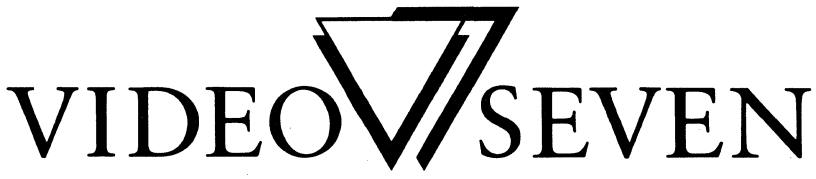
V7 VGA

Technical Reference Manual

6/30/88



We make a clear difference.

Headland

Technology Inc

V7VGA

Technical Reference Manual

6/30/88

Written by:

Dean A. Hays
Video Seven
Cupertino, California

with contributions from:

Michael Abrash
Dave Miller
Tom Wilson

Video Seven Incorporated
46335 Landing Parkway
Fremont, California 94538

(415) 656-7800

Table of Contents

(continued)

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Scope of Document

This manual provides technical coverage of the V7 VGA chip. Topics include the modes of operation of the V7 VGA, the major components and registers, the BIOS functions, and programming information including examples. In addition, detailed information is presented on each of the V7 VGA registers.

The following definitions are used throughout this manual:

EGA	Industry Standard Enhanced Graphics Adapter.
CGA	Industry Standard Color Graphics Display Adapter.
MDA	Industry Standard Monochrome Display Adapter.
MGA	Video-Seven Monochrome Graphics Adapter.
HGC	Hercules™ Graphics Card (MGA Compatible).
ED/ECD	Industry Standard Enhanced Color Display.
CD	Industry Standard Color Display.
MD	Industry Standard Monochrome Display.
AD/ACD	Analog Display / Analog Color Display
MS	Multiple-Frequency Color Display (MultiSync™, MultiScan™, etc.)

1.2 Chip Revisions Covered

This manual documents the following chip revisions:

Revision 0 - Available Jan 1988	(L1Axxxx)
Revision 1 - Available Mar 1988	(L1A4080)
Revision 2 - Available Apr 1988	(L1A4152)
Revision 3 - Available May 1988	(L1A4199)

1.3 Intended Audience

This manual is directed toward the technically sophisticated audience. It assumes the reader is familiar with assembly language programming on the 80286 or similar microprocessor, and understands the fundamentals of video display terminology.

CHAPTER

2

Overview

The V7VGA chip is hardware / software compatible with the IBM VGA and provides improved performance and additional functionality. The V7VGA supports high resolution graphics and alphanumeric display modes for both monochrome and color, and for high resolution displays such as the NEC Multisync™ and Sony MultiScan™ monitors. Additional functions are also available, including hardware cursor, high-resolution 256-color modes, extended attribute control, masked-write capability (with V-Rams), and extensive enhancements to the ability to manipulate foreground / background data patterns.

The V7VGA implements all control and data registers of the VGA. It also implements all of the data manipulation capabilities and data paths of the VGA.

All memory cycles not used to refresh the display or video memory are allocated to process CPU memory requests. A one-to-one interleave can be achieved between CPU and display refresh accesses to the video memory for typical VGA-compatible modes using V-Rams. Various interleave settings from 1:1 to 1:4 may be programmed depending on dot clock frequency, mode of operation, and ram access time.

Hardware support is provided to display a 32x32 pixel pattern for use as a mouse pointer. Additional text cursor controls are provided (blink disable and OR/XOR mode control).

2.1 Features

The following is a list of the major features of the V7VGA:

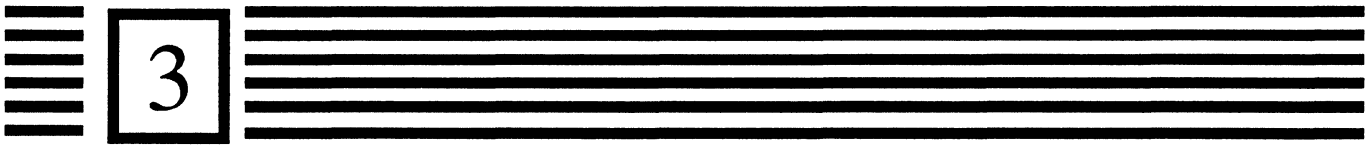
- Fast host access to video memory: every cycle for all VGA standard modes using V-Rams
- 32-bit video ram access for display refresh
- 16-bit CPU access (memory, I/O, and BIOS)
- High speed operation (up to 65 MHz dot clock rate)
- Increased video resolutions:

  Text: up to 132 columns or 60 lines
  8 or 9-bit character widths

  Graphics: up to 1024x768 four bit/pixel
  and 800x600 eight bit/pixel

- Allows up to eight fonts to be loaded simultaneously (any two selectable at once)
- Hardware graphics cursor (mouse pointer)
- Supports memory configurations of 256 KB, 512 KB or 1 MB using either 256Kb or 1Mb ram chips
- Supports both D-Ram and V-Ram configurations
- Supports multiple monitor types including NEC MultiSync™ and Sony MultiScan™.
- Host CPU-readable registers for save/restore
- Scan-line doubling for improved readability of 200-line graphics modes on high-resolution monitors
- Data path enhancements for foreground / background data manipulation
- Extended underlining control
- Masked-write capability (with V-Rams)

CHAPTER



Register Definitions

The following sections present information on the registers of the V7VGA chip. The register name is shown first (full name and an abbreviation), along with the port address and index if applicable. Below that is a figure illustrating the register configuration and a discussion of the register contents.

Note: In all register descriptions in this chapter, unused and reserved bits return 0 when read, except where specifically indicated otherwise. One general exception is that all reserved extension registers read back bits 3-0 as 1.

V7VGA I/O PORT SUMMARY

Port Address	VGA Mode			
2B0 / 3B0				
2B1 / 3B1				
2B2 / 3B2				
2B3 / 3B3				
2B4 / 3B4	CRTC Index (RW)			
2B5 / 3B5	CRTC Data (RW)			
2B6 / 3B6				
2B7 / 3B7				
2B8 / 3B8				
2B9 / 3B9				
2BA / 3BA	Feature Ctrl(W), Display Status(R)			
2BB / 3BB				
2BC / 3BC				
2BD / 3BD				
2BE / 3BE				
2BF / 3BF				
2C0 / 3C0	Attr Ctrlr Index/Data (W), Index (R)			
2C1 / 3C1	Attr Ctrlr Index/Data (W), Data (R)			
2C2 / 3C2	Misc Output (W), Feature (R)			
2C3 / 3C3	Video Subsystem Enable (RW)			
2C4 / 3C4	Sequencer/Extensions Index (RW)			
2C5 / 3C5	Sequencer/Extensions Data (RW)			
2C6 / 3C6	Palette Pixel Mask (RW)			
2C7 / 3C7	Palette Index Rmode(W),DACstate(R)			
2C8 / 3C8	Palette Index Wmode (RW)			
2C9 / 3C9	Palette Data (RW)			
2CA / 3CA	Feature Control (R)			
2CB / 3CB				
2CC / 3CC	Misc Output (R)			
2CD / 3CD				
2CE / 3CE	Graphics Controller Index (RW)			
2CF / 3CF	Graphics Controller Data (RW)			
2D0 / 3D0				
2D1 / 3D1				
2D2 / 3D2				
2D3 / 3D3				
2D4 / 3D4	CRTC Index (RW)			
2D5 / 3D5	CRTC Data (RW)			
2D6 / 3D6				
2D7 / 3D7				
2D8 / 3D8				
2D9 / 3D9				
2DA / 3DA	Feature Ctrl(W), Display Status(R)			
2DB / 3DB				
2DC / 3DC				
2DD / 3DD				
2DE / 3DE				
2DF / 3DF				

VGA COMPATIBLE REGISTERS

PAGE	ABBREV	VGA / EGA REGISTER NAME	BITS	REG TYPE	READ/WRITE	INDEX	MONO PORT	COLOR PORT
3-6	MISC	Miscellaneous Output	8	VGA	RW	--	3C2(W),3CC(R)	3C2(W),3CC(R)
3-7	FC	Feature Control	3	VGA	RW	--	3BA(W),3CA(R)	3DA(W),3CA(R)
3-8	FEAT	Feature Read (Input Status 0)	4	VGA	R	--	3C2	3C2
3-9	STAT	Display Status (Input Status 1)	7	VGA	R	--	3BA	3DA
3-10	DACMASK	Color Palette Pixel Mask	8	DAC	RW	--	3C6	3C6
3-11	DACSTATE	Color Palette State	2	VGA	R	--	3C7	3C7
3-12	DACRX	Color Palette Read-Mode Index	8	DAC	W	--	3C7	3C7
3-13	DACWX	Color Palette Write-Mode Index	8	DAC	RW	--	3C8	3C8
3-14	DACDATA	Color Palette Regs 0-255	18	DAC	RW	00-FF	3C9	3C9
3-15	VSE	Video Subsystem Enable	1	VGA	RW	--	3C3	3C3
3-16	ALTVSE	Video Subsystem Enable (Alt)	1	VGA	RW	--	102	102
3-17	ROMMAP	Video Subsystem Control	5	EXT	W	--	46E8	46E8
3-18	CACHE	I/O Cache	5+8	IOU	RW	0-F	4BC4-4BC5	4BC4-4BC5
3-5	EXPAN	Expansion Connector Ports	8	EXT	RW	--	2E9,AxE8	2E9,AxE8
3-20	SRX	Sequencer / Extensions Index	7	VGA	RW	--	3C4	3C4
3-21	SR0	Reset	2	VGA	RW	00	3C5	3C5
3-22	SR1	Clocking Mode	6	VGA	RW	01	3C5	3C5
3-24	SR2	Plane Mask	4	VGA	RW	02	3C5	3C5
3-25	SR3	Character Map Select	6	VGA	RW	03	3C5	3C5
3-26	SR4	Memory Mode	4	VGA	RW	04	3C5	3C5
3-27	SR6	Extensions Control	1	VGA	RW	06	3C5	3C5
3-28	SR7	Reset Hor Character Counter	0	VGA	W	07	3C5	3C5
3-30	CRX	CRTC Index	6	VGA	RW	--	3B4	3D4
3-31	CR0	Horizontal Total	8	VGA	RW	00	3B5	3D5
3-32	CR1	Horizontal Display End	8	VGA	RW	01	3B5	3D5
3-33	CR2	Horizontal Blanking Start	8	VGA	RW	02	3B5	3D5
3-34	CR3	Horizontal Blanking End	5+2+1	VGA	RW	03	3B5	3D5
3-35	CR4	Horizontal Retrace Start	8	VGA	RW	04	3B5	3D5
3-36	CR5	Horizontal Retrace End	5+2+1	VGA	RW	05	3B5	3D5
3-37	CR6	Vertical Total	8	VGA	RW	06	3B5	3D5
3-38	CR7	Overflow	8	VGA	RW	07	3B5	3D5
3-39	CR8	Screen A Preset Row Scan	5+2	VGA	RW	08	3B5	3D5
3-40	CR9	Character Cell Height	5+3	VGA	RW	09	3B5	3D5
3-41	CRA	Cursor Start	5+1	VGA	RW	0A	3B5	3D5
3-42	CRB	Cursor End	5+2	VGA	RW	0B	3B5	3D5
3-43	CRC	Screen A Start Address High	8	VGA	RW	0C	3B5	3D5
3-44	CRD	Screen A Start Address Low	8	VGA	RW	0D	3B5	3D5
3-45	CRE	Cursor Location High	8	VGA	RW	0E	3B5	3D5
3-46	CRF	Cursor Location Low	8	VGA	RW	0F	3B5	3D5
3-47	LPENH	Light Pen High	8	VGA	R	10	3B5	3D5
3-48	LPENL	Light Pen Low	8	VGA	R	11	3B5	3D5
3-49	CR10	Vertical Retrace Start	8	VGA	W, RW	10	3B5	3D5
3-50	CR11	Vertical Retrace End	4+4	VGA	W, RW	11	3B5	3D5
3-51	CR12	Vertical Display End	8	VGA	RW	12	3B5	3D5
3-52	CR13	Offset	8	VGA	RW	13	3B5	3D5
3-53	CR14	Underline Row Scan	5+2	VGA	RW	14	3B5	3D5
3-54	CR15	Vertical Blanking Start	8	VGA	RW	15	3B5	3D5
3-55	CR16	Vertical Blanking End	8	VGA	RW	16	3B5	3D5
3-56	CR17	CRT Mode Control	7	VGA	RW	17	3B5	3D5
3-58	CR18	Line Compare	8	VGA	RW	18	3B5	3D5
3-59	CR1F	V7VGA Identification	8	VGA	R	1F	3B5	3D5
3-60	CR22	Graphics Ctrlr Data Latches	8	VGA	R	22	3B5	3D5
3-61	CR24	Attribute Ctrlr Index/Data Latch	7	VGA	R	24	3B5	3D5
3-62	CR3x	Clear Vertical Display Enable FF	1	VGA	W	3x	3B5	3D5
3-64	GRX	Graphics Controller Index	4	VGA	RW	--	3CE	3CE
3-65	GR0	Set/Reset	4	VGA	RW	00	3CF	3CF
3-66	GR1	Enable Set/Reset	4	VGA	RW	01	3CF	3CF
3-67	GR2	Color Compare	4	VGA	RW	02	3CF	3CF
3-68	GR3	Data Rotate	5	VGA	RW	03	3CF	3CF
3-69	GR4	Read Map Select	3	VGA	RW	04	3CF	3CF
3-70	GR5	Mode	7	VGA	RW	05	3CF	3CF
3-72	GR6	Miscellaneous	4	VGA	RW	06	3CF	3CF
3-73	GR7	Color Don't Care	4	VGA	RW	07	3CF	3CF
3-74	GR8	Bit Mask	8	VGA	RW	08	3CF	3CF
3-76	ARX	Attribute Controller Index	6	VGA	RW	--	3C0 (3C1)	3C0 (3C1)
3-77	AR0-F	Internal Palette Regs 0-15	8	VGA	RW	00-0F	3C0 (3C1)	3C0 (3C1)
3-78	AR10	Mode Control	7	VGA	RW	10	3C0 (3C1)	3C0 (3C1)
3-80	AR11	Overscan Color	8	VGA	RW	11	3C0 (3C1)	3C0 (3C1)
3-81	AR12	Color Plane Enable	6	VGA	RW	12	3C0 (3C1)	3C0 (3C1)
3-82	AR13	Horizontal Pixel Panning	4	VGA	RW	13	3C0 (3C1)	3C0 (3C1)
3-83	AR14	Color Select	4	VGA	RW	14	3C0 (3C1)	3C0 (3C1)

V7VGA EXTENSION REGISTERS

PAGE	ABBREVIATIONS		V7VGA REGISTER NAME	BITS	REG TYPE	READ/WRITE	INDEX	PORT	RESET
	ER80-82		(TEST, GPOS1-2 in VEGA VGA)	0	--	--	80-82	3C5	--
3-76	ER83	ARX	* Attribute Controller Index	7	EXTENSION	R/W	83	3C5	--
	ER84-86		(WRC,TC,BWC in VEGA VGA)	0	--	--	84-86	3C5	--
	ER87-89		(ROMC,HSPS, FONTC in VEGA VGA)	0	--	--	87-89	3C5	--
	ER8A	SBHP	-reserved- (Screen B H Pixel Pan)	3	EXTENSION	R/W	8A	3C5	xxxxx000
	ER8B	SBPR	-reserved- (Screen B Preset Rowscan)	5	EXTENSION	R/W	8B	3C5	xxx00000
	ER8C	SBSH	-reserved- (Screen B Start Address H)	8	EXTENSION	R/W	8C	3C5	00000000
	ER8D	SBSL	-reserved- (Screen B Start Address L)	8	EXTENSION	R/W	8D	3C5	00000000
3-86	ER8E	REV	V7VGA Revision Code	8	EXTENSION	R	8E	3C5	01110000
3-86	ER8F	REV	V7VGA Revision Code	8	EXTENSION	R	8F	3C5	01110000
	ER90-93		(CR10-11,LPENH/L in VEGA VGA)	0	--	--	90-93	3C5	--
3-87	ER94	PPA	Pointer Pattern Address	8	EXTENSION	R/W	94	3C5	11111111
	ER95		(CADJ in VEGA VGA)	0	--	--	95	3C5	--
	ER96	CW	-reserved- (Caret Width)	8	EXTENSION	R/W	96	3C5	--
	ER97	CH	-reserved- (Caret Height)	8	EXTENSION	R/W	97	3C5	--
	ER98	CXH	-reserved- (Caret H Position High)	3	EXTENSION	R/W	98	3C5	--
	ER99	CXL	-reserved- (Caret H Position Low)	8	EXTENSION	R/W	99	3C5	--
	ER9A	CYH	-reserved- (Caret V Position High)	2	EXTENSION	R/W	9A	3C5	--
	ER9B	CYL	-reserved- (Caret V Position Low)	8	EXTENSION	R/W	9B	3C5	--
3-89	ER9C	PXH	Pointer Horizontal Position High	3	EXTENSION	R/W	9C	3C5	--
3-90	ER9D	PXL	Pointer Horizontal Position Low	8	EXTENSION	R/W	9D	3C5	--
3-91	ER9E	PYH	Pointer Vertical Position High	2	EXTENSION	R/W	9E	3C5	--
3-92	ER9F	PYL	Pointer Vertical Position Low	8	EXTENSION	R/W	9F	3C5	--
3-93	ERA0	GRL0	Graphics Ctrlr Memory Latch 0	8	EXTENSION	R/W	A0	3C5	--
3-94	ERA1	GRL1	Graphics Ctrlr Memory Latch 1	8	EXTENSION	R/W	A1	3C5	--
3-95	ERA2	GRL2	Graphics Ctrlr Memory Latch 2	8	EXTENSION	R/W	A2	3C5	--
3-96	ERA3	GRL3	Graphics Ctrlr Memory Latch 3	8	EXTENSION	R/W	A3	3C5	--
3-97	ERA4	CLK	Clock Select	6	EXTENSION	R/W	A4	3C5	xxx0xxxx
3-98	ERA5	CURS	Cursor Attributes	3	EXTENSION	R/W	A5	3C5	0xxx0xx0
	ERA6-A9		(ISR,SWITCH,NMI1-2 in VEGA VGA)	0	--	--	A6-A9	3C5	--
	ERAA-AC		(unused,NSTAT1-2 in VEGA VGA)	0	--	--	AA-AC	3C5	--
	ERAD-AF		(PG256,CACHE,STATE in VEGA VGA)	0	--	--	AD-AF	3C5	--
	ERB0-BF		(Scratch Registers 0-F in VEGA VGA)	0	--	--	B0-BF	3C5	--
	ERC0-E9		-reserved for V7VGA extensions-	0	--	--	C0-E9	3C5	--
3-99	EREA	SWSTB	** Switch Strobe	0	EXTENSION	W	EA	3C5	--
3-100	EREB	NMICTRL	Emulation Control	8	EXTENSION	R/W	EB	3C5	00000000
3-102	EREC	FGLAT0	Foreground Latch 0	8	EXTENSION	R/W	EC	3C5	--
3-102	ERED	FGLAT1	Foreground Latch 1	8	EXTENSION	R/W	ED	3C5	--
3-102	EREE	FGLAT2	Foreground Latch 2	8	EXTENSION	R/W	EE	3C5	--
3-102	EREF	FGLAT3	Foreground Latch 3	8	EXTENSION	R/W	EF	3C5	--
3-103	ERF0	FFGLD	Fast Foreground Latch Load	8	EXTENSION	R/W	F0	3C5	--
3-104	ERF1	FLLSTATE	Fast Latch Load State	4	EXTENSION	R/W	F1	3C5	--
3-105	ERF2	FBGLD	Fast Background Latch Load	8	EXTENSION	R/W	F2	3C5	--
3-106	ERF3	MWCTRL	Masked Write Control	2	EXTENSION	R/W	F3	3C5	xxxxxxx0
3-107	ERF4	MWMASK	Masked Write Mask	8	EXTENSION	R/W	F4	3C5	--
3-108	ERF5	FBPAT	Foreground/Background Pattern	8	EXTENSION	R/W	F5	3C5	--
3-109	ERF6	RAMBANK	1 Mb RAM Bank Select	8	EXTENSION	R/W	F6	3C5	00000000
3-110	ERF7	SWITCH	Switch Readback	8	EXTENSION	R/W	F7	3C5	--
3-111	ERF8	CLKCTRL	Extended Clock Control	8	EXTENSION	R/W	F8	3C5	xxx00x00
3-113	ERF9	PGSEL	Extended Page Select	1	EXTENSION	R/W	F9	3C5	xxxxxxx0
3-114	ERFA	FGCOLOR	Extended Foreground Color	4	EXTENSION	R/W	FA	3C5	--
3-115	ERFB	BGCOLOR	Extended Background Color	4	EXTENSION	R/W	FB	3C5	--
3-116	ERFC	COMPAT	Compatibility Control	8	EXTENSION	R/W	FC	3C5	0000x0x0
3-119	ERFD	TIMING	Extended Timing Select	8	EXTENSION	R/W	FD	3C5	00000000
3-121	ERFE	FBCTRL	Foreground/Background Control	3	EXTENSION	R/W	FE	3C5	xxxx00xx
3-122	ERFF	16BIT	16-Bit Interface Control	8	EXTENSION	R/W	FF	3C5	xxx00000

* Duplicated VGA/EGA register also accessible as extension register for state save/restore

** Writes to this register cause the dipswitch settings on bus bits 15-8 to be strobed into the Switch Readback register (ERF7)

Miscellaneous Registers

6/20/88

The miscellaneous registers are those not contained in the other major functional blocks (Sequencer, Attribute Controller, Graphics Controller, and CRT Controller).

Page	Abbreviation	Register Name	Port Addresses
3-6	MISC	Misc Output Register	3C2 (W), 3CC (R)
3-7	FC	Feature Control Register	3?A (W), 3CA (R)
3-8	FEAT	Input Status Register 0 (Feature Read)	3C2 (R)
3-9	STAT	Input Status Register 1 (Display Status)	3?A (R)
3-10	DACMASK	Color Palette Pixel Mask	3C6 (RW)
3-11	DACSTATE	Color Palette State	3C7 (R)
3-12	DACRX	Color Palette Read Mode Index Register	3C7 (W)
3-13	DACWX	Color Palette Write Mode Index Register	3C8 (RW)
3-14	DACDATA	Color Palette Data	3C9 (RW)
3-15	VSE	Video Subsystem Enable	3C3 (RW)
3-16	ALTVSE	Alternate Video Subststem Enable	102 (W)
3-17	ROMMAP	ROM Mapping and Video Subsystem Control	46E8 (W) 56E8 (W) 66E8 (W) 76E8 (W)
3-18	CACHE	I/O Data Cache	4BC4-4BC5 (RW)
-	EXPANSION	Expansion Connector (long-slot boards only)	2E9 (RW)
		Expansion Connector (long-slot boards only)	A4E8 (W)
		Expansion Connector (long-slot boards only)	A5E8 (W)
		Expansion Connector (long-slot boards only)	A6E8 (W)
		Expansion Connector (long-slot boards only)	A7E8 (W)

Note: '?' in the above port address is 'B' in monochrome mode and 'D' in color mode (see MISC[0]).

Note: the ports above indicated as 'expansion connector' ports are not defined in this document. The port address ranges indicated are decoded by boards that provide the expansion connectors (i.e., read accesses to 2E9 will drive the bus even if no expansion board is installed). This capability is provided for compatibility with the IBM 'Display Adapter' (VGA) designed for use in the PC/AT bus. Contact Video Seven for availability of future products for use with these connectors.

I/O Port 3CC (R)

	Bit #	Description	3C2 Access	3CC Access	Reset By	Reset State
msb	7	Vertical Retrace Polarity	W	R	Hard Reset	0
	6	Horizontal Retrace Polarity	W	R	Hard Reset	0
	5	Page Select	W	R	Hard Reset	0
	4	-reserved-	-	reads back 0	-	-
	3	Clock Select Bit-1	W	R	Hard Reset	0
	2	Clock Select Bit-0	W	R	Hard Reset	0
	1	Enable Ram	W	R	Hard Reset	0
lsb	0	CRTC I/O Address	W	R	Hard Reset	0

This register was write-only at I/O port 3C2 in the EGA. Reading I/O port 3C2 returns the contents of the Feature Read Register (FEAT) (also called Input Status 0). For state save/restore and VGA compatibility, the V7VGA allows this register to also be read at I/O port 3CC.

Bit Descriptions

Bit 7 Vertical Retrace Polarity: 0 = active high (positive), 1 = active low (negative)

Bit 6 Horizontal Retrace Polarity: 0 = active high (positive), 1 = active low (negative)

Bit 5 Page Bit for Odd/Even: This bit acts as the lsb of the display memory address when in the 'Odd/Even' modes (SR4 bit-2 = 1). Setting this bit to 0 selects odd memory locations; setting it to 1 selects even locations. This bit is set for modes 0, 1, 2, 3, and 7 (text modes). This bit has no effect if chain (GR6 bit-1) or chain4 (SR4 bit-3) are enabled.

Bit 4 Reserved (reads back 0)

Bit 3-2 Clock Select: These bits select the clock source as follows:

Bit-3	Bit-2	Clock Source (CLK[4]=0)	Clock Source (CLK[4]=1)
0	0	25.175 MHz (X25M pin)	50.350 MHz (XD0M pin)
0	1	28.332 MHz (X28M pin)	65.000 MHz (XD1M pin)
1	0	Feature Connector Input (FCLK pin)	Feature Connector Input (FCLK pin)
1	1	0 MHz (XRESM pin)	40.000 MHz (XD2M pin)

This field has been extended to a 3-bit field by adding an msb in the Clock Select (CLK) register (extension register ERA4). This field should not be changed except during synchronous reset (SR0 bit-1 = 0) or display memory contents may be corrupted.

Bit 1 Enable RAM: When this bit is set to 0, display RAM is disabled from access by the processor. When this bit is set to 1, display RAM will respond at addresses set by the value programmed into the Control Data Select of the Graphics Controller.

Bit 0 CRTC I/O Address: This bit selects I/O addresses for monochrome or color mode.
0 sets the CRTC to 3BxH and the Input Status Register 1 to 3BAH (monochrome mode).
1 sets the CRTC to 3DxH and the Input Status Register 1 to 3DAH (color mode).
In register descriptions, port addresses 3Bx or 3Dx as selected by this bit are indicated as '3?x'. The registers affected by this bit are the Display Status register, Feature Control register, CRTC index register, and CRTC data registers.

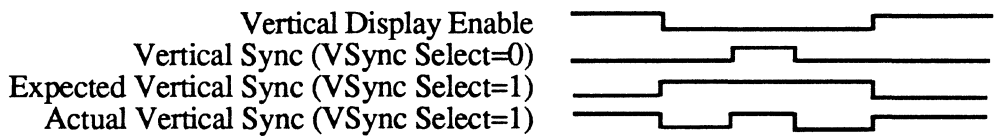
	Bit #	Description	3?A Access	3CA Access	Reset By	Reset State
msb	7	-unused-	-	-		
	6	-unused-	-	-		
	5	-unused-	-	-		
	4	-unused-	-	-		
	3	VSync Select	W	R		
	2	-unused-	-	-		
	1	Feature Control Bit 1	W	R		
lsb	0	Feature Control Bit 0	W	R		

This register is write-only at I/O port 3?A in the EGA. Reading I/O port 3?A returns the contents of the Display Status Register (STAT) (also called Input Status 1). For state save/restore and VGA compatibility, the V7VGA allows this register to also be read at I/O port 3CA.

Bit Descriptions

Bit 7-4 Unused

Bit-3 VSync Select - When this bit is set to 1, the vertical sync output becomes the logical OR of vertical sync and vertical display enable:



The 'expected' result would have matched what the old IBM CGA board produced, which might have conceivably been useful. However, as it is, this is not a really interesting capability. Some monitors have problems with this kind of waveform and some don't, depending on how their internal circuitry is implemented. This capability is implemented strictly for IBM VGA compatibility.

Bit-2 Unused

Bit 1-0 These bits convey information to the feature connector. They drive the feature control 1 and feature control 0 pins of the feature connector.

	Bit #	Description	Access	Reset By	Reset State
msb	7	Vertical Interrupt	R		
	6	Feature Code Bit-1	reads back as 1		
	5	Feature Code Bit-0	reads back as 1		
	4	Switch Sense	R		
	3	-unused-	reads back as 0		
	2	-unused-	reads back as 0		
	1	-unused-	reads back as 0		
lsb	0	-unused-	reads back as 0		

Bit Descriptions

Bit 7 CRT Interrupt: 0 indicates no interrupt is pending; 1 indicates an interrupt is pending. CRTC register 11 (CR11) bit-5 enables CRT interrupts to occur at the leading edge of vertical sync if it is set to 0. The interrupt is normally connected to IRQ2 in a PC/AT.

Note: In the IBM EGA, the CRT Interrupt status bit does not properly indicate that the CRT Controller is requesting service of a frame interrupt. IBM's EGA implementation has this signal connected directly to the IRQ2 Bus. If other boards use IRQ2, this bit might be a 1 even if a CRT interrupt is not pending. The V7VGA does **not** duplicate this behavior; this bit reflects the state of the interrupt flip-flop in the chip (which is held clear if CR11[4]=0 and set one scan line after vertical display enable ends). In the VEGA VGA, this bit is independent of whether the interrupt is enabled or disabled (CR11[5]=1 disables the interrupt). In the V7VGA, this bit is 0 if the interrupt is disabled.

Bit 6-5 Feature Code: These bits are input from the feature connector as feature code bits 0 & 1 and normally read back as 1 if nothing is connected to the feature connector..

Bit 4 Switch Sense: This bit returns the state of the SWITCH pin, typically connected to the output of the LM339 monitor ID comparator.

	Bit #	Description	Access	Reset By	Reset State	
msb	7	Not Vertical Retrace (Rev 4 only)	R			
	6	-unused-	reads back as 0			
	5	Diagnostic Use Bit-1	R			
	4	Diagnostic Use Bit-0	R			
	3	Vertical Retrace	R			
	2	Fake Light Pen Switch	reads back as 1			
	1	Fake Light Pen Flip Flop	reads back as 0	Reset	0	
lsb	0	Display Disabled	R			

When this register is read, the Attribute Controller index/data toggle is reset to index state.

Bit Descriptions

Bit 7 Not Vertical Retrace: 0 indicates that vertical retrace is in progress. This bit is not implemented prior to chip revision 4; it reads back 0 in chip revision 3 and earlier.

Bit 6 Unused (reads back 0)

Bit 5-4 Diagnostic Usage: These bits are connected to 2 of the 8 outputs of the Attribute Controller (video output data during display periods and overscan color during non-display periods). Selection of one of four pairs of bits is controlled by bits 5-4 of Color Plane Register AR12. Note that IBM's EGA had 01 and 10 reversed (IBM's EGA Technical Reference Manual matches the VGA but their hardware doesn't). In VGA mode, the V7VGA matches IBM's VGA documentation and hardware.

Color Plane Register		Display Status (IBM EGA Hardware)		Display Status (V7VGA)	
Bit 5	Bit 4	Bit 5	Bit 4	Bit-5	Bit-4
0	0	Video 2	Video 0	Video 2	Video 0
0	1	Video 3	Video 1	Video 5	Video 4
1	0	Video 5	Video 4	Video 3	Video 1
1	1	Video 7	Video 6	Video 7	Video 6

'Video n' indicates the output of a pair of 4-bit Attribute Controller multiplexers that provide output for video pin 'n' through the final 8-bit output stage. This is equivalent to the values on pins V0-V7 in non-256-color modes. However, when MUX256 (GR5 bit-6) is 1, each lower nibble out of the Attribute Controller palette ram spends one dot clock in the V3-V0 position and one dot clock in the V7-V4 position, so can show up at either of two diagnostic muxing selections. In other words, in 256-color mode, the Display Status register diagnostic bits don't read back the 8-bit pixel value that goes out on V7-V0 every two dot clocks, but the two 4-bit intermediate values that get assembled one stage later into the 8-bit pixel value.

Bit 3 Vertical Retrace: 1 indicates that vertical retrace is in progress.

Bit 2 Fake Light Pen Switch: 1 indicates that the light pen switch is open (off). Since the light pen is not implemented in the V7VGA, this bit always reads as 1.

Bit 1 Fake Light Pen Flip Flop: 0 indicates that a light pen trigger pulse has not been received. Since the light pen is not implemented in the V7VGA, this bit always reads as 0.

Bit 0 Display Disable: 0 indicates that display of video data is enabled
1 indicates that a vertical or horizontal retrace interval is in progress

	Bit #	Description	Access	Reset By	Reset State
msb	7	Palette Pixel Mask 7	RW		
	6	Palette Pixel Mask 6	RW		
	5	Palette Pixel Mask 5	RW		
	4	Palette Pixel Mask 4	RW		
	3	Palette Pixel Mask 3	RW		
	2	Palette Pixel Mask 2	RW		
	1	Palette Pixel Mask 1	RW		
lsb	0	Palette Pixel Mask 0	RW		

Bit Descriptions

Bits 7-0 The contents of this register are logically ANDed with the 8 bits of video data coming into the color palette. This provides a method of altering displayed colors without changing display memory or the contents of the color palette. By partitioning color information by one or more bits in the color palette, such effects as rapid animation, overlays, and flashing objects can be produced.

This register is physically located in the color palette chip.

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	-unused-	-		
	4	-unused-	-		
	3	-unused-	-		
	2	-unused-	-		
	1	Palette State 1	R		
lsb	0	Palette State 0	R		

Bit Descriptions

Bits 7-2 Unused

Bits 1-0 Palette State - These bits are the saved lsbs of the last I/O write to ports 3C6-3C9. The interesting combinations are:

00 - the last I/O write was to 3C8, the color palette 'write-mode' index register
11 - the last I/O write was to 3C7, the color palette 'read-mode' index register

This register is implemented in the V7VGA chipset; the other registers in the 3C6-3C9 range are implemented in the color palette chip. Read accesses to 3C7 disable I/O decode of the palette chip and substitute the contents of this register.

This register is required for implementation of state save/restore, since the color palette chip automatically increments the index register differently depending on whether the index is written at 3C7 or 3C8 (and it doesn't provide the ability to read all required internal state information).

	Bit #	Description	Access	Reset By	Reset State
msb	7	Color Palette Index 7	W		
	6	Color Palette Index 6	W		
	5	Color Palette Index 5	W		
	4	Color Palette Index 4	W		
	3	Color Palette Index 3	W		
	2	Color Palette Index 2	W		
	1	Color Palette Index 1	W		
lsb	0	Color Palette Index 0	W		

Bit Descriptions

Bits 7-0 This register contains the index value for read access to the 256 registers of color palette. Each of the data registers are 18-bits in length (6 bits each for red, green, and blue), so must be read as a sequence of 3 bytes. After writing the index to this register, data values may be read from the DACDATA register (port 3C9) in sequence: 1) red, 2) green, 3) blue.

After writing the index value to this register, it is saved in an internal register then incremented automatically. The save register is used to point at the current data register until completion of the 3-byte read sequence. After reading the third value in the 3-byte sequence (the blue value), the save register is automatically updated from the previously incremented index register and the index register is again automatically incremented. This allows the entire palette (or any subset) to be read by writing the index of the first color in the set, then sequentially reading the values for each color, without having to reload the index every three bytes.

When writing the index for purposes of reading color data values, the index is written at 3C7; when writing the index for purposes of writing color data values, the index is written at 3C8. The color palette chip contains only one index register, but automatically increments it differently depending on whether reading or writing of data is being performed. Thus there are two ports for writing the index (3C7 and 3C8), but only one for reading it back (3C8). The color palette chip actually would read back the index value from 3C7 also if allowed, but read accesses to 3C7 are intercepted and the DACSTATE register contents are substituted instead. This is because the color palette chip doesn't save the state of whether the chip is being read or written and this information is required for saving and restoring the state of the video subsystem during interrupt service. The state of the RGB sequence is not saved, so the user must access the data values in uninterruptable sequences of 3 bytes. Whenever either color palette index register is written, any 3-byte read or write sequence in progress is aborted and a new one started.

The index registers are physically located in the color palette chip.

	Bit #	Description	Access	Reset By	Reset State
msb	7	Color Palette Index 7	RW		
	6	Color Palette Index 6	RW		
	5	Color Palette Index 5	RW		
	4	Color Palette Index 4	RW		
	3	Color Palette Index 3	RW		
	2	Color Palette Index 2	RW		
	1	Color Palette Index 1	RW		
lsb	0	Color Palette Index 0	RW		

Bit Descriptions

Bits 7-0 This register contains the index value for write access to the 256 registers of color palette. Each of the data registers are 18-bits in length (6 bits each for red, green, and blue), so must be written as a sequence of 3 bytes. After writing the index to this register, data values may be written to the DACDATA register (port 3C9) in sequence: 1) red, 2) green, 3) blue.

After reading the third value in the 3-byte sequence (the blue value), the index register is automatically incremented. This allows the entire palette (or any subset) to be written by writing the index of the first color in the set, then sequentially writing the values for each color, without having to reload the index every three bytes.

When writing the index for purposes of reading color data values, the index is written at 3C7; when writing the index for purposes of writing color data values, the index is written at 3C8. The color palette chip contains only one index register, but automatically increments it differently depending on whether reading or writing of data is being performed. Thus there are two ports for writing the index (3C7 and 3C8), but only one for reading it back (3C8). The color palette chip actually would read back the index value from 3C7 also if allowed, but read accesses to 3C7 are intercepted and the DACSTATE register contents are substituted instead. This is because the color palette chip doesn't save the state of whether the chip is being read or written and this information is required for saving and restoring the state of the video subsystem during interrupt service. The state of the RGB sequence is not saved, so the user must access the data values in uninterruptable sequences of 3 bytes. Whenever either color palette index register is written, any 3-byte read or write sequence in progress is aborted and a new one started.

The index registers are physically located in the color palette chip.

