bit can be set for other BitBLT types but it is only intended for Move BLTs. It is recommended that all other BitBLT types be run incremental only.

An incremental BLT proceeds from the upper left corner to the lower right corner. By setting the Direction to decrement the BLT will proceed from the lower right corner to the upper left. See the application notes section for a description of when to set a decrement Move BLT.

Dst Linear and Src Linear bits further configure Move BLTs. They describe the action the BLT engine will take at the end of each “row” of information. When set these bits cause their respective sources to not step to the next line of display memory when the current one completes. Instead the next sequential DWORD is addressed.

If not for this ability saving a strip of pixels 8 wide and the 480 high in packed pixel mode would require an amount of off-screen memory to what the screen uses in visible memory. (640 x 480 = 307,200 bytes). Using the linear capability reduces the size of memory required to temporarily store the strip of video memory to 3840 bytes (8 * 480 = 3,840)

Bits 2 through 0 of this register specify the desired BitBLT type. The value set here must correspond to the supported SPC8110 BLT types. In numerical order the operations are:

0 - Write
1 - Color Expand
2 - Solid Fill
3 - Move
4 - Pattern Fill
5 - Read
6 - Destination Invert

e.g. To perform a Destination Invert BLT, 110b would be written to the BitBLT operation field.

Register 5: BitBLT Function Register								
3D3h	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
3D2h	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
3D1h	n/a	n/a	Auto Update	Auto Start	n/a	n/a	Transparent Background	Translate Select

With the exception of Translate Select the bits in this register are Color Expand BLT modifiers.

Auto Update, when enabled, will cause the destination address register to be updated to point to the pixel just to the right of the top right point of the current BLT. This feature should be used when Color Expand is used to draw text on a graphical display as it saves calculating and updating the destination register after each character is drawn.

Auto Start is used to automatically restart the BLT engine when the BLT width is set. When enabled this bit causes the SPC8110 to start the next BLT operation after the BitBLT Width register is written to. As proportional fonts are not all the same width, setting the BitBLT width will be a common occurrence in character reproduction. By enabling this bit you eliminate the need to explicitly start each BitBLT operation. Used in conjunction with auto update, text display can be performed quite rapidly as the only the width register which has to be updated between characters.

Transparent Background instructs Color Expand BLTs to set only the foreground color when a “1” is found in the data stream. Bits set to “0” in the data stream are left untouched in video memory. This option is intended to overlay text on a graphical image.

Translate is designed for use with 16 color Read and Write BLTs. In translated mode each byte contains two pixels of information. Each nibble of data is rotated into the corresponding plane (i.e., the MSB is placed in plane 3 and the LSB is placed in plane 0). In non-translated mode each DWORD contains four individual planes of information.

Programming The BitBLT Engine

Performing a BitBLT operation consists of setting up the SPC8110 BLT registers and if necessary transferring data to or from video memory. Three of the BitBLT operations (Write, Color Expand and Read) require data transfers. As the SPC8110 is not a bus mastering device these data transfers must be performed by the CPU. The remaining BitBLT types are automatic and, once started, complete on their own.

For the BLTs where the CPU must oversee the data transfer the SPC8110 has a feature which makes the task somewhat easier. The BitBLT engine interprets each video memory cycle and reads/writes the data at the correct location. The actual video memory address used is discarded during BitBLT operations. One result is that all operations could be done with a pointer pointing at the first byte of video memory, A000:0000. Any valid video memory address can be used and the pointer does not have to be updated during the course of operation. Note: On PCI systems faster throughput can be obtained by incrementing the video memory address to engage PCI burst mode.

Programming Notes

This section is a collection of topics related to the operation of the BitBLT engine.

DWORDs

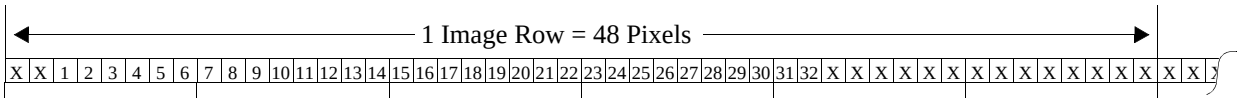
In reference to the SPC8110 BitBLT accelerator the term DWORD can have one of two meanings.

The first refers to the addresses used to access memory. This paper assumes that all addresses lie on natural DWORD boundaries. That is, the address will be evenly divisible by powers of four.

The second usage for the term DWORD refers to the number of 32-bit memory cycles a particular BLT operation requires to complete.When performing Write, Color Expand or Read BLTs each memory cycle must consist of a DWORD, regardless of the display mode. For planar graphics modes this works out to eight pixels of data being transferred per memory cycle. In packed pixel display modes it is four pixels.

The number of DWORDs required to complete each row of a BitBLT operation may not be the result of a division of the width of the image by the number of pixels per DWORD. The actual number of DWORDs required is dependent on the source or destination address as well. This is because the number of DWORDs expected by the BitBLT engine is always the count as seen by the CPU side of the BitBLT after taking natural DWORD boundaries into account. An easy way to envision this requirement is with the following example.

Consider a planar graphics image stored in system memory. The image is 48 pixels wide, we wish to use a Write BLT to copy thirty-two pixels to display memory starting at the third pixel in the source image. An incorrect calculation would lead to the conclusion that four DWORDs are required for each row of the image to be transferred (32 (pixels) / 8 (pixels/DWORD) = 4 DWORDs). As the figure below demonstrates this is not the case.



The figure depicts one plane of the four used by this display mode. Each byte contains eight pixels. The other three bytes of data will, of course, be similarly aligned. Reading one byte in each of four planes gives us the DWORD required by the BitBLT engine. In this example both the partial first and last DWORDs must be transferred to the BitBLT engine. This will require a transfer of five DWORDs. Note that performing the data gymnastics to align the data prior to sending it to the BitBLT engine is unnecessary overhead and should be avoided.

In packed pixel modes it would be possible to 'slip' the system memory address to achieve a more desirable alignment of the data. Highest throughput speed will however be obtained by using natural DWORD addresses.

Below are a series of formulas for calculating the number of DWORDs required to complete an operation where the CPU must read or write video memory.

To determine the number of DWORDs required for planer Write and Color Expand BLTs:

DWORDs = (SrcAlign + Width + 7) / 8

for packed pixel modes the equation is:

DWORDs = (SrcAlign + Width + 3) / 4

where: SrcAlign is the source phase alignment factor as described in the next section.
Width is the width of the BitBLT in pixels.

To determine the number of DWORDs required for planer Read BLTs use:

DWORDs = (DstAlign + Width + 7) / 8

for packed pixel modes the equation is:

DWORDs = (DstAlign + Width + 3) / 4

where: DstAlign is the destination phase alignment factor as described in the next section.
Width is the width of the BitBLT in pixels.

It is likely that for any one BLT the width in pixels is not an even multiple of DWORDs (e.g. a BitBLT operation of 14 pixels wide). In this case the BitBLT engine discards the extra pixels in the last DWORD at the end of each row during Write and Color Expand BLTs. The extra pixels in the last DWORD at the end of each row of a Read BLT are indeterminate and should be ignored.

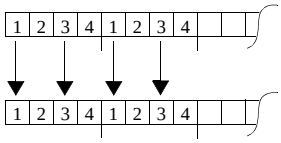
It is important to keep in mind that all BLT operations which involve data transfers between CPU and video memory must consist of 32-bits. The final DWORD in a row may have to padded with dummy data. On a multiple row BLT operation the BitBLT engine does not consider unused data in the last DWORD to extend into the next row. It is up to the programmer to ensure that bitmap images are padded at the end of each logical row to a DWORD width. DIB images in Windows are padded in such a fashion.

It must also be remembered that Read BitBLT operations (video to system memory), will have this padding placed in the last DWORD of each row by the BLT engine. The padding data may have to be stripped out before saving such a bitmap.

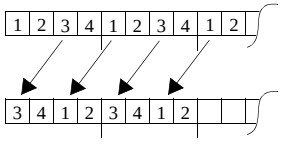
Phase Alignment

Phase alignment refers to the pixel displacement which can be present between source and destination data. The SPC8110 automatically adjusts for phase alignment based on the values written to the BLT source and destination address registers. Having an understanding the alignment is important to 1) calculate the number of DWORDs a BLT operation will require and 2) to aid in understanding what the resulting memory image will be after a BLT operation.

The following diagram depicts the most simple case where there is no offset between the source and the destination pixels. The first pixel of each source DWORD will end up in the first pixel of each destination DWORD. This is a zero phase alignment.



As the situation may have it, within a given transfer it may be that, the source and destination pixels do not align within a DWORD. In the case below the third pixel from the source is being written to the first pixel of the destination. (Note: the BLT engine will automatically perform the alignment. Pixels 1 and 2 would be written to the BLT engine as part of the first DWORD, the BLT engine will discard them as it builds the destination data)



The alignment values which the BLT engine uses may be calculated by performing a modulo division of the absolute pixel location with the number of pixels per DWORD. i.e. *pixel = (row x pels/row) + column*. Therefore to indicate a starting pixel of (column 5, row 0) int mode 12h the calculation would be 0 x 640 + 5 = 5.

In planar display modes this is pixel address modulus 8. For packed pixel display modes this is pixel address modulus 4 (Note: in assembler modulo function scan be performed with the AND instruction using the number of pixels per DWORD less one). These calculations do not have to be done on BLT register data.

e.g., Calculate the source alignment value for a packed pixel mode write BLT from pixel offset five.

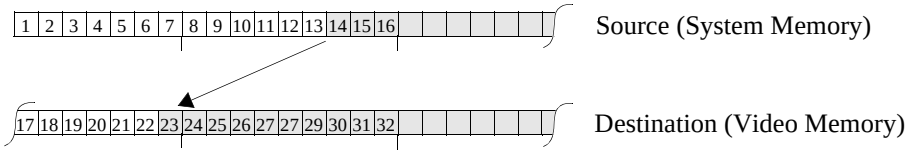
5 % 4 = 1	(modulus)
5 AND 3 = 1	(logical)

e.g., Calculate the destination alignment value for a planar mode read BLT to CPU pixel offset eighteen.

18 % 8 = 2	(modulus)
12h AND 7 = 2	(logical)

For Write and Color Expand BitBLT operations the BLT destination register is set to indicate the first pixel to be written by the data. To indicate a starting pixel of (column 7, row 23) the BLT destination register would be programmed with 14,727 decimal assuming a mode with a width of 640 pixels (23 x 640 + 7). Pixels in the DWORD which precede the starting pixel will not be affected by the write operation.

The source address register specifies where the first valid pixel is to be found in the data stream. As depicted in the diagram below the fourteenth pixel in the source bitmap is to be considered as the first pixel in the data stream passed to the BLT engine. Fourteen will be written to the BLT source register. The net result can be seen in the diagram. The fourteenth pixel in the source bitmap will be placed in the seventh pixel of the destination bitmap. The first DWORD the CPU will read from system memory is the one containing pixels 9 - 16. The data manipulations to correct for phase are performed automatically.

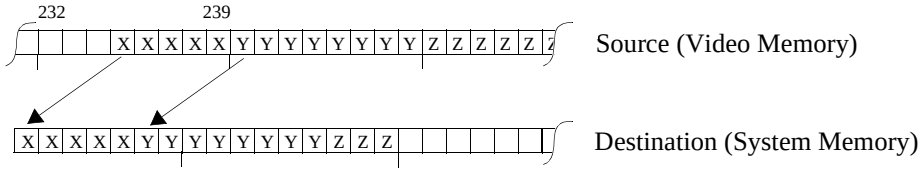


The source alignment value for this BLT is 6 and is required for the calculation involving the number of DWORDs to be passed to the BLT engine. The total number of DWORDs required can be determined by multiplying the number of DWORDs per row by the number of rows in the BLT.

For Read BLT operations the roles of the source and destination registers are reversed. The BLT source register specifies the pixel in display memory where the BLT will start from. The BLT destination register implicitly sets the alignment for the data being read.

For example, a planar mode read BLT the source is to come from the pixel at (235, 0). The destination is an array in system memory. In this case the BLT source register will be set to 235 (there is no Y component to calculate in for the register value). The destination register will be programmed to 0 in order to begin placing pixels at the leftmost position within the system memory DWORD.

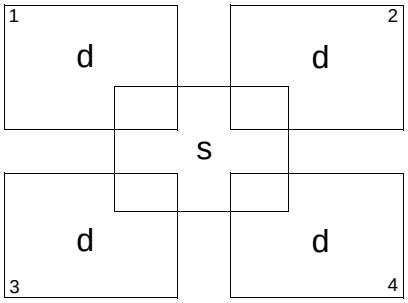
The last five pixels of the DWORD containing the X pattern, in video memory, will be placed in the first five pixels of the first DWORD in system memory. The remaining three pixels will come from the first three pixels from the video memory DWORD with the Y pattern. Remember that on each 'line' of the image in system memory the last DWORD may be padded with indeterminate data.



Overlap

Overlap refers to the situation which occurs when a Move BitBLT is specified and the source bitmap overlaps a portion of the destination. When this happens it is possible for the destination to overwrite the source data before the source data is used.

Normally a BitBLT runs in a forward direction. That is, it starts in the top left corner and proceeds right and down. Referring to the figure below it can be seen that in cases 1 and 2 this presents no difficulties. In cases 3 and 4 such a BitBLT will overwrite the source data before it is used.



The BitBLT engine in the SPC8110 is capable of running in reverse, or decrementing. In this configuration the BitBLT begins in the lower right corner and proceeds to the left and up. All overlap conditions can be handled with the combination of incrementing and decrementing BitBLT directions.

To determine whether a BitBLT should be run incremental or decremental compare the start addresses of the source and destination rectangles. If the source address is greater than the destination address, as in cases 1 and 2, then the BitBLT can be run incrementally. In the case where the source address is less than the destination address, as in 3 and 4, the BitBLT should be run decremental.

e.g., A Move BitBLT transferring data where the source and destination rectangles start at 200, 210 to 100, 110 in packed pixel video mode 101h (640x480x256). Assume the rectangle size to be large enough so that overlap occurs.

In this case:
Source address is 200 * 640 + 210 = 128,210 = 0x1F4D2
Destination address is 100 * 640 + 110 = 64,110 = 0xFA6E.
SrcAddress > DstAddress so the BitBLT direction selected must be incremental.

e.g., A Move BitBLT to transfer a 200x200 pixel rectangle starting at 51,50 to 151,150 in planar video mode 12h (640x480x16)

In this case:
Source address is 50 * 640 + 51 = 32,051 = 0x7D33
Destination address is 150 * 640 + 151 = 96,151 = 0x17797
SrcAddress < DstAddress so the BitBLT direction selected must be decremental.

Protected Mode Programming

Operating systems such as Windows can introduce a very subtle and difficult to trace anomaly during Read BitBLT operations. The strange behavior has to do with the method by which the operating system performs its memory management.

It is not unusual for a protected mode memory manager to allocate but not physically commit memory. When a CPU instruction attempts to access memory outside the committed range an exception is generated. The memory manager is notified and determines whether the memory cycle was to allocated memory, if so more physical memory is committed. The memory manager then instructs the CPU to restart the instruction which caused the exception.

When programing the Read BLT, after all the BLT registers are setup and pointers initialized, the typical way to copy data is through the use of the “REP MOVSD” instruction, however, if the CPU memory being written to causes a memory exception the memory manager will perform the above restart sequence. After physical memory is committed the MOVSD instruction is restarted. The effect of this restart is to generate one extra read cycle to video memory each time this occurs, typically every 4K bytes. The BLT engine has no way of knowing the restarted instruction is a repeat and will count it as a normal read cycle. The result of this sequence of events is that the BLT engine will complete the BLT operation prematurely. In addition the read data will be periodically shifted by the number of pixels per DWORD. In short, garbage data will be read into the CPU buffer.

The way to avoid this is to remove the “REP MOVSD” instruction and replace it with a two step process to perform the transfer as depicted below. Now if a memory exception is thrown the CPU will restart the “MOV [EDI], EAX” instruction and a false read to the BLT engine is not generated.

```
@@:
MOV     EAX, [ESI]
MOV     [EDI], EAX
DEC     ECX
LOOP    @B
```

This condition does not affect Write or Color Expand BLTs as the offending portion of the MOVSD instruction is generated when the CPU attempts to read main memory. This occurs before the attempt to perform a write cycle to video memory.

Known Limitations

There are two known limitations with the SPC8110 BitBLT accelerator.

1. Certain sequences may cause the system to lock-up. During Write, Color Expand or Read BitBLT operations, performing a video memory cycle opposing the direction of the BitBLT will likely cause the bus to hang. For example, the computer will lock-up if video memory is read while performing a Write BLT. There are two situations where this error can occur.

   The first occurs when debugging. While single stepping most debuggers display the contents of the memory location about to be affected by a write operation. If a Write BLT is in progress the read of video memory by the debugger will cause the system to hang. To work around this, step over the procedures and loops which contain BitBLT code.

   The second instance of this limitation can occur in multitasking operating systems. In theory it is conceivable for a second thread of operation to engage the BitBLT engine prior to the BitBLT engine completing its task from a previous operation. This memory access by the second thread could cause the system to hang. While this situation has never been observed, it may be desirable to ensure that it never will. The solution is to test the BitBLT status register prior to allowing any video memory access. If the BitBLT status is found to be busy the routine will have to 'loop', waiting for the status to clear before returning.

2. During the BitBLT operations Write, Color Expand, Move and Read where the width is equal to or less than the number of pixels per DWORD, 8 pixels in planar and 4 pixels in packed modes, the SPC8110 may hang the bus.

   Below are the exact cases where the BitBLT engine fails. The error conditions have been separated into planar and packed pixel cases. As the BitBLT engine relies on 32-bit transfers internally, the two descriptions below show the same conditions. Note the decrement direction applies to Move BitBLTs only, as all other types are presumed to be increment BitBLTs.

   Planar Mode (4bpp) Writes, Color Expands, Moves and Reads

   Direction Positive (increment)
   - SourceAlignment + Width <= 8
   - SourceAlignment > DestinationAlignment

   Direction Negative (decrement)
   - DestinationAlignment + Width <= 8
   - SourceAlignment <= DestinationAlignment

   Packed Pixel Mode (8bpp) Writes, Color Expands, Moves and Reads

   Direction Positive (increment)
   - SourceAlignment + Width <= 4
   - SourceAlignment > DestinationAlignment

   Direction Negative (decrement)
   - DestinationAlignment + Width <= 4
   - SourceAlignment <= DestinationAlignment

   Any BLTs which fit into the conditions above must be handled by using standard memory access techniques.

The following sample code contains examples to detect both limitations. These routines are IsBltOkay and CheckBltStatus.

Program Examples

The following program examples contain code for both Planer and Packed Pixel video modes.

BLT-SAMP.C - Sample Programming Code for the SPC8110

```
#pragma warning(disable:4001) // Disable the 'single line comment' warning.

//=============================================================================
//  BLT-SAMP.C - Sample Programming Code for the SPC8110
//-----------------------------------------------------------------------------
//  The code contained in this file demonstrates the basics of programming
//  the BLT engine on the SPC8110.
//
//  This code has been compiled and tested with the MS C/C++ 8.00c compiler.
//  In addition to this file the assembler support file 'bltasm.asm'
//  is required.
//-----------------------------------------------------------------------------

#include <graph.h>
#include <stdio.h>


//=============================================================================
//  CONSTANTS, TYPDEFS
//=============================================================================
#define BLUE                    0x01
#define RED                     0x04
#define WHITE                   0x0F

#define WRITE                   0
#define CLR_EX                  1
#define FILL                    2
#define MOVE                    3
#define PATTERN                 4
#define READ                    5
#define INVERT                  6

#define LINEAR                  0
#define SQUARE                  1

#define PACKED                  0
#define SQUARE                  1

#define DIR_POS                 0
#define DIR_NEG                 1

#define FALSE                   0
```

```
#define TRUE                    1

#define       BLT_PLANER        0
#define       BLT_PACKED        1

typedef unsigned char           BYTE;
typedef unsigned short          WORD;
typedef unsigned long           DWORD;
typedef int                     BOOL;

typedef BYTE  far * LPBYTE;
typedef DWORD far * LPDWORD;


#define _MK_FP(seg, offset) (void __far *)(((unsigned long)seg << 16) + \
                                (unsigned long)(unsigned)offset)

//=========================================================================
//  PROTOTYPES for the functions in BLTASM.ASM
//=========================================================================
BYTE asmIOR8 ( WORD wPort );
void asmIOW8 ( WORD wPort, BYTE byValue );
void asmIOW32( WORD wPort, DWORD dwValue );
void asmMoveDWs( LPDWORD lpDst, LPDWORD lpSrc, int nCount );
void asmPackedColorExpand( void );
void asmPlanarColorExpand( void );
void asmSetDAC( void );


//=========================================================================
//  GLOBAL DATA
//=========================================================================
DWORD dwReadBltData[50];        // Buffer to hold data for read blts.

WORD  BytesScanLine;            // The number of bytes on each row of the display.
BOOL  BltType;                  // PACKED or PLANER
BOOL  SrcType;                  // LINEAR or SQUARE
BOOL  DstType;                  // LINEAR or SQUARE
```

```
////////////////////////////////////////////////////////////////////////
// CheckBltStatus
//
//  This routine can be used in two different situations.
//
//  1) After using a BitBLT operation call CheckBltStatus to wait
//     for the blt to complete.
//
//  2) Call before comencing any video memory operation. This ensures
//              that it is safe to access video memory.
//
// NOTE:
//  Used in the post blt mode this test can determine when to return
//    to the caller. For blts where data is transferred between system
//  and video memory this test is at best an assurance.If we overran
//  the expected data count we will never know. If we underrun the
//  expected count we will never return!!!
////////////////////////////////////////////////////////////////////////
void CheckBltStatus( void )
{
      while (0x80 == (0x80 & asmIOR8( 0x3D0 )));
}


////////////////////////////////////////////////////////////////////////
//  GetMoveDirection - determines whether a move blt should be run in the
//                     positive or negative direction. This routine relies
//                     on the state of global variables. (in this example)
////////////////////////////////////////////////////////////////////////

BOOL CalcDirection( DWORD dwBltSrc, DWORD dwBltDst, DWORD dwWidth, DWORD dwHeight )
{
      //  This calculation requires knowledge of several items which
      //  haven't been passed as arguements anywhere else. I've elected
      //  to make them globals for purposes of explanation. Their actual
      //  values will be modified in the main() procedure.
      //
      //  BytesScanLine - the number of bytes on earch row of the display.
      //  BltType     - PACKED or PLANER
      //  SrcType     - LINEAR or SQUARE
      //  DstType     - LINEAR or SQUARE

      int wPelPerDWminus1;
      DWORD dwSrcOffset, dwDstOffset;

      //  In planar graphics there are eight pels/DW otherwise we have four pels/DW.
      if (BLT_PLANER == BltType)
               wPelPerDWminus1 = 7;
      else
```

```
        wPelPerDWminus1 = 3;

//  Moves can consist of linear or rectangular sourc and destination
//  rectangles. The determination of where the end of the blt area
//  is changes according to the shape of the source and destinations.

if (LINEAR == SrcType)
        dwSrcOffset = (dwBltSrc & wPelPerDWminus1) + dwWidth +
                        wPelPerDWminus1;
else
        dwSrcOffset = (dwHeight - 1) * BytesScanLine + (dwWidth - 1);

if (LINEAR == DstType)
        dwDstOffset = (dwBltDst & wPelPerDWminus1) + dwWidth +
                        wPelPerDWminus1;
else
        dwDstOffset = (dwHeight - 1) * BytesScanLine + (dwWidth - 1);

// Normalize the blt starting offsets. This is done by subtracting the
// smaller from the larger value then zeroing the smaller value.
if (dwBltSrc < dwBltDst)
{
        dwBltDst -= dwBltSrc;
        dwBltSrc = 0;
}
else
{
        dwBltSrc -= dwBltDst;
        dwBltDst = 0;
}

//  Determine the blt direction based on source/destination overlaps.
//  The general algorythm looks like this:
//
//  if (Src > Dst)
//      {
//              if (LastSrc > LastDst)
//                      +ve
//              else
//                      -ve
//      }
//      else
//      {
//              if (LastSrc <= LastDst)
//                      -ve
//              else
//                      +ve
//      }
```

```
	if (dwBltSrc > dwBltDst)
	{
		if ((dwBltSrc + dwSrcOffset) > (dwBltDst + dwDstOffset))
			return DIR_POS;
		else
			return DIR_NEG;
	}
	else
	{
		if ((dwBltSrc + dwSrcOffset) <= (dwBltDst + dwDstOffset))
			return DIR_NEG;
		else
			return DIR_POS;
	}
}

////////////////////////////////////////////////////////////////////////////
//  IsBltOkay()
//
//  There are certain blt operations for which data must be checked to
//  ensure the blt engine can handle it.
//
//  This routine returns TRUE if the blt engine can handle the parameters
//  and FALSE if the blt must be special cased.
////////////////////////////////////////////////////////////////////////////

BOOL IsBltOkay( WORD wRop, DWORD dwSrc, DWORD dwDst, DWORD dwWidth, DWORD dwHeight )
{
	DWORD dwSrcAlign = dwSrc & 3;
	DWORD dwDstAlign = dwDst & 3;

	switch ( wRop )
	{
		case WRITE:
		case CLR_EX:
		case READ:
			if ((dwSrcAlign + dwWidth <= 8) && (dwSrcAlign > dwDstAlign))
				return FALSE;
			break;

		case MOVE:
			if (DIR_NEG == CalcDirection( dwSrc, dwDst, dwWidth, dwHeight
))
			{
				if ((dwDstAlign + dwWidth <= 8) && (dwSrcAlign <=
dwDstAlign))
				return FALSE;
			}
			else
```

```
                        {
                                if ((dwSrcAlign + dwWidth <= 8) &&
                                      (dwSrcAlign > dwDstAlign))
                                return FALSE;
                        }
                        break;

                case PATTERN:
                case INVERT:
                case FILL:
                default:
                        break;
        }

        return TRUE;
}

////////////////////////////////////////////////////////////////////////////
// DoPackedWriteBlt
//   This routine:
//    - BLTs a 10w x 10h white smiley on a red background to 200,210.
//      In this example I expect the right edge of the destination
//      to have the same phase relationship as the source.
//      During the BLT the video memory pointer is not updated in order
//      to demonstrate that it is not required. On a PCI bus better
//      performance can be achieved by updating this pointer and
//      thereby engaging PCI burst write mode.
////////////////////////////////////////////////////////////////////////////

// Data pattern for the smiley face.
// The periods are colored red (0x04) and the X's are white (0xF).
// As this is intel architecture the byte had to be DWORD rearranged.

DWORD PackedWrtData[] =
{
        0x0F040404, 0x040F0F0F, 0x00000404,// ...XXXX...
        0x040F0404, 0x0F040404, 0x00000404,// ..X....X..
        0x04040F04, 0x04040404, 0x0000040F,// .X......X.
        0x0F04040F, 0x040F0404, 0x00000F04,// X..X..X..X
        0x0404040F, 0x04040404, 0x00000F04,// X........X
        0x0F04040F, 0x040F0404, 0x00000F04,// X..X..X..X
        0x0404040F, 0x04040F0F, 0x00000F04,// X...XX...X
        0x04040F04, 0x04040404, 0x0000040F,// .X......X.
        0x040F0404, 0x0F040404, 0x00000404,// ..X....X..
        0x0F040404, 0x040F0F0F, 0x00000404 // ...XXXX...
};
```

```
void DoPackedWriteBlt( void )
{
      DWORD dwTmp;
      int   nDwords;
      DWORD dwSrc, dwDst, dwWidth, dwHeight;

      // First check that we can safely write to video memory.
      // NOTE: this only needs to be done in a multi-tasking
      //       environment where there may already be a blt
      //       operation in progress.
      CheckBltStatus();

      // Check that the blt doesn't have to be special cased.
      dwSrc    = 0;
      dwDst    = (210 * BytesScanLine) + 200;
      dwWidth  = 10;
      dwHeight = 10;

      if (FALSE == IsBltOkay( WRITE, dwSrc, dwDst, dwWidth, dwHeight))
      {
              // If the test failed then we have to handle the blt using
              // standard video memory access techniques.
              // It's a small blt so it won't take much time.
      }

      // Set-up the BLT registers.
      // Do the setup in reverse order. This ensures the auto-start
      // bit in register 5 is reset. If not done this way the blter may
      // engage after setting the width field in register 3.

      // Set the Blt Function register.
      // - reset auto-start
      asmIOW32( 0x3D0, 0x00000005 );

      // Set the Blt Description register.
      // - only the ROP is required - 0.
      asmIOW32( 0x3D0, 0x00000004 );

      // Set the Blt Width and Foreground Color register.
      // The smiley image is 10 pixels wide.
      // Note: Write BLTs do not use foreground color.
      dwTmp = ((DWORD)(10 - 1) << 16) | 3;
      asmIOW32( 0x3D0, dwTmp );

      // Set the Blt Height and Background Color register.
      // The smiley image is 10 pixels high.
      // Note: Write BLTs do not use Background color.
      dwTmp = ((DWORD)(10 - 1) << 16) | 2;
      asmIOW32( 0x3D0, dwTmp );
```

```
	// Set the Destination Address register. (200,210)
	dwTmp = ((210 * 640l) + 200) << 8 | 1;
	asmIOW32( 0x3D0, dwTmp );

	// Set the Source Address register (In this case no alignment to set)
	asmIOW32( 0x3D0, 0x00000000 );

	// Engage the BLT engine.
	asmIOW8( 0x3D0, 0x80 );

	// Calculate the number of DWORDs required.
	// DWORDs = Rows * ((SrcAlign + Width + (pels/DWORD - 1)) / (pels/DWORD))
	nDwords =  10 * ((0 + 10 + 3) / 4);

	asmMoveDWs( _MK_FP( 0xA000, 0x0000 ), (LPDWORD)&PackedWrtData, nDwords );

	CheckBltStatus();
}

////////////////////////////////////////////////////////////////////////////
// DoPackedColorExpandBlt
//    This routine:
//    - use color expand to draw the characters A-Z to co-ordinates 0,200
////////////////////////////////////////////////////////////////////////////

void DoPackedColorExpandBlt( void )
{
	DWORD dwTmp;
	DWORD dwSrc, dwDst, dwWidth, dwHeight;

	// First check that we can safely write to video memory.
	// NOTE: this only needs to be done in a multi-tasking
	//       environment where there may already be a blt
	//       operation in progress.
	CheckBltStatus();

	// Check that the blt doesn't have to be special cased.
	dwSrc    = 0;
	dwDst    = (200 * BytesScanLine) + 0;
	dwWidth  = 26 * 8;
	dwHeight = 16;

	if (FALSE == IsBltOkay( CLR_EX, dwSrc, dwDst, dwWidth, dwHeight))
	{
		// If the test failed then we have to handle the blt using
		// standard video memory access techniques.
		// It's a small blt so it won't take much time.
	}
```

```
        // Set the Blt Function register.
        // - set auto-update and auto-start.
        asmIOW32( 0x3D0, 0x00003005 );

        // Set the Blt Description register.
        // - only the ROP is required - 1.
        asmIOW32( 0x3D0, 0x00000104 );

        // Set the Blt Width and Foreground Color register.
        // - we will use the 8x16 font so the width is 8 - 1.
        // - set the foreground color to WHITE.
        dwTmp = ((DWORD)(8 - 1) << 16) | WHITE << 8 | 3;
        asmIOW32( 0x3D0, dwTmp );

        // Set the Blt Height and Background Color register.
        // - we will use the 8x16 font so the height is 16 - 1.
        // - set the background color to RED.
        dwTmp = ((DWORD)(16 - 1) << 16) | (WORD)RED << 8 | 2;
        asmIOW32( 0x3D0, dwTmp );

        // Set the Destination Address register. (0,200)
        dwTmp = (200 * 640l) << 8 | 1;
        asmIOW32( 0x3D0, dwTmp );

        // Set the Source Address register.
        // - set to 0, no phase alignment.
        asmIOW32( 0x3D0, 0x00000000 );

        // Engage the BLT engine.
        // NOTE: Register index 3 is being specifically SET here. Without this
        //       the assembler code will fail to auto-start!
        asmIOW8( 0x3D0, 0x83 );

        // The technique to BLT characters is clearer in assembly.
        // The code is in the asm support file rather than use in-line assembler.
        asmPackedColorExpand();

        // Do an assurance test on the BLT status.
        CheckBltStatus();
}
```

```
////////////////////////////////////////////////////////////////////////
// DoPackedSolidFillBlt
//    This routine:
//    - BLTs a 100w x 100h red rectangle to screen co-ordinates 200,210.
////////////////////////////////////////////////////////////////////////

void DoPackedSolidFillBlt( void )
{
      DWORD dwTmp;

      // First check that we can safely write to video memory.
      // NOTE: this only needs to be done in a multi-tasking
      //       environment where there may already be a blt
      //       operation in progress.
      CheckBltStatus();

      // Set-up the blter registers.
      // The setup is done in reverse order. This ensures the auto-start
      // bit in register 5 is reset. If this is not done the blter may
      // engage after setting the width field in register 3.

      // Set the Blt Function register.
      // Ensure auto-start is reset.
      asmIOW32( 0x3D0, 0x00000005 );

      // Set the Blt Description register.
      // - only the ROP has to be set - 2.
      asmIOW32( 0x3D0, 0x00000204 );

      // Set the Blt Width and Foreground Color register.
      dwTmp = ((DWORD)(100 - 1) << 16) | RED << 8 | 3;
      asmIOW32( 0x3D0, dwTmp );

      // Set the Blt Height and Background Color register.
      // - Solid fill does not use the background color.
      dwTmp = ((DWORD)(100 - 1) << 16) | 2;
      asmIOW32( 0x3D0, dwTmp );

      // Set the Destination Address register. (location 200,210)
      dwTmp = ((210 * 640l) + 200) << 8 | 1;
      asmIOW32( 0x3D0, dwTmp );

      // Set the Source Address register.
      // - solid fill does not actually require this.
      asmIOW32( 0x3D0, 0x00000000 );

      // Engage the BLT engine.
      asmIOW8( 0x3D0, 0x80 );
```

```
      // Wait till the BLT completes.
      CheckBltStatus();
}


////////////////////////////////////////////////////////////////////////
// DoPackedMoveBlt
//    This routine:
//    - calls DoPackedWriteBlt() to draw a smiley to 210,200.
//    - uses the BLT engine to copy the image to location 300,300
//      Because the result is going to be visible on screen I am not
//      engaging either Src-Linear Select or Dst-Linear Select.
//      There is no overlap of the source and destination
//      so the BLT is run in an increment direction.
////////////////////////////////////////////////////////////////////////

void DoPackedMoveBlt( void )
{
      DWORD dwTmp;
      DWORD dwSrc, dwDst, dwWidth, dwHeight;

      // First check that we can safely write to video memory.
      // NOTE: this only needs to be done in a multi-tasking
      //       environment where there may already be a blt
      //       operation in progress.
      CheckBltStatus();

      // Check that the blt doesn't have to be special cased.
      dwSrc    = 0;
      dwDst    = (210 * BytesScanLine) + 200;
      dwWidth  = 10;
      dwHeight = 10;

      if (FALSE == IsBltOkay( MOVE, dwSrc, dwDst, dwWidth, dwHeight))
      {
              // If the test failed then we have to handle the blt using
              // standard video memory access techniques.
              // It's a small blt so it won't take much time.
      }

      // Use the Write blt to make the first smiley on the display.
      DoPackedWriteBlt();

      // Set-up the blter registers.
      // The setup is done in reverse order. This ensures the auto-start
      // bit in register 5 is reset. If not done this way the blter may
      // engage after setting the width field in register 3.
```

```
      // Set the Blt Function register.
      // - ensure auto-start is reset.
      asmIOW32( 0x3D0, 0x00000005 );

      // Set the Blt Description register.
      // - the ROP for MOVE BLT is 3.
      // - none of the move modifiers need be set.
      asmIOW32( 0x3D0, 0x00000304 );

      // Set the Blt Width and Foreground Color register.
      // - move BLT does not use foreground color.
      dwTmp = ((DWORD)(10 - 1) << 16) | 3;
      asmIOW32( 0x3D0, dwTmp );

      // Set the Blt Height and Background Color register.
      // - move BLT does not use background color.
      dwTmp = ((DWORD)(10 - 1) << 16) | 2;
      asmIOW32( 0x3D0, dwTmp );

      // Set the Destination Address register. (dst = 300,300)
      dwTmp = ((300 * 640l) + 300) << 8 | 1;
      asmIOW32( 0x3D0, dwTmp );

      // Set the Source Address register. (210,200 from DoPackeWriteBlt())
      dwTmp = ((210 * 640l) + 200) << 8 | 0;
      asmIOW32( 0x3D0, dwTmp );

      // Engage the BLT engine.
      asmIOW8( 0x3D0, 0x80 );

      // Wait for the BLT to complete.
      CheckBltStatus();
}


////////////////////////////////////////////////////////////////////////
// DoPackedPatternFillBlt
// This routine:
// - Uses a Write BLT sequence to place a pattern into off-screen memory.
// - Uses the pattern BLT to replicate the pattern over a 200 x 200 area
//   starting at 200,210.
////////////////////////////////////////////////////////////////////////

// Data pattern used for this BLT.
// The periods are colored red (0x04) and the X's are white (0xF).
// As this is intel architecture the bytes had to be DWORD rearranged.
```

```
DWORD PackedPatternData[] =
{
      0x0F040404, 0x04040404,  // ...X....
      0x04040404, 0x0404040F,  // ....X...
      0x04040404, 0x0F040F04,  // .....X.X
      0x0404040F, 0x040F0404,  // X.....X.
      0x04040F04, 0x04040F04,  // .X...X..
      0x040F0404, 0x0404040F,  // ..X.X...
      0x0F040404, 0x04040404,  // ...X....
      0x040F0404, 0x04040404,  // ..X.....
};

void DoPackedPatternFillBlt( void )
{
      DWORD dwTmp;
      int   nDwords;

      // First check that we can safely write to video memory.
      // NOTE: this only needs to be done in a multi-tasking
      //       environment where there may already be a blt
      //       operation in progress.
      CheckBltStatus();

      // The first thing that must be done is a pattern has to be loaded into
      // video memory. A Write BLT will be used to do put a pattern at 0,480
      // (the first byte of off-screen video memory). The steps to do the
      // Write BLT are the same as DoPackedWriteBlt() uses.

      asmIOW32( 0x3D0, 0x00000005 );//
      asmIOW32( 0x3D0, 0x00000004 );// write BLT.
      asmIOW32( 0x3D0, 0x003F0003 );// 64 pixels.
      asmIOW32( 0x3D0, 0x00000002 );// 1 line.
      asmIOW32( 0x3D0, ((480 * 640l)) << 8 | 1 );
      asmIOW32( 0x3D0, 0x00000000 );
      asmIOW8( 0x3D0, 0x80 );  // start the BLT
      nDwords =  8 * ((8 + 3) / 4);
      asmMoveDWs( _MK_FP(0xA000, 0x0000), (LPDWORD)&PackedPatternData, nDwords );

      // The pattern is now in off-screen video memory.

      // Set-up the blter registers.
      // The setup is done in reverse order. This ensures the auto-start
      // bit in register 5 is reset. If not done this way the blter may
      // engage after setting the width field in register 3.

      // Set the Blt Function register.
      // - ensure auto-start is reset.
      asmIOW32( 0x3D0, 0x00000005 );
```

```
      // Set the Blt Description register.
      // - pattern fill ROP is 4.
      asmIOW32( 0x3D0, 0x00000404 );

      // Set the Blt Width and Foreground Color register.
      // - pattern fill does not use foreground color.
      dwTmp = ((DWORD)(200 - 1) << 16) | 3;
      asmIOW32( 0x3D0, dwTmp );

      // Set the Blt Height and Background Color register.
      // - pattern fill does not use background color.
      dwTmp = ((DWORD)(200 - 1) << 16) | 2;
      asmIOW32( 0x3D0, dwTmp );

      // Set the Destination Address register. (Dst = 200,210)
      dwTmp = ((210 * 640l) + 200) << 8 | 1;
      asmIOW32( 0x3D0, dwTmp );

      // Set the Source Address register.
      // - in this case set to the starting pixel of our pattern.
      dwTmp = (480 * 640l) << 8 | 0;
      asmIOW32( 0x3D0, dwTmp );

      // Engage the BLT engine.
      asmIOW8( 0x3D0, 0x80 );

      // Wait for the BLT to complete.
      CheckBltStatus();
}


////////////////////////////////////////////////////////////////////////////
// DoPackedReadBlt
//    This routine:
//    - Calls DoPackedWriteBlt() to draw a 10x10 smiley at 200,210.
//    - Uses a Read BLT to read the data back into a memory array.
//    - Compares the results and reports any errors.
////////////////////////////////////////////////////////////////////////////

void DoPackedReadBlt( void )
{
      DWORD dwTmp;              //
      int nDwords, nRow;
      LPDWORD lpdwRead = dwReadBltData;
      LPDWORD lpdwWrt = PackedWrtData;
      DWORD dwSrc, dwDst, dwWidth, dwHeight;

      // First check that we can safely write to video memory.
      // NOTE: this only needs to be done in a multi-tasking
```

```
//       environment where there may already be a blt
//       operation in progress.
CheckBltStatus();

// Check that the blt doesn't have to be special cased.
dwSrc    = (210 * BytesScanLine) + 200;
dwDst    = 0;
dwWidth  = 10;
dwHeight = 10;

if (FALSE == IsBltOkay( READ, dwSrc, dwDst, dwWidth, dwHeight))
{
        // If the test failed then we have to handle the blt using
        // standard video memory access techniques.
        // It's a small blt so it won't take much time.
}

// We need something to read. Call DoPackedWriteBlt() to give us
// the smiley face at 200,210.
DoPackedWriteBlt();

// The smiley is in video memory. We are now ready to read it.

// Set-up the BLT registers.
// The setup is done in reverse order. This ensures the auto-start
// bit in register 5 is reset. If not done this way the BLT may
// engage after setting the width field in register 3.

// Set the Blt Function register.
// - ensure auto-start is reset.
asmIOW32( 0x3D0, 0x00000005 );

// Set the Blt Description register.
// - read BLT ROP is 5.
asmIOW32( 0x3D0, 0x00000504 );

// Set the Blt Width and Foreground Color register.
// - the smiley is 10 pixels wide.
// - read BLT does not use foreground color.
dwTmp = ((DWORD)(10 - 1) << 16) | 3;
asmIOW32( 0x3D0, dwTmp );

// Set the Blt Height and Background Color register.
// - the smiley is 10 pixels high.
// - read BLT does not use background color.
dwTmp = ((DWORD)(10 - 1) << 16) | 2;
asmIOW32( 0x3D0, dwTmp );
```

```
	// Set the Destination Address register.
	// - read into position 0 in the buffer.
	// (This doesn't demonstrate phase alignment but it will make
	// it easier to compare the buffer to the expected values)
	asmIOW32( 0x3D0, 1 );

	// Set the Source Address register. (Src = 200,210)
	dwTmp = ((210 * 640l) + 200) << 8 | 0;
	asmIOW32( 0x3D0, dwTmp );

	// Engage the BLT engine.
	asmIOW8( 0x3D0, 0x80 );

	// Calculate the number of DWORDs required for the read.
	// DWORDs = Rows * ((DstAlign + Width + (pels/DWORD - 1)) / (pels/DWORD))
	nDwords =  10 * ((0 + 10 + 3) / 4);

	asmMoveDWs( (LPDWORD)&dwReadBltData, _MK_FP(0xA000, 0x0000), nDwords );

	// Do an assurance check on the BLT status.
	CheckBltStatus();

	// The first 10 pixels of data in dwReadBltData and PackedWrtData should
	// match. The unused portion of each row end DWORD likely will not.
	for (nRow = 0; nRow < 10; nRow++ )
	{
		if (*lpdwWrt != *lpdwRead)
			printf( "Error in packed Read BLT compare!\n" );
		lpdwWrt++;
		lpdwRead++;

	if (*lpdwWrt != *lpdwRead)
		printf( "Error in packed Read BLT compare!\n" );
		lpdwWrt++;
		lpdwRead++;

		if ((*lpdwWrt & 0xFFFF0000) != (*lpdwRead & 0xFFFF0000))
			printf( "Error in packed Read BLT compare!\n" );
		lpdwWrt++;
		lpdwRead++;
	}
}
```

```
////////////////////////////////////////////////////////////////////////////
// DoPackedInvertBlt
//    This routine:
//    - Calls DoPackedPatternBlt() to display a pattern on screen.
//    - Inverts the pattern area.
//    - Note: two DAC entries must be set so the inverted image looks correct.
////////////////////////////////////////////////////////////////////////////

void DoPackedInvertBlt( void )
{
      DWORD dwTmp;

      // First check that we can safely write to video memory.
      // NOTE: this only needs to be done in a multi-tasking
      //       environment where there may already be a blt
      //       operation in progress.
      CheckBltStatus();

      // There is one little snag. The default BIOS setting for mode 101h
      // leaves the last eight DAC entries blank. The inverted image will
      // look rather strange. Set DAC entries 0xF0 to BLACK and entry
      // 0xFB to CYAN to correct for this.
      asmSetDAC();

      // We need something to invert. Call the pattern BLT routine to
      // give us a 200 x 200 pattern at 200,210.
      DoPackedPatternFillBlt();

      // The pattern is in video memory. We are now ready to invert it.

      // Set-up the blter registers.
      // The setup is done in reverse order. This ensures the auto-start
      // bit in register 5 is reset. If not done this way the blter may
      // engage after setting the width field in register 3.

      // Set the Blt Function register.
      // - ensure auto-start is reset.
      asmIOW32( 0x3D0, 0x00000005 );

      // Set the Blt Description register.
      // - invert BLT ROP is 6.
      asmIOW32( 0x3D0, 0x00000604 );

      // Set the Blt Width and Foreground Color register.
      // - invert BLT does not use foreground color.
      dwTmp = ((DWORD)(200 - 1) << 16) | 3;
      asmIOW32( 0x3D0, dwTmp );
```

```
      // Set the Blt Height and Background Color register.
      // - invert BLT does not use background color.
      dwTmp = ((DWORD)(200 - 1) << 16) | 2;
      asmIOW32( 0x3D0, dwTmp );

      // Set the Destination Address register. (address = 200,210)
      dwTmp = ((210 * 640l) + 200) << 8 | 1;
      asmIOW32( 0x3D0, dwTmp );

      // Set the Source Address register.
      // - invert BLT does not actually require this to be set.
      asmIOW32( 0x3D0, 0 );

      // Engage the BLT engine.
      asmIOW8( 0x3D0, 0x80 );

      // Wait for the BLT to complete.
      CheckBltStatus();
}

////////////////////////////////////////////////////////////////////////////
// DoPlanarWriteBlt
//    This routine:
//    - BLTs a 10w x 10h white smiley on a blue background to 100,110.
//      In this example I expect the right edge of the destination
//      rectangle to have the same phase relationship as the source alignment.
//      Pixel data is in packed format. In this format each byte contains two
//      pixels. The blter will handle rotating each byte into the correct plane.
//      The video memory pointer is not updated to demonstrate that it is not
//      essential. On PCI bus systems better performance can be achieved by
//      updating this pointer and thereby engaging PCI burst write mode.
////////////////////////////////////////////////////////////////////////////

// Data for the smiley. This data has been assembled as packed planar bytes.
// The periods are colored blue (0x1) and the X's are white (0xF).
// As this is intel architecture the bytes had to be DWORD rearranged.
DWORD PlanarWrtData[] =
{
      0xF1FF1F11, 0x44444411,  // ...XXXX...
      0x1F11F111, 0x44444411,  // ..X....X..
      0x1111111F, 0x444444F1,  // .X......X.
      0xF1111FF1, 0x4444441F,  // X..X..X..X
      0x111111F1, 0x4444441F,  // X........X
      0xF1111FF1, 0x4444441F,  // X..X..X..X
      0x11FF11F1, 0x4444441F,  // X...XX...X
      0x1111111F, 0x444444F1,  // .X......X.
      0x1F11F111, 0x44444411,  // ..X....X..
      0xF1FF1F11, 0x44444411   // ...XXXX...
};
```

```
void DoPlanarWriteBlt( void )
{
      DWORD dwTmp;
      int   nDwords;
      DWORD dwSrc, dwDst, dwWidth, dwHeight;

      // First check that we can safely write to video memory.
      // NOTE: this only needs to be done in a multi-tasking
      //       environment where there may already be a blt
      //       operation in progress.
      CheckBltStatus();

      // Check that the blt doesn't have to be special cased.
      dwSrc    = 0;
      dwDst    = (110 * BytesScanLine) + 100;
      dwWidth  = 10;
      dwHeight = 10;

      if (FALSE == IsBltOkay( WRITE, dwSrc, dwDst, dwWidth, dwHeight))
      {
              // If the test failed then we have to handle the blt using
              // standard video memory access techniques.
              // It's a small blt so it won't take much time.
      }

      // Set-up the blter registers.
      // The setup is done in reverse order. This ensures the auto-start
      // bit in register 5 is reset. If not done this way the blter may
      // engage after setting the width field in register 3.

      // Set the Blt Function register.
      // - ensure auto-start is reset and enable translates.
      asmIOW32( 0x3D0, 0x00000105 );

      // Set the Blt Description register.
      // - write BLT ROP is 0.
      asmIOW32( 0x3D0, 0x00000004 );

      // Set the Blt Width and Foreground Color register.
      // - write BLT does not use foreground color.
      asmIOW32( 0x3D0, ((long)(10 - 1) << 16) | 3 );

      // Set the Blt Height and Background Color register.
      // - write BLT does not use background color.
      asmIOW32( 0x3D0, ((long)(10 - 1) << 16) | 2 );
```

```
      // Set the Destination Address register. (Dst = 100,100)
      dwTmp = ((110 * 640l) + 100) << 8 | 1;
      asmIOW32( 0x3D0, dwTmp );

      // Set the Source Address register.
      // - in this case no source alignment to be set.
      asmIOW32( 0x3D0, 0x00000000 );

      // Engage the BLT engine.
      asmIOW8( 0x3D0, 0x80 );

      // Calculate the number of DWORDs required for this BLT.
      //      = Rows * ((SrcAlign + Width + (pels/DWORD - 1)) / (pels/DWORD))
      nDwords = 10 * ((0 + 10 + 7) / 8);

      asmMoveDWs( _MK_FP( 0xA000, 0x0000 ), (LPDWORD)&PlanarWrtData, nDwords );

      // Verify we are done.
      CheckBltStatus();
}

////////////////////////////////////////////////////////////////////////////
// DoPlanarColorExpandBlt
//    This routine:
//    - uses the BLT engine to draw the first 26 uppercase letters to 0,200
////////////////////////////////////////////////////////////////////////////

void DoPlanarColorExpandBlt( void )
{
      DWORD dwTmp;
      DWORD dwSrc, dwDst, dwWidth, dwHeight;

      // First check that we can safely write to video memory.
      // NOTE: this only needs to be done in a multi-tasking
      //       environment where there may already be a blt
      //       operation in progress.
      CheckBltStatus();

      // Check that the blt doesn't have to be special cased.
      dwSrc    = 0;
      dwDst    = (210 * BytesScanLine) + 0;
      dwWidth  = 26;
      dwHeight = 16;

      if (FALSE == IsBltOkay( CLR_EX, dwSrc, dwDst, dwWidth, dwHeight))
      {
```

```
            // If the test failed then we have to handle the blt using
            // standard video memory access techniques.
            // It's a small blt so it won't take much time.
      }

      // Set-up the blter registers.

      // Set the Blt Function register.
      // - set auto-update and auto-start.
      asmIOW32( 0x3D0, 0x00003005 );

      // Set the Blt Description register.
      // - color expand ROP is 1.
      asmIOW32( 0x3D0, 0x00000104 );

      // Set the Blt Width and Foreground Color register.
      // - we will use the 8x16 font so the width is 8 - 1.
      // - set the foreground color to WHITE.
      dwTmp = ((DWORD)(8 - 1) << 16) | WHITE << 8 | 3;
      asmIOW32( 0x3D0, dwTmp );

      // Set the Blt Height and Background Color register.
      // - we will use the 8x16 font so the height is 16 - 1.
      // - set the background color to BLUE.
      dwTmp = ((DWORD)(16 - 1) << 16) | BLUE << 8 | 2;
      asmIOW32( 0x3D0, dwTmp );

      // Set the Destination Address register. (Dst = 0,200)
      dwTmp = (200 * 640l) << 8 | 1;
      asmIOW32( 0x3D0, dwTmp );

      // Set the Source Address register.
      // - color expand uses this value for phase alignment.
      asmIOW32( 0x3D0, 0x00000000 );

      // Engage the BLT engine.
      // NOTE: Register index 3 is being specifically SET here. Without this
      //       the assembler code will fail to auto-start!
      asmIOW8( 0x3D0, 0x83 );

      // The technique to BLT characters is clearer in assembly.
      // The code is in the asm support file rather than use in-line assembler.
      asmPlanarColorExpand();

      // Do an assurance test.
      CheckBltStatus();
}
```

```
////////////////////////////////////////////////////////////////////////////
// DoPlanarSolidFillBlt
//    This routine:
//    - BLTs a 200w x 200h blue rectangle to screen co-ordinates 100,110.
////////////////////////////////////////////////////////////////////////////

void DoPlanarSolidFillBlt( void )
{
      DWORD dwTmp;

      // First check that we can safely write to video memory.
      // NOTE: this only needs to be done in a multi-tasking
      //       environment where there may already be a blt
      //       operation in progress.
      CheckBltStatus();

      // Set-up the blter registers.
      // The setup is done in reverse order. This ensures the auto-start
      // bit in register 5 is reset. If this is not done the blter may
      // engage after setting the width field in register 3.

      // Set the Blt Function register.
      // - ensure auto-start is reset.
      asmIOW32( 0x3D0, 0x00000005 );

      // Set the Blt Description register.
      // - solid fill ROP is 2.
      asmIOW32( 0x3D0, 0x00000204 );

      // Set the Blt Width and Foreground Color register.
      dwTmp = ((DWORD)(200  - 1) << 16) | BLUE << 8 | 3;
      asmIOW32( 0x3D0, dwTmp );

      // Set the Blt Height and Background Color register.
      // - solid fill does not use the background color.
      dwTmp = ((DWORD)(200 - 1) << 16) | 2;
      asmIOW32( 0x3D0, dwTmp );

      // Set the Destination Address register. (Dst = 100,110)
      dwTmp = ((DWORD)(110 * 640l) + 100) << 8 | 1;
      asmIOW32( 0x3D0, dwTmp );

      // Set the Source Address register.
      // solid fill does not use source address so we could omit this step.
      asmIOW32( 0x3D0, 0x00000000 );
```

```
      // Engage the BLT engine.
      asmIOW8( 0x3D0, 0x80 );

      CheckBltStatus();
}

////////////////////////////////////////////////////////////////////////
// DoPlanarMoveBlt
//    This routine:
//    - calls DoPlanarWriteBlt() to draw a smiley to screen co-ordinates 100,110
//    - uses the BLT engine to copy the image to location 300,330
//      As the result will be visible on screen I am not engaging Src-Linear
//      Select or Dst-Linear Select.
//      There is no overlap so the BLT is run in an increment direction.
////////////////////////////////////////////////////////////////////////

void DoPlanarMoveBlt( void )
{
      DWORD dwTmp;
      DWORD dwSrc, dwDst, dwWidth, dwHeight;

      // First check that we can safely write to video memory.
      // NOTE: this only needs to be done in a multi-tasking
      //       environment where there may already be a blt
      //       operation in progress.
      CheckBltStatus();

      // Check that the blt doesn't have to be special cased.
      dwSrc    = (110 * BytesScanLine) + 100;
      dwDst    = (210 * BytesScanLine) + 200;
      dwWidth  = 10;
      dwHeight = 10;

      if (FALSE == IsBltOkay( MOVE, dwSrc, dwDst, dwWidth, dwHeight))
      {
              // If the test failed then we have to handle the blt using
              // standard video memory access techniques.
              // It's a small blt so it won't take much time.
      }

      // We need an image to BLT.
      // Call DoPlanarWriteBlt() to draw a smiley at 100,100.
      DoPlanarWriteBlt();

      // Set-up the blter registers.
      // The setup is done in reverse order. This ensures the auto-start
      // bit in register 5 is reset. If not done this way the blter may
      // engage after setting the width field in register 3.
```

```
      // Set the Blt Function register.
      // -ensure auto-start is reset.
      asmIOW32( 0x3D0, 0x00000005 );

      // Set the Blt Description register.
      // - the ROP for move BLT is 3.
      // None of the move modifiers need be set.
      asmIOW32( 0x3D0, 0x00000304 );

      // Set the Blt Width and Foreground Color register.
      // - move BLT does not use foreground color.
      dwTmp = ((DWORD)(10 - 1) << 16) | 3;
      asmIOW32( 0x3D0, dwTmp );

      // Set the Blt Height and Background Color register.
      // - move BLT does not use background color.
      dwTmp = ((DWORD)(10 - 1) << 16) | 2;
      asmIOW32( 0x3D0, dwTmp );

      // Set the Destination Address register. (Dst = 300,330)
      dwTmp = ((330 * 640l) + 300) << 8 | 1;
      asmIOW32( 0x3D0, dwTmp );

      // Set the Source Address register. (110,100 from DoPlanarWriteBlt())
      dwTmp = ((110 * 640l) + 100) << 8 | 0;
      asmIOW32( 0x3D0, dwTmp );

      // Engage the BLT engine.
      asmIOW8( 0x3D0, 0x80 );

      // Wait for the BLT to complete.
      CheckBltStatus();
}


////////////////////////////////////////////////////////////////////////
// DoPlanarPatternFillBlt
//    This routine:
//    - Uses a Write BLT sequence to place a pattern into off-screen memory.
//    - Uses pattern BLT to replicate the pattern over a 200 x 200 area
//      starting at 100,110.
////////////////////////////////////////////////////////////////////////

// Data pattern used for this BLT.
// The periods are colored blue and the X's are white.
// As this is intel architecture the bytes had to be DWORD rearranged.

DWORD PlanarPatternData[] =
{
```

```
      0x11111F11,             // ...X....
      0x11F11111,             // ....X...
      0x1F1F1111,             // .....X.X
      0xF11111F1,             // X.....X.
      0x111F111F,             // .X...X..
      0x11F1F111,             // ..X.X...
      0x11111F11,             // ...X....
      0x11111F11,             // ..X.....
};

void DoPlanarPatternFillBlt( void )
{
      DWORD dwTmp;
      int     nDwords;

      // First check that we can safely write to video memory.
      // NOTE: this only needs to be done in a multi-tasking
      //       environment where there may already be a blt
      //       operation in progress.
      CheckBltStatus();

      // The first thing that must be done is a pattern has to be loaded into
      // video memory. A Write BLT will be used to do put a pattern at 0,480
      // (the first byte of off-screen video memory). The procedure is the same
      // as DoPlanarWriteBlt() uses.

      asmIOW32( 0x3D0, 0x00000105 );// enable translate.
      asmIOW32( 0x3D0, 0x00000004 );//
      asmIOW32( 0x3D0, 0x003F0003 );// 64 pixels.
      asmIOW32( 0x3D0, 0x00000002 );// 1 line.
      asmIOW32( 0x3D0, ((480 * 640l)) << 8 | 1 );
      asmIOW32( 0x3D0, 0x00000000 );
      asmIOW8( 0x3D0, 0x80 );
      nDwords =  8 * ((8 + 7) / 8);
      asmMoveDWs( _MK_FP(0xA000, 0x0000), (LPDWORD)&PlanarPatternData, nDwords );

      // The pattern is no in off-screen video memory.

      // Set-up the blter registers.
      // The setup is done in reverse order. This ensures the auto-start
      // bit in register 5 is reset. If not done this way the blter may
      // engage after setting the width field in register 3.

      // Set the Blt Function register.
      // - ensure auto-start is reset.
      asmIOW32( 0x3D0, 0x00000005 );
```

```
      // Set the Blt Description register.
      // - the ROP for a pattern BLT is 4.
      asmIOW32( 0x3D0, 0x00000404 );

      // Set the Blt Width and Foreground Color register.
      // - foreground color is not used.
      dwTmp = ((DWORD)(200 - 1) << 16) | 3;
      asmIOW32( 0x3D0, dwTmp );

      // Set the Blt Height and Background Color register.
      // - background color is not used.
      dwTmp = ((DWORD)(200 - 1) << 16) | 2;
      asmIOW32( 0x3D0, dwTmp );

      // Set the Destination Address register. (pattern to 100,110)
      dwTmp = ((110 * 640l) + 100) << 8 | 1;
      asmIOW32( 0x3D0, dwTmp );

      // Set the Source Address register.
      // - set to the starting pixel of our pattern.
      dwTmp = (480 * 640l) << 8 | 0;
      asmIOW32( 0x3D0, dwTmp );

      // Engage the BLT engine.
      asmIOW8( 0x3D0, 0x80 );

      // Wait for the BLT to complete.
      CheckBltStatus();
}


////////////////////////////////////////////////////////////////////////////
// DoPlanarReadBlt
//    This routine:
//    - Calls DoPlanarWriteBlt() to draw a 10x10 smiley at 100,110.
//    - Uses a Read BLT to read the smiley data back into a memory array.
//    - Compares the read data to expected values and reports any differences.
////////////////////////////////////////////////////////////////////////////

void DoPlanarReadBlt( void )
{
      DWORD dwTmp;
      int nDwords, nRow;
      LPDWORD lpdwRead = dwReadBltData;
      LPDWORD lpdwWrt = PlanarWrtData;
      DWORD dwSrc, dwDst, dwWidth, dwHeight;

      // First check that we can safely access video memory.
      // NOTE: this only needs to be done in a multi-tasking
```

```
//       environment where there may already be a blt
//       operation in progress.
CheckBltStatus();

// Check that the blt doesn't have to be special cased.
dwSrc    = (110 * BytesScanLine) + 100;
dwDst    = 0;
dwWidth  = 10;
dwHeight = 10;

if (FALSE == IsBltOkay( READ, dwSrc, dwDst, dwWidth, dwHeight))
{
        // If the test failed then we have to handle the blt using
        // standard video memory access techniques.
        // It's a small blt so it won't take much time.
}

// We need something to read. Call the DoPlanarWriteBlt() to
// give us the smiley face at 100,110.
DoPlanarWriteBlt();

// The smiley is in video memory. We are now ready to read it.

// Set-up the blter registers.
// The setup is done in reverse order. This ensures the auto-start
// bit in register 5 is reset. If not done this way the blter may
// engage after setting the width field in register 3.

// Set the Blt Function register.
// - ensure auto-start is reset. Enable Translate
//  (so the data we read back can be compared to the original)
asmIOW32( 0x3D0, 0x00000105 );

// Set the Blt Description register.
// - the ROP for a READ BLT is 5.
asmIOW32( 0x3D0, 0x00000504 );

// Set the Blt Width and Foreground Color register.
// - the smiley is 10 pixels wide.
// - foreground color is not used.
dwTmp = ((DWORD)(10 - 1) << 16) | 3;
asmIOW32( 0x3D0, dwTmp );

// Set the Blt Height and Background Color register.
// - the smiley is 10 pixels high.
// - background color is not used.
dwTmp = ((DWORD)(10 - 1) << 16) | 2;
asmIOW32( 0x3D0, dwTmp );
```

```
	// Set the Destination Address register.
	// - read into position 0 in the buffer.
	//   (This doesn't demonstrate phase alignment but it will
	//    make it easier to compare the buffer to the expected values)
	asmIOW32( 0x3D0, 1 );

	// Set the Source Address register. (read from 100,110)
	dwTmp = ((110 * 640l) + 100) << 8 | 0;
	asmIOW32( 0x3D0, dwTmp );

	// Engage the BLT engine.
	asmIOW8( 0x3D0, 0x80 );

	// Calculate the number of DWORDs required.
	// DWORDs = Rows * ((DstAlign + Width + (pels/DWORD - 1)) / (pels/DWORD))
	nDwords =  10 * ((0 + 10 + 7) / 3);

	asmMoveDWs( (LPDWORD)&dwReadBltData, _MK_FP( 0xA000, 0x0000 ), nDwords );

	// Do an assurance check on the BLT status.
	CheckBltStatus();

	// The first ten pixels of data in dwReadBltData and PlanarWrtData should
	// match. The unused portion of each last DWORD read likely will not.
	for (nRow = 0; nRow < 10; nRow++ )
	{
		if (*lpdwWrt != *lpdwRead)
			printf( "Error in Read BLT compare!\n" );
		lpdwWrt++;
		lpdwRead++;

		if ((*lpdwWrt & 0x000000FF) != (*lpdwRead & 0x000000FF))
			printf( "Error in Read BLT compare!\n" );
		lpdwWrt++;
		lpdwRead++;
	}
}

////////////////////////////////////////////////////////////////////////
// DoPlanarInvertBlt
//    This routine:
//    - Calls DoPlanarPatternBlt() to display a pattern on screen.
//    - Inverts the pattern area.
////////////////////////////////////////////////////////////////////////

void DoPlanarInvertBlt( void )
{
	DWORD dwTmp;
```

```
        // First check that we can safely write to video memory.
        // NOTE: this only needs to be done in a multi-tasking
        //       environment where there may already be a blt
        //       operation in progress.
        CheckBltStatus();

        // We need something to invert. Call the pattern BLT routine to
        // give us a 200 x 200 pattern at 100,110.
        DoPlanarPatternFillBlt();

        // The pattern is in video memory. We are now ready to invert it.

        // Set-up the blter registers.
        // The setup is done in reverse order. This ensures the auto-start
        // bit in register 5 is reset. If not done this way the blter may
        // engage after setting the width field in register 3.

        // Set the Blt Function register.
        // - ensure auto-start is reset.
        asmIOW32( 0x3D0, 0x00000005 );

        // Set the Blt Description register.
        // - the ROP for an INVERT BLT is 6.
        asmIOW32( 0x3D0, 0x00000604 );

        // Set the Blt Width and Foreground Color register.
        // - foreground color is not used.
        dwTmp = ((DWORD)(200 - 1) << 16) | 3;
        asmIOW32( 0x3D0, dwTmp );

        // Set the Blt Height and Background Color register.
        // - background color is not used.
        dwTmp = ((DWORD)(200 - 1) << 16) | 2;
        asmIOW32( 0x3D0, dwTmp );

        // Set the Destination Address register. (Dst = 100,110)
        dwTmp = ((110 * 640l) + 100) << 8 | 1;
        asmIOW32( 0x3D0, dwTmp );

        // Set the Source Address register.
        // - invert BLT does not use this so it doesn't really need to be done.
        asmIOW32( 0x3D0, 0 );

        // Engage the BLT engine.
        asmIOW8( 0x3D0, 0x80 );

        // Wait for the BLT to complete.
        CheckBltStatus();
}
```

```
////////////////////////////////////////////////////////////////////////////
//  C support routines.
////////////////////////////////////////////////////////////////////////////

void DisplayCopyright( void )
{
      printf( "BLT-SAMP.EXE - Sample BitBlt Programming Code - version 1.0\n");
      printf( "Copyright (c) Seiko Epson Corp. 1996. All rights reserved.\n\n");
      printf( "This program demonstrates the SPC8110 blter capabilities\n\n" );
      printf( "Press <ENTER> to continue" );
}


void DoVideoMode( short nMode, char *pStr )
{
      _setvideomode( nMode);
      printf( pStr );
      printf( "Press <ENTER> to continue or <CTRL-C> to break\n" );
}

////////////////////////////////////////////////////////////////////////////
// PROGRAM ENTRY POINT
////////////////////////////////////////////////////////////////////////////

void main( int argc, char * argv)
{
      _setvideomode( 3 );
      DisplayCopyright();
      getchar();

      // Set the global variables.

      BytesScanLine = 640;     // The number of bytes on earch row of the display.
      BltType = PACKED;        // PACKED or PLANER
      SrcType = SQUARE;        // LINEAR or SQUARE
      DstType = SQUARE;        // LINEAR or SQUARE

      DoVideoMode( 0x101, "Packed Pixel Write BLT\n" );
      DoPackedWriteBlt();
      getchar();

      DoVideoMode( 0x101, "Packed Pixel Color Expand Blt\n" );
      DoPackedColorExpandBlt();
      getchar();

      DoVideoMode( 0x101, "Packed Pixel Solid Fill Blt\n" );
      DoPackedSolidFillBlt();
      getchar();
```

```
DoVideoMode( 0x101, "Packed Pixel Move Blt\n" );
DoPackedMoveBlt();
getchar();

DoVideoMode( 0x101, "Packed Pixel Pattern Fill Blt\n" );
DoPackedPatternFillBlt();
getchar();

DoVideoMode( 0x101, "Packed Pixel Read Blt\n" );
DoPackedReadBlt();
getchar();

DoVideoMode( 0x101, "Packed Pixel Invert Blt\n" );
DoPackedInvertBlt();
getchar();

// Set the global variables.

BytesScanLine = 640;     // The number of bytes on earch row of the display.
BltType = PACKED;        // PACKED or PLANER
SrcType = SQUARE;        // LINEAR or SQUARE
DstType = SQUARE;        // LINEAR or SQUARE

DoVideoMode( 0x12, "Planar Write BLT\n" );
DoPlanarWriteBlt();
getchar();

DoVideoMode( 0x12, "Planar Color Expand Blt\n" );
DoPlanarColorExpandBlt();
getchar();

DoVideoMode( 0x12, "Planar Solid Fill Blt\n" );
DoPlanarSolidFillBlt();
getchar();

DoVideoMode( 0x12, "Planar Move Blt\n" );
DoPlanarMoveBlt();
getchar();

DoVideoMode( 0x12, "Planar Pattern Fill Blt\n" );
DoPlanarPatternFillBlt();
getchar();

DoVideoMode( 0x12, "Planar Read Blt\n" );
DoPlanarReadBlt();
getchar();
```

```
      DoVideoMode( 0x12, "Planar Invert Blt\n" );
      DoPlanarInvertBlt();
      getchar();

      _setvideomode( 3 );

      // Quiet the compiler warning.
      argc;
      argv;
}
```