	Bit #	First Access	Second Access	Third Access	Access	Reset By	Reset State
msb	7	-Reserved-	-Reserved-	-Reserved-	RW		
	6	-Reserved-	-Reserved-	-Reserved-	RW		
	5	Color Palette Red 5	Color Palette Green 5	Color Palette Blue 5	RW		
	4	Color Palette Red 4	Color Palette Green 4	Color Palette Blue 4	RW		
	3	Color Palette Red 3	Color Palette Green 3	Color Palette Blue 3	RW		
	2	Color Palette Red 2	Color Palette Green 2	Color Palette Blue 2	RW		
	1	Color Palette Red 1	Color Palette Green 1	Color Palette Blue 1	RW		
lsb	0	Color Palette Red 0	Color Palette Green 0	Color Palette Blue 0	RW		

Bit Descriptions

Bits 7-0 The color palette contains 256 registers, each 18 bits in length (6 bits each for red, green, and blue). Each is accessed as a sequence of 3 bytes. After writing the index to the DACRX or DACWX register (port 3C7 or 3C8), data values may be read from or written to port 3C9 in sequence: 1) red, 2) green, 3) blue. The index register is automatically incremented for each 3-byte set to allow multiple color registers to be read or written, without having to reload the index every three bytes.

When writing the index for purposes of reading color data values, the index is written at 3C7; when writing the index for purposes of writing color data values, the index is written at 3C8. The color palette chip contains only one index register, but automatically increments it differently depending on whether reading or writing of data is being performed. Thus there are two ports for writing the index (3C7 and 3C8), but only one for reading it back (3C8). The color palette chip actually would read back the index value from 3C7 also if allowed, but read accesses to 3C7 are intercepted and the DACSTATE register contents are substituted instead. This is because the color palette chip doesn't save the state of whether the chip is being read or written and this information is required for saving and restoring the state of the video subsystem during interrupt service. The state of the RGB sequence is not saved, so the user must access the data values in uninterruptable sequences of 3 bytes. Whenever either color palette index register is written, any 3-byte read or write sequence in progress is aborted and a new one started.

The color palette registers are physically located in the color palette chip.

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	reads back as 1		
	6	-unused-	reads back as 1		
	5	-unused-	reads back as 1		
	4	-unused-	reads back as 1		
	3	-unused-	reads back as 1		
	2	-unused-	reads back as 1		
	1	-unused-	reads back as 1		
lsb	0	Video Subsystem Enable	RW	Reset	1

This register may be used to enable or disable V7VGA memory and I/O addressing.

Bit Descriptions

Bits 7-1 Unused (read back as 1)

Bit 0 Video Subsystem Enable - When this bit is set to 0 and ARMVSE is 1 (bit-7 of extension register FCh), all I/O and memory accesses to the video subsystem are disabled (except for ports 102 and 3C3, which remain enabled to allow the Video Subsystem to be re-enabled). When this bit is set to 1 (the default state) and ARMVSE is 1, I/O and memory accesses to the video subsystem work as documented in this manual. If ARMVSE is 0, accesses of this port are ignored.

The ARMVSE bit described above is reset to 0 by a hardware reset (i.e., the default for this mechanism on powerup is **disarmed** and **awake**).

The ARM mechanism is set up so that bit-0 of the VSE register is prevented from changing unless ARMVSE is set to 1 (VSE armed). This is to prevent the VSE mechanism from being disarmed while disabled (in which case it would not be able to be rearmed or re-enabled without hardware reset).

Note that port 102 has the same capability to disable the V7VGA as port 3C3, except that port 3C3 is normally not implemented (ARMVSE should be set to 0 on the V7VGA chip) in Display Adapter (PC/AT Bus VGA Card) configurations. ARMVSE has no effect on port 102. Either 102 (if in setup mode) or 3C3 (if armed) may be used to disabled the V7VGA; both ports must be set to the enable state for the V7VGA to be enabled.

The DISABLE pin may also be used to disable the V7VGA; if DISABLE is 1, the V7VGA will be disabled, independent of the state of port 102 or 3C3. DISABLE is normally driven by the inversion of 46E8 bit-3 in PC/AT bus configurations.

The V7VGA continues to display video data while disabled (memory and registers are 'write-protected'). When the V7VGA is dsabled, accesses to the DAC registers are also disabled.

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	reads back as 0		
	6	-unused-	reads back as 0		
	5	-unused-	reads back as 0		
	4	-unused-	reads back as 0		
	3	-unused-	reads back as 0		
	2	-unused-	reads back as 0		
	1	-unused-	reads back as 0		
lsb	0	Video Subsystem Enable	RW	Reset	1

This register may be used to enable V7VGA memory and I/O addressing in setup mode.

Bit Descriptions

Bits 7-1 Unused (read back as 0)

Bit 0 VGA Enable - When in setup mode (SETUP* pin = 0), setting this bit to 1 enables V7VGA memory and I/O addressing; setting this bit to 0 disables V7VGA memory and I/O addressing (except for ports 102 and 3C3 to allow the V7VGA to be re-enabled). This register is not accessable (accesses to port 102 are ignored) when not in setup mode (SETUP* pin = 1).

In PC/AT systems where the VGA is implemented as an optional board (for installation in a slot on the bus), the SETUP* pin is normally driven by bit-4 of I/O port 46E8h which is implemented as an on-board register external to the V7VGA chip. Port 46E8 is not implemented in PS/2 systems and is not documented in IBM VGA documentation.

In PS/2 systems where the VGA is implemented as a standard feature (as part of the motherboard logic), the SETUP* pin is driven by bit-5 of I/O port 94h which is implemented as part of the motherboard logic. Port 94 is not implemented in PC/AT systems.

Note that port 3C3 has the same capability to disable the V7VGA as port 102, except that port 3C3 is normally not implemented (ARMVSE should be set to 0 on the V7VGA chip) in PC/AT bus configurations. ARMVSE has no effect on port 102. Either 102 (if in setup mode) or 3C3 (if armed) may be used to disabled the V7VGA; both ports must be set to the enable state for the V7VGA to be enabled.

The DISABLE pin may also be used to disable the V7VGA; if DISABLE is 1, the V7VGA will be disabled, independent of the state of port 102 or 3C3. DISABLE is normally driven by the inversion of 46E8 bit-3 in PC/AT bus configurations.

The V7VGA continues to display video data while disabled (memory and registers are 'write-protected'). When the V7VGA is dsabled, accesses to the DAC registers are also disabled.

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	-unused-	-		
	4	Setup Mode (0 = normal, 1 = setup)	W	Reset	0
	3	Video Subsystem Enable (0 = disable, 1 = enable)	W	Reset	0
	2	ROM Bank Control Bit-2	W	Reset	0
	1	ROM Bank Control Bit-1	W	Reset	0
lsb	0	ROM Bank Control Bit-0	W	Reset	0

This register may be used to control memory mapping of the BIOS ROM and V7VGA memory and I/O addressing. It is implemented external to the V7VGA chip in PC/AT bus configurations and is not implemented in PS/2 systems (and is therefore not documented in the PS/2 VGA documentation). In PS/2 systems, BIOS ROM mapping capability is not implemented (there is no BIOS ROM as the video BIOS is implemented as part of the motherboard BIOS). In PS/2 systems, setup mode is controlled by I/O port 94h bit-5 and video subsystem enable/disable capability is provided by ports 3C3 and 102 only. I/O port 46E8 is sparse decoded such that 56E8, 66E8, and 76E8 access the same register.

Bit Descriptions

Bit 4 Setup Mode - The inversion of this bit is used to drive the V7VGA SETUP* pin. When this bit is 1, the V7VGA is in setup mode. When in setup mode (SETUP* pin = 0), setting port 102 bit-0 to 0 disables V7VGA memory and I/O addressing (except for ports 102 and 3C3 to allow the V7VGA to be re-enabled). When this bit is 0, accesses to port 102 are ignored.

Bit 3 Video Subsystem Enable - The inversion of this bit is used to drive the V7VGA DISABLE pin. When this bit is 0, the V7VGA will be disabled. When the V7VGA is disabled with this bit, all video memory and I/O port accesses will be ignored, including ports 102 and 3C3 and the DAC registers, but not including port 46E8. The V7VGA continues to display video data while disabled (memory and registers are 'write-protected').

Bits 2-0 ROM Map Control - These bits control which 4K bank of the 32KB BIOS ROM to map to C7000h to C7FFFh:

Bits 2-0 (Bank)	ROM Segment mapped to C7000h to C7FFFh
0	1st 4K (C0000h to C0FFFh if the ROM were mapped linearly)
1	2nd 4K (C1000h to C1FFFh if the ROM were mapped linearly)
2	3rd 4K (C2000h to C2FFFh if the ROM were mapped linearly)
3	4th 4K (C3000h to C3FFFh if the ROM were mapped linearly)
4	5th 4K (C4000h to C4FFFh if the ROM were mapped linearly)
5	6th 4K (C5000h to C5FFFh if the ROM were mapped linearly)
6	7th 4K (C6000h to C6FFFh if the ROM were mapped linearly)
7	8th 4K (C7000h to C7FFFh if the ROM were mapped linearly)

The first 24KB of the BIOS ROM are mapped linearly from C0000 to C5FFF. Accesses to C6000-C67FF are ignored (this 2K section of the ROM is instead accessable at CA000h to CA7FFh). Accesses to C6800-C6FFF select the last 2KB of the BIOS ROM (the upper half of bank 7 if the ROM were mapped linearly).

This register is set to 0Eh during initialization by the video BIOS (normal mode, video subsystem enabled, and ROM bank 6). The value in 46E8 is not changed after initialization except for bit-3 which is changed by the BIOS call to enable/disable the video subsystem.

	Bit #	Description (port 4BC4)	Description (port 4BC4)	Access	Reset By	Reset State
msb	7	Cache Enable	Cache Data Bit-7	-		
	6	-unused-	Cache Data Bit-6	-		
	5	-unused-	Cache Data Bit-5	-		
	4	-unused-	Cache Data Bit-4	-		
	3	Cache Index Bit-3	Cache Data Bit-3	RW		
	2	Cache Index Bit-2	Cache Data Bit-2	RW		
	1	Cache Index Bit-1	Cache Data Bit-1	RW		
lsb	0	Cache Index Bit-0	Cache Data Bit-0	RW		

These registers may be used to save I/O port access information for use by CGA/MDA/Hercules emulation utility software. They are implemented external to the V7VGA chip using the Video Seven IOU chip.

I/O write caching capability is required due to the fact that 80286 and 80386 processors do not immediately respond to NMI requests, but continue to execute instructions from their prefetch queue until it is empty. This will result in as many as 3-4 I/O write operations executed before the NMI is serviced, depending on the exact code sequence.

Note that since the V7VGA chip TRAP* pin is multiplexed with the Microchannel Bus ADL signal (address latch), that this capability cannot be used on a PS/2 bus card.

Bit Descriptions

4BC4 Bit 7 Cache Enable - Setting this bit to 1 enables the cache to initiate a cache trap sequence (save I/O write address and data information in the cache) whenever the TRAP* signal is generated.

4BC4 Bits 3-0 Cache Index - These bits provide an index which points to one of sixteen cache data registers at I/O port 4BC5h. This field may be programmed directly to read or write any of the cache data registers. It is normally programmed to 0 when enabling the cache so that cache data is saved starting at the first register of the cache. This field is automatically incremented in hardware when performing a 'cache trap sequence' (saving address and data information in the cache). A cache trap sequence is initiated for selected I/O write accesses, controlled by V7VGA extension register 0EBh (i.e., the detection of which I/O write accesses to cache is controlled by the V7VGA chip). I/O read accesses are ignored. The cache trap sequence first saves the 8 lsbs of the I/O address in the cache data register pointed to by the index field, increments the index to save the I/O data in the next cache data register, then increments the index again to set up for the next sequence. The index will not automatically increment past 13, leaving cache data registers 14 and 15 for use as scratch registers. This allows a minimum of six unambiguous I/O write operations to be stored in the cache. A seventh would be stored in registers 12 and 13, but if an eighth occurred, the seventh would be overwritten. No more than 3-4 should ever occur in normal operation.

4BC5 Bits 7-0 Cache Data - These 16 registers provide storage for cache data. The register to be accessed is pointed to by the cache index field at I/O port 4BC4h. All 16 registers may be read/write accessed by the processor. I/O address/data write information is automatically written into pairs of registers in the range of 0-13 during 'cache trap sequences' when the cache is enabled by 4BC4 bit-7.

Sequencer Registers

6/20/88

The Sequencer generates memory timing for the display RAMs and the character clock for controlling display memory refresh reads.

Page	Abbreviation	Register Name	Port Address	Index	Access
3-20	SRX	Sequencer / Extensions Index Register	3C4	--	R/W
3-21	SR0	Reset	3C5	00	R/W
3-22	SR1	Clocking Mode	3C5	01	R/W
3-24	SR2	Plane Mask	3C5	02	R/W
3-25	SR3	Character Map Select	3C5	03	R/W
3-26	SR4	Memory Mode	3C5	04	R/W
-	SR5	-ignored-	3C5	05	-
3-27	SR6	Extensions Control	3C5	06	R/W
3-28	SR7	Reset Horizontal Character Counter	3C5	07	W

Sequencer Operation

The sequencer generates all display memory timing including RAS and CAS to the display memory chips. It also refreshes display memory. During each horizontal scan, display memory accesses alternate between CRT accesses and CPU accesses in a ratio controlled by the current timing state (see Extended Timing Select Register at extensions index FD). When display enable ends at the end of each horizontal scan line (after the proper number of displayed characters have been read), CRT accesses are not required until the start of the next scan line, so the cycles are free for other use. The first five are used by the Sequencer to generate refresh accesses to display memory; the next two are used if required to read graphics pointer pattern information; the remaining cycles are available for access by the CPU. The total number of cycles (character clocks) available during each horizontal blanking interval is 'CR0-CR1+4'. If the number of cycles drops below 7, the graphics pointer may not be used; if it drops below 5, display memory may not be adequately refreshed.

	Bit #	Description	Access	Reset By	Reset State
msb	7	Sequencer / Extensions Index Bit-7 (msb)	R/W		
	6	Extensions Index Bit-6	R/W		
	5	Extensions Index Bit-5	R/W		
	4	Extensions Index Bit-4	R/W		
	3	Extensions Index Bit-3	R/W		
	2	Sequencer/Extensions Index Bit-2	R/W		
	1	Sequencer/Extensions Index Bit-1	R/W		
lsb	0	Sequencer/Extensions Index Bit-0 (lsb)	R/W		

The Sequencer / Extensions Index Register points to the Sequencer registers and to the V7VGA Extension registers. The three least significant bits determine the Sequencer register which will be pointed to in the next register read/write operation. The seven least significant bits determine the Extension register which will be pointed to in the next register read/write operation.

If the msb of the Index register is set to 0, or access to the extension registers is disabled, the Sequencer registers will be accessed per the three lsbs of the index (the upper bits of the index will be ignored, mapping the first 8 sequencer registers repeatedly thoughout the range of indices from 0 to 7F). If the index register msb is set to 1 and write access to the extension registers is enabled, the V7VGA extension registers will be accessed per the 7 lsbs of the index.

In other words, if extensions access is disabled, sequencer registers SR0-7 may be accessed anywhere in the range of indices from 00 to FF (0 same as 8, 10, 18, etc.). If extensions are enabled, sequencer registers SR0-7 are accessed in 8-register blocks from 00 to 7F. Extension registers are accessed at 80-FF. If a value is written to the index register that points to SR0-7 (0-FF with extensions disabled or 0-7F with extensions enabled), then readback of the index register will return a value from 0-7 (as if index bits 3-7 are not implemented). If a value is written to the index register that points to the extension registers (80-FF with extensions enabled), readback of the index register will return the value written.

Index 00

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	-unused-	-		
	4	-unused-	-		
	3	-unused-	-		
	2	-unused-	-		
	1	Synchronous Reset 1*	R/W		
lsb	0	Synchronous Reset 2*	R/W		

Bit Descriptions

Bit 1 Synchronous Reset 1*: Setting this bit to 0 causes the Sequencer to clear synchronously and halt (disabling display memory refresh, display memory access, and H/V sync signals to the display). Setting this bit to 1 causes the Sequencer to run unless bit-0 (synchronous reset 2) is cleared to 0. Both Reset register bits must be 1 to allow the Sequencer to operate. In order to preserve display memory contents, this bit should be left set to 0 only for short periods of time (a few tens of microseconds at most). The following registers should not be changed unless this bit is 0:

Clocking Mode Register (SR1) bits 0 and 3
Misc Output Register bits 2-3
Extensions CLK Register bit 4
Extensions 'Extended Timing Select' Register bits 0-7

Bit 0 Synchronous Reset 2*: This bit performs the same function as bit-1 with the addition that when it transitions from 1 to 0, it also resets SR3 (character map select) to 0.

I/O Port 3C5 (RW)	Sequencer Clocking Mode Register	SR1

Index 01

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	Full Bandwidth / Video Off	R/W		
	4	Shift 32	R/W		
	3	Dot Clock (1 = divide master clock by 2)	R/W		
	2	Shift Load	R/W		
	1	-unused-	-		
lsb	0	8/9 Dot Clocks (0 = 9 dots, 1 = 8 dots)	R/W		

Bit Descriptions

Bits 7-6 Unused

Bit 5 Full Bandwidth / Video Off - Setting this bit to 1 blanks the screen (video outputs are at black level, not overscan color). This also gives the maximum number of video memory cycles to the CPU.

The blanking mechanism does not stop H and V sync, blanking, or display enable signals. For example, the DE bit in the Display Status register still toggles during screen blanking.

Bit 4 Shift 32 - Setting this bit to 1 causes the display data serializers in the Graphics Controller to be reloaded every 4 character clocks. If this bit is not set, serializer loading occurs every 1 or 2 character clocks as determined by the state of bit-2 of this register (if shift-32 is set, bit-2 is ignored). When shift-32 mode is enabled, the sequencer is forced into a '2:7' interleave.

This bit is typically only set for high resolution monochrome graphics modes (32 pixels per CRTC memory access). It is not used by standard IBM VGA BIOS code nor is it set by any currently defined IBM VGA graphics mode.

Bit 3 Dot Clock - Setting this bit to 0 selects the Sequencer master clock input to be output on the Dot Clock output pin. Setting this bit to 1 causes the master clock to be divided by 2 to generate the dot clock. As the dot clock is the primary clock used by the system, all other timings will be stretched as they are derived from the dot clock. Dot clock divided by 2 is used for 320 x 200 modes (except 256-color mode).

Bit 2 Shift Load - Setting this bit to 0 causes the display data serializers in the Graphics Controller to be reloaded every character clock. Setting this bit to 1 causes the display serializers in the Graphics Controller to be reloaded every other character clock. This mode is useful when 16 pixels are fetched every CRTC memory access. This bit is typically only set for monochrome graphics modes.

(continued on following page)

Bit 1 Unused (reads back as 0)

In the EGA, this bit selected the ratio of display memory accesses allowed by the CPU relative to accesses by VGA hardware to refresh the CRT display. The only allowable selections in the IBM EGA and VGA are via this bit: a setting of 0 indicates that a memory access by the CPU may occur only once for every 4 CRT accesses (referred to as 1:4 interleave) and a setting of 1 indicates that CPU memory accesses may occur 3 times for every 2 CRT accesses (3:2 interleave).

However, higher resolution modes must fetch more data from memory in a given period of time to refresh the CRT display, so allow the CPU to access display memory less often. To accomodate existing EGA and VGA modes plus new extended modes, additional timing states must therefore be provided to allow selection of the best possible performance for the current mode of operation. In the V7VGA, the bandwidth control mechanisms have been extended via the Extended Timing Select Register at extensions index FD. The new register allows a wider selection of timing states than are available in the standard EGA and VGA to accomodate new high resolution modes.

Refer to the Extended Timing Select register (extensions index FD) for additional detail.

Bit 0 8/9 Dot Clocks: Setting this bit to 0 causes the Sequencer to generate character clocks which are 9 dots wide. Setting this bit to 1 causes the sequencer to generate character clocks which are 8 dots wide. The IBM modes that use 9-dot wide character clocks are monochrome text mode (720 x 350 resolution), and the new VGA 400-line text modes (9x16 font, 40x25 and 80x25 text modes). All other standard modes use 8 dot wide character clocks.

Index 02

	Bit #	Description	Access	Reset By	Reset State
msb	7	Reserved for future use	-		
	6	Reserved for future use	-		
	5	Reserved for future use	-		
	4	Reserved for future use	-		
	3	Enable Plane 3	R/W		
	2	Enable Plane 2	R/W		
	1	Enable Plane 1	R/W		
lsb	0	Enable Plane 0	R/W		

A "1" in any of the bits 0 through 3 enables the CPU to write to the corresponding memory planes 0-3. When this register is loaded with 0Fh, the CPU can do a 32-bit write operation in one memory cycle.

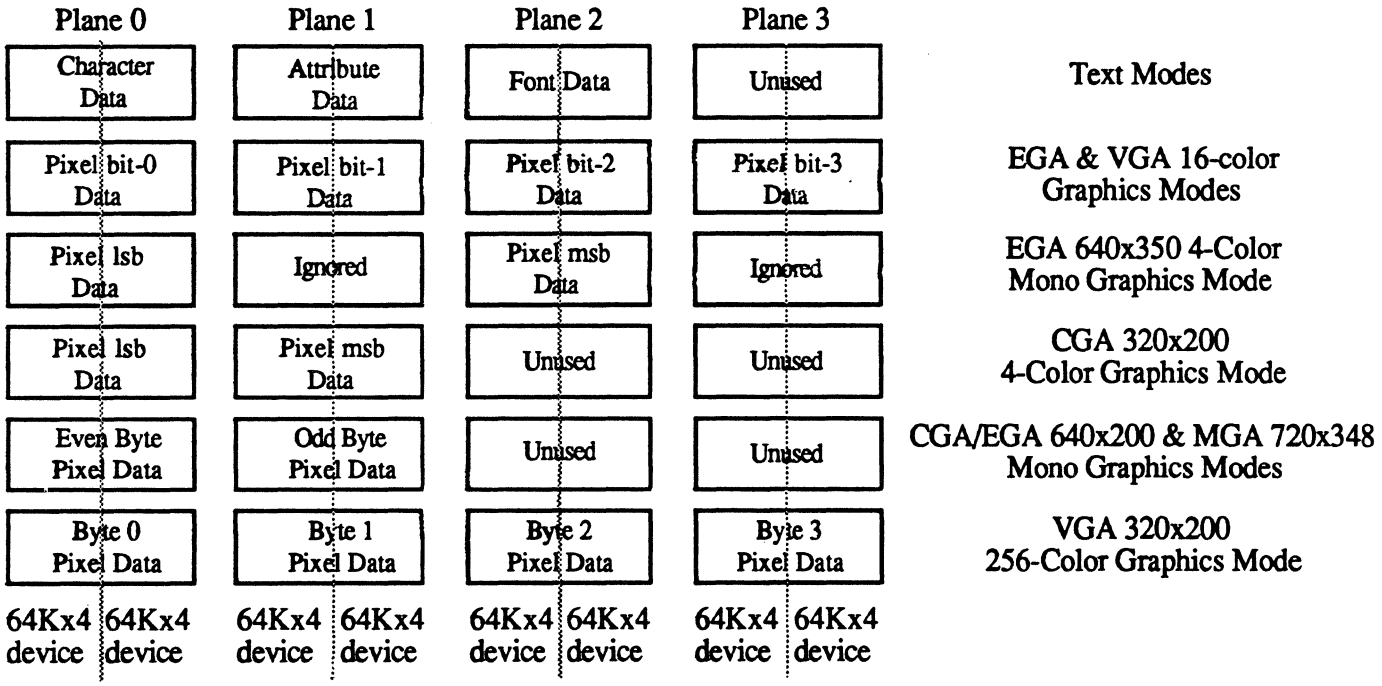


Figure 3-4. Display Memory Plane Mapping

In EGA and VGA 4-bit per pixel graphics modes, this register should be set to 0Fh (planes 0-3 each contain 1 bit of the pixel value). In text modes, this register should be set to 3 (the CPU needs to access planes 0 and 1; the font information is retrieved directly by hardware independent of the contents of this register).

When odd/even modes are selected (by clearing bits 2 and 3 of Memory Mode register SR4) planes 0/1 and planes 2/3 should have the same plane mask value. In odd/even and chain4 modes, this register is still in effect, and is ANDed with the plane select generated by the odd/even circuitry. For example, in odd/even mode the odd/even circuitry causes planes 0 and 2 to be enabled on CPU writes to even addresses and planes 1 and 3 to be enabled on CPU writes to odd addresses. However, if the plane mask setting is 3, only plane 0 (even addresses) and plane 1 (odd addresses) can actually be written to.

Index 03

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	Secondary Character Map Select bit-0 (lsb)	R/W	SR0[0]=0	0
	4	Primary Character Map Select bit-0 (lsb)	R/W	SR0[0]=0	0
	3	Secondary Character Map Select bit-2 (msb)	R/W	SR0[0]=0	0
	2	Secondary Character Map Select bit-1	R/W	SR0[0]=0	0
	1	Primary Character Map Select bit-2 (msb)	R/W	SR0[0]=0	0
lsb	0	Primary Character Map Select bit-1	R/W	SR0[0]=0	0

This register (along with the character attribute value) determines where font information is located for EGA and VGA text modes.

Bit Descriptions

Bits 3,2,5 Secondary Character Map Select - These bits select the bank used to generate text characters when attribute bit-3 is "1" according to the following table:

SR3[3]	SR3[2]	SR3[5]	Font #	Table Location
0	0	0	0	1st 8K of Plane 2
0	0	1	1	2nd 8K of Plane 2
0	1	0	2	3rd 8K of Plane 2
0	1	1	3	4th 8K of Plane 2
1	0	0	4	5th 8K of Plane 2
1	0	1	5	6th 8K of Plane 2
1	1	0	6	7th 8K of Plane 2
1	1	1	7	8th 8K of Plane 2

Bit 1,0,4 Primary Character Font Select - These bits select the plane used to generate text characters when attribute bit-3 is "0" according to the following table:

SR3[1]	SR3[0]	SR3[4]	Font #	Table Location
0	0	0	0	1st 8K of Plane 2
0	0	1	1	2nd 8K of Plane 2
0	1	0	2	3rd 8K of Plane 2
0	1	1	3	4th 8K of Plane 2
1	0	0	4	5th 8K of Plane 2
1	0	1	5	6th 8K of Plane 2
1	1	0	6	7th 8K of Plane 2
1	1	1	7	8th 8K of Plane 2

In text modes, bit-3 of the attribute byte normally turns the foreground intensity on or off. This bit may be redefined to be a switch between character sets. This function is enabled when there is a difference between the values of Primary and Secondary Character Map Select bits. Whenever the two values are the same, the character select function is disabled.

The format of Plane 2 font address bits 15-0 is: F2 F1 F0 C7 C6 C5 C4 C3 C2 C1 C0 R4 R3 R2 R1 R0 where F2-0 is the font # (bits 3/2/5 or bits 1/0/4), C7-0 is the character code, and R4-0 is the character row address. In the EGA, F0 is not implemented and is effectively always 0 (bits 4 and 5 of this register are ignored), limiting selection to only 4 of the 8 potential font storage areas of plane 2 (the even numbered fonts in the tables above).

Font changes take effect at the start of the next character line.

Index 04

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	-unused-	-		
	4	-unused-	-		
	3	Chain4	R/W		
	2	Odd/Even* (0=Text/CGA, 1=MGA/EGA graphics)	R/W		
	1	Extended Memory (0=64KB ram, 1=256KB ram)	R/W		
lsb	0	-unused-	-		

Bit Descriptions

Bit 7-4 unused

Bit 3 Chain4 (double odd/even): This bit is used for implementation of 256-color modes to control the generation of display memory addresses. It only effects display memory accesses from the CPU; it has no effect on CRTC accesses.

When this bit is 1, A0 provides plane select bit-0 and A1 provides plane select bit-1. This is like odd/even mode, except that A1 is used as well as A0. This bit takes priority over bit-2 (odd/even*) and GR5[4] (odd/even); when this bit is set to 1, those bits have no effect. There is no separate bit in the Graphics Controller to select chain4 (double odd/even) mode as is the case with odd/even mode; this bit is used for both.

The Graphics Controller Read Map register is ignored when this bit is 1. This bit controls plane selection for both reads and writes. The Plane Mask register (SR2) bit for planes selected by the chain4 mechanism must also be 1 for writes to go through. In other words, the plane select generated by the chain4 circuitry is logically ANDed with the Plane Mask register selects to generate the actual plane select.

Bit 2 Odd/Even*: Setting this bit to 0 will put the sequencer into the odd/even mode. '0' directs even CPU addresses to access planes 0 and 2 while odd CPU addresses access planes 1 and 3. '1' causes CPU addresses to sequentially access data within a bit plane. The planes are accessed according to the value in the Plane Mask Register (SR2).

This bit should be set to 0 for text modes. This bit should also be set to 0 when emulating CGA graphics mode with EGA/VGA hardware. The function of this bit should track the function of bit-4 of the Graphics Controller Mode Register (GR5 Odd/Even bit). Note: the binary values will be opposite.

Bit 1 Extended Memory: 256KB display memory is standard on the V7VGA board, so this bit is usually set to 1. This bit may, however, be set to 0 to allow emulation of IBM EGA modes which assume a display memory size of 64KB.

Bit 0 Unused/Ignored. In the original EGA, this bit was programmed to indicate text mode versus graphics mode (1 indicated text mode). It enabled the Character Map Select Register (SR3). There are actually 2 other bits (GR6 bit-0 and AR10 bit-0) that indicate the same information but are the opposite polarity so this bit was eliminated by the VGA.

Index 06

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	-unused-	-		
	4	-unused-	-		
	3	-unused-	-		
	2	-unused-	-		
	1	-unused-	-		
lsb	0	Extensions Access Enable	R	Reset	0

Access to the extended registers of the V7VGA chip set (registers pointed to by Sequencer indices 80-FF) is enabled and disabled by issuing write operations to port address 3C5 with an index of 6 stored in the Sequencer / Extensions Index Register. Access is enabled by writing hex 0EAH; access is disabled by writing 0AEH. Reading from 3C5 with an index of 6 stored in the index register returns the state of the access enable flag in the lsb (0 = disabled, 1 = enabled).

Access to the extension registers is disabled on reset. The capability to disable access to the extension registers is provided to allow the on-board BIOS to initialize the chip set to a particular mode of operation (especially one of the backwards-compatibility modes), with assurance that extension register contents won't be clobbered inadvertently by older non-V7VGA-aware user programs.

Index 07

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	-unused-	-		
	4	-unused-	-		
	3	-unused-	-		
	2	-unused-	-		
	1	-unused-	-		
lsb	0	-unused-	-		

Writing to SR7 with any data will cause the horizontal character counter to be held reset (character counter output = 0) until a write to SR0-6 with any data value. The write to SR0-6 clears the latch that is holding the reset condition on the character counter.

The vertical line counter is clocked by a signal derived from horizontal display enable (which does not occur if the horizontal counter is held reset). Therefore, if the write to SR7 occurs during vertical retrace, the horizontal and vertical counters will both be zeroed. A write to SR0-6 may then be used to start both counters with reasonable syncronization to an external event via software control.

This register is implemented starting with V7VGA chip revision 4.

CRT Controller Registers

6/21/88

The CRT Controller provides synchronization signals for the display monitor and addressing for non-CPU accesses of display memory. The registers are shown in the table below.

Page	Abbreviation	Register Name	I/O Port
3-30	CRX	CRTC Index Register	3?4 (RW)
3-31	CR0	Horizontal Total	3?5 Index 00 (RW)
3-32	CR1	Horizontal Display End	3?5 Index 01 (RW)
3-33	CR2	Horizontal Blanking Start	3?5 Index 02 (RW)
3-34	CR3	Horizontal Blanking End	3?5 Index 03 (RW)
3-35	CR4	Horizontal Retrace Start	3?5 Index 04 (RW)
3-36	CR5	Horizontal Retrace End	3?5 Index 05 (RW)
3-37	CR6	Vertical Total	3?5 Index 06 (RW)
3-38	CR7	Overflow	3?5 Index 07 (RW)
3-39	CR8	Screen A Preset Row Scan	3?5 Index 08 (RW)
3-40	CR9	Character Cell Height	3?5 Index 09 (RW)
3-41	CRA	Cursor Start	3?5 Index 0A (RW)
3-42	CRB	Cursor End	3?5 Index 0B (RW)
3-43	CRC	Screen A Start Address High	3?5 Index 0C (RW)
3-44	CRD	Screen A Start Address Low	3?5 Index 0D (RW)
3-45	CRE	Cursor Location High	3?5 Index 0E (RW)
3-46	CRF	Cursor Location Low	3?5 Index 0F (RW)
3-47	LPENH	Light Pen High	3?5 Index 10 (R)
3-48	LPENL	Light Pen Low	3?5 Index 11 (R)
3-49	CR10	Vertical Retrace Start	3?5 Index 10 (W,RW)
3-50	CR11	Vertical Retrace End	3?5 Index 11 (W,RW)
3-51	CR12	Vertical Display End	3?5 Index 12 (RW)
3-52	CR13	Offset	3?5 Index 13 (RW)
3-53	CR14	Underline Row Scan	3?5 Index 14 (RW)
3-54	CR15	Vertical Blanking Start	3?5 Index 15 (RW)
3-55	CR16	Vertical Blanking End	3?5 Index 16 (RW)
3-56	CR17	CRT Mode Control	3?5 Index 17 (RW)
3-58	CR18	Line Compare	3?5 Index 18 (RW)
3-59	CR1F	V7VGA Identification	3?5 Index 1F (R)
3-60	CR22	Graphics Ctrlr Data Latches	3?5 Index 22 (R)
3-61	CR24	Attr Ctrlr Index/Data Latch	3?5 Index 24 (R)
3-62	CR3x	Clear Vert Display Enable FF	3?5 Index 3x (W)

Note: '?' in the above port address is 'B' in monochrome mode and 'D' in color mode

Note: All CRTC registers except CRC-F (RW) and LPENH/L (R/O) are write-only in the original EGA. In addition, CRTC registers CR10-11 and LPENH/L are at conflicting locations in the EGA. The VGA provides software control (CR3 bit-7) of whether CR10-11 or LPENH/L are readable at indices 10-11.

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	CRTC Index Bit-5	R/W		
	4	CRTC Index Bit-4	R/W		
	3	CRTC Index Bit-3	R/W		
	2	CRTC Index Bit-2	R/W		
	1	CRTC Index Bit-1	R/W		
lsb	0	CRTC Index Bit-0	R/W		

The CRTC Index register points to the internal registers of the CRT Controller. The six least significant bits determine which register will be pointed to in the next register read/write operation to I/O port 3B5/3D5.

Since only 6 bits of the index register are used, CRTC registers 0-3F may also be addressed using index ranges 40-7F, 80-BF, and C0-FF. This, however, is not recommended, as higher index ranges are reserved for future use and this may not be true in future chip revisions.

Index 00 | **Write Protected by EREB[5] and CR11[7]**

	Bit #	Description	Access	Reset By	Reset State
msb	7	Horizontal Total Bit-7	R/W		
	6	Horizontal Total Bit-6	R/W		
	5	Horizontal Total Bit-5	R/W		
	4	Horizontal Total Bit-4	R/W		
	3	Horizontal Total Bit-3	R/W		
	2	Horizontal Total Bit-2	R/W		
	1	Horizontal Total Bit-1	R/W		
lsb	0	Horizontal Total Bit-0	R/W		

The Horizontal Total register defines the total number of characters in a horizontal scan line, including the retrace time. Together with the value in the Retrace Timing registers CR4 and CR5, the period of the retrace output signal is determined by the value in this register. The character clock input to the device is counted by a character counter. The value of the character counter is compared with the value in this register to provide the horizontal timing. All horizontal and vertical timing is based upon the contents of this register.

The value in the register = Total Number of Characters - 5

In the EGA, this register was programmed with the total number of characters - 2 instead of 5.

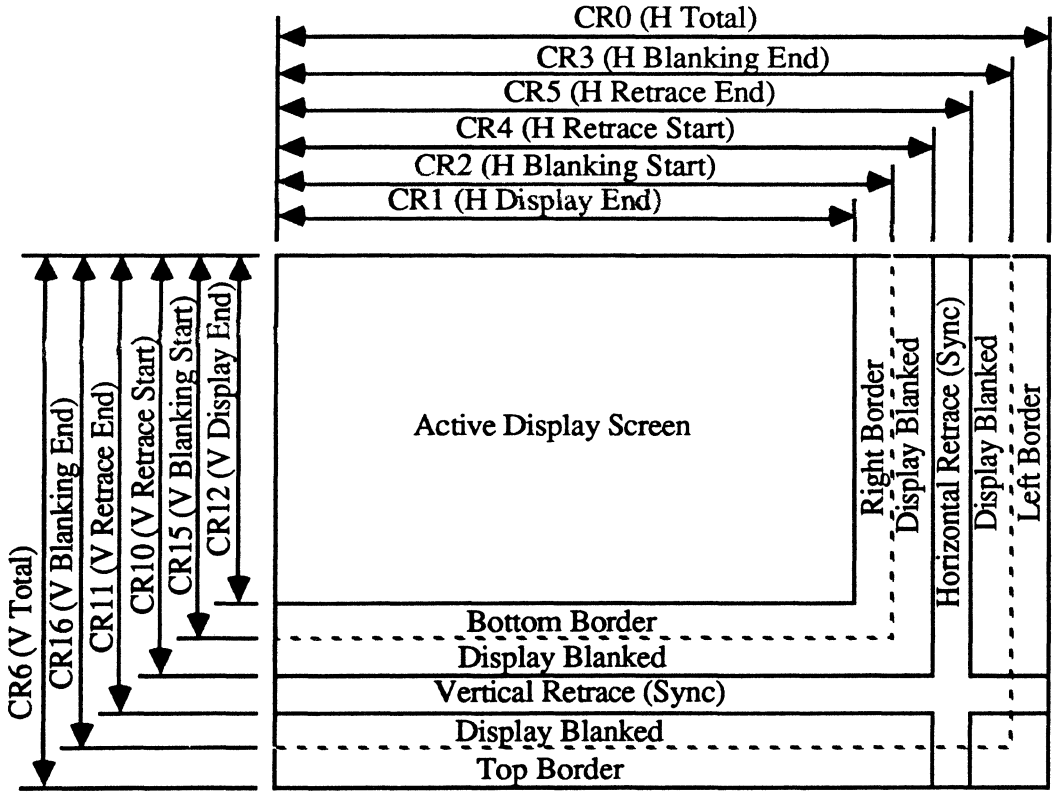


Figure 3-5. CRTC Timing Registers

Index 01

Write Protected by EREB[5] and CR11[7]

	Bit #	Description	Access	Reset By	Reset State
msb	7	Horizontal Display End Bit-7	R/W		
	6	Horizontal Display End Bit-6	R/W		
	5	Horizontal Display End Bit-5	R/W		
	4	Horizontal Display End Bit-4	R/W		
	3	Horizontal Display End Bit-3	R/W		
	2	Horizontal Display End Bit-2	R/W		
	1	Horizontal Display End Bit-1	R/W		
lsb	0	Horizontal Display End Bit-0	R/W		

The Horizontal Display Enable End register defines the total number of displayed characters in a horizontal line.