Assembler Support Routines for BLT-SAMP.C

```
;=============================================================================
;  BLTASM.ASM - Assembler support routines for BLT-SAMP.C
;
;             This file provides support for machine operations not
;             provided by the C compiler libraries.
;             It also provides a clearer view of the process used to
;             perform a color expand BLT.
;-----------------------------------------------------------------------------
; This file has been compiled and tested with MS Macro Assembler 6.11
;;============================================================================

      .MODEL  LARGE, C
      .386

      .DATA

      .CODE

;=============================================================================
;  Port I/O routines.
;-----------------------------------------------------------------------------
;
;  Purpose:  These routines provide byte, word and dword port I/O.
;            No other description necessary, see bltasm.h for syntax.
;
;  Returns:  AL, AX or EAX will equal the value written or read.
;
;=============================================================================

asmIOW8       PROC  C  wPort:WORD, wVal:WORD

      mov     ax, wVal
      mov     dx, wPort
      out     dx, al
      ret
```

```
asmIOW8       ENDP

;-------------------------------------------------------------------------

asmIOW16 PROC  C  wPort:WORD, wVal:WORD

      mov     ax, wVal         ; Load AX with the 16-bit value.
      mov     dx, wPort        ; Load DX with the port to write.
      out     dx, ax           ; Do the write.
      ret                      ; Done.

asmIOW16 ENDP

;-------------------------------------------------------------------------

asmIOW32 PROC  C wPort:WORD, dwVal:DWORD

      mov     eax, dwVal       ; Load EAX with the 32-bit value.
      mov     dx, wPort        ; Load DX with the port to write.
      out     dx, eax          ; Do the write.
      ret                      ; Done.

asmIOW32ENDP

;-------------------------------------------------------------------------

asmIOR8 PROC  C  wPort:WORD

      mov     dx, wPort
      in      al, dx
      xor     ah, ah
      xor     dx, dx
      ret

asmIOR8       ENDP
;=========================================================================
;  asmMoveDWs
;-------------------------------------------------------------------------
;
;   Purpose:  Moves nCount DWORDS from the address pointed at by lpSrc
;           to the address pointed at by lpDst.
;
;    Usage:  void MoveDWs( LPDWORD lpDst, LPDWORD lpSrc, int nCount );
;
;   Returns:  [nothing]
;
;  Modifies:  nCount DWORDs at lpDst
;
```

```
;===========================================================================
asmMoveDWs PROC  C USES DS ES SI DI,
              lpDst:DWORD, lpSrc:DWORD, nCount:WORD

      mov     cx, nCount       ; Load the count.
      lds     si, lpSrc        ; Load the source address.
      les     di, lpDst        ; Load the destination address.

      nop                      ; Needed or we hang!
      nop                      ; And I don't know why!

      rep     movsd            ; Move the DWORDS.
      ret                      ; Done.
asmMoveDWs ENDP

;===========================================================================
;  asmPackedColorExpand
;---------------------------------------------------------------------------
;
;   Purpose: This routine demonstrates the technique to perform an
;            auto-start auto-update Color Expand BLT in packed pixel mode.
;
;    Usage:
;
;   Returns:
;
;  Modifies:
;
;===========================================================================
asmPackedColorExpand PROC C USES DS ES SI

      mov     ax, 1130h        ; Point DS:SI to the 16 point
      mov     bh, 6            ; font table.
      push    bp               ;
      int     10h              ;
      mov     si, bp           ;
      push    es               ;
      pop     ds               ;
      pop     bp               ;

      mov     ax, 'A'          ; Point SI to the letter 'A'
      shl     ax, 4            ;
      add     si, ax           ;

      mov     bx, 0A000h       ; Point ES to video memory.
      mov     es, bx           ;

      mov     bx, 26           ; Draw 26 letters.
```

```
NextChar:
      mov     cx, 16          ; 16 Rows to do.

NextScan:
      lodsb                   ; Get character scan data.
      mov     ah, al          ;
      shr     ah, 4           ;
      nop
      mov     es:[0], ah      ; NOTE: the blter counts all writes
      mov     es:[0], al      ; as 32-bits. But the upper bits are
      loop    NextScan        ; discarded sooo...

      mov     dx, 3D0h        ;
WaitBlt:
      in      al, dx          ; Wait for the status to clear
      test    al, 80h         ; before starting the next character.
      loopnz  WaitBlt         ;

      dec     bx              ; Are we done the string?
      jz      Quit            ;

      mov     dx, 3D2h        ; Setting the width starts the
      mov     ax, 07h         ; next BLT.
      out     dx, ax          ;

      jmp     NextChar        ; Go do the next row.

Quit:
      ret
asmPackedColorExpand ENDP


;============================================================================
;  asmPlanarColorExpand
;----------------------------------------------------------------------------
;
;   Purpose: This routine demonstrates the technique to perform an
;            auto-start auto-update Color Expand BLT in planar display modes.
;
;    Useage:
;
;   Returns:
;
;  Modifies:
;
;============================================================================
asmPlanarColorExpand PROC C USES DS ES SI
```

```
      mov     ax, 1130h       ; Point DS:SI to the 16 point font.
      mov     bh, 6           ;
      push    bp              ;
      int     10h             ; Use BIOS services to get the pointer.
      mov     si, bp          ;
      push    es              ;
      pop     ds              ;
      pop     bp              ;

      mov     ax, 'A'         ; Start with the letter 'A'
      shl     ax, 4           ;
      add     si, ax          ; Offset SI to the letter 'A'
      mov     ax, 0A000h      ; Point ES to video memory.
      mov     es, ax          ;
      mov     bx, 26          ; Do 26 letters.

NextChar:
      mov     cx, 16          ; Each character cell has 16 rows.

NextScan:
      lodsb                   ; Get character scan data.
      mov     es:[0], al      ; Send it to the blter.
      loop    NextScan        ;

      mov     dx, 3D0h        ; Wait for the blter to signal the
WaitBlit:                     ; current BLT as done before starting
      in      al, dx          ; the next.
      test    al, 80h         ;
      loopnz  WaitBlit;

      dec     bx              ; Are we done yet?
      jz      Quit            ;
      mov     dx, 3D2h        ; Set the next width, which auto-
      mov     ax, 07h         ; starts the BLT engine. Auto-update
      out     dx, ax          ; has already set the destination.

      jmp     NextChar        ;
Quit:
      ret                     ;
asmPlanarColorExpand ENDP
```

```
;=============================================================================
;  asmSetDAC
;-----------------------------------------------------------------------------
;
;   Purpose: Sets RAMDAC entry 0F0h to BLACK and entry 0FBh to CYAN
;
;    Usage: void asmSetDAC( void )
;
;   Returns: [nothing]
;
;  Modifies: RAMDAC
;
;=============================================================================
asmSetDAC PROC  C

      mov     dx, 3C8h         ; Set DAC entry F0h to BLACK.
      mov     al, 0F0h         ;
      out     dx, al           ;
      inc     dx               ;
      mov     al, 0            ;
      out     dx, al           ;
      out     dx, al           ;
      out     dx, al

      mov     dx, 3C8h         ; Set DAC entry FBh to CYAN.
      mov     al, 0FBh         ;
      out     dx, al           ;
      inc     dx               ;
      mov     al, 0            ;
      out     dx, al           ;
      mov     al, 2Ah          ;
      out     dx, al           ;
      mov     al, 2Ah          ;
      out     dx, al           ;

      ret
asmSetDAC ENDP

      END
```

PROGRAMMING THE SPRITE / HARDWARE CURSOR

The SPC8110 video controller implements a hardware sprite/cursor capable of displaying a 64 x 64 x 4 color pixel image regardless of the video display mode. Programming is performed through two index/data register pairs located in the Auxiliary and CRTC registers. The sprite image is stored in off-screen video memory. The image may be stretched horizontally and vertically.

The following information and code samples assume the reader has some understanding of the index/ data architecture of a VGA controller. The code samples presented are C-like and are intended to show concepts, and as such have not been compiled or run. Throughout this document the term sprite and cursor may be freely interchanged. The differences between a sprite and a cursor are: i) size: a typical cursor is 32 x 32 pixels while the sprite hardware is capable of a 64 x 64 pixel image; ii) color: sprite palette schemes support four display colors while the cursor mode supports two colors with the remaining two colors representing screen and not screen (transparent and invert).

The first sprite register pair is located in the CRTC (port 3D4h/3D5h) at indices 30h through 3Fh. These indices are beyond the IBM defined registers for a VGA and control functions such as sprite location in off-screen memory, display position and clipping.

The second index/data pair is located in the SPC8110 auxiliary register (port 3DEh/3DFh) at indices 1 and 5. These registers control enabling the sprite, enabling the sprite palette registers and supplemental support to the memory mechanisms for accessing sprite video memory.

Programming the SPC8110 hardware sprite/cursor involves:

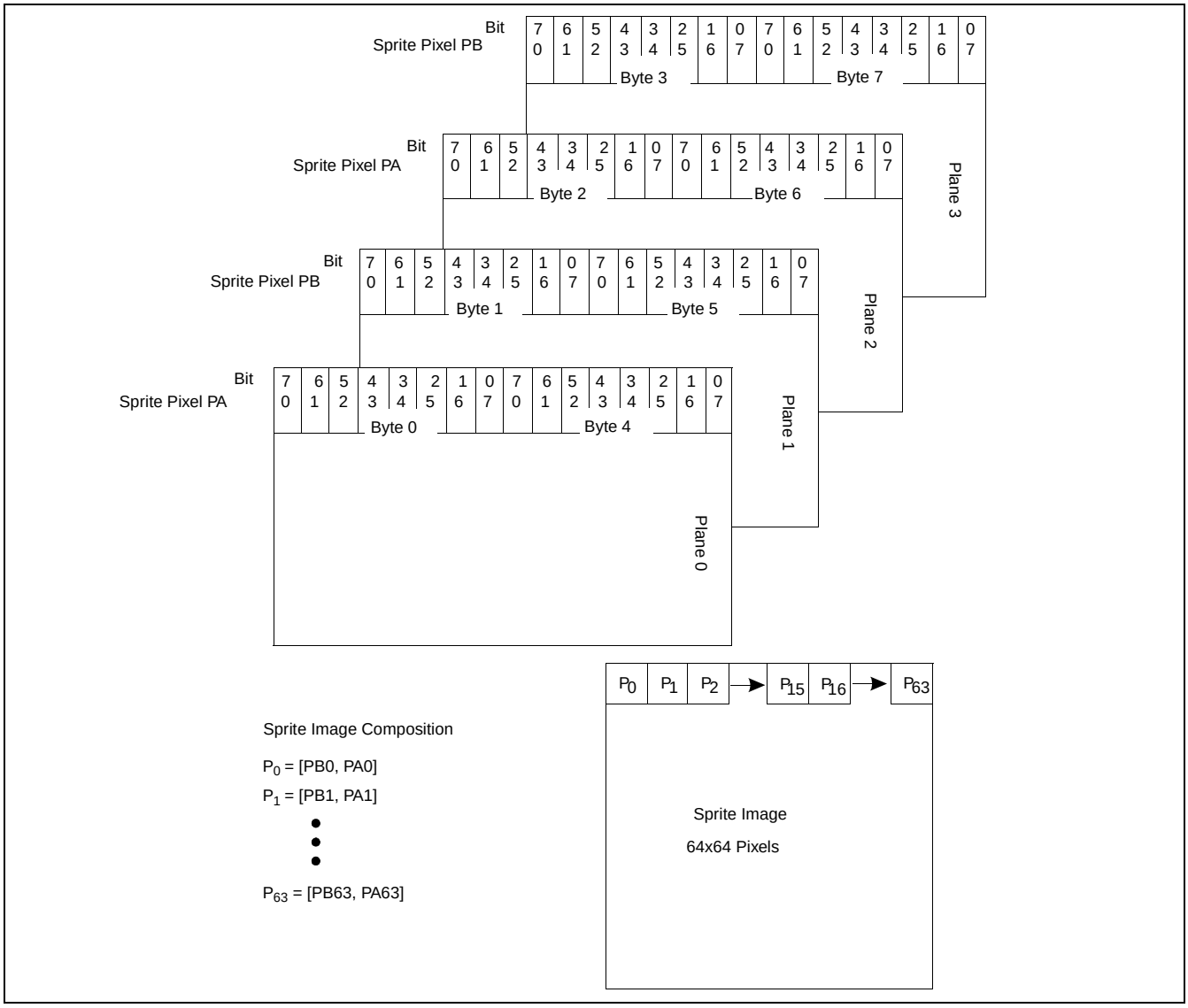
1. Loading an image into a display memory location.
2. Loading the palette with the appropriate colors.
3. Enabling the sprite.
4. Setting the sprite position registers.

There are two means of placing an image into sprite memory. One involves accessing memory in a planar mode, such as standard IBM mode 12h. Each plane is accessed and the appropriate data pattern is written. The drawback to this method occurs when a non-planar mode is active and the sprite is to be changed. In this situation a quasi-planar mode would have to be setup in the VGA, the sprite data written and the VGA registers restored to their original state. In addition to being more complex this method, depending on the current video mode, can induce display flicker.

The method implemented by the example code uses a technique of bypassing the graphics controller. Using this method requires setting only one bit in the auxiliary register.

A packed pixel structure for the sprite image is not supported, therefore prior to loading sprite memory the image data must be broken down into two arrays of pixels. This format is typical of the XOR and AND arrays provided in such popular GUI environments such as Windows and OS/2. The following diagram depicts the VGA memory in a planer format and the sprite as displayed on the screen. The two mask structures represent the strings of pixel information where each bit of Bytes

0 through 3 represent one pixel on the display (bits 0 through 31). The high order bits of the image (cursor XOR mask) are placed in the even planes (planes 0 and 2) and the low order bits (AND mask) are place in the odd planes (planes 1 and 3).



One sprite image requires 1024 bytes of display memory. This memory may be accessed as either 256 consecutive planar memory locations (x 4 planes) or as 1024 graphics controller bypass locations.

Image Loading and Hardware Enabling

The following code demonstrates how to load a sprite image (64 x 64) into slot 511 (512th 1K sprite location). This sprite location is chosen completely at random and will not be an appropriate location for the sprite in certain video modes. For example VESA mode 105h (1024 x 768 by 256 colors) requires 768K bytes of video memory and would conflict with the memory required for this sprite example. In general it is best to keep the sprite as high in video memory as possible, however locating the sprite in the last page (64K) of memory has the potential for memory conflicts as well. The Seiko Epson BIOS uses memory in the last 64K for an LCD panel half frame buffer. The safest, although not the only, location for the cursor should be at the end of display memory minus one 64K byte page.

```
char far *lpVidMem, *pXorArray, *pAndArray;
char EvenArray[512], OddArray[512];
int nCnt, nAux;

#define  AUX 0x3DE
#define CRTC 0x3D4

/* Load the Image into Display Memory */
/* ---------------------------------- */
outp (AUX, 5);                  // Bypass GRFX controller and enable paging.
outp (AUX + 1, inp (AUX + 1) | 0x11);

outp (0x3CD, 0x77);             // Set the page. Each page can access 64 sprites
                                // Therefore (slot) 511 / 64 = 7.xxx or page 7.

// In this example code I utilize an image which is already separated
// into two arrays representing the high and low order bits of each pixel.
// This is consistent with the cursor format supplied by Microsoft Windows.
// We have 1024 bytes to write (512 from each array) and we want to place
// the XOR mask on the even planes and the AND mask on the odd planes.

pXorArray = EvenArray;          // Set the pointers.
pAndArray = OddArray;           //
lpVidMem = _MK_FP(0xA000, (511 % 64) * 1024);  // 0xFC00

for (nCnt = 0; nCnt < 512; nCnt++) {  // 512 bytes from each array.
    *lpVidMem++ = *pXorArray;  // XOR mask.
    *lpVidMem++ = *pAndArray;  // AND mask.
    pXorArray++;               // Bump the pointers.
    pAndArray++;               //
       }

outp (AUX, 5);                  // Disable the page register, turn off graphics
outp (AUX + 1, inp (AUX + 1) | 0x20); // bypass and enable the sprite.
outp (AUX + 1, inp (AUX + 1) & 0xEE);

/* Set the CRTC Sprite Registers */
/* ----------------------------- */
// Inform the chip where in memory to find the image. (slot 511 = 0x1FF)
outp (CRTC, 0x36);              // Sprite Address b9:8
outp (CRTC +1, 1);              //
outp (CRTC, 0x37);              // Sprite Address b7:0
outp (CRTC +1, 0xFF);           //

// For four color sprites CRTC[38] b1:0 must be 0.
// For cursors CRTC[38] b1:0 is set to 1x.
outp (CRTC, 0x38);              // No vertical or horizontal doubling,
outp (CRTC +1, 2);              // use the cursor palette scheme.
```

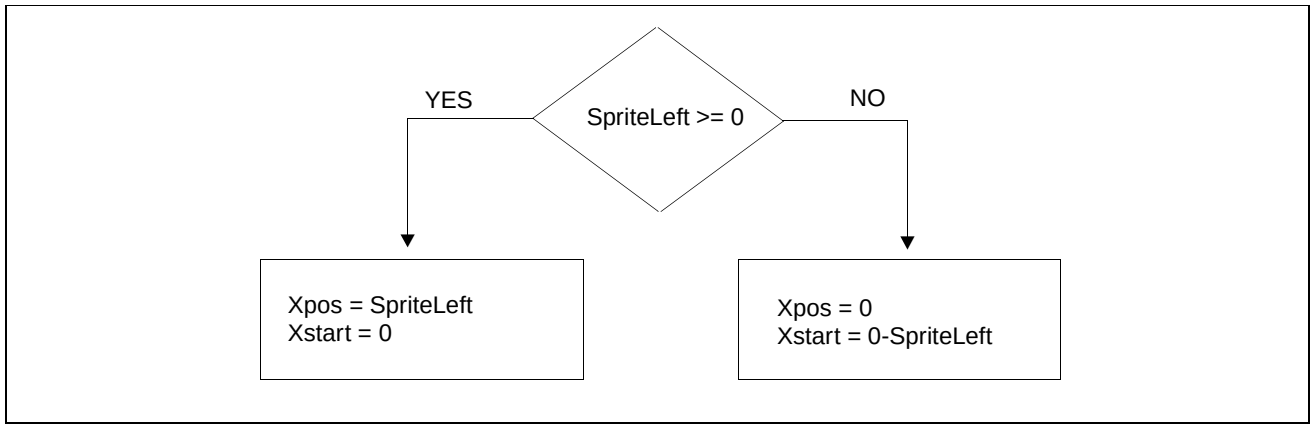
```
outpw (CRTC, 0x0030);          // Set the X position to 0, left hand edge.
outpw (CRTC, 0x0031);          //
outpw (CRTC, 0x0032);          // Set the Y position to 0, top edge.
outpw (CRTC, 0x0033);          //
outpw (CRTC, 0x0034);          // No clipping in the X direction.
outpw (CRTC, 0x0035);          // No clipping in the Y direction.

/* Set the colors. */
/*------------------*/
outp (AUX, 0);                 // Set the AUX port to index 0.
nAux = inp (AUX + 1);          // Read the current setting.
outp (AUX +1, (nAux | 0x80))   // Mask on the Sprite Palette Enable bit.

outp (0x3C8, 0);               // Set the RAMDAC write index to 0.
outp (0x3C9, RED1);            // Set the cursor foreground/sprite color 1.
outp (0x3C9, BLU1);            //
outp (0x3C9, GRN1);            //

outp (0x3C9, RED2);            // Set the cursor background/sprite color 2.
outp (0x3C9, BLU2);            //
outp (0x3C9, GRN2);            //

outp (0x3C9, RED3);            // Set sprite color 3.
outp (0x3C9, BLU3);            // Note: colors 3 and 4 are not used by the
outp (0x3C9, GRN3);            //       sprite while in cursor mode.


outp (0x3C9, RED4);            // Set sprite color 4.
outp (0x3C9, BLU4);            //
outp (0x3C9, GRN4);            //

outp (AUX + 1, nAux);          // Restore AUX[0] register.

/* Sprite initialization complete. */
/*---------------------------------*/
// The sprite/cursor image may be removed from the display by resetting AUX[5] b5.
// While I loaded all 1024 sprite bytes with data for a complete 64 x 64 image
// a cursor under Windows is only 32 x 32 pixels requiring only 512 bytes of data.
// I also programmed all four palette indices whereas a cursor only uses
// indices 0 and 1.
```

Setting the Sprite Position

The displayed position of the sprite and any required clipping is done with CRTC registers 30H through 35H. Clipping of the sprite image on the right and bottom of the display is done automatically by the hardware. If the sprite image needs to be clipped on the top or left of the display the clipped amount needs to be calculated and set into the appropriate registers.

All position calculations are based on absolute pixel positions. This holds even for text modes which means the sprite image location is not limited to character cell boundaries but may be placed on any pixel on the display. Sprite positioning information is always relative to the upper left corner of the sprite image and to the upper left corner of the display. The following example demonstrates how to program a cursor image located in the lower left quadrant of the sprite as this covers all positioning considerations.

In general the algorithm for cursor position and clipping is:



Note
For Y positioning, substitute top for left and Y for X.

The following example is typical of programming a cursor position with coordinates passed by Microsoft Windows.

```
/* Calculate the X position. */
/*---------------------------*/

nAdjX = nCursPos - HotSpot     // nAdjX = adjusted X position.
if (nAdjX > 0) {               // If > 0 then no clipping required.
      Xpos = nAdjX;            // Set X position to AdjX and
      Xstart = 0;              // X start to 0.
      }
else {                         // Clipping is required.
      Xpos = 0;                // Set the X position to 0 and
      Xstart = 0 - nAdjX       // X start to clip at Xadj.
      }

/* Program the X position registers. */
// First set clipping. This value informs the chip how
// many pixels to count into the image before starting display.
outp (CRTC, 34H);
outp (CRTC + 1, nXstart);
```

```
// Set the X position. As the X position may be > 255 two data register
// writes are required. IMPORTANT - The position in these registers does not
// latch until the MSBs (register 30h) are written.

outp (CRTC, 31H);
outp (CRTC + 1, LOBYTE(Xpos));
outp (CRTC, 30H);
outp (CRTC + 1, HIBYTE(Xpos));

/* Calculate the Y position. */
/*---------------------------*/
nAdjY = nCursorPos - nHotSpot  // nAdjY = adjusted Y value.
if (nCursorY > 0) {            // If > 0 then no Y clipping required.
      nYpos = nAdjY;           //
      nYstart = 0;             //
  }
else {                         // Clipping is required.
      nYpos = 0;               //
      nYstart = 0 - nAdjY;     //
      }

/* Program the Y registers. */
// First set the clipping amount. This value informs the chip
// how many pixels to count down into the image before starting display.
outp (CRTC, 35H);
outp (CRTC +1, nYstart);

// Next set the Y position. As the Y position may be > 255 two data register
// writes are required. IMPORTANT - The position registers do not latch until
// the MSB (register 33h) is written.
outp (CRTC, 34H);
outp (CRTC + 1, LOBYTE(Ypos));
outp (CRTC, 33H);
outp (CRTC + 1, HIBYTE(Ypos));

/* End of programming sprite/cursor position. */
```

PROGRAMMING THE PLL AND FIFO

The purpose of this section is to demonstrate how to program the PLL and FIFO by supplying sample code. The order of programming is important; the FIFO must always be programmed after the PLL clocks are programmed.

How To Program The PLL Clock

1. Select a frequency for a PLL clock.
   Try to use only VGA standard clocks for text/graphic modes:
   - PLL1 Set 0 = 25 MHz
   - PLL1 Set 1 = 28 MHz
   - PLL2 = 39.4 MHz
2. To program the PLL, first determine the PLL numerator and denominator. Calculate the numerator and denominator for all possible combinations (a numerator and denominator each have 4 bits, for a total of 256 possible frequencies). Choose the closest matching calculated frequency to your desired frequency and use its appropriate numerator and denominator.

$$frequency = \frac{\text{REFERENCE} \cdot \times numerator}{denominator}$$

   where *frequency* is the PLL clock frequency in MHz
   REFERENCE is the reference frequency in MHz (typically 14.31818)
   numerator is the PLL numerator (a value between 1 and 15)
   denominator is the PLL denominator (a value between 1 and 15)

3. Determine the gain bits:
   a. If Freq < 31 MHz, Gain Select Bit 0 = 1. Otherwise Gain Select Bit 0 = 0.
   b. If Freq < 51 MHz, Gain Select Bit 1 = 1. Otherwise Gain Select Bit 1 = 0.
4. Program the PLL denominator in the appropriate PLL register.
5. Combine the Gain Select Bits with the numerator, and program the PLL numerator in the appropriate PLL register. Gain Select Bit 0 is bit 6 in the numerator, and Gain Select Bit 1 is bit 7 in the numerator. Note that the numerator must always be programmed AFTER the denominator.
6. Program the FIFO (see next section).

How To Program The FIFO

Recommended Procedure For Non-VGA Compatible Systems

For Systems Using A CRT

- Set Miscellaneous Output Register 3C2h bit 3 = 1
  This selects the PLL1 Set 2 pixel clock, which requires only the Enhanced FIFO at AUX[3Ch] to be programmed.
- Program the Enhanced FIFO whenever a pixel or memory clock is changed.
- Before programming the Enhanced FIFO, determine the display mode intended for the PLL1 Set 2 as shown in Table 3. This mode indicates which set of constants to use in Table 4 for FIFO calculations.
- Use the provided equations to calculate the FIFO register values. Note that the equations require the numerator and denominator of the pixel clock. Use the pixel clock which matches the given FIFO register, as shown in Table 2.

For Systems Using An LCD

- Miscellaneous Output Register 3C2h bits 3:2 are ignored.
  PLL1 Set 0 pixel clock is used for all FIFOs.
- Program all FIFO registers whenever a pixel or memory clock is changed.
- The Enhanced FIFO will only be used for Enhanced Text and High-Resolution 256 modes.
- Determine the display mode intended for PLL1 Set 0 as shown in Table 3. This mode indicates which set of constants to use in Table 4 for FIFO calculations.
- Text fonts are always 8 dots. Consequently when programming a pixel clock for VGA Text, use the constants under VGA Text 8 Dot in Table 4.
- Use the provided equations to calculate the FIFO register values. Note that the equations require the numerator and denominator of the pixel clock. Since the PLL1 Set 0 pixel clock is used for all FIFOs, use this pixel clock for all FIFO register calculations.

Table 1 FIFO Values for C and K

			VGA Slow Text	VGA Slow Graphics	VGA Text 8 Dot	VGA Text 9 Dot	VGA Graphics	Enhanced Text	High-Res 256
Pixels per Level			16	16	8	9	8	8	4
512 KB	C		15.45	15.45	15.45	15.45	15.45	15.45	15.45
	K	CRT, Single Mono, Single Color, TFT	29.5	19.5	29.5	29.5	19.5	29.5	19.5
		Dual Mono	33.5	23.5	33.5	33.5	23.5	33.5	23.5
		Dual Color	49.5	39.5	49.5	49.5	39.5	49.5	39.5
1024 KB	C		15.45	7.45	15.45	15.45	7.45	15.45	7.45
	K	CRT, Single Mono, Single Color, TFT	27.5	15.5	27.5	27.5	15.5	27.5	15.5
		Dual Mono	29.5	17.5	29.5	29.5	17.5	29.5	17.5
		Dual Color	37.5	25.5	37.5	37.5	25.5	37.5	25.5

Recommended Procedure For VGA Compatible Systems

For Systems Using A CRT

- Program PLL1 Set 0 to 25 MHz. Never change this pixel clock.
- Program PLL1 Set 1 to 28 MHz. Never change this pixel clock.
- When calculating the FIFO values for the VGA Text FIFO, use the PLL1 Set 1 pixel clock.
- When calculating the FIFO values for the VGA Slow Text FIFO, use the PLL1 Set 1 pixel clock.
- If a pixel clock was changed, only program those FIFO registers related to that clock; Table 2 lists which FIFO registers must be programmed for a given pixel clock. In the special case of the VGA Text and Slow Dot Text FIFOs, use the PLL1 Set 1 pixel clock. In addition, special consideration must be given to the Enhanced FIFO, which must be programmed if PLL1 Set 2 has been changed and if the Clock Select bits are set to 1x (PLL1 Set 2 clock). Before programming the Enhanced FIFO, determine the graphics mode. This mode indicates which set of constants to use in Table 4 for FIFO calculations. If PLL1 Set 2 is not selected, it is not necessary to program the Enhanced FIFO's registers.
- If the memory clock was changed, program all FIFO registers.
- Use the provided equations to calculate the FIFO register values. Note that the equations require the numerator and denominator of the pixel clock. Use the pixel clock which matches the given FIFO register, as shown in Table 2.

For Systems Using An LCD

- The PLL1 Set 0 pixel clock is used for all FIFOs.
- Program all FIFO registers whenever a pixel or memory clock is changed.
- The Enhanced FIFO would only be used for Enhanced Text and High-Resolution 256 modes.
- Determine the display mode intended for PLL1 Set 0 as shown in Table 3. This mode indicates which set of constants to use in Table 4 for FIFO calculations.
- Text fonts are always 8 dots. Consequently when programming a pixel clock for VGA Text, use the constants under VGA Text 8 Dot in Table 4.
- Use the provided equations to calculate the FIFO register values. Note that the equations require the numerator and denominator of the pixel clock. Since the PLL1 Set 0 pixel clock is used for all FIFOs, use this pixel clock for all FIFO register calculations.

The following FIFO equations are simplified formulae from the SPC8110 Hardware Functional Specification:

$$LowLevel = TRUNCATE\left\{\left[\frac{C \times PLL1n \times PLL2d}{PixPerLevel \times PLL1d \times PLL2n}\right] + 4\right\}$$

$$HighLevel = ROUNDUP\left\{\left[\frac{(K \times PLL1n \times PLL2d) + ((43.3 \times 10^{-9}) \times REF \times PLL1n \times PLL2n)}{PixPerLevel \times PLL1d \times PLL2n}\right] - 1\right\}$$

where LowLevel is the FIFO low level.

HighLevel is the FIFO high level.

TRUNCATE is an operation which truncates the fractional part of a floating point number to create an integer (e.g. TRUNCATE{4.9} = 4).

ROUNDUP is an operation which truncates the fractional part of a floating point number to create an integer and increments the integer if the fractional part was non-zero (e.g. ROUNDUP{1.001} = 2).

PixPerLevel is the number of Pixels per Level (see Table 4).

C is a constant (see Table 4).

K is a constant (see Table 4).

REF is the reference frequency in Hz (typcially 14.31818×10^6).

PLL1n is the numerator of the selected pixel clock (PLL1 Set 0, PLL1 Set 1, or PLL1 Set 2).

PLL1d is the denominator of the selected pixel clock (PLL1 Set 0, PLL1 Set 1, or PLL1 Set 2).

PLL2n is the numerator of the memory clock (PLL2).

PLL2d is the denominator of the memory clock (PLL2).

Table 2 FIFO/Pixel Clocks

3C2 bits 3:2 Clock Select	Pixel Clock	FIFO Registers	
0 0	PLL1 Set 0 (25 MHz for VGA)	VGA Graphics VGA Text (200/350 lines) Slow Dot Graphics Slow Dot Text (200/350 lines)	AUX[38h] AUX[39h] AUX[3Ah] AUX[3Bh]
0 1	PLL1 Set 1 (28 MHz for VGA)	VGA Text (400 lines) Slow Dot Text (400 lines)	AUX[39h] AUX[3Bh]
1 X	PLL1 Set 2 Custom Frequency	Enhanced	AUX[3Ch]

Table 3 Determining the Display Mode

Display Mode	AUX[5] bit 7 Hi-Res 256	AUX[5] bit 6 Enhanced Text	GR[6] bit 0 Graphics Mode	SEQ[1] bit 3 Slow Dot	SEQ[1] bit 0 0=9 dot clk 1=8 dot clk
VGA Slow Text	0	0	0	1	X
VGA Slow Graphics	0	0	1	1	1
VGA Text 8 Dot	0	0	0	0	1
VGA Text 9 Dot	0	0	0	0	0
VGA Graphics	0	0	1	0	1
Enhanced Text	0	1	X	X	X
Hi-Res 256	1	X	X	X	X

Table 4 FIFO Values for C and K

			VGA Slow Text	VGA Slow Graphics	VGA Text 8 Dot	VGA Text 9 Dot	VGA Graphics	Enhanced Text	High-Res 256
Pixels per Level			16	16	8	9	8	8	4
512 KB	C		15.45	15.45	15.45	15.45	15.45	15.45	15.45
	K	CRT, Single Mono, Single Color, TFT	29.5	19.5	29.5	29.5	19.5	29.5	19.5
		Dual Mono	33.5	23.5	33.5	33.5	23.5	33.5	23.5
		Dual Color	49.5	39.5	49.5	49.5	39.5	49.5	39.5
1024 KB	C		15.45	7.45	15.45	15.45	7.45	15.45	7.45
	K	CRT, Single Mono, Single Color, TFT	27.5	15.5	27.5	27.5	15.5	27.5	15.5
		Dual Mono	29.5	17.5	29.5	29.5	17.5	29.5	17.5
		Dual Color	37.5	25.5	37.5	37.5	25.5	37.5	25.5

Example 1
Program the PLL1 Set 0 clock frequency to 25 MHz

To determine the numerator and denominator for PLL1 Set 0, calculate the numerator and denominator for all possible combinations. Use the combination which is the closest match to the frequency 25 MHz (numerator = 7, denominator = 4).

Next, calculate the Gain Select Bits:

- Freq < 31 MHz, so Gain Select Bit 0 = 1
- Freq < 51 MHz, so Gain Select Bit 1 = 1

```
PLL1_Set0_Denominator = 4;
PLL1_Set0_Numerator = 7;
PLL1_Set0_Gain = 0xc0;
outpw(0x3de, (PLL1_Set0_Denominator << 8) | 0x21);
outpw(0x3de, ((PLL1_Set0_Numerator | PLL1_Set0_Gain) << 8) | 0x20);
```

Note that after PLL1 Set 0 is changed, the FIFO must be programmed.

Example 2
Program the proper FIFO registers given the following:

- PLL1 Set 0 changed to 25 MHz (pixel clock)
- PLL2 currently at 39.4 MHz (memory clock)
- CRT is used
- in VGA graphics mode
- 1024KB video memory

To determine which FIFO registers to program, refer to Table 2, “FIFO/Pixel Clocks,” on page 123. This table indicates that PLL1 Set 0 requires the VGA Graphics AUX[38h], Slow Dot Graphics AUX[3Ah], and Slow Dot Text AUX[3Bh] FIFO registers to be programmed.

To determine the numerators and denominators for the PLL clocks, read the respective registers:
PLL1n = 7, PLL1d = 4, PLL2n = 11, PLL2d = 4

The constants, taken from Table 4, are as follows:

Table 5 C and K FIFO Values for Example 2

	VGA Slow Text	VGA Slow Graphics	VGA Graphics
Pixels per Level	16	16	8
C	15.45	7.45	7.45
K	27.5	15.5	15.5

$$LowLevel[VGAGraphicsFIFO] = TRUNCATE\left\{\left[\frac{C \times PLL1n \times PLL2d}{PixPerLevel \times PLL1d \times PLL2n}\right] + 4\right\}$$

$$= TRUNCATE\left\{\left[\frac{7.45 \times 7 \times 4}{8 \times 4 \times 11}\right] + 4\right\}$$

$$= TRUNCATE\{4.59\}$$

$$= 4$$

$$HighLevel[VGAGraphicsFIFO] = ROUNDUP\left\{\left[\frac{(K \times PLL1n \times PLL2d) + ((43.3 \times 10^{-9}) \times REF \times PLL1n \times PLL2n)}{PixPerLevel \times PLL1d \times PLL2n}\right] - 1\right\}$$

$$= ROUNDUP\left\{\left[\frac{(15.5 \times 7 \times 4) + ((43.3 \times 10^{-9}) \times (14.31818 \times 10^{6}) \times 7 \times 11)}{8 \times 4 \times 11}\right] - 1\right\}$$

$$= ROUNDUP\{0.37\}$$

$$= 1$$

$$LowLevel[SlowGraphicsFIFO] = TRUNCATE\left\{\left[\frac{C \times PLL1n \times PLL2d}{PixPerLevel \times PLL1d \times PLL2n}\right] + 4\right\}$$

$$= TRUNCATE\left\{\left[\frac{7.45 \times 7 \times 4}{16 \times 4 \times 11}\right] + 4\right\}$$

$$= TRUNCATE\{4.30\}$$

$$= 4$$

$$HighLevel[SlowGraphicsFIFO] = ROUNDUP\left\{\left[\frac{(K \times PLL1n \times PLL2d) + ((43.3 \times 10^{-9}) \times REF \times PLL1n \times PLL2n)}{PixPerLevel \times PLL1d \times PLL2n}\right] - 1\right\}$$

$$= ROUNDUP\left\{\left[\frac{(15.5 \times 7 \times 4) + ((43.3 \times 10^{-9}) \times (14.31818 \times 10^{6}) \times 7 \times 11)}{16 \times 4 \times 11}\right] - 1\right\}$$

$$= ROUNDUP\{-0.32\}$$

$$= 0$$

From similar calculations the following values would be found:

$$LowLevel[SlowTextFIFO] = 4$$
$$HighLevel[SlowTextFIFO] = 1$$

```
WriteAux(0x38, (HighLevel[VGAGraphicsFIFO] << 4) | LowLevel[VGAGraphicsFIFO]);
WriteAux(0x3A, (HighLevel[SlowGraphicsFIFO] << 4) | LowLevel[SlowGraphicsFIFO]);
WriteAux(0x3B, (HighLevel[SlowTextFIFO] << 4) | LowLevel[SlowTextFIFO]);
```

Example 3

Program the proper FIFO registers given the following:

- PLL1 Set 0 changed to 14 MHz (pixel clock)
- PLL2 currently at 39.4 MHz (memory clock)
- dual monochrome LCD is used
- High-Resolution 256 mode
- 512KB video memory

Because this system uses an LCD, all FIFO registers must be programmed using the PLL1 Set 0 pixel clock.

$$
\begin{aligned}
LowLevel[EnhancedFIFO] &= TRUNCATE\left\{\left\lceil\frac{C\times PLL1n\times PLL2d}{PixPerLevel\times PLL1d\times PLL2n}\right\rceil+4\right\}\\
&= TRUNCATE\left\{\left\lceil\frac{15.45\times 1\times 4}{4\times 1\times 11}\right\rceil+4\right\}\\
&= TRUNCATE\{5.40\}\\
&= 5
\end{aligned}
$$

$$
\begin{aligned}
HighLevel[EnhancedFIFO] &= ROUNDUP\left\{\left\lceil\frac{(K\times PLL1n\times PLL2d)+((43.3\times 10^{-9})\times REF\times PLL1n\times PLL2n)}{PixPerLevel\times PLL1d\times PLL2n}\right\rceil-1\right\}\\
&= ROUNDUP\left\{\left\lceil\frac{(23.5\times 1\times 4)+((43.3\times 10^{-9})\times(14.31818\times 10^{6})\times 1\times 11)}{4\times 1\times 11}\right\rceil-1\right\}\\
&= ROUNDUP\{1.29\}\\
&= 2
\end{aligned}
$$

From similar calculations the following values would be found:

$$
\begin{aligned}
LowLevel[VGAGraphicsFIFO] &= 4\\
HighLevel[VGAGraphicsFIFO] &= 1\\
LowLevel[VGATextFIFO] &= 4\\
HighLevel[VGATextFIFO] &= 1\\
LowLevel[SlowGraphicsFIFO] &= 4\\
HighLevel[SlowGraphicsFIFO] &= 0\\
LowLevel[SlowTextFIFO] &= 4\\
HighLevel[SlowTextFIFO] &= 0
\end{aligned}
$$

```
WriteAux(0x38, (HighLevel[VGAGraphicsFIFO] << 4) | LowLevel[VGAGraphicsFIFO]);
WriteAux(0x39, (HighLevel[VGATextFIFO] << 4) | LowLevel[VGATextFIFO]);
WriteAux(0x3A, (HighLevel[SlowGraphicsFIFO] << 4) | LowLevel[SlowGraphicsFIFO]);
WriteAux(0x3B, (HighLevel[SlowTextFIFO] << 4) | LowLevel[SlowTextFIFO]);
WriteAux(0x3C, (HighLevel[EnhancedFIFO] << 4) | LowLevel[EnhancedFIFO]);
```

MANUAL SETTING OF POWER STATES ON THE SPC8110

This section provides a brief description of how to implement the same default SPC8110 Power Save States the SPC8110 BIOS uses as default, with accompanying sample code. These states correspond to the VESA VBE/PM states ON, STAND BY, SUSPEND, OFF, and REDUCED ON and are the recommended power state settings.

Comments

- The PS Skew values must be programmed only for LCD panels. Refer to the SPC8110 Hardware Functional Specification document for details.
- The BIOS uses PSM 2 only and does not use PSM 1, 3, or 4. All VBE/PM power states can be obtained by using one software PSM.
- If the CRT capabilities are unknown, on the side of caution you may wish to program the state of the DPMS bits to ON. However, most new CRTs will support DPMS states.
- When a PSM is initiated, certain actions can be programmed to occur:
    - Gray scale reduced
    - PCLK slowed down
    - Display turned off
    - new DPMS state (for VESA Display Power Management Signal protocol)
    - LCD backlight can be turned off
    - PCLK turned off
    - MCLK turned off
- Power Save State 0 powers up the system
- Power Save States 1-4 power down the system, but each will save a different amount of power; the amount of power saved is based on the number and type of SPC8110 features which have been turned off.
- When in Power Up, wait for PLL lock before reading or writing to either the display memory or the RAMDAC.

Example code

```
Power Save State 0: ON
//
// Clear PSM2 enable bit in AUX[15h]
//
outp(0x3de, 0x15);
val = inp(0x3df) & 0xdf;
outp(0x3df, val);
//
// Wait until PLL locked.
// Choose an appropriate time-out delay.
//
timeout = 0xffff;
do
   {
   outp(0x3de, 0x28);
   val = inp(0x3df) & 0x03; // bits PLL1 LOCK and PLL2 LOCK
   }
while ((timeout-- > 0) && (val != 0x03));
----------------------------
Power Save State 1: STAND BY
//
// Reduce Gray Scale : YES (Valid only for LCD)
// PCLK Slow         : NO
// Display Off       : YES
// DPMS              : STAND BY
// LCD Backlight Off : YES
// PCLK Off          : NO
// MCLK Off          : NO
//
outp(0x3de, 0x12);
outp(0x3df, 0x2d);  // AUX[12h] = 00101101b
//
// Set PSM2 enable bit in AUX[15h]
//
outp(0x3de, 0x15);
val = inp(0x3df) | 0x20;
outp(0x3df, val);
---------------------------
Power Save State 2: SUSPEND
//
// Reduce Gray Scale : YES (Valid only for LCD)
// PCLK Slow         : NO
// Display Off       : YES
// DPMS              : SUSPEND
// LCD Backlight Off : YES
// PCLK Off          : NO
// MCLK Off          : YES
```

```
//
outp(0x3de, 0x12);
outp(0x3df, 0xb5);  // AUX[12h] = 10110101b
//
// Set PSM2 enable bit in AUX[15h]
//
outp(0x3de, 0x15);
val = inp(0x3df) | 0x20;
outp(0x3df, val);
-----------------------
```

Power Save State 3: OFF

```
//
// Reduce Gray Scale : YES (Valid only for LCD)
// PCLK Slow         : NO
// Display Off       : YES
// DPMS              : OFF
// LCD Backlight Off : YES
// PCLK Off          : YES
// MCLK Off          : YES
//
outp(0x3de, 0x12);
outp(0x3df, 0xfd);  // AUX[12h] = 11111101b
//
// Set PSM2 enable bit in AUX[15h]
//
outp(0x3de, 0x15);
val = inp(0x3df) | 0x20;
outp(0x3df, val);
-----------------------------
```

Power Save State 4: REDUCED ON

```
//
// Reduce Gray Scale : YES (Valid only for LCD)
// PCLK Slow         : NO
// Display Off       : NO
// DPMS              : ON
// LCD Backlight Off : NO
// PCLK Off          : NO
// MCLK Off          : NO
//
outp(0x3de, 0x12);
outp(0x3df, 0x01);  // AUX[12h] = 00000001b
//
// Set PSM2 enable bit in AUX[15h]
//
outp(0x3de, 0x15);
val = inp(0x3df) | 0x20;
outp(0x3df, val);
```

IDENTIFYING THE SPC8110

The purpose of the code examples below are to demonstrate the methods to identifying the SPC8110 graphics controller. We have two recommended methods:

1. Identifying the SPC8110 through a Seiko Epson BIOS specific BIOS call:

Sample code (in C):

```
//-------------------------------------------------------------------------

 // try SOLLEX call to get chip product code & revision

 regs.x.ax = 0x7fff;
 regs.h.bl = 1;
 int86(0x10, &regs, &regs);

 if (regs.x.ax == 0x7f)    //SOLLEX call is successful
    {
    if (regs.h.bh == 0xf4)
       {
       if (regs.h.bl <= 2)
          printf( "SPC8110 Test Samples detected.\n" );
       else if (regs.h.bl == 0x0a)
          printf( "Mass Production version of SPC8110 detected.\n");
       else
          printf( "SPC8110 not found.\n" );
       }
    else
       printf( "SPC8110 not found.\n" );
    }
//-------------------------------------------------------------------------
```

2. By reading registers within the SPC8110 and interpreting the register contents.

Sample code (in C):

```
//---------------------------------------------------------------------------

 outp(0x3de, 8);
 tmp = inp(0x3df);                    // read AUX[8]

 if ((tmp & 0xe0) == 0xe0)
    {
    outp(0x3de, 0xf);                 // read AUX[0f]
    tmp1 = inp(0x3df);

    if ((tmp1 & 0xf8) == 0xa0)
       {
       if (tmp1 == 0xa2)
          {
          outp(0x3de, 0x20);

          if (inp(0x3df) & 0x20)
             printf( "Mass Production version of SPC8110 detected.\n" );
          else
             printf( "SPC8110 Test Samples detected.\n" );
          }
       else
          printf( "SPC8110 Test Samples detected.\n" );
       }
    else
       printf( "SPC8110 not detected.\n" );
    }
 else
    printf( "SPC8110 not detected.\n" );

//---------------------------------------------------------------------------
```

ID BIT LIMITATION

Bit 5 of AUX[20h] is bit 3 of the Revision Code for the SPC8110F0A. While always a readback of 1, this bit must always be written as a 0. Software should take the precaution of masking this bit to 0 before proceeding to write anything to AUX[20h].

ENABLING AND DISABLING THE SPC8110

The information on Enabling the SPC8110 is contained in the examples located earlier in the documents “Initializing the SPC8110 in Mode 03h without a BIOS”; “Initializing the SPC8110 in Mode 12h without a BIOS”; and “Initializing the SPC8110 in Mode 101h without a BIOS”. Step 1 of each of these examples demonstrates the way to use port 3C3h to enable the chip.

The code required to disable the chip is:

```
mov     dx, 3C3h
mov     al, 0
out     dx, al
ret
```

USING THE SPC8110 DISPLAY MEMORY

The SPC8110F0A can support up to 1 MB of display memory. Any memory above 256 KB is considered non-VGA and no standard VGA interface for accessing this memory exists. The purpose of this section is to describe to a programmer how to access all memory available to the SPC8110F0A.

There are three ways of accessing upper memory:

1. Paging
2. Linear Mapping
3. Using VESA BIOS Extensions

Paging

- The SPC8110 can support 1024 KB (1 MB), but because of the PC architecture, the SPC8110 is limited to accessing this memory through 64 KB windows below the 1 MB segment. This limitation occurs while the processor is in Real Mode, because while in Real Mode, the Intel architecture can only access up to 1 MB of RAM. To access the entire 1024 KB, the SPC8110 segments the memory into 16 pages of 64 KB each, which are selected using a paging register.
- To access a given page, the Page Select Register must be programmed with the desired page number, causing the SPC8110 to map the page into the 64 KB block located at A0000-AFFFF.

Linear Mapping

- The Linear mapping technique maps all of the display memory to one large block of memory on the host computer.
- On PC-compatible computers, linear mapping requires the computer to be placed in protected mode in order to address memory above 1 MB.

Using VESA BIOS Extensions

- For systems with a VESA compatible BIOS, paging can also be performed using standard VESA techniques. Refer to the "VBE Core Functions Version 2.0" document for more information.

Special Notes About Hardware Cursor And Half Frame Buffer

- When using upper memory, never overwrite the hardware cursor memory location or half frame buffer location.
- The hardware cursor requires a 1K byte block of graphics memory.
- Typically, the hardware cursor and Half Frame Buffer are located by the BIOS and Drivers at the end of display memory. To avoid these memory sections, determine the starting address locations for both sections, and use the smaller address-1 as the upper limit.
- If the hardware cursor is not enabled, it is not necessary to avoid the hardware cursor memory section.
- The Half Frame Buffer is only used to support dual drive passive panels. This is not a concern for other panels. If the system does not use a dual drive passive panel, it is not necessary to avoid the Half Frame Buffer memory section.

Banking Sample Code

```
//
// The following sample code gets the address of the hardware cursor and
// half frame buffer, and then determines which of these two addresses is
// the smallest. The smallest address-1 is considered the largest address
// usable by the program for display. Display memory is then filled from
// display memory location 00000h to this upper limit. Each bank of
// display memory is filled with a different character. For example,
// bank 0 is filled with the ASCII code for 'A', bank 1 is filled with
// 'B', etc.
//
// Each bank points to a 64KB aperture within the 1 MB display memory.
//

//
// Get Hardware Cursor address
//
outp(0x3d4, 0x36);
pCursor = (inp(0x3d5) & 0x03) << 8;

outp(0x3d4, 0x37);
pCursor |= inp(0x3d5);

pCursor <<= 8;

printf("pCursor = %lXh\n", pCursor);

//
// Get Half Frame Buffer address
// Assume that system is not using flag addressing mode
//
outp(0x3de, 0x1c);
pBuffer = (inp(0x3df) & 0x0f) * 0x10000L;

printf("pBuffer = %lXh\n", pBuffer);

//
// Find the smallest address.
// The maximum usable display address is one less than this address.
//
if (pCursor < pBuffer)
   pMaxAddr = pCursor-1;
else
   pMaxAddr = pBuffer-1;

printf("Available Display Memory: 0h - %lXh\n", pMaxAddr);

//
```

```
// Make sure Memory Paging is enabled
//
outp(0x3de, 5);
val = (unsigned char) inp(0x3df);

val |= 0x01;   // Set Memory Page Enable
val &= 0xfd;   // Clear Flag Buffer enable

outp(0x3df, val);

//
// Fill display memory using banking
//
for (bank = 0; bank < 16; ++bank)
   {
   FP_SEG(pDisplay) = 0xa000;
   FP_OFF(pDisplay) = 0x0000;

   outp(0x3cd, bank);

   //
   // Multiply bank by 64 KB to determine display address
   // (00000h = start of display memory)
   //
   DisplayAddr = bank * 0x10000L;

   for (count = 0; (count < 0x10000) && (DisplayAddr <= pMaxAddr);
                  ++count, ++DisplayAddr)
      *pDisplay++ = (unsigned char) ('A' + bank);
   }
```

Linear Mapping Sample Code

```
//
// The following sample code writes 0x55 to 1 MB of display memory
// using linear mapping in planar mode 12h. It is assumed that the host computer is
// already in protected mode. The following steps are followed:
//
// 1. Get the physical address of the Video Buffer.
// 2. Translate the physical address into the logical address.
//    This is done by TRANSLATE(), which must be provided by
//    the protected mode libraries.
// 3. Set up the SPC8110; go into mode 12h and go into linear mapping.
// 4. Write to 1 MB memory. This memory is treated as one large array.
// 5. Get out of linear mapping by issuing a mode set command.
//    A mode set reinitializes the SPC8110 registers.
//

//
```

```
// Get address of Video Buffer. Note that registers provide A31-A20.
//
outp(0x3de, 0x1a);
CurrentBufferAddr = (inp(0x3df) & 0x0f) << 8;
outp(0x3de, 0x1b);
CurrentBufferAddr |= inp(0x3df);
CurrentBufferAddr <<= 20;

//
// The protected mode libraries must provide a function such as
// TRANSLATE, which translates from a physical address to a
// logical address.
//
LogicalAddress = TRANSLATE(CurrentBufferAddr);


//
// Go to mode 12h
//
_asm
   {
   mov ax,12h
   int 10h
   }

//
// Go to linear mapping
//
outp(0x3de, 5);
val = (unsigned char) inp(0x3df);
val|= 0x02;
outp(0x3df, val);

for (count = 0; count < 0x100000; ++count)
   LogicalAddress[count] = 0x55;

//
// Go to mode 3h
// A mode set on the Seiko Epson BIOS reinitializes the SPC8110 registers,
// and thus gets out of linear mapping mode.
//
_asm
   {
   mov ax,3h
   int 10h
   }
```

FRAME BUFFER FOR DUAL DRIVE PANELS

Dual Drive Passive LCD panels require a half-frame buffer on the SPC8110F0A. When initializing the SPC8110 to use a Dual Drive Passive panel the Frame Buffer must be initialized in the Auxiliary Registers. The best place to locate this buffer is in the last 64K bytes of graphics memory. Calculation of the Frame Length is critical, use exact panel sizes. To setup the Half Frame Buffer the following registers must be programmed; AUX[1Ch], AUX[1Dh], and AUX[1Eh].

- The value for AUX[1Ch] is determined by the table below.

Table 6 Recommended Values for AUX[1Ch]

	AUX[1Ch] Values for Available Memory	
	512K Bytes	1024K Bytes
Color Panels ≥ 800x600	06h*	0Eh
All Panels < 800x600	07h	0Fh

**Limited to 16 colors @ 800x600*

- The values for AUX[1Dh] and AUX[1Eh] are calculated with the formula:

$$Length = \left[\left(\frac{Height \times Width}{2} \times Colors\right) / 32\right] - (2 \times Colors) - 1$$

where Colors = 1 for monochrome panel, 3 for color panel.

Example: 640x480 dual color panel in 512KB of RAM

$$AUX[1Ch] = 07h$$

$$Length = \left[\left(\frac{640 \times 480}{2} \times 3\right) / 32\right] - (2 \times 3) - 1$$

$$= (460800 / 32) - (7)$$

$$= 14393$$

$$= 3839h$$

$$\therefore AUX[1Dh] = 38h$$

$$AUX[1Eh] = 39h$$

640x480 2-BITS PER PIXEL (BPP) GRAPHICS MODE PROCEDURE

This Packed Pixel mode is the graphics memory model used in Microsoft Windows CE version 1.x. This mode is not supported by the BIOS, but can be set by following the procedure below:

1. Use BIOS call to set mode 12h.
2. Set CRTC Index [17h] to A3h.
3. Set Graphics Controller Index [05h] to 30h.
4. Set Graphics Controller Index [06h] to 03h. (to enable both A000 & B000 Segment)
5. Set Graphics Controller Index [07h] to 00h.
6. Set Attributes Controller Index [12h] to 03h.
7. Set Attributes Controller Index to 32h. (to re-enable EGA palette)
8. Set Sequencer Index [00h] to 00h. (turn off sequencer)
9. Set Sequencer Index [02h] to 03h.
10. Set Sequencer Index [04h] to 02h.
11. Set Sequencer Index [00h] to 03h. (reset sequencer).
12. Program the DAC registers to the desired 4 grey shades.
13. You are now ready to access the video memory.

The first 8000h pixels can be accessed through segment A000h. The rest of the pixels can be accessed through segment B000h.

The following are the VGA register values for this video mode. Note that a 640x480 Dual Monochrome STN panel is attached, with 1024KB of RAM on the SDU8110B0E evaluation board:

```
CRT controller Registers Set A:
CRT index = 0f
CRTC A[00]=5f  CRTC A[01]=4f  CRTC A[02]=50  CRTC A[03]=82  CRTC A[04]=54
CRTC A[05]=80  CRTC A[06]=0b  CRTC A[07]=3e  CRTC A[08]=00  CRTC A[09]=40
CRTC A[0a]=00  CRTC A[0b]=00  CRTC A[0c]=00  CRTC A[0d]=00  CRTC A[0e]=00
CRTC A[0f]=f0  CRTC A[10]=ea  CRTC A[11]=8c  CRTC A[12]=df  CRTC A[13]=28
CRTC A[14]=00  CRTC A[15]=e7  CRTC A[16]=04  CRTC A[17]=a3  CRTC A[18]=ff
CRTC A[24]=80  CRTC A[30]=00  CRTC A[31]=00  CRTC A[32]=00  CRTC A[33]=00
CRTC A[34]=00  CRTC A[35]=00  CRTC A[36]=00  CRTC A[37]=00  CRTC A[38]=00

//640x480 dual mono panel is attached
CRT controller Registers Set B:
CRTC B[00]=1e  CRTC B[01]=50  CRTC B[06]=f1  CRTC B[07]=00  CRTC B[10]=eb
CRTC B[11]=0d  CRTC B[12]=78

Sequencer Registers Set A:
SEQ index = 02
SEQ A[00]=03   SEQ A[01]=01   SEQ A[02]=0f   SEQ A[03]=00   SEQ A[04]=02
```

```
Graphics Controller Registers:
GFX index = 03
GFX[00]=00     GFX[01]=00     GFX[02]=00     GFX[03]=00     GFX[04]=00
GFX[05]=30     GFX[06]=03     GFX[07]=00     GFX[08]=ff

Attribute Controller Registers set A:
Attrib index = 20  Palette enabled
ATTR A[00]=00  ATTR A[01]=01  ATTR A[02]=02  ATTR A[03]=03  ATTR A[04]=04
ATTR A[05]=05  ATTR A[06]=14  ATTR A[07]=07  ATTR A[08]=38  ATTR A[09]=39
ATTR A[0a]=3a  ATTR A[0b]=3b  ATTR A[0c]=3c  ATTR A[0d]=3d  ATTR A[0e]=3e
ATTR A[0f]=3f  ATTR A[10]=01  ATTR A[11]=00  ATTR A[12]=03  ATTR A[13]=00
ATTR A[14]=00

General Registers:
Input Status Register 0 = 10
Input Status Register 1 = 04
Miscellaneous Output Register = e3
Video Subsystem Enable Register (3c3) = 01
Page Register: 00
```

S-MOS SYSTEMS
A Seiko Epson Affiliate

SPC8110F0A VGA LCD/CRT Controller

8110CFG.EXE BIOS Configuration Utility

Drawing Office No. X07-UI-015-02

Copyright © 1996, 1997 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

8110CFG.EXE BIOS CONFIGURATION UTILITY

8110CFG is the OEM utility for custom configuring the SPC8110 VGA Controller BIOS binary image. This utility allows the OEM, or product support people to easily customize the Seiko Epson SPC8110 BIOS for the OEM's requirements. These configurations may be as simple as modifying the Signon Message or as complex as modifying the default panel support.

8110CFG consists of four files; 8110cfg.exe, 8110cfg.hlp, 8110cfg.hx, and 8110cfg.hs. This program has three user levels incorporated into it for ease of operation. These levels are:

- **typical**
- **sophisticated**
- **expert**

We highly recommend reading the on-line help documentation for details on the configuration options before making any changes.

Program Requirements

Video Controller	**:**	Any VGA
Display Type	**:**	Any VGA
BIOS	**:**	Any manufacturer's VGA BIOS
DOS Program	**:**	Yes
DOS Version	**:**	3.0 or greater
Windows Program	**:**	No
Windows DOS Box	**:**	Yes, Windows 3.1x and Windows 95
Windows DOS Full Screen	**:**	Yes, Windows 3.1x and Windows 95
OS/2 DOS Full Screen	**:**	2.0 or greater

Installation

Copy the files **8110cfg.exe**, **8110cfg.hlp**, **8110cfg.hx**, and **8110cfg.hs** to a directory that is in the DOS path on your hard drive.

Usage

8110CFG is invoked from the DOS command line as follows:

```
8110cfg [/s|/x][/?][filename]
```

where:

no parameters	starts 8110CFG in the typical user mode.
/s	starts 8110CFG in the sophisticated user mode.
/x	starts 8110CFG in the expert user mode.
/?	displays the copyright notice and version number.
filename	The optimal filename of the EPRON image will be loaded at startup. If no file is specified, you will be required to load the file using the FILE OPEN option from the program menus.

Comments

The 8110CFG program is menu driven with extensive on-line context sensitive help. The user interface is by either keyboard or mouse.

The following BIOS properties can be configured:

By the Typical User - These are the most common BIOS configuration changes, and typically can be modified without risking functionality.

- IBM compatibility (mode 7, mode 11)
- Compatible DAC programming - IBM ISA or IBM PS/2
- BIOS boot segment
- BIOS signon message
- Panel preferences for each supported panel:
  - text mode expansion
  - text mode screen polarity
  - graphic mode expansion
  - graphic mode screen polarity
  - gray scale method
  - autocenter
  - blink rate
  - cursor shape
- VESA DDC support

By the Sophisticated User - These are more sophisticated changes which can affect the functionality of the graphics system. These changes are typically made by the system board designer who understands the layout.

Same as the Typical User plus the following:

- Re-assign MD lines - any of the 16 supported panels can be assigned to any MD lines value, thus selecting a “default” panel
- IRQ2 enable / tristate
- Number of gray levels - 32 or 64
- Linear buffer location (for VL-BUS implementations)
- Memory Clock frequency
- Panel Dithering options

By the Expert User - These are the most sophisticated changes that can be made without producing a custom BIOS. These options allow the definition of new panel types, and should only be made by qualified individuals who understand both the chip and panel interface issues. Changes with this option enabled may create a BIOS which can damage panels.

Same as the Sophisticated User plus the following:

Edit a new panel entry - up to 16 different panels. Any panel can be assigned to any MD lines value.

A variety of configurable options are available to accommodate various panel types.

The following properties are configurable independently for each panel:

- panel type (dual, single, 8-bit, 4-bit, MIM ...)
- frame rate
- pixel clock
- TFT vertical line start
- panel dimensions
- miscellaneous LCD support bits:
  - LP timings - 8 / 6 clocks
  - WF count - 0 to 63
  - configurable power off protection for BIOS video mode set (skew A, B, C, D)

Using Your Newly Configured BIOS

To use your newly configured BIOS, program a new EPROM with the new image. Depending on your EPROM programmer, you may need to convert the image format from binary to one which your EPROM programmer can use.

THIS PAGE LEFT BLANK

S-MOS
SYSTEMS
A Seiko Epson Affiliate

BOLD.EXE Display Utility

Drawing Office No. X00Z-B-004-02

Copyright © 1991, 1997 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

BOLD.EXE DISPLAY UTILITY

BOLD.EXE provides more readable text on LCD or CRT displays by changing the default text attribute to high-intensity white from its normal low-intensity white (for text modes only). It can be installed as either a DOS device driver or as a Terminate and Stay Resident (TSR) program. BOLD modifies the user palette profile feature of a VGA BIOS to change the standard low-intensity text attribute 07h to the value 3Fh.

Note
Other text mode application programs which modify the RAMDAC values of entry 3Fh may interfere with the operation of BOLD.

Program Requirements

Video Controller	**:**	Any VGA
Display Type	**:**	LCD or CRT
BIOS	**:**	Any manufacturer's IBM compatible VGA BIOS
DOS Program	**:**	Yes
DOS Version	**:**	3.0 or greater
Windows Program	**:**	No
Windows DOS Box	**:**	Windows 95 only
Windows DOS Full Screen	**:**	Windows 95 only
OS/2 DOS Full Screen	**:**	Yes

Installation

Copy the program file **bold.exe** to a directory that is in the DOS path on your hard drive.

Usage

BOLD may be installed either from the DOS command line as a TSR, or from the CONFIG.SYS file as a device driver.

DOS Command Line

```
bold on|off
```

When BOLD is loaded from the DOS command line with the "on" option the text attribute is changed to high intensity. Running BOLD with the "off" option returns the text attribute to normal.

For example: typing at the DOS prompt

```
bold on
```

will load BOLD.EXE as a TSR and will change the text attribute.

An incorrect command line parameter will cause the display of a usage screen.

DOS Device Driver

To load the program as a device driver, add the following line to the CONFIG.SYS file:

```
device=[path]bold.exe on|off
```

For example the line:

```
device=c:\bin\bold.exe on
```

in the config.sys file loads BOLD.EXE as a device driver and changes the text attribute.

Comments

- The program will install under graphics modes but will not take effect until the display is in a text mode.
- This program cannot be installed from within a DOS shell (when a program provides access to the DOS command line and returns to the program via the EXIT command, such as a Windows 3.1x DOS box). It **will** operate in a DOS session under OS/2 or Windows 95.

Program Messages

`ERROR: Cannot operate in a DOS command shell.`

The program cannot be installed in a DOS shell. Exit your application and re-run BOLD.EXE.

`ERROR: VGA required.`

The program works in VGA text mode only.

`ERROR: BOLD is already installed as a device driver and cannot be installed again.`

BOLD cannot be installed more than once in the CONFIG.SYS file. Remove the redundant line from the CONFIG.SYS.

`Changes to the text mode attribute will not affect the current graphics mode. Changes will appear when the display is placed in text mode.`

BOLD is a text mode program. When the display is in graphics mode BOLD will not alter the appearance of the display. When the display is switched to text mode any changes invoked during graphics mode will become apparent.

S-MOS
SYSTEMS
A Seiko Epson Affiliate

DISPLAY.COM Display Selection Utility

Drawing Office No. X00Z-B-013-02

Copyright © 1993, 1997 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

DISPLAY.COM DISPLAY SELECTION UTILITY

DISPLAY.COM is a DOS program which allows the selection of the active display device. When both a CRT and an LCD are attached, DISPLAY will allow switching to either or both displays.

Program Requirements

Video Controller	**:**	SPC8106, SPC8108, SPC8110
Display Type	**:**	LCD or CRT
BIOS	**:**	Seiko Epson Video BIOS
DOS Program	**:**	Yes
DOS Version	**:**	3.0 or greater
Windows Program	**:**	No
Windows DOS Box	**:**	No
Windows DOS Full Screen	**:**	Yes, Windows 3.1x and Windows 95
OS/2	**:**	No

Installation

Copy the file **display.com** to a directory that is in the DOS path on your hard drive.

Usage

From the DOS command line type:

```
display [crt | lcd | doublescan]
```

To view the available options and other relevant program information, at the DOS prompt type:

```
display
```

DISPLAY will display a usage screen similar to the following:

```
DISPLAY.COM Version 1.00
Utility to select the active display.
Copyright (c) Seiko Epson Corp. 1994. All rights reserved.
         usage:   DISPLAY CRT turns on CRT only
                  DISPLAY LCD turns on LCD only
                  DISPLAY DOUBLESCAN turns on CRT and LCD in 480 line modes
                                     turns on CRT only in all other modes
```

Comments

- If the active display mode is currently at a higher resolution than the LCD can support, and a switch to simultaneous display (doublescan) or to the LCD occurs, DISPLAY will fail the request.

Program Messages

`ERROR: Requires Seiko Epson Video BIOS Extensions.`

The Seiko Epson Video BIOS Extensions are not installed as part of the video system.

`ERROR: Requires analog monitor.`

The video controller cannot detect an analog monitor attached to the system.

`ERROR: Cannot switch at this time.`

An attempt to switch to the LCD display occurred while in an operational display mode beyond the capabilities of the LCD display.

`ERROR: Cannot switch while in a Windows DOS Box`

An attempt to switch displays occurred while in a Windows emulated DOS box. Use <ALT>-<Enter> to switch to DOS full screen and try again.

S-MOS
SYSTEMS
A Seiko Epson Affiliate

SPC8110F0A VGA LCD/CRT Controller

HD.EXE Display Utility

Drawing Office No. X07-UI-007-02

Copyright © 1993, 1997 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

HD.EXE DISPLAY UTILITY

HD selects the active display via a Hotkey when both a CRT display and LCD panel are available.

Program Requirements

Video Controller	**:**	SPC8110
Display Type	**:**	LCD or CRT
BIOS	**:**	Seiko Epson VGA BIOS
DOS Program	**:**	Yes
DOS Version	**:**	2.0 or greater
Windows Program	**:**	No
Windows DOS Box	**:**	Yes (with limitations), Windows 3.1x and Windows 95 (see note below)
Windows DOS Full Screen	**:**	Yes (with limitations), Windows 3.1x and Windows 95 (see note below)
OS/2	**:**	No

Note
HD must to be installed outside of a DOS shell. However, once it is installed a hotkey can be pressed either from a Windows DOS box or Windows DOS full screen session to switch the active display.

Installation

Copy the file **hd.exe** to a directory that is in the DOS path on your hard drive. The directory does not need to be in the DOS path; if it is not in the path, then a full path name will have to be specified each time HD is run.

Usage

HD may be loaded as as an executable from the DOS command prompt, or as a device driver from CONFIG.SYS during the boot sequence.

DOS Command Line

To load HD as a TSR, add the following line to your AUTOEXEC.BAT file or enter it at the DOS command prompt:

```
hd [/c][/u][/?]
```

Where:

- /c Changes the hotkey sequence. This switch causes HD to prompt for the new hotkey sequence. Press the desired keys in sequence to form a hotkey sequence. A minimum of one to a maximum of three keys may be used to form the sequence. Use any keys except the following: Caps Lock, Num Lock, Scroll Lock, Print Screen and Pause/Break.
- /u Removes HD from Memory.
- /? Displays HD help message.

DOS Device Driver

To load HD as a device driver, add the following line to your CONFIG.SYS file:

```
device=[path]hd.exe
```

Options are not valid in CONFIG.SYS.

Using HD

HD allows the user to switch displays on the fly. HD is a TSR program, therefore it can be used while running another program. To switch display, use the hotkey sequence displayed when HD is loaded. Press the keys in the sequence indicated and release them simultaneously. Pressing the LCD/CRT Hot Key Sequence (default is F9) will switch among CRT display, LCD panel and Simultaneous Display mode.

Active Display before Key Pressed	Active Display after Key Pressed
CRT	LCD
LCD	Simultaneous Display
Simultaneous Display	CRT

Comments

- HD will install in the primary copy of the DOS command shell only. HD will not install from menu or shell programs.
- This utility requires the Seiko Epson video BIOS and Extensions.
- Rather than keeping a separate data file to store Hot Key Sequence, HD modifies the HD.EXE file on disk. For this reason some virus scanners (i.e. those that maintain checksums or CRCs) may warn of possible virus infection in HD.EXE. This will only occur after a change to the hotkey sequence.
- HD may not always be able to remove itself from resident memory if other TSRs or Device Drivers are installed after it.
- HD occupies 2K of memory.
- A Beep indicates that an error has happened. This can be one of the following cases:

    1. Hot Key Sequence is entered to switch from CRT to LCD. However, LCD display may not be able to handle the current video mode. (i.e., 640x480 LCD panel cannot display mode 6Ah (800x600).)
    2. Hot Key Sequence is entered to switch from LCD to Simultaneous Display mode. However, a CRT display is not attached to the video board.
    3. Hot Key Sequence is entered, however the Video Controller is not ready to switch display yet. An error happened while using Seiko Epson BIOS Extension calls to switch display.

Program Messages

`ERROR: HD is already installed and cannot be installed again.`

HD has been installed as a TSR or a device driver. It cannot be installed twice.

`ERROR: VGA required.`

A VGA display adapter must be present to run HD.

`ERROR: Requires Seiko Epson Video BIOS Extensions.`

The Seiko Epson Video BIOS Extensions are not installed as part of the video system.

`ERROR: As a device driver, HD does not accept any command line argument.`

Run HD without any command line argument in CONFIG.SYS to install HD as a device driver.

`ERROR: SPC8110 required.`

HD requires an SPC8110 LCD VGA Controller.

`ERROR: DOS move file pointer error on HD.EXE.`

A DOS file error has occurred.

`ERROR: DOS read error on HD.EXE.`

A DOS file error has occurred.

`ERROR: DOS write error on HD.EXE.`

A DOS file error has occurred.

`ERROR: DOS close error on HD.EXE.`

A DOS file error has occurred.

`ERROR: cannot operate in a command shell.`

HD cannot install as a TSR when it is run within another program or within an environment like Windows.

`ERROR: Cannot remove HD FROM MEMORY.`

Other TSR programs have been installed after HD in the memory.

`ERROR: cannot open HD.EXE. Update not done.`

HD cannot save the hotkey sequence changes. The file HD.EXE may be read-only; if this is the case, change the attribute of HD.EXE and run it again.

THIS PAGE LEFT BLANK

S-MOS SYSTEMS

A Seiko Epson Affiliate

SPC8110F0A VGA LCD/CRT Controller

LCDF8110.EXE Display Utility

Drawing Office No. X07-UI-012-03

Copyright © 1993, 1997 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

LCDF8110.EXE DISPLAY UTILITY

LCDF8110 is a display utility for changing the LCD panel frame rate using the SPC8110 VGA Controller. This is desirable for adjusting the contrast and viewability of the display under different lighting conditions. LCDF8110 can be loaded as a command line executable with parameters, or as an interactive executable when no parameters are specified. To change the frame rate, LCDF8110 modifies a variety of chip specific registers. LCDF8110 is provided for demonstration purposes only and is not intended for distribution to end-users.

Program Requirements

Video Controller	**:**	SPC8110
Display Type	**:**	LCD
BIOS	**:**	Any manufacturer's IBM compatible VGA BIOS
DOS Program	**:**	Yes
DOS Version	**:**	3.0 or greater
Windows Program	**:**	No
Windows DOS Box	**:**	No
Windows DOS Full Screen	**:**	Yes, Windows 3.1x and Windows 95
OS/2 DOS Full Screen	**:**	2.0 or greater

Installation

Copy the file **lcdf8110.exe** to a directory that is in the DOS path on your hard drive.

Usage

LCDF8110 is invoked from the DOS command line as follows:

```
lcdf8110 [n][/?]
```

where:

- n will set the desired frame rate frequency in decimal Hertz. (e.g. 78 = 78Hz) and will display the parameters programmed as well as the copyright notice and version number.
- /? displays the copyright notice, version number and usage screen.

When no command line option is specified the interactive mode of the program is run. When in interactive mode, LCDF8110 displays a graphic pattern and current frame rate.

The frame rate can be increased/decreased by repeatedly pressing 'U' or 'D'. The quality of the displayed pattern is used to determine the optimal frame rate for a given panel.

Pressing the 'Q' key exits the program.

Comments

- The program will automatically determine the panel type (single or dual) and calculate the correct frame rate.
- High frame rates could potentially damage the panel. Consult the manufacturer's specification for recommended values.
- The frame rate will always return to the default when the computer is restarted.

Program Messages

`ERROR: SPC8110 not present!`

This program requires the SPC8110 VGA LCD controller.

`ERROR: Frame rate not valid!`

The frame rate requested exceeds the video controller limits, try a number closer to 78 Hz.

S-MOS
SYSTEMS
A Seiko Epson Affiliate

LOADROM.EXE BIOS Utility

Drawing Office No. X00Z-B-001-02

Copyright © 1997 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

LOADROM.EXE BIOS UTILITY

LOADROM is a BIOS load program which will take a BIOS image and store the image into system memory, then initialize the load sequence. This can be done two ways:

1. As a Terminate and Stay Resident program (TSR).
2. As a MS-DOS device driver.

This program is useful for testing BIOS images, or loading BIOS images for evaluation boards that do not have a ROM BIOS.

Program Requirements

Video Controller	**:**	Any SPC81xx VGA/LCD Controller
Display Type	**:**	LCD or CRT
BIOS	**:**	Seiko Epson VGA BIOS
DOS Program	**:**	Yes
DOS Version	**:**	3.0 or greater
Windows Program	**:**	No, but can be used in Win 95 CONFIG.SYS
Windows DOS Box	**:**	No
Windows DOS Full Screen	**:**	No
OS/2 DOS Full Screen	**:**	No

Installation

Copy the file **loadrom.exe** to a directory that is in the DOS path on your hard drive. It is recommended that this location be the same location that the BIOS image will reside.

Usage

LOADROM may be installed either from a DOS command line as a TSR, or from the CONFIG.SYS file as a device driver.

DOS Command Line

LOADROM FileSpec

FileSpec	Must be a valid ROM binary image with header 55, AA, size in 512-byte blocks. File byte check sum must be zero.

A typical example of using LOADROM as a TSR is:

```
loadrom eprom.bin
```

DOS Device Driver

To load the program as a device driver, add the following line to the CONFIG.SYS file:

```
device=[path]loadrom.exe [path]binary filename
```

For example the line:

```
device=c:\loadrom.exe c:\eprom.bin
```

in the config.sys file loads eprom.bin into memory by using loadrom as a device driver.

Comments

- This program takes up approximately 32KB of system RAM each time it is executed with a valid BIOS image.
- The BIOS when loaded re-vectors INT 10h, INT 42h and changes low memory contents. If run after other TSR or device drivers are loaded, those vectors and low memory locations will not be the same as before, therefore unexpected results may occur. It is best to run LOADROM as early in the boot process as possible.

Program Messages

`Error reading file!`

A DOS error has occurred while reading the BIOS image file. Check your media using scandisk or equivalent program.

`ROM image does not have signature 55AA!`

The image specified is not in standard BIOS image format and may not be a BIOS image at all. Check your command line.

`ROM image fails check sum!`

The BIOS image fails to check sum to 00h. It is likely that the image was modified and not check summed correctly.

`Error opening file!`

Loadrom had trouble either locating the file, or does not have access privileges to open the file. Check your command line and privilege levels.

S-MOS
SYSTEMS
A Seiko Epson Affiliate

SPC8110F0A Color VGA/CRT Controller

PLLSET.EXE Display Utility

Drawing Office No. X07M-B-001-01

Copyright © 1997 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

PLLSET.EXE DISPLAY UTILITY

PLLSET reads and sets the clock output from PLLs on the SPC8110.

Program Requirements

Video Controller	**:**	SPC8110
Display Type	**:**	LCD or CRT
BIOS	**:**	IBM compatible VGA BIOS
DOS Program	**:**	Yes
DOS Version	**:**	3.0 or greater
Windows Program	**:**	No
Windows DOS Box	**:**	No
Windows DOS Full Screen	**:**	No
OS/2 DOS Full Screen	**:**	No

Installation

Copy the files **pllset.exe** and **pllset.hlp** to a directory that is in the DOS path on your hard drive.

Usage

Pllset can be run as a DOS command line program or as a menu-driven program. As a menu-driven program, the user can execute all the sub-functions through the mouse or tab key. In the command line program, all the sub-functions can be executed through the command line options.

Where:	`PLLSET`	starts PLLSET in menu-driven mode
	`PLLSET /R1`	reads PLL1
	`PLLSET /R2`	reads PLL2
	`PLLSET /W1 v0 v1 v2 [/N]`	programs PLL1 set 0 to v0 MHz programs PLL1 set 1 to v1 MHz programs PLL1 set 2 to v2 MHz
	`PLLSET /W2 v0 [/N]`	programs PLL2 to v0 MHz
	`PLLSET /?`	displays help message

v0, v1, v2 are floating point numbers between 0.0 and 65.0.

/N does not program PLL gain bits.

Comments

- The input clock frequency to the PLLs is assumed to be 14.31818 MHz.
- The mouse can appear frozen after simultaneously pressing the Alt key and a non-letter key. To fix this problem, press the Esc key to go back to normal.

Program Messages

`ERROR: Cannot find gain bits.`

Only SPC8110 Rev. A has gain bits, some test sample chips do not have this function.

S-MOS
SYSTEMS
A Seiko Epson Affiliate

PWRSAVE.EXE Power Save Utility

Drawing Office No. X00Z-B-006-02

Copyright © 1995, 1997 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

PWRSAVE POWER MANAGEMENT UTILITY

PWRSAVE is a power saving utility for SPC81xx series LCD Controllers. It is a multi-purpose utility which allows the OEM to do power measurements or demonstrate the power management capabilities of the VGA LCD controller. PWRSAVE can be configured as a terminate and stay resident (TSR) program or it can be run as a normally terminated DOS application. PWRSAVE is provided for demonstration purposes only and is not intended for distribution to end users.

Program Requirements

Video Controller	**:**	All SPC81xx VGA LCD Controllers
Display Type	**:**	640x480 LCD or CRT
BIOS	**:**	Seiko Epson VGA BIOS
DOS Program	**:**	Yes
DOS Version	**:**	3.2 or greater
Windows Program	**:**	No
Windows DOS Box	**:**	see Note below
Windows DOS Full Screen	**:**	see Note below
OS/2 DOS Full Screen	**:**	2.0 or greater

Note

1. PWRSAVE [x], PWRSAVE /r, and PWRSAVE /off will work under Windows 95.
2. PWRSAVE /u and PWRSAVE /Tn x do not work under Windows 95.

There is an option under the Windows 95 desktop "Prevent DOS Application from Detecting Windows 95". By default this option is turned off. Users should check to ensure that this option is really turned off, otherwise you may have problems running PWRSAVE.

Installation

Copy the file **pwrsave.exe** to a directory that is in the DOS path on your hard drive.

Usage

PWRSAVE can be run from the DOS prompt or from a batch file. To automatically initiate power saving, place "PWRSAVE" in the system's AUTOEXEC.BAT file.

PWRSAVE is invoked from the DOS command line as follows:

```
pwrsave [x] [/u] [/off] [/Tn x] [/r] [/?]
```

Where: x sets Power Save mode x, and returns to original Power Save mode upon keystroke or mouse movement. PWRSAVE.EXE does not stay resident in memory in this case.

/u removes PWRSAVE from memory.

/off disables all power savings.

/Tn x installs PWRSAVE in memory and sets Power Save mode x if there is no keystroke or mouse movement occuring for n seconds. Returns to original Power Save mode upon keystroke or mouse movement.

/r remain in reduced gray scale mode

/? displays the help message.

Note
n is a decimal number from 5 to 3600 and x is a decimal number from 1 to 4 for the SPC8110. PWRSAVE 0 is equivalent to PWRSAVE /off.

Comments

- The /u and /Tn x options do not work when in a Windows DOS full screen session.
- This utility requires the Seiko Epson Video BIOS and Extensions.

Program messages

ERROR: DOS Version 3.2 or later required.

PWRSAVE requires MS-DOS 3.2 or later.

ERROR: SPC81XX with Seiko Epson Video BIOS Extensions required.

PWRSAVE requires an SPC81xx and Seiko Epson Video BIOS. PWRSAVE will fail to operate with any other configuration.

ERROR: Cannot remove PWRSAVE from memory.

PWRSAVE cannot be removed from memory. There may be some other TSRs installed after PWRSAVE.

ERROR: Cannot set Power Save mode x.

An error ocurred while setting Power Save mode x.

ERROR: Cannot operate in a Command Shell.

PWRSAVE has detected that it was requested to run in a DOS command shell as a child program. It will not operate in this mode. Exit the command shell and the application spawning the command shell, and re-run PWRSAVE.

ERROR: This option does not work with Windows 95.

PWRSAVE /u and PWRSAVE /Tn x do not work under Windows 95.

S-MOS
SYSTEMS
A Seiko Epson Affiliate

READCHIP.EXE Diagnostic Utility

Drawing Office No. X00Z-B-007-02

Copyright © 1994, 1997 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

READCHIP.EXE DIAGNOSTIC UTILITY

READCHIP is an OEM utility program which enables the user to read the VGA register contents, lower BIOS data area and video RAM. It is a useful utility for OEMs wishing to submit a problem report for the graphics controller. If run with a Seiko Epson BIOS, it will try to interpret the BIOS settings.

Program Requirements

Video Controller	**:**	Any VGA
Display Type	**:**	LCD or CRT
BIOS	**:**	Any manufacturer's IBM compatible VGA BIOS
DOS Program	**:**	Yes
DOS Version	**:**	3.0 or greater
Windows Program	**:**	No
Windows DOS Box	**:**	Yes (with limitations), Windows 3.1x and Windows 95 (see note below)
Windows DOS Full Screen	**:**	Yes, Windows 3.1x and Windows 95
OS/2 DOS Full Screen	**:**	Yes

Note

1. Registers read while in a Windows DOS box may not contain the expected value. This is because the VDD has virtualized all the VGA ports, in other words in a Windows DOS box the text mode is emulated with graphics.
2. READCHIP uses "stdout" calls and may be redirected to a file or piped to a DOS filter such as MORE.COM.
3. The "/f" option should only run from a DOS full screen. Running READCHIP with the "/f" option from a Windowed DOS prompt or an OS/2 DOS prompt can have unpredictable results.

Installation

Copy the file **readchip.exe** to a directory that is in the DOS path on your hard drive.

Usage

From the DOS prompt, type the following:

```
readchip [/d] [/f]
```

Where:	`readchip`	without any argument reads the Lower BIOS Data Area, CRT Controller, Graphics Controller, Attribute Controller, Sequencer, Auxiliary Registers and other General Registers.
	`/d`	reads the DAC Registers (color look up table).
	`/f`	reads the VGA Registers and attempts to calculate the Frame Rate.

To generate a report, simply type

```
readchip /d > report.txt
```

and the information which READCHIP obtains from the video controller will be stored in the file report.txt.

Comments

- If READCHIP encounters a non-Seiko Epson VGA chip, it will do a standard IBM VGA chip read and will only report values in registers associated with an IBM VGA.
- READCHIP will try and interpret BIOS settings such as panel type. If the BIOS is configured using 81xxCFG the settings may actually be different than default and these interpretations may be incorrect.

Program Messages

`ERROR: VGA required.`

READCHIP can only read IBM compatible VGA controllers, if any other video controller is present READCHIP will not run.

S-MOS
SYSTEMS
A Seiko Epson Affiliate

REVERSE.COM Display Utility

Drawing Office No. X00Z-B-005-02

Copyright © 1993, 1997 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

REVERSE.COM DISPLAY UTILITY

REVERSE.COM provides a flexible method for the user to change the screen polarity between normal and reverse video (i.e., white on black becomes black on white). REVERSE is not a Terminate and Stay Resident (TSR) program. It can be run multiple times with no side effects.

Note
REVERSE.COM makes BIOS calls which modify the LCD registers within the video controller which only effects the LCD outputs. If run on a CRT the change will not occur until a switch back to the LCD display.

Program Requirements

Video Controller	**:**	Any SPC81xx or SPC80xx
Display Type	**:**	LCD
BIOS	**:**	Seiko Epson Video BIOS
DOS Program	**:**	Yes
DOS Version	**:**	3.0 or greater
Windows Program	**:**	No
Windows DOS Box	**:**	Yes (with limitations), Windows 3.1x and Windows 95
Windows DOS Full Screen	**:**	Yes, Windows 3.1x and Windows 95
OS/2 DOS Full Screen	**:**	Yes

Installation

Copy the program file **reverse.com** to a directory that is in the DOS path on your hard drive.

Usage

REVERSE.COM is run from the DOS command line as follows:

```
reverse on|off|text|graphics
```

Where:

on	reverses the screen in all modes
off	restores the screen to its default condition
text	reverses the screen for text modes only
graphics	reverses the screen for graphics modes only

An incorrect command line parameter will cause the display of a usage screen.

Comments

- This utility requires the Seiko Epson video BIOS.
- In a Windows DOS box, the DOS box text mode is emulated with graphics. Text modes are not actually text modes at all, but are graphics modes. Running REVERSE in this case behaves as would be expected when running in graphics mode. Switching to a full screen DOS box using <ALT>-<Enter> switches the emulated text mode to an actual text mode and the program will then behave normally (text, reverse, on and off will apply).

Program Messages

`ERROR: Requires Seiko Epson Video BIOS Extensions.`

This program works with the Seiko Epson LCD Extensions only.

S-MOS
SYSTEMS
A Seiko Epson Affiliate

SESHELL.EXE Shell Utility

Drawing Office No. X00Z-B-009-03

Copyright © 1992, 1997 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

SESHELL.EXE SHELL UTILITY

SESHELL provides a menu interface to demonstrate the operation and command line interfaces for SPC81xx DOS utilities. Not all programs shipped with the SPC81xx will run under SESHELL, but the majority will.

Program Requirements

Video Controller	:	Any SPC81xx
Display Type	:	LCD or CRT
BIOS	:	Any manufacturer's VGA BIOS
DOS Program	:	Yes
DOS Version	:	3.0 or greater
Windows Program	:	No
Windows DOS Box	:	Yes, Windows 3.1x and Windows 95
Windows DOS Full Screen	:	Yes, Windows 3.1x and Windows 95
OS/2 DOS Full Screen	:	Yes

Installation

Copy the files **seshell.exe** and **seshell.ini** to a directory that is in the DOS path on your hard drive.

Usage

To use start program simply type:

```
seshell
```

and press **<Enter>**.

The Demo Shell has three windows. The Programs Window displays a list of the available programs. The Commands Window displays each program with the appropriate parameters for various options. The Description Window provides instructions for each program. The SESHELL screen looks like the following figure:

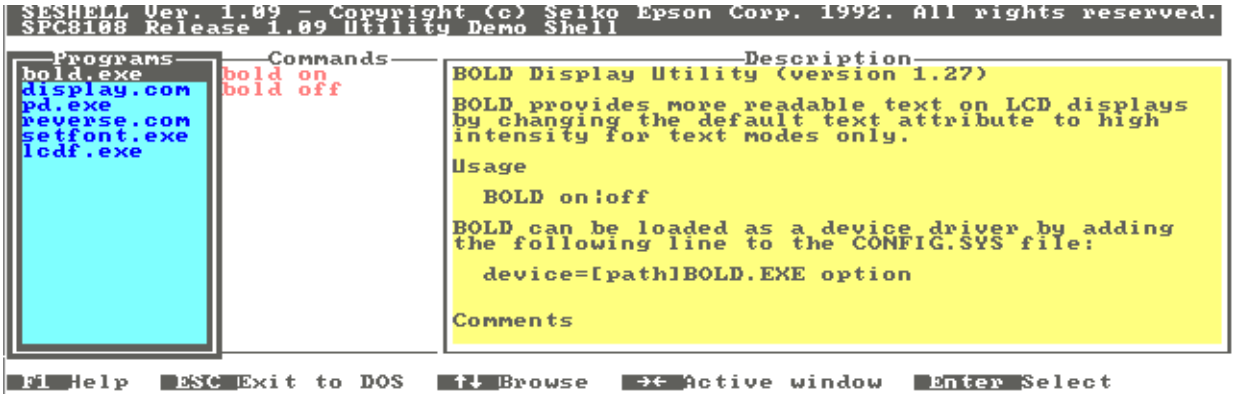


Figure 1 : SESHELL Screen

Running a Program

To run a program select the program in the Programs Window using Up-Arrow or Down-Arrow keys. Activate the Command Window using the Right-Arrow key and select the desired option using the Up- or Down-Arrow keys. Press ENTER to execute the program.