The value in the register = Total Number of Characters - 1.

Refer to Figure 3-4 (see register CR0) for a summary of CRTC timing registers.

Index 02 **Write Protected by EREB[5] and CR11[7]**

	Bit #	Description	Access	Reset By	Reset State
msb	7	Horizontal Blanking Start Bit-7	R/W		
	6	Horizontal Blanking Start Bit-6	R/W		
	5	Horizontal Blanking Start Bit-5	R/W		
	4	Horizontal Blanking Start Bit-4	R/W		
	3	Horizontal Blanking Start Bit-3	R/W		
	2	Horizontal Blanking Start Bit-2	R/W		
	1	Horizontal Blanking Start Bit-1	R/W		
lsb	0	Horizontal Blanking Start Bit-0	R/W		

The contents of this register define the time when the horizontal blanking will start. The register is defined in terms of the number of horizontal character clocks assuming character positions are numbered 0-n where position 0 is the first displayed character position at the left side of the screen. The horizontal blanking signal becomes active when the horizontal character count is equal to the contents of this register.

Refer to Figure 3-4 (see register CR0) for a summary of CRTC timing registers.

Index 03 | **Write Protected by EREB[5] and CR11[7]**

	Bit #	Description	Access	Reset By	Reset State
msb	7	Compatibility Read	R/W		
	6	Display Enable Skew Control Bit-1	R/W		
	5	Display Enable Skew Control Bit-0	R/W		
	4	Horizontal Blanking End Bit-4	R/W		
	3	Horizontal Blanking End Bit-3	R/W		
	2	Horizontal Blanking End Bit-2	R/W		
	1	Horizontal Blanking End Bit-1	R/W		
lsb	0	Horizontal Blanking End Bit-0	R/W		

The contents of this register define the time when the horizontal blanking will end. The register is defined in terms of the number of horizontal character clocks assuming character positions are numbered 0-n where position 0 is the first displayed character position at the left side of the screen.

Bit Descriptions

Bit 7 Compatibility Read: If this bit is 1, CR10 and CR11 read back at indices 10 and 11 instead of the Light Pen Registers.

Bit 6-5 Display Enable Skew Control: Prior to displaying data on the screen, the CRT controller has to access the display buffer to obtain a character to be displayed, the attribute code for it, and the character generator font information. These accesses require the display enable signal to be skewed by one character clock to allow for synchronization with horizontal and vertical retrace. The display enable skew bits in this register allow for this skew. The skew can be programmed from 0-3 character clocks as follows:

Bit-6	Bit-5	Skew in character clocks	
0	0	0	<= typical setting in the VGA
0	1	1	<= typical setting in the EGA
1	0	2	
1	1	3	

Bit 4-0 Horizontal Blanking End: The horizontal blanking signal width is determined as follows:

Value in Horizontal Blanking Start Register (CR2) + Width of Blanking Signal = value to be programmed into the Horizontal Blanking End register.

The lsb's of the horizontal character counter are compared with the contents of this register (plus bit-5 from CR5 bit-7). When a match occurs, the horizontal blanking pulse becomes inactive. The length of this register limits the length of the blanking pulse to 63 character clocks. In the IBM EGA, if the blanking interval extended beyond the end of the line, erratic behavior would result since the character counter got cleared after the number of character times programmed in the H Total register. This restriction was removed in the VGA.

Refer to Figure 3-4 (see register CR0) for a summary of CRTC timing registers.

I/O Port 3?5	CRTC Horizontal Retrace Start Register	CR4
Index 04		Write Protected by EREB[5] and CR11[7]

	Bit #	Description	Access	Reset By	Reset State
msb	7	Horizontal Retrace Start Bit-7	R/W		
	6	Horizontal Retrace Start Bit-6	R/W		
	5	Horizontal Retrace Start Bit-5	R/W		
	4	Horizontal Retrace Start Bit-4	R/W		
	3	Horizontal Retrace Start Bit-3	R/W		
	2	Horizontal Retrace Start Bit-2	R/W		
	1	Horizontal Retrace Start Bit-1	R/W		
lsb	0	Horizontal Retrace Start Bit-0	R/W		

This register defines the character position at which the Horizontal Retrace Pulse becomes active assuming character positions are numbered 0-n where position 0 is the first displayed character position at the left side of the screen. This register centers the monitor screen horizontally. The value in the register is the character count at which the Horizontal Retrace Pulse becomes active.

Refer to Figure 3-4 (see register CR0) for a summary of CRTC timing registers.

Index 05 **Write Protected by EREB[5] and CR11[7]**

	Bit #	Description	Access	Reset By	Reset State
msb	7	H Blank End Bit-5	R/W		
	6	Horizontal Retrace Delay Bit-1	R/W		
	5	Horizontal Retrace Delay Bit-0	R/W		
	4	Horizontal Retrace End Bit-4	R/W		
	3	Horizontal Retrace End Bit-3	R/W		
	2	Horizontal Retrace End Bit-2	R/W		
	1	Horizontal Retrace End Bit-1	R/W		
lsb	0	Horizontal Retrace End Bit-0	R/W		

This register defines the character position at which the Horizontal Retrace Pulse becomes inactive assuming character positions are numbered 0-n where position 0 is the first displayed character position at the left side of the screen.

Bit Descriptions

Bit 7 Horizontal Blank End Bit-5: This bit is an extension bit for H Blank End (H Blank End is 5 bits in the EGA and 6 bits in the VGA).

In the EGA, this bit was used for a different purpose. It determined whether CRT memory addresses started with even (0) or odd(1) values after a horizontal retrace. This was used in horizontal panning. The VGA now has different mechanisms to accomplish the same thing (see CR8[5-6]).

Bit 6-5 Horizontal Retrace Delay: The skew of the horizontal retrace signal is controlled by these bits. In some modes, it is necessary to provide a horizontal retrace signal that takes up the entire blanking period. The horizontal retrace signal also triggers some internal timings on the falling edge of the signal. To ensure that the signals are latched properly, the retrace signal is started before the end of the display enable signal. It is then skewed several character clock times to provide the proper screen centering.

Bit-6	Bit-5	Skew in character clocks
0	0	0
0	1	1
1	0	2
1	1	3

Bit 4-0 End Horizontal Retrace: The horizontal retrace signal becomes inactive after the character count becomes equal to the count in these bits. The width of the retrace signal is determined as follows:

> Value in Retrace Start Register (CR4) + Width of Retrace Signal = 5-bit value to be programmed into the Horizontal Retrace End register.

The five lsbs of the horizontal character counter are compared to the contents of this register. When a match occurs, the horizontal retrace pulse becomes inactive. The length of this register limits the retrace signal to 31 character clocks. In the EGA, if the retrace interval extends beyond the end of the line, erratic behavior will result since the horizontal character counter gets cleared after the number of character times programmed in the Horizontal Total register. This restriction has been removed in the VGA.

Refer to Figure 3-4 (see register CR0) for a summary of CRTC timing registers.

Index 06

Write Protected by EREB[5] and CR11[7]

	Bit #	Description	Access	Reset By	Reset State
msb	7	Vertical Total Bit-7	R/W		
	6	Vertical Total Bit-6	R/W		
	5	Vertical Total Bit-5	R/W		
	4	Vertical Total Bit-4	R/W		
	3	Vertical Total Bit-3	R/W		
	2	Vertical Total Bit-2	R/W		
	1	Vertical Total Bit-1	R/W		
lsb	0	Vertical Total Bit-0	R/W		

The Vertical Total register defines the number of horizontal raster scans on the CRT screen, including the vertical retrace. The Vertical Total register contains the low order 8 bits of a 9-bit or 10-bit register. The ninth bit is located in the CRT Controller Overflow register (CR7 bit-0). In the VGA, a tenth bit is located in CR7 bit-5.

The value programmed into Vertical Total is the total number of scan lines - 2 for both EGA and VGA.

Refer to Figure 3-4 (see register CR0) for a summary of CRTC timing registers.

I/O Port 3?5	CRTC Overflow Register	CR7
Index 07		**Write Protected by EREB[5] and CR11[7]**

	Bit #	Description		Write Protected By	Access
msb	7	Bit-9 of the V Retrace Start Register	(CR10)	EREB[5] and CR11[7]	R/W
	6	Bit-9 of the V Display End Register	(CR12)	EREB[5] and CR11[7]	R/W
	5	Bit-9 of the V Total Register	(CR6)	EREB[5] and CR11[7]	R/W
	4	Bit-8 of the Line Compare Register	(CR18)	EREB[5]	R/W
	3	Bit-8 of the Vertical Blanking Start Register	(CR15)	EREB[5] and CR11[7]	R/W
	2	Bit-8 of the Vertical Retrace Start Register	(CR10)	EREB[5] and CR11[7]	R/W
	1	Bit-8 of the Vertical Display End Register	(CR12)	EREB[5] and CR11[7]	R/W
lsb	0	Bit-8 of the Vertical Total Register	(CR6)	EREB[5] and CR11[7]	R/W

The CRT Controller Overflow register is used in conjunction with other control registers and contains the ninth and tenth bits of various other registers. Bits 0-4 of this register are the same as the original EGA; bits 5-7 were added in the VGA.

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	Byte Pan Control Bit-1	R/W		
	5	Byte Pan Control Bit-0	R/W		
	4	Screen A Preset Row Scan Bit-4	R/W		
	3	Screen A Preset Row Scan Bit-3	R/W		
	2	Screen A Preset Row Scan Bit-2	R/W		
	1	Screen A Preset Row Scan Bit-1	R/W		
lsb	0	Screen A Preset Row Scan Bit-0	R/W		

The 5 lsbs of this register specify the starting row scan count of the character cell after a vertical retrace (assuming the scan lines of a character row are numbered starting with 0). This is the start of the top half of the screen (referred to as 'Screen A') if split screen mode is in effect. The row scan counter is reset to 0 in the normal way when the count reaches the value programmed in CR9 (Character Cell Height). If this register is programmed with a value greater than the character cell height, the row scan counter will count up to 1Fh before rolling over. In text and certain graphics modes, this register can be used for smooth scrolling by setting the register value between 0 and the value in CR9. For example, by setting the Preset Row Scan to 1 instead of 0, the next frame will start at scan line 1 of the character cell, which will give the effect of shifting vertically by 1 row, or vertical scrolling. This register should be changed only during vertical retrace.

Refer also to the description of the 'Line Compare' register (CR18) for more information on how to implement split-screen mode.

In the VGA, bits 5 and 6 of this register were added to control byte panning. In the EGA, bits 5 and 6 were ignored.

Bit-6	Bit-5	Byte Panning
0	0	0 bytes (display shifts 0 pixels left)
0	1	1 byte (display shifts 8 pixels left)
1	0	2 bytes (display shifts 16 pixels left)
1	1	3 bytes (display shifts 24 pixels left)

This field, in conjunction with AR13 (Horizontal Pixel Panning), allows pixel panning in word and doubleword modes, by up to 15 and 31 pixels, respectively.

When the line compare condition becomes true and the Pixel Panning Compatibility bit (AR10 bit-5) is 1, the outputs of bits 5 and 6 are forced to 0 until the start of the next vertical sync pulse.

	Bit #	Description	Access	Reset By	Reset State
msb	7	Scan Double	R/W		
	6	Line Compare (CR18) Bit-9	R/W		
	5	Vert Blank Start (CR15) Bit-9	R/W		
	4	Character Cell Height Bit-4	R/W		
	3	Character Cell Height Bit-3	R/W		
	2	Character Cell Height Bit-2	R/W		
	1	Character Cell Height Bit-1	R/W		
lsb	0	Character Cell Height Bit-0	R/W		

Bit Descriptions

Bit 7 Scan Double: When this bit is 0, scan lines are generated for the monitor as in the normal EGA and VGA. When this bit is 1, every scan line in the normal display is displayed twice in succession. In other words, it is the completion of 'scan blocks', rather than scan lines, that clock the row scan address counter. With scan doubling enabled, each scan block is 2 scan lines high; with scan doubling disabled, each scan block is 1 scan line high. Thus, all row scan address counter-based timing (including character height and cursor and underline locations) double, as measured in scan lines, when scan doubling is enabled.

Scan doubling only effects the way in which data is displayed; it does not effect display timing. If this bit is set without changing anything else, the data currently displayed will simply appear taller; horizontal and vertical sync, blanking, etc., will remain the same.

This bit is only available in the VGA; the original EGA had no standard way of controlling scan doubling.

Bit 6 Line Compare Bit-9 - This bit is bit-9 of the Line Compare register (CR18 contains bits 0-7 and CR7 bit-4 contains bit-8). In the EGA, this bit was ignored.

Bit 5 Vertical Blank Start Bit-9 - This bit is bit-9 of the Vertical Blank Start register (CR15 contains bits 0-7 and CR7 bit-3 contains bit-8). In the EGA, this bit was ignored.

Bit 4-0 Character Cell Height - This field specifies the number of scan lines per character row minus one.

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	Cursor Disable	R/W		
	4	Cursor Start Bit-4	R/W		
	3	Cursor Start Bit-3	R/W		
	2	Cursor Start Bit-2	R/W		
	1	Cursor Start Bit-1	R/W		
lsb	0	Cursor Start Bit-0	R/W		

This register specifies the scan line of the character row where the cursor is to begin assuming the scan lines of a character row are numbered starting with 0. This is true for both the EGA and VGA. Cursor operation is different between EGA and VGA modes in two ways:

1) Cursor wraparound (the EGA does, the VGA doesn't)
2) Cursor end value (EGA programmed value = end scan line + 1; VGA value = end scan line)

Some examples of VGA cursor operation are shown in the figure below:

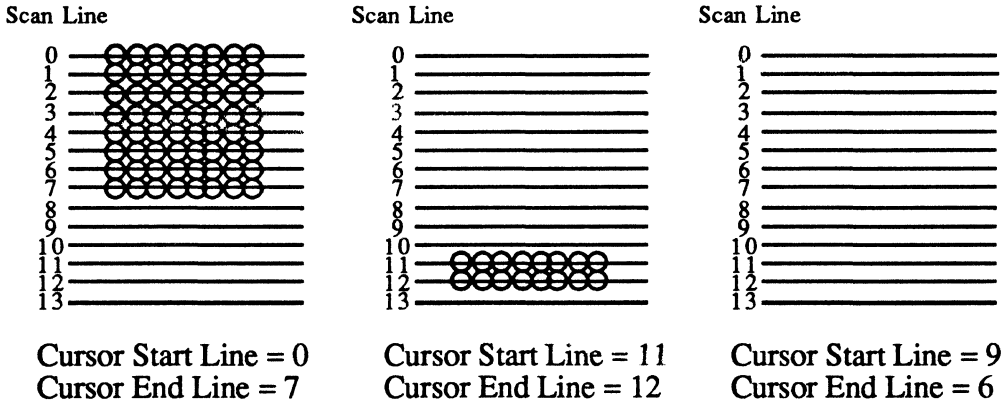


Figure 3-6. VGA-Mode CRTC Cursor Programming Examples

Note that in the EGA, if the cursor start register value was greater than the cursor end register value, the cursor wrapped around, resulting in a two-part cursor. This does not occur (no cursor is displayed), in the VGA. Note also that in the EGA, the end register value must be one greater than that required for VGA cursor programming. (Put another way, in the VGA the CRTC cursor is programmed just like a 6845 cursor).

If the cursor start value is the same as the cursor end value, a 1 line cursor will result. In the IBM EGA, only the 4 lsbs were compared, so that a one-line cursor would result if the start and end registers were identical or different by exactly 16. The 4-bit comparison is **not** duplicated by the V7VGA; a one-line cursor results only for the expected cases of the start and end registers being the same. In the VGA, the cursor size is always End-Start+1.

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	Cursor Skew Control Bit-1	R/W		
	5	Cursor Skew Control Bit-0	R/W		
	4	Cursor End Bit-4	R/W		
	3	Cursor End Bit-3	R/W		
	2	Cursor End Bit-2	R/W		
	1	Cursor End Bit-1	R/W		
lsb	0	Cursor End Bit-0	R/W		

The Cursor End register specifies the scan line of the character row where the cursor is to end assuming the scan lines of the character row are numbered starting with 0. This is one different than the EGA, which was programmed with the scan line of the character row where the cursor is to end plus one.

This register also controls cursor skew as described below:

Bit Descriptions

Bit 6-5 These bits control the cursor skew as follows:

Bit-6	Bit-5	Skew	Comment
0	0	Zero character skew	
0	1	Zero character skew	
1	0	One character skew	IBM EGA Cursor 2 characters wide in column 1
1	1	Two character skew	IBM EGA Cursor 3 characters wide in column 1

Programming this field with 0 or 1 in the IBM EGA and VGA will result in the cursor being located over the character pointed at by the cursor location registers CRE and CRF. This is the desired result. Programming this field, however, with 2 or 3 in the EGA, resulted in the cursor being located 1 or 2 characters to the right of that position in most cases, but resulted in a cursor that was more than one character wide when in column 1. This is usually non-interesting and is not emulated in the VGA, which does what you would expect..

Bit 4-0 These bits define the scan line of the character cell where the cursor is to end (plus 1 for the EGA). The 'plus 1' part of the definition for the EGA limited the maximum cursor height to 31 lines as compared to 32 for the VGA. An end value greater than the height of the character cell results in a full block cursor the same height as the character cell. An end value less than the start value results in a wrap-around cursor in the EGA and no cursor in the VGA.

Refer to the definition of the CRTC cursor start register (CRA) on the previous page for CRTC cursor programming examples and further description of cursor programming and operation.

Index 0C **Write Protected by EREB[3]**

	Bit #	Description	Access	Reset By	Reset State
msb	7	Screen A Start Address Bit-15	R/W		
	6	Screen A Start Address Bit-14	R/W		
	5	Screen A Start Address Bit-13	R/W		
	4	Screen A Start Address Bit-12	R/W		
	3	Screen A Start Address Bit-11	R/W		
	2	Screen A Start Address Bit-10	R/W		
	1	Screen A Start Address Bit-9	R/W		
lsb	0	Screen A Start Address Bit-8	R/W		

The Screen A Start Address register is a 16-bit value which specifies first display memory address after a vertical retrace at which the display on the screen begins on each screen refresh. This register contains 8 high order bits of the address, while the Screen A Start Address Low register (CRD) specifies the 8 low-order bits.

The reason that the name of this register is qualified with 'Screen A' is that under some circumstances, two logical screens may be present (split-screen mode). In this case, this register specifies the start address of the first of the two (the top one). The start address of screen B (the bottom one) is always 0. The bottom screen's start scan line on the screen is determined by the line compare register (CR18). Refer to the description of the line compare register for a diagram of split-screen mode.

Note: This register is also part of the mechanism used to identify the V7VGA chip. Any value written to this register can be read back exclusive-or'd with 'EA' hex (binary '11101010') at CRTC index 1F.

Index 0D

	Bit #	Description	Access	Reset By	Reset State
msb	7	Screen A Start Address Bit-7	R/W		
	6	Screen A Start Address Bit-6	R/W		
	5	Screen A Start Address Bit-5	R/W		
	4	Screen A Start Address Bit-4	R/W		
	3	Screen A Start Address Bit-3	R/W		
	2	Screen A Start Address Bit-2	R/W		
	1	Screen A Start Address Bit-1	R/W		
lsb	0	Screen A Start Address Bit-0	R/W		

The Screen A Start Address register is a 16-bit value which specifies the first display memory address after a vertical retrace at which the display on the screen begins on each screen refresh. This register contains the 8 low order bits of the address, while the Screen A Start Address High register (CRC) specifies the 8 high-order bits.

Index 0E

	Bit #	Description	Access	Reset By	Reset State
msb	7	Cursor Location Bit-15	R/W		
	6	Cursor Location Bit-14	R/W		
	5	Cursor Location Bit-13	R/W		
	4	Cursor Location Bit-12	R/W		
	3	Cursor Location Bit-11	R/W		
	2	Cursor Location Bit-10	R/W		
	1	Cursor Location Bit-9	R/W		
lsb	0	Cursor Location Bit-8	R/W		

The Cursor Location register contains a 16-bit value which specifies the offset of the cursor location from the start of physical display memory in character positions. This register contains the 8 high order bits of the value, while the Cursor Location Low register (CRF) specifies the 8 low-order bits.

When the screen start address registers (CRC and CRD) contain 0, programming the cursor location registers (this register and CRF) to 0 positions the cursor over the upper left character of the screen (row 1, column 1); programming them to 1 positions the cursor over the character in the next column to the right (row 1 column 2), etc. If the screen start registers are changed, the cursor will remain pointed at the same character (i.e., the cursor will effectively move the same number of characters as the displayed screen contents to remain pointed at the same displayed character). The value in the cursor location registers is relative to the start of physical display memory, not to the start of the screen.

Since information is stored in display memory as character/attribute pairs, the address of the character under the cursor will be exactly two times the value in the cursor location registers (plus the base address of the screen).

I/O Port 3?5	CRTC Cursor Location Register Low	CRF

Index 0F

	Bit #	Description	Access	Reset By	Reset State
msb	7	Cursor Location Bit-7	R/W		
	6	Cursor Location Bit-6	R/W		
	5	Cursor Location Bit-5	R/W		
	4	Cursor Location Bit-4	R/W		
	3	Cursor Location Bit-3	R/W		
	2	Cursor Location Bit-2	R/W		
	1	Cursor Location Bit-1	R/W		
lsb	0	Cursor Location Bit-0	R/W		

The Cursor Location register contains a 16-bit value which specifies the offset of the cursor location from the start of physical display memory in character positions. This register contains the 8 low order bits of the value, while the Cursor Location High register (CRE) specifies the 8 high-order bits.

When the screen start address registers (CRC and CRD) contain 0, programming the cursor location registers (this register and CRE) to 0 positions the cursor over the upper left character of the screen (row 1, column 1); programming them to 1 positions the cursor over the character in the next column to the right (row 1 column 2), etc. If the screen start registers are changed, the cursor will remain pointed at the same character (i.e., the cursor will effectively move the same number of characters as the displayed screen contents to remain pointed at the same displayed character). The value in the cursor location registers is relative to the start of physical display memory, not to the start of the screen.

Since information is stored in display memory as character/attribute pairs, the address of the character under the cursor will be exactly two times the value in the cursor location registers (plus the base address of the screen).

	Bit #	Description	3?5 Access	Reset By	Reset State
msb	7	Light Pen Address Bit-15	R		
	6	Light Pen Address Bit-14	R		
	5	Light Pen Address Bit-13	R		
	4	Light Pen Address Bit-12	R		
	3	Light Pen Address Bit-11	R		
	2	Light Pen Address Bit-10	R		
	1	Light Pen Address Bit-9	R		
lsb	0	Light Pen Address Bit-8	R		

In the EGA, the Light Pen High register contains the 8 high-order bits of the CRTC memory address counter at the time the light pen flip flop is set. The low order 8 bits are stored in the Light Pen Low register (LPENL at CRTC index 11).

In the VGA, the light pen 'register' is held in a transparent state, such that reading from LPENH at CRTC index 10 returns the 8 high-order bits of the free-running memory address counter at the time of the read. There is no way to latch the value read to make sure that the high and low readings match, or even to be sure that the memory address counter isn't in the process of changing during a read and hence temporarily invalid. This may not sound very useful, except for generating random numbers, but it emulates the IBM VGA exactly.

CRTC registers CR10-11 and LPENH/L are at conflicting locations in the EGA. The VGA provides software control (CR3 bit-7) of whether CR10-11 or LPENH/L are readable at CRTC indices 10-11.

	Bit #	Description	3?5 Access	Reset By	Reset State
msb	7	Light Pen Address Bit-7	R		
	6	Light Pen Address Bit-6	R		
	5	Light Pen Address Bit-5	R		
	4	Light Pen Address Bit-4	R		
	3	Light Pen Address Bit-3	R		
	2	Light Pen Address Bit-2	R		
	1	Light Pen Address Bit-1	R		
lsb	0	Light Pen Address Bit-0	R		

In the EGA, the Light Pen Low register contains the 8 low-order bits of the CRTC memory address counter at the time the light pen flip flop is set. The high order 8 bits are stored in the Light Pen High register (LPENH at CRTC index 10).

In the VGA, the light pen 'register' is held in a transparent state, such that reading from LPENL at CRTC index 11 returns the 8 low-order bits of the free-running memory address counter at the time of the read. There is no way to latch the value read to make sure that the high and low readings match, or even to be sure that the memory address counter isn't in the process of changing during a read and hence temporarily invalid. This may not sound very useful, except for generating random numbers, but it emulates the IBM VGA exactly.

CRTC registers CR10-11 and LPENH/L are at conflicting locations in the EGA. The VGA provides software control (CR3 bit-7) of whether CR10-11 or LPENH/L are readable at CRTC indices 10-11.

	Bit #	Description	3?5 Access	Reset By	Reset State
msb	7	Vertical Retrace Start Bit-7	W or R/W		
	6	Vertical Retrace Start Bit-6	W or R/W		
	5	Vertical Retrace Start Bit-5	W or R/W		
	4	Vertical Retrace Start Bit-4	W or R/W		
	3	Vertical Retrace Start Bit-3	W or R/W		
	2	Vertical Retrace Start Bit-2	W or R/W		
	1	Vertical Retrace Start Bit-1	W or R/W		
lsb	0	Vertical Retrace Start Bit-0	W or R/W		

The Vertical Retrace Start register was a 9-bit register in the EGA, but is a 10-bit register in the VGA. It defines the position of the vertical retrace start signal in terms of horizontal scan lines. The programmed value assumes scan lines are numbered starting from 0 at the top of the screen. The low order 8 bits are programmed through this register, while the high order bits are programmed through the CRTC Overflow register (CR7 bit-2 is Vertical Retrace Start bit-8 and CR7 bit-7 is Vertical Retrace Start bit-9).

If the 'Compatibility Read' bit, CR3 bit-7, is 0, this register is write-only at CRTC index 10 (read-back at this index returns the Light Pen High Address Register). If the Compatibility Read bit is set, the Vertical Retrace Start register may be accessed read/write at CRTC index 10 (LPENH is not accessable via the CRTC in this state).

Refer to Figure 3-4 (see register CR0) for a summary of CRTC timing registers.

	Bit #	Description	3?5 Access	Reset By	Reset State
msb	7	Write Protect CR0-7	W	Reset	0
	6	Refresh Cycle Select	W		
	5	0 = Enable Vertical Interrupt	W	Reset	1
	4	0 = Clear Vertical Interrupt	W	Reset	0
	3	Vertical Retrace End Bit-3	W		
	2	Vertical Retrace End Bit-2	W		
	1	Vertical Retrace End Bit-1	W		
lsb	0	Vertical Retrace End Bit-0	W		

If the 'Compatibility Read' bit, CR3 bit-7, is 0, this register is write-only at CRTC index 11 (read-back at this index returns the Light Pen Low Address Register). If the Compatibility Read bit is set, this register may be accessed read/write at CRTC index 11 (LPENL is not accessable via the CRTC in this state).

Bit Descriptions

Bit 7 Write Protect CR0-7 - When this bit is set to 1 in the VGA, CRTC registers 0-7 are write protected except for CR7 bit-4. This bit was a test bit in the EGA.

Bit 6 Refresh Cycle Select / Test - In the VGA, this bit controls whether 3 or 5 display memory refresh cycles are generated at the end of each horizontal scan line (0 = 3 cycles, 1 = 5 cycles). In the EGA, setting this bit to 1 caused line counter bits 7-8 to be forced to 1's (sort of a '6845-compatibility' mode).

Bit 5 Vertical Interrupt Enable - This bit controls whether interrupts are generated on the CRTINT pin of the chip from the CRTINT status bit (Input Status Register 0: 3C2 bit-7). Setting this bit to 0 enables interrupts. If interrupts are disabled, the status bit still gets set if enabled by bit-4 below.

Bit 4 Clear Vertical Interrupt - Setting this bit to 0 clears the vertical interrupt flip-flop (and de-asserts the interrupt signal if it was asserted). Setting this bit to 1 allows the vertical interrupt flip-flop to be set at the start of the next vertical retrace interval (to be exact, one scan line after Vertical Display Enable goes away). The vertical interrupt flip-flop may be read at I/O port 3C2 bit-7 (Input Status Register 0 also called the 'Feature Read' register).

Bit 3-0 Vertical Retrace End: These four bits specify the horizontal scan line count at which the vertical retrace output pulse becomes inactive assuming the scan lines are numbered starting from 0 at the top of the screen. The four bits are compared with the four lsbs of the vertical scan line counter. When the four counter bits are equal to the contents in this register, the vertical retrace is terminated. The width, W, of the vertical retrace pulse is determined by the following algorithm:

> Value in Vertical Retrace Start register (CR10) + W = 4-bit value to be programmed into the Vertical Retrace End register.

Note that the four lsbs of the algorithm result are to be programmed into this register. Thus the maximum retrace pulse width can be only 15 scan lines. Note also that in the EGA, if the blanking interval extended beyond the end of the screen, erratic behavior would result since the vertical scan line counter got cleared after the number of scan lines programmed in the vertical total register. In the VGA, this restriction has been removed.

Refer to Figure 3-4 (see register CR0) for a summary of CRTC timing registers.

Index 12

	Bit #	Description	Access	Reset By	Reset State
msb	7	Vertical Display End Bit-7	R/W		
	6	Vertical Display End Bit-6	R/W		
	5	Vertical Display End Bit-5	R/W		
	4	Vertical Display End Bit-4	R/W		
	3	Vertical Display End Bit-3	R/W		
	2	Vertical Display End Bit-2	R/W		
	1	Vertical Display End Bit-1	R/W		
lsb	0	Vertical Display End Bit-0	R/W		

The Vertical Display Enable End register defines 8 bits of a 9-bit register in the EGA and a 10-bit register in the VGA. It specifies the scan line position where the display on the screen ends. Scan lines are assumed to be numbered starting from 0 at the top of the screen. The ninth bit of the Vertical Display Enable End register is located in the CRTC Overflow register (CR7 bit-1). The tenth bit is also located in the CRTC Overflow register (CR7 bit-6).

Refer to Figure 3-4 (see register CR0) for a summary of CRTC timing registers.

Index 13

	Bit #	Description	Access	Reset By	Reset State
msb	7	Logical Screen Line Width Bit-7	R/W		
	6	Logical Screen Line Width Bit-6	R/W		
	5	Logical Screen Line Width Bit-5	R/W		
	4	Logical Screen Line Width Bit-4	R/W		
	3	Logical Screen Line Width Bit-3	R/W		
	2	Logical Screen Line Width Bit-2	R/W		
	1	Logical Screen Line Width Bit-1	R/W		
lsb	0	Logical Screen Line Width Bit-0	R/W		

The Offset register contents define the logical line width of the screen. The starting address of the next character row is determined by the value in the Offset register.

The following figure is a functional diagram of how the Offset register is used. The register start address is sent to the memory address counter. When the memory address counter counts bytes, the next line address is the current line start address + 2 times the Offset register contents. This is shown in the figure by the fact that the adder has one of the input port's least significant bits forced to 0. When the memory address counter is counting words, the next line address is the current line start address + 4 times the Offset register contents. The byte or word mode for the memory address counter is selected by the Mode Control register (CR17), bit 6. The Start Address High and Low bytes in the figure correspond to the first address after a vertical retrace at which the display on the screen begins.

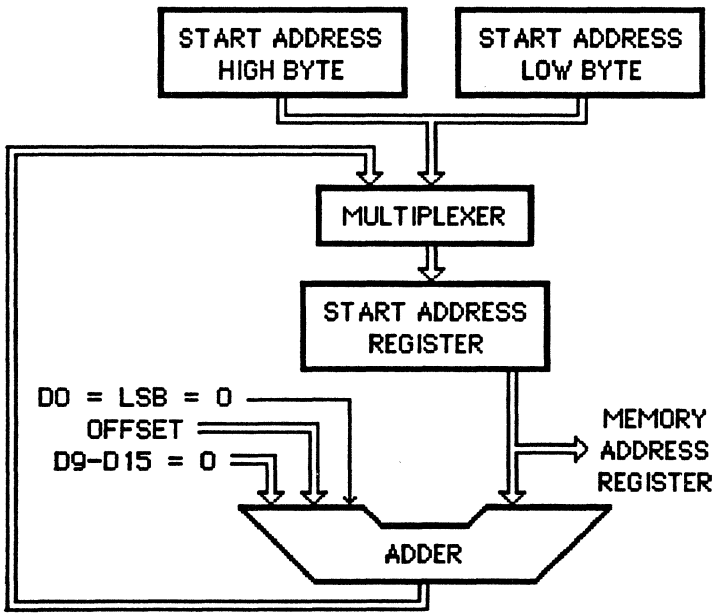


Figure 3-7. CRTC Offset Register Operation.

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	Doubleword Mode	R/W		
	5	Count by 4	R/W		
	4	Underline Row Scan Bit-4	R/W		
	3	Underline Row Scan Bit-3	R/W		
	2	Underline Row Scan Bit-2	R/W		
	1	Underline Row Scan Bit-1	R/W		
lsb	0	Underline Row Scan Bit-0	R/W		

Bit Descriptions:

Bit 7 Unused

Bit 6 Doubleword - This bit is provided for VGA compatibility and was ignored by the EGA. When this bit is set to 1, the CRTC memory address counter is shifted up two bits to provide the linear address to display memory (this is like 'word' mode except shifted up one more bit). In this mode, display memory address bit-0 is driven from CRTC memory address counter bit-12 and display memory address bit-1 is driven from CRTC memory address counter bit-13. When this bit is set, the CRTC Mode register 'Byte Mode' bit (CR17 bit-6) is ignored.

See CR17 bit-6 for further information.

Bit 5 Count by 4 - This bit is provided for VGA compatibility and was ignored in the EGA. When this bit is 1 and CR17 bit-3 (Count by 2) is 0, the memory address counter is clocked by the character clock / 4. When 'Count by 2' is 1, the memory address counter is clocked by the character clock / 2 and 'Count by 4' is ignored. When 'Count by 2' and 'Count by 4' are both 0, the memory address counter is clocked by the unmodified character clock.

Bits 4-0 Underline Row Scan - The 5 lsbs of this register specify the horizontal row scan of the character cell at which the underline will occur. The scan lines of the character cell are assumed to be numbered from the top of the cell starting at 0.

Underlining occurs in text (alphanumeric) modes only when an attribute value of binary 'b000i001' is detected (where b indicates blink and i indicates intensified).

Underlining is normally only enabled while in monochrome modes (EGA/VGA mode 7 and Hercules/MGA text modes for example) by setting this register to 13 (the last scan line of the 8x14 character cell). For color modes, this register is normally programmed to a value larger than the size of the character cell to effectively disable underlining. This is due to bits 0-2 and 4-6 of the attribute value being used for foreground and background colors respectively in color modes (activating underlining when the character attributes are set to foreground color 1 and background color 0 is usually not desirable).

Index 15

	Bit #	Description	Access	Reset By	Reset State
msb	7	Vertical Blanking Start Bit-7	R/W		
	6	Vertical Blanking Start Bit-6	R/W		
	5	Vertical Blanking Start Bit-5	R/W		
	4	Vertical Blanking Start Bit-4	R/W		
	3	Vertical Blanking Start Bit-3	R/W		
	2	Vertical Blanking Start Bit-2	R/W		
	1	Vertical Blanking Start Bit-1	R/W		
lsb	0	Vertical Blanking Start Bit-0	R/W		

This register contains the low order 8 bits of a 9-bit register in the EGA and a 10-bit register in the VGA which indicates the horizontal scan line count at which the vertical blanking pulse becomes active. Scan lines are assumed to be numbered starting from 0 at the top of the screen. The ninth bit is located in the CRTC Overflow register (CR7 bit-3). The tenth bit is located in the CRTC Character Cell Height register (CR9 bit-5).

Refer to Figure 3-4 (see register CR0) for a summary of CRTC timing registers.

Index 16

	Bit #	Description	Access	Reset By	Reset State
msb	7	Vertical Blanking End Bit-7	R/W		
	6	Vertical Blanking End Bit-6	R/W		
	5	Vertical Blanking End Bit-5	R/W		
	4	Vertical Blanking End Bit-4	R/W		
	3	Vertical Blanking End Bit-3	R/W		
	2	Vertical Blanking End Bit-2	R/W		
	1	Vertical Blanking End Bit-1	R/W		
lsb	0	Vertical Blanking End Bit-0	R/W		

This register specifies the horizontal scan line count at which the vertical blanking pulse becomes inactive assuming the scan lines are numbered starting from 0 at the top of the screen. The vertical blanking width (W) is determined from the following algorithm:

Value of Start Vertical Blanking register (CR15) + W = Value to be programmed into the Vertical Blanking End register.