Optionally, after executing the utility, SESHELL may place the user in a DOS shell. To return to SESHELL, at the DOS command prompt enter:

```
exit
```

Viewing the Help File

To view program documentation, activate the Description Window using the Right-Arrow key. Use the Up- and Down-Arrow keys to scroll through the document.

Exiting SESHELL

To exit SESHELL, press **<ESC>**.

The SESHELL.INI File

The SESHELL program reads a configuration file called SESHELL.INI. This file contains information on the programs available through the shell, their parameters and their help files. SESHELL.INI must reside in the same directory as SESHELL.EXE.

Comments

- Any program that loads as a TSR will uninstall upon exiting SESHELL. SESHELL is designed to demonstrate programs and their command line parameters, not as a program launcher.
- Users should not attempt to use the uninstall features of a program that was loaded TSR prior to running SESHELL. This may cause unpredictable results.
- SESHELL will run under the DOS session of OS/2, but some programs may not install as a TSR.

OEM Note

SESHELL can be used by an OEM for end-users if desired. Scripting information can be made available.

Program Messages

`SESHELL already loaded.`

SESHELL cannot be run from within SESHELL.

`Failed to open seshell.ini.`

The SESHELL.INI file is missing.

`Failed to run PROGRAM NAME.`

SESHELL cannot find or cannot load the requested program. Strike any key to return to SESHELL. Make sure the program is found in the path and retry.

THIS PAGE LEFT BLANK

S-MOS
SYSTEMS
A Seiko Epson Affiliate

SETFONT.EXE Display Utility

Drawing Office No. X00Z-B-003-02

Copyright © 1992, 1997 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

SETFONT.EXE DISPLAY UTILITY

SETFONT is a display enhancement utility which can change the size and appearance of on-screen fonts. SETFONT's primary function is to load alternative font sets from the default font sets within the BIOS.

Program Requirements

Video Controller	**:**	SPC8104, SPC8106, SPC8108, SPC8110
Display Type	**:**	640x480 LCD
BIOS	**:**	Seiko Epson VGA BIOS
DOS Program	**:**	Yes
DOS Version	**:**	3.0 or greater
Windows Program	**:**	No
Windows DOS Box	**:**	Windows 95 only
Windows DOS Full Screen	**:**	Windows 95 only
OS/2 DOS Full Screen	**:**	2.0 or greater

Installation

Copy the file **setfont.exe** to a directory that is in the DOS path on your hard drive.

Usage

SETFONT can be run three ways, as a DOS command line program, a DOS device driver, or a menu interface. SETFONT is a Terminate and Stay Resident (TSR) program.

DOS Command Line

SETFONT can be invoked from the DOS command line as follows.

```
setfont [-iFontHeight] [-sScanLines] [ChildProgram] [/?]
```

Where**:**

`FontHeight` can be 8, 14, 16, or 19.

`ScanLines` can be 400 or 480.

`ChildProgram` can be used to load a program in conjunction with a font change. When the child program is exited any font changes stipulated with the loading of the program will be undone. This feature gives the user an easy way to make temporary font changes for use with specific programs.

`/?` produces the usage message shown above.

Example: the DOS command line **`setfont.exe -i8 -s480 x.exe`** will run the program x.exe in 80 characters x 60 rows display mode. On exiting x.exe, the screen will return to the previous setting.

DOS Device Driver

SETFONT can be loaded as a device driver. Place a line with the following format in the CONFIG.SYS file.

```
device=[path]setfont.exe [-iFontHeight] [-sScanLines]
```

Example: the line **device=c:\bin\setfont.exe -i8 -s480** in your config.sys file will set the display to 80 x 60 display mode. DOS will now operate with 60 visible lines on the screen verses the original 25 lines.

Font changes stipulated in the CONFIG.SYS take effect on boot-up and remain in effect until altered by running SETFONT from the DOS prompt.

Menu Interface

Running SETFONT without parameters produces a menu interface (see below). The menu displays the current state and a font preview window displays the currently selected font. To select an option from a menu, move the highlight using the UP- or DOWN-arrow keys. With the desired options highlighted press ENTER to make the changes and exit SETFONT.

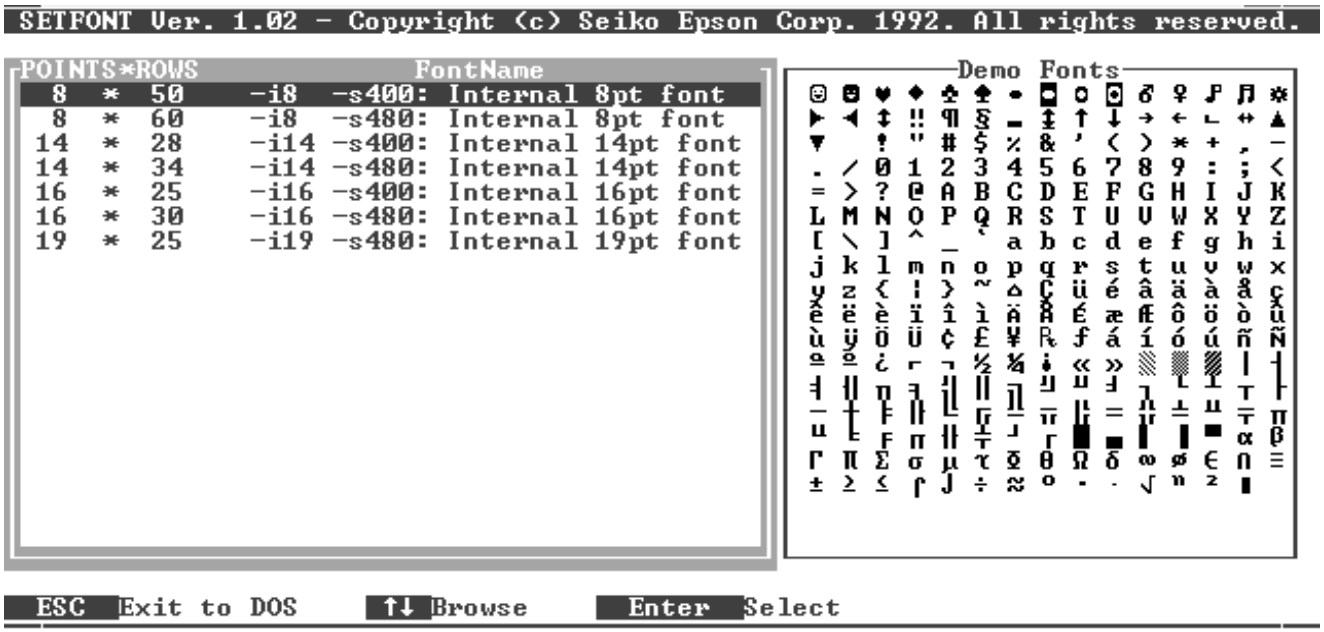


Figure 1 : SETFONT Menu Interface

Comments

- When SETFONT is run a portion of the program remains resident in memory and occupies approximately 8K bytes of memory.
- Some application programs assume a static font height and loading a different sized font may produce unexpected results in the display.
- The internal 19-point display font is actually a stretched 16-point font on an LCD display. Selecting the 19-point font selects Text Vertical Expansion Enable.

Program Messages

ERROR: VGA required.

A VGA display must be present to run SETFONT.

ERROR: VGA must be the active display.

The active display must be the VGA to run SETFONT. If the active display is currently a monochrome adapter, run the DOS command **mode co80** to switch to the VGA device, and re-run SETFONT.

ERROR: As a device driver SETFONT cannot run a child program.

To load a child program run SETFONT from the DOS prompt.

ERROR: SETFONT not installed.

SETFONT failed to load as a device driver. For some reason DOS has failed to load the device driver. Check the CONFIG.SYS file for errors.

ERROR: Not enough memory to load SETFONT.

The amount of free system memory below 640 Kilobytes is not enough to run SETFONT. SETFONT requires 8 Kilobytes of system memory. Free up more system memory and run SETFONT again.

ERROR: Cannot execute command line.

SETFONT could not load the specified child program. This may be due to one of many factors. Ensure that the command line is correct and that the child program can be loaded in a command shell.

ERROR: Cannot operate in a command shell.

SETFONT cannot install as a TSR when it is run within another program or within an environment like Windows.

ERROR: SETFONT is already installed as a device driver and cannot be installed again.

SETFONT cannot be installed more than once in the CONFIG.SYS file. Remove the redundant line from CONFIG.SYS.

ERROR: SPC8109 /SPC8108 /SPC8106 /SPC8110 /SPC8104 required.

This version of SETFONT requires an SPC8104, SPC8106, SPC8108, SPC8109 or SPC8110 VGA Controller.

ERROR: Seiko Epson Video BIOS Extension required.

SETFONT requires the Seiko Epson Video BIOS and its Extensions. If your system uses another manufacturer's BIOS this program will not run.

THIS PAGE LEFT BLANK

S-MOS
SYSTEMS
A Seiko Epson Affiliate

SHOWMODE.EXE Demonstration Program

Drawing Office No. X00Z-B-010-02

Copyright © 1995, 1997 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

SHOWMODE.EXE DEMONSTRATION PROGRAM

SHOWMODE displays all available video modes for Seiko Epson VGA/SVGA controllers with Seiko Epson Video BIOS Extensions. It also displays standard VGA modes for other makes of IBM compatible VGA controllers.

Program Requirements

Video Controller	**:**	Any VGA
Display Type	**:**	LCD or CRT
BIOS	**:**	Any manufacturer's VGA BIOS
DOS Program	**:**	Yes
DOS Version	**:**	3.0 or greater
Windows Program	**:**	No
Windows DOS Box	**:**	Windows 95 only
Windows DOS Full Screen	**:**	Yes, Windows 3.1x and Windows 95
OS/2 DOS Full Screen	**:**	Yes

Installation

Copy the file **showmode.exe** to a directory that is in the DOS path on your hard drive.

Usage

To display all available video modes (including both VGA and SVGA modes) in order, enter the following command line from the DOS prompt:

```
showmode [ModeNumber] [/f] [/1] /s] [/?]
```

Where:	`ModeNumber`	displays a specific video mode where the ModeNumber is the hex number of the desired video, as in showmode 101.
	`/f`	displays available video modes and attempts to calculate their frame rates.
	`/l`	lists all available video modes.
	`/s`	displays available video modes sequentially starting from the specified mode number.
	`/?`	displays the SHOWMODE usage message.

Comments

- SHOWMODE displays the standard IBM VGA modes assuming standard IBM VGA numbers (i.e., mode 3 is 80x25 characters). This program may have difficulties displaying some customized video modes if other applications have changed the display from the IBM default settings.
- If a non-Seiko Epson BIOS or non-Seiko Epson chip is present, SHOWMODE will restrict its display options to IBM standard modes 0 through 13 hex.
- The "/f" option should run only from a DOS full screen. Avoid running SHOWMODE with the "/f" option from a Windows DOS prompt or OS/2 DOS prompt.

Program Messages

`ERROR: cannot allocate memory.`

There is not enough memory to run this program. Free up more system memory below 640 Kilobytes and re-run SHOWMODE.

`ERROR: VESA function #n not supported.`

The VESA function #n is not supported. The Video Electronics Standards Association (VESA) BIOS extensions are required for extended mode support.

`ERROR: VESA function #n not successful.`

An error occurred while setting VESA function #n. This is an unknown error condition which could mean that in the current hardware setup the function failed. Change the hardware or BIOS configuration and try again.

`Error setting mode xxh. This mode is not supported in hardware.`

VESA BIOS supports mode xxh, however, the hardware does not support this mode. This error condition can occur when a high resolution mode is not supported by the LCD panel. i.e., mode 102h is an 800x600 high resolution mode, CRT monitors can handle it, but LCD panels (640x480) do not handle this mode.

`Error reading port 0x3DA.`

This error is due possibly to Windows or OS/2 Virtualization.

S-MOS
SYSTEMS
A Seiko Epson Affiliate

SM.EXE Display Utility

Drawing Office No. X00Z-B-002-02

Copyright © 1996, 1997 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

SM.EXE DISPLAY UTILITY

SM.EXE is a utility program designed to set all available graphics modes on a graphics controller. SM.EXE will set all standard IBM compatible modes as well as VESA modes and OEM modes if they are listed in the VESA mode list. SM.EXE is not a TSR program.

Program Requirements

Video Controller	**:**	Any VGA
Display Type	**:**	LCD or CRT
BIOS	**:**	Any manufacturer's VGA BIOS
DOS Program	**:**	Yes
DOS Version	**:**	3.0 or greater
Windows Program	**:**	No
Windows DOS Box	**:**	Yes
Windows DOS Full Screen	**:**	Yes, Windows 3.1 and Windows 95
OS/2 DOS Full Screen	**:**	Yes

Installation

Copy the file **sm.exe** to a directory that is in the DOS path on your hard drive.

Usage

From DOS prompt, type the following:

`sm [ModeNumber] [/l] [/q] [/200] [/350] [/400] [/?]`

Where:	`ModeNumber`	sets a specific video mode where the ModeNumber is the hex number of the desired video.
	`/l`	lists all available modes
	`/q`	runs in quiet mode (no output to display)
	`/200`	sets 200 scan lines (text mode only)
	`/350`	sets 350 scan lines (text mode only)
	`/400`	sets 400 scan lines (text mode only) (default)
	`/?`	shows this help display

Examples:

`sm /l`	lists all available display modes.
`sm 3 /200`	sets mode 3 in 200 scan line mode (8x8 font).
`sm 101`	sets a VESA mode 101 if available.

Comment

- If a desired display mode does not have TTY function support, you will see a blank or unitialized screen after the mode is set.

Program Messages

ERROR: Can only change scan lines for modes 0, 1, 2, 3, 7.

/200, /350 or /400 was used on the command line in combination with a graphics mode. These options can only be used for text modes.

ERROR: Invalid IBM Mode.

An attempt to set mode 8, 9, A, B or C was made. These are invalid IBM modes. Choose a different mode.

ERROR: Could not set mode.

An attempt to set a mode that is not contained in the list of available modes was made.

ERROR: Cannot select multiple scanlines.

It is not possible to select more than one scan line option on the command line. Ensure that only one of /200, /350 or /400 is on the command line.

ERROR: Invalid Number of Scan Lines.

An attempt to set an invalid number of scan lines was made. The only supported scan line options are: /200, /350 or /400.

ERROR: VESA function %d not supported.

The BIOS VESA implementation does not support the VESA function call.

ERROR: VESA function %d not successful.

The BIOS VESA implementation has failed to complete the VESA function call for some unknown reason.

ERROR: Cannot allocate memory.

You do not have enough memory to execute the program. Please free up some conventional RAM and re-run the program.

ERROR: /L cannot be used with other options.

The /L option is a query function and cannot be used with any other options which are set functions.

ERROR: Invalid command line argument.

An invalid command line argument was encountered.

S-MOS
SYSTEMS
A Seiko Epson Affiliate

SPRITEST.EXE Demonstration Program

Drawing Office No. X00Z-B-012-02

Copyright © 1993, 1997 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

SPRITEST.EXE DEMONSTRATION PROGRAM

SPRITEST demonstrates the sprite capabilities of the video chip by moving a demonstration sprite around the screen. It loads as a Terminate and Stay Resident (TSR) program and will run in conjunction with most programs.

Program Requirements

Video Controller	**:**	SPC8110, SPC8106, SPC8108
Display Type	**:**	LCD or CRT
BIOS	**:**	Seiko Epson VGA BIOS
DOS Program	**:**	Yes
DOS Version	**:**	2.0 or greater
Windows Program	**:**	No
Windows DOS Box	**:**	No
Windows DOS Full Screen	**:**	No
OS/2	**:**	No

Installation

Copy the file **spritest.exe** to a directory that is in the DOS path on your hard drive.

Usage

SPRITEST is invoked with the following command structure.

```
spritest [/An] [/3] [/9] [/Sxy][/V] [/H] [/Q] [/U] [/W] [/?] [ImageFile]
```

Example: **spritest /A3 animate**
or
spritest

ImageFile

The ImageFile parameter specifies a Windows format .BMP file (64 x 64) which will be displayed as the sprite image. SPRITEST will also display a series of images to create an animation effect. The file names must conform to the following convention:

ImageFil.000
ImageFil.001
. . .
ImageFil.nnn

Note
You need only to supply the last filename in the sequence. SPRITEST will automatically load files .001 to .nnn.

Switches

All switches must be preceded by either "-" or "/".

Image Control Switches

/An Animate n frames. Files ImageFil.001 to ImageFil.n are displayed in sequence to create an animation effect.

/3 The sprite is displayed in three color mode with the fourth color transparent. This is just one of the color schemes available to the sprites. Without this option, SPRITEST would be displayed in 4-color mode.

/Sxy /Sxy can be specified as /Sx, /Sy, or /Sxy to stretch the sprite in the x, y, or both axes.

/V Vertical expansion disable. On LCD displays where vertical expansion is enabled, sprite movement may look jerky and uneven. Disabling vertical expansion corrects this problem.

Miscellaneous Switches

/? Display the help screen.

/Q Quiet mode causes SPRITEST to load without displaying either the copyright notice or errors. This option is useful for demonstrations where multiple images are loaded.

/U Uninstall SPRITEST. Attempts to remove the TSR portion of Spritest from memory.

/H Specify the height of the display (in pixel). i.e. /H640

/W Specify the width of the display (in pixel).

Program Messages

`Installation successful.`

SPRITEST has installed as a TSR and successfully loaded the image file.

`ERROR: cannot install - Spritest is already installed.`

Only one copy of SPRITEST can be resident in memory at any one time. SPRITEST can be invoked with additional command parameters to adjust the display of the sprite image.

`ERROR: cannot install - DOS 2.0 or later required.`

Some of the DOS calls used by SPRITEST require MS-DOS version 2.0 or later.

`ERROR: cannot install - no TSR ID numbers available.`

To remedy this problem remove one or more TSR programs from the start-up sequence and reboot.

`ERROR: cannot operate in an OS/2 session.`

This version of SPRITEST will not execute in an OS/2 DOS session.

`ERROR: SPRITEST cannot operate in a command shell.`

Solution: Run SPRITEST from the primary command shell and load other programs after it.

`ERROR: unknown video controller. Spritest not loaded.`

SPRITEST checks for the presence of supported Seiko Epson video controllers before loading.

`Uninstall successful.`

All memory used by the SPRITEST is returned to the DOS memory pool and interrupt vectors used by SPRITEST are restored.

`ERROR: cannot uninstall - Spritest is not installed.`

SPRITEST was not loaded when the uninstall command was given.

`ERROR: cannot uninstall - interrupt vectors have changed.`

Another TSR has loaded after SPRITEST and altered the interrupt vectors. Spritest cannot restore the interrupt vector and therefore will not uninstall.

`ERROR: cannot uninstall - cannot free memory block.`

Successful uninstallation also depends on the proper restoration of memory to the DOS memory free pool. This message indicates that the memory management block associated with SPRITEST has been damaged in some way. The only way to remove SPRITEST now is to reboot the computer.

`ERROR: cannot uninstall - cannot free memory control block`

Same meaning as above.

`ERROR: could not open image file.`

The name given (or interpreted) as the filename parameter could not be found. Specify a different filename or give the complete path to the image file.

`ERROR: while reading image file`

DOS reported an error while reading the file. Most likely the image data file has been corrupted.

`ERROR: file does not appear to be a .BMP file.`

The check for the BMP file signature failed. Even if the rest of the image file format is correct the image will not be loaded.

`ERROR: BMP image is incorrect. Images must be 64x64 and 16 or less colors.`

If the image stored in the BMP file is greater than 64 pixels vertically or horizontally or contains more than sixteen colours the image will not be loaded.

`Defaulting to the internal image.`

This message is seen in conjunction with several image errors messages and indicates that SPRITEST will display the internal image. The internal image is included to ensure that an image file will always be available for display.

`ERROR: unknown error encountered.`

This error message should not occur. It is actually part of the general message display routine and no tested error should point to this message.

`ERROR: SPRITEST cannot operate under Windows 95 shell.`

Re-start your system in DOS mode and re-run SPRITEST.

S-MOS
SYSTEMS
A Seiko Epson Affiliate

VESADDC.EXE Display Utility

Drawing Office No. X00-UI-005-02

Copyright © 1996, 1997 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

VESADDC.EXE DISPLAY UTILITY

VESADDC displays DDC information: EDID or VDIF blocks. This is a DOS program that calls the Video Electronic Standards Association (VESA) BIOS extensions for Display Data Channel (VBE/DDC) management. For more information on VESA DDC please contact VESA.

Program Requirements

Video Controller	:	Any VESA VBE/DDC compliant controller
Display Type	:	DDC Monitor
BIOS	:	Any VESA VBE/DDC compliant BIOS
DOS Program	:	Yes
DOS Version	:	2.0 or greater
Windows Program	:	No
Windows DOS Box	:	Yes[a]
Windows DOS Full Screen	:	Yes[b]
OS/2 DOS Full Screen	:	Yes[c]

[a] Virtual Display Driver must be written in such a way to accommodate this.
[b] Same as footnote a.
[c] Same as footnote a.

Installation

Copy the file **vesaddc.exe** to a directory that is in the DOS path on your hard drive.

Usage

VESADDC.EXE is run from the DOS command line as follows:

```
VESADDC [EDID | VDIF n | /V]
```

Where:

EDID	Shows EDID block.
VDIF n	Shows VDIF block n (decimal). Most monitors only support VDIF blocks 0 and 1.
/v	Runs in verbose mode. Has no effect with VDIF. When used in conjunction with EDID, displays interpreted EDID blocks.

If executed with no parameters, VESADDC detects the presence of a VESA DDC capable BIOS and reports the DDC levels supported.

Comment

- The program supports EDID 1.0 & 1.1 formats. VDIF gets only dumped out in hex format.

Program Messages

ERROR: Checksum not zero.

The obtained EDID block is incorrect. This could be that the information is corrupted during the communication.

ERROR: Failed to obtain EDID block.

The monitor is not DDC2 compatible.

ERROR: Failed to obtain VDIF xx block.

Error happened while getting VDIF block. (xx is an integer).

ERROR: VBE/DDC not supported.

Either BIOS or monitor does not support VBE/DDC.

ERROR: VBE/DDC Function call failed.

This VESA call failed.

ERROR: Invalid EDID Header.

EDID block should have a header start with 00, ff, ff, ff, ff, ff, ff, 00.

ERROR: Invalid descriptor block. Flags not zero.

The obtained information does not comply with the VESA DDC Specification.

ERROR: Invalid descriptor block. Byte 13 not 0Ah.

The obtained information does not comply with the VESA DDC Specification.

ERROR: Undefined descriptor block. Tag = xx

The obtained information does not comply with the VESA DDC Specification. (xx is a hex number)

ERROR: byte 11 not 0Ah.

The obtained information does not comply with the VESA DDC Specification.

ERROR: byte xx not 20h.

The obtained information does not comply with the VESA DDC Specification. (xx is a hex number)

S-MOS
SYSTEMS
A Seiko Epson Affiliate

VESAPD.EXE Display Utility

Drawing Office No. X00-UI-003-02

Copyright © 1996, 1997 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

VESAPD.EXE DISPLAY UTILITY

VESAPD is an OEM utility to demonstrate the power saving capabilities of the graphics controller. This is a DOS program that calls the Video Electronic Standards Association (VESA) BIOS extensions for Power Management (VBE/PM) to set the power state. The power states mirror the display power management signalling (DPMS) standard as defined by VESA.For more information on VBE/PM please contact VESA.

Program Requirements

Video Controller	**:**	Any VESA VBE/PM compliant controller
Display Type	**:**	LCD or CRT
BIOS	**:**	Any VESA VBE/PM compliant BIOS
DOS Program	**:**	Yes
DOS Version	**:**	2.0 or greater
Windows Program	**:**	No
Windows DOS Box	**:**	Yes
Windows DOS Full Screen	**:**	Yes
OS/2	**:**	Yes

Installation

Copy the file **vesapd.exe** to a directory that is in the DOS path on your hard drive. The directory does not need to be in the DOS path; if it is not in the path then a full path name will have to be specified each time VESAPD is run.

Usage

VESAPD.EXE is run from the DOS command line as follows:

```
VESAPD [on | standby | suspend | off | reduced ]
```

Where:

on	is full power
standby	is VBE/PM state standby
suspend	is VBE/PM state suspend
off	is VBE/PM state off
reduced	is VBE/PM state reduced on

If executed with no parameters VESAPD detects the presence of a VBE/PM capable BIOS and reports the states available. If VBE/PM is not supported, VESAPD reports an error message and displays the usage screen.

Program Messages

`Power Down Mode "request" is set.`

The VBE/PM power state "request" was set correctly.

`ERROR: VBE/PM not supported.`

VESAPD did not locate a VBE/PM compatible BIOS.

`ERROR: cannot set power mode "request"!`

The VBE/PM power state "request" failed to set. This may be due to the BIOS not supporting this mode. Please use one of the power states reported back by the usage screen.

`ERROR: VBE/PM Function call failed.`

The BIOS reported back that VBE/PM was supported, but did not operate correctly due to an unknown reason.

S-MOS
SYSTEMS
A Seiko Epson Affiliate

VRTEXP.EXE Display Utility

Drawing Office No. X00Z-B-011-02

Copyright © 1995, 1997 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

VRTEXP.EXE DISPLAY UTILITY

The VRTEXP program is used to control Text and Graphics Vertical Expansion on an LCD panel. This program calls the Seiko Epson BIOS (SOLLEX) Extentions function which modifies LCD control bits within the Seiko Epson video chip. VRTEXP.EXE does not Terminate and Stay Resident (TSR) and can be run as many times as the user desires.

Program Requirements

Video Controller	**:**	SPC8104, SPC8106, SPC8108, SPC8110
Display Type	**:**	LCD
BIOS	**:**	Seiko Epson VGA BIOS
DOS Program	**:**	Yes
DOS Version	**:**	3.0 or greater
Windows Program	**:**	No
Windows DOS Box	**:**	Yes (with limitations), Windows 3.1x and Windows 95
Windows DOS Full Screen	**:**	Yes, Windows 3.1x and Windows 95
OS/2 DOS Full Screen	**:**	Yes

Installation

Copy the file **vrtexp.exe** to a directory that is in the DOS path on your hard drive.

Usage

VRTEXP.EXE is run from the command line as follows:

```
vrtexp /T | /G | /B | /N
```

Where:

/T	enables Text Vertical Expansion and disables Graphics Vertical Expansion.
/G	enables Graphics Vertical Expansion and disables Text Vertical Expansion.
/B	enables both Text & Graphics Vertical Expansion (BOTH)
/N	disables both Text & Graphics Vertical Expansion (NONE)

Comments

- This utility requires the Seiko Epson video BIOS.
- In a Windows DOS box, the DOS box text mode is emulated with graphics. Text modes are not actually text modes at all, but are graphics modes. Running VRTEXP in this case behaves as would be expected when running in graphics mode. Switching to a full screen DOS box using <ALT>-<Enter> switches the emulated text mode to an actual text mode and the program will then behave normally.

Program Messages

`ERROR: Seiko Epson BIOS Extensions required.`

This program works with the Seiko Epson LCD Extensions only.

`ERROR: SPC8104, 8106, 8108, 8109 or 8110 required.`

This version of VRTEXP requires an SPC8104, SPC8106, SPC8108, SPC8109, or SPC8110 VGA controller.

S-MOS
SYSTEMS
A Seiko Epson Affiliate

WGS.EXE Windows Gray-Scale Utility

Drawing Office No. X00-Z-B-008-01

Copyright © 1993, 1996 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product

THIS PAGE LEFT BLANK

WGS.EXE WINDOWS GRAY-SCALE UTILITY

Under some circumstances while using monochrome LCD panels, the translation of colors to shades of gray produces unattractive and unreadable low contrast results. WGS.EXE is a Windows application which allows the user to select between gray-scaling algorithms which may improve the display quality. WGS takes advantage of the graphics controller's special features which offer a selection of two different gray-scaling algorithms.

Program Requirements

Video Controller	:	SPC8104, SPC8106, SPC8107, SPC8108, SPC8110
Display Type	:	Monochrome LCD
BIOS	:	Seiko Epson VGA BIOS
DOS Program	:	No
DOS Version	:	N/A
Windows Program	:	Yes, Windows 3.1x and Windows 95
Windows DOS Box	:	N/A
Windows DOS Full Screen	:	N/A
OS/2	:	No

Installation

Copy **wgs.exe** to a sub-directory on your computer hard drive.

In Windows, use the "New" option from the Program Manager File Menu to create a new item and install an icon for WGS in a program group. Alternately, WGS can be run by selecting the "Run" option from the File Menu in either Program Manager or File Manager.

Usage

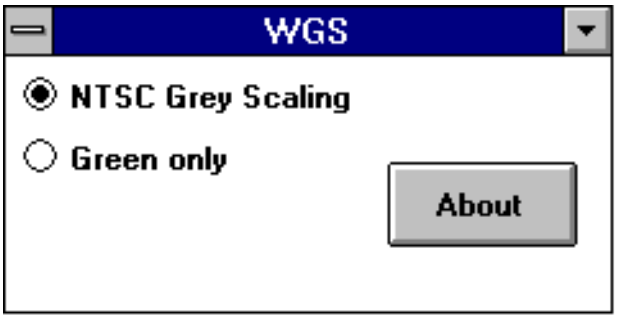


Figure 1 : The WGS Dialog Box

To select a gray-scaling algorithm, click the mouse pointer in the corresponding check box. Changes will take place immediately.

The program options are:

NTSC Gray Scaling

NTSC uses an approximation of the standard 59% Green, 30% Red, 11% Blue standard gray-scaling algorithm.

Green Only

Green Only ignores the Red and Blue components of a color and bases the gray-scale on the Green component only.

About

Clicking on the “About” button will display information about the program including the version number and the copyright notice. Click on the OK button to return to the main dialog box.

Comments

- The **last** selected gray-scale algorithm selected remains active in a DOS box and after exiting Windows.
- This program has no visible effect while running on a CRT, but will take effect when a switch to LCD occurs.
- changes take effect immediatly upon selecting a button.

Program Messages

The above error message appears when the program is run on a system which does not use the Seiko Epson Video Controller Extensions (SOLLEX).

S-MOS
SYSTEMS
A Seiko Epson Affiliate

SPC8110F0A Color VGA LCD/CRT Controller

Windows 3.1x Display Drivers

Drawing Office No. X07G-E-002-01

Copyright © 1995, 1996 S-MOS Syatems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

WINDOWS 3.1X DISPLAY DRIVERS

The Windows 3.1x Display drivers are designed to be used with the SPC8110F0A and take advantage of the hardware acceleration features found in the graphics controller. The driver comes in two color depths 16-color and 256-color. This driver works with all versions of Microsoft Windows 3.10 and 3.11, and will run in both the enhanced and standard modes of Windows.

Program Requirements

Video Controller	:	SPC8110F0A
Display Type	:	LCD or CRT
BIOS	:	Seiko Epson VGA BIOS
DOS Program	:	No
Dos Version	:	N/A
Windows Program	:	Yes, Windows 3.1x video driver
Windows DOS Box	:	N/A
Windows Full Screen	:	N/A
OS/2	:	N/A

Installation

If you have not already installed Windows 3.1x, install Windows with the default VGA driver selected. Proceed to the next step.

Run SETUP.EXE (found in the Windows directory) to install the SPC8110 Driver. Follow the instructions under “Installing a Device Driver Not Supplied with Microsoft Windows 3.1" in the Microsoft Windows User’s Guide. From the list of displays presented by SETUP, choose “Other display (requires disk from OEM)”. When prompted, insert the SPC8110 Utilities Diskette in the appropriate drive and follow the on-screen instructions.

Once the diskette has been inserted and recognized by Windows Setup, a menu will be displayed with 6 options. These are; 640x480, 800x600 and 1024x768 at 16 and 256 colors. Select the driver you wish to run and select OK. If you have a 512KB evaluation card, the installation program will allow you to select 1024x768 by 256 colors, but this driver will not work correctly. If you accidentally select 1024x768 at 256 colors on a 512KB board, just re-execute SETUP and select a different driver.

When prompted for the Windows 3.1x original diskettes, you have the option of inserting the original diskettes into the floppy drive or typing the path to your Windows system directory (typically c:\windows\system) at the prompt.

Operation

There are no operational differences between the SPC8110F0A Windows 3.1x driver and the standard VGA driver.

Comments

- To run the driver in Standard mode, you must use Windows 3.10 or Windows 3.11 only, not Windows for Work-groups; and you must have EMM386.EXE removed from your CONFIG.SYS.
- If the 256 color driver fails to initialize correctly, but the 16 color driver works, try changing the address of the linear frame buffer. This is located in WINSPEED.INI on the line **addr=0xcf00**. Early VL-BUS motherboards wrapped system memory, so the default linear frame buffer may overwrite a normal system RAM. Note that this only occurs on VL-BUS systems. If using a PCI system, the linear frame buffer location is guaranteed by the system.

S-MOS
SYSTEMS
A Seiko Epson Affiliate

SPC8110F0A Color VGA LCD/CRT Controller

Windows 95 / Windows NT 4.0 Display Driver

Drawing Office No. X00Z-E-001-03

Copyright © 1995, 1997 S-MOS Syatems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

WINDOWS 95 / WINDOWS NT 4.0 DISPLAY DRIVER

The Windows 95 / Windows NT 4.0 Display drivers are designed to be used with a Seiko Epson VGA LCD controller and take advantage of the hardware acceleration features found in the graphics controller. These drivers come in one color depth: 8 bits-per-pixel, 256-color.

Program Requirements

Video Controller	:	SPC8106F0C, SPC8110F0A
Display Type	:	LCD or CRT
BIOS	:	Seiko Epson VGA BIOS or Seiko Epson Card - 486D4 BIOS
DOS Program	:	No
Dos Version	:	N/A
Windows Program	:	Yes, Windows 95 / Windows NT 4.0 display driver
Windows DOS Box	:	N/A
Windows Full Screen	:	N/A
OS/2	:	N/A

Installation

If you have not already installed Windows 95 or Windows NT 4.0, install the operating system using the default VGA driver. If you have previously installed Windows, ensure that the current display driver is “VGA” prior to proceeding.

Note
The SDU81xxB0x series of PCI evaluation boards do not decode a BIOS EPROM. Windows NT requires that the INT 10h BIOS exists at segment C000h, therefore an external EPROM decoder card must be used in conjunction with these boards. For Windows 95 operation see SDU8110B0x PCI Boot Procedures, Drawing Office Number X07-AN-016-xx, or use an external EPROM decoder card.

1. Re-start the system.
2. Under the **Start** menu select **Settings**.
3. Within the **Settings** menu select **Control Panel**.
4. Within **Control Panel** select **Display** then **Settings**.
5. For Windows 95; within **Settings** select **Change Display Type**.
   For Windows NT 4.0; within **Settings** select **Display Type**.
6. Under the **Adapter Type** menu select **Change** and then **Have Disk**.
7. Follow the instructions provided.

Once the driver files have been copied and loaded by Windows, do the following:

1. Select the resolution and color depth, using the controls provided.
2. Restart you computer for the changes to take effect.

Operation

There are no operational differences between the Seiko Epson Windows 95 / Windows NT 4.0 driver and the standard VGA driver. The resolution can be changed using the Control Panel at any time.

Comments

- For the SPC8110F0A this is a 256 color driver with display resolutions of 640x480, 800x600 and 1024x768. If the controller has only 512K bytes of RAM available to it, the 1024x768 resolution is not supported.
- For the SPC8106F0C this is a 256 color driver with a display resolution of 640x480 only.

S-MOS
SYSTEMS
A Seiko Epson Affiliate

SPC8110F0A VGA LCD/CRT Controller

Windows NT 3.5x Display Driver

Drawing Office No. X07G-E-001-01

Copyright © 1993, 1996 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

Windows NT 3.5x Display Driver

The Windows NT Display driver is designed to be used with the SPC8110F0A and take advantage of the hardware acceleration features found in the graphics controller. This 256 color driver works with Microsoft Windows NT 3.5x only.

Requirements

Video Controller	**:**	SPC8110F0A
Display Type	**:**	LCD or CRT
BIOS	**:**	Seiko Epson VGA BIOS
DOS Program	**:**	No
DOS Version	**:**	3.0 or greater
Windows Program	:	Windows NT 3.5x only

Installation

To install the Windows NT 3.5x display driver, do the following:

1. If you have not already installed Windows NT, install Windows NT with the default VGA driver selected.
2. Start Windows NT in VGA mode.
3. From the *Main* group, select the *Control Panel*.
4. From the *Control Panel*, select *Display*.
5. In the *Display Settings* window, select **Change Display Type**.
6. In the *Change Display* window, select **Other**.
7. Insert the manufacturers disk in Drive A:.
8. Choose **OK**.

   In the *Change Display* window, three options are added in the *Select Driver* option list:

   SPC8110 640 x 480 256 Colors
   SPC8110 800 x 600 256 Colors
   SPC8110 1024 x 768 256 Colors
9. Highlight one of the three added options and select **Install**.
10. A dialogue box appears warning you that you are about to change the system configuration. Select **Yes**.
11. If the drivers are previously installed, another dialogue box appears asking you whether you want to replace these drivers or use the drivers that are already installed. If this dialogue appears, select **New**.
12. You are then prompted for the manufacturer's disk. Insert the disk in drive A:.
13. Select **Continue**.
14. A message will appear indicating either a successful or an unsuccessful copy. If the installation was unsuccessful, you may have a bad diskette.
15. Another dialogue appears prompting you to restart your system. Select **Restart Now**.

Note
Remove your floppy from drive A: before you reboot.

The system will shut down and reboot with the new driver installed.

Limitations

- If Windows NT does not boot correctly and displays only a blue screen, then one of the following may have occurred:
  1. The driver was run on a PCI evaluation board. This driver does not support PCI.
  2. The frame buffer location is hardcoded in the driver and is incompatible with the motherboard.

  If either of these situations occurs, you may need a custom driver developed for your system.
- If you have a 512KB evaluation card, the installation program will allow you to select 1024x768 by 256 colors, but this driver will not work correctly. If you accidentally select 1024x768 at 256 colors on a 512KB board, restart NT using VGA mode and re-install the driver with the new resolution.

Operation

There are no operational differences between the SPC8110F0A Windows NT driver and the standard VGA driver.

S-MOS
SYSTEMS
A Seiko Epson Affiliate

SPC8110F0A VGA LCD/CRT Controller

Autocad Display Driver

Drawing Office No. X07-DI-003-01

Copyright © 1993, 1996 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

TABLE OF CONTENTS

ABOUT THIS DOCUMENT 5
Requirements 5
TURBODLD CLASSIC: DRIVERS FOR AUTOCAD R12, R11 AND R10/386 6
Features 6
System Requirements 6
INTRODUCTION 7
About This Manual 7
AutoCAD Installations 7
TurboDLD Installation 7
Naming Conventions 7
Keyboard References 7
QUICK START 8
What does it do? 8
What is the Catch? 8
Thanks for the Memory 8
Installation 8
To Configure AutoCAD Release 12, Release 13 8
To Configure AutoCAD Release 11/386, Release 10/386 9
Driver Configuration 9
What If I Have Problems? 10
PANACEA'S TURBODLDCLASSIC 11
INSTALLING AND CONFIGURING TURBODLD CLASSIC 12
Installation 12
Driver Configuration 13
Select Graphics Board/Resolution Menu 14
Basic Configuration Menu 14
User Interface Menu 15
Verifying Your TurboDLDClassic Installation 17
Reconfiguring TurboDLDClassic 17
Configuring The AutoCAD R10/11 Operating Environment 18
Configuring TurboDLDClassic for Rendering 18
CHANGING COLORS 19
TURBODLD FEATURES 21
MEMORY USAGE AND LISTS 24
How much memory? 24

TIPS, TRICKS & TROUBLESHOOTING 25
Memory Management 25
Upgrading From Older Panacea DLD Drivers 25
Maneuvering Through AutoCAD 26
Using AutoCAD Commands 27
Third-Party Software 28
Troubleshooting TIGA Installations 28
Color Considerations 29
Corrupt TurboDLDClassic Data Files 29
Configuring TurboDLD for Your Graphics Board 29
Static BP in Panning Mode 29
Pull-Downs and Crosshairs Leave Trash on the Screen 29
RENDERING WITH TURBODLD CLASSIC 30
Autodesk 3D Studio 30
Configuring AVE Render 32
Autoshade 2.0 with Renderman 33
Support Information 34
TurboDLD Classic Commands 35
Changes From Previous Versions 36
Changes From TurboDLD v1.0 36
Changes From MACH-12 36

ABOUT THIS DOCUMENT

Seiko Epson has licensed from Panacea Inc. the TurboDLD*Classic* AutoCAD® display driver. This document contains the original TurboDLD*Classic* **User's Guide** as supplied by Panacea Inc.

Requirements

Video Controller	**:**	SPC8110F0A
Display Type	**:**	LCD or CRT
BIOS	**:**	Seiko Epson VGA BIOS
DOS Program	**:**	Yes
DOS Version	**:**	3.0 or greater
Windows Program	**:**	No

TURBODLD CLASSIC: DRIVERS FOR AUTOCAD R12, R11 AND R10/386

Version 2.1 for AutoCAD® Release 13, 12, 11/386, and 10/386, AutoShade 2 with RenderMan, and 3D Studio, part of the CAD Accelerators™ family from Panacea Inc.

Information in this document is subject to change without notice and does not represent a commitment on the part of Panacea Inc. (Late additions to this document may be found in the READ.ME file on the distribution disk.)

Features

- Includes a bird's eye view and a graphics-mode text window.
- Accelerates redraws, pans, and zooms.
- Easy to use - no new commands or special menus to learn.
- Protected-mode ADI 4.2 driver - completely compatible with AutoCAD Release 13, 12, Release 11/386, Release 10/386, 3D Studio and AutoShade 2 with RenderMan.
- No memory conflicts. Works with AutoCAD's built-in Virtual Memory Manager.
- Includes CustomColorsTM, which lets you interactively customize your logical and physical colors from within AutoCAD.
- Completely compatible with all AutoCAD ADI 4.2 compatible third party software.
- Will support higher resolutions on all popular graphics boards.
- Supports all AutoCAD Release 12 features, including rendering to viewports and 31-Bit regen space.

System Requirements

- A '386, '486 or Pentium based PC which supports AutoCAD Release 13, 12, 11/386, 10/386, AutoShade 2 with RenderMan, or 3D Studio v1, v2 or v3.
- A compatible graphics board. TurboDLD*Classic* will work with all popular graphics boards - VGA, SuperVGA, 8514/A, TIGA, XGA, S3, Weitek P9000 and Compaq QVision.
- Additional extended memory is recommended for optimal performance.

INTRODUCTION

About This Manual

Thank you for purchasing TurboDLD*Classic* v2.0 -- and also for reading this manual. We are very excited about this release of TurboDLD, mainly because of you, the AutoCAD user. Many of the features and enhancements of this version are the result of your comments and requests. With that input, we are able to offer you a complete productivity solution at an affordable price.

This introduction provides a quick overview of the terms and conventions used throughout the manual. Please take a few minutes to familiarize yourself with it.

AutoCAD Installations

We know that every AutoCAD installation is different. However each one requires the configuration of the same basic parameters in order to function. The examples we use in this manual assume that you have installed AutoCAD on drive C using the installation defaults. This means that your main AutoCAD directory is C:\ACAD and all other directories (for example, SUPPORT, FONTS, DRV, and so on) fall under it.

We also assume that you are using the AutoCAD created ACADR12.BAT file to start AutoCAD Release 12. For AutoCAD Releases 10/386 and 11, it is not as critical to use a batch file to start the program, however it is much easier. If you have not configured AutoCAD using defaults, don't worry, you can still easily install TurboDLD*Classic*. All you have to do is substitute your specific sub-directory names with those in our examples. If you are using a menuing system or other AutoCAD start-up file, use that when we reference ACADR12.BAT.

TurboDLD Installation

Our references to the location of TurboDLD files assume that you have chosen to install TurboDLD*Classic* into the C:\ACAD\DRV sub-directory for AutoCAD Release 12, or in a sub-directory named C:\TURBODLD for AutoCAD Release 10 or 11.

Naming Conventions

To save some space (not to mention a tree or two) and to make the manual a little easier to read, we have abbreviated some name references. If you see the terms below used later in these pages, here's what they mean.

BP = Big Picture, TurboDLD's bird's eye view
R13= AutoCAD Release 13
R12= AutoCAD Release 12
R11= AutoCAD Release 11
R10= AutoCAD Release 10/386

Keyboard References

CTRL-C = Hold the CTRL key down and then press the “C” key.

ESC = The ESCape key in the upper left-hand corner of your keyboard.

ARROW key references are to the cursor control keys on your keyboard, the four keys in-between the main keys and the numeric keypad.

QUICK START

These pages sum up the basics of installing and using TurboDLD*Classic*. Please read the rest of the manual for complete details.

What does it do?

TurboDLD*Classic* 's main function is to speed up AutoCAD pans, zooms and redraws. The driver accomplishes this by creating and maintaining a Display List - a fast-displaying object list of the current drawing - dramatically increasing performance on pans and zooms. In addition to offering display list speed enhancement, TurboDLD*Classic* provides add-on productivity features - a bird's-eye view and a graphics mode text window - to further speed up the use of AutoCAD.

What is the Catch?

The only resource used by TurboDLD*Classic* is memory. The driver actively uses about 300Kbytes of RAM from extended memory for its operation. This memory is drawn from AutoCAD's memory pool and therefore does not affect normal DOS operation. The driver is loaded by AutoCAD at AutoCAD load time and is unloaded when AutoCAD is exited.

Additionally, the Display List size can vary from one tenth to three times the size of the current drawing file. We recommend that you use 8Mb of RAM with AutoCAD Release 10/386 or 11, 12Mb for AutoCAD Release 12, and 16Mb for AutoCAD Release 13

Thanks for the Memory

You may wish to purchase and install additional RAM before installing TurboDLD*Classic*, since it shares memory with AutoCAD. If AutoCAD is using a lot of memory, TurboDLD*Classic* may not have enough. If there is significant hard disk activity while you are using TurboDLD*Classic*, it may be an indication that you should add more memory to your system.

Installation

Run the TurboDLD*Classic* installation program by typing:

```
A: (or B: if you are using drive B)
INSTALL
```

TurboDLD*Classic*'s install program will prompt for AutoCAD version and sub-directory information. After all information has been provided, TurboDLD*Classic* files will be copied from the distribution floppy to the hard drive.

To Configure AutoCAD Release 12, Release 13

Since the start-up is a little bit different for AutoCAD Release 12 and 13 than for older versions, please follow the instructions for the version of AutoCAD for which you are installing TurboDLD*Classic*.

Begin AutoCAD with the reconfigure switch by typing

```
ACADR12 -R
(or ACADR13 -R)
```

Choose option 3, **Configure Video Display** from the AutoCAD configuration menu.

Type **Y** at the "**Do you want to select...**" message to display the available video options for AutoCAD.

Select "**TurboDLD** *Classic* **by Panacea Inc.**" from the list of display options.

Alternately, to get to the AutoCAD Configuration menu described above, start AutoCAD as you normally would, and then type CONFIG at the Command prompt.

Continue installation at the Driver Configuration section below.

To Configure AutoCAD Release 11/386, Release 10/386

Run the **FASTACAD** batch file created in your TurboDLD*Classic* sub- directory and then start AutoCAD.

Reconfigure AutoCAD to use TurboDLD*Classic* by selecting option 5, **Configure AutoCAD** from the AutoCAD main menu. From the next menu, select option 3, **Configure Video Display**. Choose Item 1, **P386 ADI 4.0/4.1** (R11) or **ADI P386** (R10), as your display device. (For more information, see your AutoCAD "Installation and Performance Guide.")

Driver Configuration

After selecting the proper display device, the TurboDLD Driver Configuration menu will appear.

Set-up of TurboDLD*Classic* requires configuration of its various operating parameters. These parameters have been logically grouped into menus based on their interaction with AutoCAD. A quick trip through each menu will complete the configuration process. Context sensitive help can be obtained by typing "?" at any menu option.

The first menu, **Select Graphics Board/Resolution**, configures TurboDLD*Classic* for the graphics board, and display and rendering resolutions to be used.

- Select Graphics Board - Selects graphics platform to be used
- Select Display Resolution - Selects AutoCAD, 3D Studio and AutoShade main display screen resolutions
- Select Rendering Resolution - Selects AVE Render, 3D Studio and AutoShade rendering resolutions

Note
Rendering menu options only appear when configuring the rendering screens of AVE Render, 3D Studio and AutoShade2 with RenderMan. You will not see both menu options at the same time.

Basic Configuration options set AutoCAD screen characteristics, number of text lines in the command-line area, font size and screen configuration mode.

- AutoCAD Text Lines - Selects number of lines in AutoCAD command prompt area
- Screen Configuration - Provides a choice of single screen, dual screen or single screen with a graphics mode text window.
- Font Selection - Allows selection of TurboDLD*Classic* supplied fonts or Autodesk supplied .BDF fonts menus and pull-downs.

User Interface options include the parameters for manoeuvring within AutoCAD.

- Double Click - Sets mouse/digitizer double click detect time.
- BP Button - Select which mouse or tablet button to use for bird's eye pop-up.
- BP Highlight Mode - Controls how the Big Picture is displayed.
- BP Refresh - Controls the refresh mode of the Big Picture.

Expert Configuration options set the function of TurboDLD itself. This menu allows for customization of the driver for speed or to adjust for memory constraints.

- Display List - Enables or disables the display list feature of TurboDLD*Classic*.
- AutoCAD Logical Drawing Space - Sets AutoCAD's logical space to 15-bits or 31-bits (R12 only).
- Internal Command Echo - Enables or disables the echo of TurboDLD*Classic* internal commands at the AutoCAD Command line when such commands are executed.
- Big Picture Zoom Mode - Sets Big Picture zoom area display options
- Display List in PaperSpace - Disable or enables display list while in PaperSpace

Once all options have been set, scroll down to the **Save and Exit** option and then press the **ENTER** key to continue.

Next, configure the AutoCAD screen display characteristics and then type **Y** to accept the changes.

Exit to the AutoCAD drawing editor to begin using TurboDLD*Classic*.

The last step in setting-up TurboDLD*Classic* is to configure the colors for AutoCAD. From the AutoCAD Command Line, type **DLDCOLOR**, to start *CustomColors*, *TurboDLD*'s color configuration utility. Make any desired color changes and then **Save** the new color palette and then **Exit** to return to the drawing editor.

What If I Have Problems?

If you encounter a problem or abnormality, please review the installation procedure and *Chapter 7 - Tips, Tricks & Trouble-shooting*. to be sure you haven't overlooked or skipped a step. If you still cannot resolve the problem, refer to *Appendix A* to obtain Technical Support.

PANACEA'S TURBODLD*CLASSIC*

Panacea's TurboDLDClassic has only two purposes:

1. To speed up AutoCAD REDRAWs, PANs, and ZOOMs.
2. To provide a more productive, user- friendly, interface to AutoCAD (via the features described in *Chapter 5 - TurboDLD Features*).

The driver is memory-resident and inserts itself between AutoCAD and the graphics board. It has no other effect on AutoCAD's operation besides speeding the program up; it runs with AutoShade 2 with RenderMan and all versions of 3D Studio, but does not affect the speed of these programs, since they do not support Display Lists.

Installing TurboDLD*Classic* does not change any of the AutoCAD program files or alter any of the drawing files stored on disk. TurboDLD*Classic* was designed to be an easy-to-use, feature-filled product that makes using AutoCAD faster and more productive, not one that makes AutoCAD more complicated and difficult to use.

How does TurboDLD*Classic* make AutoCAD run faster? There are mainly two things the driver does to speed operation:

- AutoCAD stores drawings in a hierarchical structure, with simple elements intermixed with complex ones. Every time the screen is updated, AutoCAD must decode this structure. TurboDLD*Classic* works differently. While you are working, it translates the normal hierarchical structure into a Display List, a series of vectors or polygon fills. (The ability to support display lists was specifically added to AutoCAD with Release 10 via the ADI 4.0 interface.) When you pan or zoom, TurboDLD*Classic* uses the Display List, then writes the resulting vectors to the video board hardware. Since the hierarchical structure does not have to be decoded, drawing proceeds very quickly.
- Lastly, TurboDLD*Classic* gives you new user-interface features, including a bird's eye view and a graphics-mode text window. (See *Chapter 5 - TurboDLD Features*.)

By how much does TurboDLD*Classic* increase the speed of AutoCAD? PANs and ZOOMs, aided by only the Display List, run from two to twelve times faster than a non-display list driver. TurboDLD*Classic*'s internal display list processing can speed things up to the point that REDRAWs can be up to twenty-five times faster than the graphics drivers shipped with AutoCAD.

INSTALLING AND CONFIGURING TURBODLD CLASSIC

Before starting, please make a backup copy of the distribution disk, using the DOS **DISKCOPY** command, and put it in a safe place. Also, if you have been using an older version of a Panacea driver - DLD-VGA, DLD386- VGA, MACH-12... - be sure to remove any references to that driver which may be in your AutoCAD start-up files or in your AUTOEXEC.BAT file.

The TurboDLDClassic installation procedure below is exact, and describes what to do at each step. Installation is slightly different for older releases of AutoCAD than it is with AutoCAD Release 12 and 13. Be sure to follow any directions that may be specific to your particular AutoCAD release. For brief installation instructions, see the Quick Start section at the front of the manual.

Installation

Insert your TurboDLD*Classic* distribution disk into a floppy disk drive and then type:

```
{Drive}:INSTALL
```

Where **{Drive}** is the floppy drive that holds the TurboDLD*Classic* disk.

The first time the installation program is run, you will be prompted for registration information. This information must be given. If you choose to ignore this step, TurboDLD*Classic* will not run.

After completing the registration information, TurboDLD*Classic*'s installation will prompt you for the version of AutoCAD you are installing the driver for.

Next, you will be asked for the name of the directory you wish to install TurboDLD*Classic* into. In the space provided, enter a complete sub-directory path name.

Once the path name has been entered, verify that the information is correct and then press ENTER. The TurboDLD*Classic* files are then copied from the distribution floppy to your hard drive.

Note
For TIGA installations, you must copy the file DLDEXT.RLM from your TurboDLD*Classic* sub-directory to your TIGA software sub-directory before reconfiguring AutoCAD.

Once all of the program files have been successfully transferred, it is necessary to reconfigure AutoCAD to use TurboDLD-*Classic*. Follow the steps below for use with either AutoCAD Release 12 and 13, or Release 10/11, and then proceed to the Driver Configuration section below, to complete the installation.

For Use With AutoCAD Release 12 or 13

Begin AutoCAD with the reconfigure switch by typing

```
ACADR12 -R
or ACADR13 -R)
```

Choose option 3, **Configure Video Display** from the AutoCAD configuration menu.

Type Y at the "**Do you want to select...**" message to display the available video options for AutoCAD.

Select "**TurboDLD *Classic* by Panacea Inc. - v2.1**" from the list of display options.

Alternately, to get to the AutoCAD Configuration menu described above, start AutoCAD as you normally would, and then type **CONFIG** at the Command prompt.

Note
TurboDLD*Classic* usually appears as option 11 in most installations of AutoCAD Release 12. If TurboDLD does not appear in the list of available video drivers, be sure that the AutoCAD environment variables in your AutoCAD start-up file, ACADDRV in particular, have been properly set before typing ACAD -R.

Continue to the "*Driver Configuration*" section below.

For Use With AutoCAD Release 10/386 or 11/386

Run the FASTACAD.BAT file that was copied to the TurboDLD*Classic* sub- directory by typing:

```
C:\TURBODLD\FASTACAD
```

The above example assumes that TurboDLD*Classic* was installed on drive C: in a sub-directory named TURBODLD. FASTACAD sets up the necessary AutoCAD environment variables for use with TurboDLD*Classic*. For more information on FASTACAD.BAT, refer to the Configuring the AutoCAD Operating Environment section at the end of this chapter.

Start AutoCAD and reconfigure the program to use TurboDLD*Classic* by selecting option 5, **Configure AutoCAD** from the AutoCAD main menu. From the next menu, select option 3, **Configure Video Display**. Choose Item 1, **P386 ADI 4.0/4.1** (R11) or **ADI P386** (R10) as your display device. (for more information about configuring for ADI drivers, see the AutoCAD "*Installation and Performance Guide*".

Driver Configuration

Once you have selected the proper display device, the *Turbo*DLD Welcome screen will appear. This screen is an informational screen about what the TurboDLD*Classic* setup program does. Press any key to continue once you have read the overview.

As you continue past the Welcome screen, the TurboDLD*Classic* Driver Configuration menu will appear.

Note
Help during TurboDLD*Classic* setup is only a keystroke away. At any time, you may press the question mark ("**?**") key to get help information pertaining to the current prompt. Please note that the **DLDSETUP.HLP** file must be present in order for help to appear. If you get an error message, be sure that the file DLDSETUP.HLP is in the ACAD\DRV sub- directory (R12/13), or in the TURBODLD sub-directory (R10/R11). Also, as mentioned earlier, be sure that there are no references to older DLD drivers which may cause TurboDLD*Classic* to look elsewhere for DLDSETUP.HLP.

Note
If at any point during the configuration process you want to change an item on a previous screen, just press the **ESC** key to back up. Selecting **NO SAVE, EXIT** at the main configuration menu will return you to the AutoCAD configuration menu without making any changes.

Set-up of TurboDLD*Classic* requires configuration of the various operating parameters of the driver. These parameters have been logically grouped into five menus based on their interaction with AutoCAD. A quick trip through each menu will complete the configuration process.

Note
If none of TurboDLD*Classic*'s options are configured, the driver will be automatically configured as a Generic VGA, with 640 x 480 resolution at 16 colors. All other driver options will assume Panacea's defaults.

Select Graphics Board/Resolution Menu

The **Select Graphics Board/Resolution** menu configures TurboDLD*Classic* for video parameters. From this menu, graphics board/type and display and rendering resolutions for TurboDLD*Classic* will be determined.

Select Graphics Board allows you to manually select your computer's graphics board. A list of video cards is given, based on the findings of TurboDLD*Classic*'s built-in graphics platform detection utility. Use the Up and Down **arrow keys**, and the **Pg Up** and **Pg Dn** keys to scroll through the list and highlight your graphics board. Please note that graphics boards are listed by both platform type and manufacturer's name, alphabetically, therefore you may need to scroll to the bottom of the list to find your particular board.

If your graphics board is not listed, you may choose the generic board option for your video chip type.

Press **ENTER** to accept your choice and return to the Select Graphics Board/Resolution menu.

Select Display Resolution configures the resolution of the AutoCAD drawing screen. Display screen options are based on the findings of TurboDLD*Classic*'s graphics platform detection utility. Use the Up and Down **arrow keys**, and the **Pg Up** and **Pg Dn** keys to scroll through and highlight the resolution and color option you wish to use.

Press the **ENTER** key to accept your resolution choice and return to the Select Graphics Board/Resolution menu.

Note
When configuring TurboDLD*Classic*'s video options, the Rendering Board/Resolution options described below will automatically appear **ONLY** when configuring the rendering modes of AVE Render, 3D Studio or AutoShade 2 with RenderMan. Otherwise, the Display Board/Resolution options will be the only menu choices available.

Similar to the Select Display Resolution menu, **Select Rendering Resolution** configures the resolution of the AutoCAD rendering screen. Rendering screen options are also based on the findings of TurboDLD*Classic*'s detection utility. Use the Up and Down **arrow keys**, and the **Pg Up** and **Pg Dn** keys to scroll through and highlight the resolution and color option you wish to use. Press **ENTER** to accept your choice and return to the Select Graphics Board/Resolution menu.

Note
TurboDLD*Classic*'s graphics platform detection utility queries the graphics card for graphics platform type and installed memory. Based on the detector's information, TurboDLD*Classic* will list a set of Display and Rendering resolutions. Due to the variety of graphics boards and their specific implementations of platform chip sets, the detector is general purpose and cannot always be 100% accurate. Therefore, when selecting a Display or Rendering resolution, be sure to use only resolutions that you know your graphics board, and monitor, will support. If you are unsure which resolutions are supported by your graphics board and monitor, refer to your owner's manual(s).

Highlight **Return To Previous Menu** and press the RETURN key to return to the Main Configuration Menu.

Basic Configuration Menu

The **Basic Configuration** menu configures TurboDLD*Classic*'s AutoCAD screen characteristics; text lines; screen mode; and font size.

The first Basic option is the number of text lines you wish to use for the AutoCAD command line. The default is 3 lines, but values from 1 to 10 can be specified. A number larger than 3 might be useful if you are using this driver in a high resolution mode with small fonts, eliminating the need to frequently swap back and forth between the text and graphics screens.

If you do not want any lines of text at the bottom of the display, use AutoCAD to disable the command area (see the AutoCAD “Interface, Installation and Performance Guide” for more information).

The **Screen Configuration** section is used to tell TurboDLD*Classic* under which of three available modes you wish to use AutoCAD -- dual screen, one monitor for graphics and a second for text; single screen with combined text and graphics; or single screen with combined text and graphics using a graphics mode text window.

Use the arrow keys to highlight a choice based on the following explanations of each mode. Select **Dual** to run in dual screen mode.

Note
Do not select Dual if you don't have a second video board in your system, or unpredictable results will occur. To use dual screens, you must have either a Monochrome board installed with a VGA, TIGA, 8514/A or XGA board, or a VGA board installed with a TIGA, 8514/A or XGA board. Please also note that you cannot have two VGA boards in the same system.

Choose **Single** to run AutoCAD in the standard single screen mode, using a video mode switch to standard VGA text mode during a flip screen. To eliminate the video mode switch to text mode and back into high resolution graphics, choose **Text Window on Single** to configure the single screen to use a graphics mode text window. Using graphics mode to display text allows an instantaneous return from a flip screen. The graphics text window is especially useful when using large monitors which tend to be slow when switching modes. A SHELL command will switch to text mode regardless of the setting.

TurboDLD*Classic* offers you a selection of fonts to use for AutoCAD menus, pull-downs, and dialog boxes. Provided by Panacea are **6x8, 8x8, 8x14, 8x16, 12x20 and 12x24** font options. Also available for use are any of the .BDF fonts, included with AutoCAD Release 12, you may have on your system. For resolutions over 800x600 we recommend the 12x20 or 12x24 fonts.

To make a font selection, highlight **Select Font From List** to display a list of available fonts. Use the Up and Down **arrow keys**, and the **Pg Up** and **Pg Dn** keys to scroll through the list and highlight the font you wish to use. Press the **ENTER** key to return to the Basic Configuration menu. To limit the available font choices, choose the **Select Font Filter** menu selection to choose a .BDF font type from those listed. When you return to the Select Font From List option again, only those files matching the filter name will be available for use.

Note
If you do not see any .BDF font selections listed in the Basic Configuration menu, you will need to add the following line to your AutoCAD start-up batch file.

```
SET DLDFONT = C:\ACAD\DRV
```

Highlight **Return To Previous Menu** and press the **RETURN** key to return to the Main Configuration Menu.

User Interface Menu

The **User Interface** options configure the productivity features of TurboDLD*Classic*. These features change the way you manoeuvre around AutoCAD and execute commands.

Double Click Timing

The first timing option sets the **Double Click** delay. This time represents the largest amount of time that can elapse between the release of the digitizer button and the subsequent press of the same button in order to detect a double click. Setting this number too low may result in the computer seeing two separate clicks, while setting the time too high may slow down response time for single button presses. Values between 10 and 30 are recommended. Double Click time is measured in hundredths of seconds. PC systems have a timing resolution of 5/100ths of a second, therefore, the entered time is rounded internally by the driver to most closely approximate the nearest multiple of 5. This means that a time setting of 23 is the same as a setting of 27, both of which get rounded to 25 (or one-quarter of a second). In simplest terms, when adjusting these values, it is best to use multiples of 5.

Big Picture Options

The Big Picture, Panacea's bird's-eye view, can be configured to suit your needs or allow for better differentiation within a complex drawing.

At the BP Button menu option, enter the number of the button on your mouse or tablet that you wish to use as your bird's eye “hot key”. A Double Click on that button will cause TurboDLD*Classic*'s bird's eye to pop-up on -screen.

BP Highlight controls how the current viewport reference in the Big Picture appears on the screen. **Patt Line** displays the viewport reference using dotted lines. **XOR Rect** uses a contrasting rectangle to display the viewport reference. The **Both** option will use a combination of Patt Line and XOR Rect to display the viewport reference.

In its fastest mode, TurboDLD*Classic* only refreshes the Big Picture when objects are added to a drawing, until a DLDRE-FRESH command is issued. Moved or erased objects will remain in the Big Picture. Depending on a drawing's complexity, if BP Refresh is Disabled, the Big Picture could become confusing, displaying vectors that are actually erased, and no longer part of the drawing. To configure the Big Picture to be updated as objects are drawn AND erased, **ENABLE** the BP Refresh. If speed is a major concern, **DISABLE** it.

Highlight **Return To Previous Menu** and press the **RETURN** key to return to the Main Configuration Menu.

Expert Configuration Menu

The **Expert Configuration** menu sets the basic internal functions of TurboDLD*Classic*. Options in this menu allow for customization of the driver for speed or to adjust for memory constraints.

Display List enables or disables the display list feature of TurboDLD*Classic*', the core of the driver's AutoCAD performance boost. This option should always be set to **ENABLE**, as disabling the Display List will cause TurboDLD*Classic* to run as an ordinary non-display list driver.

Use AutoCAD 31-bit Space, configures TurboDLD*Classic* for use with AutoCAD R12's 31-bit logical drawing space. When set to Yes, the driver will use AutoCAD's extended 31-bit drawing space. Selecting No, will use a 15-bit drawing space, similar to that of AutoCAD R10/R11.

The 31-bit logical space allows you to extend your Regen-less zooming ability by a factor of several million, at the cost of more memory. Additional memory is used by AutoCAD for the drawing space, and by TurboDLD*Classic* for the Display List. 31-bit zooming and panning is about 10%-20% slower than for 15-bit logical space.

The Use AutoCAD 31-bit Space setting is ignored for AutoCAD R10/R11 installations as these versions only support a 15-bit logical drawing space.

The echo of TurboDLD*Classic* internal commands can be enabled or disabled with the **Internal Command Echo** option. If you would like to see TurboDLD*Classic*'s internal commands display at the AutoCAD command line as they are executed, enable this option.

BP Zoom Mode sets the Big Picture (BP) zoom definition area display options. In a zoomed view of the static BP, as the current viewport is zoomed or panned, **FLOAT** mode causes the image in the BP to move around within the bird's-eye window, keeping the zoomed viewport area fixed in the center of the bird's-eye. **FREEZE** mode will lock the current BP contents into place to provide a better overall frame of reference. Disable DL in PSpace allows you to turn off the Display List while in PaperSpace only. This function is useful for PSpace users who find the display list size grows excessively for their drawings.

After all Expert options have been set, highlight **Return To Previous Menu** and press the **RETURN** key to return to the Main Configuration Menu.

Saving Configuration Information

Once all of the TurboDLD*Classic* parameters have been configured, select **SAVE, EXIT** to save the configuration information and create **DLDSETUP.DAT** in either the ACAD\DRV sub-directory (R12/13) or the TURBODLD (R10/R11) sub-directory. DLDSETUP.DAT is TurboDLD*Classic*'s configuration.

After answering AutoCAD's standard driver configuration questions, you will return back to the AutoCAD configuration menu. To begin your accelerated AutoCAD, for AutoCAD R12/13, select option 0, **Exit to Drawing Editor**; for AutoCAD R10/11, choose option 1 or 2, **Open New/Existing Drawing**.

Verifying Your TurboDLDClassic Installation

To verify that TurboDLD*Classic* is running and installed correctly, follow one of these two simple tests.

- If you have the AutoCAD side menu enabled, look for the Panacea Logo in the lower right-hand corner.
- If you are running AutoCAD without a side menu, type DLDVER at the AutoCAD command prompt. If TurboDLD-*Classic* is loaded and running, this command should return your current version, serial number and the registered user's name.

If TurboDLD*Classic* was not correctly installed, you will receive an AutoCAD error when the DLDVER command is entered.

Reconfiguring TurboDLDClassic

To reconfigure TurboDLD*Classic* follow the instructions for AutoCAD Release 12/13 or Release 11/10 below.

AutoCAD Release 12/13

If you need to make changes to your Release 12/13 TurboDLD*Classic* configuration, type **CONFIG** at the AutoCAD command prompt or use **ACADR12 -R (ACADR13 -R)** when starting the program. Select option 3, **Configure Video Display**. Answer **No** to the “Select a new video driver...” prompt to start the TurboDLD*Classic* configuration program. Make the desired changes to the driver and then Save and Exit to continue to the AutoCAD drawing editor.

AutoCAD Release 10/11

To change a Release 10/11 configuration, select option 5, **Configure AutoCAD** from the AutoCAD main menu. and then from the configuration menu, select option 3, **Configure Video Display**. Answer **No** to the "**Select a new video driver...**" prompt to start the TurboDLD*Classic* configuration program. Change the desired driver options. Save and Exit to return to the AutoCAD configuration menu. Open or begin a new drawing.

Note
To completely reconfigure TurboDLD*Classic* using Panacea's defaults, delete **DLDSETUP.DAT** from either the \ACAD\DRV sub-directory (R12/13) or the TURBODLD sub-directory (R10/11) and then follow the Driver Configuration instructions at the beginning of this chapter.

Configuring The AutoCAD R10/11 Operating Environment

During the TurboDLD*Classic* installation process, a FASTACAD.BAT file is created and placed into the TURBODLD sub-directory. FASTACAD.BAT contains four lines that set four separate environment variables: **DLDCFG** - used by TurboDLD*Classic* to find all of its configuration files; and **DSPADI, RCPADI, RDPADI** - used by AutoCAD, 3D Studio and AutoShade 2 w/RenderMan, respectively, to find the driver file. FASTACAD.BAT must be run prior to starting AutoCAD R10 or R11 and only needs to be run once per system boot. For automatic loading of the environment variables, FASTACAD.BAT may be added to the AUTOEXEC.BAT file or an AutoCAD start-up batch file.

To add execution of FASTACAD.BAT to your AUTOEXEC.BAT file, insert the line

```
CALL D:\TURBODLD\FASTACAD
```

anywhere in the file. The above example assumes that FASTACAD.BAT resides in a sub-directory on drive **D:** called **TURBODLD**.

If you don't wish to put FASTACAD in your AUTOEXEC.BAT file, you may put it in a batch file which also starts AutoCAD, or simply remember to run the file before starting AutoCAD. (You should also stop using any other ADI display driver that may have come with your graphics board, since TurboDLD*Classic* is a superset of that type of driver).

Note
If you get the message

```
Out of Environment Space
```

when you run FASTACAD.BAT, you will need to enlarge your system's environment. This is accomplished by adding the following line,

```
SHELL=C:\COMMAND.COM /P /E:768
```

to your **CONFIG.SYS** file. The **/E:768** specifies an environment size of **768 bytes**. Change this number as appropriate. You will need to reboot if you modify your CONFIG.SYS file, in order for the changes to take effect.

Configuring TurboDLDClassic for Rendering

To install and configure TurboDLD*Classic* for use with AutoCAD R12/R13's AVE Render, 3D Studio or AutoShade 2 w/RenderMan, please read *Chapter 8 - Rendering with TurboDLDClassic*.

CHANGING COLORS

Included with TurboDLD*Classic* is Panacea's revolutionary ***CustomColors*** color configuration utility. What's so unique about CustomColors is that it gives you the ability to modify all your changeable AutoCAD colors, including menu colors, text colors, dialog box colors, and even drawing colors, while running AutoCAD.

CustomColors simulates an AutoCAD screen, complete with all possible objects. At the AutoCAD drawing editor command line, type **DLDCOLOR** to edit the color configuration. If you have configured TurboDLD*Classic* for 256 colors, you will be greeted with an informational warning message. The message appears to remind you that if you assign a drawing color to a physical color above color 15, you may experience unusual results from AVE Render. Once you enter the utility, you are provided with the following configuration menu at the bottom of the screen:

```
(O)bject, (D)rawing, (P)hysical, (A)DI reset, (V)GA reset, (L)oad, (S)ave, (E)xit
```

The menu items perform the following functions.

Object Selecting this option will allow you to change the color of any AutoCAD screen object, such as the graphics area background color, the menu area text color, or the border line color. Select the screen object whose color you want to change by moving the highlight box around with the left, right, up and down cursor control keys. Press **ENTER** when you have highlighted the desired object. Note that a one line description of the object type is displayed at the top of the screen as you move the box around.

Once you have selected the object, another highlight box will appear around the color boxes nearest the bottom of the screen. These are the physical colors that the video board supports. Use the left and right cursor keys to select the physical color to be used for the object you have selected and press **ENTER**. The screen will quickly redraw with the new color selection for the object you have chosen, and you will be back at the start of object selection in case you want to change the color of another object.

Note
Special Keys

Note that at any time during color configuration, you can press the ESC key once or twice and return to AutoCAD. Also, pressing the question mark (“?”) key will provide you with context sensitive help.

Drawing This 16 color configuration option allows you to modify all AutoCAD drawing in a fashion similar to the way you change object colors. When this option is selected, a highlight square appears in the drawing color area of the simulated AutoCAD display. The square can be manipulated using the four cursor keys. You may notice that the drawing color portion of the display is laid out just like the CHROMA drawing supplied with AutoCAD. When you press ENTER, the highlight moves down to the 16 physical colors, just as it does during the object color selection. Using the arrow keys, pick the physical color you want to represent the selected drawing color. Press **ENTER** for your selection to take effect. When using 256 colors, you do not have a (D)rawing Color option. All 256 colors are available as drawing colors.

Physical Choosing the Physical option from the menu allows you to alter the red, green, and blue components of the physical colors. A highlight box will appear in the row of 16 physical colors at the bottom of the screen. Use the **arrow keys** to select the color you wish to edit and press **ENTER**. Three “sliders” will appear near the bottom of the display, with the horizontal position of the slider for each of the color components (red, green, or blue - RGB) indicating the relative intensity of the component. You can use the left and right arrow keys to move the slider for the selected component, or type in a number from 0 to 255.

Note
Many graphics boards do not support 255 different intensities for each color component. Therefore, when you enter an intensity, *Custom Colors* will round it to the nearest intensity which your graphics board supports.

For example, VGA supports 64 intensities per RGB color component. This means intensities increase in multiples of 4 (256/64), therefore an intensity of 0 is the same as an intensity of 3.

The up and down arrows allow you to select which component you want to alter. As you manipulate the sliders, all objects on the display that are of the same color as the selected physical color will change in hue. This way you can visually determine the most appropriate setting for your display. Pressing ENTER will set the RGB values you have selected for the physical color you were modifying.

ADI reset This constructs a default ADI color palette, as defined in the Autodesk Device Interface Driver Development Kit, and in Panacea's 8514/A and XGA non-display list drivers shipped with AutoCAD Release 12.

VGA reset This constructs a default VGA color palette in the first 16 color entries. The remaining palette colors remain identical to those used for ADI devices.

Load This reloads the color information from the file DLDCOLOR.DAT. It will be loaded from the directory pointed to by DLDCFG (AutoCAD R10/R11), or from the AutoCAD R12/R13 \DRV directory. If DLDCOLOR.DAT cannot be found, an error message will be displayed. This command is useful because it allows you to go back to your previously edited color configuration in case you have made some mistakes in configuring your colors that you want to undo.

Caution

Don't Accidentally Wipe Out Your Changes.
Selecting **ADIreset**, **VGAreset**, or **Load** will irrevocably wipe out any current color changes you may have made, unless you have just saved your new setting, so use them only when you really need to, such as when you have made so many color changes that you can't seem to get back to a reasonable place and just want to start over.

Save Saves the current color palette to DLDCOLOR.DAT. It will be saved in the directory pointed to by the DLDCFG environment variable in FASTACAD, or to the \ACAD\DRV (R12/R13) directory if FASTACAD is not used.

Exit Exits *CustomColors*. If you have made palette changes, but not saved them, you will be asked if you want to save your changes before exiting.

Note

When running at more than 8 bits-per-pixel (256 colors), setting physical colors via DLDCOLOR will have no visible effect until a SAVE and EXIT occurs.

TURBODLD FEATURES

TurboDLD*Classic* offers AutoCAD users many features and productivity options. This chapter alphabetically lists the basic commands found in all TurboDLD versions and provides the correct syntax for their usage. For a brief summary of TurboDLD*Classic* commands, see Appendix B or, while running AutoCAD, type **DLDHELP** at the command prompt.

The Big Picture

The most important feature of TurboDLD is the support of a world view called *The Big Picture*™ or just **BP** for short. Bringing up the BP allows you to view where in the current drawing you are currently located in your active viewport, and it also gives you the ability to dynamically move to another part of the drawing without having to exit the function you're currently executing.

DLDBIGPIC

To call up the Big Picture, at the AutoCAD command line, type DLDBIGPIC, or double click the right mouse button or button 2 on your digitizer puck (assuming you haven't redefined your Button 2 double click function - see *Chapter 3*). This will bring up the BP on the display. You will see cross-hairs and a highlighted section indicating which part of the whole drawing you're currently viewing - the size of the image will be determined by the AutoCAD logical drawing space being used, 15-Bit or 31-Bit. The selection or pick box will have an “X” through it. Move the mouse/digitizer around to locate your pick box. If you can't see a small box moving, click the left mouse button (the pick button) to size down the pick area. In size mode, the pick box will have a right pointing arrow in it (“-->”). If the contents of the BP are too small to work with, you may increase the size of the image in the window by pressing the “+” key **on your numeric keypad** this performs an incremental zoom within the BP. Likewise, pressing the “-” key will reduce the size of the contents of the BP. Pressing the **HOME** and **END** keys will put the BP into its smallest and largest size, respectively. The smallest BP size is defined as being exactly the same view as that in the viewport referenced by the BP. The largest BP size is defined as the largest image possible without causing a Regen.

The BP pick box is always proportioned to the proper aspect ratio for the current viewport. Clicking the pick button again will put you back into the pick box move mode (“X”). This operation is very similar to using the Zoom Dynamic feature of AutoCAD. Once you have positioned the pick box on the area you would like displayed in the active viewport, click any button other than the pick button to initiate the zoom, or press the Enter key. If you want to abort the operation, just hold any mouse or digitizer button down, other than the pick button, until the BP disappears. You can also hit any key on the keyboard, other than Enter, to cancel the BP. Note that the BP is updated with new drawing commands, but that moves or erases will not update the image unless BPREFRESH is enabled. A DLDREFRESH or a REGEN will manually update the BP with the current drawing changes.

DLDBPDIM

Resizing and repositioning the BP can be accomplished by typing this command at the AutoCAD prompt. DLDBPDIM allows you to tell the driver where you want the BP placed and how large you want it to be. The size is limited to being between one-quarter and one-half the width of the display. This command functions like the pan/zoom selection box in the BP, as far as positioning and dimensioning goes. To abort, hold the right mouse button, or any other digitizer button, down until the moving box disappears.

DLDBPFREEZE

Sets the Big Picture (BP) zoom definition area display options. DLDBPFREEZE toggles the Big Picture Zoom Mode parameter in TurboDLD's Expert Configuration Menu.

In a zoomed view of the static BP, as the current viewport is zoomed or panned, **Float** mode causes the image in the BP to move around within the bird's-eye window, keeping the zoomed viewport area fixed in the center of the bird's-eye. **Fixed** mode will lock the current BP contents into place to provide a better overall frame of reference.

DLDBPHILIGHT* Changes the highlight used to display the Big Picture among three modes, **Patt Line, XOR Rect**, and **Both**. Patt Lines will use dotted lines to outline the Big Picture pick box. XOR Rect will use a contrasting rectangle to highlight the pick box, and Both will use a combination of Patt Lines and XOR Rect.

DLDBPMODE Toggles the Big Picture between Panning and Zooming modes. In panning mode, the AutoCAD drawing will dynamically display the contents of the Big Picture pick box as you size and move it. In Zoom mode, the AutoCAD drawing remains stationary until the area in the Big Picture pick box is accepted. At which time it changes to the chosen zoom level.

DLDBPREFRSH* This convenience feature of TurboDLD, toggles the Big Picture update mode. If disabled, the BP is updated manually by issuing a DLDREFRESH command. When enabled, DLDBPREFRSH will cause the BP to refresh automatically when an object is drawn or erased. The BP operations therefore will be slower with DLDBPREFRSH enabled.

DLDBPSTATIC Toggles the BP to stay on the screen at all times. Use this command if you want the BP on the screen in order to conveniently move into it whenever you want to zoom or pan. Typing DLDBPSTATIC will put up the BP at the position you set with DLDBPDIM. Note that the static BP will disappear when other menus pop-up on the screen, and it will also disappear if you draw any objects which may overwrite the BP. With these exceptions, the BP will remain on-screen until you either exit the drawing screen or you type DLDBPSTATIC again.

DLDCOLOR Invokes *CustomColors*™, Panacea's color configuration program. See *Chapter 4 - Configuring Colors* for more information on configuring TurboDLD colors.

DLDCOMPACT Forces a manual garbage collect of Display List memory thereby returning unused display list memory back to the AutoCAD memory pool.

DLDDLIST* This command toggles the display list function of TurboDLD*Classic* on and off.

Note
Please note that if the display list is turned off, you will be running AutoCAD as though you were using a standard non display list driver - pans, zooms and redraws will be **MUCH** slower with DLDDLIST **disabled**.

DLDECHO* Toggles internal TurboDLD command echoing at the AutoCAD command line. When TurboDLD commands are executed via the digitizer, they generate internal commands which will be displayed at the AutoCAD command line if DLDECHO is enabled. Disable DLDECHO to simplify the command line.

DLDHELP Provides a list of TurboDLD commands with one-line description of each, within AutoCAD. It's recommended that you flip to the text screen by pressing the **<F1>** key to view the output.

DLDREFRESH Refreshes the Big Picture to reflect the most current changes to a drawing. If the static BP is up, it will be updated. If not, the next time the BP is brought up, it will contain a current representation of your drawing, provided no changes were made between the DLDREFRESH command and the DLDBIGPIC command. For Automatic update of the Big Picture, see the DLDBPREFRSH command above.

DLDSTAT Displays the current TurboDLD status. A listing of the current TurboDLD parameters is displayed at the AutoCAD command line. A flip to the graphics screen is recommended to view the status report.

DLDUSAGE	Use DLDUSAGE if you want a to-the-byte breakdown of how memory is being used, specifically for display list processing. DLDUSAGE returns information regarding the memory each viewport is occupying. Since AutoCAD supports multiple viewports, it is possible to have multiple display lists. A new or empty drawing will have a display list of 4k bytes.
DLDVER	Displays the TurboDLD version, serial number and registered user's name at the AutoCAD command line.
PD, ZD, ZT	Included in TurboDLD for backwards compatibility with MACH-12. These commands will invoke the Big Picture
ZI, ZO	ZI - Zoom In and ZO - Zoom Out will zoom the current viewport in or out incrementally.
ZP, ZA	ZP - Zoom Previous and ZA- Zoom All are passed directly to AutoCAD.
ZW	ZW - Zoom Window. Invokes an AutoCAD Zoom Window command
CD	Returns any unused DLM memory to the VMM.

*These commands, when issued within AutoCAD, will override the selections made during TurboDLD*Classic* configuration, for the current drawing session only. Exiting AutoCAD and subsequently restarting will cause all feature settings to revert back to those selected in the TurboDLD*Classic* configuration menu. If you wish to make the current changes permanent, reconfigure TurboDLD*Classic* by following the reconfiguration instructions in *Chapter 3*.

MEMORY USAGE AND LISTS

TurboDLD*Classic* shares extended memory with AutoCAD via the PharLap Virtual Memory Manager. This means that TurboDLD*Classic* will automatically page to disk if it uses up all the RAM that AutoCAD has left for its use. See the AutoCAD “Installation and Performance Guide” for more information on Virtual Memory Management.

If you start seeing excessive hard disk accesses during PANs, REDRAWs, and ZOOMs while using AutoCAD with TurboDLD*Classic*, try using the DLDCOMPACT command. If this doesn't affect the amount of disk access it's probably time to add more memory to your system. Contact your AutoCAD dealer for assistance in upgrading your memory.

Regarding display list memory, it's important to realize that TurboDLD*Classic* speeds up AutoCAD operations by creating a Display List in memory, and sending that list to the graphics board for pans, zooms and redraws. But a display list takes up memory.

How much memory?

For production use, we recommend that at least 1 megabyte is available for TurboDLD*Classic*. To determine how much memory AutoCAD is using, use the status command while in AutoCAD (refer to the AutoCAD “*Installation and Performance Guide*” for more information). The Display List for a simple drawing like the shuttle Columbia might only require 20K bytes for the Display List. Complex drawings may require several megabytes. We have seen Display Lists for a drawing range from one-tenth the size of the drawing file to three times the size; in general, the Display List averages about one and one-half times the size of the DWG file size. This means that you should reserve approximately two and one-half times your average DWG file size in RAM for the drawing and the Display List. Also, complex objects such as circles and text “expand” when translated into display list format, so a drawing with a lot of complex objects and text will have a larger Display List than a simpler drawing. If there is significant hard disk activity while you are using TurboDLD*Classic*, it may be an indication that you should add more memory to your system.

Note
Text takes up a disproportionate amount of space in the Display List. To keep the list small, put text in its own layer of the drawing. Then don't display the text layer when editing the rest of the figure. This will keep memory consumption down and speed up PANs, REDRAWs, and ZOOMs.

Note
The AutoCAD manuals have an excellent section on performance, concentrating on memory usage. To get the most performance out of the program, read the appropriate sections of those manuals as well as this one.

TIPS, TRICKS & TROUBLESHOOTING

Because TurboDLD*Classic* supports such a wide range of graphics board platforms, it is necessary to be aware of certain idiosyncrasies. This chapter outlines known configuration and usage problems of TurboDLD*Classic* and of various graphics platforms. Also included are tips to help gain performance from AutoCAD itself. These tips and tricks are a compilation of the most commonly asked Technical Support questions. Therefore, if you are having problems, please review this chapter. You may be able to quickly answer your own question.

Memory Management

System lock-ups, random re-boots or missing displays, indicate you may have a memory conflict on your system. Different video platforms access different, specific, portions of upper memory blocks. If these memory addresses are in use by TSRs or other programs or memory managers, the above symptoms will occur. To avoid these problems it is necessary to use an expanded memory manager to allocate the specific address range to the graphics board. The address to exclude usually falls within the range of A000 to C7FF. XGAs and some TIGAs go beyond C7FF up to DFFF. Some common expanded memory managers are QEMM, 386MAX and EMM386 (which comes with DOS 5.0 and later and Windows).

QEMM and EMM386 use similar commands to perform the exclusion.

```
X=B000-B7FF
```

is added to the QEMM386.SYS or EMM386.EXE command line in the CONFIG.SYS file.

386MAX uses a video memory parameter which look like this

```
VIDMEM=B000-B7FF
```

Note that it may be necessary to run an optimization utility with some memory managers in order for the exclusion to take effect. Refer to your memory manager's documentation for specific details.

If the range of A000-C7FF seems like a large area to exclude, you may decrease the range by changing the ending address to determine the exact range that you need to exclude. Typically B000-B7FF, as shown in the examples above, will be sufficient. You may try other ranges, by altering the beginning and/or ending address, to isolate the exact region the graphics board requires. Remember to re-boot after changing the CONFIG.SYS file. Some common requirements are shown below.

Caution

Due to the manner in which they re-map video BIOS memory regions in RAM, memory optimization utilities, such as QEMM's Stealth and EMM386's HIGHSCAN, should be avoided.

XGA	XGA-based graphics boards must have at least the range of B000-DFFF available.
SVGA and S3	SVGA and S3 boards usually use the area of B000-B7FF.
TI 34020-based TIGA Boards	TI 34020 graphics chip-based TIGA boards request the range of A000-DFFF, however, C000-CFFF usually works.

Upgrading From Older Panacea DLD Drivers

Be sure to remove any reference to previous DLD driver commands (FASTACAD calls or SET DLDCFG, RCPADI, RDPADI and DSPADI parameters) or sub-directories that may be in your AUTOEXEC.BAT file or in AutoCAD start-up batch files. Such references could cause AutoCAD and TurboDLD*Classic* to look in the wrong place for setup information.

If have been using a previous Panacea DLD driver and wish to use the color palette you customized for it with TurboDLD-*Classic*, you may copy the DLDCOLOR.DAT file from your older DLD's sub-directory to the sub- directory you specified during the TurboDLD*Classic* installation process.

Maneuvering Through AutoCAD

Zooming

Watch the Snap When zooming into an image, be aware of your AutoCAD grid snap setting. If you are zoomed extremely far into a drawing and you are having trouble moving your digitizer cursor, you may be snapping to a point which is not part of the zoomed viewport. If the cursor only moves to a single point, or is not on screen at all Snap might be turned on. Check the status bar to be sure. If you see “Snap” listed, press the F8 key to turn it off.

Zoom $1 An undocumented AutoCAD command can help speed up performance in some situations, ZOOM $1. ZOOM $1 will allow you to set the extents of AutoCAD Release 12's logical drawing space. by issuing the following command sequence:

```
ZOOM
$1
CTRL-C
CTRL-C
CTRL-C
CTRL-C
```

The catch is that the command is only in effect until a REGEN is issued.

Be sure to zoom to the view of your drawing you wish to use as the extents before issuing the above. If you find this command to be useful, you may also want to try configuring for a 15-bit drawing space -- See *Chapter 3 - Installing and Configuring, “Expert Configuration” menu.*

TurboDLDClassic and Paper Space

TurboDLD*Classic* features do not work in AutoCAD's Paper Space. The execution of any TurboDLD*Classic* command in Paper Space, due to a limitation in AutoCAD, will result in an error message at the command prompt.

Zoom Dynamic

We do not recommended using ANY *Classic* features while in the Zoom Dynamic mode of AutoCAD. Because TurboDLD*Classic* cannot detect when Zoom Dynamic has been initiated, unpredictable results will occur. Use the Big Picture feature of TurboDLD*Classic* to accomplish the same task as Zoom Dynamic, in a much more efficient fashion.

AutoCAD Release 12 and 13's CONFIG Command

Configuring TurboDLD*Classic* in the middle of a drawing session using the CONFIG command is a bit like exiting and re-starting AutoCAD.

Sticky Cursors

In the Big Picture:

Sometimes, when using the Big Picture, the digitizer cursor will appear to stick to the edge of the bird's-eye. This is normal and is the result of the digitizer puck responding to the whole screen area while the Big Picture only occupies a small portion of the screen. Because of the difference in resolutions between the screen and the digitizer, when the screen cursor enters the Big Picture area, the digitizer puck has more drawing area to cover in order to get to the same location as the screen cursor.

Within AutoCAD

The following solutions have been used to fix slow mouse/digitizer response problems when using TurboDLD drivers.

- Make sure that you are not using a real mode driver for your digitizer. Real-mode drivers do not interact properly with protected-mode drivers. AutoCAD Releases 10/386, 11/386, and 12, are all protected-mode. We recommended that you switch to either the AutoCAD supplied driver or the manufacturer's latest ADI 4.2 driver for the digitizer.
- Make sure that no digitizer key re-assigning TSR's are being used. Some digitizers, specifically the Kurta IS/One, use TSR's to allow digitizer programming; these will not work with TurboDLD.
- Try deleting all files with a .DAT extension from your TurboDLD sub-directory and re-configuring TurboDLD from scratch. Occasionally, these .DAT files can be corrupted or updated improperly during installation or reconfiguration.
- Try changing the double-click interval time to 5 or 10. This can be adjusted from the User Interface Menu of TurboDLD.
- If a specific digitizer key is causing a problem, use the Delete key to remove any AutoCAD commands from that button. Button programming commands are in the User Interface menu within the Button Configuration Menu. Be sure to use the Delete key, not Spaces to remove existing commands.
- Depending on the digitizer you are using and how it is configured, try adjusting CFIG386's -VSCAN parameter. Check the AutoCAD “Interface, Installation and Performance Guide” for specific instructions on how to use CFIG386.
- Try using the CFIG386 -DEMANDLOAD ON setting for AutoCAD.
- If there is one available, try to duplicate the problem with a second mouse or digitizer, this will help to isolate the cause of the problem. - If your digitizer is capable of it, try configuring for SummaSketch emulation and use the driver included with AutoCAD. - If there is a choice in your digitizer's configuration, try configuring it for Polled vs. Interrupt driven, or vice versa.

Using AutoCAD Commands

Since TurboDLD*Classic* is totally transparent to users with respect to using normal AutoCAD commands to REDRAW, PAN, and ZOOM, you still have to suffer from some of AutoCAD's nuances. One of these nuances is that ZOOM ALL and ZOOM EXTENTS both force a REGEN, because AutoCAD does not keep track of various boundaries necessary to avoid the REGENs. And REGENs are rather time-consuming and don't use any display list processing to speed themselves up. One way around this problem is to use another of AutoCAD's built-in features, namely the VIEW command.

When you first load your image and see the whole drawing on the screen at once, just type **VIEW Save ALL**, which will save the display position you see under a view named “All”. Then, after you've done some detailed editing and want to return back to the big picture, type **VIEW Restore ALL**, instead of ZOOM All or ZOOM Extents, and the full drawing will be restored to the display at display list speeds, without a REGEN.

Another nuance of AutoCAD is that if you zoom in too far or pan over too far, you may inadvertently cause a REGEN. AutoCAD again provides a very simple solution: the REGENAUTO command. Just type **REGENAUTO Off** at the AutoCAD command prompt, and automatic REGENs will be disabled. The REGENAUTO setting is also saved as part of your drawing file, so you only need to execute it once per drawing. You may even want to set REGENAUTO off in your ACAD.DWG drawing template so that all your drawings are created with REGENAUTO set off.

Third-Party Software

If you are having trouble with TurboDLD*Classic* and third-party AutoCAD applications, be sure that the third-party application supports ADI 4.2. In order to use the ADI 4.2 specification, third-party applications require new T-Drivers and therefore must be revised. If an application does not specifically say that it is ADI 4.2 compatible, it probably is not. Check with the manufacturer to be sure.

If your third-party application is ADI 4.2 compatible and you are having trouble using TurboDLD*Classic*, try running AutoCAD without the third- party application to try to isolate the problem. Also, try the third- party application with the SVADI driver shipped with AutoCAD, this will also help to isolate the source of the problem.

Also note that any third-party TSR that needs to access the display may not work properly when using AutoCAD with any advanced ADI display driver, especially if the TSR switches graphics modes. Most TSRs will not support the same graphics platforms as TurboDLD*Classic* and therefore, will not be able to accommodate mode switching back and forth.

Troubleshooting TIGA Installations

TIGA board software version 2.0 or later MUST be loaded prior to installing and using TurboDLD*Classic*. Usually with TIGA boards, the problem is with the TIGA software initialization and not with TurboDLD*Classic*.

If you are having trouble configuring your TIGA graphics board, check the following:

1. Make sure that the TIGA software is the correct version and is loaded properly by changing into the TIGA software sub-directory and typing **TIGACD**. This command will return the version and status of the software.
2. Make sure that the TIGA environment variables are pointing to the correct TIGA sub-directory and that a system compatible interrupt is being specified.

   ```
   SET TIGA = -mC:\TIGA -lC:\TIGA -i0x60
   ```

   Using the example above, the -m and -l parameters point to a TIGA v2.0 or later software sub-directory named TIGA. The -i parameter indicates the hex value of 0x60. Another common interrupt is 0x7F. (Please note that the -m, -l and -i references above are lower case M, L and I, respectively.)

   If you have upgraded your TIGA software, and are using your AUTOEXEC.BAT file to initialize the TIGA board, be sure that the path (see above example) is pointing to the new software and not the previous version. Try remarking out the TIGA initialization line(s) from the AUTOEXEC.BAT and entering them manually at the DOS prompt.
3. Make sure that the DLDEXT.RLM file from the TurboDLD*Classic* sub- directory has been copied into the TIGA sub-directory.
4. Try running your TIGA demo program if you have one. If you get errors similar to “Can't initialize TIGA..” verify the installation per the steps above.

Slow Scrolling

If you have configured for the graphics mode text window and are experiencing slow script file execution, it might be your graphics board. Some graphics boards do not handle graphics mode text scrolling very quickly. To test this, try configuring for the SINGLE screen option in the Basic Configuration sub-menu of the TurboDLD*Classic* Configuration menu.

Color Considerations

Switching Color Modes When switching color modes, (i.e. from 16 to 256 or vice versa) the DLDCOLOR command should be run in order to reconfigure your color palette for the number of colors selected.

Changing to a Light Background If you would like to switch from TurboDLD*Classic*'s default black background to a white or light background, type

```
DLDCOLOR
```

at the Command prompt to start the CustomColors utility. Once there, press O for (O)bject and then move the highlight to the Background item and then press ENTER. Move the color highlight to the desired new background color and press ENTER. Next, move the highlight to the Cursor item and press ENTER. Move the color highlight to the desired new Cursor color and press ENTER. Press the ESC key and then S to (S)ave and E to (E)xit. You should now have a light background with a dark crosshair.

Corrupt TurboDLDClassic Data Files

If you are having problems configuring TurboDLD*Classic*, be sure that you are using either the original distribution diskettes or a faithful copy made with the DOS DISKCOPY command. We have at times received Tech calls where we could not trace a specific recurring problem which later turned out to be a bad duplicate disk.

If everything appears to have installed correctly and you still can't seem to figure out what's happening, try deleting the DLDSETUP.DAT, from your TurboDLD*Classic* sub-directory. You'll then have to reconfigure the driver from scratch. We've seen some strange things happen when these files have become corrupted.

Configuring TurboDLD for Your Graphics Board

You may need to look for your graphics board by its video chip instead of its name. There are many graphics board manufacturers, some not so famous enough to be known by a brand name. In these cases, you must configure for your graphics board by its video chip type. Check your documentation, or if you already know it, use the Generic selection for your chip type.

Static BP in Panning Mode

When using the static Big Picture in panning mode, the drawing screen will continue to pan until you move the cursor out of the Big Picture viewport. The drawing screen will display the selected view correctly.

Pull-Downs and Crosshairs Leave Trash on the Screen

If you are using an S3-based graphics board, a problem may exist if you have more than two COM ports configured. Internally, the S3 chip uses address 2E8h which is also used for a COM4 address. If you have a modem or another COM port assigned as COM4, you will have a conflict. The problem shows up in graphics mode and usually leaves lines from the cross-hair cursor or pull-downs on the screen. To correct the problem, remove the conflicting COM port. If your modem or COM board allows you to assign it to other ports, change the settings according to your device to leave COM4 available for your S3 based graphics board.

RENDERING WITH TURBODLD CLASSIC

Since TurboDLD*Classic* is a combined, display and rendering, device driver, it can be used to generate high-resolution renderings with Autodesk's 3D Studio and AutoShade programs, as well as with AutoCAD R12's built-in AVE Render. Configuration for these programs basically involves selecting the proper Display and Rendering resolutions from the TurboDLD*Classic* configuration menu. The other driver configuration options, except for font selection options for 3D Studio, are ignored for rendering because a Display List, and its configuration parameters, is not used during the rendering process.

Autodesk 3D Studio

Configuring 3D Studio requires three steps. First, you need to set the environment variables for 3D Studio by running the **PANA3DS.BAT** file that was copied into your TurboDLD*Classic* sub-directory.

For 3DS v1.x and 2.x, you must edit the **3DS.SET** file located in your 3DS directory. The following three lines of the 3DS.SET file tell 3D Studio which driver to use for the rendering screen, main screen and materials editor screen, respectively. Locate the lines that begin with:

```
RENDER-DISPLAY
MAIN-DISPLAY
MATERIAL-DISPLAY
```

and change them to use the RCPADI driver, as follows:

```
RENDER-DISPLAY=RCPADI
MAIN-DISPLAY=RCPADI
MATERIAL-DISPLAY=RCPADI
```

Make sure to remove the **";"** or any spaces that may be present at the beginning of the line. Use any text editor to make the above changes.

Note
Note that it is only necessary for the RENDER-DISPLAY line to be set to RCPADI in order to render at high resolution. If you do not need a high resolution main display screen or if you will not be using the materials editor, you may keep MAIN-DISPLAY and MATERIAL-DISPLAY set to their defaults.

For 3D Studio v3.x and 4.x, use the Autodesk supplied utility to edit the 3DS.SET file. From your 3DS sub-directory, type:

```
3DS VIBCFG
```

to display the configuration screen. Use the **TAB** key to move the cursor among the available parameters. The **ENTER** key allows you to select an available driver. Change the RENDER-DISPLAY, MAIN-DISPLAY and MATERIAL- DISPLAY fields to use RCPADI and then press **SHIFT-TAB** to move to OK. Press **ENTER** and save your changes.

Note
For 3D Studio 1.x, there is no RENDER-DISPLAY line. The corresponding line is **DEFAULT-DISPLAY** and should be changed to read

```
DEFAULT-DISPLAY="RCPADI"
```

The quotes around RCPADI must be used for this version of 3D Studio.

Caution

The MATERIAL-DISPLAY should only be configured for RCPA-DI when BOTH the Display AND Rendering Resolutions of TurboDLD*Classic* are configured for 256 color resolutions. An unpredictable Materials Editor screen will appear if anything other than 256 colors is selected. If you are unsure about the function of the Materials Editor screen, use the Materials Editor as VGA; you should not have any problems at all. Please note too, that the use of the Materials Editor will not give you any more colors to choose from, just higher resolution.

After saving 3DS.SET changes, the final steps in the configuration process configure TurboDLD*Classic* for use with 3D Studio.

Delete (or rename) the 3DADI.CFG file from your 3D Studio sub-directory by typing,

```
DEL 3DADI.CFG
```

Deleting this file will cause 3D Studio to start in its reconfiguration mode. 3DADI.CFG will be recreated after 3D Studio has been reconfigured.

Start the 3D Studio reconfiguration by typing;

```
3DS
```

During the 3D Studio start-up, you are prompted with a series of configuration questions. After each question is answered, TurboDLD*Classic* will display either a Display Screen or a Rendering Screen Configuration menu, depending on whether the 3D Studio question references a display or a rendering option.

To configure the 3D Studio Main Display for your graphics board, from the Display Configuration menu, select a **display graphics board** and then a **display resolution**.

To configure 3D Studio for the Rendering and/or Materials screens, select a **rendering graphics board** (usually the same as the Display board) and a **rendering resolution**. (For detailed information on choosing graphics boards or display and rendering resolutions, See *Chapter 3 - Installing and Configuring*.)

Because RCPADI device drivers, by definition, are combined Display and Rendering devices, during 3D Studio reconfiguration, you may be brought to the TurboDLD*Classic* Configuration menu more than one time each. It is not necessary to select display and/or rendering resolutions a second time. Simply press the **ENTER** key to remove the help screen and then highlight **Save and Exit** to continue to the next question. Also, if FASTACAD.BAT is used instead of PANA3DS.BAT to define the AutoCAD operating environment, the TurboDLD*Classic* Configuration Menu will also appear for RDPADI and DSPADI devices if they are present.

Note
Some 3D Studio Limitations

TurboDLD*Classic* only provides still rendering support for 3D Studio. This driver will not play back rendered FLI or FLC files at resolutions above the 320x200 default of 3D Studio.

If you have a VESA compatible graphics board you may be able to use the RCPVESA.EXP driver that comes with 3D Studio to display animations at high resolution. Refer to your 3D *Studio Installation and Performance Guide* for specific information.

Configuring AVE Render

AutoCAD 12 and 13's AVE Render uses TurboDLD*Classic*'s rendering capabilities to render objects and drawings. If AVE Render has never been configured, when you select the AutoCAD render command, you are required to run through its configuration process. In this case, you are prompted through the following questions and/or menus.

1. **Select a Rendering Display device**. Since TurboDLD*Classic* is a combined display/rendering device, choose item 1, P386 ADI Combined Display/Rendering Driver from the available choices.
2. **Configure the Rendering Graphics Board and Resolution.** Here the TurboDLD*Classic* configuration program will appear on the screen. Press the Return key to continue past the help screen and display the Rendering Configuration menu. Choose **Select Graphics Board/Resolution** to display graphics board and resolution menu selections. Select a rendering graphics board and then a rendering resolution.
3. **Select Render Mode**. The next step is to select the rendering mode for TurboDLD*Classic*. Select either **Render to Viewport** or **Render to Screen** depending on how you wish to view your renderings. Note that in order to render to a viewport you must be using a Display resolution of more than 16 colors. Otherwise, AVE Render will not allow the render to viewport selection.
4. **Select Render Hard Copy Device**. If you are using a rendering hard copy device select your device type from the list of choices. If you are not using a hard copy device, accept the default of **NULL**.

After configuring AVE Render, you will return to the drawing to render the current drawing.

If you have previously configured AVE Render for another driver, you will need to reconfigure. Type **RCONFIG** at the AutoCAD command prompt to manually invoke the Render Configuration Menu. Follow these steps to re- configure AVE Render.

1. **Select option 2, Configure Rendering Device** to choose a new rendering driver. Answer **YES** to the "**Select Different Rendering Device**" question.
2. **Select option 1, P386 Combined Display/Rendering Driver** to choose the type of driver being used.
3. **Configure the Rendering Graphics Board and Resolution.** At this point, the TurboDLD*Classic* configuration program will appear on the screen. Press the Return key to continue past the help screen and display the Rendering Configuration menu. Choose **Select Graphics Board/Resolution** to display graphics board and resolution menu selections. Select a rendering graphics board and then a rendering resolution.
4. **Select Render Mode**. The next step is to select the rendering mode for TurboDLD*Classic*. Select either **Render to Viewport** or **Render to Screen** depending on how you wish to view your renderings. Note that in order to render to a viewport you must be using a Display resolution of at least 256 colors. Otherwise, AVE Render will not allow a render to viewport selection.
5. **Select Exit to the Drawing Editor** from the Render Configuration menu and then type **Y** to keep the changes you've just made. Press the **<F1>** key to change to the graphics screen if necessary.