The three most significant bits of this register were ignored in EGA mode. When the five (EGA) or eight (VGA) least significant bits of the vertical scan line counter are equal to the value in this register, vertical blanking is terminated. Note that the maximum vertical blanking interval is limited to 31 scan lines for the EGA and 255 for VGA.

In the EGA, if the blanking interval extended beyond the end of the screen, erratic behavior would result since the vertical scan line counter got cleared after the number of line times programmed in the vertical total register. This restriction has been removed in the VGA.

Refer to Figure 3-4 (see register CR0) for a summary of CRTC timing registers.

Index 17

	Bit #	Description	Access	Reset By	Reset State
msb	7	H/V Retrace Enable	R/W	Reset	0
	6	Byte Mode (1), Word Mode (0)	R/W		
	5	Address Wrap	R/W		
	4	Unused (IBM EGA: CRTC Output Driver Control)	-		
	3	Count by Two	R/W		
	2	Multiply Vertical by 2 (CR6,10,12,15,18)	R/W		
	1	Select Row Scan Counter	R/W		
lsb	0	Compatibility Mode Support	R/W		

The Mode Control register is a multi-function register with each bit defining a different option. The following is a description of these bits:

Bit Descriptions

Bit 7 H/V Retrace Enable: 0 disables vertical and horizontal retrace. 1 enables vertical and horizontal retrace.

Bit 6 Byte Mode: 1 selects byte mode. 0 selects word mode. Word mode causes the memory address counter bits to shift down one bit, and the most significant bit of the counter appears on the least significant bit of the memory address output. If CR14 bit-6 is set to 1 (Doubleword Mode), this bit is ignored. Doubleword mode causes the memory address counter bits to shift down 2 bits. For more information on how the two low order address bits are handled in doubleword mode, refer to the extension registers F6, F9, and FC.

Internal Memory Address Counter/Output Multiplexer Relationship

CRTC Out Pin	Byte Address Mode	Word Address Mode	Doubleword Address Mode
xA15	MA15	MA14	MA13
xA14	MA14	MA13	MA12
xA13	MA13	MA12	MA11
.	.	.	.
.	.	.	.
.	.	.	.
xA3	MA3	MA2	MA1
xA2	MA2	MA1	MA0
xA1	MA1	MA0	MA13
xA0	MA0	MA13 / MA15	MA12

Bit 5 Address Wrap: This bit selects the correct memory address counter bit to be output on xA0 in word mode. MA13 is selected if this bit is 0 and MA15 is selected if this bit is 1. When byte mode is selected through bit-6 of this register, MA0 counter output appears on the xA0 output pin. This bit is set to 0 in the IBM EGA when less than 64K of memory is configured and to 1 if 256K of memory is configured. The V7VGA comes standard with 256K of memory configured, so this bit is normally always set to 1. If set to 0, MA14 and MA15 are forced to 0.

Bit 4 Unused (this bit was programmed to 0 in the IBM EGA to enable the CRTC output drivers)

Bit 3 Count by Two: This bit defines whether the contents of the Offset register (CR13) are a word or a double word value. When this bit is 0, the memory address counter is clocked by the character clock. When this bit is 1, the memory address is clocked by the character clock divided by 2. This bit also creates either a byte or word refresh address for the display memory.

Bit 2 Multiply Vertical by 2: This bit controls the vertical resolution capability of the CRT Controller. In the EGA, the vertical counter had a maximum resolution of 512 scan lines as defined by the Vertical Total register (1024 in the VGA). If the vertical retrace counter is clocked with the horizontal retrace clock divided by 2, the vertical resolution is doubled to 1024 horizontal scan lines in the EGA (2048 in the VGA). Setting this bit to 0 selects the horizontal retrace clock; setting it to 1 selects the horizontal retrace clock divided by 2.

If this bit is set, the following vertical registers must be programmed to half their normal value to result in the same number of scan lines:

CR6	Vertical Total
CR10	Vertical Retrace Start
CR12	Vertical Display End
CR15	Vertical Blanking Start
CR18	Line Compare

Note that these are the same registers that have overflow bits in the CRTC Overflow register CR7. (In the VGA, some also have additional overflow bits in CR9).

Bit 1 Select Row Scan Counter: This bit allows compatibility with the Hercules graphics card and other 400 line graphics systems. If this bit is set to 0, row scan counter bit-1 is substituted for memory address bit 14 during active display time. If this bit is set to 1, no such substitution takes place.

Bit 0 Compatibility Mode Support: This bit allows compatibility with the IBM Color Graphics Adapter. When this bit is 0, row scan counter bit-0 is substituted for memory address bit-13 during active display time. When this bit is 1, no such substitution takes place.

Index 18

	Bit #	Description	Access	Reset By	Reset State
msb	7	Line Compare Bit-7	R/W		
	6	Line Compare Bit-6	R/W		
	5	Line Compare Bit-5	R/W		
	4	Line Compare Bit-4	R/W		
	3	Line Compare Bit-3	R/W		
	2	Line Compare Bit-2	R/W		
	1	Line Compare Bit-1	R/W		
lsb	0	Line Compare Bit-0	R/W		

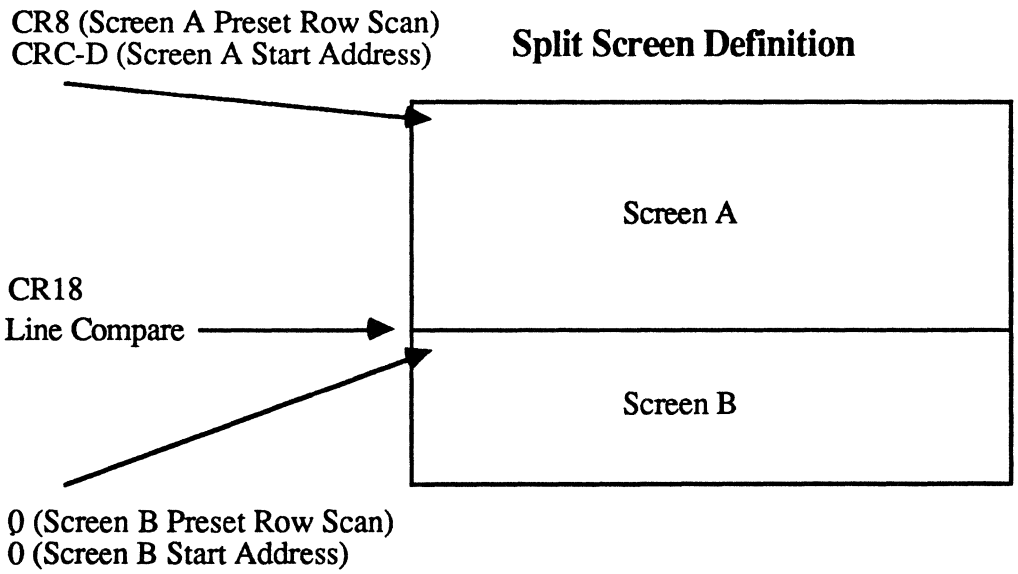
The Line Compare register is used to implement the split screen function. It was a 9-bit register in the EGA which has been expanded to a 10-bit register in the VGA. The 8 lsbs of the Line Compare are in this register, bit-8 is in the CRTC Overflow register CR7 bit-4, and in the VGA, bit-9 is in the Character Cell Height register at CR9 bit-6. When the horizontal scan line counter value is equal to the contents of the Line Compare register, the memory address generator and the character row scan count are cleared.

The screen area above where the Line Compare register points is called Screen A and the screen area below that point is called Screen B (see figure below). In the standard EGA and VGA, Screen A may be smooth scrolled vertically and panned horizontally, but Screen B cannot. The VGA provides a control over whether screen B pans with screen A or is stationary when screen A is panned (AR10 bit-5).

The Line Compare register determines the point where Screen A ends and Screen B begins. It is typically set to a value of FF (along with one bits in CR7 bit-4 and CR9 bit-6) to disable the split-screen feature (no comparison ever occurs so Screen B never starts).

Figure 3-8.

Split Screen Mode



Index 1F

	Bit #	Description	Access	Reset By	Reset State
msb	7	V7VGA Identification: CRTC Register C bit-7 xor 1	R		
	6	V7VGA Identification: CRTC Register C bit-6 xor 1	R		
	5	V7VGA Identification: CRTC Register C bit-5 xor 1	R		
	4	V7VGA Identification: CRTC Register C bit-4 xor 0	R		
	3	V7VGA Identification: CRTC Register C bit-3 xor 1	R		
	2	V7VGA Identification: CRTC Register C bit-2 xor 0	R		
	1	V7VGA Identification: CRTC Register C bit-1 xor 1	R		
lsb	0	V7VGA Identification: CRTC Register C bit-0 xor 0	R		

This read-only register may be used to determine whether the graphics adapter supports the Video Seven extension registers. The value read back from this register is the current value in CRTC register C (Screen A Start Address High) exclusive-or'd with hex 'EA'. For example, if CRC contains 0, this register will read back hex 'EA'; if CRC contains hex 'FF', this register will read back hex '15'; and so forth. Any EGA, VGA, CGA, or MGA can safely be examined for presence of Video Seven extensions with no ill effects from programming on the CRTC index register, reading back CRTC data registers, and, if desired for absolute certainty, programming CRTC register C (which is a read/write register even in 6845s and old EGA CRTCs).

Once it is known that the Video Seven extension registers can safely be accessed, the chip revision registers (at extensions indices 8Eh and 8Fh) may be read to determine the exact nature and capabilities of the chip(s) installed.

Writes to this register are ignored.

Index 22

	Bit #	Description	Access	Reset By	Reset State
msb	7	Graphics Controller Data Latch N Bit-7	R		
	6	Graphics Controller Data Latch N Bit-6	R		
	5	Graphics Controller Data Latch N Bit-5	R		
	4	Graphics Controller Data Latch N Bit-4	R		
	3	Graphics Controller Data Latch N Bit-3	R		
	2	Graphics Controller Data Latch N Bit-2	R		
	1	Graphics Controller Data Latch N Bit-1	R		
lsb	0	Graphics Controller Data Latch N Bit-0	R		

This register may be used to read the state of Graphics Controller Data Latch 'n', where 'n' is controlled by the Graphics Controller Read Map Select Register (GR4 bits 0-1) and is in the range 0-3.

Writes to this register are not decoded and will be ignored.

Index 24

	Bit #	Description	Access	Reset By	Reset State
msb	7	Index (0) / Data (1)	R		
	6	-unused-	-		
	5	Palette Address Source	R		
	4	Attribute Controller Index 4	R		
	3	Attribute Controller Index 3	R		
	2	Attribute Controller Index 2	R		
	1	Attribute Controller Index 1	R		
lsb	0	Attribute Controller Index 0	R		

This register may be used to read back the state of the attribute controller index/data latch. In the V7VGA, reading from this CRTC index returns the same information as returned by extensions index 83 (note that ER83 is also writable, however).

Writes to this register are not decoded and will be ignored.

Index 30-3F

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	-unused-	-		
	4	-unused-	-		
	3	-unused-	-		
	2	-unused-	-		
	1	-unused-	-		
lsb	0	Clear Display Enable Flip Flop	W		

Writing odd data values to CRTC index 30-3F causes the vertical display enable flip-flop to be cleared. The flip-flop is automatically set by reaching vertical total. The effect of this is to force a longer vertical retrace period. There are two side effects of terminating vertical display enable early: first, the screen blanks early for one frame causing a minor visual disturbance and second, the sequencer gives more display memory cycles to the CPU because vertical display is not enabled.

Reads from this register are not decoded and will return indeterminate data.

Graphics Controller Registers

6/21/88

The Graphics Controller directs data from the display memory to the Attribute Controller and the CPU. The Graphics Controller registers are listed in the following table:

Page	Abbrev	Register Name	Read/Write Port
3-64	GRX	Graphics Controller Index Register	3CE
3-65	GR0	Set/Reset Register	3CF Index 00
3-66	GR1	Enable Set/Reset Register	3CF Index 01
3-67	GR2	Color Compare Register	3CF Index 02
3-68	GR3	Data Rotate Register	3CF Index 03
3-69	GR4	Read Map Select Register	3CF Index 04
3-70	GR5	Mode Register	3CF Index 05
3-72	GR6	Miscellaneous Register	3CF Index 06
3-73	GR7	Color Don't Care Register	3CF Index 07
3-74	GR8	Bit Mask Register	3CF Index 08

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	-unused-	-		
	4	-unused-	-		
	3	Graphics Controller Index Bit-3	R/W		
	2	Graphics Controller Index Bit-2	R/W		
	1	Graphics Controller Index Bit-1	R/W		
lsb	0	Graphics Controller Index Bit-0	R/W		

The Graphics Controller Index Register points to other internal registers of the Graphics Controller. The four least significant bits determine the register which will be pointed to in the next Graphics Controller register read/write operation.

Index 00

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	-unused-	-		
	4	-unused-	-		
	3	Set/Reset Plane 3	R/W		
	2	Set/Reset Plane 2	R/W		
	1	Set/Reset Plane 1	R/W		
lsb	0	Set/Reset Plane 0	R/W		

The bits in this register define the value written to the corresponding memory planes when the processor does a memory write with Write Mode 0 selected and the Set/Reset mode enabled with the Enable Set/Reset Register. Note that this can be done on an individual memory plane with separate OUT commands to the Enable Set/Reset Register.

For example, if the Set/Reset register contents are 1101, then a write to display memory will result in the following:

	7	6	5	4	3	2	1	0
Plane 3	1	1	1	1	1	1	1	1
Plane 2	1	1	1	1	1	1	1	1
Plane 1	0	0	0	0	0	0	0	0
Plane 0	1	1	1	1	1	1	1	1

This assumes the Enable Set/Reset register (GR1) contents are 1111, all planes are enabled (Sequencer SR2 = 1111) and all bits are unmasked (GR8 = FFh).

Index 01

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	-unused-	-		
	4	-unused-	-		
	3	Enable Set/Reset Plane 3	R/W		
	2	Enable Set/Reset Plane 2	R/W		
	1	Enable Set/Reset Plane 1	R/W		
lsb	0	Enable Set/Reset Plane 0	R/W		

The bits in this register enable the Set/Reset function in conjunction with the Set/Reset Register. If the mode register is programmed to write mode 0, the contents of the Set/Reset register are written to the respective display memory planes. If the write mode is 0 and Set/Reset is not enabled on a plane, the plane is written with the data from the CPU data bus.

For example, if the Set/Reset register (R0) contents are 0100, the contents of the Enable Set/Reset register (R1) are 0101 (enable Set/Reset on planes 0 and 2) and a write of 11001101 is done to display memory, it will result in the following:

	7	6	5	4	3	2	1	0
Plane 3	1	1	0	0	1	1	0	1
Plane 2	1	1	1	1	1	1	1	1
Plane 1	1	1	0	0	1	1	0	1
Plane 0	0	0	0	0	0	0	0	0

This assumes write mode 0, all planes enabled (Sequencer SR2 = 1111), and all bits unmasked (GR8 = FFh).

Index 02

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	-unused-	-		
	4	-unused-	-		
	3	Color Compare Plane 3	R/W		
	2	Color Compare Plane 2	R/W		
	1	Color Compare Plane 1	R/W		
lsb	0	Color Compare Plane 0	R/W		

If the Mode Register has Read mode set, the data read from display memory planes 0 to 3 is compared to the bits 0 to 3 in the Color Compare Register. A match will cause a 1 to be output on the corresponding data bus bit.

For example, if the contents of the Color Compare register are 0011 (to compare planes 0 and 1) and the contents of the plane are as follows:

	7	6	5	4	3	2	1	0
Plane 3	0	0	0	0	0	0	0	0
Plane 2	1	1	1	1	1	1	1	0
Plane 1	0	0	0	0	0	0	0	1
Plane 0	1	1	1	1	1	1	1	1

The data bus will contain the following:

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1

This assumes the Color Don't Care register (GR7) = 1111.

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	-unused-	-		
	4	Function Select Bit-1	R/W		
	3	Function Select Bit-0	R/W		
	2	Rotate Count Bit-2	R/W		
	1	Rotate Count Bit-1	R/W		
lsb	0	Rotate Count Bit-0	R/W		

This register performs a right rotate function on the data written by the CPU. If the Mode Register is programmed for write mode 0, the value in the Rotate Count field represents the number of bits the CPU data will be right rotated during CPU write cycles.

The Function Select Bits 3 and 4 allow data in the CPU latches to be logically operated on by the data written into the memory. The bits operate as shown:

Bit-4	Bit-3	Operation
0	0	No change
0	1	Logical 'AND' between Data and latched data
1	0	Logical 'OR' between Data and latched data
1	1	Logical 'XOR' between Data and latched data

Data may be any of the options available with the Write Mode Register. Data cannot be the CPU latched data. For example, if the contents of the Data Rotate register bits 2-0 are 011 and a program is writing CAh to display memory:

PC Data = 1 1 0 0 1 0 1 0 = CAh
the Result Stored is = 0 1 0 1 1 0 0 1 = 59h (the result is shifted 3 bits to the right, circularily)

If the contents of Data Rotate register bits 3 and 4 are binary 11 (the XOR function) and the Graphics CPU latches have been loaded (by a read of display memory) data will appear as follows:

	7	6	5	4	3	2	1	0
Plane 3 Latch	1	0	0	1	0	1	1	1
Plane 2 Latch	0	1	1	1	1	0	0	1
Plane 1 Latch	1	1	0	1	0	1	0	1
Plane 0 Latch	1	0	1	1	0	0	0	0

With a write from the PC with data 00111100, an XOR function will be performed on the PC data and the CPU latch, with a result in display memory as follows:

	7	6	5	4	3	2	1	0
Plane 3	1	0	1	0	1	0	1	1
Plane 2	0	1	0	0	0	1	0	1
Plane 1	1	1	1	0	1	0	0	1
Plane 0	1	0	0	0	1	1	0	0

This assumes write mode 1, all planes enabled (Sequencer SR2 = 1111) and all bits unmasked (GR8 = FF).

Index 04

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	-unused-	-		
	4	-unused-	-		
	3	-unused-	-		
	2	-unused-	-		
	1	Map Select Bit-1	R/W		
lsb	0	Map Select Bit-0	R/W		

The two least significant bits of this register designate the memory plane from which the CPU reads the data. This register does not effect the read operation performed through the Color Compare register. The four memory planes are selected as follows:

Bit-1	Bit-0	Plane Selected
0	0	Plane 0
0	1	Plane 1
1	0	Plane 2
1	1	Plane 3

If the double odd/even bit (SR4 bit-3, also called 'Chain4') is set, the contents of this register are ignored.

Index 05

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	Shift 256	R/W		
	5	Shift Register	R/W		
	4	Odd/Even	R/W		
	3	Read Mode	R/W		
	2	Test Condition	R/W		
	1	Write Mode Bit-1	R/W		
lsb	0	Write Mode Bit-0	R/W		

Bit Descriptions

Bit 6 Shift 256 - This bit is implemented for VGA compatibility and was ignored by the EGA. When this bit is 1, the video shift register is set up for 256-color mode. If this bit is set, bit-5 is ignored.

Bit 5 Shift Register - The data bits in the memory planes 0-3 are represented as M0D0-M0D7, M1D0-M1D7, M2D0-M2D7, and M3D0-M3D7 respectively. When this bit is 1, the data in the four serial shift registers will be formatted as follows:

MSB							LSB	Output to:
M1D0	M1D2	M1D4	M1D6	M0D0	M0D2	M0D4	M0D6	ATR0
M1D1	M1D3	M1D5	M1D7	M0D1	M0D3	M0D5	M0D7	ATR1
M3D0	M3D2	M3D4	M3D6	M2D0	M2D2	M2D4	M2D6	ATR2
M3D1	M3D3	M3D5	M3D7	M2D1	M2D3	M2D5	M2D7	ATR3

The least significant bit is shifted out first.

When this bit is 0, M0D7-M0D0, M1D7-M1D0, M2D7-M2D0, and M3D7-M3D0 are shifted out with bit-7 going out first in all cases. The outputs are ATR0-ATR3 respectively for the M0-M3 planes, and are internally connected to Attribute Controller inputs 0-3 in the V7VGA chip.

Bit 4 Odd/Even - Setting this bit to 1 will put the Graphics Controller in the Odd/Even addressing mode. This option is used to emulate the CGA. The function of this bit should track the function of bit-2 of the Sequencer Memory Mode register (note: the binary values will be opposite).

Bit 3 Read Mode - When this bit equals 0 the CPU reads the data from the display memory planes. The plane is selected through the Read Map Select register. When this bit is 1 the CPU reads the result of the logical comparison between the data from the four display memory planes and the contents of the Color Compare register.

(continued)

Bit 2 Test Condition - This bit is ignored by the V7VGA. Setting this bit to 1 in the IBM EGA disabled various Graphics Controller outputs for chip testing.

Bit 1-0 Write Mode

Mode 0: Each of the four display memory planes is written with the CPU data rotated by the number of counts in the Rotate register. This is always true except when the Set/Reset register is enabled for any of the four planes. In this case the corresponding plane is written with the data stored in the Set/Reset register.

Mode 1: Each of the four display memory planes is written with the data in the CPU latches. These latches are loaded during a previous CPU read operation. Bit mask values are over-ridden in this mode. The effect is the same as if the bit mask register is programmed to all zeroes.

Mode 2: Memory planes 0-3 are filled with the value of data bits 0-3, respectively. For example, memory plane 0 is filled with the value of data bit-0, memory plane 1 is filled with the value of data bus bit-1, etc.

Mode 3: This mode is implemented for VGA compatibility and is undefined in the IBM EGA. In this mode, the CPU data is rotated, ANDed with the Bit Mask register, and the result fed into the bit mask in place of the Bit Mask register. In addition, Set/Reset is enabled for all planes in this mode.

Index 06

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	-unused-	-		
	4	-unused-	-		
	3	Memory Map Bit-1	R/W		
	2	Memory Map Bit-0	R/W		
	1	Chain Odd Maps to Even	R/W		
lsb	0	Graphics Mode	R/W		

Bit Descriptions

Bit 3-2 Memory Map - These bits control the mapping of the address memory buffers into the CPU address space.

Memory Map 0: A000h for 128K
Memory Map 1: A000h for 64K
Memory Map 2: B000h for 32K
Memory Map 3: B800h for 32K

Bit 1 Chain Odd Maps to Even - When this bit is 1, CPU address bit A0 is replaced by a higher order address bit. The contents of A0 determine which memory plane is selected. Zero selects planes 0 and 2, one selects planes 1 and 3.

Bit 0 Graphics Mode - When this bit is 1, graphics mode is selected. This will disable the character generator latches.

Index 07

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	-unused-	-		
	4	-unused-	-		
	3	Color Plane 3 = Don't Care	R/W		
	2	Color Plane 2 = Don't Care	R/W		
	1	Color Plane 1 = Don't Care	R/W		
lsb	0	Color Plane 0 = Don't Care	R/W		

Bit Descriptions

Bit 3 0 indicates that color plane 3 is a "don't care" when the Color Compare register test is performed.

Bit 2 0 indicates that color plane 2 is a "don't care" when the Color Compare register test is performed.

Bit 1 0 indicates that color plane 1 is a "don't care" when the Color Compare register test is performed.

Bit 0 0 indicates that color plane 0 is a "don't care" when the Color Compare register test is performed.

For example, if the contents of the Color Compare register (GR2) are 0011 (to compare planes 0 and 1) and the contents of the Color Don't Care register (GR7) are 1011 (ignore plane 2)

	7	6	5	4	3	2	1	0
Plane 0	1	1	1	1	1	1	1	1
Plane 1	0	1	0	0	1	1	0	1
Plane 2	1	1	0	1	0	1	1	0
Plane 3	0	0	0	1	1	1	0	0

The data bus will contain the following:

7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	1

Index 08

	Bit #	Description	Access	Reset By	Reset State
msb	7	Write Enable Data Bit-7	R/W		
	6	Write Enable Data Bit-6	R/W		
	5	Write Enable Data Bit-5	R/W		
	4	Write Enable Data Bit-4	R/W		
	3	Write Enable Data Bit-3	R/W		
	2	Write Enable Data Bit-2	R/W		
	1	Write Enable Data Bit-1	R/W		
lsb	0	Write Enable Data Bit-0	R/W		

Any bit programmed to 0 in this register will cause the corresponding bit in each of the four memory planes to be immune to change. The data written into memory in this case will be the data which was read in the previous cycle, and was stored in an internal latch on the Graphics Controller.

Any bit programmed to 1 will allow unrestricted manipulation of the data in the corresponding bit in each of the four memory planes.

The bit mask is applicable to any data written by the CPU, including rotate, logical functions (AND, OR, XOR), Set/Reset and No Change. The data to be preserved using the bit mask must be latched internally by reading the location. The bit mask applies to all the four planes simultaneously.

For example, if the contents of the Bit Mask register are 01101001 and the data latches have been loaded as follows:

	7	6	5	4	3	2	1	0
Plane 0 Latch	1	0	1	0	1	0	1	0
Plane 1 Latch	1	1	0	0	1	1	0	1
Plane 2 Latch	0	0	1	0	1	0	1	1
Plane 3 Latch	0	1	0	1	0	0	1	0

With a write from the PC with data 01100110, will result in display memory as follows:

	7	6	5	4	3	2	1	0	
Plane 3	0	1	1	1	0	0	1	0	
Plane 2	0	1	1	0	0	0	1	0	
Plane 1	1	1	1	0	0	1	0	0	
Plane 0	1	1	1	0	0	0	1	0	
Effect	L	B	B	L	B	L	L	B	(L=Latched data, B=Bus Data)

This assumes all planes are enabled (Sequencer SR2 = 1111).

Attribute Controller Registers

6/21/88

The Attribute Controller provides a palette of 16 colors selectable from a possible 64. The Attribute Controller also controls blinking and underline operations.

The Attribute Controller registers are summarized in the following table:

Page	Abbrev	Register Name	Port Address	Port Address
3-76	ARX	Attribute Controller Index Register	3C0 (R/W)	3C5 index 83 (R/W)
3-77	AR0-F	Palette Registers	3C0 index 00-0F (W)	3C1 index 00-0F (R)
3-78	AR10	Mode Control	3C0 index 10 (W)	3C1 index 10 (R)
3-80	AR11	Overscan Color	3C0 index 11 (W)	3C1 index 11 (R)
3-81	AR12	Color Plane Enable	3C0 index 12 (W)	3C1 index 12 (R)
3-82	AR13	Horizontal Pixel Panning	3C0 index 13 (W)	3C1 index 13 (R)
3-83	AR14	Color Select	3C0 index 14 (W)	3C1 index 14 (R)

Note: The Attribute Controller Index register (ARX) is readable at extensions index 83 for state save and restore. An extra bit is available (the data/index pointer) when reads are performed at the extension port. This bit is not available when ARX is read at the 3C0 port. Writes to ARX at the 3C0 port toggle the data/index pointer; writes to ARX at 3C0 and read/write accesses at the extension port do not. In addition, the Attribute Controller Index register is also readable at CRTC index 24 for VGA compatibility.

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	Palette Address Source	R/W		
	4	Attribute Controller Index Bit-4	R/W		
	3	Attribute Controller Index Bit-3	R/W		
	2	Attribute Controller Index Bit-2	R/W		
	1	Attribute Controller Index Bit-1	R/W		
lsb	0	Attribute Controller Index Bit-0	R/W		

The Attribute Index Register points to the other internal registers of the Attribute Controller. The five least significant bits determine which data register is accessed on subsequent data port I/O operations. The index register is accessed at the same I/O port address as the data registers in the standard EGA/VGA; accesses to 3C0 are therefore directed to index and data on alternate accesses. The 3C0 I/O port index/data pointer may be initialized for access of the index register by reading the Display Status register (also called Status Register 1) at I/O port 3BA/3DA.

Attribute Controller operations are further complicated in the original IBM EGA by having only write access to both index and data. There is no provision in the standard EGA for determining the current state of the Attribute Controller registers or the flip flop which determines whether index or data registers are to be accessed next at port 3C0. To minimize these problems, the V7VGA Attribute Controller implements two extensions to the basic functionality of the standard EGA:

1) The Attribute Controller index may be read at 3C0; the data registers may be read at 3C1.

2) An alternate port (extensions index 83 of port 3C4/3C5) is provided to read or write the flip flop state which determines index or data access at 3C0. For convenience, the remainder of the Attribute Controller Index register bits may also be read or written at the extension port.

The data/index toggle indicates whether the Attribute Controller is ready to accept an access to its index register (0) or its data registers (1) for read or write accesses to I/O port 3C0. The toggle is cleared (to set the Attribute Controller for index accesses at port 3C0) by reading I/O port 3BA or 3DA (Status Register 1). The toggle is inverted by writes to I/O port 3C0 (and not by reads).

Bit Descriptions

Bit 7 This bit is unused.

Bit 6 This bit is unused.

Bit 5 Video Enable - When this bit is 0, the screen displays the color indicated by overscan register AR11 (normally black); when set to 1, normal video display is enabled. In the standard EGA/VGA, this bit also selects the address source for the palette registers (0 = CPU and 1 = Video), which requires that CPU writes to the palette registers only take place when this bit is 0 (or else the data will be written to random palette register locations as determined by the video data stream at the time of the write). The V7VGA duplicates this structure.

Bit 4-0 These bits form a 5-bit field for storing an index to the data registers in the Attribute Controller.

Index 00-0F

	Bit #	Description	Access	Reset By	Reset State
msb	7	Video 7	R/W		
	6	Video 6	R/W		
	5	Video 5	R/W		
	4	Video 4	R/W		
	3	Video 3	R/W		
	2	Video 2	R/W		
	1	Video 1	R/W		
lsb	0	Video 0	R/W		

These sixteen 8-bit registers are pointed to when the contents of the Index register are 00h through 0Fh.

These registers allow a dynamic mapping between the text attribute or graphic color input and the display color on the CRT screen. In all modes except 256-color mode, raw (premapped) color values are 4 bits maximum. The 4-bit value becomes an address into the 16 Attribute Controller color registers. The actual color output is the contents of the selected register. In 256-color mode, color values are 8 bits, and the Attribute Controller color registers are bypassed.

The standard IBM EGA and VGA have 6-bit palette registers. 6 bits, however, is not adequate for 8-bit/pixel modes. To solve this problem in the VGA, instead of just increasing the width of the palette registers, IBM added the 'AR14' mechanism. In this mechanism, video bits 6-7 are supplied from AR14 bits 2-3. In addition, video bits 4-5 may come either from color palette register bits 4-5 or from AR14 bits 0-1 (controllable by AR10[7]). The V7VGA implements the AR14 mechanism.

To support 'analog' monitors, 8 bits of color value output from the Attribute Controller gets mapped again by the external color palette (also called the 'DAC' since it also performs the 'Digital-to-Analog Conversion' of the digital color information into the final analog video output signals). The DAC uses the 8 video output bits from the attribute controller as an index into a group of 256 registers, each of which contains three six-bit color values (one each for R, G, and B). This results in 256K displayable colors on analog color displays. Analog monochrome monitors connect to the G output only (R and B are ignored), so a maximum of 64 shades of gray are displayable.

Index 10

	Bit #	Description	Access	Reset By	Reset State
msb	7	Alternate Video Source	R/W		
	6	Pixel Width	R/W		
	5	Pixel Pan Compatibility	R/W		
	4	-unused-	-		
	3	Blink Enable	R/W		
	2	Line Graphics Enable	R/W		
	1	Monochrome Attributes Enable	R/W		
lsb	0	Graphics Mode	R/W		

Bit Descriptions

Bit 7 Alternate Video Source - This bit controls the source of video output bits 4-5. If this bit is 0, video output bits 4-5 are driven from Attribute Controller palette register bits 4-5. If this bit is 1, video output bits 4-5 are driven by Color Select Register AR14 bits 0-1. If 256-color mode is in effect, this bit is ignored and video outputs 4-5 are driven from the palette.

Bit 6 Pixel Width - If this bit is set to 1, the video shift register is clocked at half speed for implementation of 256-color mode. In addition, the internal attribute controller color palette is bypassed (the 8 video bits are passed directly to the external palette).

Bit 5 Pixel Panning Compatibility - If this bit is set to 1, the output of the Horizontal Pixel Panning register (AR13) is forced to 0 when the line compare condition comes true, and remains in that state until the next vertical retrace interval. Similarly, the output of bits 6 and 5 of the Preset Row Scan register (CR8) (the byte panning controls) are also forced to 0 between line compare and end of screen. The result of this is to allow panning of the top half of a split screen without panning the bottom half. If this bit is set to 0, both halves of a split screen will pan together.

Note that the IBM VGA forces AR13's outputs to 0 even in 9-dot mode, which results in a 1-bit left shift in that mode. The V7VGA emulates this behavior.

Bit 4 -unused-

(continued)

Bit 3 Blink Enable - Setting this bit to 1 enables character blink at a rate determined by the current vertical retrace frequency divided by 32 (16 frames in one state and 16 frames in the other state). This is approximately 1/4 of a second each at 60 Hz and about 1/3 of a second at 50Hz. This is the same rate as the cursor 'slow' blink.

Blinking is implemented by toggling data at the msb of the palette address input. This toggles the palette between registers 0-7 and 8-F. The action of this bit is effected by bit-1 of this register (Monochrome Attributes). Refer to the table below for additional details.

This bit is effective in both text and graphics modes.

Bit 2 Line Graphics Enable - Setting this bit enables the special line graphics character codes by forcing the ninth dot of a line graphics character to be identical to the eighth dot of the character. The line graphics character codes are C0h through DFh.

For 9-bit wide character modes, the left-most 8 bits are determined by data from the font tables; a ninth bit is added on the right of the character cell. Clearing this bit makes the ninth dot the same as the background. For fonts which do not use the line graphics codes from C0h to DFh, this bit should be set to "0". If character widths of 8 dots or less are selected, this bit is a don't care.

This bit is effective in text mode only; it is ignored in graphics mode.

Bit 1 Monochrome Attributes - This bit is programmed to 1 for monochrome '4-color' modes to control the way blinking is handled (see bit-3 of this register). The meaning of the pixel patterns in graphics '4-color' mode (mode 'F') are black (00), white (01), blinking (10), and intensified white (11). These patterns map to palette entries 0, 1, 4, and 5 if plane 3 is off and 8, 9, C, and D if plane 3 is on (2 bits per pixel get mapped to planes 0 and 2 with planes 1 and 3 = 0). The '10' pattern is caused to blink by placing different contents in the two palette entries corresponding to pixel pattern '10' (entries 4 and C).

This bit works in graphics mode only.

Bit 0 Graphics Mode - "1" selects graphics mode, "0" selects text mode

Summary of Operation of AR10 (in graphics mode, planes 0-2 select palette inputs A0-2)

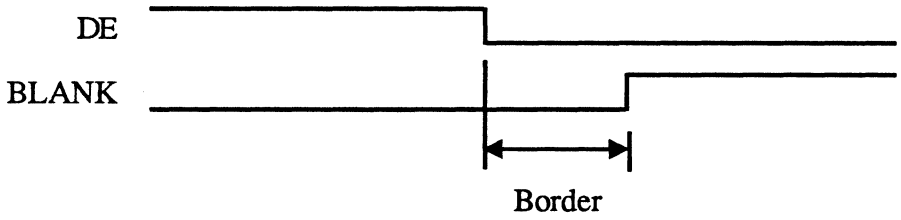
Bit-3	Bit-2	Bit-1	Bit-0	Mode	Description
0	x	x	1	Graphics	Plane 3 selects palette A3
1	x	0	1	Graphics	If plane 3 data = 0 then palette input A3 = 1 If plane 3 data = 1 then palette input A3 is blinked
1	x	1	1	Graphics	Palette input A3 is blinked (toggled on/off at the blink rate)
BL	LG	x	0	Text	If BL=0, characters don't blink (attribute bit-7 controls BG intensity) If BL=1, characters blink if attribute bit-7=1 (BG is non-intensified) Character blink toggles the character between foreground color (attribute bits 0-3) and non-intensified background color (attribute bits 4-6).

Index 11

	Bit #	Description	Access	Reset By	Reset State
msb	7	Video 7	R/W		
	6	Video 6	R/W		
	5	Video 5	R/W		
	4	Video 4	R/W		
	3	Video 3	R/W		
	2	Video 2	R/W		
	1	Video 1	R/W		
lsb	0	Video 0	R/W		

This register defines the overscan or border color displayed on the CRT screen. The overscan color is displayed when both BLANK and DE (Display Enable) signals are inactive.

Refer to the description of Palette Registers 0-F for information on how the video output bits are connected, especially bits 4-7 and how they interact with AR14.



Index 12

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	Video Status Mux Bit-1	R/W		
	4	Video Status Mux Bit-0/Cursor Blink Disable	R/W		
	3	Enable Color Plane 3	R/W		
	2	Enable Color Plane 2	R/W		
	1	Enable Color Plane 1	R/W		
lsb	0	Enable Color Plane 0	R/W		

Bit Descriptions

Bit 5-4 Display Status Mux - These bits select two of the eight outputs of the Attribute Controller (video output data during display periods and overscan color during non-display periods).

Color Plane Register		Display Status Register	
Bit 5	Bit 4	Bit-5	Bit-4
0	0	Video 2	Video 0
0	1	Video 5	Video 4
1	0	Video 3	Video 1
1	1	Video 7	Video 6

This capability can be used to run diagnostics on the color subsystem card.

Setting bit-4 will also disable the cursor blink counter. Bit-4 must be clear for the cursor blink counter to function.

Bit 3-0 Enable Color Plane - Setting any bit in this group to 1 enables the respective display memory color plane 0-3. A zero in any bit forces the corresponding display memory color plane bit to 0 at the address input of the color palette.

Index 13

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	-unused-	-		
	4	-unused-	-		
	3	Horizontal Pixel Panning Shift Count Bit-3	R/W		
	2	Horizontal Pixel Panning Shift Count Bit-2	R/W		
	1	Horizontal Pixel Panning Shift Count Bit-1	R/W		
lsb	0	Horizontal Pixel Panning Shift Count Bit-0	R/W		

Bits 0-3 of this register select the number of picture elements (pixels) to shift the display data horizontally to the left. Pixel panning is available in both alphanumeric and graphics modes. The start address register specifies the byte of the upper left corner of the screen display, and pixel panning makes it possible to move it in portions of a byte, pixel by pixel.