Autoshade 2.0 with Renderman

To configure AutoShade v2.0 to use TurboDLD*Classic*, first run the **FASTACAD.BAT** file from your TurboDLD*Classic* sub-directory to set the AutoShade environment variables. Next, start AutoShade with **SHADE /R**, which will allow you to reconfigure AutoShade. For the display device, select **P386 Autodesk Device Interface display driver**, and for the rendering display select the **P386 Autodesk Device Interface rendering driver**. If you are running the display and rendering screen on the same monitor (i.e. single monitor), make sure to tell AutoShade this. A single monitor approach will require a redraw of the display screen after a rendering screen.

Once the P386 ADI devices have been selected, the TurboDLD*Classic* configuration menu will be displayed. Follow the same installation steps that were used to select the graphics board and display and rendering resolutions for AutoCAD (see *Chapter 3 - Installing and Configuring*).

Appendix A Support Information

Just the Fax, Ma'am - (XXX) XXX-XXXX ----------

When possible, please send problem reports by fax, on-line message, or regular mail, including the information requested below. Our fax number is (XXX) XXX-XXXX. You can also send electronic mail to:

Tech_Support@XXXXX.com

via INTERNET or CompuServe. It is much easier and more productive for us to work with a page of information or an on-line message than it is to try and get the information from you during a voice phone call over a busy, noisy line. Once we get your faxed request, we will call you or fax you back with an answer or with additional questions to learn more about the problem. Our voice number is (XXX) XXX-XXXX. Please send mail to:

XXXXX Inc. ## XXXXXXX Street XXXX, XX ##### Attn: Technical Support. Please put the following information in a faxed request, or have this information ready for a phone request:

- Your **name**, and the **serial** and **version** number of your copy of TurboDLD*Classic*.
- A list of what appears on the screen when you type "**SET**" at a DOS prompt. (Or print out a screen shot and attach it to the fax).
- Print-outs of your **DLDSETUP.DAT** and **FASTACAD.BAT** (R10/11) files.
- Print-outs of your **CONFIG.SYS**, **AUTOEXEC.BAT** and any AutoCAD start-up files.
- A complete **description** of what the problem was and how many times, and under which circumstances, it has occurred. We can help you most efficiently if you can recreate the problem with some consistency.

Many of the problems which may occur in setting up and using TurboDLD*Classic* are not really problems with the driver; instead, the problems occur in installing and getting access to additional memory for the driver.

Memory management programs, such as QEMM, 386MAX, and EMM386, can cause memory conflicts on certain graphics platforms. In order to avoid these conflicts, it may be necessary to exclude certain memory regions from being used by the memory manager.

Also, if you are using a memory manager, avoid using any switches or utilities of the manager that may re-map upper memory blocks in order to load as many programs and TSR's "high" as possible (i.e. QEMM's "Stealth" mode or EMM386's HIGHSCAN). These features can make it difficult or impossible for TurboDLD*Classic* utility programs to locate certain video BIOS information. If you have memory problems, carefully check both the system documentation and the memory manager documentation before contacting Technical Support. System memory problems probably won't be solvable by us.

We aren't able to fix mis-set jumpers and incorrect configuration options over the phone. Please be sure that all hardware is installed and operating correctly before assuming that the driver is at fault. Carefully check your hardware documentation for the correct switch and jumper settings.

Our support policy is to offer free phone and fax support to our users. This policy is basically unlimited; we want our users to enjoy TurboDLD*Classic* and get the most out of it. We are especially appreciative if you can help us identify a problem in using the driver or the documentation which we can fix in a subsequent upgrade, benefiting all customers. We reserve the option to limit, withhold, or charge for support.

Appendix B TurboDLD Classic Commands

This chapter alphabetically lists all TurboDLD*Classic* commands providing a quick reference. For complete information regarding the use of these commands, see *Chapter 5 - TurboDLD Features*.

DLDBIGPIC	Calls up the Big Picture - use digitizer clicks to define an area and zoom into it.
DLDBPDIM	Allows resizing and repositioning of the Big Picture.
DLDBPFREEZE	Toggles the Big Picture Zoom Mode between Float and Freeze modes.
DLDBPHILIGHT	Selects the highlight mode used to display the Big Picture.
DLDBPMODE	Toggles Big Picture between Zoom and Pan modes.
DLDBPREFRSH	Enables and Disables the Big Picture automatic update mode.
DLDBPSTATIC	Toggles the Big Picture to stay on the screen at all times.
DLDCOLOR	Invokes CustomColorsTM, Panacea's color configuration program.
DLDCOMPACT	Forces a manual clean-up of Display List memory returning any unused memory back to the AutoCAD memory pool.
DLDDLIST	Toggles the display list function of TurboDLD*Classic* on and off.
DLDECHO	Toggles internal TurboDLD*Classic* command echoing at the AutoCAD command line.
DLDHELP	Provides a list of TurboDLD*Classic* commands with one-line description of each, within AutoCAD. It's recommended that you flip to the text screen to view the output.
DLDREFRESH	Refreshes the Big Picture to reflect the most current changes to a drawing.
DLDSTAT	Displays a listing of the current TurboDLD*Classic* parameters. A flip to the graphics screen is recommended for this command.
DLDUSAGE	Gives a to-the-byte breakdown of how memory is being used for display list processing.
DLDVER	Displays the TurboDLD*Classic* version, serial number and registered user's name at the AutoCAD command line.
PD, ZD, ZT	Included in TurboDLD*Classic* for backwards compatibility with MACH-12. These commands will invoke the Big Picture.
ZI, ZO	ZI - Zoom In and ZO - Zoom Out will zoom the current viewport in or out incrementally.
ZA, ZP	ZA - Zoom All and ZP-Zoom Previous are passed directly to AutoCAD.
ZW	Invokes an AutoCAD Zoom Window command.
CD	Returns any unused DLM memory to the VMM.

Appendix C Changes From Previous Versions

This Appendix describes the major program changes in this version compared to earlier versions of TurboDLD and MACH-12.

Changes From TurboDLD v1.0

BP Cache

The memory cache for TurboDLD commands has been removed. A separate cache is no longer necessary due to the optimization of the driver's Display List performance. The effect of the BP Cache is now achieved internally, for all graphics platforms. This affects the BP Cache command DLDBPCACHE and the Configuration menu option to enable BP Cache.

Fast/Incremental REGEN

The Fast/Incremental REGEN mode option has also been removed from TurboDLD. TurboDLD v2.0 now offers a fast, incremental REGEN eliminating the need for a toggle command.

Changes From MACH-12

During the integration of MACH-12 and TurboDLD, we found that various commands were duplicated. Because of this overlap, it was necessary to combine commands where it was possible and eliminate them where it was not. The following have been removed or are different from earlier MACH- 12 versions.

ALLON
All Color Toggle commands
CA - Cancel
CCP - Continuous Center Pan
RD - Quick Redraw
FS - Fill Enable/Disable
SCP - Single Center Pan

Anti-Aliasing

Because only a limited number of graphics boards (those with the Edsun CEG RAMDAC in 256-color modes and Sierra HiColor RAMDAC in 32k-color modes) support anti-aliasing, TurboDLD*Classic* does not include this MACH-12 feature. We have chosen to offer speed instead of a limited, less useful feature. The anti-aliasing commands

AP - Anti-Aliasing On
AO - Anti-Aliasing On
AF - Anti-Aliasing Off
AS - Enable Split Screen Anti-Aliasing

are no longer available.

Display-List (DL) Compression

Is now done automatically, internally, by TurboDLD*Classic* as needed.

Bird's Eye redraw ratio

TurboDLD*Classic*'s DLDBPREFRSH command functions the same as this MACH-12 configuration setting. The difference however, is that the MACH-12 command was configurable. DLDBPREFRSH is not, it is either enabled or disabled.

Flip Screen Save

This feature is no longer necessary and has been removed.

Set Custom Command Names

Command names cannot be changed through TurboDLD*Classic* or the driver configuration utility. If you wish to change, or Alias, command names, you may do so by editing the ACAD.PGP file in your AutoCAD SUPPORT sub- directory.

Set Command Names Sent to Foreign AutoCAD

This MACH-12 setting is no longer necessary. TurboDLD*Classic* supports English versions of AutoCAD. For use with foreign AutoCAD versions, TurboDLD*Classic* supports .BDF fonts (supplied by Autodesk) for menus and pull-downs. This allows support for foreign versions of AutoCAD via font selection.

THIS PAGE LEFT BLANK

S-MOS SYSTEMS
A Seiko Epson Affiliate

SPC8110F0A LOCAL BUS LCD/CRT VGA CONTROLLER

SDU8110B0C Rev. 1.0 VL-Bus Evaluation Board

Drawing Office No. X07-AN-017-02

Copyright © 1996 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

TABLE OF CONTENTS

1 SDU8110B0C Rev. 1.0 Evaluation Board User Manual 7
1.1 Features 7
1.2 Installation and Configuration 8
1.2.1 DIP Switch Configuration Settings Table 8
1.2.2 Hard-Wired Configuration Inputs Table 8
1.2.3 DRAM Refresh Rate Select Table 9
1.2.4 LCD Panel Configuration Table 9
1.2.5 MDA0 PLL1(PCLK) Select 10
1.2.6 MDA1 LCDBACK Polarity Select 10
1.2.7 MDA2 LCDLOGIC Polarity Select 10
1.2.8 MDA3 LCDBIAS Polarity Select 10
1.2.9 MDA4 Bus Interface 10
1.2.10 MDA5 PLL2(MCLK) Select 10
1.2.11 MDA6-MDA7 Test Mode Inputs 10
1.2.12 MDA8-MDA12 LCD Panel Configuration Inputs 10
1.2.13 MDA13-MDA14 DRAM Refresh Rate Select 11
1.2.14 MDA15 Video DRAM Configuration Select 11
2 LCD Signal Connector Pinouts 12
3 Technical Description 13
3.1 32-Bit VL-Bus Support 13
3.2 ISA Bus Interface 13
3.3 1024K DRAM Support 13
3.4 Monochrome LCD Support 13
3.5 Color LCD Support 13
3.6 TFT Support 13
3.7 LCD Panel Control Signals 14
3.8 CRT Support 14
3.9 DDC2B Support 14
3.10 VGA BIOS Support 14
3.11 Power Save Modes 14
3.12 Power Down Clock Support 14
3.13 External REFCLK Support 14
3.14 Positive Power Supply 14
3.15 Negative Power Supply 15
3.16 Fixed 3.3 Volt Power Supply 15
3.17 PCB Layout Considerations 15
3.18 Schematic Notes 16
Appendix A Parts List 17
Appendix B SDU8110B0C Schematic Diagrams 18

THIS PAGE LEFT BLANK

LIST OF TABLES

Table 1-1 DIP Switch Configuration Settings 8
Table 1-2 Hard-Wired Configuration Inputs 8
Table 1-3 DRAM Refresh Rate Select 9
Table 1-4 LCD Panel Configuration 9
Table 2-1 LCD Signal Connector Pinout 12

LIST OF FIGURES

Figure 1 : SDU8110B0C Rev. 1.0 Schematic Diagram (1 of 7) 18
Figure 2 : SDU8110B0C Rev. 1.0 Schematic Diagram (2 of 7) 19
Figure 3 : SDU8110B0C Rev. 1.0 Schematic Diagram (3 of 7) 20
Figure 4 : SDU8110B0C Rev. 1.0 Schematic Diagram (4 of 7) 21
Figure 5 : SDU8110B0C Rev. 1.0 Schematic Diagram (5 of 7) 22
Figure 6 : SDU8110B0C Rev. 1.0 Schematic Diagram (6 of 7) 23
Figure 7 : SDU8110B0C Rev. 1.0 Schematic Diagram (7 of 7) 24

THIS PAGE LEFT BLANK

1 SDU8110B0C REV. 1.0 EVALUATION BOARD USER MANUAL

This manual reflects the use of the SDU8110B0C Rev. 1.0 evaluation board in conjunction with the SPC8110F0A VGA Controller chip while operating in a VL-Bus environment. All appropriate components are surface-mount to reduce cost and minimize board size. The SDU8110B0C Rev. 1.0 will operate as a stand-alone video adapter card with on-board video BIOS support. Refer to the SPC8110F0A Functional Specification, Drawing Office No. X07-SP-001-xx for further details.

1.1 Features

- 208 pin QFP8-208 package
- 32-bit VL-Bus support
- 3.3V Core voltage and 5.0V I/O voltage operation
- SMD technology for all appropriate devices
- Color/Monochrome STN LCD Panel support
- TFT Panel support
- CRT support
- DDC2B support
- Limited ISA support for BIOS EPROM
- On-board video BIOS with additional support logic provided
- Plug and Play BIOS compatibility
- Power Down Clock support
- External REFCLK support
- 1024K DRAM support (2 x 256K x 16, symmetrical)
- Support for Software Power Save Modes
- On-board +35 V adjustable power supply (VDDH for color panels)
- On-board -23V adjustable power supply (VLCD for monochrome panels)
- On-board fixed 3.3V Core Voltage (SOGVDD, AVDD)
- LCD Backlight/Logic/Bias Control

1.2 Installation and Configuration

The SDU8110B0C Rev. 1.0 directly supports the 32-bit VL-Bus architecture, while providing limited ISA Bus support for the on-board Video BIOS. The SPC8110F0A has 16 configurable inputs (MDA[15...0]) which are read on power-up. For the purpose of this design, most of the configuration inputs have been hard-wired and are set to power-up as appropriate. Due to the variety of LCD panels supported, a five position DIP Switch has been provided for different panel configuration. Refer to LCD Panel and DIP Switch Configuration Table below for details.

1.2.1 DIP Switch Configuration Settings Table

Table 1-1 DIP Switch Configuration Settings

Switch	Signal	Closed	Open
SW1-1	MDA8	LCD Panel Config bit 0	See LCD Panel Configuration Table
SW1-2	MDA9	LCD Panel Config bit 1	
SW1-3	MDA10	LCD Panel Config bit 2	
SW1-4	MDA11	LCD Panel Config bit 3	
SW1-5	MDA12	LCD Panel Config bit 4	

0 = Open, 1 = Closed

1.2.2 Hard-Wired Configuration Inputs Table

Table 1-2 Hard-Wired Configuration Inputs

	Signal	Closed	Open
	MDA0	PLL1 By-pass Mode (PCLK)	PLL1 enabled (PCLK)
	MDA1	LCDBACK Polarity (Active high)	LCDBACK Polarity (Active low)
	MDA2	LCDLOGIC Polarity (Active high)	LCDLOGIC Polarity (Active low)
	MDA3	LCDBIAS Polarity (Active high)	LCDBIAS Polarity (Active low)
	MDA4	BUS I/F Select (PCI Bus)	BUS I/F Select (32-bit VL-Bus)
	MDA5	PLL2 By-pass Mode (MCLK)	PLL2 enabled (MCLK)
	MDA6	RESERVED for Test Mode use only	RESERVED for Test Mode use only
	MDA7	RESERVED for Test Mode use only	RESERVED for Test Mode use only
	MDA13	DRAM Refresh Rate bit 0	See DRAM Refresh Rate Select Table
	MDA14	DRAM Refresh Rate bit 1	
	MDA15	DRAM type (2 CAS) (symmetrical)	DRAM type (2 WE) (asymmetrical)

0 = Open, 1 = Closed

= Hard-Wired board configuration

1.2.3 DRAM Refresh Rate Select Table

Table 1-3 DRAM Refresh Rate Select

MD14	MD13	512 cycles	1024 cycles
0	0	8 msec	16 msec
0	1	64 msec	128 msec
1	0	128 msec	256 msec
1	1	Self Refresh	

0 = Open, 1 = Closed

= Hard-wired Configuration

1.2.4 LCD Panel Configuration Table

The BIOS configuration for different LCD panels is done through the state of the MDA lines. The following tables describes the settings and the associated panel type.

Table 1-4 LCD Panel Configuration

Bit 4 (MDA12)	Bit 3 (MDA11)	Bit 2 (MDA10)	Bit 1 (MDA9)	Bit 0 (MDA8)	Resolution	Mode	Bits	Comments
1	1	1	1	1	640x480	Dual	8	Monochrome
1	1	1	1	0	640x480	Single	8	Monochrome
1	1	1	0	1	640x200	Single	4	Monochrome
1	1	1	0	0	320x240	Single	4	Monochrome
1	1	0	1	1	640x480	Dual	8	Color STN / 2 shift clocks
1	1	0	1	0	640x480	Single	8	Color STN
1	1	0	0	1	640x480	Dual	16	Color STN
1	1	0	0	0	640x480	Single	16	Color STN
1	0	1	1	1	640x480	Dual	6	Color (InFocus)
1	0	1	1	0	640x480	TFT	18	TFT
1	0	1	0	1	640x480	TFT	12	TFT
1	0	1	0	0	640x480	TFT	9	TFT
1	0	0	1	1	640x480	TFT	9	MODULATED
1	0	0	1	0	Half Size	Single	8	Optrex panel only
1	0	0	0	1	640x480	Dual	8	Color STN / 1 shift clock
1	0	0	0	0	OEM Config			
0	1	1	1	1	800x600	Dual	16	Color STN
0	1	1	1	0	Reserved			
through								
0	0	0	0	0				

Note
OEM Configuration options allows for very specific resolution/timing criteria to be programmed for a given panel. Refer to SPC8110F0A Functional Specification, Drawing Office No. X07-SP-001-xx for further details.

1.2.5 MDA0 PLL1(PCLK) Select

MDA0 configuration input selects between on-chip internal PLL or external source for PCLK. MDA0 input is left open to support on-chip internal PLL for PCLK. For the purpose of this design, External PCLK source is not supported.

1.2.6 MDA1 LCDBACK Polarity Select

MDA1 configuration input selects between active high or active low polarity of LCDBACK signal. MDA1 input is left open to select active low polarity of LCDBACK signal.

1.2.7 MDA2 LCDLOGIC Polarity Select

MDA2 configuration input selects between active high or active low polarity of LCDLOGIC signal. MDA2 input is left open to select active low polarity of LCDLOGIC signal.

1.2.8 MDA3 LCDBIAS Polarity Select

MDA3 configuration input selects between active high or active low polarity of LCDBIAS signal. MDA3 input is left open to select active low polarity of LCDBIAS signal.

1.2.9 MDA4 Bus Interface

MDA4 configuration input selects between 32-bit VL-Bus or PCI-Bus interface. MDA4 input is left open to select 32-bit VL-Bus interface support from SPC8110F0A. For the purpose of this design, PCI-Bus interface is not supported

1.2.10 MDA5 PLL2(MCLK) Select

MDA5 configuration input selects between on-chip internal PLL or external source for MCLK. MDA5 input is left open to support on-chip internal PLL for PCLK. For the purpose of this design, External MCLK source is not supported.

1.2.11 MDA6-MDA7 Test Mode Inputs

MDA6-MDA7 configuration inputs are reserved for test mode use purposes only. For the purpose of this design, these inputs are not used and left open.

1.2.12 MDA8-MDA12 LCD Panel Configuration Inputs

MDA8-MDA12 configuration inputs are used to configure the size and type of LCD panel being used. The state of these inputs is selected through the five position DIP switch. Refer to *LCD Panel Configuration Table on page 9* for specific settings required for specific LCD Panel.

1.2.13 MDA13-MDA14 DRAM Refresh Rate Select

MDA13 and MDA14 configuration inputs are bit 0 and bit 1 of DRAM Refresh Rate Select source. During Power Save Modes, provision must be made to refresh the video DRAM. As shown in the *DRAM Refresh Rate Select Table on page 9*, there are four options to refresh the video DRAM. Self-refresh DRAM is used for this design. Therefore, MDA13 and MDA14 inputs are set to logic high to select the Self Refresh option according to the table.

1.2.14 MDA15 Video DRAM Configuration Select

MDA15 configuration input selects between symmetrical or asymmetrical type DRAM support. The SDU8110B0C board is assembled with two 5V, 256Kx16 configuration, symmetrical architecture , Self Refresh, SMD, SOJ package DRAMs. For this design, MDA15 input is set to logic high to support symmetrical architecture DRAM.

2 LCD SIGNAL CONNECTOR PINOUTS

Table 2-1 LCD Signal Connector Pinout

SPC8110Pin Name	LCD Connector Pin No.	TFT			Color STN LCD				Mono STN LCD	
		18-bit color	12-bit color	9-bit color	16-bit	8-bit Half panel	8-bit	6-bit	8-bit	4-bit
LD0	1	R4	R2	R1	LD0		LD0	LB	LD0	
LD1	3	R5	R3	R2	LD1		LD1	LG	LD1	
LD2	5	G0			LD2		LD2	LR	LD2	
LD3	7	G1			LD3		LD3		LD3	
LD4	9	R0			LD4				UD0	UD0
LD5	11	R1			LD5				UD1	UD1
LD6	13	R2	R0		LD6				UD2	UD2
LD7	15	R3	R1	R0	LD7				UD3	UD3
UD0	17	B0			UD0	UD0	UD0	UB		
UD1	19	B1			UD1	UD1	UD1	UG		
UD2	21	B2	B0		UD2	UD2	UD2	UR		
UD3	23	B3	B1	B0	UD3	UD3	UD3			
UD4	25	G2	G0		UD4	UD4				
UD5	27	G3	G1	G0	UD5	UD5				
UD6	29	G4	G2	G1	UD6	UD6				
UD7	31	G5	G3	G2	UD7	UD7				
TFTB4	33	B4	B2	B1						
TFTB5	35	B5	B3	B2						
XSCL	37				XSCL	XSCL	XSCL	XSCL	XSCL	XSCL
XSCLU	39	CLK	CLK	CLK			XSCLU			
LP	40	HS	HS	HS	LP	LP	LP	LP	LP	LP
YD	38	VS	VS	VS	YD	YD	YD	YD	YD	YD
WF	36	DE	DE	DE	WF	WF	WF	WF/DE	WF	WF
VDDH	34				VDDH		VDDH			
VLCD	32								VLCD	VLCD
LCD BIAS	30	LCD BIAS	LCD BIAS	LCD BIAS	LCD BIAS	LCD BIAS	LCD BIAS	LCD BIAS	LCD BIAS	LCD BIAS
LCD BACK	28	LCD BACK	LCD BACK	LCD BACK	LCD BACK	LCD BACK	LCD BACK	LCD BACK		
LCD LOGIC	26	LCD LOGIC	LCD LOGIC	LCD LOGIC	LCD LOGIC	LCD LOGIC	LCD LOGIC	LCD LOGIC	LCD LOGIC	LCD LOGIC
+5 V	24	+5 V	+5 V	+5 V	+5 V	+5 V	+5 V	+5 V	+5 V	+5 V
+12 V	22	+12 V	+12 V	+12 V	+12 V	+12 V	+12 V	+12 V	+12 V	+12 V
GRND	20-2	GRND	GRND	GRND	GRND	GRND	GRND	GRND	GRND	GRND

3 TECHNICAL DESCRIPTION

3.1 32-Bit VL-Bus Support

This board directly supports the 32-bit VL-Bus and is intended to be used as a VL-Bus Target add-in-board. All SPC8110F0A related electrical connections come from the standard VL-Bus connector, while the in-line ISA-Bus connectors are used to provide the Video BIOS interface. The SPC8110F0A can be configured as 3.3V or 5.0V for the 32-bit VL-Bus I/O interface. For the purpose of this design, only 5.0V I/O VL-Bus interface is supported.

3.2 ISA Bus Interface

Limited support for the 8-bit ISA-Bus provides an on-board video BIOS. Only those signals necessary for this function are available.

3.3 1024K DRAM Support

A two chip 1024K solution is supported using 2 x 256K x 16 DRAM. Only symmetrical DRAM in an SOJ package is supported.

3.4 Monochrome LCD Support

The SPC8110F0A supports 4- and 8-bit Dual and Single monochrome STN LCD panels. All the necessary signals are provided on the 40-pin ribbon cable connector (J5). The interface signals are alternated with grounds on the cable to reduce cross talk and noise related problems.

Refer to *LCD Panel Configuration Table and LCD Signal Connector Pinouts* above for your specific settings.

3.5 Color LCD Support

The SPC8110F0A also supports 8- and 16-bit Dual and Single color STN LCD panels. All the necessary signals can also be found on the 40-pin ribbon cable connector (J5). The interface signals are alternated with grounds on the cable to reduce cross talk and noise related problems.

Refer to *LCD Panel Configuration Table and LCD Signal Connector Pinouts* above for your specific settings

3.6 TFT Support

The SPC8110F0A also supports 9-, 12-, and 18-bit TFT panels. All the necessary signals can also be found on the 40-pin ribbon cable connector (J5). The interface signals are alternated with grounds on the cable to reduce cross talk and noise related problems.

Refer to *LCD Panel Configuration Table and LCD Signal Connector Pinouts* above for your specific settings

3.7 LCD Panel Control Signals

Control signals are provided on the connector (LCDLOGIC, LCDBIAS, LCDBACK). These signals are low current direct outputs from the SPC8110F0A and are intended to be used in-conjunction with external circuitry to provide higher current switching.

3.8 CRT Support

The SPC8110F0A has a built-in RAMDAC for direct support of CRT monitors. All standard VGA resolutions as well as extended modes up to 1024 x 768 are supported.

3.9 DDC2B Support

The SDU8110B0C Rev. 1.0 evaluation board is capable of directly supporting the VESA DDC2B monitor standard. This board supports hardware DDC2B and the BIOS supports software layer VBE/DDC version 1.0. For further information on DDC2B contact the Video Electronics Standards Association (VESA).

3.10 VGA BIOS Support

A 32K EPROM contains the VGA BIOS. Support logic consists of an Address Decoder (C000H-C7FFH), and a Data Buffer operating from the ISA-Bus. With the BIOS installed the board will operate as a stand-alone Video adapter. This BIOS is Plug and Play compatible

3.11 Power Save Modes

The SPC8110F0A supports 2 software and 2 hardware Power Save Modes. A utility program is supplied to control the software modes. The software modes are controlled by directly writing the SPC8110F0A associated internal registers. For the purpose this design, hardware Power Save Modes are not supported.

3.12 Power Down Clock Support

A 32.768 Khz oscillator is provided for connection to the PSCLK signal (pin 153). This clock source is required to provide the DRAM refresh cycles.

3.13 External REFCLK Support

The SPC8110F0A requires 14.318Mhz external frequency source for use by the internal PLL. An on-board SMD package 14.318Mhz oscillator is used to provide this function.

3.14 Positive Power Supply

The majority of Color STN LCD panels require a positive power supply to provide between +23 V and +40 V (I_{out}=45mA) (VDDH). For ease of implementation, such a power supply has been provided as an integral part of this design. The signal VDDH can be found on J5 (pin 34) and its output voltage can be controlled by R21 and control signal LCDBIAS.

R21 (200K potentiometer) can be adjusted to provide an output voltage from +23 V to +40 V.

LCDBIAS is a control signal provided by the SPC8110F0A to enable/disable the VDDH power supply. This signal guarantees correct power sequencing and prohibits subsequent panel damage.

Determine the panel's specific power requirements and set R21 accordingly before connecting the panel.

Note
LCDBIAS must be set to Active low polarity before power-up.

3.15 Negative Power Supply

The majority of Monochrome STN LCD panels require a negative power supply to provide between -18 V and -23 V (I_{out}=45mA) (VLCD). For ease of implementation, such a power supply has been provided as an integral part of this design. The signal VLCD can be found on J5 (pin 32) and its output voltage can be controlled by R3 and control signal LCDBIAS.

R3 (100K potentiometer) can be adjusted to provide an output voltage from -18 V to -23 V.

LCDBIAS is a control signal provided by the SPC8110F0A which is used to enable/disable the VLCD power supply. This signal guarantees correct power sequencing and prohibits subsequent panel damage.

Determine the panel's specific power requirements and set R3 accordingly before connecting the panel.

Note
LCDBIAS must be set to Active low polarity before power-up.

3.16 Fixed 3.3 Volt Power Supply

An independent 3.3V power supply at 0.5A (maximum) has been provided as an integral part of this design using National LP2960AIN-3.3 Micropower 0.5A Low-Dropout Voltage Regulator for the core and analog logic sections (SOGVDD, MCLVDD, AVDD) of SPC8110F0A.

3.17 PCB Layout Considerations

The physical board dimensions are determined by the VL-BUS 2.0p PROPOSAL rev 0.93p.

The CRT connector is available on the outside edge of the card, facing the back of the computer. To provide low signal noise as well as reduced capacitive loading, the LCD Connector is located near the SPC8110F0A.

The VL-Bus specification specifies a maximum allowable trace length from the VL-Bus connector to any component. This evaluation board supports the VL-Bus as in-line with the standard AT-Bus connectors. Given the above trace length restriction and general location of the VL-Bus connector, the SPC8110F0A controller is located near the back of the evaluation board (furthest away from the back of the computer).

The board has 4 layers to accommodate various power/ground requirements.

3.18 Schematic Notes

The evaluation boards may have been modified and therefore the following schematic diagrams may not reflect the actual implimentation. Please request updated information before starting any hardware design.

Note
External PLL Loop Values. For revision 1.0 of the SDU8110B0C evaluation board capacitors C31 and C32 are 0.1 uF, R14 and R15 are 70 ohms.
DDC2B Support. A 2.2K ohm pull-up resistor has been added on the GENIO0 line (pin 161) of the SPC8110F0A (U1) for DDC2B support.

APPENDIX A PARTS LIST

Item #	Qty/ board	Designation	Part Value	Description
1	24	C1- C24	0.01uF	0.01uF, 1206 pckg
2	2	C25, C27	47uF	47uF / 15V Tantalum D-SIZE
3	4	C26, C28, C29, C38	10uF	10uF / 25V Tantalum D-SIZE
4	1	C30	1uF	1uF Tantalum A-SIZE
5	3	C31,C32,C39	0.1uF	0.1uF, 1206 pckg
6	3	C33-C35	10uF / 63V	Electrolytic / Radial (LXF63VB10RM5X11LL)
7	3	C36,C37,C41	56uF/35V	LXF35VB56RM6X11LL
8	1	C40	33uF	33uF / 10V Tantalum D-SIZE
9	6	D1-D6	1N4148	Signal Diode / PTH
10	1	J5	CON40A	Shrouded Header 40 pin Dual-row center-key , PTH
11	1	J6	PS/2 CONNECTOR	Assman A-HDF 15 A KG/T or equivalent
12	5	L1-L5	Ferrite Bead	Fair-rite 2743001111 PTH
13	1	L6	1uH	Dale Inductor IM-4-1.0uH PTH
14	1	Q1	2N3905	PNP Signal Transistor TO-92 PTH
15	1	Q2	2N3903	NPN Signal Transistor TO-92 PTH
16	4	R1, R4-R6	10K	10K Ohm/1206/5%
17	1	R2	10K	10K 9-pin SIP 4609X-101-103 PTH
18	1	R3	100K	100K Ohm Trim POT Bourns 3386W-1-104 (or eqiuv)
19	2	R7, R8	39	39 Ohm /1206 / 5%
20	3	R9-R11	150	150 Ohm / 1206 / 1%
21	1	R12	20.6	20.6 Ohm / 1206 / 1%
22	1	R13	16.7	16.7 Ohm / 1206 / 1%
23	2	R14, R15	240	240 Ohm / 1206 / 1%
24	2	R16, R17	1K	1K Ohm /1206 / 5%
25	2	R18, R19	100K	100K Ohm /1206 / 5%
26	1	R20	470K	470K Ohm / PTH / .25W / 5%
27	1	R21	200K	200K Ohm Trim POT Bourns 3386W-1-204 (or equiv)
28	1	R22	14K	14K Ohm / PTH / .25W / 5%
29	1	S1	SW-DIP-5	DIP Switch 5-position
30	1	U1	SPC8110F0A	QFP8-208/ 208 pin S-MOS will supply
31	2	U2, U3	uPD42S4260LE-70	256Kx16 , Self-Refresh, Symmetrical DRAM
32	1	U4	27C256	ST M27C256B-15FI 28 pin DIP/ socket
33	1	U5	74LS244	IC 74LS244 Buffer, DW020 pckg.
34	1	U6	74LS688	IC 74LS688 Comparator, DW020 pckg.
35	1	U7	ML13FAD-32.768Khz	32.768Khz Oscillator 14-pin DIP pckg.
36	1	U8	LM334Z	TO-92 pckg.
37	1	U9	RD-0412	XENTEK - Positive Power Supply
38	1	U10	EPN001	XENTEK - Negative Power Supply
39	1	U11	LP2960AIN-3.3	National 3.3V FIXED Voltage Regulator N16G 16-pin DIP pckg.
40	1	U12	TCO-711JTC-14.31818Mhz	14.31818 Mhz Oscillator 4-pin SMO-J pckg.

APPENDIX B SDU8110B0C SCHEMATIC DIAGRAMS

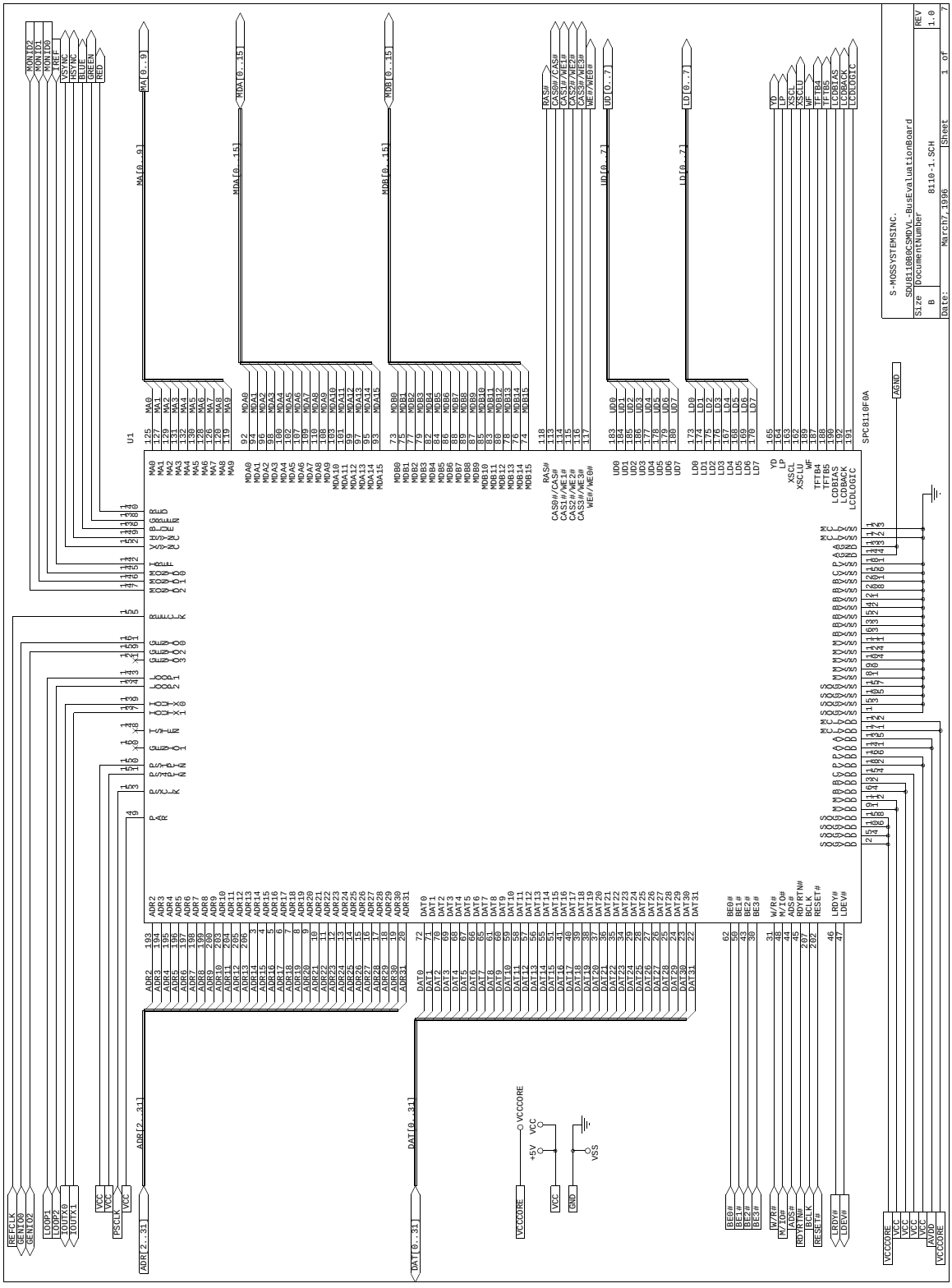


Figure 1 : SDU8110B0C Rev. 1.0 Schematic Diagram (1 of 7)

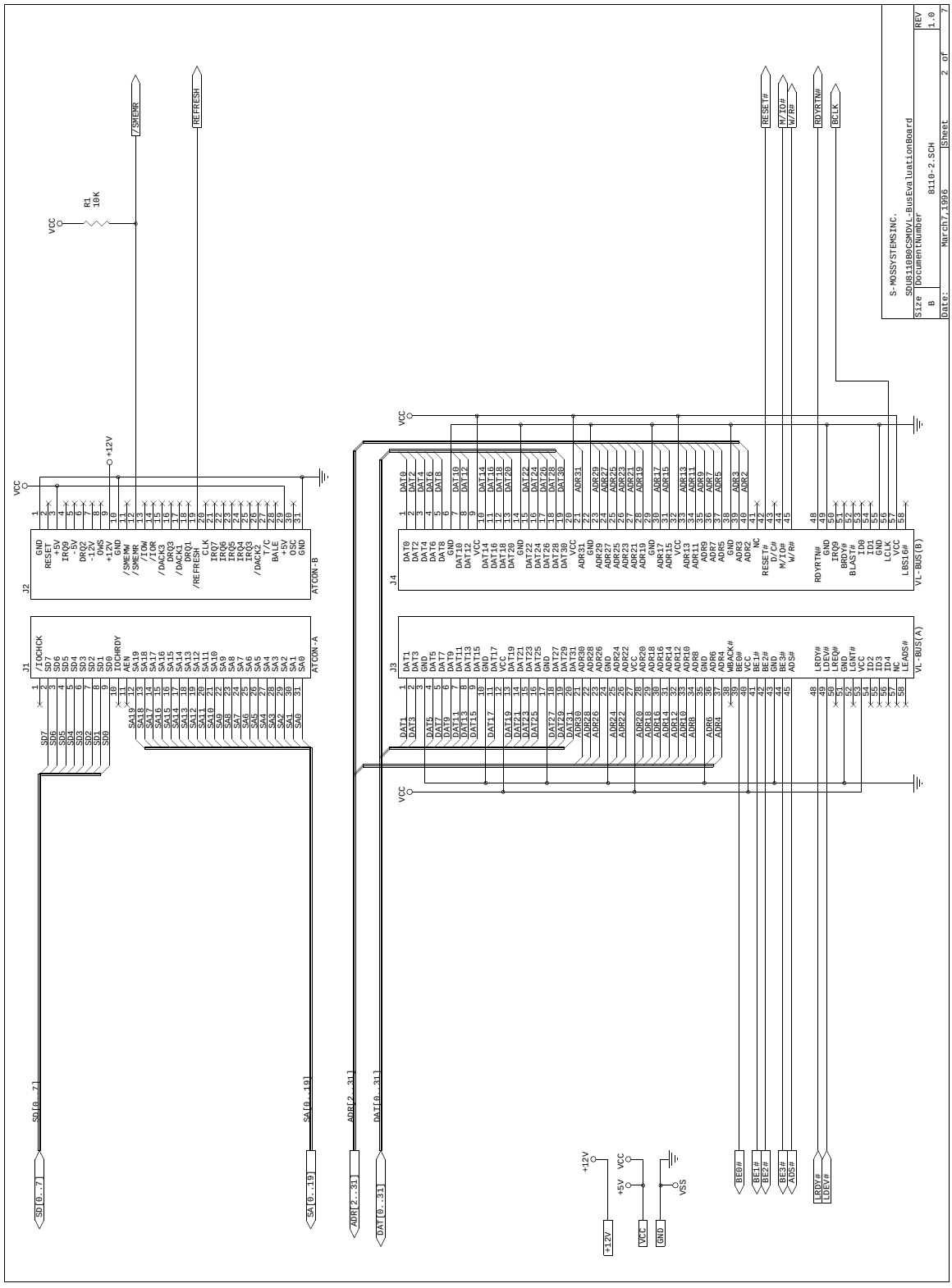


Figure 2 : SDU8110B0C Rev. 1.0 Schematic Diagram (2 of 7)

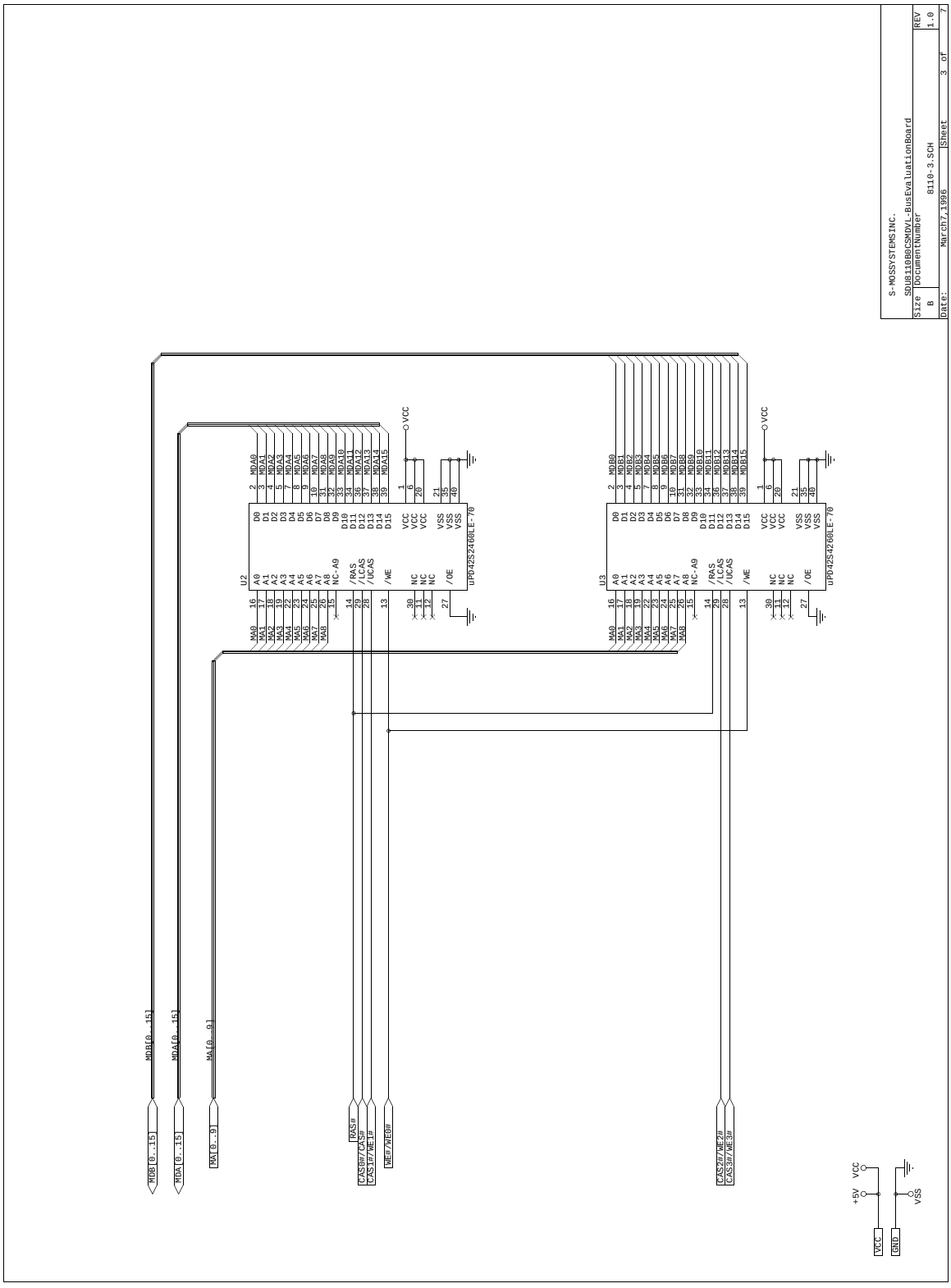


Figure 3 : SDU8110B0C Rev. 1.0 Schematic Diagram (3 of 7)

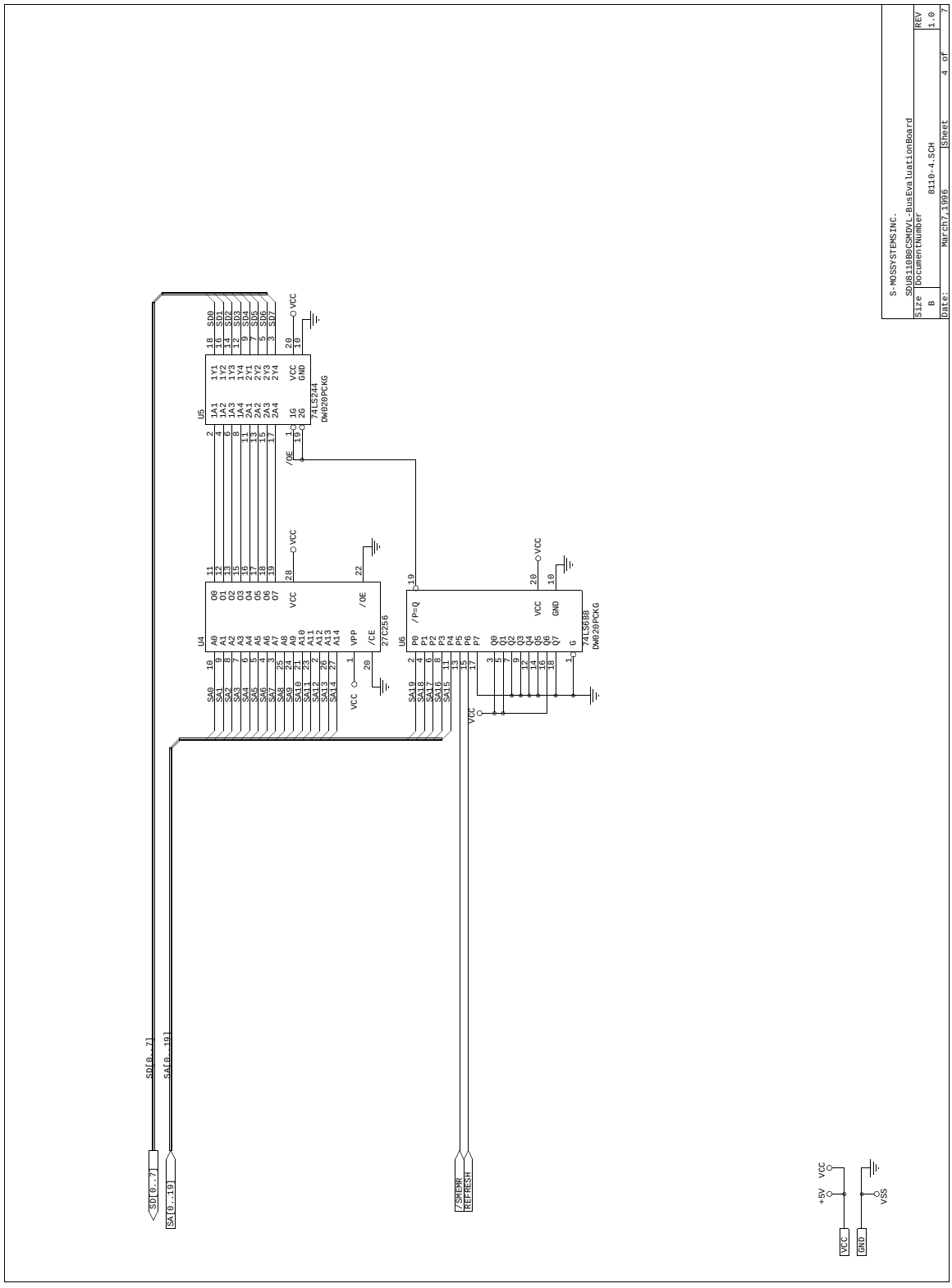


Figure 4 : SDU8110B0C Rev. 1.0 Schematic Diagram (4 of 7)

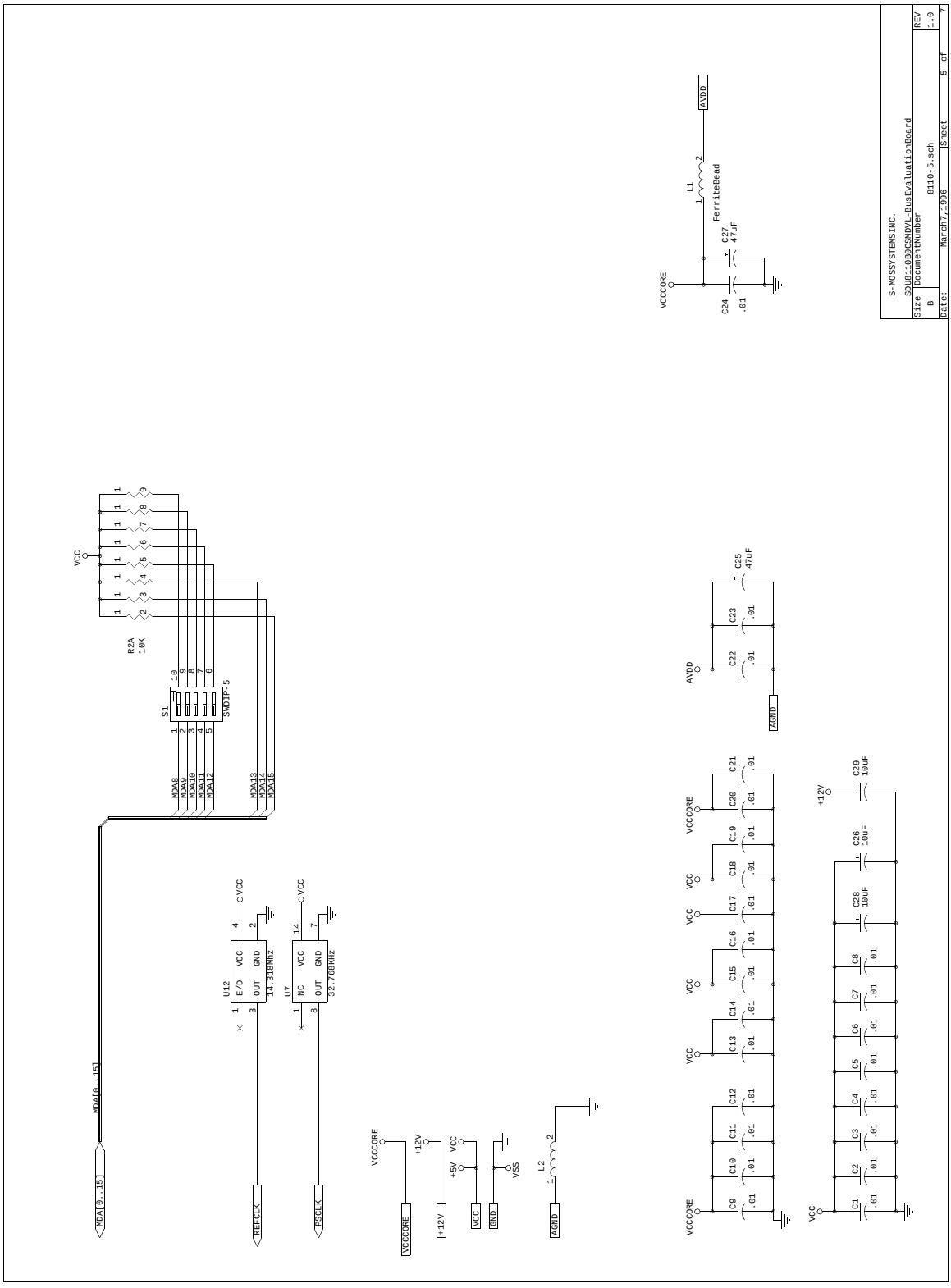


Figure 5 : SDU8110B0C Rev. 1.0 Schematic Diagram (5 of 7)

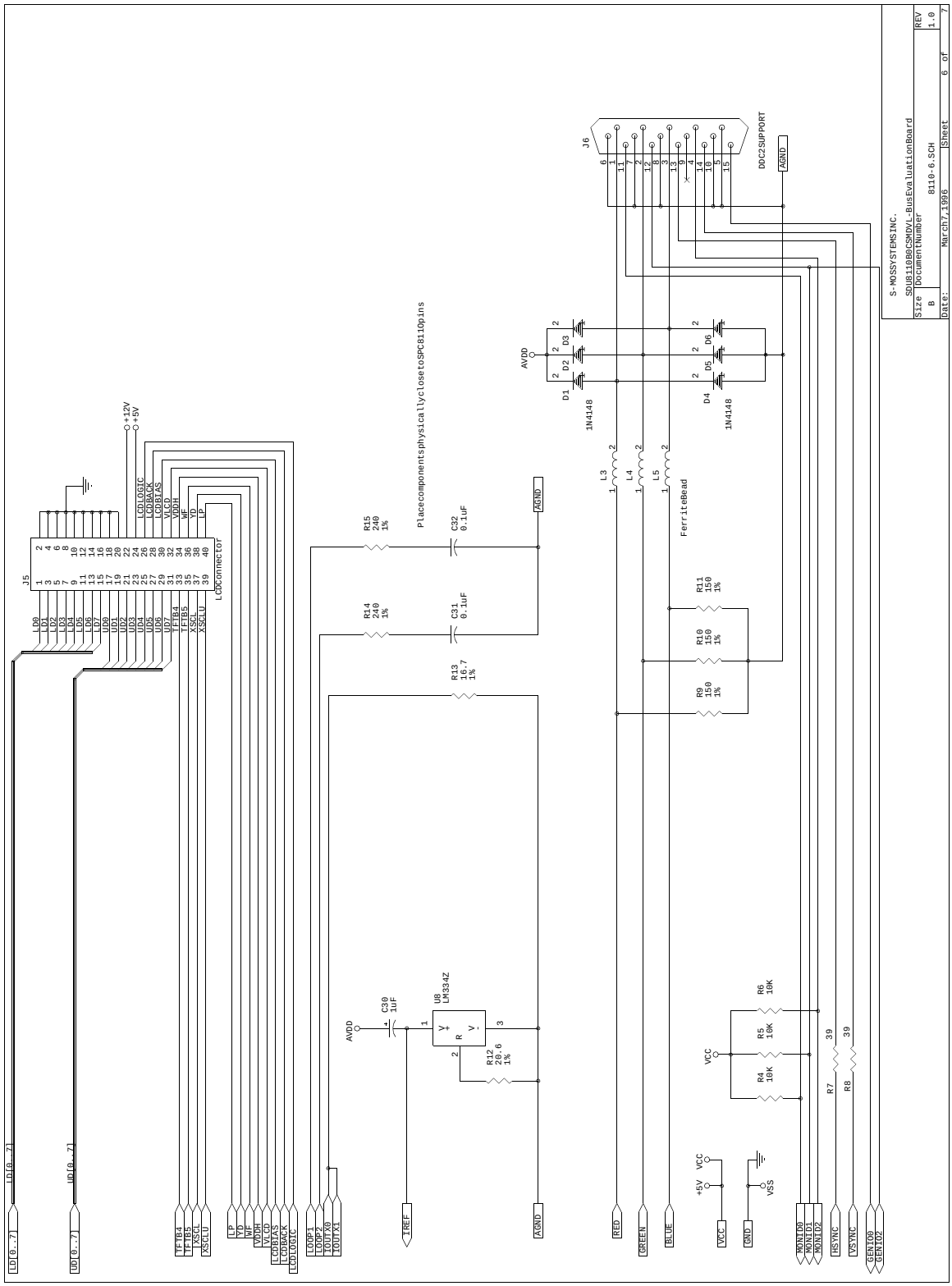


Figure 6 : SDU8110B0C Rev. 1.0 Schematic Diagram (6 of 7)

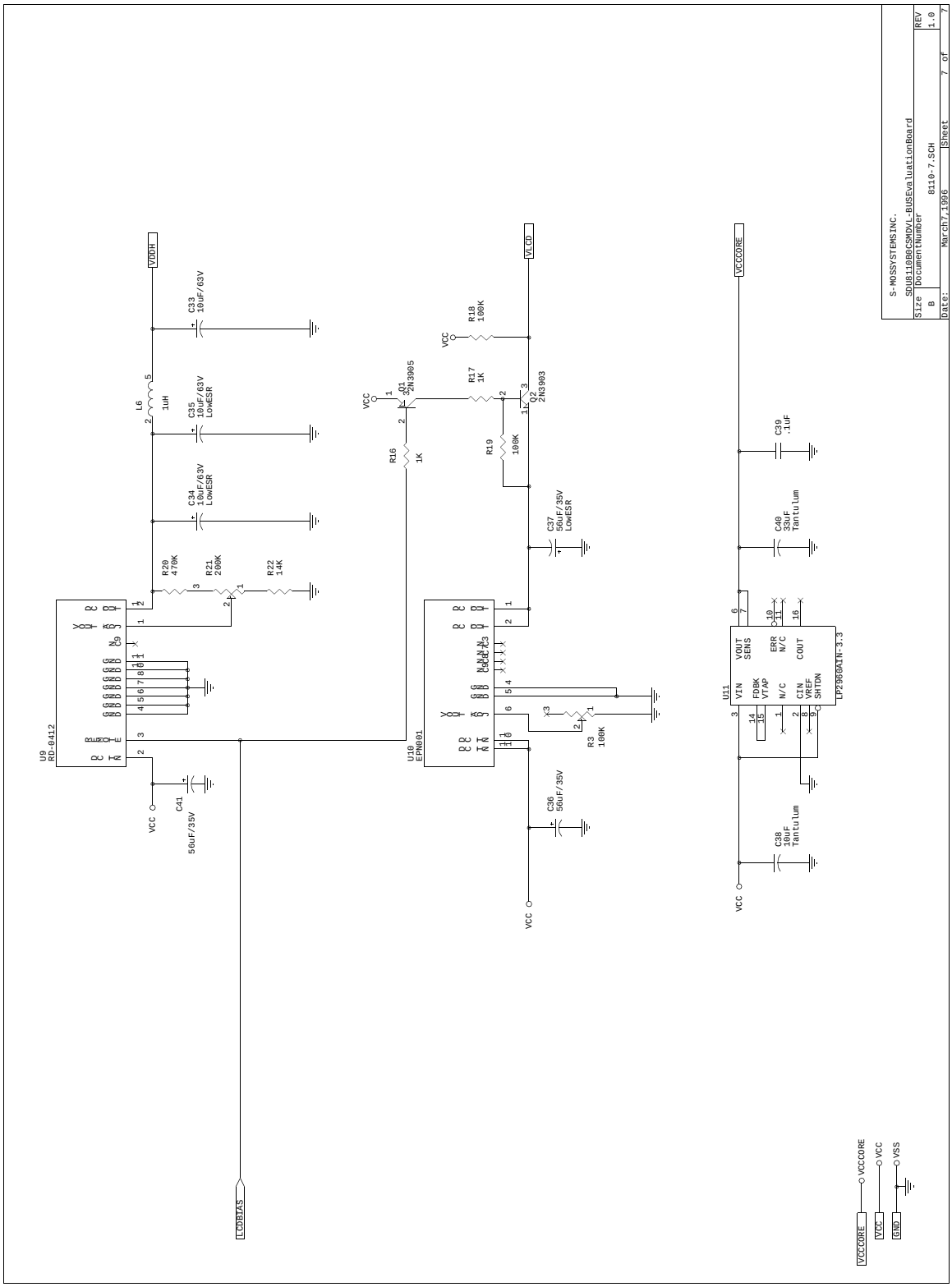


Figure 7 : SDU8110B0C Rev. 1.0 Schematic Diagram (7 of 7)

S-MOS SYSTEMS
A Seiko Epson Affiliate

SPC8110F0A LOCAL BUS LCD/CRT VGA CONTROLLER

SDU8110B0E Rev. 1.0 PCI Bus Evaluation Board

Drawing Office No. X07-AN-018-01

Copyright © 1996 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

TABLE OF CONTENTS

1 SDU8110B0E Rev. 1.0 Evaluation Board User Manual 7
1.1 Features 7
1.2 Installation and Configuration 8
1.2.1 DIP Switch Configuration Settings Table 8
1.2.2 Hard-Wired Configuration Inputs Table 8
1.2.3 DRAM Refresh Rate Select Table 9
1.2.4 LCD Panel Configuration Table 9
1.2.5 MDA0 PLL1 (PCLK) Select 10
1.2.6 MDA1 LCDBACK Polarity Select 10
1.2.7 MDA2 LCDLOGIC Polarity Select 10
1.2.8 MDA3 LCDBIAS Polarity Select 10
1.2.9 MDA4 Bus Interface 10
1.2.10 MDA5 PLL2 (MCLK) Select 10
1.2.11 MDA6-MDA7 Test Mode Inputs 10
1.2.12 MDA8-MDA12 LCD Panel Configuration Inputs 10
1.2.13 MDA13-MDA14 DRAM Refresh Rate Select 11
1.2.14 MDA15 Video DRAM Configuration Select 11
2 LCD Signal Connector Pinouts 12
3 Technical Description 13
3.1 32-Bit PCI-Bus Support 13
3.2 Video BIOS Support 13
3.3 1024K DRAM Support 13
3.4 Monochrome LCD Support 14
3.5 Color LCD Support 14
3.6 TFT Support 14
3.7 LCD Panel Control Signals 14
3.8 CRT Support 14
3.9 DDC2B Support 14
3.10 Power Save Modes 15
3.11 Power Down Clock Support 15
3.12 External REFCLK Support 15
3.13 Positive Power Supply 15
3.14 Negative Power Supply 15
3.15 Fixed 3.3 Volt Power Supply 15
3.16 PCB Layout Considerations 16
3.17 Schematic Notes 16
Appendix A PARTS LIST 17
Appendix B SDU8110B0E Schematic Diagrams 19

THIS PAGE LEFT BLANK

LIST OF TABLES

Table 1-1 DIP Switch Configuration Settings 8
Table 1-2 Hard-Wired Configuration Inputs 8
Table 1-3 DRAM Refresh Rate Select 9
Table 1-4 LCD Panel Configuration 9
Table 2-1 LCD Signal Connector Pinouts 12

LIST OF FIGURES

Figure 1 : SDU8110B0E Rev. 1.0 Schematic Diagram (1 of 6) 18
Figure 2 : SDU8110B0E Rev. 1.0 Schematic Diagram (2 of 6) 19
Figure 3 : SDU8110B0E Rev. 1.0 Schematic Diagram (3 of 6) 20
Figure 4 : SDU8110B0E Rev. 1.0 Schematic Diagram (4 of 6) 21
Figure 5 : SDU8110B0E Rev. 1.0 Schematic Diagram (5 of 6) 22
Figure 6 : SDU8110B0E Rev. 1.0 Schematic Diagram (6 of 6) 23

THIS PAGE LEFT BLANK

1 SDU8110B0E REV. 1.0 EVALUATION BOARD USER MANUAL

This manual reflects the use of the SDU8110B0E Rev. 1.0 evaluation board in conjunction with the SPC8110F0A VGA Controller chip while operating in a PCI-Bus environment. All appropriate components are surface-mount to reduce cost and minimize board size. Refer to the SPC8110F0A Functional Specification, Drawing Office No. X07-SP-001-xx for further details.

1.1 Features

- 208 pin QFP8 package
- 32-bit PCI-Bus support
- 3.3V Core voltage and 5.0V I/O voltage operation
- SMD technology for all appropriate devices
- Color/Monochrome STN LCD Panel support
- TFT Panel support
- CRT support
- DDC2B support
- Power Down Clock support
- External REFCLK support
- 1024K DRAM support (2 x 256K x 16, symmetrical, SOJ package)
- Support for Software Power Save Modes
- On-board +35 V adjustable power supply (VDDH for color panels)
- On-board -23V adjustable power supply (VLCD for monochrome panels)
- On-board fixed 3.3V Core Voltage (SOGVDD, AVDD)
- LCD Backlight/Logic/Bias Control

1.2 Installation and Configuration

The SDU8110B0E Rev. 1.0 directly supports the 32-bit PCI Bus architecture. The SPC8110F0A has 16 configurable inputs (MDA[15:0]) which are read on power-up. For the purpose of this design, most of the configuration inputs have been hard-wired and are set to power-up as appropriate. Due to the variety of LCD panels supported, a five position DIP Switch has been provided for different panel configurations. Refer to LCD Panel and DIP Switch Configuration Table below for details.

1.2.1 DIP Switch Configuration Settings Table

Table 1-1 DIP Switch Configuration Settings

Switch	Signal	Closed	Open
SW1-1	MDA8	LCD Panel Config bit 0	See LCD Panel Configuration Table
SW1-2	MDA9	LCD Panel Config bit 1	
SW1-3	MDA10	LCD Panel Config bit 2	
SW1-4	MDA11	LCD Panel Config bit 3	
SW1-5	MDA12	LCD Panel Config bit 4	

0 = Open, 1 = Closed

1.2.2 Hard-Wired Configuration Inputs Table

Table 1-2 Hard-Wired Configuration Inputs

	Signal	Closed	Open
	MDA0	PLL1 By-pass Mode (PCLK)	PLL1 enabled (PCLK)
	MDA1	LCDBACK Polarity (Active high)	LCDBACK Polarity (Active low)
	MDA2	LCDLOGIC Polarity (Active high)	LCDLOGIC Polarity (Active low)
	MDA3	LCDBIAS Polarity (Active high)	LCDBIAS Polarity (Active low)
	MDA4	BUS I/F Select (PCI Bus)	BUS I/F Select (32-bit VL-Bus)
	MDA5	PLL2 By-pass Mode (MCLK)	PLL2 enabled (MCLK)
	MDA6	RESERVED for Test Mode use only	RESERVED for Test Mode use only
	MDA7	RESERVED for Test Mode use only	RESERVED for Test Mode use only
	MDA13	DRAM Refresh Rate bit 0	See DRAM Refresh Rate Select Table
	MDA14	DRAM Refresh Rate bit 1	
	MDA15	DRAM type (2 CAS) (symmetrical)	DRAM type (2 WE) (asymmetrical)

0 = Open, 1 = Closed

= Hard-Wired board configuration

1.2.3 DRAM Refresh Rate Select Table

Table 1-3 DRAM Refresh Rate Select

MD14	MD13	512 cycles	1024 cycles
0	0	8 msec	16 msec
0	1	64 msec	128 msec
1	0	128 msec	256 msec
1	1	Self Refresh	

0=Open, 1=Closed

=Hard-wired Configuration

1.2.4 LCD Panel Configuration Table

The BIOS configuration for different LCD panels is done through the state of the MDA lines. The following tables describes the settings and the associated panel type.

Table 1-4 LCD Panel Configuration

Bit 4 (MDA12)	Bit 3 (MDA11)	Bit 2 (MDA10)	Bit 1 (MDA9)	Bit 0 (MDA8)	Resolution	Mode	Bits	Comments
1	1	1	1	1	640x480	Dual	8	Monochrome
1	1	1	1	0	640x480	Single	8	Monochrome
1	1	1	0	1	640x200	Single	4	Monochrome
1	1	1	0	0	320x240	Single	4	Monochrome
1	1	0	1	1	640x480	Dual	8	Color STN / 2 shift clocks
1	1	0	1	0	640x480	Single	8	Color STN
1	1	0	0	1	640x480	Dual	16	Color STN
1	1	0	0	0	640x480	Single	16	Color STN
1	0	1	1	1	640x480	Dual	6	Color (InFocus)
1	0	1	1	0	640x480	TFT	18	TFT
1	0	1	0	1	640x480	TFT	12	TFT
1	0	1	0	0	640x480	TFT	9	TFT
1	0	0	1	1	640x480	TFT	9	MODULATED
1	0	0	1	0	Half Size	Single	8	Optrex panel only
1	0	0	0	1	640x480	Dual	8	Color STN / 1 shift clock
1	0	0	0	0	OEM Config			
0	1	1	1	1	800x600	Dual	16	Color STN
0	1	1	1	0	Reserved			
through								
0	0	0	0	0				

Note
OEM Configuration options allow for very specific resolution/timing criteria to be programmed for a given panel. Refer to SPC8110F0A Functional Specification, Drawing Office No. X07-SP-001-xx for further details.

1.2.5 MDA0 PLL1 (PCLK) Select

MDA0 configuration input selects between on-chip internal PLL or external source for PCLK. MDA0 input is left open to support on-chip internal PLL for PCLK. For the purpose of this design, External PCLK source is not supported.

1.2.6 MDA1 LCDBACK Polarity Select

MDA1 configuration input selects between active high or active low polarity of LCDBACK signal. MDA1 input is left open to select active low polarity of LCDBACK signal.

1.2.7 MDA2 LCDLOGIC Polarity Select

MDA2 configuration input selects between active high or active low polarity of LCDLOGIC signal. MDA2 input is left open to select active low polarity of LCDLOGIC signal.

1.2.8 MDA3 LCDBIAS Polarity Select

MDA3 configuration input selects between active high or active low polarity of LCDBIAS signal. MDA3 input is left open to select active low polarity of LCDBIAS signal.

1.2.9 MDA4 Bus Interface

MDA4 configuration input selects between 32-bit VL-Bus or PCI-Bus interface. MDA4 input is set to logic high to select 32-bit PCI-Bus interface support from SPC8110F0A. For the purpose of this design, VL-Bus interface is not supported.

1.2.10 MDA5 PLL2 (MCLK) Select

MDA5 configuration input selects between on-chip internal PLL or external source for MCLK. MDA5 input is left open to support on-chip internal PLL for PCLK. For the purpose of this design, External MCLK source is not supported.

1.2.11 MDA6-MDA7 Test Mode Inputs

MDA6-MDA7 configuration inputs are reserved for test mode use purposes only. For the purpose of this design, these inputs are not used and left open.

1.2.12 MDA8-MDA12 LCD Panel Configuration Inputs

MDA8-MDA12 configuration inputs are used to configure the size and type of LCD panel being used. The state of these inputs is selected through the five position DIP switch. Refer to LCD Panel Configuration table above for specific settings required for specific LCD Panel.

1.2.13 MDA13-MDA14 DRAM Refresh Rate Select

MDA13 and MDA14 configuration inputs are bit 0 and bit 1 of DRAM Refresh Rate Select source. During Power Save Modes, provision must be made to refresh the video DRAM. As shown in *Table 1-3, "DRAM Refresh Rate Select," on page 9*, there are four options to refresh the video DRAM. Self-refresh DRAM is used for this design. Therefore, MDA13 and MDA14 inputs are set to logic high to select the Self Refresh option according to the table.

1.2.14 MDA15 Video DRAM Configuration Select

MDA15 configuration input selects between symmetrical or asymmetrical type DRAM support. The SDU8110B0E board is assembled with two 5V, 256Kx16 configuration, symmetrical architecture , Self Refresh, SMD, SOJ package DRAMs. For this design, MDA15 input is set to logic high to support symmetrical architecture DRAM.

2 LCD SIGNAL CONNECTOR PINOUTS

Table 2-1 LCD Signal Connector Pinouts

SPC8110Pin Name	LCD Connector Pin No.	TFT			Color STN LCD				Mono STN LCD	
		18-bit color	12-bit color	9-bit color	16-bit	8-bit Half panel	8-bit	6-bit	8-bit	4-bit
LD0	1	R4	R2	R1	LD0		LD0	LB	LD0	
LD1	3	R5	R3	R2	LD1		LD1	LG	LD1	
LD2	5	G0			LD2		LD2	LR	LD2	
LD3	7	G1			LD3		LD3		LD3	
LD4	9	R0			LD4				UD0	UD0
LD5	11	R1			LD5				UD1	UD1
LD6	13	R2	R0		LD6				UD2	UD2
LD7	15	R3	R1	R0	LD7				UD3	UD3
UD0	17	B0			UD0	UD0	UD0	UB		
UD1	19	B1			UD1	UD1	UD1	UG		
UD2	21	B2	B0		UD2	UD2	UD2	UR		
UD3	23	B3	B1	B0	UD3	UD3	UD3			
UD4	25	G2	G0		UD4	UD4				
UD5	27	G3	G1	G0	UD5	UD5				
UD6	29	G4	G2	G1	UD6	UD6				
UD7	31	G5	G3	G2	UD7	UD7				
TFTB4	33	B4	B2	B1						
TFTB5	35	B5	B3	B2						
XSCL	37				XSCL	XSCL	XSCL	XSCL	XSCL	XSCL
XSCLU	39	CLK	CLK	CLK			XSCLU			
LP	40	HS	HS	HS	LP	LP	LP	LP	LP	LP
YD	38	VS	VS	VS	YD	YD	YD	YD	YD	YD
WF	36	DE	DE	DE	WF	WF	WF	WF/DE	WF	WF
VDDH	34				VDDH		VDDH			
VLCD	32								VLCD	VLCD
LCD BIAS	30	LCD BIAS	LCD BIAS	LCD BIAS	LCD BIAS	LCD BIAS	LCD BIAS	LCD BIAS	LCD BIAS	LCD BIAS
LCD BACK	28	LCD BACK	LCD BACK	LCD BACK	LCD BACK	LCD BACK	LCD BACK	LCD BACK		
LCD LOGIC	26	LCD LOGIC	LCD LOGIC	LCD LOGIC	LCD LOGIC	LCD LOGIC	LCD LOGIC	LCD LOGIC	LCD LOGIC	LCD LOGIC
+5 V	24	+5 V	+5 V	+5 V	+5 V	+5 V	+5 V	+5 V	+5 V	+5 V
+12 V	22	+12 V	+12 V	+12 V	+12 V	+12 V	+12 V	+12 V	+12 V	+12 V
GRND	20-2	GRND	GRND	GRND	GRND	GRND	GRND	GRND	GRND	GRND

3 TECHNICAL DESCRIPTION

3.1 32-Bit PCI-Bus Support

This board directly supports the 32-bit 5V PCI-Bus and is intended to be used as a PCI-Bus Target add-in-board. The SPC8110F0A can be configured as 3.3V or 5.0V for the 32-bit PCI-Bus I/O interface. For the purpose of this design, only the 5.0V I/O PCI-Bus interface is supported. All SPC8110F0A related electrical connections come from the standard PCI-Bus connector. A utility program is supplied to support video BIOS in software. Refer to SDU8110B0x PCI Boot Procedure Utility, Drawing Office No. X07-AN-016-xx for further information.

3.2 Video BIOS Support

In a "normal" system implementation the SPC8110F0A would be designed directly onto the motherboard, as opposed to a peripheral board design. For this reason the SPC8110F0A does not provide the internal BIOS decoding logic, as the video BIOS would normally be incorporated into the system BIOS. For the purposes of this evaluation board, external decode logic is not possible or practical, given certain limitations in the PCI Bus Specification regarding the number of devices interfacing to one bus slot (refer to the latest PCI Specification for further details). Because of these limitations the video BIOS is supported in software. A utility program is used to initialize the BIOS code after the system has booted.

To use this evaluation board:

1. The following three files are required, copy them to the computer's hard-drive root directory (i.e., c:\):
   - PCIREG4.exe : enables the PCI Configuration registers
   - LOADROM.exe: utility to initialize the binary BIOS file
   - HxxxDPCI.bin: 8110 PCI video BIOS binary file
2. Add these lines to the end of the AUTOEXEC.bat file:
   - PCIREG4 3
   - LOADROM HxxxDPCI.bin
3. Program the computers Standard CMOS Set-up for "No Video". This will allow the system to boot without looking for a video card.

Note
xxx in the file name HxxxDPCI.bin is the revision level of the binery file.

As configured above, the computer will boot with no display until after the AUTOEXEC.bat file has completed. The display will then become active and the computer and video card will operate as normal.

The above described procedure is not necessary in a "normal" motherboard implementation and is only required for this particular evaluation board design.

3.3 1024K DRAM Support

A two chip 1024K configuration is supported using 2 x 256K x 16 DRAM. Only symmetrical DRAM in an SOJ package is supported.

3.4 Monochrome LCD Support

The SPC8110F0A supports 4- and 8-bit Dual and Single monochrome STN LCD panels. All the necessary signals are provided on the 40-pin ribbon cable header. The interface signals are alternated with grounds on the cable to reduce cross talk and noise related problems.

Refer to *Table 1-4, "LCD Panel Configuration," on page 9* and *Table 2-1, "LCD Signal Connector Pinouts," on page 12* for specific settings.

3.5 Color LCD Support

The SPC8110F0A also supports 8- and 16-bit Dual and Single color STN LCD panels. All the necessary signals can also be found on the 40-pin ribbon cable connector. The interface signals are alternated with grounds on the cable to reduce cross talk and noise related problems.

Refer to *Table 1-4, "LCD Panel Configuration," on page 9* and *Table 2-1, "LCD Signal Connector Pinouts," on page 12* for specific settings.

3.6 TFT Support

The SPC8110F0A also supports 9-, 12-, and 18-bit TFT panels. All the necessary signals can also be found on the 40-pin ribbon cable connector. The interface signals are alternated with grounds on the cable to reduce cross talk and noise related problems.

Refer to *Table 1-4, "LCD Panel Configuration," on page 9* and *Table 2-1, "LCD Signal Connector Pinouts," on page 12* for specific settings.

3.7 LCD Panel Control Signals

Control signals are provided on the connector (LCDLOGIC, LCDBIAS, LCDBACK). These signals are low current direct outputs from the SPC8110F0A and are intended to be used in-conjunction with external circuitry to provide higher current switching.

3.8 CRT Support

The SPC8110F0A has a built-in RAMDAC for direct support of CRT monitors. All standard VGA resolutions as well as extended modes up to 1024 x 768 are supported.

3.9 DDC2B Support

The SDU8110B0E Rev. 1.0 evaluation board is capable of directly supporting the VESA DDC2B monitor standard. This board supports hardware DDC2B and the BIOS supports software layer VBE/DDC version 1.0. For further information on DDC2B contact the Video Electronics Standards Association (VESA).

3.10 Power Save Modes

The SPC8110F0A supports two software and two hardware Power Save Modes. A utility program is supplied to control the software modes. The software modes are controlled by directly writing the SPC8110F0A associated internal registers. For the purpose this design hardware Power Save Modes are not supported.

3.11 Power Down Clock Support

A 32.768 KHz oscillator is provided for connection to the PSCLK signal (pin 153). This clock source is required to provide the DRAM refresh cycles.

3.12 External REFCLK Support

The SPC8110F0A requires 14.318Mhz external frequency source for use by the internal PLL. An on-board SMD package 14.318Mhz oscillator is used to provide this function.

3.13 Positive Power Supply

The majority of Color STN LCD panels require a positive power supply to provide between +23 V and +40 V (I_{out}=45mA) (VDDH). For ease of implementation, such a power supply has been provided as an integral part of this design. The signal VDDH can be found on J3 (pin 34) and its output voltage can be controlled by R29 and the control signal LCDBIAS.

R29 (200K potentiometer) can be adjusted to provide an output voltage from +23 V to +40 V.

LCDBIAS is a control signal provided by the SPC8110F0A to enable/disable the VDDH power supply. This signal guarantees correct power sequencing and prohibits subsequent panel damage.

Determine the panel's specific power requirements and set R29 accordingly before connecting the panel.

3.14 Negative Power Supply

The majority of Monochrome STN LCD panels require a negative power supply to provide between -18 V and -23 V (I_{out}=45mA) (VLCD). For ease of implementation, such a power supply has been provided as an integral part of this design. The signal VLCD can be found on J3 (pin 32) and its output voltage can be controlled by R32 and control signal LCDBIAS.

R32 (100K potentiometer) can be adjusted to provide an output voltage from -18 V to -23 V.

LCDBIAS is a control signal provided by the SPC8110F0A which is used to enable/disable the VLCD power supply. This signal guarantees correct power sequencing and prohibits subsequent panel damage.

Determine the panel's specific power requirements and set R32 accordingly before connecting the panel.

3.15 Fixed 3.3 Volt Power Supply

An independent 3.3V power supply at 0.5A (maximum) has been provided as an integral part of this design using a National LP2960AIN-3.3 Micropower 0.5A Low-Dropout Voltage Regulator for the core and analog logic sections (SOGVDD, MCLVDD, AVDD) of SPC8110F0A.

3.16 PCB Layout Considerations

The physical board dimensions and component placement are determined by the PCI Bus specification Rev 2.0. Maximum trace length parameters will determine the SPC8110F0A placement / rotation relative to the Bus. The CRT connector must be accessible at the rear of the computer. While the position of the LCD connector is not specified, its location will be determined by the SPC8110F0A placement / rotation.

The board has 4 layers to accommodate various power/ground requirements.

The Power plane is divided into four sections providing for;

VCC: +5V
PSVCC: isolated from VCC power supply via a ferrite bead to reduce VDDH and VLCD power supply generated noise.
VCCCORE: +3.3V
AVDD: isolated from VCCCORE (3.3V) fixed power supply via a ferrite bead.

The Ground plane is divided into three sections providing for;

GND: normal digital ground
AGND: all DAC and PLL related signals will be routed over this plane only
PSGND: isolated from GND via a ferrite bead to reduce VDDH and VLCD power supply generated noise.

All ground sections are connected together through ferrite beads.

The actual routing of both the power and ground planes follows layout guidelines that no digital signals will be routed across the AGND plane.

Traces being routed from the PCI Bus to the SPC8110F0A should not exceed 1.5" in length.

3.17 Schematic Notes

The evaluation boards may have been modified and therefore the following schematic diagrams may not reflect the actual implimentation. Please request updated information before starting any hardware design.

Note
External PLL Loop Values. For revision 1.0 of the SDU8110B0E evaluation board capacitors C29 and C30 are 0.1 uF, R12 and R13 are 70 ohms.

APPENDIX A PARTS LIST

Item #	Qty/ Board	Designation	Part Value	Description
1	24	C1-C22, C27, C36	0.01uF	0.01uF, 1206 pckg
2	1	C26	47uF	47uF / 15V Tantalum D-SIZE
3	3	C23-C25	10uF	10uF / 25V Tantalum D-SIZE
4	1	C28	1uF	1uF Tantalum A-SIZE
5	3	C29,C30	0.56uF	0.56uF, 1206 pckg
6	1	C37	0.1uF	0.1uF, 1206 pckg
7	3	C31-C33	10uF / 63V	Electrolytic / Radial (LXF63VB10RM5X11LL)
8	3	C34,C35,C39	56uF/35V	LXF35VB56RM6X11LL
9	1	C38	33uF	33uF / 10V Tantalum D-SIZE
10	6	D1-D6	1N4148	Signal Diode / PTH
11	1	J3	CON40A	Shrouded Header 40 pin Dual-row center-key PTH
12	1	J4	PS/2 CONNECTOR	Assman A-HDF 15 A KG/T or equivalent
13	7	L1-L5, L7-L8	Ferrite Bead	Fair-rite 2743001111 PTH
14	1	L6	1uH	Dale Inductor IM-4-1.0uH PTH
15	1	Q1	2N3903	PNP Signal Transistor TO-92 PTH
16	1	Q2	2N3906	NPN Signal Transistor TO-92 PTH
17	5	R1-R5	4.7K	4.7K Ohm/1206/5%
18	1	R6	10K	10K 10-pin SIP 4610X-101-103 PTH
19	2	R7, R8	39	39 Ohm /1206 / 5%
20	3	R9-R11	150	150 Ohm / 1206 / 1%
21	2	R12, R13	68	68 Ohm / 1206 / 1%
22	1	R14	16.5	16.5 Ohm / 1206 / 1%
23	1	R15	20.5	20.5 Ohm / 1206 / 1%
24	4	R16-R19	2.2K	2.2K Ohm/1206/5%
25	5	R20-R24	33	33 Ohm /1206 / 5%
26	2	R25, R31	1K	1K Ohm /1206 / 5%
27	2	R26, R27	100K	100K Ohm /1206 / 5%
28	1	R28	470K	470K Ohm / PTH / .25W / 5%
29	1	R29	200K	200K Ohm Trim POT Spectrol 63S204T607
30	1	R30	14K	14K Ohm / PTH / .25W / 5%
31	1	R32	100K	100K Ohm Trim POT Spectrol 63S104T607
32	1	S1	SW-DIP-5	DIP Switch 5-position
33	1	U1	SPC8110F0A	QFP8-208/ 208 pin S-MOS will supply
34	2	U2, U3	uPD42S4260LE-70	256Kx16 , Self-Refresh, Symmetrical DRAM
35	1	U4	ML13FAD-32.768Khz	32.768Khz Oscillator 14-pin DIP pckg.
36	1	U5	TCO-711JT-14.31818Mhz	14.31818 Mhz Oscillator 4-pin SMO-J pckg.

Item #	Qty/ Board	Designation	Part Value	Description
37	1	U6	LM334Z	TO-92 pckg.
38	1	U7	RD-0412	XENTEK - Positive Power Supply
39	1	U8	EPN001	XENTEK - Negative Power Supply
40	1	U9	LP2960AIN-3.3	National 3.3V FIXED Voltage Regulator N16G 16-pin DIP pckg.

APPENDIX B SDU8110B0E SCHEMATIC DIAGRAMS

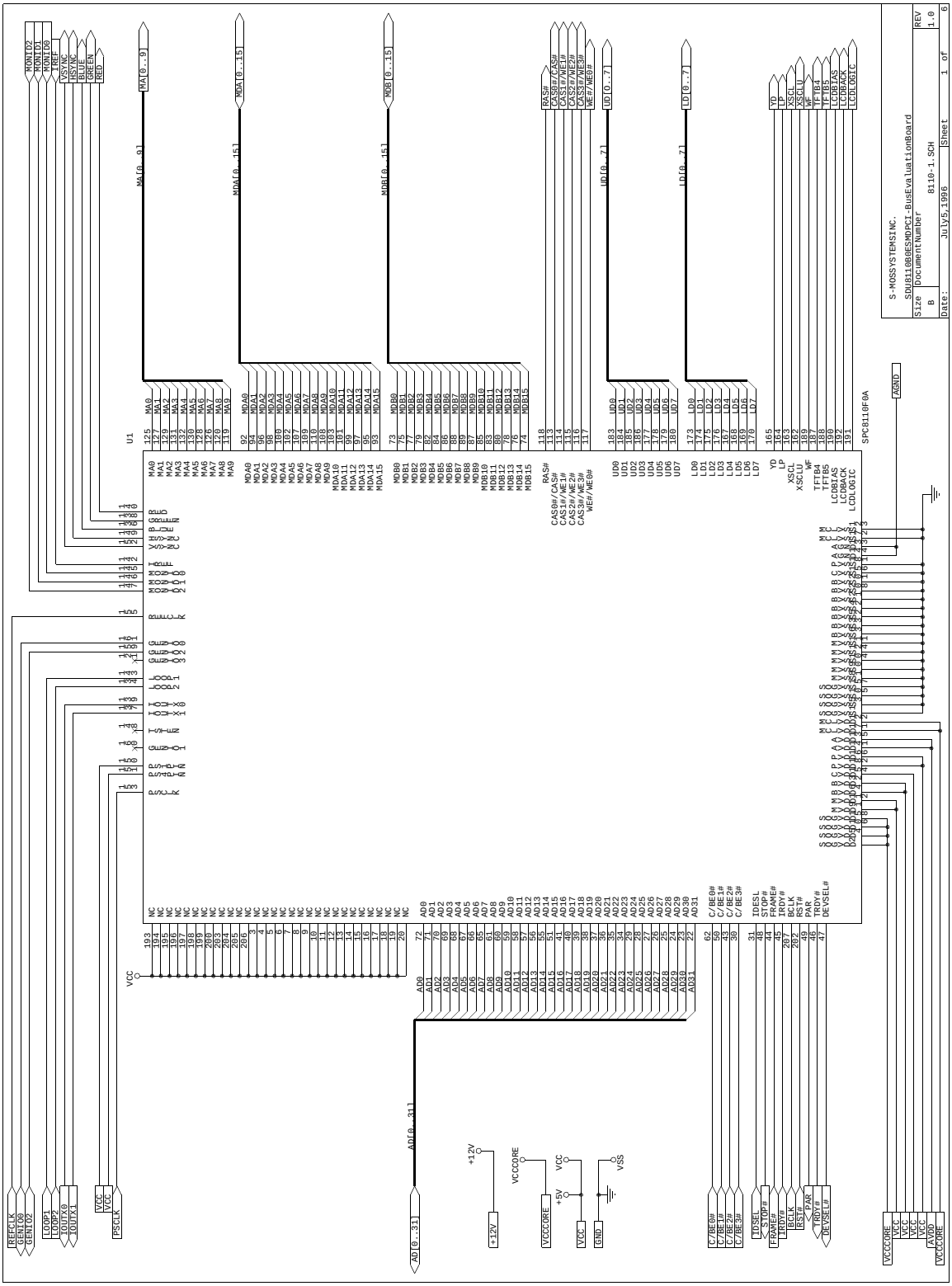


Figure 1 : SDU8110B0E Rev. 1.0 Schematic Diagram (1 of 6)

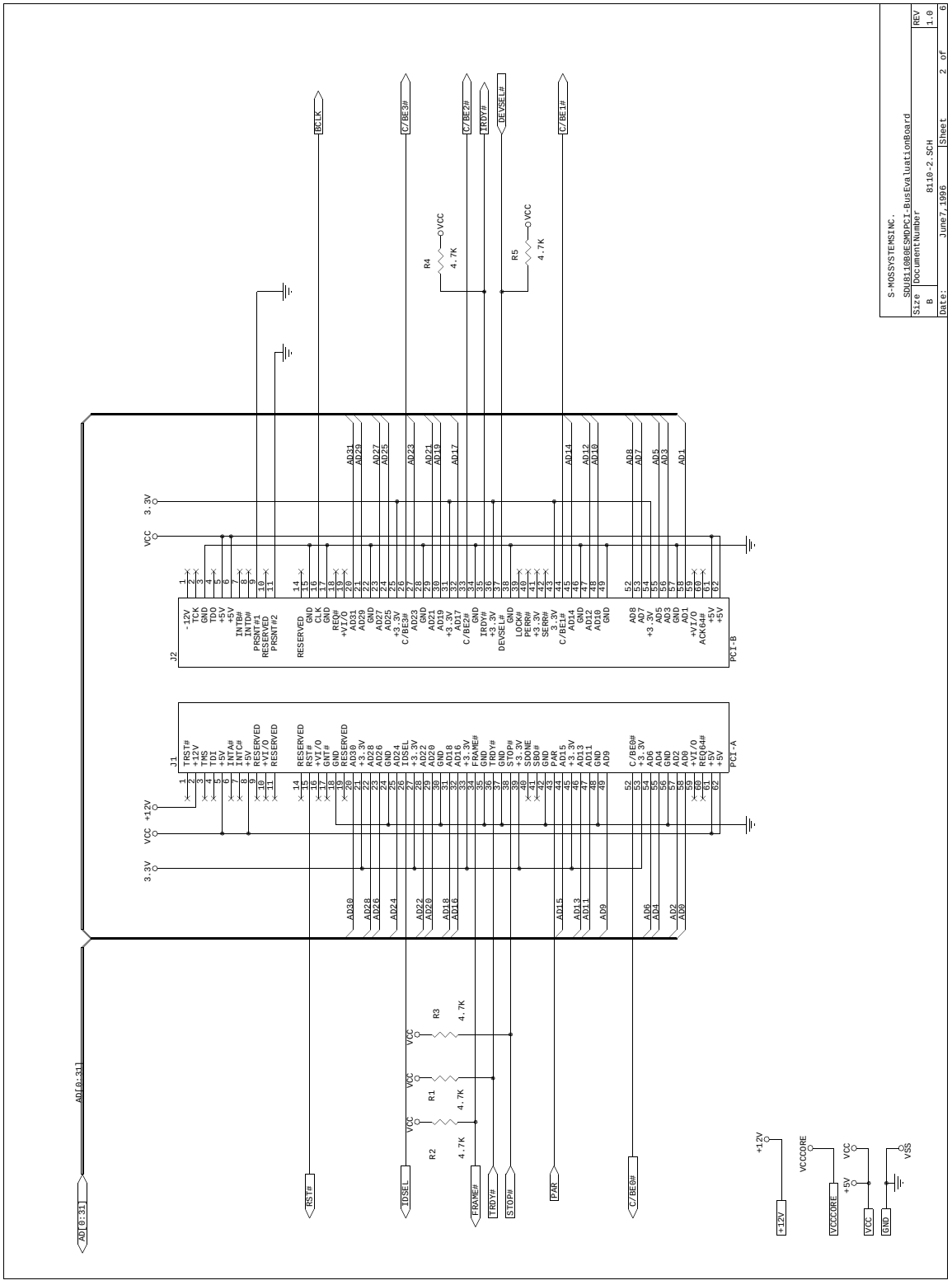


Figure 2 : SDU8110B0E Rev. 1.0 Schematic Diagram (2 of 6)

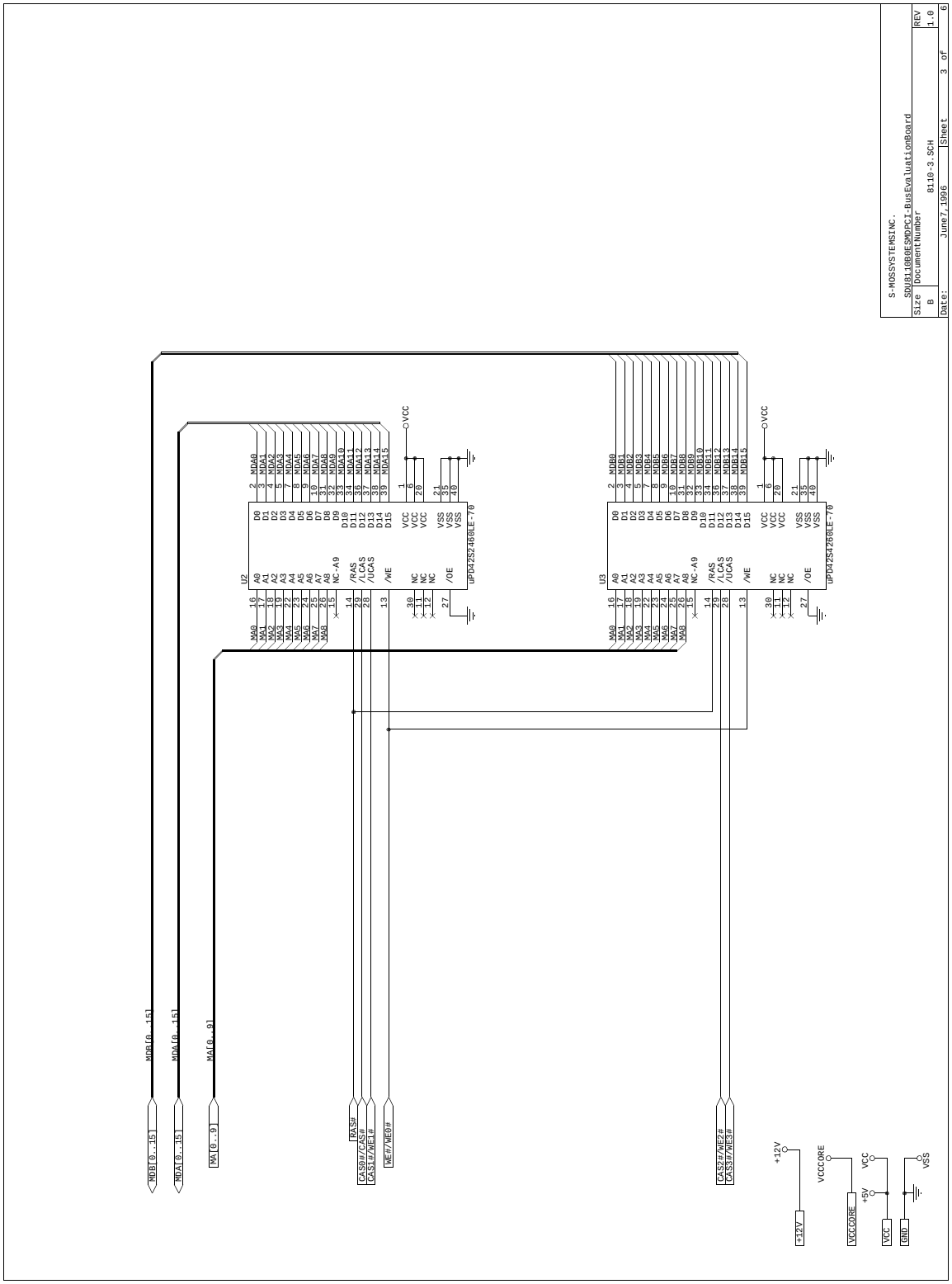


Figure 3 : SDU8110B0E Rev. 1.0 Schematic Diagram (3 of 6)

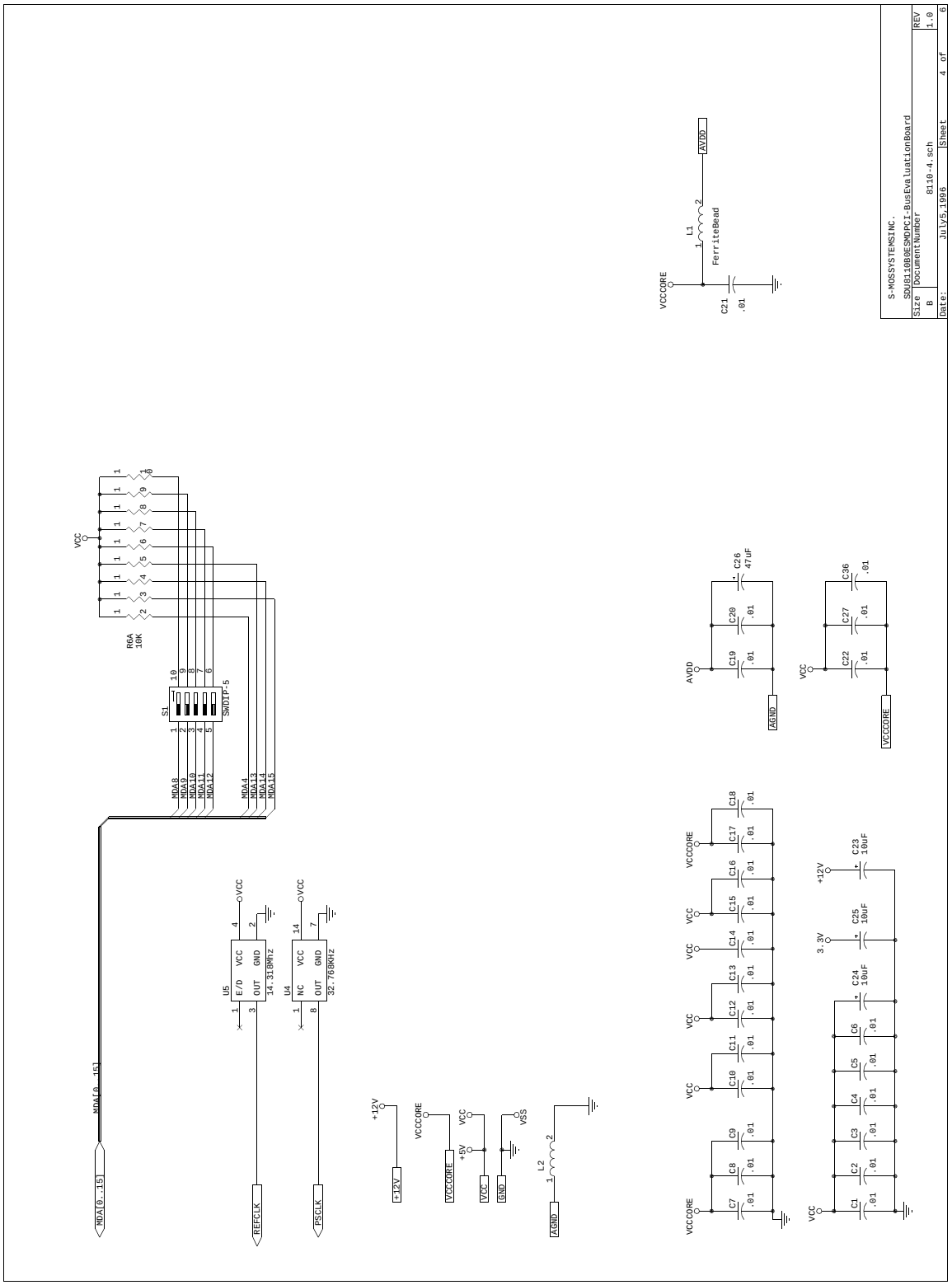


Figure 4 : SDU8110B0E Rev. 1.0 Schematic Diagram (4 of 6)

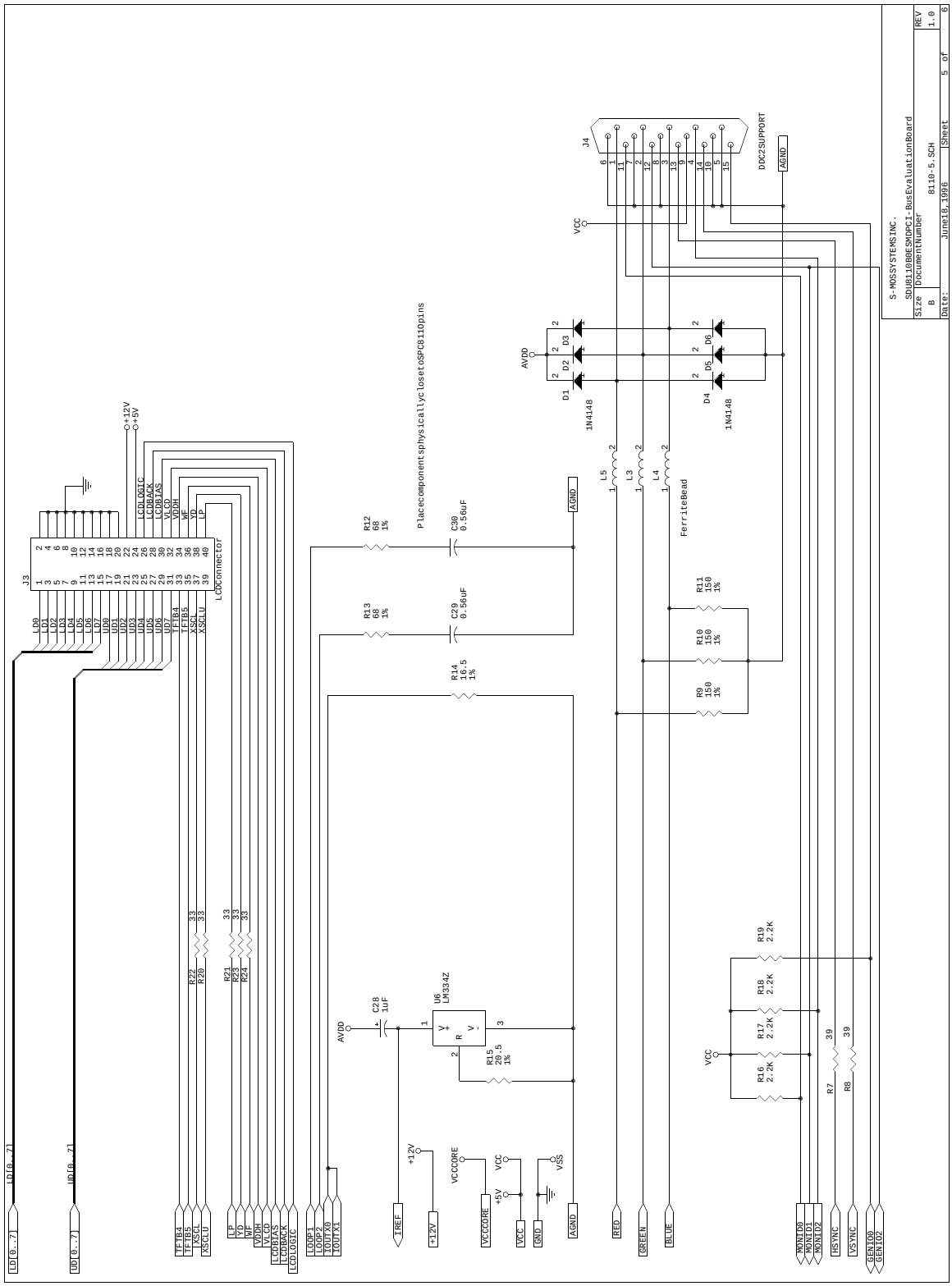


Figure 5 : SDU8110B0E Rev. 1.0 Schematic Diagram (5 of 6)

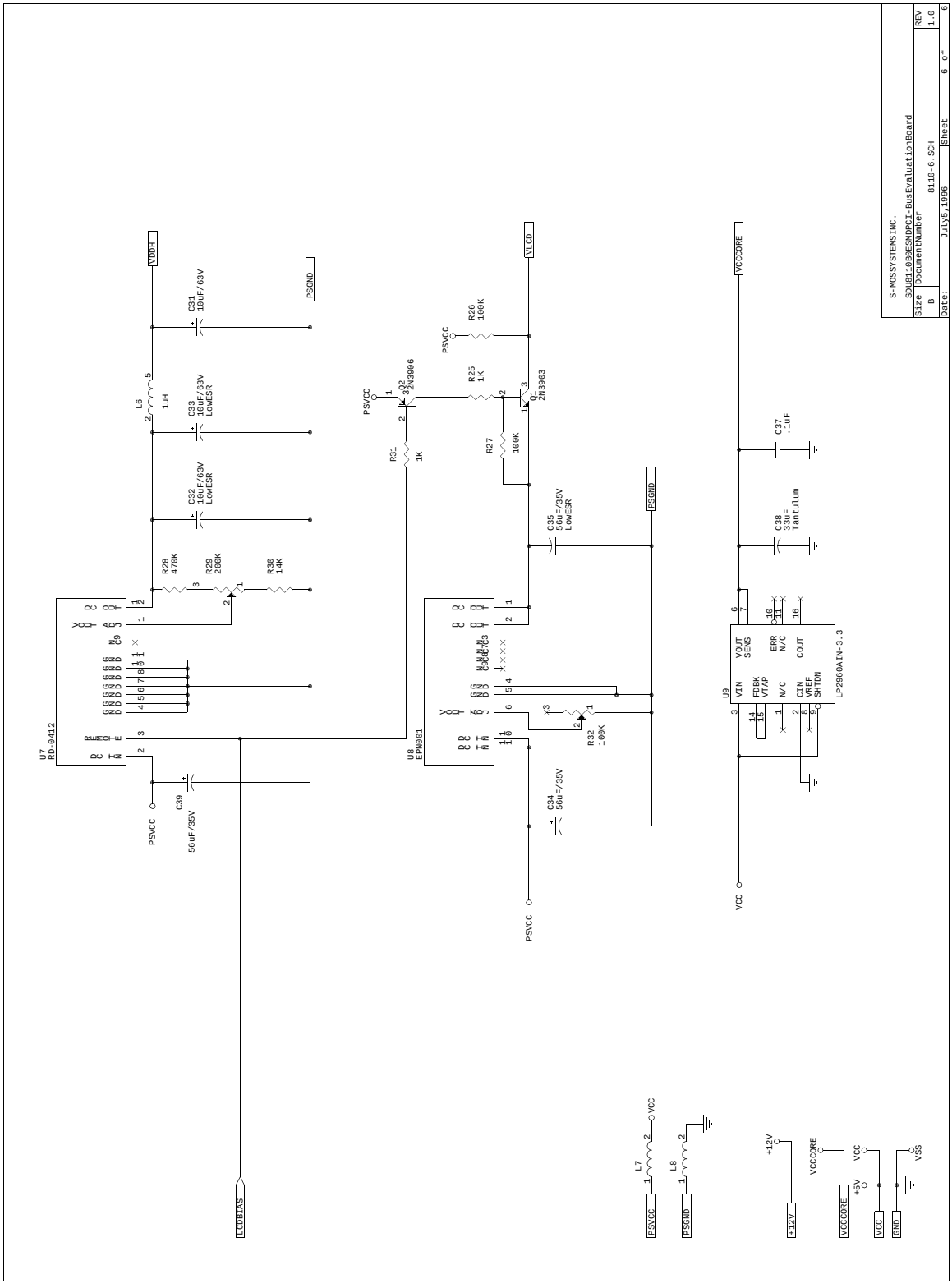


Figure 6 : SDU8110B0E Rev. 1.0 Schematic Diagram (6 of 6)

S-MOS
SYSTEMS
A Seiko Epson Affiliate

SPC8110F0A LCD/VGA Controller

SPC8110F0A Power Consumption

Drawing Office No. X07-AN-015-01

Copyright © 1995, 1996 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

SPC8110F0A POWER CONSUMPTION

INTRODUCTION

This document is intended for OEM reference for the SPC8110 power consumption. It defines multiple test conditions for which power consumption was measured. The measurement was based on average of three samples in a typical PC.