The amount of shift varies with the character width according to the following table:

Count	9-bit Characters	8-bit Characters	256-Color Mode
0	1 pixel left	no shift	no shift
1	2 bits left	1 pixel left	no shift
2	3 pixels left	2 pixels left	1 pixel left
3	4 pixels left	3 pixels left	1 pixel left
4	5 pixels left	4 pixels left	2 pixels left
5	6 pixels left	5 pixels left	2 pixels left
6	7 pixels left	6 pixels left	3 pixels left
7	8 pixels left	7 pixels left	3 pixels left
8-F	no shift	1 pixel right	1 pixel right

The Horizontal Pixel Panning register should be changed only during vertical retrace intervals to prevent distorting the display images.

The Offset Register (CR13) should be set to at least one more than normal when characters are not aligned with the character cell, since there is a partial character displayed on the left and the right (for 81 characters total, for example, in 80 column text mode).

Note: when the pixel panning compatibility bit (AR10[7]) is 1, the output (not the contents) of horizontal pixel panning register is forced to 0 (no shift) by a successful line compare and remains in that state until the end of vertical sync (i.e., for the rest of the screen). In other words, during split screen operation, if AR10[7] is not set, both screens pan per AR13; if AR10[7] is set, only the upper screen pans.

Note: cursor and underlining pan along with the character position they are associated with. The hardware graphics pointer does not pan (see extension registers 94, 9C-9F, and A5).

Index 14

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	-unused-	-		
	4	-unused-	-		
	3	Video 7	R/W		
	2	Video 6	R/W		
	1	Video 5	R/W		
lsb	0	Video 4	R/W		

This register is used to provide video output information. Bits 2-3 of this register are used to drive video output bits 6-7. If AR10[7] is set to 1, bits 0-1 of this register are used to drive video output bits 4-5 instead of color palette register bits 4-5.

V7VGA Extension Register Summary

6/21/88

The extension registers provide additional functions to the V7VGA beyond the standard EGA and VGA.

Page	Abbreviations		Register Name	Port	Index	Access
	ER80		-reserved-	3C5	80	
	ER81		-reserved-	3C5	81	
	ER82		-reserved-	3C5	82	
3-76	ER83	ARX	* Attribute Controller Index	3C5	83	R/W
	ER84		-reserved-	3C5	84	
	ER85		-reserved-	3C5	85	
	ER86		-reserved-	3C5	86	
	ER87		-reserved-	3C5	87	
	ER88		-reserved-	3C5	88	
	ER89		-reserved-	3C5	89	
	ER8A		-reserved-	3C5	8A	
	ER8B		-reserved-	3C5	8B	
	ER8C		-reserved-	3C5	8C	
	ER8D		-reserved-	3C5	8D	
3-86	ER8E	REV	Chip Revision Level	3C5	8E	R
3-86	ER8F	REV	Chip Revision Level	3C5	8F	R
	ER90		-reserved-	3C5	90	
	ER91		-reserved-	3C5	91	
	ER92		-reserved-	3C5	92	
	ER93		-reserved-	3C5	93	
3-87	ER94	PPA	Pointer Pattern Address	3C5	94	R/W
	ER95		-reserved-	3C5	95	
	ER96		-reserved-	3C5	96	
	ER97		-reserved-	3C5	97	
	ER98		-reserved-	3C5	98	
	ER99		-reserved-	3C5	99	
	ER9A		-reserved-	3C5	9A	
	ER9B		-reserved-	3C5	9B	
3-89	ER9C	PXH	Pointer Horizontal Position High	3C5	9C	R/W
3-90	ER9D	PXL	Pointer Horizontal Position Low	3C5	9D	R/W
3-91	ER9E	PYH	Pointer Vertical Position High	3C5	9E	R/W
3-92	ER9F	PYL	Pointer Vertical Position Low	3C5	9F	R/W
3-93	ERA0	GRL0	GC Memory Latch 0	3C5	A0	R/W
3-94	ERA1	GRL1	GC Memory Latch 1	3C5	A1	R/W
3-95	ERA2	GRL2	GC Memory Latch 2	3C5	A2	R/W
3-96	ERA3	GRL3	GC Memory Latch 3	3C5	A3	R/W
3-97	ERA4	CLK	Clock Select	3C5	A4	R/W
3-98	ERA5	CURS	Cursor Attributes	3C5	A5	R/W
	ERA6		-reserved-	3C5	A6	
	ERA7		-reserved-	3C5	A7	
	ERA8		-reserved-	3C5	A8	
	ERA9		-reserved-	3C5	A9	
	ERAA		-reserved-	3C5	AA	
	ERAB		-reserved-	3C5	AB	
	ERAC		-reserved-	3C5	AC	
	ERAD		-reserved-	3C5	AD	
	ERAE		-reserved-	3C5	AE	
	ERAF		-reserved-	3C5	AF	
	ERB0-BF		-reserved-	3C5	B0-BF	

Note: The above VEGA VGA-compatible registers are only accessable with extensions enabled (see SR6)

* Duplicated VGA / EGA registers also read/write accessible as extension registers for state save/restore

V7VGA Extension Register Summary (continued)

The extension registers provide additional functions to the V7VGA beyond the standard EGA and VGA.

Page	Abbreviations		Register Name	Port	Index	Access
	ERC0-F		-reserved-	3C5	C0-CF	
	ERD0		-reserved-	3C5	D0	
	ERD1		-reserved-	3C5	D1	
	ERD2		-reserved-	3C5	D2	
	ERD3		-reserved-	3C5	D3	
	ERD4		-reserved-	3C5	D4	
	ERD5		-reserved-	3C5	D5	
	ERD6		-reserved-	3C5	D6	
	ERD7		-reserved-	3C5	D7	
	ERD8		-reserved-	3C5	D8	
	ERD9		-reserved-	3C5	D9	
	ERDA		-reserved-	3C5	DA	
	ERDB		-reserved-	3C5	DB	
	ERDC		-reserved-	3C5	DC	
	ERDD		-reserved-	3C5	DD	
	ERDE		-reserved-	3C5	DE	
	ERDF		-reserved-	3C5	DF	
	ERE0		-reserved-	3C5	E0	
	ERE1		-reserved-	3C5	E1	
	ERE2		-reserved-	3C5	E2	
	ERE3		-reserved-	3C5	E3	
	ERE4		-reserved-	3C5	E4	
	ERE5		-reserved-	3C5	E5	
	ERE6		-reserved-	3C5	E6	
	ERE7		-reserved-	3C5	E7	
	ERE8		-reserved-	3C5	E8	
	ERE9		-reserved-	3C5	E9	
3-99	EREA	SWSTB	Switch Strobe	3C5	EA	W
3-100	EREB	NMICTRL	Emulation Control	3C5	EB	R/W
3-102	EREC	FGLAT0	Foreground Latch 0	3C5	EC	R/W
3-102	ERED	FGLAT1	Foreground Latch 1	3C5	ED	R/W
3-102	EREE	FGLAT2	Foreground Latch 2	3C5	EE	R/W
3-102	EREF	FGLAT3	Foreground Latch 3	3C5	EF	R/W
3-103	ERF0	FFGLD	Fast Foreground Latch Load	3C5	F0	R/W
3-104	ERF1	FLLSTATE	Fast Latch Load State	3C5	F1	R/W
3-105	ERF2	FBGLD	Fast Background Latch Load	3C5	F2	R/W
3-106	ERF3	MWCTRL	Masked Write Control	3C5	F3	R/W
3-107	ERF4	MWMASK	Masked Write Mask	3C5	F4	R/W
3-108	ERF5	FBPAT	Foreground / Background Pattern	3C5	F5	R/W
3-109	ERF6	RAMBANK	1 Mb RAM Bank Select	3C5	F6	R/W
3-110	ERF7	SWITCH	Switch Readback	3C5	F7	R/W
3-111	ERF8	CLKCTRL	Extended Clock Control	3C5	F8	R/W
3-113	ERF9	PGSEL	Extended Page Select	3C5	F9	R/W
3-114	ERFA	FGCOLOR	Extended Foreground Color	3C5	FA	R/W
3-115	ERFB	BGCOLOR	Extended Background Color	3C5	FB	R/W
3-116	ERFC	COMPAT	Compatibility Control	3C5	FC	R/W
3-119	ERFD	TIMING	Extended Timing Select	3C5	FD	R/W
3-121	ERFE	FBCTRL	Foreground / Background Control	3C5	FE	R/W
3-122	ERFF	16BIT	16-Bit Interface Control	3C5	FF	R/W

Note: The above V7VGA-specific registers are only accessable with extensions enabled (see SR6)

Index 8E and 8F

	Bit #	Description	Access	Reset By	Reset State
msb	7	Chip Revision Bit-7	R		0
	6	Chip Revision Bit-6	R		1
	5	Chip Revision Bit-5	R		1
	4	Chip Revision Bit-4	R		1
	3	Chip Revision Bit-3	R		0
	2	Chip Revision Bit-2	R		0
	1	Chip Revision Bit-1	R		0
lsb	0	Chip Revision Bit-0	R		0 (rev 1-3), 1 (rev 4)

The chip revision is determined by reading this register. The value returned is fixed for each chip revision.

The value returned is 070h for chip revisions 1-3.
The value returned is 071h for chip revision 4.

The range of revision values from 70-7Fh is reserved for the V7VGA chip.

The range of revision values from 80-FFh is reserved for the VEGA VGA chip.

The range of revision values from 00-6Fh is reserved for future Video Seven products.

Index 94

	Bit #	Description	Access	Reset By	Reset State
msb	7	Pointer Pattern Address Bit-13	R/W	Reset	1
	6	Pointer Pattern Address Bit-12	R/W	Reset	1
	5	Pointer Pattern Address Bit-11	R/W	Reset	1
	4	Pointer Pattern Address Bit-10	R/W	Reset	1
	3	Pointer Pattern Address Bit-9	R/W	Reset	1
	2	Pointer Pattern Address Bit-8	R/W	Reset	1
	1	Pointer Pattern Address Bit-7	R/W	Reset	1
lsb	0	Pointer Pattern Address Bit-6	R/W	Reset	1

This register contains the msbs of the address of the graphics pointer pattern in display memory. The format of the address into the 64Kx32 display memory are shown below:

Bit	Description
17	Extension Register FF bit-6
16	Extension Register FF bit-5
15	1
14	1
13	PPA bit-7
12	PPA bit-6
11	PPA bit-5
10	PPA bit-4
9	PPA bit-3
8	PPA bit-2
7	PPA bit-1
6	PPA bit-0
5	Mask (0 = 'and' mask, 1 = 'xor' mask)
4	Pattern line # bit 4 (msb)
3	Pattern line # bit 3
2	Pattern line # bit 2
1	Pattern line # bit 1
0	Pattern line # bit 0 (lsb)

The Pointer Pattern Address register allows the user to place the 32 x 32 pointer pattern on any 64-byte boundary in the display memory linear address range 0C000h through 0FFFFh. The V7VGA's pointer is a graphics image that is displayed in front of normal video data (essentially in a different plane), and which consequently does not interfere with bit map manipulation, removing the time-consuming necessity to carefully preserve the integrity of both the pointer and the bit map while drawing. The V7VGA's pointer also makes it possible to implement a non-flickering pointer.

The pointer pattern takes up 256 bytes (but since it spans all four planes, it occupies only 64 bytes of address space in each plane) and its location defaults to the last 64 bytes of display memory. The pointer pattern consists of a 128-byte AND mask followed by a 128-byte XOR mask.

The patterns each consist of 32 consecutive 32-bit values which represent the 32 successive lines of the

pointer pattern. These 32-bit patterns are stored in display memory across all 4 planes so that they can be fetched from display memory with two memory read operations (one for the AND mask and one for the XOR mask) during the two successive memory accesses immediately following refresh during the horizontal non-display enable interval prior to the scan line on which they are required. For each scan line, the 32 bits of AND mask data and the 32 bits of XOR mask data provide the pointer information of 32 pixels, as follows: Bit 7 of plane 0 is shifted out first (for the leftmost pixel of the pointer), followed by bit 6 of plane 0; bit 0 of plane 3 is shifted out last.

The insertion of the pointer pattern into the video data stream occurs at the very end of the video data path, after the attribute controller palette RAM and other multiplexings (but before the RAMDAC color palette since it is external to the V7VGA chip). Consequently, the pointer pattern modifies the 8 bits of video data that would normally (in the absence of the pointer) appear on output pins V7-V0. For each pixel, one bit of the pointer pattern AND mask is ANDed with each of the 8 video data bits, and then one bit of the pointer pattern XOR mask is XORed with each of the 8 bits produced by the AND operation, and the resultant 8 bits go to output pins V7-V0. This is the mechanism for producing video data for the 32 x 32 pixel area covered by the graphics pointer.

Correspondence between the above display memory address and the location of the pattern in the system memory space is determined by the state of the Graphics Controller registers, but typically the pointer is loaded in a mode in which display memory is mapped as four linear planes in the range A000:0000 through A000:FFFF, with the pointer pattern AND mask (in the default case) loaded at A000:FFC0 through A000:FFDF and the pointer pattern XOR mask loaded at A000:FFE0 through A000:FFFF. Once the pointer pattern is loaded, the configuration of display memory becomes irrelevant; the sequencer always uses the contents of the Pointer Pattern Address register to construct a linear display memory address in order to fetch pointer pattern data.

The upper left corner of the 32 x 32 pixel region of the screen at which the pointer appears is controlled by extension registers 9C through 9F.

If the pointer is positioned so that its right or bottom edge is off the screen, that part of the pointer is not seen. The V7VGA guarantees that such portions of the pointer are suppressed in hardware.

Note: If double-scanning is enabled, the pointer double scans right along with other video data. However, the sort of double-scanning used by mode 13h of the VGA, where the maximum scan line is set to 1, does not affect the cursor. Basically, the pointer pattern scan line advances every time the row scan address counter counts or turns over.

The Pointer Pattern Address register is set to 0FFh at power-up.

The Pointer Pattern Address register is only accessible when access to the Video Seven extension registers is enabled by writing 0EAh to SR6.

Index 9C

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	-unused-	-		
	4	-unused-	-		
	3	-unused-	-		
	2	Pointer Horizontal Position Bit-10	R/W		
	1	Pointer Horizontal Position Bit-9	R/W		
lsb	0	Pointer Horizontal Position Bit-8	R/W		

This register contains the upper 3 bits of the pointer horizontal position in pixels from the left edge of the display screen. A value of 0 in horizontal position bits 0-10 would position the left-most pixel of the pointer pattern over the left-most pixel of the display screen.

Index 9D

	Bit #	Description	Access	Reset By	Reset State
msb	7	Pointer Horizontal Position Bit-7	R/W		
	6	Pointer Horizontal Position Bit-6	R/W		
	5	Pointer Horizontal Position Bit-5	R/W		
	4	Pointer Horizontal Position Bit-4	R/W		
	3	Pointer Horizontal Position Bit-3	R/W		
	2	Pointer Horizontal Position Bit-2	R/W		
	1	Pointer Horizontal Position Bit-1	R/W		
lsb	0	Pointer Horizontal Position Bit-0	R/W		

This register contains the lower 8 bits of the pointer horizontal position in pixels from the left edge of the display screen. A value of 0 in horizontal position bits 0-10 would position the left-most pixel of the pointer pattern over the left-most pixel of the display screen.

Index 9E

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	-unused-	-		
	4	-unused-	-		
	3	-unused-	-		
	2	-unused-	-		
	1	Pointer Vertical Position Bit-9	R/W		
lsb	0	Pointer Vertical Position Bit-8	R/W		

This register contains the upper 2 bits of the pointer vertical position in scan lines from the upper edge of the display screen. A value of 0 in vertical position bits 0-9 would position the upper-most pixel of the pointer pattern over the upper-most pixel of the display screen.

Index 9F

	Bit #	Description	Access	Reset By	Reset State
msb	7	Pointer Vertical Position Bit-7	R/W		
	6	Pointer Vertical Position Bit-6	R/W		
	5	Pointer Vertical Position Bit-5	R/W		
	4	Pointer Vertical Position Bit-4	R/W		
	3	Pointer Vertical Position Bit-3	R/W		
	2	Pointer Vertical Position Bit-2	R/W		
	1	Pointer Vertical Position Bit-1	R/W		
lsb	0	Pointer Vertical Position Bit-0	R/W		

This register contains the lower 8 bits of the pointer vertical position in scan lines from the upper edge of the display screen. A value of 0 in vertical position bits 0-9 would position the upper-most pixel of the pointer pattern over the upper-most pixel of the display screen.

Index A0

	Bit #	Description	Access	Reset By	Reset State
msb	7	Graphics Controller Memory Latch 0 Bit-7	R/W		
	6	Graphics Controller Memory Latch 0 Bit-6	R/W		
	5	Graphics Controller Memory Latch 0 Bit-5	R/W		
	4	Graphics Controller Memory Latch 0 Bit-4	R/W		
	3	Graphics Controller Memory Latch 0 Bit-3	R/W		
	2	Graphics Controller Memory Latch 0 Bit-2	R/W		
	1	Graphics Controller Memory Latch 0 Bit-1	R/W		
lsb	0	Graphics Controller Memory Latch 0 Bit-0	R/W		

This register is actually the memory data latch which gets loaded from plane 0 data whenever video memory is read by the CPU. This register exists in a standard EGA and VGA, it just can't be accessed directly as it can in the V7VGA.

Index A1

	Bit #	Description	Access	Reset By	Reset State
msb	7	Graphics Controller Memory Latch 1 Bit-7	R/W		
	6	Graphics Controller Memory Latch 1 Bit-6	R/W		
	5	Graphics Controller Memory Latch 1 Bit-5	R/W		
	4	Graphics Controller Memory Latch 1 Bit-4	R/W		
	3	Graphics Controller Memory Latch 1 Bit-3	R/W		
	2	Graphics Controller Memory Latch 1 Bit-2	R/W		
	1	Graphics Controller Memory Latch 1 Bit-1	R/W		
lsb	0	Graphics Controller Memory Latch 1 Bit-0	R/W		

This register is actually the memory data latch which gets loaded from plane 1 data whenever video memory is read by the CPU. This register exists in a standard EGA and VGA, it just can't be accessed directly as it can in the V7VGA.

Index A2

	Bit #	Description	Access	Reset By	Reset State
msb	7	Graphics Controller Memory Latch 2 Bit-7	R/W		
	6	Graphics Controller Memory Latch 2 Bit-6	R/W		
	5	Graphics Controller Memory Latch 2 Bit-5	R/W		
	4	Graphics Controller Memory Latch 2 Bit-4	R/W		
	3	Graphics Controller Memory Latch 2 Bit-3	R/W		
	2	Graphics Controller Memory Latch 2 Bit-2	R/W		
	1	Graphics Controller Memory Latch 2 Bit-1	R/W		
lsb	0	Graphics Controller Memory Latch 2 Bit-0	R/W		

This register is actually the memory data latch which gets loaded from plane 2 data whenever video memory is read by the CPU. This register exists in a standard EGA and VGA, it just can't be accessed directly as it can in the V7VGA.

Index A3

	Bit #	Description	Access	Reset By	Reset State
msb	7	Graphics Controller Memory Latch 3 Bit-7	R/W		
	6	Graphics Controller Memory Latch 3 Bit-6	R/W		
	5	Graphics Controller Memory Latch 3 Bit-5	R/W		
	4	Graphics Controller Memory Latch 3 Bit-4	R/W		
	3	Graphics Controller Memory Latch 3 Bit-3	R/W		
	2	Graphics Controller Memory Latch 3 Bit-2	R/W		
	1	Graphics Controller Memory Latch 3 Bit-1	R/W		
lsb	0	Graphics Controller Memory Latch 3 Bit-0	R/W		

This register is actually the memory data latch which gets loaded from plane 3 data whenever video memory is read by the CPU. This register exists in a standard EGA and VGA, it just can't be accessed directly as it can in the V7VGA.

Index A4

NOTE: Only change this register with Sequencer Synchronous Reset Active

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	-unused-	-		
	4	Clock Select Bit-2	R/W	Reset	0
	3	-unused-	-		
	2	-unused-	-		
	1	-unused-	-		
lsb	0	-unused-	-		

Bit Descriptions

Bit 7-5 Unused

Bit 4 When this bit is set to 0, the MISC Output Register clock select bits (bits 2 and 3) are used to select clock sources as defined in the VGA. When this bit is set to 1, four additional clock sources can be selected (typically, the additional clock sources are set up for extended high resolution modes).

Bit 3-0 Unused

This register may be used to select between two groups of four clock sources. The two clock select bits in the Misc Output Register (bits 2 and 3) are used to select one of the four clock sources from each group.

CLK Bit-4	MISC Bit-3	MISC Bit-2	Clock Source	Comment
0	0	0	25.175 MHz	VGA Standard
0	0	1	16.257 MHz	VGA Standard
0	1	0	Feature Connector	VGA Standard
0	1	1	00.000 MHz	VGA Standard
1	0	0	50.350 MHz	V7VGA Extension
1	0	1	65.000 MHz	V7VGA Extension
1	1	0	Feature Connector	VGA Standard
1	1	1	40.000 MHz	V7VGA Extension

Index A5

	Bit #	Description	Access	Reset By	Reset State
msb	7	Pointer Enable	R/W	Reset	0
	6	-unused-	-		
	5	-unused-	-		
	4	-unused-	-		
	3	Cursor Mode	R/W	Reset	0
	2	-unused-	-		
	1	-unused-	-		
lsb	0	Cursor Blink Disable	R/W	Reset	0

This register is used to enable and disable the mouse pointer.

Bit Descriptions

Bit 7 Pointer Enable - Setting this bit to 1 enables the pointer logic to display the 32x32-pixel hardware mouse pointer on the screen at a location determined by the Pointer Horizontal and Vertical Position Registers (PXH/PXL and PYH/PYL). Setting this bit to 0 (the default state) disables this feature. The screen mask pattern for the pointer is fetched from display memory at absolute location Ann00 or Bnn00 for 128 bytes where nn is the contents of the Pointer Position Address Register (PPA at extensions index 94). The screen mask is ANDed with video memory data at the output of the color palette. The pointer mask pattern for the pointer is fetched from display memory at absolute location Ann80 for 128 bytes. The pointer mask is XORed with the results of the previous screen mask AND operation to produce the final video output. This results in the following truth table for the mask data:

ScreenMask	PointerMask	Resulting Screen Pixel
0	0	Black
0	1	White
1	0	Same as original pixel (pointer transparent)
1	1	Inverse of original pixel

Bits 6-4 Unused (read back as 0)

Bit 3 Cursor Mode - When this bit is 0, the text cursor replaces whatever pixels it is over. When this bit is 1, the text cursor is XORed with whatever pixels it is over.

Bits 2-1 Unused (read back as 0)

Bit 0 Cursor Blink Disable - When this bit is 0, the text cursor blinks normally, as described under discussions of CR0A, CR0B, CR0E, CR0F, and AR10 bit-3. When this bit is 1, the text cursor is always on and does not blink.

Index EA

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	W		
	6	-unused-	W		
	5	-unused-	W		
	4	-unused-	W		
	3	-unused-	W		
	2	-unused-	W		
	1	-unused-	W		
lsb	0	-unused-	W		

This register has no read/write bits implemented, but is instead used to strobe the state of on-board switches into the extensions switch readback register (extensions index F7). If this register is written with any data value, the state of data bus bits 15-8 are written into this register. Data bus bits 15-8 are typically connected to on-board switches; during the write to SWSTB, the bus data buffer for bits 15-8 are disabled so that the state of the switches determines the state of the data pins connected to the V7VGA chip.

The Switch Strobe register is nothing more than an I/O decode, and is decoded only on I/O writes; the value written to the Switch Strobe register is ignored.

This register may be written at any time, but is usually done only once at BIOS initialization time to determine the initial state of various power-up configuration options.

The Switch Strobe register is only accessible when access to the Video Seven extensions registers is enabled by writing 0EAh to SR6.

Index EB

	Bit #	Description	Access	Reset By	Reset State
msb	7	Emulation Enable	R/W	Reset	0
	6	Hercules Bit Map Enable	R/W	Reset	0
	5	Write Protect Range 2 (CR00-CR08)	R/W	Reset	0
	4	Write Protect Range 1 (CR09-CR0B)	R/W	Reset	0
	3	Write Protect Range 0 (CR0C)	R/W	Reset	0
	2	NMI Enable Range 2 (CR00-CR08)	R/W	Reset	0
	1	NMI Enable Range 1 (CR09-CR0B)	R/W	Reset	0
lsb	0	NMI Enable Range 0 (CR0C)	R/W	Reset	0

The Emulation Control register provides backward-compatibility related controls. The compatibility circuitry is designed to work with the Video Seven IOU chip.

The Emulation Control register is only accessible when access to the Video Seven extensions registers is enabled by writing 0EAh to SR6.

Bit Descriptions

Bit 7 Emulation Enable - When this bit is 1, a strobe to the IOU is generated on the TRAP* pin whenever certain CGA/Hercules registers are written to. When bit 0 of 3C2h is 1 (CGA emulation), a strobe is generated on OUTs to 3D8h (color mode), 3D9h (color palette), and 3DEh (AT&T 6300); also, when any of bits 2-0 of this register are 1, then a strobe is generated on OUTs to 3D4h (color CRTC index). When bit 0 of 3C2h is 1, a strobe is generated on OUTs to 3B8h (monochrome mode) and 3BFh (Hercules configuration); also, when any of bits 2-0 of this register are 1, then a strobe is generated on OUTs to 3B4h (monochrome CRTC index).

When the Emulation Enable bit is 0, no strobes to the IOU are generated on accesses to 3B8h, 3BFh, 3D8h, 3D9h, and 3DEh.

Note: The Emulation Enable bit has no effect on the bit-map, write protects, and NMI enables provided by bits 6-0 of this register. IOU emulation is disabled by writing 0 to this register.

Bit 6 Hercules Bit Map Enable - When this bit is 1, the V7VGA is forced to decode a memory mapping at 0B0000h for 64K, regardless of the setting of bits 3 & 2 of GR6. When this bit is 0, bits 3 & 2 of GR6 determine the V7VGA's memory mapping.

Bit 5 Write Protect Range 2 - When this bit is 1, registers CR00-CR08 are write protected; that is, writes to those registers do not modify the contents of the registers. When this bit is 0, registers CR00-CR08 are writable or not according to the state of bit 7 of CR11; refer to the discussion of that bit for details.

Note: If both the Write Protect Range 2 bit and bit 7 of CRTC11 are 1, then registers CR00-CR08 are write protected.

Bit 4 Write Protect Range 1 - When this bit is 1, registers CR09-CR0B are write protected; that is, writes to those registers do not modify the contents of the registers. When this bit is 0, registers CR09-CR0B are writable.

Bit 3 Write Protect Range 0 - When this bit is 1, register CR0C is write protected; that is, writes to that register do not modify the contents of the register. When this bit is 0, register CR0C is writable.

(continued)

Bit 2 NMI Enable Range 2 - When this bit is 1, a strobe intended for the IOU's external event pin is generated on the TRAP* pin on writes to any of registers CR00-CR08. When this bit is 0, no strobe is generated on access to any of registers CR00-CR08.

Bit 1 NMI Enable Range 1 - When this bit is 1, a strobe intended for the IOU's external event pin is generated on the TRAP* pin on writes to any of registers CR09-CR0B. When this bit is 0, no strobe is generated on access to any of registers CR09-CR0B.

Bit 0 NMI Enable Range 0 - When this bit is 1, a strobe intended for the IOU's external event pin is generated on the TRAP* pin on writes to register CR0C. When this bit is 0, no strobe is generated on access to register CR0C.

Index EC-EF

	Bit #	Description	Access	Reset By	Reset State
msb	7	Foreground Latch Bit-7	R/W		
	6	Foreground Latch Bit-6	R/W		
	5	Foreground Latch Bit-5	R/W		
	4	Foreground Latch Bit-4	R/W		
	3	Foreground Latch Bit-3	R/W		
	2	Foreground Latch Bit-2	R/W		
	1	Foreground Latch Bit-1	R/W		
lsb	0	Foreground Latch Bit-0	R/W		

The four Foreground Latch registers provide the CPU-side ALU input bytes for the four planes when bits 3 & 2 of extension register FE (Foreground / Background Control) are 1 & 0, respectively. Refer to the discussion of bits 3 & 2 of extension register FE for details.

Foreground Latch registers 0-3 are only accessible when access to the Video Seven extension registers is enabled by writing 0EAh to SR6.

Index F0

	Bit #	Description	Access	Reset By	Reset State
msb	7	Foreground Latch Bit-7	R/W		
	6	Foreground Latch Bit-6	R/W		
	5	Foreground Latch Bit-5	R/W		
	4	Foreground Latch Bit-4	R/W		
	3	Foreground Latch Bit-3	R/W		
	2	Foreground Latch Bit-2	R/W		
	1	Foreground Latch Bit-1	R/W		
lsb	0	Foreground Latch Bit-0	R/W		

The Fast Foreground Latch register provides a quick way to load the foreground latches for all four planes. Normally, the foreground latches must be loaded by performing four index/data OUTs to extension registers EC-EF. However, writes to the Fast Foreground Latch register are automatically routed to the foreground latch register for the plane selected by bits 5 & 4 of extension register F1 (the Fast Latch Load State Register), and the field composed of bits 5 & 4 of extension register F1 is automatically incremented modulo 4 after writes to the Fast Foreground Latch register. Moreover, bits 5 & 4 of extension register F1 are reset to 0 by a read from the Fast Foreground Latch register. This provides a very fast way, requiring only 1 IN, 4 OUTs, and no INCs and DECs, to load the foreground latch registers.

Note: There is no such physical register as the Fast Foreground Latch register; it is simply an I/O decode at which the foreground latch register pointed to by bits 5 & 4 of extension register F1 can be written to

The Fast Foreground Latch register is only accessible when access to the Video Seven extensions registers is enabled by writing 0EAh to SR6.

Index F1

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	Foreground Latch Load State Bit-1	R/W		
	4	Foreground Latch Load State Bit-0	R/W		
	3	-unused-	-		
	2	-unused-	-		
	1	Background Latch Load State Bit-1	R/W		
lsb	0	Background Latch Load State Bit-0	R/W		

The Fast Latch Load State register selects the foreground and background latch registers written to on I/O writes to the Fast Foreground Latch Load and Fast Background Latch Load registers.

The Fast Latch Load State register is only accessible when access to the Video Seven extensions registers is enabled by writing 0EAh to SR6.

Bit Descriptions

Bits 7-6 Unused

Bits 5-4 Foreground Latch Load State - These bits select the foreground latch register written to on I/O writes to extension register F0, as follows:

Bit-5	Bit-4	Foreground Latch Loaded
0	0	Plane 0 Foreground Latch (extension register EC)
0	1	Plane 1 Foreground Latch (extension register ED)
1	0	Plane 2 Foreground Latch (extension register EE)
1	1	Plane 3 Foreground Latch (extension register EF)

The Foreground Latch Load State bits are automatically incremented modulo 4 after I/O writes to extension register F0.

The Foreground Latch Load State bits are both reset to 0 by I/O reads from extension register F0.

Bits 3-2 Unused

Bits 1-0 Background Latch Load State - These bits select the background latch register written to on I/O writes to extension register F2, as follows:

Bit-1	Bit-0	Background Latch Loaded
0	0	Plane 0 Background Latch (extension register A0)
0	1	Plane 1 Background Latch (extension register A1)
1	0	Plane 2 Background Latch (extension register A2)
1	1	Plane 3 Background Latch (extension register A3)

The Background Latch Load State bits are automatically incremented modulo 4 after I/O writes to extension register F2.

The Background Latch Load State bits are both reset to 0 by I/O reads from extension register F2.

Index F2

	Bit #	Description	Access	Reset By	Reset State
msb	7	Background Latch Bit-7	R/W		
	6	Background Latch Bit-6	R/W		
	5	Background Latch Bit-5	R/W		
	4	Background Latch Bit-4	R/W		
	3	Background Latch Bit-3	R/W		
	2	Background Latch Bit-2	R/W		
	1	Background Latch Bit-1	R/W		
lsb	0	Background Latch Bit-0	R/W		

The Fast Background Latch register provides a quick way to load the background (Graphics Controller) latches for all four planes. Normally, the background latches must be loaded by performing four index/data OUTs to extension registers A0-A3. However, writes to the Fast Background Latch register are automatically routed to the background latch register for the plane selected by bits 1 & 0 of extension register F1 (Fast Latch Load State), and the field composed of bits 1 & 0 of extension register F1 is automatically incremented modulo 4 after writes to the Fast Background Latch register. Moreover, bits 1 & 0 of extension register F1 are reset to 0 by a read from the Fast Background Latch register. This provides a very fast way, requiring only 1 IN, 4 OUTs, and no INCs and DECs, to load the background latch registers.

Note: There is no such physical register as the Fast Background Latch register; it is simply an I/O decode at which the background latch register pointed to by bits 1 & 0 of extension register F1 can be written to.

The Fast Background Latch register is only accessible when access to the Video Seven extensions registers is enabled by writing 0EAh to SR6.

Index F3

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	-unused-	-		
	4	-unused-	-		
	3	-unused-	-		
	2	-unused-	-		
	1	Masked Write Source	R/W		
lsb	0	Masked Write Enable	R/W	Reset	0

The Masked Write Control register enables masked writes to V-RAMs and controls the source of the mask on masked writes

The Masked Write Control register is only accessible when access to the Video Seven extensions registers is enabled by writing 0EAh to SR6.

Bit Descriptions

Bits 7-2 Unused (read back as 0)

Bit 1 Masked Write Source - When masked writes to V-RAMs are enabled by setting bit 0 of this register to 1, then the source of the byte mask applied to the byte written by each plane is selected by the Masked Write Source bit. When the Masked Write Source bit is 1, then the rotated CPU bit provides the masked write mask. When the Masked Write Source bit is 0, then extension register F4 provides the masked write mask.

Bit 0 Masked Write Enable - The V7VGA supports V-RAM operation and V-RAMs support masked writes (write-per-bit writes), whereby selected bits of a memory byte can be modified without first reading the byte. When the Masked Write Enable bit is 1, masked write operation is selected. The source of the masked write mask is determined by bit 1 of this register: When bit 1 is 1, the rotated CPU byte is the mask for each plane; when bit 1 is 0, the contents of extension register F4 (MWMASK) is the mask for each plane. The 8-bit mask is applied equally to each of the four bytes going to the four planes.

When the Masked Write Enable bit is 0, masked write operation is disabled.

Masked write mode will only work when the V7VGA is driving V-RAMs. It will not work when the V7VGA is driving D-RAMs.

Index F4

	Bit #	Description	Access	Reset By	Reset State
msb	7	Masked Write Mask Bit-7	R/W		
	6	Masked Write Mask Bit-6	R/W		
	5	Masked Write Mask Bit-5	R/W		
	4	Masked Write Mask Bit-4	R/W		
	3	Masked Write Mask Bit-3	R/W		
	2	Masked Write Mask Bit-2	R/W		
	1	Masked Write Mask Bit-1	R/W		
lsb	0	Masked Write Mask Bit-0	R/W		

The Masked Write Mask register provides the byte used to mask off bits in each plane during masked writes to V-RAM when masked write mode is enabled. (Masked writes are enabled when bit 0 of extension register F3 is 1.) When masked writes are enabled, for each bit of either the Masked Write register (if bit 1 of extension register F3 is 0) or the rotated CPU byte (if bit 1 of extension register F3 is 1) that is 0, the corresponding bit in each byte of display memory written to remains unchanged; for each bit of either the Masked Write register or the rotated CPU byte that is 1, the corresponding bit in display memory bytes is replaced with the data written by the V7VGA.

The Masked Write Mask register is only accessible when access to the Video Seven extensions registers is enabled by writing 0EAh to SR6.

Index F5

	Bit #	Description	Access	Reset By	Reset State
msb	7	Foreground / Background Pattern Bit-7	R/W		
	6	Foreground / Background Pattern Bit-6	R/W		
	5	Foreground / Background Pattern Bit-5	R/W		
	4	Foreground / Background Pattern Bit-4	R/W		
	3	Foreground / Background Pattern Bit-3	R/W		
	2	Foreground / Background Pattern Bit-2	R/W		
	1	Foreground / Background Pattern Bit-1	R/W		
lsb	0	Foreground / Background Pattern Bit-0	R/W		

The Foreground/Background Pattern register is one possible source of the basic 8-bit pattern used in solid foreground/background mode, which is active when bits 3 & 2 of extension register FE (Foreground / Background Control) are 0 & 1, respectively. See the description of bits 3 & 2 of extension register FE for details.

The Foreground/Background Pattern register is only accessible when access to the Video Seven extensions registers is enabled by writing 0EAh to SR6.

Index F6

	Bit #	Description	Access	Reset By	Reset State
msb	7	Line Compare Bank Reset	R/W	Reset	0
	6	Counter Bank Enable	R/W	Reset	0
	5	CRTC Read Bank Select 1	R/W	Reset	0
	4	CRTC Read Bank Select 0	R/W	Reset	0
	3	CPU Read Bank Select 1	R/W	Reset	0
	2	CPU Read Bank Select 0	R/W	Reset	0
	1	CPU Write Bank Select 1	R/W	Reset	0
lsb	0	CPU Write Bank Select 0	R/W	Reset	0

This register is used to control bank selection when using 1 Mb ram chips. It allows independent selection of the one of four 256 Kb banks in 1 Mb D-RAMs in which CPU reads, CPU writes, and CRTC reads take place. This register also allows use of bit-maps spanning banks in 1 Mb D-RAMs.

The contents of the 1 Mb D-RAM Bank Select register are automatically used when 1 Mb D-RAMs are attached.