Conditions

1. V_{DD}(core): 3.3 volts
   V_{DD}(I/O): 5.0 volts

2. Controller operating in Single Panel Color LCD Display mode, with no display connected.

3. PClk (Pixel Clock) = 28.322MHz (Single panel mode).

4. Mode 3 Active: Screen filled with 4Dh 07h (640x400 text mode, screen filled with letter M).

5. Mode 12 Active: Screen filled with 55h (640x480 Graphics mode, 16 color, screen filled with alternate on/off pixel pattern).

6. Mode 101 Active: Screen filled with pattern (640x480 Graphics mode 256 color, screen filled with alternate on/off pixel pattern).

7. Power Save Condition #1: Mode 101, screen filled with alternate on/off pixel pattern, Reduce Gray Scale.

   Aux[11] bit 0 = 1 - This sets the pixel path leading to the VGA palette and is redirected to a gray scale reduction palette, thus bypassing the VGA palette entirely.

8. Power Save Condition #2: Mode 101, screen filled with alternate on/off pixel pattern, PCL Slow.

   Aux[11] bit 1 = 1 - This function selects the Set1 PLL1 registers that are programmed with values that would lower the PClk frequency. It also selects the appropriate slow dot filo level, therefore decreasing the power consumption in proportion to the frequency.

9. Power Save Condition #3: Display Off.

   Aux[11] bit 2 = 1 - All circuits that contribute to the displaying of an image are shut down. The display FIFO, hardware cusor, frame buffer and the LCD gray-scale circuits become static. The VGA palette is partially shut down, while maintaining full CPU access.

10. Power Save Condition #4: PClk.

    Aux[11] bit 6 = 1 - This function is nearly identical to the Display Off function with two exceptions: circuitry that generates sync signals for the CRT are shut down and the PLL that supplies the pixel clock is shut down.

11. Power Save Condition #5: MClk Off.

    Aux[11] bit 7 = 1 - Entering a power save mode with this condition set causes PLL2 to be powered down.

12. Power Save Condition #6: MClk and PClk Off.

    Aux[11] bit 6,7 = 1,1 - This condition causes PLL1 and PLL2 to be powered down.

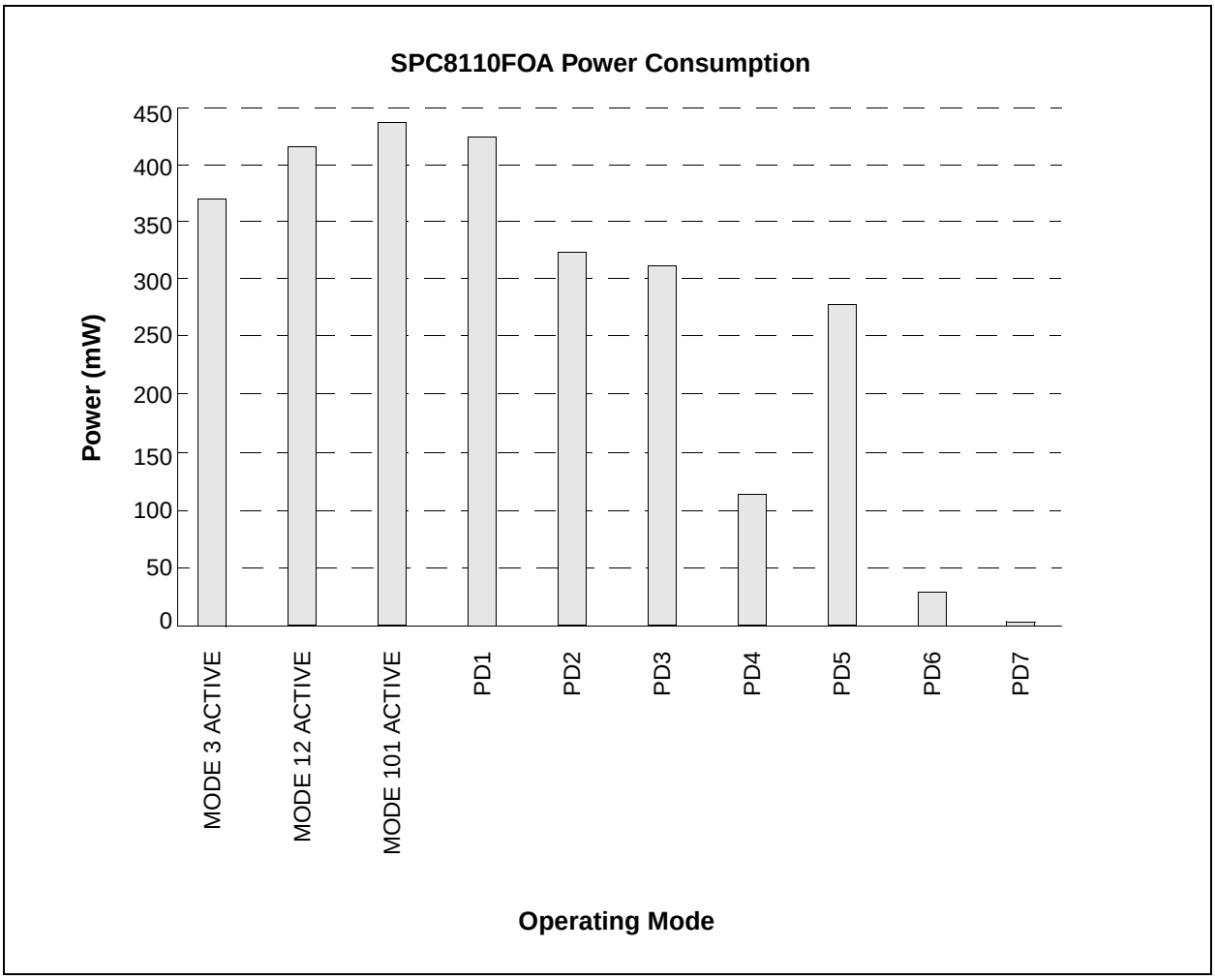
13. Power Save Condition #7: MClk, PClk, Bus I/F Off.

    Aux[11] bit 6, 7 = 1, 1 and Aux[15] bit 0 = 1 - This condition causes PLL1 and PLL2 to be powered down. Also, the bus output buffers are tri-stated.

Notes

1. For more details, please reference *Section 9.7 Power Save Mode* in the *SPC8110 Hardware Functional Specification.*

2. These data are results from typical measurements and that they may vary based on environment.



	Mode 3 Active	Mode 12 Active	Mode 101 Active	PS Condition H1	PS Condition H2	PD3	PD4	PD5	PD6	PD7
Power mW	369	418	434	429	328	307	104	270	33	0.9

THIS PAGE LEFT BLANK

S-MOS SYSTEMS
A Seiko Epson Affiliate

SPC8110F0A LCD/CRT VGA Controller

PCB Layout Guidelines

Drawing Office No. X07G-G-001-01

Copyright © 1996 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

Table of Contents

1 INTRODUCTION 5

2 Internal Power Rail Architecture 5

3 PCB Layout Considerations 5

3.1 Analog / Digital Isolation 5

3.2 Trace Lengths 6

3.2 CPU Interface 6

3.2 Input Clock 6

3.2 Memory Interface 6

3.2 LCD Panel Connection 6

3.3 Bypass Capacitors 7

4 Summary 8

THIS PAGE LEFT BLANK

1 INTRODUCTION

The SPC8110F0A is a mixed signal device, having both digital and analog internal components. The internal PLL and the integrated RAMDAC make up the analog portion with the remaining functional blocks all digital. In any mixed signal environment certain PCB layout guidelines should be followed in order to prevent any unwanted digital signal noise from being coupled into the analog logic.

The following guidelines should be followed when using the SPC8110F0A.

2 Internal Power Rail Architecture

The SPC8110F0A has seven individual power rails. All VDD pins are internally separated allowing for mixed voltage operation. All VSS pins are internally common with the exception of AGND (Analog Ground) allowing for separate external grounds separating the digital and analog signals.

3 PCB Layout Considerations

This document makes recommendations based on a 4 layer PCB design (2 signal, 2 power).

3.1 Analog / Digital Isolation

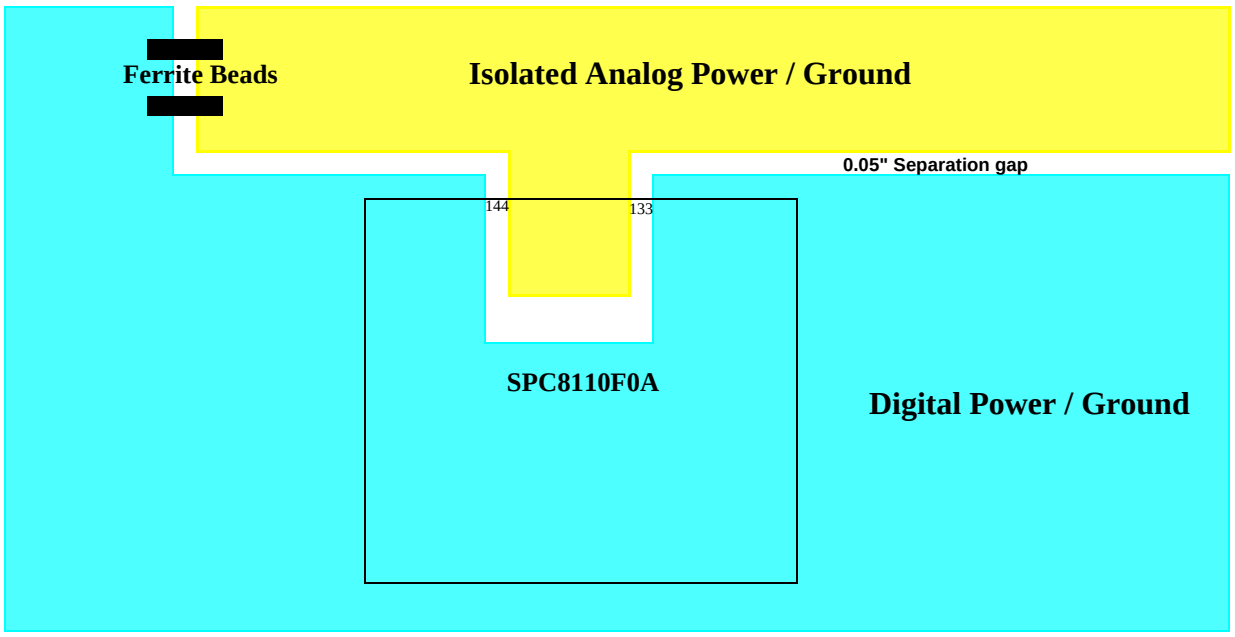
To minimize the possibility of digital signal noise being coupled into the analog logic, the analog power / ground plane must be electrically isolated from the digital power / ground planes. This can be accomplished by providing a physical separation between the analog and digital sections of the PCB design. Separate analog power / ground planes should be accommodated in the layout with a minimum 0.05" gap between the digital power / ground planes.

Only analog signals should only be routed over the isolated analog planes. If it does become necessary for digital signals to be routed in, or cross over this area they should be separated from analog signals by routing them on another layer with a power plane separating the two layers.

Digital and analog signals should not be routed on the same layer without adequate separation as this will increase the possibility for coupled noise and loss of signal integrity.

The analog power / ground planes should be electrically connected to the digital power / ground planes through some inductance (Ferrite Beads). This provides a high impedance path for high frequency digital signals therefore preventing the analog sections from being used as an AC return path.

Pins 133-144 on the SPC8110F0A encompass all the analog signals. The separate power planes should include these pins, refer to the following drawing;



All analog components should be located within the isolated analog power planes.

3.2 Trace Lengths

CPU Interface

The individual CPU interfaces supported are VL-Bus, PCI-Bus, and generic ’486-Bus, each having it’s own trace lengths specification, refer to the individual Bus Interface specifications for details.

Input Clock

The SPC8110F0A requires two clock inputs (14.318MHz - REFCLK, 32KHz - RTCLK,). These clock traces should be kept as short as possible with the physical components located near the SPC8110F0A.

Memory Interface

The SPC8110F0A uses a 40MHz memory clock to access the display buffer. This clock speed / edge rate combined with the DRAM input impedance plus the trace impedance determine the maximum trace length for the memory interface signals. Series termination may be required for excessive trace lengths resulting in signal reflections.

LCD Panel Connection

The SPC8110F0A uses 24mA output drivers for the LCD signals. The length of the trace from the controller to the panel includes the cable length. Maximum cable length in this case is partially based on the pixel clock frequency required for the specific panel (e.g. different panel types require different frequencies (less than 14MHz to 28MHz). The higher the frequency the shorter the cable.

There are many factors in maximum cable length calculations;

- clock frequency
- signal edge rates
- signal orientation along the cable
- type of medium (cable type)
- output drive capability

The above factors help in determining maximum cable length based on;

- percentage of coupling (inductive and or capacitive)
- reflection voltage levels (signal edge rate determining the wave length of a given signal)
- load impedance (determined by the type pf medium and any external termination).

A generalization of the above rules would be; keep the cable as short as possible while having adequate signal separation within the cable assembly / routing.

3.3 Bypass Capacitors

The necessity of bypass capacitors on a 4-layer PCB can be argued however that is beyond the scope of this application note.

If bypass capacitors are to be used they must be surface-mount packages and be located close to the power pin they are supposed to filter. Any additional trace length or component leads will add inductance and thereby cancel any capacitive affect.

Power Rail filter capacitors are used where the voltage supply may suddenly drop for very short periods of time when excessive load is encountered. These power rail capacitors are used to maintain the voltage level during these 'glitches'. A typical power rail filter capacitor is a polarized tantalum with a value ranging from 5.6uF to 47uF. The voltage range of this capacitor should always be twice the voltage as the rail it is connected to.

4 Summary

Following the recommendations listed above will result in an electrically ’clean’ environment reducing the possibility of problems arising from loss of signal integrity.

S-MOS
SYSTEMS
A Seiko Epson Affiliate

SPC8110F0A VGA Controller

SDU8110B0x PCI Boot Procedure

Drawing Office No. X07-AN-016-04

Copyright © 1996, 1997 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

SDU8110B0X PCI BOOT PROCEDURE

Introduction

The SDU8110B0x PCI Bus Evaluation Board does not contain any BIOS EPROM decode logic. Therefore, it is necessary to use a “Soft” Boot procedure in order to use this board as your primary display adapter. The Boot procedure is required to enable the display using an SDU8110B0x PCI card. If the BIOS is loaded in the Motherboard System ROM, this procedure is not needed. The following list shows the operating systems on which this procedure works:

DOS	:	ok
Windows 3.x	:	ok
Windows 95	:	ok
Windows NT (3.x)	:	no
OS/2 Warp	:	no

As you can see from the above list, OS/2 Warp and Windows NT 3.x cannot be used with the SDU8110B0x PCI evaluation board. For the operating systems that do work, follow the Boot Procedure below.

Boot Procedure

The SDU8110B0x PCI Boot procedure is as follows:

1. Use a third party video board to boot up your system.
2. Copy the file loadrom.exe and the BIOS binary image to a subdirectory on your hard drive.
3. Modify your CONFIG.SYS file by placing the following lines at the beginning:

   device=drive:\path\loadrom.exe drive:\path\filename.bin

   e.g. device=c:\bin\loadrom.exe c:\h122pci.bin

   Where drive:\path is the hard drive and subdirectory in which you have placed the files (i.e., c:\bin) and filename.bin is the BIOS binary boot image file (i.e. h122pci.bin).

   LOADROM.EXE reads the BIOS image file specified on the command line and attempts to load it in RAM as though it were a ROM. The video interrupt pointer will then point to the BIOS image in RAM instead of the default ROM location C000h segment. Note that LOADROM.EXE should be one of the first things in your CONFIG.SYS file and must be loaded prior to the mouse driver.
4. Shut down your system, remove the 3rd party video card and install the SDU8110B0x PCI bus evaluation board in your system.
5. Restart your system.

Comments

- LOADROM.EXE determines the physical address where the boot image will be placed in memory based on availability.
- Any program which takes the VGA BIOS interrupt must be loaded after LOADROM.EXE. When the BIOS is loaded it trashes any existing VGA BIOS interrupt pointers. Therefore, the device which grabbed the interrupt will not operate correctly.
- Some very old software detects the presence of VGA by looking for the BIOS at address C000h. In this case the software will not detect the BIOS image and thus will not work correctly.
- Since LOADROM.EXE places the BIOS image in read/write memory, there is a possibility that the BIOS image can be over-written by external software.
- Since the BIOS is not loaded in normal location (C000h), and BIOS is loaded late in the boot process in RAM not ROM, the graphics segments A000-BFFF may be re-assigned by the chipset or memory manager prior to LOADROM being loaded. In this rare case, there may be memory conflicts in the system.
- Since LOADROM.EXE is a DOS based TSR and Device Driver, it cannot work with Windows NT 3.x or OS/2 Warp.

Limitations

Windows 95 — Because the system is already starting to boot before the device driver gets loaded, you will not see the starting Windows 95 message or the opening Windows 95 logo.

System Setup — You will not be able to use this procedure to modify your system CMOS setup. You will have to put a 3rd party graphics card in your system to modify system CMOS settings.

S-MOS SYSTEMS
A Seiko Epson Affiliate

SPC8110F0A Color LCD/CRT VGA Controller

External Reference Component Values

Drawing Office No. X07G-G-002-01

Copyright © 1996 S-MOS Systems Inc. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of S-MOS Systems Inc. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with S-MOS Systems Inc. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by S-MOS Systems Inc. This document does not necessarily describe any current planned or future product by S-MOS Systems Inc., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

Table of Contents

1 External PLL Reference Values 5

2 Video DAC Reference Current 6

List of Figures

Figure 1: Resistor And Capacitor Loop Values For SPC8110F0A 5

Figure 2: Components For SPC8110F0A IREF Input. 6

List of Tables

Table 1-1: Parts Required For PLL External Loop 5

Table 2-1: Parts Required For IREF Input 6

THIS PAGE LEFT BLANK

1 External PLL Reference Values

The internal PLL on the SPC8110F0A requires a loop resistance of 70 ohms and a capacitance of 0.1 microfarads on each loop input (pins 143 and 134) as shown in Figure 1.

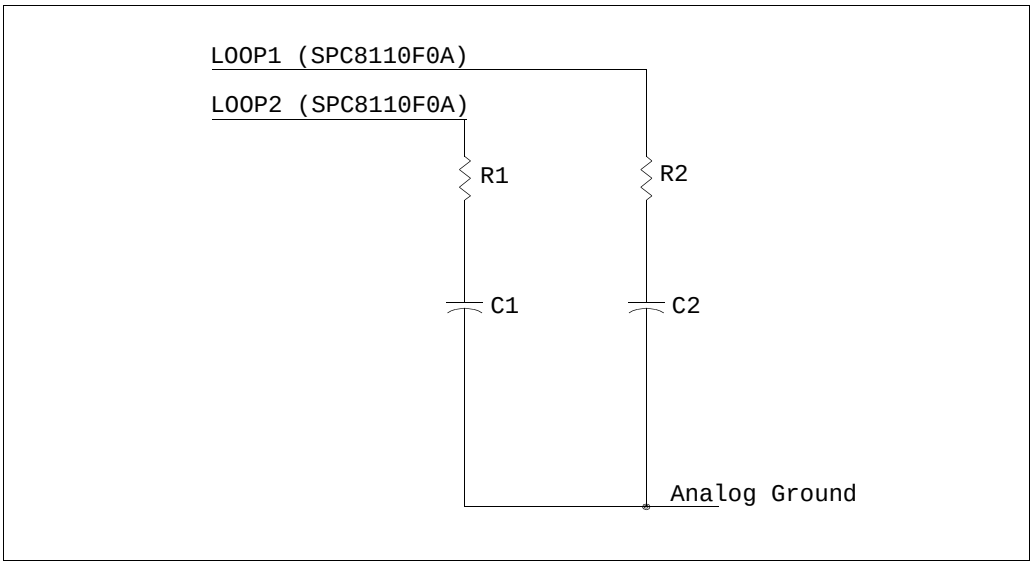


Figure 1: Resistor And Capacitor Loop Values For SPC8110F0A

Table 1-1: Parts Required For PLL External Loop

Item #	Designation	Part Value	Description
1	R1, R2	70Ω	70Ω, 1%, 0.1W Resistor
2	C1, C2	0.1 uF	Monolithic Ceramic Capacitor

2 Video DAC Reference Current

The SPC8110F0A requires a Video DAC reference current on pin 142 (IREF) of 3.3 mA. The Figure below shows the components and Table 2-1 shows the values used to obtain this reference current.

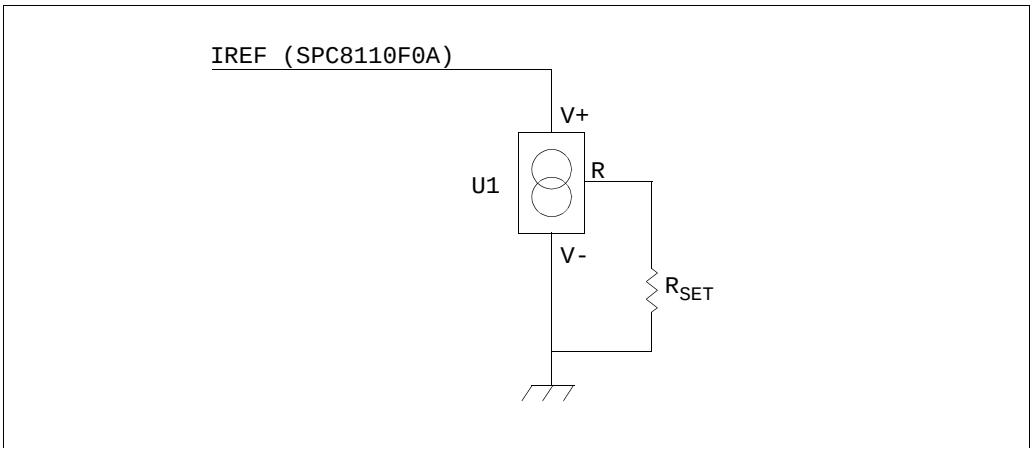


Figure 2: Components For SPC8110F0A IREF Input

Table 2-1: Parts Required For IREF Input

Item #	Designation	Part Value	Description
1	R_{SET}	20Ω	20Ω, 1%, 0.1W Resistor
2	U1	LM334	3-Terminal Adjustable Current Source

EPSON

SPC8110 VGA LCD/CRT Controller

Planar EL640.480-AA1 Panel Interface

Drawing Office Number: X07M-G-001-02

Copyright © 1997, Seiko Epson Corp. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of Seiko Epson Corp. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with Seiko Epson Corp. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by Seiko Epson Corp. This document does not necessarily describe any current planned or future product by Seiko Epson Corp., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

Table Of Contents

1 Introduction 5
2 Display 6
3 SPC8110 Configuration 7
3.1 Palette Programming 8
3.2 Register Settings 9
4 Windows 3.1 Driver 12
5 Conclusion 13

List Of Tables

Table 2-1: Planar L640.480-AA1 Color Mapping 6
Table 3-1: Pin Mapping. 7
Table 3-2: 9bpp TFT Internal Palette Mapping. 8
Table 3-3: Panel Color Mapping 8

THIS PAGE LEFT BLANK

1 Introduction

This document describes the hardware/software environment necessary to interface the SPC8110 LCD/CRT Controller to the Planar EL640.480-AA1 Multicolor Display.

Contact Planar America, Inc., 1400 NW Compton Drive, Beaverton, Oregon 97006, or telephone (503) 690-1100 for information concerning Planar products. Refer to the SPC8110 Hardware Functional Specification, Drawing Office No. X07-SP-001-xx for further information on the SPC8110.

2 Display

The EL640.480-AA1 display is a VGA compatible, high-resolution, electroluminescent display. It is capable of displaying 8 different hues of red, green, and yellow with excellent image clarity.

The panel supports three different “Display Operation Modes;” for the purpose of this application the display must be configured in “TFT LCD Mode.” In this mode, the panel data format and AC timing requirements are compatible with the SPC8110 TFT output.

The most important aspect of this interface is the data mapping between the controller and display. The panel has a 4bpp data format with a very specific “Color Mapping” table. The actual data output pins used from the SPC8110 are not important and only affect the subsequent programming of the SPC8110 palette (or Look-Up-Table).

Table 2-1: Planar L640.480-AA1 Color Mapping

D3	D2	D1	D0	AA1 Color
0	0	0	0	Black
0	0	0	1	Black
0	0	1	0	Green
0	0	1	1	Green
0	1	0	0	Red
0	1	0	1	Red
0	1	1	0	Yellow
0	1	1	1	Lt. Yellow
1	0	0	0	Yellow
1	0	0	1	Yellow
1	0	1	0	Lt. Green
1	0	1	1	Green-Yellow
1	1	0	0	Lt. Red
1	1	0	1	Orange
1	1	1	0	Lt. Yellow
1	1	1	1	Lt. Yellow

3 SPC8110 Configuration

The SPC8110 supports a variety of display modes and types. For the purpose of this application it will be configured to support a 640x480, 16 color (Mode 12), 9bpp TFT panel. Refer to the SPC8110 Hardware Functional Specification for complete register descriptions.

In TFT display modes, the SPC8110 data output comes directly from the internal 18 bit palette (LUT). As stated above, the SPC8110 is configured for a 9bpp TFT panel and due to the 4bpp panel data format, only 4 bits of the available 9 are used. The actual data pins used only affect the programing of the palette to guarantee correct “Color Mapping.”

The table below shows the actual pin mapping between the controller and the display.

Table 3-1: Pin Mapping

SPC8110 Pin #/Name	**EL640.480-AA1 Pin Name (#)**	**Description**
YD (4)	VS (24	Vertical Sync
LP (164)	HS (22)	Horizontal Sync
WF (189)	_BLANK (20)	(Data Enable) Frames data
XSCL (163)	VCLK (18)	Shift Clock
UD6 (Green bit 1)	D0	Data bit 0
UD7 (Green bit 2)	D1	Data bit 1
LD0 (Red bit 1)	D2	Data bit 2
LD1 (Red bit 2)	D3	Data bit 3
	VMODE	must be tied low
	ENABLE	NC

Only control signals are shown in Table 3-1. For all other connections (i.e. Power/Ground) refer to the Planar panel specification.

3.1 Palette Programming

The following is a brief description of the SPC8110 palette organization; for a full description refer to the Hardware Functional Specification.

The internal palette is 18-bits wide (6bpp/primary color; Red, Green, Blue); however in 9bpp TFT mode there are only 3bpp, therefore the msb's are used with the lsb's "don't care."

Table 3-2: 9bpp TFT Internal Palette Mapping

D5	D4	D3	D2	D1	D0		D5	D4	D3	D2	D1	D0		D5	D4	D3	D2	D1	D0
R2	**R1**	**R0**	**XX**	**XX**	**XX**		**G2**	**G1**	**G0**	**XX**	**XX**	**XX**		**B2**	**B1**	**B0**	**XX**	**XX**	**XX**
D3	D2						D1	D0											

As stated above the SPC8110 is operating in a 16 color mode. In this mode data from memory is used by the Attributes Controller to select an "index" into the palette. The value stored in the palette location is then output to the LCD. In this example, the Attributes Controller is programmed in a linear fashion so it indexes palette locations 00h - 0Fh.

Table 2-1 shows that the Planar panel maps 16 possible data bit combinations to display 9 different colors. The following example shows the palette contents of the first 16 locations required to match the "Color Mapping" of the display.

Table 3-3: Panel Color Mapping

Palette Location	Red R2=D3 R1=D2	Green G2=D1 G1-D0	Blue (not used)	Resulting Planar Color
0	00h	00h	00h	Black
1	00h	10h	00h	Black
2	00h	20h	00h	Green
3	00h	30h	00h	Green
4	10	00h	00h	Red
5	10h	10h	00h	Red
6	10h	20h	00h	Yellow
7	10h	30h	00h	Lt. Yellow
8	20h	00h	00h	Yellow
9	20h	10h	00h	Yellow
0Ah	20h	20h	00h	Lt. Green
0Bh	20h	30h	00h	Green-Yellow
0Ch	30h	00h	00h	Lt. Red
0Dh	30h	10h	00h	Orange
0Eh	30h	20h	00h	Lt. Yellow
0Fh	30h	30h	00h	Lt. Yellow

Note

Table 3-2 and 3-3 show that the desired data output pattern is achieved by determining which SPC8110 data output bits were used and then programming the palette accordingly.

3.2 Register Settings

Note

All register settings reflect a Mode 12, 640x480x16 color 9bpp TFT setting. There have been some slight CRTC adjustments to accommodate for this panel's particular AC Timing requirements.

CRT controller Registers Set A:

```
CRT index = 13
CRTC A[00]=5f    CRTC A[01]=4f    CRTC A[02]=50    CRTC A[03]=82    CRTC A[04]=54
CRTC A[05]=80    CRTC A[06]=0b    CRTC A[07]=3e    CRTC A[08]=00    CRTC A[09]=40
CRTC A[0a]=00    CRTC A[0b]=00    CRTC A[0c]=00    CRTC A[0d]=00    CRTC A[0e]=00
CRTC A[0f]=00    CRTC A[10]=ea    CRTC A[11]=8c    CRTC A[12]=df    CRTC A[13]=28
CRTC A[14]=00    CRTC A[15]=e7    CRTC A[16]=04    CRTC A[17]=e3    CRTC A[18]=ff
CRTC A[24]=00    CRTC A[30]=00    CRTC A[31]=32    CRTC A[32]=00    CRTC A[33]=45
CRTC A[34]=00    CRTC A[35]=20    CRTC A[36]=00    CRTC A[37]=9c    CRTC A[38]=02
```

CRT controller Registers Set B:

```
CRTC B[00]=1e    CRTC B[01]=50    CRTC B[06]=e1    CRTC B[07]=05    CRTC B[10]=e1
CRTC B[11]=01    CRTC B[12]=78
```

Sequencer Registers Set A:

```
SEQ index = 02
SEQ A[00]=03     SEQ A[01]=01     SEQ A[02]=0f     SEQ A[03]=00     SEQ A[04]=06
```

Graphics Controller Registers:

```
GFX index = 01
GFX[00]=0f       GFX[01]=00       GFX[02]=00       GFX[03]=00       GFX[04]=03
GFX[05]=00       GFX[06]=05       GFX[07]=0f                        GFX[08]=ff
```

Attribute Controller Registers set A:

```
Attrib index = 31  Palette enabled
ATTR A[00]=00    ATTR A[01]=0c    ATTR A[02]=0a    ATTR A[03]=0e    ATTR A[04]=01
ATTR A[05]=15    ATTR A[06]=23    ATTR A[07]=07    ATTR A[08]=0f    ATTR A[09]=24
ATTR A[0a]=12    ATTR A[0b]=36    ATTR A[0c]=09    ATTR A[0d]=2d    ATTR A[0e]=1b
ATTR A[0f]=3f    ATTR A[10]=01    ATTR A[11]=00    ATTR A[12]=0f    ATTR A[13]=00
ATTR A[14]=00
```

General Registers:

```
Input Status Register 0 = 10
Input Status Register 1 = 04
Miscellaneous Output Register = e3
Video Subsystem Enable Register (3c3) = 01
Page Register: 00
```

BitBlt registers (port 3D0 to 3D3):

```
BitBlt index = 04
BitBlt[00]=00000200    BitBlt[01]=0093a201    BitBlt[02]=00000002
BitBlt[03]=0007e003    BitBlt[04]=00000204    BitBlt[05]=00000005
```

Auxiliary Registers:

```
AUX index = 05 (AUX enabled)
AUX[00]=60        AUX[01]=00        AUX[02]=f7        AUX[03]=6a        AUX[04]=01
AUX[05]=21        AUX[06]=00        AUX[07]=07        AUX[08]=e0        AUX[0a]=10
AUX[0b]=cf        AUX[0c]=00        AUX[0d]=0c        AUX[0f]=a2        AUX[10]=00
AUX[11]=00        AUX[12]=04        AUX[13]=ff        AUX[14]=ff        AUX[15]=16
AUX[16]=28        AUX[17]=00        AUX[1a]=00        AUX[1b]=00        AUX[1c]=06
AUX[1d]=38        AUX[1e]=39        AUX[1f]=80        AUX[20]=e7        AUX[21]=04
AUX[22]=c2        AUX[23]=01        AUX[24]=00        AUX[25]=00        AUX[26]=4b
AUX[27]=04        AUX[28]=07        AUX[34]=aa        AUX[38]=15        AUX[39]=25
AUX[3a]=04        AUX[3b]=14        AUX[3c]=36        AUX[de]=1a
```

Sprite Palettes are:

```
entry 0:   00    00    00
entry 1:   3f    3f    3f
entry 2:   2d    37    27
entry 3:   3e    37    3f
```

Interpretation of the Auxiliary registers:

- B Set Registers disabled
- LCD display hardware enabled
- CRT display hardware enabled
- Autocenter disabled
- Single Color TFT panel
- Self-Refresh DRAM
- 512k of Video RAM
- Memory Paging enabled
- Linear Addressing disabled
- Graphics Controller Bypass disabled
- Sprite/HW Cursor enabled
- Fast Text Mode disabled
- High Resolution 256 Color Mode Select is disabled
- VL-BUS/Intel486 Bus Interface
- Text Mode Vertical Expansion enabled
- Graphics Mode Vertical Expansion enabled
- PSM1 hardware power down enabled
- PSM4 hardware power down disabled
- PLL1 Set0 is set to 25.056Mhz
- PLL1 Set1 is set to 28.636Mhz
- PLL1 Set2 is set to ??.???
- PLL2 is set to 39.374Mhz

4 Windows 3.1 Driver

Epson has developed an SPC8110, 16 color Windows 3.1 Driver to demonstrate this panel interface. The driver automatically maps the standard 16 colors into those required by this panel having the connections as described above.

Driver Name: W31H10.DRV version 1.02C1.

Release Name: PL640480.ZIP

5 Conclusion

The SPC8110 can easily interface to the Planar 640.480-AA1. Although this application note and associated information are specific to the SPC8110, other Epson LCD Controllers will also provide a compatible interface.

Refer to the SPC8110 Hardware Functional Specification along with the Planar EL640.480-AA1 Product Profile for further details.

THIS PAGE LEFT BLANK

EPSON

SPC8110 VGA LCD/CRT Controller

Optrex DMF-50418NF-FW Panel Interface

Drawing Office Number: X07M-G-002-01

Copyright © 1997, Seiko Epson Corp. All rights reserved.

This document, and any text derived, extracted or transmitted from it, is the sole property of Seiko Epson Corp. and may not be used, copied, transmitted or otherwise made available to any other person, unless specifically allowed under license agreement with Seiko Epson Corp. If furnished under a license, this document may be used and copied only in accordance with the terms of such license, and with the inclusion of this copyright notice. No title or ownership of the technology herein is transferred.

The information in this document is subject to change without notice and should not be construed as a commitment by Seiko Epson Corp. This document does not necessarily describe any current planned or future product by Seiko Epson Corp., nor does it represent any commitment to implement any such product.

THIS PAGE LEFT BLANK

Table of Contents

1 Introduction 5
2 Display 6
3 SPC8110 Configuration 8
4 Register Settings 9
5 Conclusion 12
6 Parts List 13
7 Schematic Diagram 14

List of Figures

Figure 2-1: Interface Circuitry Functionality 6
Figure 7-1: Optrex DMF-50418NF-FW Panel Interface Circuit Schematic 14

List of Tables

Table 2-1: DMF-50418NF-FW Timing Specifications and Interface Board Timing Comparison 7

THIS PAGE LEFT BLANK

1 Introduction

This document contains the necessary hardware information to interface the SPC8110 LCD/CRT Controller to the Optrex DMF-50418NF-FW Monochrome Display.

Contact Optrex America Inc., 44160 Plymouth Oaks Blvd., Plymouth MI 48170, or telephone (313) 416-8500 for information concerning Optrex products. Refer to the SPC8110 Hardware Functional Specification, Drawing Office No. X07-SP-001-xx for further information concerning the SPC8110.

2 Display

The Optrex DMF-50418NF-FW is a high resolution (1024x768) NTN monochrome display.

The SPC8110 does not directly support this panel, however by using minimal external logic a successful interface can be achieved.

Logic Description

The SPC8110 is actually configured in 8-bit dual monochrome LCD mode. External circuitry is necessary to provide the 16 bits required by the panel (LD[7..0],UD[7..0]) on the falling edge of the panel clock. The external circuitry doubles the period of the XSCL signal provided by the SPC8110, the doubled XSCL is then used as the panel clock. The upper nibbles (LD[7..4],UD[7..4]) of the display data provided by the SPC8110 are latched on the rising edge of the panel clock. On the falling edge of the panel clock, the lower nibbles (LD[3..0],UD[3..0]) are on their respective pins. All 16 bits needed by the panel are available as specified on the falling edge of the panel clock.

Refer to the timing diagram below and Figure 7-1: “Optrex DMF-50418NF-FW Panel Interface Circuit Schematic,” on page 14 for further details.

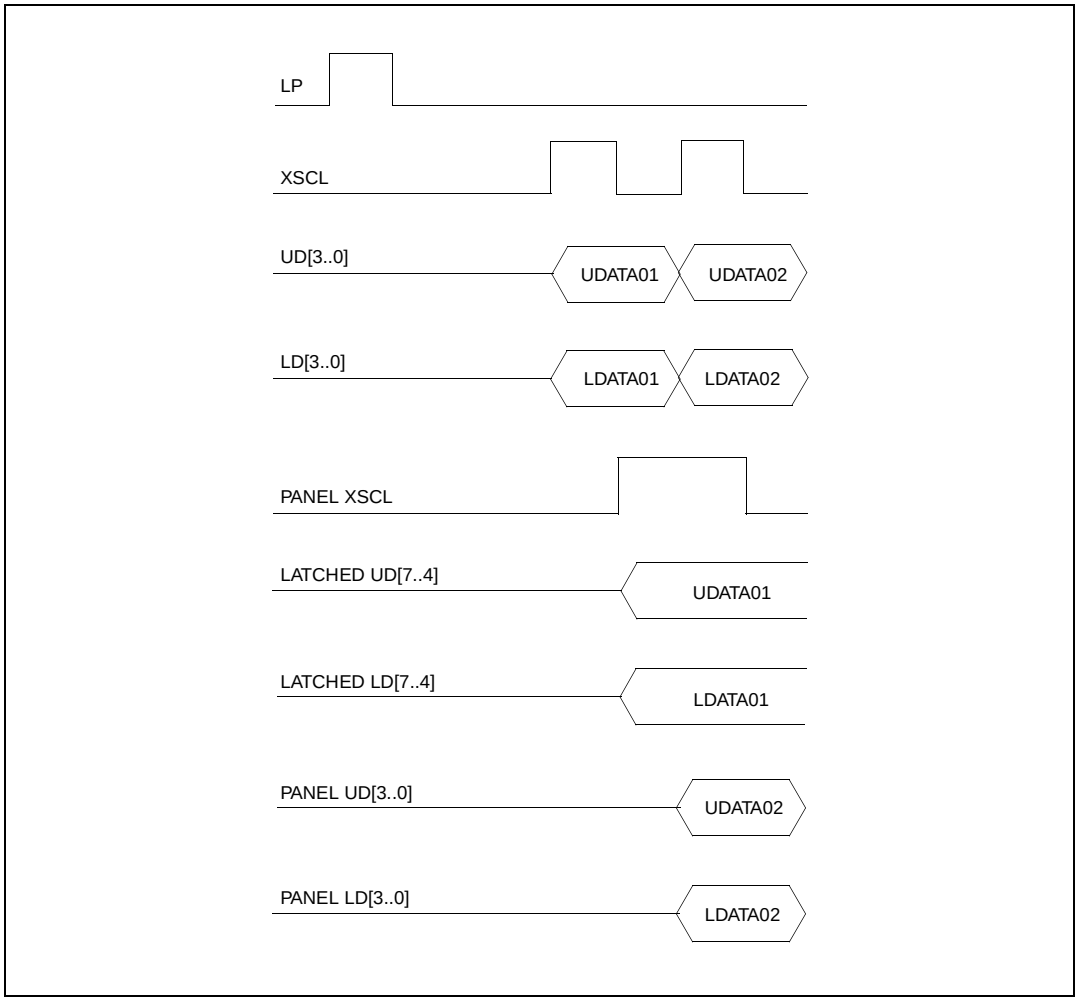


Figure 2-1: Interface Circuitry Functionality

Table 2-1: DMF-50418NF-FW Timing Specifications and Interface Board Timing Comparison

ITEM	MEASURED SIGNALS	PANEL SPEC. MAX.	PANEL SPEC MIN.	UNIT
XSCL	282	-	153	ns
LP CLOCK CYCLE	3966	-	500	ns
LP “H’ LEVEL PULSE WIDTH	100	-	50	ns
XSCL “H’ LEVEL PULSE WIDTH	152	-	40	ns
XSCL “L” LEVEL PULSE WIDTH	130	-	40	ns
XSCL SETUP TIME	110	-	30	ns
INPUT SIGNAL RISE/FALL TIME	26	30	-	ns
DATA SET UP TIME	94	-	50	ns
DATA HOLD TIME	43	-	50	ns
LATCHED DATA SET UP TIME	116	-	50	ns
LATCHED DATA HOLD TIME	148	-	50	ns
DIM SET UP TIME	3942	-	100	ns
DIM HOLD TIME	197	-	30	ns

3 SPC8110 Configuration

The SPC8110 supports a variety of display modes and types. To make full use of the panel viewing area, the SPC8110 will be configured as 1024x768, 16 gray scale (Mode 104). This is achieved by using the 8-bit monochrome LCD default pin configuration.

A custom Bios must be used in order to set-up the SPC8110 registers properly for the different video modes. Refer to “Register Settings” on page 9 for complete details.

See Figure 7-1: “Optrex DMF-50418NF-FW Panel Interface Circuit Schematic,” on page 14 for the hardware and interface signals between the SPC8110, DMF-50418NF-FW, and the DMF-50418NF-FW Interface Board.

4 Register Settings

Note: All registers reflect a Mode 104, 1024x768, 16 gray scale 8-bit monochrome LCD setting.

CRT controller Registers Set A:

```
CRT index = 0f
CRTC A[00]=a3   CRTC A[01]=7f   CRTC A[02]=80   CRTC A[03]=06
CRTC A[04]=82   CRTC A[05]=93   CRTC A[06]=24   CRTC A[07]=fd
CRTC A[08]=00   CRTC A[09]=60   CRTC A[0a]=00   CRTC A[0b]=00
CRTC A[0c]=00   CRTC A[0d]=00   CRTC A[0e]=01   CRTC A[0f]=80
CRTC A[10]=02   CRTC A[11]=88   CRTC A[12]=ff   CRTC A[13]=40
CRTC A[14]=00   CRTC A[15]=02   CRTC A[16]=22   CRTC A[17]=e3
CRTC A[18]=ff   CRTC A[24]=00   CRTC A[30]=00   CRTC A[31]=00
CRTC A[32]=00   CRTC A[33]=00   CRTC A[34]=00   CRTC A[35]=00
CRTC A[36]=00   CRTC A[37]=00   CRTC A[38]=00
```

CRT controller Registers Set B:

```
CRTC B[00]=1e   CRTC B[01]=80   CRTC B[06]=81   CRTC B[07]=01
CRTC B[10]=eb   CRTC B[11]=0d   CRTC B[12]=c0
```

Sequencer Registers Set A:

```
SEQ index = 02
SEQ A[00]=03    SEQ A[01]=01    SEQ A[02]=0f    SEQ A[03]=00
SEQ A[04]=06
```

Graphics Controller Registers:

```
GFX index = 03
GFX[00]=00      GFX[01]=00      GFX[02]=00      GFX[03]=00
GFX[04]=00      GFX[05]=00      GFX[06]=05      GFX[07]=0f
GFX[08]=ff
```

Attribute Controller Registers set A:

```
Attrib index = 20 Palette enabled
ATTR A[00]=00   ATTR A[01]=01   ATTR A[02]=02   ATTR A[03]=03
ATTR A[04]=04   ATTR A[05]=05   ATTR A[06]=14   ATTR A[07]=07
ATTR A[08]=38   ATTR A[09]=39   ATTR A[0a]=3a   ATTR A[0b]=3b
ATTR A[0c]=3c   ATTR A[0d]=3d   ATTR A[0e]=3e   ATTR A[0f]=3f
ATTR A[10]=01   ATTR A[11]=00   ATTR A[12]=0f   ATTR A[13]=00
ATTR A[14]=00
```

General Registers:

```
Input Status Register 0 = 00
Input Status Register 1 = 04
Miscellaneous Output Register = eb
Video Subsystem Enable Register (3c3) = 01 Page Register: 00
```

BitBlt registers (port 3D0 to 3D3) :

```
BitBlt index = 00    BitBlt[00]=00000000  BitBlt[01]=00000001
BitBlt[02]=00000002  BitBlt[03]=00000003  BitBlt[04]=00000004
BitBlt[05]=00000005
```

Auxiliary Registers:

```
AUX index = 05 (AUX enabled)
AUX[00]=30   AUX[01]=42   AUX[02]=48   AUX[03]=08
AUX[04]=9c   AUX[05]=01   AUX[06]=20   AUX[07]=00
AUX[08]=e7   AUX[0a]=48   AUX[0b]=03   AUX[0c]=8c
AUX[0d]=00   AUX[0f]=a2   AUX[10]=00   AUX[11]=03
AUX[12]=03   AUX[13]=b7   AUX[14]=b7   AUX[15]=0c
AUX[16]=4f   AUX[17]=01   AUX[1a]=00   AUX[1b]=20
AUX[1c]=0f   AUX[1d]=2f   AUX[1e]=fd   AUX[1f]=00
AUX[20]=e2   AUX[21]=01   AUX[22]=c2   AUX[23]=01
AUX[24]=e2   AUX[25]=01   AUX[26]=4e   AUX[27]=05
AUX[28]=03   AUX[34]=00   AUX[38]=14   AUX[39]=25
AUX[3a]=14   AUX[3b]=14   AUX[3c]=14   AUX[de]=1a
```

Sprite Palettes are:

```
entry 0: 00 00 00  entry 1: 14 14 14   entry 2: 28 28 28
entry 3: 3f 3f 3f
```

Interpretation of the Auxiliary registers:

- B Set Registers disabled
- LCD display hardware enabled
- CRT display hardware disabled
- Autocenter enabled
- 8 ms refresh rate for Symmetrical DRAM
- 1024k of Video RAM Memory
- Paging enabled
- Linear Addressing disabled
- Graphics Controller Bypass disabled
- Sprite/HW Cursor disabled
- Fast Text Mode disabled
- High Resolution 256 Color Mode Select is disabled
- VL-BUS/Intel486 Bus Interface
- Text Mode Vertical Expansion disabled
- Graphics Mode Vertical Expansion disabled
- PSM1 hardware power down disabled
- PSM4 hardware power down disabled
- PLL1 Set0 is set to 28.636Mhz
- PLL1 Set1 is set to 28.636Mhz
- PLL1 Set2 is set to 28.636Mhz
- PLL2 is set to 40.090Mhz

5 Conclusion

The SPC8110 can be easily interfaced with minimal external circuitry to the Optrex DMF-50418NF-FW. Refer to the SPC8110 Hardware Functional Specification as well as the Optrex Product Specifications for further details.

6 Parts List

Item#	Quantity	Designation	Part Value	Description
1	3	C1-C3	0.1uF	0.1uF mono
2	1	U1	74LS374	Octal D-Type Flip-Flop
3	1	U2	74LS74	Dual D-Type Flip-Flop
4	1	U3	74LS04	Hex Inverter

7 Schematic Diagram

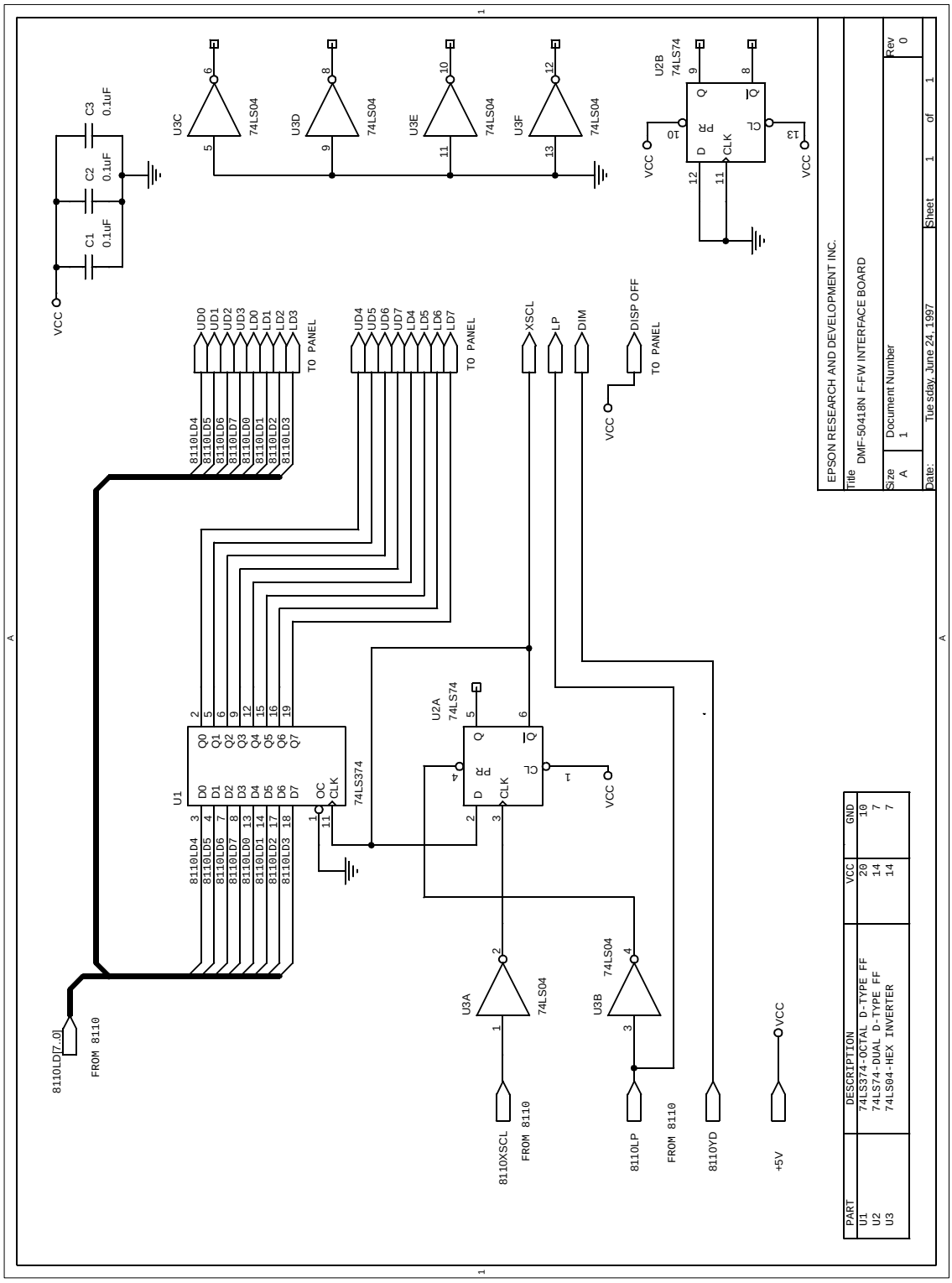


Figure 7-1: Optrex DMF-50418NF-FW Panel Interface Circuit Schematic