When four banks of 256Kb each are attached and the 256K Bank Enable bit (bit 4 of extension register FF) is 1, then bank controls are provided in accordance with the settings of the bits of the 1Mb D-RAM Bank Select register. This supports up to four virtual VGAs or high-resolution modes that require more than 256K bytes. This mode of operation works with either D-RAMs or V-RAMs. Consequently, the name of this register is somewhat misleading.

The 1 Mb D-RAM Bank Select register is only accessible when access to the Video Seven extensions registers is enabled by writing 0EAh to SR6.

Bit Descriptions

Bit 7 Line Compare Bank Reset - When this bit is 1, then when the line compare condition becomes true, bits 17 & 16 of the memory address counter are reset to 0. When the Line Compare Bank Load bit is 0, then when the line compare condition becomes true, bits 17 & 16 of the memory address counter are loaded from bits 5 & 4 of extension register F6. This allows line compare to either reset to the beginning of bank 0 of 1Mb DRAMs or to the start of the current bank.

Bit 6 Counter Bank Enable - When this bit is 1, address bits 17 & 16 presented to 1Mb D-RAMs or used to generate the one of four bank select are the output of bits 17 & 16 of the memory address counter. When this bit is 0, address bits 17 & 16 presented to 1Mb D-RAMs or used to generate the bank select are the output of bits 5 & 4 of extension register F6. This allows display memory scanning to either cross bank boundaries or wrap back to the start of the current bank.

Note: Font fetches are an exception to this. Address bits 17 & 16 always come from bits 5 & 4 of extension register F6 for font fetches. This is necessary since fonts can't stretch across bank boundaries like bit-maps can. You should generally set the Counter Bank Enable bit to 0 in text mode, so that font and character/attribute pairs both come from the same bank.

Bits 5-4 CRTC Read Bank Select - These bits select one of four 256Kb banks from which the CRTC fetches video data when fetching from display memory. Refer to the discussion of bits 7 & 6 of this register for additional information about CRTC scanning of 1Mb DRAMs and 256Kb banks.

Bits 3-2 CPU Read Bank Select - These bits select one of four 256Kb banks to which the CPU writes when writing to display memory.

Bits 1-0 CPU Write Bank Select - These bits select one of four 256Kb banks to which the CPU writes when writing to display memory.

Index F7

	Bit #	Description	Access	Reset By	Reset State
msb	7	Switch Readback Register Bit-7	R/W		
	6	Switch Readback Register Bit-6	R/W		
	5	Switch Readback Register Bit-5	R/W		
	4	Switch Readback Register Bit-4	R/W		
	3	Switch Readback Register Bit-3	R/W		
	2	Switch Readback Register Bit-2	R/W		
	1	Switch Readback Register Bit-1	R/W		
lsb	0	Switch Readback Register Bit-0	R/W		

The Switch Readback register can be used to determine the state of up to 8 switches connected to the CPU data line bits 15-8, as follows. Whenever a byte (8-bit) OUT is performed to the Switch Strobe register (extension register EA), the V7VGA turns off the buffer controlling CPU data lines 15-8, allowing switches attached to those lines through pull-down resistors to determine the state of the lines. The V7VGA then loads the state of those 8 lines into the Switch Readback register, rather than the CPU byte. This value can then be read normally from the Switch Readback register at any time. The Switch Readback register can be written at any time. The Switch Readback register is also directly writable at extensions index F7.

The Switch Readback register is only accessible when access to the Video Seven extensions registers is enabled by writing 0EAh to SR6.

Index F8

	Bit #	Description	Access	Reset By	Reset State
msb	7	Extended Clock Output 2	R/W		
	6	Extended Clock Output 1	R/W		
	5	Extended Clock Output 0	R/W		
	4	Clock 3 On	R/W	Reset	0
	3	External Clock Override	R/W	Reset	0
	2	Extended Clock Output Source	R/W		
	1	Extended Clock Direction	R/W	Reset	0
lsb	0	Clock 0 Only	R/W	Reset	0

The Extended Clock Control register controls the multiplexing of clock inputs and the direction of the three high-order extended clock pins and the data that can be placed on them.

Note: Synchronous reset must be in effect when the current clock is changed by altering any of the bits in this register, or display memory may be randomly altered.

This register powers up with bits 4, 3, 1, and 0 set to 0.

The Extended Clock Control register is only accessible when access to the Video Seven extensions registers is enabled by writing 0EAh to SR6.

Bit Descriptions

Bits 7-5 Extended Clock Output - When the Extended Clock Direction bit (bit 1 of this register) is 1, the XD2M, XD1M, and XD0M pins become outputs rather than inputs. In this case, the Extended Clock Output Source bit (bit 2 of this register) selects the data to be placed on those pins. When the Extended Clock Output Source bit is 1, then the Extended Clock Output bits are placed on the XD2M, XD1M, and XD0M pins, respectively. When the Extended Clock Output Source bit is 0, then clock select bits 2-0 (bit 4 of extension register A4 and bits 3 & 2 of 3C2h, respectively) are placed on the XD2M, XD1M, and XD0M pins, respectively, and the Extended Clock Output bits are ignored.

When the Extended Clock Direction bit is 0, the Extended Clock Output bits have no effect.

Bit 4 Clock 3 On - When the Clock 3 On bit is 0, the clock selected when bits 3 & 2 of the Miscellaneous Output register (3C2h) are both 1 (clock 3) is forced to be grounded, for IBM VGA compatibility, regardless of the mode of clock operation selected by the Extended Clock Select register and bits 2-0 of the Extended Clock Control register.

When the Clock 3 On bit is 1 and bits 3 & 2 of the Miscellaneous Output register are both 1, then whatever clock input is selected according to the Extended Clock Select register and bits 2-0 of the Extended Clock Control register is active.

Bit 3 External Clock Override - When this bit is 0, then whenever bits 3 & 2 of the Miscellaneous Output register (3C2h) are 1 & 0, respectively, the External Clock input is selected regardless of the mode of clock operation selected by the Extended Clock Select register and bits 2-0 of the Extended Clock Control register.

When this bit is 1 and bits 3 & 2 of the Miscellaneous Output register are 1 & 0, respectively, then whatever clock input is selected according to the Extended Clock Select register and bits 2-0 of the Extended Clock Control register is active.

Bit 2 Extended Clock Output Source - When the Extended Clock Direction bit (bit 1 of this register) is 1, the XD2M, XD1M, and XD0M pins become outputs rather than inputs. In this case, this bit selects the data to be placed on those pins. When this bit is 1, then the Extended Clock Output bits (bits 7-5 of this register) are placed on the XD2M, XD1M, and XD0M pins, respectively. When this bit is 0, then clock select bits 2-0 (bit 4 of extension register A4 and bits 3 & 2 of 3C2h, respectively) are placed on the XD2M, XD1M, and XD0M pins, respectively.

When the Extended Clock Direction bit is 0, this bit has no effect.

Bit 1 Extended Clock Direction - When this bit is 1, the XD2M, XD1M, and XD0M pins become outputs. In this case, the Extended Clock Output Source bit selects the data to be placed on those pins. When the Extended Clock Output Source bit (bit 2 of this register) is 1, then the Extended Clock Output bits (bits 7-5 of this register) are placed on the XD2M, XD1M, and XD0M pins, respectively. When the Extended Clock Output Source bit is 0, then clock select bits 2-0 (bit 4 of extension register A4 and bits 3 & 2 of 3C2h, respectively) are placed on the XD2M, XD1M, and XD0M pins, respectively.

Also, when this bit is 1, the high bit of the 3-bit clock select field to the circuitry which selects the clock driving the V7VGA (which normally comes from bit 4 of extension register A4) is forced to 0, limiting selection among the clock input pins to the lower four clock inputs, X25M, X28M, FCLK, and XRESM. This makes sense because when the Extended Clock Direction bit is 1, the upper clock pins (XD2M, XD1M, and XD0M) are outputs. However, all three clock select bits, explicitly including bit 4 of extension register A4, are placed on XD2M, XD1M, and XD0M when the Extended Clock Output Source bit is 0. A maximum of 11 clocks could theoretically be accessed by selecting 8 clocks from an external clock generator on X25M (by turning on the Clock 0 Only bit, bit 0 of this register), and three additional clocks on X28M, FCLK, and XRESM (with the Clock 0 Only bit off).

When this bit is 0, the XD2M, XD1M, and XD0M pins become inputs. These clock inputs can be selected as the current clock driving the V7VGA via clock select bits 2-0, as described in the discussion of extension register A4 and I/O port 3C2h.

Bit 0 Clock 0 Only - When the Clock 0 Only bit is 1, the current clock to the V7VGA is derived from the X25M pin (the pin normally selected when the clock select bits (bit 4 of extension register A4 and bits 3 & 2 of 3C2h) are 000b, or 0). This allows an external dot clock generator to be attached to X25M, with the clock select bits going out on XD2M, XD1M, and XD0M to program the external generator. To an application program, it will still appear as if several crystals are attached to the various clock pins as the program sets bits 3 & 2 of 3C2h.

However, the Clock 0 Only bit can be overridden by two conditions. First, if the Clock 3 On bit (bit 4 of this register) is 0, then when bits 3 & 2 of 3C2h are both 1, the clock input to the V7VGA is grounded (no clock). Second, if the External Clock Override bit (bit 3 of this register) is 0, then when bits 3 & 2 of 3C2h are 1 and 0, respectively, then the clock input to the V7VGA comes from the FCLK pin. These two overrides provide the potential of full IBM VGA compatibility with a single external clock generator attached to X25M and driven by the clock select bits via XD2M, XD1M, and XD0M.

When the Clock 0 Only bit is 0, then the clock select fields and bits 4, 3, and 1 of this register determine which clock input drives the V7VGA.

Index F9

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	-unused-	-		
	4	-unused-	-		
	3	-unused-	-		
	2	-unused-	-		
	1	-unused-	-		
lsb	0	Extended Page Select	R/W	Reset	0

The Extended Page Select register provides additional CPU addressing control in some extended 256-color modes.

The Extended Page Select register is only accessible when access to the Video Seven extensions registers is enabled by writing 0EAh to SR6.

Bit Descriptions

Bits 7-1 Unused (read back as 0)

Bit 0 Extended Page Select - When the Extended 256-Color Enable bit (bit 2 of extension register FC) is 1, the Chain 4 bit (bit 3 of SR4) is 1, and the Extended 256-Color Mode bit (bit 1 of extension register FC) is 0, then the Extended Page Select bit is placed on linear address bit 0 during CPU accesses. This bit is used to support 640x400 256-color modes with all display memory mapped into a 64K CPU address space.

If any of the above conditions are not true, the Extended Page Select bit has no effect.

Index FA

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	-unused-	-		
	4	-unused-	-		
	3	Foreground Color Bit-3	R/W		
	2	Foreground Color Bit-2	R/W		
	1	Foreground Color Bit-1	R/W		
lsb	0	Foreground Color Bit-0	R/W		

The Extended Foreground Color register provides the foreground color used in solid foreground / background mode, which is active when bits 3 & 2 of extension register FE (Foreground / Background Control) are 0 & 1, respectively. See the description of bits 3 & 2 of extension register FE for details.

When bits 3 & 2 of extension register FE are not 0 & 1, respectively, the Extended Foreground Color register has no effect.

The Extended Foreground Color register is only accessible when access to the Video Seven extensions registers is enabled by writing 0EAh to SR6.

Index FB

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	-unused-	-		
	4	-unused-	-		
	3	Background Color Bit-3	R/W		
	2	Background Color Bit-2	R/W		
	1	Background Color Bit-1	R/W		
lsb	0	Background Color Bit-0	R/W		

The Extended Background Color register provides the background color used in solid foreground / background mode, which is active when bits 3 & 2 of extension register FE (Foreground / Background Control) are 0 & 1, respectively. See the description of bits 3 & 2 of extension register FE for details.

When bits 3 & 2 of extension register FE are not 0 & 1, respectively, the Extended Background Color register has no effect.

The Extended Background Color register is only accessible when access to the Video Seven extensions registers is enabled by writing 0EAh to SR6.

Index FC

	Bit #	Description	Access	Reset By	Reset State
msb	7	Internal 3C3 Enable (ARMVSE)	R/W	Reset	0
	6	Extended Display Enable Skew	R/W	Reset	0
	5	Sequential Chain 4	R/W	Reset	0
	4	Sequential Chain	R/W	Reset	0
	3	Refresh Skew Control	R/W	Reset	0
	2	Extended 256-Color Enable	R/W	Reset	0
	1	Extended 256-Color Mode	R/W		
lsb	0	Extended Attribute Enable	R/W	Reset	0

The Compatibility Control register enables selection and aspects of operation of enhanced 256-color graphics modes, enhanced attributes in text mode, refresh and extended display enable skews, and masked V-RAM writes.

The Compatibility Control register is only accessible when access to the Video Seven extensions registers is enabled by writing 0EAh to SR6.

Bit Descriptions

Bit 7 Internal 3C3 Enable (ARMVSE) - When this bit (ARMVSE) is 1, the DISABLE pin is 1, and bit 0 of 3C3h is 1, the V7VGA is mapped into the host's I/O and memory address space. When ARMVSE 1, the DISABLE pin is 1, and bit 0 of 3C3h is 0, 3C3h is the only I/O address at which the V7VGA answers, and all V7VGA memory addressing (to the CPU) is disabled. In both these cases, the V7VGA answers at I/O address 3C3h.

When ARMVSE is 0 and the DISABLE pin is 1, then bit 0 of 3C3h has no effect and 3C3h is not decoded as a V7VGA register.

When the DISABLE pin is 0, all V7VGA I/O and memory addressing is disabled.

(continued)

Bit 6 Extended Display Enable Skew - When this bit is 1, the display enable skew is 1 greater than the skew selected via bits 6 & 5 of CR03. When this bit is 0, the display enable skew is whatever skew is selected by bits 6 & 5 of CR03.

Bit 5 Sequential Chain 4 - When the Sequential Chain 4 bit is 1, the nature of chain 4 mode (selected when bit 3 of SR4 is 1) changes. On CPU accesses, instead of routing CPU address bits A15-A02 to linear address bits LA15-LA02 and substituting certain bits on LA01 and LA00, when the Sequential Chain 4 bit is 1 A15-A02 are routed to LA13-LA00, and the same certain bits are substituted on LA15 and LA14 rather than LA01 and LA00. The net effect of this is to cause chain 4 bit-maps to be stored at consecutive display memory addresses, where they can be scanned in byte mode.

For a full description of substitution of bits in chain 4 and sequential chain 4 modes, refer to the discussion under bit 1 of GR6.

The reason for the existence of the Sequential Chain 4 bit is the need to support high-resolution doubleword modes, such as 256-color modes, with V-RAMs, which are not capable of working in word or doubleword mode with 1:2 or 1:4 interleave.

When the Sequential Chain 4 bit is 0, chain 4 mode operates as described under bit 3 of SR4 and bit 1 of GR6.

The Sequential Chain 4 bit has no effect when the Chain 4 bit (bit 3 of SR4) is 0.

Bit 4 Sequential Chain - When this bit is 1, the nature of chain mode (selected when bit 3 of SR4 is 0 and bit 1 of GR6 is 1) changes. On CPU accesses, instead of routing CPU address bits A15-A01 to linear address bits LA15-LA01 and substituting certain bits on LA00, when the Sequential Chain bit is 1 A15-A01 are routed to LA14-LA00, and the same certain bits are substituted on LA15 rather than LA00. The net effect of this is to cause chain bit-maps to be stored at consecutive display memory addresses, where they can be scanned in byte mode.

For a full description of substitution of bits in chain and sequential chain modes, refer to the discussion under bit 1 of GR6

The reason for the existence of the Sequential Chain bit is the desire to support high-resolution word modes with V-RAMs, which are not capable of working in word or doubleword mode with 1:2 or 1:4 interleave.

When the Sequential Chain 4 bit is 0, chain mode operates as described under bit 1 of GR6.

The Sequential Chain bit has no effect when the Chain 4 bit (bit 3 of SR4) is 1 or the Chain bit (bit 1 of GR6) is 0.

Bit 3 Refresh Skew Control - This bit controls whether the start of refresh occurs one character after the end of the skewed display enable signal or one character after the end of the unskewed display enable signal. When this bit is 1, refresh occurs one character after the end of the skewed display enable signal (the skew of the display enable signal is controlled by bits 6 & 5 of CR03). When this bit is 0, refresh starts one character after the end of the unskewed display enable signal.

The IBM VGA starts refresh immediately after display enable end, which causes problems in shift 4 and shift load modes, where display enable must be skewed, making it impossible to support a border in these modes. The Refresh Skew Control bit makes it possible for refresh to operate properly in modes that use display enable skew.

(continued)

Bit 2 Extended 256-Color Enable - This bit enables or disables the enhanced 256-color mode selected by bit 1 of this register. When this bit is 0, no enhanced 256-color mode is in effect: the standard IBM 256-color CPU address multiplexing is in effect whenever the Chain 4 bit (bit 3 of SR4) is set to 1, and the standard IBM 256-color CRTC address multiplexing is in effect whenever the Doubleword Mode bit (bit 6 of CR14) is 1.

When this bit is 1, the enhanced 256-color mode selected by bit 1 of this register is in effect for CPU addressing whenever the Chain 4 bit is 1, and enhanced CRTC 256-color mode (whereby memory address counter bit 15 is placed on linear address bit 1 and memory address counter bit 14 is placed on linear address bit 0) is in effect for CRTC addressing whenever the Doubleword Mode bit is 1.

Bit 1 Extended 256-Color Mode - This bit selects the type of extended 256-color CPU addressing mode that is in effect when the Extended 256-Color Enable bit (bit 2 of this register) is 1 and when the Chain 4 bit (bit 3 of SR4) is 1. The extended modes allow 256-color bit maps larger than 64K, supporting resolutions of 320x400 and 640x400, by contrast with the IBM VGA's 256-color mode, which allows only a 64K bitmap and hence a maximum resolution of 320x200.

When the Extended 256-Color Enable bit is 1, this bit selects extended 256-color modes as follows:

0 = 64K extended 256-color mode
1 = 128K extended 256-color mode

When bit 1 is 0, 64K Extended 256-Color Mode is in effect. In this mode, if the Chain 4 bit is 1, then during CPU reads and writes, CPU address bits 15 through 2 are placed on linear address bits 15 through 2, the non-inverted Page Select bit (bit 5 of the Miscellaneous Output register (3C2h)) is placed on linear address bit 1, and the Extended Page Select bit (bit 3 of this register) is placed on linear address bit 0. This mode supports 256-color modes with resolutions up to 640x400 mapped into a 64K address space.

When bit 1 is 1, 128K Extended 256-Color Mode is in effect. In this mode, if the Chain 4 bit is 1, for CPU addressing, CPU address bits 15 through 2 are placed on linear address bits 15 through 2, the non-inverted Page Select bit is placed on linear address bit 1, and CPU address bit 16 is placed on linear address bit 0. This mode supports 256-color modes with resolutions up to 640x400 mapped into a 128K address space.

Whenever the Chain 4 bit is 0, the selected extended 256-color mode has no effect on CPU addressing. The net effect of the extended 256-color modes is to substitute new CPU and CRTC address multiplexings in place of IBM's default (and only) 256-color multiplexings, which are normally selected by setting the Chain 4 and Doubleword bits to 1.

The Extended 256-Color Mode bit has no effect when the Extended 256 Color Enable bit is 0.

Bit 0 Extended Attribute Enable - This bit enables or disables extended text attribute operation. When this bit is 1, extended text attributes are enabled. In this case, the extended attribute byte for each character is fetched from plane 3 at the same time and from the same address as the character code and attribute byte. This byte is ORed with the font data on the underline scan line, in the same way as the underline is inserted. This provides a means of underlining text with any and all normal attributes.

When this bit is 0, extended text attributes are disabled, and the extended attribute byte has no effect. This is the IBM VGA compatible mode of operation.

Index FD

	Bit #	Description	Access	Reset By	Reset State
msb	7	Graphics 8-Dot Timing State Select Bit-3	R/W	Reset	0
	6	Graphics 8-Dot Timing State Select Bit-2	R/W	Reset	0
	5	Graphics 8-Dot Timing State Select Bit-1	R/W	Reset	0
	4	Graphics 8-Dot Timing State Select Bit-0	R/W	Reset	0
	3	Text 8-Dot Timing State Select Bit-3	R/W	Reset	0
	2	Text 8-Dot Timing State Select Bit-2	R/W	Reset	0
	1	Text 8-Dot Timing State Select Bit-1	R/W	Reset	0
lsb	0	Text 8-Dot Timing State Select Bit-0	R/W	Reset	0

The Extended Timing Select register controls the operative timing mode in both text and graphics 8-dot sequencer modes of operation.

Note: In 9-dot mode (SR1 bit-0 = 0), this register is ignored and a 1:4 timing state is selected. This timing state is the only one supported in text and graphics 9-dot modes. This state supports maximum dot clocks of 33 MHz with 120 ns RAMs and 40 MHz with 100 ns RAMs.

Note: Synchronous reset must be in effect when either of the fields in this register is changed when that field is providing the current timing state selection, or display memory may be randomly altered.

The Extended Timing Select register is only accessible when access to the Video Seven extensions registers is enabled by writing 0EAh to SR6.

In the tables on the following page, **N:m** interleave refers to **n** CPU accesses allowed per **m** character clocks.

(continued)

Bit Descriptions

Bits 7-4 Graphics 8-Dot Timing State Select - The 8-Dot Graphics Timing Select bits select the timing state used in 8-dot graphics mode. 8-dot graphics mode is selected when bit 0 of SR1 is 1 and bit 0 of GR6 is 1. Timing state selection in 8-dot graphics mode operates as follows.

				------ Max Dotclock (MHz) ------			
Bits 7-4	Ram Type	Timing State	Interleave	80ns	100ns	120ns	150ns
0	**D-Ram**	**8-dot Graphics**	**1:4**	**45**	**35**	**29**	**20**
1	D-Ram	invalid	none	n/a	n/a	n/a	n/a
2	**D-Ram**	**8-dot Graphics**	**1:2**	**38**	**30**	**25**	**18**
3	**D-Ram**	**8-dot Graphics**	**Paged 1:4**	**65**	**50**	**41**	**29**
4	D-Ram	invalid	none	n/a	n/a	n/a	n/a
5	D-Ram	invalid	none	n/a	n/a	n/a	n/a
6	D-Ram	invalid	none	n/a	n/a	n/a	n/a
7	D-Ram	invalid	none	n/a	n/a	n/a	n/a
8	**V-Ram**	**8-dot Graphics**	**1:1**	n/a	n/a	**33**	**25**
9	**V-Ram**	**8-dot Graphics**	**1:2**	n/a	n/a	**50**	**40**
10	**V-Ram**	**8-dot Graphics**	**1:4**	n/a	n/a	**80**	**65**
11	V-Ram	invalid	none	n/a	n/a	n/a	n/a
12	V-Ram	invalid	none	n/a	n/a	n/a	n/a
13	V-Ram	invalid	none	n/a	n/a	n/a	n/a
14	V-Ram	invalid	none	n/a	n/a	n/a	n/a
15	V-Ram	invalid	none	n/a	n/a	n/a	n/a

Note: In paged 1:4 interleave, 4 paged accesses by the CRTC are performed per CPU access. This causes panning and virtual screen sizes to be incompletely supported, since the start address can vary only by multiples of 4 and the Offset register can vary only by multiples of 2.

Note: The 8-dot 1:4 V-Ram and 8-dot Paged 1:4 D-Ram graphics timing states do not fully support byte panning (not all offset and start address settings work properly).

Bits 3-0 Text 8-Dot Timing State Select - The 8-Dot Text Timing Select bits select the timing state used in 8-dot text modes. 8-dot text mode is selected when bit 0 of SR1 is 1 and bit 0 of GR6 is 0. Timing state selection in 8-dot text mode operates as follows:

				------ Max Dotclock (MHz) ------			
Bits 3-0	Ram Type	Timing State	Interleave	80ns	100ns	120ns	150ns
0	**D-Ram**	**8-dot Text**	**1:4**	**45**	**35**	**29**	**20**
1	D-Ram	invalid	none	n/a	n/a	n/a	n/a
2	**D-Ram**	**8-dot Text**	**1:2**	**38**	**30**	**25**	**18**
3	D-Ram	invalid	none	n/a	n/a	n/a	n/a
4	D-Ram	invalid	none	n/a	n/a	n/a	n/a
5	D-Ram	invalid	none	n/a	n/a	n/a	n/a
6	D-Ram	invalid	none	n/a	n/a	n/a	n/a
7	D-Ram	invalid	none	n/a	n/a	n/a	n/a
8	V-Ram	invalid	none	n/a	n/a	n/a	n/a
9	V-Ram	invalid	none	n/a	n/a	n/a	n/a
10	**V-Ram**	**8-dot Text**	**1:4**	n/a	n/a	**40**	**33**
11	V-Ram	invalid	none	n/a	n/a	n/a	n/a
12	V-Ram	invalid	none	n/a	n/a	n/a	n/a
13	V-Ram	invalid	none	n/a	n/a	n/a	n/a
14	V-Ram	invalid	none	n/a	n/a	n/a	n/a
15	V-Ram	invalid	none	n/a	n/a	n/a	n/a

Note: The 8-dot 1:4 V-RAM text timing state does not fully support byte panning (not all offset and start address settings work properly).

Index FE

	Bit #	Description	Access	Reset By	Reset State
msb	7	-unused-	-		
	6	-unused-	-		
	5	-unused-	-		
	4	-unused-	-		
	3	Foreground / Background Mode 1	R/W	Reset	0
	2	Foreground / Background Mode 0	R/W	Reset	0
	1	Foreground / Background Source	R/W		
lsb	0	-unused-	-		

The Foreground/Background Control register selects between three modes of foreground / background operation and use of the output from the set/reset circuitry, and the data source for one foreground / background mode.

The Foreground/Background Control register is only accessible when access to the Video Seven extensions registers is enabled by writing 0EAh to SR6.

Bit Descriptions

Bits 7-4 Unused (read back as 0)

Bits 3-2 Foreground / Background Mode - These bits select the source of the byte input to each of the four plane ALUs, as follows:

Bit-3	Bit-2	Mode of CPU-Side ALU Input Operation
0	0	Set/Reset Output Mode (IBM VGA)
0	1	Solid Foreground / Background Mode
1	0	Dithered Foreground Mode
1	1	Invalid

Set/reset output mode simply means that the output of the set/reset circuity is the CPU-side ALU input, just as in the IBM VGA.

Solid foreground/background mode means that a byte is input into the foreground/background select circuitry, with each 1-bit of the byte selecting the foreground color stored in extension register FA and each 0-bit selecting the background color stored in extension register FB. The resultant byte for each plane becomes the CPU-side input to each plane's ALU. This provides the capability to generate a solid foreground against a solid background with a single write, basically expanding a binary (monochrome) pattern to a color pattern. The source of the selection byte in this mode is the Foreground/Background Pattern register (extension register F5) if bit 1 of this register is 0, and is the rotated CPU byte if bit 1 of this register is 1.

Dithered foreground mode means that the foreground latch byte for each plane (from extension registers EC-EF) is input directly to that plane's CPU-side ALU input. This provides the capability to support fully dithered foreground patterns, with dithered background patterns optionally stored in the normal latches and the two combined via the bit mask.

Bit 1 Foreground / Background Source - When the V7VGA is in solid foreground/background mode (when bits 3 & 2 of this register are 0 & 1, respectively), then the 8-bit pattern that selects between the foreground color in extension register FA and the background color in extension register FB can come either from the Foreground/Background Pattern register (extension register F5) or from the rotated CPU byte. When this bit is 1, the rotated CPU byte is the source; when this bit is 0, the Foreground/Background Pattern reigster is the source.

Bit 0 Unused (reads back as 0)

Index FF

	Bit #	Description	Access	Reset By	Reset State
msb	7	16-Bit Bus Status	R		
	6	Pointer Bank Select 1	R/W		
	5	Pointer Bank Select 0	R/W		
	4	256K Bank Enable	R/W	Reset	0
	3	16-Bit ROM Interface Enable	R/W	Reset	0
	2	Fast Write Enable	R/W	Reset	0
	1	16-Bit I/O Interface Enable	R/W	Reset	0
lsb	0	16-Bit Memory Interface Enable	R/W	Reset	0

This register is used to control 16-bit options in the interface of the V7VGA chip and associated components to the system bus. Internally, the V7VGA is an 8-bit device (because the VGA is inherently an 8-bit device), but presenting a 16-bit interface to the system and then breaking the 16-bit accesses into two 8-bit accesses internally can save a significant number of wait states, at least on the AT bus, by eliminating wait states inserted while performing 8-bit bus emulation.

The 16-Bit Control register is only accessible when access to the Video Seven extensions registers is enabled by writing 0EAh to SR6.

Bit Descriptions

Bit 7 16-Bit Bus Status - This bit reads back 1 if the bus the V7VGA is installed in is a 16-bit bus, and reads back 0 if the bus the V7VGA is installed in is an 8-bit bus. This bit is read-only.

Bits 6-5 Pointer Bank Select - These bits provide linear address bits 17 & 16 used when addressing the pointer pattern. These bits go either to 1 Mb DRAMs or are the bits that select one of up to four banks of 256 Kb each, depending on the setting of bit 4 of this register.

Bit 4 256K Bank Enable - This bit enables operation of the V7VGA with up to four banks of 256K display memory attached. When the 256K Bank Enable bit is 1, RAS is generated on one of the four pins RAS0*, RAS1*, RAS2*, or RAS3* for the one of the four 256K banks selected by ERF6. (During refresh, all four lines are driven.) Also, additional controls for four banks of VRAMs are provided on the SOE0*, SOE1*, SOE2*, SOE3*, SCLK*, and DTOE* pins. 1 Mb DRAMs are not supported when the 256K Bank Enable bit is 1. When the 256K Bank Enable bit is 0, these above pins operate in their normal non-banked modes, and support for four banks of 256K each is not provided. However, four banks within 1 Mb DRAMs are supported (via the XRAD pin) when the 256K Bank Enable bit is 0.

Bit 3 16-Bit ROM Interface Enable - The V7VGA always accepts ROM decodes on the XRDN* pin and generates the appropriate buffer enables. When the 16-Bit ROM Interface Enable bit is 1, the V7VGA generates MEM16* when ROM decode occurs, and generates the appropriate buffer enables based on SHBE* and A0. When the 16-Bit ROM Interface Enable bit is 0, the V7VGA always generates an enable for the lower byte of the data bus (for an 8-bit bus).

(continued)

Bit 2 Fast Write Enable - When the Fast Write Enable bit is 1, the address and data are latched internally on CPU writes and the CPU is released immediately, with the display memory write being completed whenever a CPU access becomes available. If the CPU attempts to access display memory again before a pending write is completed, only then is the CPU waited. This provides effectively zero-wait-state latency for CPU writes.

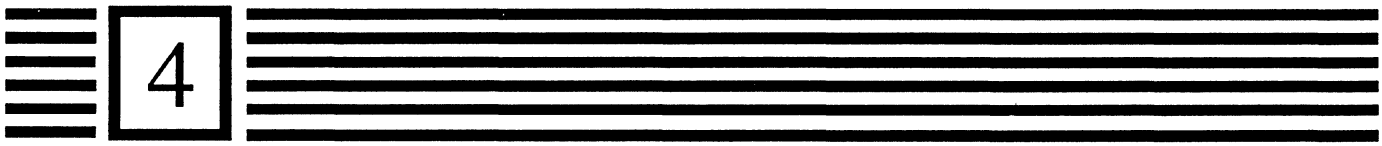
When the Fast Write Enable bit is 0, the CPU is always held until the current display memory write is completed.

Note: The Fast Write Enable bit has no effect on CPU reads.

Bit 1 16-Bit I/O Interface Enable - The 16-Bit I/O Interface Enable bit controls whether the I/O interface the V7VGA presents to the system bus is an 8- or 16-bit interface. When this bit is 1, the V7VGA provides a 16-bit I/O interface. When this bit is 0, the V7VGA provides an 8-bit I/O interface.

Bit 0 16-Bit Memory Interface Enable - The 16-Bit Memory Interface Enable bit controls whether the memory the the V7VGA presents to the system bus is an 8- or 16-bit interface. When this bit is 1, the V7VGA provides a 16-bit memory interface. When this bit is 0, the V7VGA provides an 8-bit memory interface.

CHAPTER



BIOS

4.1 BIOS Overview

The V7 VGA BIOS provides IBM VGA-compatible support for the V7 VGA chip. This includes display modes for increased color and resolution (see Chapter 2.5), predefined fonts for text and graphics modes, and power-on confidence tests to assure that the hardware is functioning properly.

4.2 BIOS Initialization and Power-up Diagnostics

The BIOS power-up sequence is as follows:

- Turn off the display
- Clear emulation state
- Set up interrupt vectors 10, 1F, 42, and 43
- Read switches and feature adapter status and store in ram
- Set video state per the switches
- Check for co-resident video adapters
- Perform video adapter tests
  - Perform read/write tests on all registers
  - Set 40-column text mode and check vertical and horizontal timing
  - Test video output using video status mux
- Perform display memory tests
  - Determine size of memory planes
  - Write all byte values 0-FF at start of each 16K block
  - Write AAAA, 5555, FFFF, and 0000 at every location

02H - Non-maskable Interrupt (NMI) (Location = 0:0008H)

The NMI vector is used by the BIOS to provide CGA and MGA register-level compatibility.

05H - Print Screen (Location = 0:0014H)

The Alternate Select BIOS function can set the print screen vector so that it points to a routine that handles non-standard rows and columns.

10H - Functions (Location = 0:0040H)

BIOS functions are accessed via this vector. Programs place a function code in AH and other calling parameters, if required, in other registers then execute an INT 10 instruction. When BIOS gains control, the appropriate code is executed to perform the function; return parameter values are left in processor registers on return to the calling program.

The functions supported by the V7 VGA BIOS allow the calling program to set the current mode, manipulate the cursor, place characters and individual pixels on the display screen, scroll the screen, load character fonts and color palette values, and read the light pen position. These functions are described in following chapters.

Functions 0-F are supported by the PC system BIOS. If an EGA/VGA board is present in the system, its BIOS takes over these functions from the system. Functions 10-FF are only available to programs if the V7 VGA board is present in the system. Of these additional functions, 10-13 are supported in the IBM EGA and VGA; functions numbered 14H and above are Vega specific. Function 6F is V7 VGA specific.

42H - Reserved (Location = 0:0108H)

When the EGA/VGA is installed, BIOS routines use INT 42 to re-vector the standard INT 10 video pointer. This is the original motherboard INT 10 vector.

43H - Graphics Character Table (Location = 0:010CH)

BIOS routines use this vector to point to a table of dot patterns that are used when graphics characters are displayed. This table is used for the first 128 characters in video modes 4, 5 and 6. This table is also used for 256 characters in all additional graphics modes (D, E, F, 10, 11, 12 and 13).

1D - 6845 Parameter Table (Location = 0:0074H)

This is used as a pointer to the 6845 CRT controller parameters as used by the CGA. This vector is used for emulation only.

1F - Upper 128 Characters (Location = 0:007CH)

This table is used for the upper 128 characters in modes 4, 5, and 6.

4.4 BIOS Standard Functions

V7 VGA BIOS-supported modes can be divided into two types, text and graphics. Some of the following functions only apply to one of these types, while others expect different parameters based on whether the current display mode is text or graphics.

V7 VGA BIOS functions are accessed using interrupt 10H. The function code is placed in register AH, and other information is placed in the corresponding registers as indicated. Table 4-1 summarizes the BIOS functions in the functional order in which they are presented in this chapter. Table 4-2 summarizes the BIOS functions in numerical order.

Table 4 - 1. BIOS Functions (Functional Order) (* VGA Extensions; ** New VGA Functions)

Page #	Func #	Name	Entry Registers	Exit Registers
4-7	*0	Set Mode	AH, AL	-
4-15	F	Get Video State	AH	AX, BH
4-9	5	Set Active Page	AH, AL	-
4-8	1	Set Cursor Type	AH, CX	-
4-8	2	Set Cursor Position	AH, BH, DX	-
4-8	3	Read Cursor Position	AH, BH	CX, DX
4-12	9	Write Character & Attribute	AH, AL, BX, CX	-
4-13	A	Write Character Only	AH, AL, BX, CX	-
4-12	8	Read Character & Attribute	AH, BH	AX
4-15	E	Write TTY	AH, AL, BL	-
4-23	13	Write String	AH, AL, BX, BP, CX,DX,ES	-
4-10	6	Scroll Up	AH, AL, BH, CX, DX	-
4-10	7	Scroll Down	AH, AL, BH, CX, DX	-
4-14	C	Write Dot	AH, AL, BH, CX, DX	-
4-14	D	Read Dot	AH, BH, CX, DX	AL
4-13	B	Set Color Palette	AH, BX	-
4-16	*10	Set Palette Registers	AH, AL, BX, DX, ES	-
4-18	*11	Load Character Font Info	AH, AL, CX, ES, BP, BX, DX	ES:BP, CX, DL
4-20	*12	Alternate Select	AH, BL, AL	AL, BX, CX
4-9	4	Read Light Pen Position	AH	BX, CX, DX
4-24	**1A	Return DCC Info	AH, AL	AL, BX
4-24	**1B	Return Functionality Info	AH, AL, ES: DI, BX	AL
4-28	**1C	Save/Restore State	AH, AL, ES: BX, CX	AL, BX
4-29	6F	V7 Extended Functions	AH, AL, BX	AX, BX, CX

Table 4 - 2. BIOS Function Summary (Numerical Order) (* VGA Extension; ** New VGA Function)

Page	Function	Subfunction	Description
4-7	*00h		Set Mode
4-8	01h		Set Cursor Type
4-8	02h		Set Cursor Position
4-8	03h		Read Cursor Position
4-9	04h		Read Light Pen Position
4-9	05h		Set Active Display Page
4-10	06h		Scroll Active Page Up
4-10	07h		Scroll Active Page Down
4-12	08h		Read Character & Attribute at Current Cursor Position
4-12	09h		Write Character & Attribute to Current Cursor Position
4-13	0Ah		Write Character Only to Current Cursor Position
4-13	0Bh		Set Color Palette
4-14	0Ch		Write Dot
4-14	0Dh		Read Dot
4-15	0Eh		Write TTY-style to Active Page
4-15	0Fh		Return Current Video State
4-16	*10h		Set Palette Registers
4-16		00h	Set Individual Palette Register
4-16		01h	Set Overscan Register
4-16		02h	Set All Palette Registers and Overscan Register
4-16		03h	Toggle Intensity / Blinking Bit
4-16		**07h	Read Individual Palette Register
4-16		**08h	Read Overscan Register
4-16		**09h	Read All Palette Registers and Overscan Register
4-16		**10h	Set Individual Color Register
4-16		**12h	Set Block of Color Registers
4-17		**13h	Select Color Page
4-17		**15h	Read Individual Color Register
4-17		**17h	Read Block of Color Registers
4-17		**1Ah	Read Current Color Page Number
4-17		**1Bh	Sum Color Values to Gray Scale
4-18	*11h		Load Font Info
4-18		00h	Load User Font
4-18		01h	Load ROM Monochrome Font
4-18		02h	Load ROM 8x8 Font
4-18		03h	Set Block Specifier
4-18		**04h	Load ROM 8x16 Font
4-18		10h	Load User Font
4-18		11h	Load ROM Monochrome Font
4-18		12h	Load ROM 8x8 Font
4-19		**14h	Load ROM 8x16 Font
4-19		20h	Load User Graphics Characters INT 1Fh (8x8)
4-19		21h	Load User Graphics Characters
4-19		22h	Load Graphics Mode ROM 8x14 Font
4-19		23h	Load Graphics Mode ROM 8x8 Font
4-19		**24h	Load Graphics Mode ROM 8x16 Font
4-19		*30h	Return Character Font Information
4-20	*12h		Alternate Select
4-20		10h	Return Video Information
4-20		20h	Select Alternate Print Screen Routine
4-20		**30h	Select Scan Lines for Text Modes
4-20		**31h	Default Palette Loading During Mode Set
4-21		**32h	Video Enable / Disable
4-21		**33h	Summing to Gray Scales
4-21		**34h	Cursor Emulation
4-22		**35h	Display Switch
4-22		**36h	Video Screen On / Off
4-23	13h		Write Text String
4-24	**1Ah		Return Display Combination Code (DCC)
4-24	**1Bh		Return Functionality / State Info
4-28	**1Ch		Save / Restore Video State

In order to make use of the predefined BIOS modes, the program must tell the BIOS what mode to use. This is done by calling a Set Mode function in the BIOS. This is defined as follows:

Entry: AH = 0

AL = mode value:

0*	CGA	40x25	monochrome	text	B8000	8 pages
1*	CGA	40x25	color	text	B8000	8 pages
2*	CGA	80x25	monochrome	text	B8000	8 pages
3*	CGA	80x25	color	text	B8000	8 pages
4	CGA	320x200	4-color	graphics	B8000	1 page
5	CGA	320x200	monochrome	graphics	B8000	1 page
6	CGA	640x200	monochrome	graphics	B8000	1 page
7*	MDA	80x25	monochrome	text	B0000	8 pages
8	Reserved					
9	Reserved					
A	Reserved					
B	Reserved					
C	Reserved					
D	EGA	320x200	16-color	graphics	A0000	8 pages
E	EGA	640x200	16-color	graphics	A0000	4 pages
F	EGA	640x350	monochrome	graphics	A0000	2 pages
10	EGA	640x350	16-color	graphics	A0000	2 pages
11	VGA	640x480	2-color	graphics	A0000	1 page
12	VGA	640x480	16-color	graphics	A0000	1 page
13	VGA	320x200	256-color	graphics	A0000	1 page

* Modes 0-3 are implemented using 8x8, 8x14, or 9x16 fonts and mode 7 using 9x14 or 9x16 fonts depending on the vertical resolution available on the attached display and previously set scan line type (Alt-Select subfunction - 30h).

Exit: None

The Set Mode function provides a means to set the system to a text mode or a graphics mode. It also allows the display to be erased. The no blanking option (setting bit 7 of the AL register) allows the program to change modes without changing the contents of the display. It loads the appropriate character font into plane 2 and set the colors to default colors in the palette. It also sets the cursor position to 0, 0 for all pages.

NOTE: If the high (MSB) bit of AL is set it prevents clearing of display memory.

Function Differences Between Text and Graphics Modes

All of the text mode and scrolling functions described in later sections are also available in graphics mode. There are some differences, however.

- Attributes are not defined for graphics modes.
- Colors can be specified in the Write Character function and in scrolling the color of the area to be blanked can be provided.
- The Write TTY function is not capable of determining the attributes of the display and will assume the default of a black background when writing text in graphics mode.
- All the text mode functions assume a character position cursor definition. Because of this, if you have a 320 x 200 graphics mode display, you can only have 40 columns and 25 rows of text to which you can position the cursor.
- Characters in graphics modes can be XOR'ed to the screen.

Cursor Functions

The cursor defines where the next character will be placed on the screen. This is often shown as a blinking underline. The shape of the cursor can be changed using the Set Cursor Type function as shown:

Int 10 - Function 1 **Set Cursor Type**

Entry: AH = 01h
CH = Cursor start line (bits 4-0)
CL = Cursor end line (bits 4-0)

Exit: None

NOTE: Setting bit 5 or 6 in start line (CH) will cause erratic blinking or no cursor at all.

The shape of the cursor can be defined as anything between a blinking box and one line, or it can be turned off completely.

The BIOS, in order to maintain backwards compatibility, makes the display look like a Color Graphics Adapter. It defines the cursor shape as though it is an 8x8 character, which is the only possibility on a Color Graphics Adapter. In VGA text modes, the character can actually be a 9x16 character. The BIOS takes the cursor shape as though it was an 8x8 character, and makes it look proper for a 9x16 character definition.

Through the BIOS all characters written to the screen are placed at the current cursor position. The program must specify where the cursor is placed. The Set Cursor Position function performs this as shown:

Int 10 - Function 2 **Set Cursor Position**

Entry: AH = 02h
DH = row
DL = column
BH = page number

Exit: None

When the cursor position is set, all character writes and reads will be at that position (0, 0 is the upper left). To determine the cursor position there is a Read Cursor Position function as shown below:

Int 10 - Function 3 **Read Cursor Position**

Entry: AH = 03h
BH = page number

Exit: DH = row
DL = column
CX = current cursor type (see function 1)

Int 10 - Function 4 **Read Light Pen Position**

Entry: AH = 4

Exit: AH = 0: Light pen switch not down/not triggered

The light pen is not supported in the VGA; this function always returns 0 in AH.

Int 10 - Function 5 **Set Active Page**

The V7 VGA contains 256K of memory, of which only a limited amount is displayed at any one time. In order to use this memory, most display modes have several pages or screens that can be displayed. Only one screen can be active at a time, the other screens are accessible by the CPU but are not displayed on the screen.

Entry: AH = 05h
AL = new page value

Exit: None

The EGA/VGA BIOS maintains the current cursor position for each page. When selecting the active page, the previous cursor position is retained. Paging can thus be used for rapidly changing the display or for animation.

Scrolling Functions

There are two scrolling functions, Scroll Up and Scroll Down. They are functionally similar, and are shown below:

Int 10 - Function 6 **Scroll Active Page Up**

Entry:
AH = 06h
AL = Number of lines (input lines blanked at bottom of window)
(AL = 0 means blank entire window)
BH = Attribute to be used on blank line
CH,CL = Row, column of upper left corner of scroll
DH,DL = Row, column of lower right corner of scroll

Exit: None

Int 10 - Function 7 **Scroll Active Page Down**

Entry:
AH = 07h
AL = Number of lines (input lines blanked at top of window)
(0 means blank entire window)
BH = Attribute to be used on blank line
CH,CL = Row, column of upper left corner of scroll
DH,DL = Row, column of lower right corner of scroll

Exit: None

These functions are used only for the currently displayed active screen page, and allows defining an area of the display which will be moved. The program defines the top left corner and the bottom right corner of this window and the number of lines to be scrolled. The lines that are moved and the ones that are being moved (scrolled) off the screen will be lost. A special characteristic of the scroll functions is that if you specify no lines (0) to be scrolled, the BIOS will clear the entire window, so that all characters within that window will be blanked. The attribute to be saved into that window is loaded into BH.

Character / Attribute Functions

Once a cursor position has been defined, the program can place text at the defined position. There are functions available to read and write characters and attributes.

The character codes are an extension of the standard ASCII character set. Some of the lower range characters which are normally control functions are also defined as displayable characters. Beyond the first 128 character codes there are extended characters that provide line drawing capability, foreign character definition, and mathematical symbols.

The attribute codes define the display characteristics. This includes blinking, foreground and background color, and in some instances, the character set used. Although the monochrome display mode allows normal video and intensified video, they are not entirely compatible with the Monochrome Display Adapter. Table 4-3 shows a listing of the color mapping / attribute codes.

Table 4-3. Color Mapping / Attribute Codes

Attribute	I R G B	Monochrome	Color
00h	0 0 0 0	Black	Black
01h	0 0 0 1	Underline	Blue
02h	0 0 1 0	Video	Green
03h	0 0 1 1	Video	Cyan
04h	0 1 0 0	Video	Red
05h	0 1 0 1	Video	Magenta
06h	0 1 1 0	Video	Brown
07h	0 1 1 1	Video	White
08h	1 0 0 0	Black	Dark Gray
09h	1 0 0 1	Underline	Light Blue
0Ah	1 0 1 0	Video	Light Green
0Bh	1 0 1 1	Video	Light Cyan
0Ch	1 1 0 0	Video	Light Red
0Dh	1 1 0 1	Video	Light Magenta
0Eh	1 1 1 0	Video	Yellow
0Fh	1 1 1 1	Video	Intensified White

For read/write character functions while in CGA graphics mode (4, 5 and 6), characters are formed from a character generator image maintained in system ROM. Only the first 128 characters are maintained there. To read/write the second 128 characters, a pointer at interrupt 1Fh (memory location 0007Ch) points to a 1K byte table containing the code points for the second 128 characters (128-255).

For EGA graphics modes (D - 10) and the new VGA graphics modes (11h-13h), all 256 graphics characters are supplied in the system ROM. There is a pointer to these characters at int 43h (location 0:010C).

For the write character functions while in graphics mode, the replication factor contained in CX on entry will produce valid results only for characters contained on the same row. Continuation to succeeding lines will not produce correctly.

Int 10 - Function 8 **Read Character and Attribute**

Entry: AH = 08h
BH = page

Exit: AL = Character read
AH = Attribute of character read (Alpha modes only)

The Read Character and Attribute function returns a character and the associated attribute as shown at the cursor position on the display. It does not need to be the current page so the character may not be visible on the screen.

NOTE: Graphics modes must have a background color of 0 for the read back to work properly.

Int 10 - Function 9 **Write Character and Attribute**

Entry: AH = 09h
AL = Character to write
BL = Attribute of character (Alpha mode)
BL = Color of character (Graphics mode)
BH = Display page
CX = Number of times to write character

Exit: None

NOTE: In graphics mode, if bit 7 of BL is 1, then the character is XOR'ed with the screen.

Int 10 - Function A **Write Character Only**

Entry: AH = 0AH
AL = Character to write
BH = Display page
BL = Foreground color (Graphics ONLY)
CX = Count of characters to write

Exit: None

The Write Character Only function will only change the character definition and not the attribute definition for that cursor position.

Color Palette Functions

The V7 VGA BIOS provides the ability for a programmer to define different colors to be displayed on the screen. This is performed with the Set Color Palette and Set Palette Registers functions (see Function 10, page 16).

Int 10 - Function B **Set Color Palette**

Entry: AH = 0Bh
BH = Palette color ID being set (0-127)
BL = Color value to be used with that color ID

Where: Color ID = 0 selects the background color (0-15)
Color ID = 1 selects the palette to be used: 0 = Green(1)/Red(2)/Brown(3)
1 = Cyan(1)/Magenta(2)/White(3)

Exit: None

In 40x25 or 80x25 text modes, the value set for palette color 0 indicates the border color (0-31, where 16-31 select the high-intensity background set.)

This function is provided for compatibility with CGA BIOS code. For the EGA and VGA, this function is needed only for 320x200 (4-color) graphics. The new EGA/VGA palette function 10h (Set Palette Registers) provides a super-set of the functionality here.

Graphics Mode Functions

The BIOS capabilities for doing graphics is limited. They allow you to write or retrieve the current value of a pixel at a given row and column location of a specific page. These two functions are relatively slow and do not provide good support for doing extensive graphics. They are primarily there to provide easy access for drawing pixels on the screen.

Int 10 - Function C **Write Dot**

Entry:
AH = 0Ch
BH = Page
DX = Row number
CX = Column number
AL = Color value for pixel

Exit: None

NOTE: If bit 7 of AL = 1, then the color value is XOR'ed with the current contents of the dot.

Using this function the program must specify a pixel row and pixel column, the page to write the pixel to, and the color value. The color value can range from 0 to 255 depending on the display mode. In a 4-color display, it can only use 0-3, in a 16-color display it can do 0-15, in a 256-color display it can do 0-255. All modes except mode 13h can also do an XOR on the current contents of the display. This is useful for changing the display temporarily for "rubber band" type operation and highlighting certain areas of the display.

Int 10 - Function D **Read Dot**

Entry:
AH = 0Dh
BH = Page
DX = Row number
CX = Column number

Exit: AL = Color of dot read

Using this function the program must specify a pixel row and pixel column, and the page to read the pixel from. The function will return the value of the color read in the AL register. The color value can range from 0 to 255 depending on the number of colors displayable by the current display mode.

Int 10 - Function E **Write TTY**

The Write TTY function allows the program to display text on the screen without having to update the cursor position before each character write. As the cursor goes to the right side of the screen it will wrap to the beginning of the next line, and as the cursor goes off the bottom of the screen it will automatically scroll up the screen.

Entry: AH = 0Eh
AL = Character to write
BL = Foreground color in graphics mode

Exit: None

NOTE: Screen width is controlled by the previous mode set.

The Write TTY function has several predefined special characters which perform special action:

- CR (carriage return, ASCII 0Dh) will return the cursor to column 0 on the same line.
- LF (line feed, ASCII 0Ah) will leave the column position the same but go down one line, scrolling the screen if the cursor is at the bottom of the screen.
- BS (back space, ASCII 08h) will move the cursor position back one position.
- BL (bell, ASCII 07h) will not change the cursor position but will cause the speaker to beep once.

Int 10 - Function F **Get Video State**

In order to determine the current state of the EGA/VGA card, there is a Get Video State function.

Entry: AH = 0Fh

Exit: AL = mode currently set
AH = number of character columns on screen
BH = current active display page

NOTE: If the program uses the no blank option on the Set Mode function, this will be returned in the Get Video State function. This flag (mode byte msb) will be set if the last call to set mode was to not blank the screen.

Color Palette Functions

The V7 VGA BIOS provides the ability for a programmer to define different colors to be displayed on the screen. This is performed with the Set Color Palette and Set Palette Registers functions.

Int 10 - Function 10 **Set Palette Registers**

The Set Palette Registers function allows the program to make full use of the extended colors on the EGA/VGA.

Entry: AH = 10h

AL = 0: Set individual palette register:
BL = Palette register to be set
BH = Value to set

AL = 1: Set overscan register:
BH = Value to set

AL = 2: Set palette registers and overscan:
ES:DX = Pointer to 17 byte table: Bytes 0-15 are palette values
Byte 16 is the overscan value

AL = 3: Toggle intensify / blinking bit:
BL = 0: Enable intensify
BL = 1: Enable blinking

This redefines one of the bits in the attribute code to allow for 16 colors in background. When intensify is enabled it provides 16 background colors and 16 foreground colors. When blinking is enabled it provides 8 background colors plus a blinking character.

AL = 7: Read individual palette register
BL = Palette register to be read
BH = Value read

AL = 8: Read overscan register
BH = Value read

AL = 9: Read all palette registers and overscan
ES:DX = Pointer to 17 byte table: Bytes 0-15 are palette values
Byte 16 is the overscan value

AL = 10: Set individual color register
BX = Color register number CH = Green value
CL = Blue value
DH = Red value

AL = 12: Set block of color registers
BX = Number of first clolor register
CX = Number of registers to be set
ES:DX = Pointer to a table of color values.
The table should contain color values be in the sequence:<(red, green, blue), (red, green, blue)....(red, green blue)>

Entry: (cont'd)

AL = 13: Select color page
BH = Paging mode or value
BL = 0, select paging mode:
BH = 0 selects four pages of 64 color registers

BH = 1 selects 16 pages of color registers
BL = 1, select page:
BH = number of the required page (0- 3 or 0-15)

AL = 15: Read individual color register
BX = Number of color register
CH = Green value

CL = Blue value

DH = Red value

AL = 17: Read block of color registers
BX = Number of first color register
CX = Number of registers to be read
ES:DX = Pointer to a table to receive the color values
The table should contain color values be in the sequence:<(red, green, blue), (red, green, blue)....(red, green blue)>

AL = 1A: Read current color page number
BH = Current page
BL = Paging mode

AL = 1B: Sum color values to gray scale
BX = first color register to be summed
CX = Number of registers to sum

Exit: None

The V7 VGA provides loadable character set capability in text mode. The fonts are saved in plane 2 of display memory. When in text mode, character font information is automatically retrieved from plane 2 for each character and displayed at the proper position on the display.

Font table manipulation provides the ability to load fonts, use fonts contained within the BIOS ROM, or define fonts. There is room for the definition of eight 256-character fonts within the VGA, two of which can be displayed at any one time.

The character font load function should only be called after doing a set mode operation and before changing any of the characteristics of the display such as the cursor size. This is because the process of loading a character font causes the EGA/VGA registers to be reprogrammed so that the BIOS can load the font.

Entry: AH = 11h

AL = 0xh Initiate mode set, completely resetting the video environment, but maintaining display memory:

AL = 00h Load User Font
ES:BP = Pointer to user table
CX = Count to store
DX = Character offset into table
BH = Number of bytes per character BL = Block to load
AL = 01h Load ROM Monochrome Font: BL = Block to load
AL = 02h Load ROM 8x8 Double Dot Font: BL = Block to load
AL = 03h Set Block Specifier
BL = Font Block Specifier D3-D2 Attr bit-3 = 1, font 0-3
D1-D0 Attr bit-3 = 0, font 0-3

Note: When using AL = 3, a function call of AX = 1000h, BX = 0712h is recommended to set the color planes resulting in 512 characters and 8 consistent colors
AL = 04h Load ROM 8x16 character set: BL = Target block

AL = 1xh Similar to (AL = 0x) functions except that:

- Page 0 must be active
- POINTS (bytes/character) will be recalculated
- ROWS will be recalculated from: INT((200, 350 or 400) / POINTS) - 1
- CRT_LEN will be calculated from: (ROWS + 1) * CRT_COLS * 2
- The CRTC will be reprogrammed as follows:

CR09 = POINTS - 1 (only in mode 7)	(Max Scan Line)
CR0A = POINTS - 2	(Cursor Start)
CR0B = 0	(Cursor End)
CR12 = ((ROWS+1)*POINTS)-1	(Vert Disp End)
CR14 = POINTS	(Underline Loc)

The above register calculations must be close to the original table values or undetermined results will occur. The functions in this group should only be called immediately after a mode set or undetermined results will occur.

AL = 10h Load User Font
ES:BP = Pointer to user table
CX = Count to store
DX = Character offset into table
BH = Number of bytes per character BL = Block to load
AL = 11h Load ROM Monochrome Font: BL = Block to load
AL = 12h Load ROM 8x8 Double Dot Font: BL = Block to load

Entry (cont'd):

AL = 14h Load ROM 8x16 character set: BL = Target block

AL = 20h Load user graphics characters int 1fh (8x8) ES:BP = Pointer to user table

AL = 21h Load user graphics characters ES:BP = Pointer to user table
CX = Points (bytes per character)
BL = Row specifier
BL = 0 User (DL = Rows)
BL = 1 14 (0Eh)
BL = 2 25 (19h)
BL = 3 43 (2Bh)

AL = 22h Load ROM 8x14 Font: BL = Row specifier

AL = 23h Load ROM 8x8 Font: BL = Row specifier

AL = 24h Load Graphics mode ROM 8x16 set
BL = Identifies the number of rows on the screen: BL = 1 14 rows
BL = 2 25 rows
BL = 3 43 rows

AL = 30h Get Font Information

CX = Points
DL = Rows

BH = 0 Return Int 1Fh Pointer: ES:BP = Pointer to table
BH = 1 Return Int 44h Pointer: ES:BP = Pointer to table
BH = 2 Return ROM 8x14 Font Pointer: ES:BP = Pointer to table
BH = 3 Return ROM 8x8 Font Pointer: ES:BP = Pointer to table
BH = 4 Return ROM 8x8 Font Pointer (Top): ES:BP = Pointer to table
BH = 5 Return ROM Alternate 9x14 Pointer: ES:BP = Pointer to table

Exit:

BH = 6 Return ROM 8x16 Font Pointer
BH = 7 Return 9x16 Replacement Font Pointer

Int 10 - Function 12 - Subfunction 10 — **Get EGA / VGA Info**

Entry: AH = 12h
BL = 10h: Return EGA/VGA information

Exit: BH = 0: Color mode in effect (3Dx)
BH = 1: Mono mode in effect (3Bx)
BL = Memory Size: 0 = 64k, 1 = 128k, 2 = 193k, 3 = 256k
CH = Feature Bits
CL = Switch Settings

This function is used to retrieve certain parameters from the EGA/VGA, or determine if an EGA/VGA is present. The best way to do this is to clear the CX register before calling this function. When the function is done, if CX is 0, there is no EGA/VGA in the system. This also provides the ability to find out if the EGA/VGA is in color mode or in monochrome mode, and to check feature bits and switch settings. It also can determine memory size, though on the V7 VGA it will always show 256K of memory. The feature bits are currently not defined since there are no feature adapters currently in use.

Int 10 - Function 12 - Subfunction 20 — **Select Alternate**

Entry: AH = 12h
BL = 20h: Select Alternate Print Screen Routine

Exit: None

The Select Alternate function allows defining an alternate print screen routine. This is an enhanced print screen routine that will print all of the rows on the screen. Most system board BIOS ROMS only print out 25 lines. This alternate print screen function will print out however many number of lines are on the screen. Note that this function only works in text mode, not graphics mode. The default is that the normal system BIOS ROM print screen function is used.

Int 10 - Function 12 - Subfunction 30 — **Select Scan Lines**

Entry: AH =12h
AL = Number of scan lines: 0 = 200 lines, 1 = 350 lines, 2 = 400 lines
BL = 30h Select scan lines for alphanumeric modes

Exit: AL = 12h to indicate that this function is supported

This function sets the number of scan lines in text mode, and takes effect in the next mode set.

Int 10 - Function 12 - Subfunction 31 — **Default Palette Loading**

Entry: AH = 12h
AL = 0 Enable palette loading
AL = 1 Disable palette loading
BL = 31h Default palette loading during mode set

Exit: AL = 12h to indicate that this function is supported

When palette loading is disabled, neither the internal palette nor the DAC will be modified during mode sets or other BIOS calls.

Int 10 - Function 12 - Subfunction 32 — **Video Enable/Disable**

Entry:	AH = 12h	
	AL = 0	enable video
	AL = 1	disable video
	BL = 32h	Select video enable or disable
Exit:	AL = 12h	to indicate that this function is supported

When the video subsystem function is disabled, all I/O and video memory R/W are disabled.

Int 10 - Function 12 - Subfunction 33 — **Summing To Gray Scales**

Entry:	AH = 12h	
	AL = 0	enable summing
	AL = 1	disable summing
	BL = 33h	Selects summing enable or disable
Exit:	AL = 12h	to indicate that this function is supported

This function will calculate a relative shade of gray based on 30% red, 59% green and 11% blue.

Int 10 - Function 12 - Subfunction 34 — **Cursor Emulation**

Entry:	AH = 12h	
	AL = 0	enable emulation
	AL = 1	disable emulation
	BL = 33h	Selects emulation enable or disable
Exit:	AL = 12h	to indicate that this function is supported

When cursor emulation is disabled, the cursor start/stop will be set exactly by the cursor type BIOS function. When cursor emulation is enabled, the following algorithm is in effect.

Parameters	Cursor Type
Bit 5 = 1	No cursor
START < END = < 3	Overbar cursor
START + 2 > = END	Underline cursor
START = > 2	Half-block cursor
START = < 2 or END < START	Full-block cursor

Int 10 - Function 12 - Subfunction 35 **Display Switch**

Entry:	AH = 12h	
	AL = 0	Inital switch off adapter video
	AL = 1	Initial switch on planar video
	AL = 2	Switch off active display
	AL = 3	Switch on inactive display
	BL = 35h	Selects or deselects video device
Exit:	AL = 12h	to indicate that this function is supported

This function will select or deslect the video device. It is used when two video boards overlap in I/O address space.

Int 10 - Function 12 - Subfunction 36 **Video Screen On/Off**

Entry:	AH = 12h	
	AL = 0	enable video output
	AL = 1	disable video output
	BL = 36h	Selects video on or off
Exit:	AL = 12h	to indicate that this function is supported

This function will disable the screen output. The CPU will not be "waited" during writes to video RAM since the CRTC does not need to access video RAM when display is disabled. It is used to update the screen quickly.

The Write String function allows writing more than one character at a time to the display. It also allows the program to write one fixed attribute for the whole screen or a character and an attribute for each position on the screen, so each character has its own attribute. It also provides the ability for the cursor position to be updated or left where it started.

Entry: AH = 13h
ES:BP = Pointer to string to be written
CX = Character only count
DX = Position to begin string, in cursor terms
BH = Page number

AL = 0: Fixed attribute, cursor not moved
BL = attribute

AL = 1: Fixed attribute, cursor is moved
BL = attribute

AL = 2: String includes attributes, cursor not moved

AL = 3: String includes attributes, cursor is moved

Exit: None

The format of the string is char, char, ... for AL = 0 and AL = 1. The format of the string for AL = 2 and AL = 3 is char, attr, char, attr, where an attribute follows each character.

This is an easier interface to use but it is only defined in the EGA/VGA and PC/AT BIOS. It is not found in the IBM PC and PC/XT motherboard BIOS. The Write String function also responds to the CR, LF, BS and Bell codes similar to the Write TTY function.

Int 10 - Function 1A **Display Combination Code (DCC) Handling**

Entry: AH = 1Ah
AL = 0

Exit: AL = 1Ah to indicate that this function is supported

BL = Active Display Device (see table below)
BH = Alternate Display Device (see table below)

00h = No Display	07h = VGA (mono)
01h = MDA	08h = VGA (color)
02h = CGA	09h = (reserved)
03h = (reserved)	0Ah = (reserved)
04h = EGA (mono)	0Bh = MCGA (mono)
05h = EGA (color)	0Ch = MCGA (color)
06h = PGA	

This function will return the display type.

Int 10 - Function 1B **Return Functionality / State Information**

Entry: AH = 1Bh
BX = 0
ES:DI = Pointer to target buffer

Exit: AL = 1Bh to indicate that this function is supported
ES:DI = points to the table described in the table below.

Return Functionality/State Information Table

Entry: BX = 00H
ES:DI = Buffer of size 40H bytes

(DI + 00H) word - Offset to static functionality information
(DI + 02H) word - Segment to static functionality information

Video States: (The following information is dynamically generated and reflects the current video state.)

(DI + 04H) byte - Video mode
(DI + 05H) byte - Columns on screen (character columns on screen)

(DI + 07H) word - Length of regenerator buffer (bytes)
(DI + 09H) word - Starting address in regenerator buffer
(DI + 0BH) word - Cursor position for eight display pages (row, column)

(DI + 1BH) word - Cursor type setting (cursor start/end value)
(DI + 1DH) byte - Active display page
(DI + 1EH) word - CRT controller address (3BX-monochrome, 3DX-color)

(DI + 20H) byte - Current setting of 3x8 register
(DI + 21H) byte - Current setting of 3x9 register

Return Functionality/State Information Table (cont'd)

(DI + 22H) byte - Rows on screen (character lines on screen)
(DI + 23H) word - Character height (scan lines per character)

(DI + 25H) byte - Display combination code (active)
(DI + 26H) byte - Display combination code (alternate)

(DI + 27H) word - Colors supported for current video mode
(DI + 29H) byte - Display pages supported for current video mode

(DI + 2AH) byte - Scan lines in current video mode
= 0 - 200 scan lines
= 1 - 350 scan lines
= 2 - 400 scan lines
= 3 - 480 scan lines
= 4 to 255 - reserved

(DI + 2BH) byte - Primary character block
= 0 - Block 0
= 1 - Block 1
= 2 - Block 2
. . . .
. . . .
= 255 - Block 255

This information is based on block specifier. [See (AH) = 11H, (AL) = 03H]

(DI + 2DH) byte - Miscellaneous state information
Bits 7 ,6 - Reserved
Bit 5 = 0 - Background intensity
= 1 - Blinking
Bit 4 = 1 - Cursor emulation active
Bit 3 = 1 - Mode set default palette loading disabled
Bit 2 = 1 - Monochrome display attached
Bit 1 = 1 - Summing active
Bit 0 = 1 - All modes on all displays active

(DI + 2EH) byte - Reserved
(DI + 2FH) byte - Reserved
(DI + 30H) byte - Reserved

(DI + 31H) byte - Video memory available
= 0 - 64Kb
= 1 - 128 Kb
= 2 - 192 Kb
= 3 - 256 Kb
= 4 to 255 - Reserved

Return Functionality/State Information Table (cont'd)

(DI + 32H) byte - Save pointer state information
Bits 7 ,6 - Reserved
Bit 5 = 1 - DCC extension active
Bit 4 = 1 - Palette override active
Bit 3 = 1 - Graphics font override active
Bit 2 = 1 - Alpha Font override active
Bit 1 = 1 - Dynamic save area active
Bit 0 = 1 - 512-character set active

(DI + 33H) to (DI + 3FH) 13 bytes - Reserved

Format of static functionality table:
0 = Not supported
1 = Supported

(00H) byte - Video modes
Bit 7 = Mode 07H
Bit 6 = Mode 06H
Bit 5 = Mode 05H
Bit 4 = Mode 04H
Bit 3 = Mode 03H
Bit 2 = Mode 02H
Bit 1 = Mode 01H
Bit 0 = Mode 00H

(01H) byte - Video modes
Bit 7 = Mode 0FH
Bit 6 = Mode 0EH
Bit 5 = Mode 0DH
Bit 4 = Mode 0CH
Bit 3 = Mode 0BH
Bit 2 = Mode 0AH
Bit 1 = Mode 09H
Bit 0 = Mode 08H

(02H) byte - Video modes
Bits 7 to 4 - Reserved
Bit 3 = Mode 13H
Bit 2 = Mode 12H
Bit 1 = Mode 11H
Bit 0 = Mode 10H

See (AH) = 00H for video mode information

(03h) to (07H) 4 bytes - Reserved

(07H) byte - Scan lines available in text modes
Bit 7 to 3 - Reserved
Bit 2 = 400 scan lines
Bit 1 = 350 scan lines
Bit 0 = 200 scan lines

See (AH) = 12H, (BL) = 30H for text mode scan line selection.

Return Functionality/State Information Table (cont'd)

(08H) byte - Character blocks available in text modes
(09H) byte - Maximum number of active character blocks in text modes

See (AH) = 11H for character block loading interfaces.

(0AH) byte - Miscellaneous functions
Bit 7 = Color paging [see (AH) = 10H]
Bit 6 = Color palette [see (AH) = 10H]
Bit 5 = EGA palette [see (AH) = 10H]
Bit 4 = Cursor emulation [see (AH) = 01H]
Bit 3 = Mode set default palette loading [see (AH) = 12H]
Bit 2 = Character font loading [see (AH) = 11H]
Bit 1 = Summing [see (AH) = 10H and (AH) = 12H]
Bit 0 = All modes on all displays

(0BH) byte - Miscellaneous functions
Bits 7 to 4 - Reserved
Bit 3 = DCC [see (AH) = 1AH]
Bit 2 = Background intensity/blinking control [see (AH) = 10H]
Bit 1 = Save/restore [see (AH) = 1CH]
Bit 0 = Light pen [see (AH) = 04H]

(0CH to 0DH) 2 bytes - Reserved

(0EH) byte - Save pointer functions
Bits 7, 6 - Reserved
Bit 5 = DCC extension
Bit 4 = Palette override
Bit 3 = Graphics font override
Bit 2 = Alpha font override
Bit 1 = Dynamic save area
Bit 0 = 512-character set

(0FH) byte - Reserved

Entry:	AH = 1CH	
	AL = 00H	Return size of save/restore buffer
	CX -	Requested states (see supported save/restore states below)
Exit:	AL = 1CH	to indicate that this function is supported
	BX	Save/restore buffer size block count [number of 64-bytes blocks for saving requested states in (CX)]
Entry:	AL = 01H	Save video state
	CX -	Requested states (see supported save/restore states below)
	(ES:BX)	Buffer pointer to save state
Exit:	AL = 1CH	to indicate that this function is supported
Entry:	AL = 1	Restore video state
	CX -	Requested states (see supported save/restore states below)
	ES:BX	Pointer to save/restore buffer
Exit:	AL = 1CH	to indicate that this function is supported

Supported save/restore states:

Bits 15 to 3 - Reserved and set to 0
Bit 2 = 1 - Save/restore video DAC state and color registers
Bit 1 = 1 - Save/restore video BIOS data area
Bit 0 = 1 - Save/restore video hardware state

This function completely saves or restores a video state to or from a RAM buffer set aside by an application program. The program should first call this function with AL = 0 and then allocate the necessary space. The current video state is altered during a save state operation. To maintain the current video state, perform a restore state operation.

Int 10 - Function 6F **Extended Functions**

Entry: AH = 6Fh

AL = 00h:	Inquire	Page 30
AL = 01h:	Get Info	Page 30
AL = 02h:	(reserved)	Not implemented in V7 VGA BIOS
AL = 03h:	(reserved)	Not implemented in V7 VGA BIOS
AL = 04h:	Get Resolution	Page 30
AL = 05h:	Extended Set Mode	Page 31
AL = 06h:	Select Autoswitch Mode	Page 31
AL = 07h:	Get Video Memory Configuration	Page 31

Exit: If invalid subfunction, AH = 2
If valid subfunction, refer to individual subfunction descriptions on following pages

Int 10 - Function 6F - Subfunction 0 **Inquire**

Exit: BX = 'V7' (indicates extensions are present)

Int 10 - Function 6F - Subfunction 1 **Get Info**

Exit: AL = Reserved (used to return the monitor-type code in VEGA VGA)

AH = Status Register Information:

Bit-7-6:	Diagnostic Bits	
Bit-5:	Display type:	0=color, 1=monochrome
Bit-4:	Monitor Resolution:	0 = High Res (> 200 lines)
		1 = Low Res (<= 200 lines)
Bit-3:	Vertical Sync	
Bit-2:	Light Pen Switch Activated	
Bit-1:	Light Pen Flip Flop Set	
Bit-0:	Display Enabled	0 = Display Enabled
		1 = Vertical or Horizontal Retrace in progress

Note that bits 0-3 are the same as the EGA/VGA status register bits 0-3

Int 10 - Function 6F - Subfunction 4 **Get Mode and Screen Resolution**

Exit:

AL =	Current Video Mode (see Extended Set Mode: subfunction 5)		
BX =	Horizontal	Columns/Pixels	(Text/Graphics)
CX =	Vertical	Rows/Pixels	(Text/Graphics)

Int 10 - Function 6F - Subfunction 5 — **Extended Set Mode**

Entry: BL = Mode Value:

00h - 13h: Standard IBM modes
14h - 3Fh: Reserved
40h - 5Fh: Extended Text Modes

40 = 80x43 (8x8)
41 = 132x25 (8x14)
42 = 132x43 (8x8)
43 = 80x60 (8x8)
44 = 100x60 (8x8)
45 = 132x28 (8x14)
46-4F = (reserved)

60h - 7Fh: Extended Graphics Modes (6xh = Color, 7xh = Monochrome)

60 = 752x410x16
61 = 720x540x16
62 = 800x600x16
63 = 1024x768x2
64 1024x768x4
65 1024x768x16
66 640x400x256
67 640x480x256
68 720x540x256
69 800x600x256 (Future)
6A-6F = (reserved)

Note that this function provides an alternative to the standard BIOS Set Mode function. It is provided to allow for a consistant way to set Video-Seven specific and IBM standard video modes.

Int 10 - Function 6F - Subfunction 6 — **Select Autoswitch Mode**

Entry: BL = Autoswitch Mode Select:

00h: Select EGA/VGA-Only Modes
01h: Select Autoswitched VGA/EGA/CGA/MGA Modes
02h: Select 'Bootup' CGA/MGA Modes

BH = Enable / Disable 0 = enable selection, 1 = disable selection

Int 10 - Function 6F - Subfunction 7 — **Get Video Memory Configuration**

Exit: AL = 6Fh
AH = Bit-7: 0 = DRAM, 1 = VRAM
Bits 6-0: # of 256K blocks of video memory
BH = Chip Rev (SR8F) (S/C Chip in VEGA VGA)
BL = Chip Rev (SR8E) (G/A Chip in VEGA VGA)
CX = 0

EGA / VGA BIOS Data Area

The BIOS data area in EGA / VGA mode is the same as the CGA and Hercules modes with the exception of the data areas described in Table 4-4. These variables are used by the EGA / VGA BIOS, and are located in the BIOS data area (RAM segment 40H).

Table 4 - 4. EGA / VGA BIOS Data Area

Name	Address	Size	Description
CRT Mode	49	Byte	Current BIOS video mode
CRT Cols	4A	Word	Number of video columns
CRT Len	4C	Word	Size of video buffer
CRT Start	4E	Word	Offset of video page
Cursor Posn	50	Word*8	Col, Row position for 8 pages
Cursor Mode	60	Word	Current cursor mode setting
Active Page	62	Byte	Current page being displayed
Addr 6845	63	Word	Base I/O address for 6845/CRTC
CRT Mode Set	65	Byte	Simulated value of CGA 3X8
CRT Palette	66	Byte	Simulated value of CGA 3X9
Rows	84	Byte	Number of character rows - 1
Points	85	Word	Bytes per character
Info	87	Byte	Miscellaneous information: Bit 7: High bit of mode (1 means don't clear) Bit 5-6: 00 - 64KB memory 01 - 128KB memory 10 - 192KB memory 11 - 256KB memory Bit 4: 0 - Pure VGA Mode 1 - Video Extensions Allowed Bit 3: 0 - EGA/VGA active 1 - EGA/VGA inactive Bit 2: 0 - Write any time 1 - Wait for retrace Bit 1: EGA/VGA has monochrome Bit 0: Emulate cursor type
Info 3	88	Byte	Bit 4-7: Feature bits Bit 0-3: Dip switches
VGA Info	89	Byte	Bit 7: Reserved Bit 6: 0 - Display switch, inactive 1 - Display switch, active Bit 5: Reserved Bit 4: 0 - 400 scan line text mode, inactive 1 - 400 scan line text mode, active

Table 4 - 4. EGA/VGA BIOS Data Area (cont'd)

			Bit 3: 0 - Default palette loading, enabled 1 - Default palette loading, disabled Bit 2: 0 - Monochrome monitor, inactive 1 - Monochrome monitor, active Bit 1: 0 - Palette summing to gray shade, inactive 1 - Palette summing to gray shade, active Bit 0: 0 - All modes on all monitors, inactive. 1 - All modes on all monitors, active.
DCC Code	8A	Byte	Display combination code. See description on page 4-35.
Save PTR	A8	Dword	Address of table pointers to save areas

EGA/VGA BIOS Save Environment Layout

Variable "Save PTR" (address 0:04A8H) in the BIOS data area provides applications with specific video information while maintaining compatibility with the BIOS data area. At initialization, this table is located in ROM. Any changes must be made in a RAM copy. The details of this information are shown in Table 4-5.

Table 4 - 5. EGA/VGA BIOS Save Environment.

Offset	Size	Description
0000	Dword	Standard Video Parameter Table Pointer. This pointer is initialized to the IBM-Standard BIOS VGA parameter table, and must exist.
0004	Dword	Dynamic Save Area Pointer (initially 0). When non-zero, this pointer points to a RAM area in which certain dynamic values are stored. The VGA BIOS will automatically maintain the first 17 locations in the save area which contain the values of the 16 palette registers and the overscan color register. This save area must be at least 256 bytes long, if present.
0008	Dword	Alpha Mode Auxiliary Character Font Pointer (initially 0). If this value is non-zero, then it must point to the following table:

Offset	Size	Description
00	Byte	Bytes/Character
01	Byte	Character font block to load
02	Word	Number of character patterns to store
04	Word	Character offset
06	Dword	Pntr to font table
0A	Byte	Displayable rows, FF = display max.
0B	Byte	Consecutive bytes for mode value with which this font description is to be used, terminated with FF

000C	Dword	Graphics Mode Auxiliary Character Font Pointer (initially 0). When non-zero, this address must point to the following table:

Offset	Size	Description
00	Byte	Displayable rows
01	Word	Bytes/Character
03	Dword	Pointer to a font table
07	Byte	Consecutive bytes for mode value with which this font description is to be used, terminated with FF

0010	Dword	Secondary Save Pointer. The pointer is initialized to the BIOS secondary save pointer. This address must contain a valid pointer and will point to Table 4 - 6 on page 35.
0014	Dword	Pointer to Video Seven Extensions. This address must be a valid pointer and will point to Table 4-7 on page 36.
0018	Dword	Reserved

Table 4 - 6. Secondary Save Pointer Data Area

Offset	Size	Description
0000	Word	Table Length. This pointer is initialized to the BIOS secondary save pointer.
0002	Dword	Display Combination Code (DCC) Table Pointer. This is initialized to ROM DCC table. Whis this value exists, it points to a table described as follows:

Size	Description
Byte	Number of entries in table
Byte	DCC table version number
Byte	Maximum display type code
Byte	Reserved

00, 00	Entry	0	No Displays
00, 01	Entry	1	MDPA
00, 02	Entry	2	CGA
02, 01	Entry	3	MDPA + CGA
00, 04	Entry	4	EGA
04, 01	Entry	5	EGA + MDPA
00, 05	Entry	6	MEGA
02, 05	Entry	7	MEGA + CGA
00, 06	Entry	8	PGC
01, 06	Entry	9	PGC + MDPA
05, 06	Entry	10	PGC + MEGA
00, 08	Entry	11	CVGA
01, 08	Entry	12	CVGA + MDPA
00, 07	Entry	13	MVGA
02, 07	Entry	14	MVGA + CGA
02, 06	Entry	15	MVGA + PGA

006	Dword	Second Alpha Mode Auxiliary Character Generator Pointer. This pointer is initialized to 00:00. When this value is non-zero, it points to a table described as follows:

Size	Description
Byte	Bytes/Character
Byte	Block to load, should be non-zero for normal operation
Byte	Reserved
Dword	Pointer to the Font table
Byte	Consecutive bytes of mode values for this font description. The end of this stream is indicated by a code of 0FFH.

NOTE: Attribute bit 3 is used to switch between primary and secondary fonts. It may be desirable to use the user palette profile to define a palette of consistent colors independent of attribute bit 3.

Table 4 - 6. Secondary Save Pointer Data Area (cont'd)

Offset	Size	Description
000A	Dword	User Palette Profile Table Pointer. This pointer is initialized to 00:00. When this value is non-zero, it points to a table described as follows.

Size	Description
Byte	Underlining flag: 1 = On; 0 = Ignore; -1 = Off (0 = normal operation)
Byte	Reserved
Word	Reserved
Word	Internal palette count (0 - 17; 17 = normal operation)
Word	Internal palette index (0 - 16; 0 = normal operation)
Dword	Pointer to internal palette (Bytes 0-15 are palette values, Byte 16 is the overscan value)
Word	External palette count (0-256; 256 = normal operation)
Word	External Palette Index (0-255; 0 = normal operation)
Dword	Pointer to external palette (The table should contain color values in the sequence: <(red, green, blue), (red, green, blue)..... (red, green, blue)>
Byte	Consecutive bytes of modes values for this font description. The end of this stream is indicated by a byte code of 0FFH.

000E	Dword	Reserved
0002	Dword	Reserved
0004	Dword	Reserved

Table 4 - 7. Video Seven Extensions

Offset	Size	Description
0000	Dword	This is a pointer to an array of extended mode table structures. This must be a valid pointer.

Size	Description
Byte	Mode number minus 40h
Byte	Monitors in this mode works in

Bit 6, 7 = 1 - Reserved
Bit 5 = 1 - Variable frequency analog
Bit 4 = 1 - Fixed frequency analog
Bit 3 = 1 - Multisync
Bit 2 = 1 - Enhanced color monitor
Bit 1 = 1 - CGA color monitor
Bit 0 = 1 - Monochrome

Byte	Scan line type (0 = 200, 1 = 350, 2 = 400, 3 = 480)	
Byte	Number of pages available in this mode	
Word	Number of colors available in this mode	
Word	Number of columns	
Word	Number of rows	
Byte	Value of Timing State Register	(SRFD)
Byte	Value of Clock Select Register	(SRA4)
Byte	(reserved)	
Byte	Value of Compatibility Control Register	(SRFC)
Byte	Value of Bank Select Register	(SRF6)
Byte	Value of Extended Clock Select Register	(SRF8)
Byte	Value of 16-Bit Interface Control Register	(SRFF)
Byte	(reserved)	
Dword	Pointer to the Video Parameter Table Format (see Table 4-6)	

Table 4 - 7. Video Seven Extensions (cont'd)

0004	Dword	Reserved
0008	Dword	This is a pointer to an array of extended register stuctures. This must be a valid pointer.

Size	Description	
Byte	Value of Timing State Register	(SRFD)
Byte	Value of Clock Select Register	(SRA4)
Byte	(reserved)	
Byte	Value of Compatibility Control Register	(SRFC)
Byte	Value of Bank Select Register	(SRF6)
Byte	Value of Extended Clock Select Register	(SRF8)
Byte	Value of 16-Bit Interface Control Register	(SRFF)
Byte	(reserved)	

000C	Dword	Reserved
0010	Dword	Reserved
0014	Dword	Reserved

Video Parameter Table Format

The Video Parameter Tables contain the necessary EGA/VGA register values to support the various display modes. The parameter table is structured as a series of mode tables, each containing 64 bytes as defined in Table 4-8.

Table 4 - 8. Video Parameter Mode Table Format.

Item	Size	Definition
Width	byte	Characters per row
Height	byte	Rows of text minus 1
Char Height	byte	Height of character in pixels
Page Size	word	Bytes per display page
Seq Regs	byte*4	Sequencer register values (SR1-SR4)
Misc Reg	byte	Miscellaneous output register
CRT Regs	byte*25	CRT controller registers (CR0-CR18)
Attr Regs	byte*20	Attribute registers (AR0-AR13)
Graph Regs	byte*9	Graphics controller registers (GR0-GR8)
Total:	byte*64	

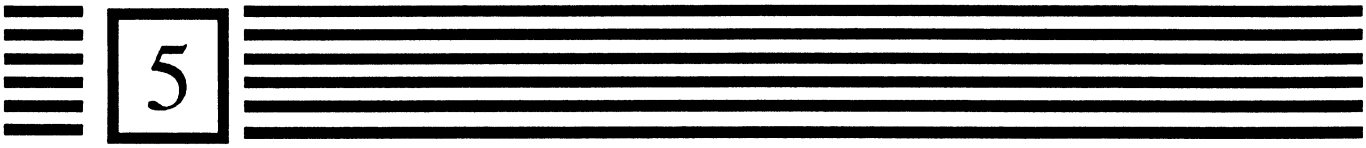
Note: SR0 is not included in the table; it is always set to 3 on a mode set
AR14 is not included in the table; it is always set to 0 on a mode set

The VGA BIOS requires that the mode tables in the Standard Video Parameter Table be in the following sequence and define the VGA register values (given above) necessary for the IBM-standard display modes.

Table 4 - 9. Standard Video Parameter Table

Entry	Mode	Characteristics				Comment	
0	0	CGA	40x25	Mono	Text	8x8 Font	
1	1	CGA	40x25	Color	Text	8x8 Font	
2	2	CGA	80x25	Mono	Text	8x8 Font	
3	3	CGA	80x25	Color	Text	8x8 Font	
4	4	CGA	320x200	Color	Graphics	8x8 Font	
5	5	CGA	320x200	Mono	Graphics	8x8 Font	
6	6	CGA	640x200	Mono	Graphics	8x8 Font	
7	7	MDA	80x25	Mono	Text	8x14 Font	
8	8	-				Reserved	
9	9	-				Reserved	
10	A	-				Reserved	
11	B	-				Reserved	
12	C	-				Reserved	
13	D	EGA	320x200	16-color	Graphics	8x8 Font	
14	E	EGA	640x200	16-color	Graphics	8x8 Font	
15	F	EGA	640x350	Mono	Graphics	64KB Display Ram	
16	10	EGA	640x350	4-color	Graphics	64KB Display Ram	
17	F*	EGA	640x350	Mono	Graphics	>64KB Display Ram	
18	10*	EGA	640x350	16-color	Graphics	>64KB Display Ram	
19	0*	EGA	40x25	Mono	Text	8x14 Font	
20	1*	EGA	40x25	Color	Text	8x14 Font	
21	2*	EGA	80x25	Mono	Text	8x14 Font	
22	3*	EGA	80x25	Color	Text	8x14 Font	
23	0/1+	VGA	40x25	Color	Text	8x16 Font	
24	2/3+	VGA	80x25	Color	Text	8x16 Font	
25	7+	VGA	80x25	Mono	Text	9x16 Font	
26	11	VGA	640x480	2-Color	Graphics	8x16 Font	
26	12	VGA	640x480	16-Color	Graphics	8x16 Font	
27	13	VGA	320x200	256-Color	Graphics	8x8 Font	
28	14	VVGA	752x410	16-Color	Graphics	8x8 Font	
29	15	VVGA	720x540	16-Color	Graphics	8x8 Font	
30	16	VVGA	800x600	16-Color	Graphics	8x8 Font	
31	17	V7VGA	1024x768	2-Color	Graphics	8x16 Font	
32	18	V7VGA	1024x768	4-Color	Graphics	8x16 Font	
33	19	V7VGA	1024x768	16-Color	Graphics	8x16 Font	
34	1A	V7VGA	640x400	256-Color	Graphics	8x16 Font	
35	1B	V7VGA	640x480	256-Color	Graphics	8x16 Font	
36	1C	V7VGA	720x540	256-Color	Graphics	8x8 Font	
37	1D	V7VGA	800x600	256-Color	Graphics	8x8 Font	**(Future)**

CHAPTER



Support Software

5.1 Utilities

The following utilities are provided with all V7VGA-based boards: (Utility Disk Release 1.00)

RAMBIOS.SYS:	Program to run the BIOS as a ram resident program for performance improvement on 286 and 386-class machines. Note that in Compaq 386 systems, the BIOS automatically moves itself into 32-bit system ram, so use of RAMBIOS.SYS is not required on those systems.
V7VGA.COM:	Mode set utility: mono: on/off — Monochrome/color select mono: full/half — Monochrome mode select pure: on/off — Enables strict IBM VGA compatibility save: on/off — Screen-saver on/off save: [n] — Screen saver enable for n minutes nosave — Same as save: off
V7ANSI.SYS	Similar to DOS ANSI.SYS with support for high-resolution text modes
INSTALL.EXE	Utility to assist in installing drivers
DIAG.COM:	V7VGA diagnostic program (also reports ROM BIOS date & switch settings)
ESU.COM:	Text screen sizing utility: 80x25 80x43 80x60 100x60 120x25 120x43 132x25 132x43
DU.COM	Directory listing utility which recognizes extended text modes
CLR.COM	Clear screen utility which recognizes extended text modes
README.TXT	Document file containing last-minute documentation changes and a list of all files on the disk and what they are
README.COM	Displays README.TXT for review

5.2 Drivers

The following drivers have been developed by Video Seven for the V7VGA:

AutoCAD 2.18 thru 2.6:	V7ADI2.COM		
AutoCAD Rel 9 and above:	V7REL9.COM (Supports pull-down menus, AutoSketch, AutoShade)		
AutoCAD Install (all revs):	V7INST.EXE (Modifies screen resolutions, interrupt vector, & colors)		
V7 AutoCAD drivers support all resolutions listed to the right:	640x350x16 (AutoSketch, all boards)	320x200x256 (AutoShade, all)	
	640x480x16 (AutoSketch, all boards)	640x400x256 (AutoShade, all)	
	752x410x16 (AutoSketch, all boards)	640x480x256 (AutoShade, *)	
	720x540x16 (AutoSketch, all boards)	720x540x256 (AutoShade, **)	
	800x600x16 (AutoSketch, all boards)	800x600x256 (AutoShade, **)	
	1024x768x4 (AutoSketch, **)	(* = not available on VGA-16)	
	1024x768x16 (AutoSketch, **)	(** = available on V-RAM VGA only)	

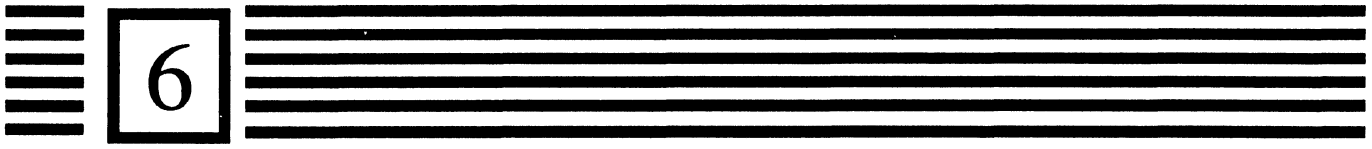
Lotus 2.0, Symphony 1.1:	GD640V20.DRV	(640x480x16 Graphics)
	GD752V20.DRV	(752x410x16 Graphics)
	GD720V20.DRV	(720x540x16 Graphics)
	GD800V20.DRV	(800x600x16 Graphics)
	VD132X25.DRV	(132x25 8x14 Color Text)
	VDMONO25.DRV	(132x25 8x14 Mono Text)
	VD132x43.DRV	(132x43 8x8 Color Text)
	VDMONO43.DRV	(132x43 8x8 Mono Text)
	VD80X60.DRV	(80x60 8x8 Color Text)
	VD100X60.DRV	(100x60 8x8 Color Text)

Microsoft Windows 2.0:	VRAM640 .DRV,GRB,LGO	(640x480x16 for V-RAM VGA)
	VRAM720 .DRV,GRB,LGO	(720x540x16 for V-RAM VGA)
	VRAM800 .DRV,GRB,LGO	(800x600x16 for V-RAM VGA)
	VRAM1024.DRV,GRB,LGO	(1024x768x16 for V-RAM VGA)
	VRAM1KM.DRV,GRB,LGO	(1024x768x2 for V-RAM VGA)
	DRAM640 .DRV,GRB,LGO	(640x480x16 for F/W & VGA-16)
	DRAM720 .DRV,GRB,LGO	(720x540x16 for F/W & VGA-16)
	DRAM800 .DRV,GRB,LGO	(800x600x16 for F/W & VGA-16)
Microsoft Windows 386:	VRAM640 .DRV,GRB,LGO,386,3EX	(640x480x16 for V-RAM VGA)
	VRAM720 .DRV,GRB,LGO,386,3EX	(720x540x16 for V-RAM VGA)
	VRAM800 .DRV,GRB,LGO,386,3EX	(800x600x16 for V-RAM VGA)
	VRAM1024.DRV,GRB,LGO,386,3EX	(1024x768x16 for V-RAM VGA)
	VRAM1KM.DRV,GRB,LGO,386,3EX	(1024x768x2 for V-RAM VGA)
	DRAM640 .DRV,GRB,LGO	(640x480x16 for F/W & VGA-16)
	DRAM720 .DRV,GRB,LGO	(720x540x16 for F/W & VGA-16)
	DRAM800 .DRV,GRB,LGO	(800x600x16 for F/W & VGA-16)
	All Windows drivers include hardware cursor support	
GEM3 / Ventura Publisher		
PCAD	(in development)	

The following is a partial list of products also supported on the V7VGA using standard VGA drivers:

VersaCAD	PC Paint / GRASP	EGA Paint
DGIS / VDI / ...	PC Paintbrush	
MetaWindows	Dr Halo / Halo Library	

CHAPTER



V7VGA Programming

NOTE: The examples in this chapter are provided for clarification only. Video-Seven assumes no responsibility for their functionality or fitness for a specific purpose.

Throughout this chapter, unless otherwise noted, all addresses refer to VGA modes, typically mode 3 for text display (80x25 text) and mode 10 for graphics display (640x350).

Text Memory Map

All EGA and VGA-compatible text modes use video memory in the same format. Memory plane 0 stores the character, plane 1 stores the attributes, plane 2 stores the font tables, and plane 3 is not used. (Note: In the V7VGA, plane 3 is sometimes used to hold font or extended attribute information). This is shown in Figure 6-1.

In text mode the program writes two bytes to video memory to display a character; a character code and an attribute byte. The hardware uses the character code as an index into a lookup table (character ROM) to retrieve the actual bitmap that will form the character as each scan line is displayed. The attribute byte selects foreground and background colors for that particular character cell. Thus, when writing to memory, even addresses store the character, odd addresses store the attribute.

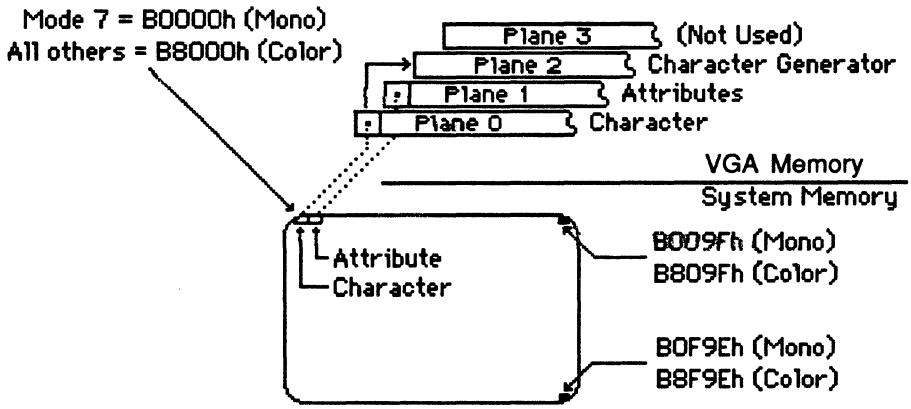


Figure 6-1. Text Mode Video Memory Mapping (Mode 3)

Memory plane 0 stores the character to be displayed on the screen. As shown in Figure 6-1 (color mode), the character at address B8000h is displayed at the top of the screen, the character stored at address B804Fh is displayed at the top right side of the screen. The character stored at address B87CFh is displayed at the bottom right side of the screen. The addresses for monochrome mode begins at address B0000h instead of B8000h, and ends at B07CFh instead of B87CFh. The character code stored at the location in VGA memory plane 0 is actually an address to the character generator in memory plane 2.

Memory plane 1 stores the attribute of the corresponding character. As shown in Figure 6-1, the attribute byte is divided into two parts, one controlling the foreground characteristics and one controlling the background characteristics.

Text Mode Character Attributes

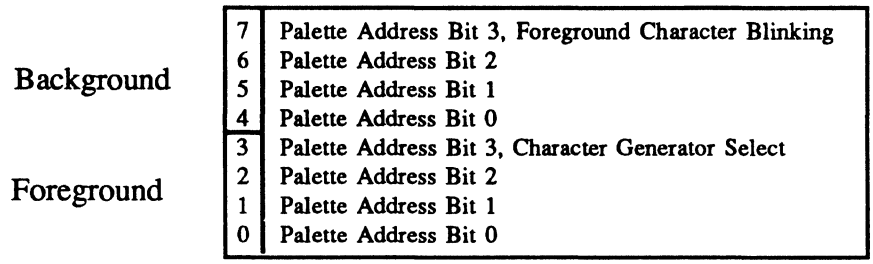


Figure 6-2. Attribute Byte Layout

Attribute bit 3 is used in conjunction with the Character Map Select Register in the Sequencer (SR3) to select one of the character fonts. When the bit is a '0' the character map pointed to by bits 1-0 of SR3 is selected. When the bit is a '1' the character map pointed to by bits 3-2 of SR3 is selected.

Attribute bit 7 enables a blinking character mode on an individual character basis. This function is enabled with bit-3 of the Attribute Controller Mode Control Register (AR10). Refer to chapter 3 for further information on this register.

As shown in Table 6-1, the attributes will have a different effect depending on the mode selected (7 for monochrome or 0-3 for color).

Table 6-1. Text Character Attributes

Foreground Bits	**0**	**1**	**2**	**3**		**Monochrome**	**Monochrome**
Background Bits	**4**	**5**	**6**	**7**	**Color**	**(foreground)**	**(background)**
	0	0	0	0	Black	Black	Black
	1	0	0	0	Blue	White	Black
	0	1	0	0	Green	White	Black
	1	1	0	0	Cyan	White	Black
	0	0	1	0	Red	White	Black
	1	0	1	0	Magenta	White	Black
	0	1	1	0	Brown	White	Black
	1	1	1	0	White	White	White
	0	0	0	1	Gray	Black	Black
	1	0	0	1	Lt. Blue	Int. White	Black
	0	1	0	1	Lt. Green	Int. White	Black
	1	1	0	1	Lt. Cyan	Int. White	Black
	0	0	1	1	Lt. Red	Int. White	Black
	1	0	1	1	Lt. Magenta	Int. White	Black
	0	1	1	1	Lt. Yellow	Int. White	Black
	1	1	1	1	Int. White	Int. White	Int. White

Character Generator

The character generator bit map is loaded from the BIOS ROM into plane 2. The addresses of the character generators are shown in Figure 6-3.

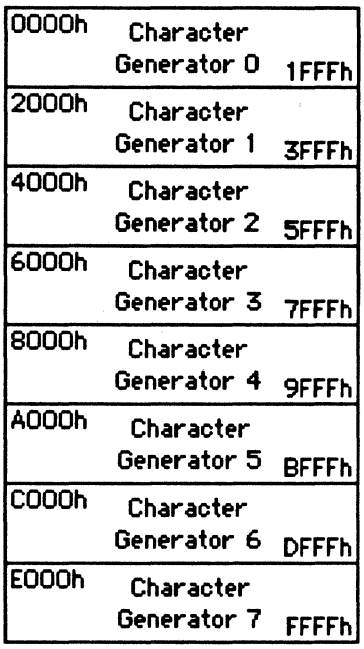


Figure 6-3. Character Font Addresses

The basic EGA only implements character generators 0, 2, 4, and 6 (the memory areas where character generators 1, 3, 5, and 7 are now located was unused in the EGA). To point to the four EGA character generators, Sequencer register 3 (SR3) bits 0-1 pointed to the primary character generator (character attribute bit-3 = 0) and bits 2-3 pointed to the secondary character generator (character attribute bit-3 = 1).

To implement VGA compatibility, the V7VGA adds new bits 4 and 5 to SR3 allowing selection of 2 of 8 fonts instead of 2 of 4. The new bit-4 of SR3 becomes the new lsb of the primary character generator pointer and bit-5 becomes the new lsb of the secondary character generator pointer. The 2 original font selection bits become bits 1 and 2 of the pointer. This scheme is EGA compatible if bits 4 and 5 are set to 0, as all values in SR3 point to the same locations in memory for fonts as before. The renumbering of the fonts shown in the figure above is only a documentation difference.

Graphics Memory Map

Figure 6-4 shows the relationship between addresses in system memory and information in the V7VGA memory.

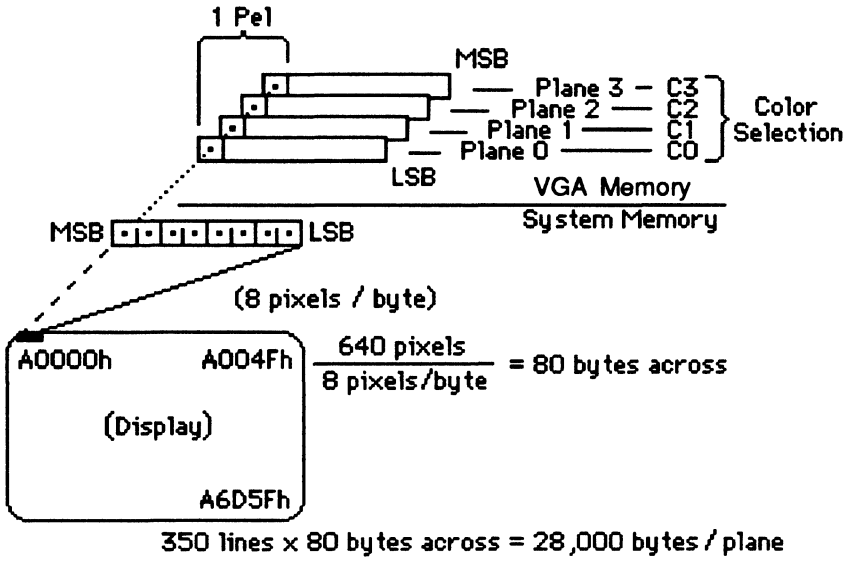


Figure 6-4. Graphics Mode Video Memory Mapping (Mode 10)

Global Programming Information

The information in this section applies to all the programming examples in the following sections.

```
;--------------------------------------------------------------------
;
;     Structure of a BIOS video parameter table for one mode
;
vid_prm                 struc
     width      db      ?           ; number of columns
     height     db      ?           ; number of rows
     chr_ht     db      ?           ; height of a character
     pg_size    dw      ?           ; size of a page in bytes
     r_seq      db      4 dup (?)   ; sequencer registers
     r_misc     db      ?           ; miscellaneous register
     r_crtc     db      25 dup (?)  ; crtc registers
     r_attr     db      20 dup (?)  ; attr. cont. registers
     r_graph    db      9 dup (?)   ; graphics registers
vid_prm         ends
;--------------------------------------------------------------------
;
;     Miscellaneous registers
;
MISC            equ     3c2h        ; miscellaneous output
FEATURE         equ     3dah        ; feature control
STATUS_0        equ     3c2h        ; input status reg 0 (feature read)
STATUS          equ     3dah        ; input status reg 1 (display status)

;--------------------------------------------------------------------
;
;     Sequencer
;
SEQUENCER       equ     3c4h        ; sequencer index
     RESET      equ     0           ; reset
     CLOCK      equ     1           ; clocking mode
     MAPMASK    equ     2           ; map mask
     CHARMAP    equ     3           ; character map select
     MODE       equ     4           ; memory mode

;--------------------------------------------------------------------
;
;     Crt Controller
;
CRTC            equ     3d4h        ; CRTC index
     OVFL       equ     07h         ; overflow register
     PRS        equ     08h         ; preset row scan
     HI_STRT    equ     0ch         ; start address high
     LO_STRT    equ     0dh         ; start address low
     HI_CPOS    equ     0eh         ; cursor position high
     LO_CPOS    equ     0fh         ; cursor position low
     VRE        equ     11h         ; vertical retrace end
     LLW        equ     13h         ; offset register
     LCR        equ     18h         ; line compare register
```

```
;-----------------------------------------------------------------------
;
;     Graphics Controller
;
GRAPHICS        equ     3ceh        ; graphics controller index
     SET_RES    equ     0           ; set/reset
     EN_SET     equ     1           ; enable set reset
     CCMP       equ     2           ; color compare
     ROTATE     equ     3           ; data rotate
     READMAP    equ     4           ; read map select
     GMODE      equ     5           ; graphics mode
     GMISC      equ     6           ; miscellaneous
     CDONTC     equ     7           ; color don't care
     BITMASK    equ     8           ; bit mask

;-----------------------------------------------------------------------
;
;     Attribute Controller
;
ATTR            equ     3c0h        ; attribute controller index
     PALETTE    equ     0           ; 0 - 15 palette registers
     AMODE      equ     10h         ; attribute mode control
     OVRSCAN    equ     11h         ; overscan color register
     PLANE      equ     12h         ; color plane enable
     PAN        equ     13h         ; horizontal panning
     COLOR      equ     14h         ; color select

;-----------------------------------------------------------------------
;
;     The following macro outputs a word value to two consecutive data
;     ports.  Its calling sequence is just as if we were outing ax to dx.
;     The implementation must preserve both the arguments.
;
PC              equ     0
PC_AT           equ     1

outwrd          macro   port,value

if PC
                out     port,value
endif

if PC_AT
                out     port,al     ; index select
                inc     port        ; data register
                xchg    ah,al       ; data into al
                out     port,al     ; data value
                dec     port        ; restore port
                xchg    ah,al       ; restore data
endif
                endm
```

6.2 Setting the Palette Registers

This section shows how to set the colors in the attribute controller. The attribute controller contains 16 registers, 0 through F, which are addressed by the four color bits (C0-C3 as shown in Figure 6-4). These bits select one of 16 colors to be displayed at the specific location. The palette register provides a choice of 64 colors, and any of the 64 colors can be mapped into any of the 16 registers. Figure 6-5 shows a color wheel of the 6 most common colors, the varying shades from low intensity (middle circle) to high intensity (outer circle). Table 6-2 shows the common colors, their hex equivalent (to be loaded into the attribute controller register) and the R, B, B, R', G', and B' equivalents. From this you can calculate the hex value of any other colors.

Table 6-2. Common Color Value Calculation

Hex Value	R G B	R'G'B'	Color
24	1 0 0	1 0 0	Intense Red
20	1 0 0	0 0 0	Bright Red
04	0 0 0	1 0 0	Light Red
2D	1 0 1	1 0 1	Intense Magenta
28	1 0 1	0 0 0	Bright Magenta
05	0 0 0	1 0 1	Light Magenta
09	0 0 1	0 0 1	Intense Blue
08	0 0 1	0 0 0	Bright Blue
01	0 0 0	0 0 1	Light Blue
1B	0 1 1	0 1 1	Intense Cyan
18	0 1 1	0 0 0	Bright Cyan
03	0 0 0	0 1 1	Light Cyan
12	0 1 0	0 1 0	Intense Green
10	0 1 0	0 0 0	Bright Green
02	0 0 0	0 1 0	Light Green
36	1 1 0	1 1 0	Intense Yellow
30	1 1 0	0 0 0	Bright Yellow
06	0 0 0	1 1 0	Light Yellow
3F	1 1 1	1 1 1	Intense White
38	1 1 1	0 0 0	Bright White
07	0 0 0	1 1 1	Gray
00	0 0 0	0 0 0	Black

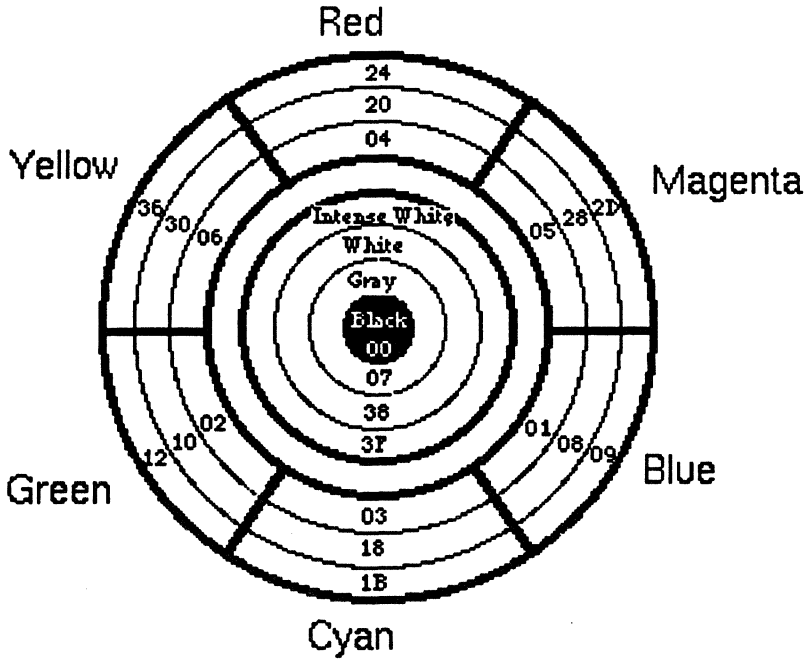


Figure 6-5. Common Color Values.

The following example shows how to set the attribute in one of the 16 registers. This is summarized as follows:

- Disable the interrupts.
- The flip-flop in the attribute controller is set to address a specific register ([bp]).
- A value for that register is output to the register.
- Since this disables video, the flip-flop in the attribute controller is loaded with a 1 in bit 5 to re-enable video.

```
mov     dx,3DAh          ; status one
cli                      ; don't want attr ctlr interupted
in      al,dx            ; set attr address flip/flop
mov     dx,3C0h          ; attr controller
mov     al,reg[bp]       ; disable palette + palette reg.
out     dx,al            ; select palette register
mov     al,val[bp]       ; the color value
out     dx,al            ; select palette value
mov     al,20h           ; palette address source
out     dx,al            ; enable palette
sti                      ; ok to be interrupted now
```

6.3 Writing a Pixel

This section provides an example of how to write a pixel using the graphics controller. This is summarized as follows:

- Calculate the location of the pixel in the display buffer (video RAM). The byte offset (from A000:0000h) is first calculated and then the bit offset into the byte is determined.
- Select write mode 2 on the graphics controller.
- Shift the bit mask into position and set the bit mask register.
- Latch the current data by reading it (4 bytes).
- Write the color out. Seven bits in four planes get values from the latches (and are immune to change). Plane 0 gets bit 0 of al, plane 1 gets bit 1 of al, etc.

```
mov     ax,80           ; bytes per scan line
mul     y[bp]           ; times row number
mov     si,x[bp]        ; column number
mov     cx,si           ; lsb's in cl
shr     si,1            ; eight pixels
shr     si,1            ;   per byte gives
shr     si,1            ;     offset into row
add     si,ax           ; plus offset to row
and     cl,00000111b    ; offset into byte

push    ds              ; save caller's ds
mov     ax,0a000h       ; regen segment
mov     ds,ax           ; into our data segment

mov     dx,3CEh         ; graphics controller
mov     ax,0200h+GMODE  ; write mode 2
outwrd  dx,ax           ; set it

mov     al,BITMASK      ; bit mask register
mov     ah,80h          ; the mask
shr     ah,cl           ; shifted into place
outwrd  dx,ax           ; set it

mov     al,[si]         ; latch old data
mov     ax,c[bp]        ; get the color
mov     [si],al         ; set it

pop     ds              ; restore caller's ds
```

6.4 Reading a Pixel

This section provides an example of how to read a pixel using the graphics controller. This is summarized as follows:

- Calculate the location of the pixel in the display buffer (video RAM). The byte offset (from A000:0000h) is first calculated and then the bit offset into the byte is determined.
- Since bh will accumulate results, put mask into bl.
- Select read mode 0 on the graphics controller.
- Read the plane (map) number into ah.
- Read a full byte from the plane number in ah.
- Make space for the new bit, test for bit "1", and if so OR it into the results.
- Loop back to pick up all the planes.
- Return the pixel value (palette register) as a word value in ax.

```
          mov     ax,80           ; bytes per scan line
          mul     y[bp]           ; times row number
          mov     si,x[bp]        ; column number
          mov     cx,si           ; lsb's in cl
          shr     si,1            ; eight pixels
          shr     si,1            ;   per byte gives
          shr     si,1            ;     offset into row
          add     si,ax           ; plus offset to row
          and     cl,00000111b    ; offset into byte
          mov     bx,80h          ; bh gets results, bl is mask

          shr     bl,cl           ; byte mask for this pixel

          mov     dx,3CEh         ; graphics controller
          mov     al,GMODE        ; graphics mode register
          xor     ah,ah           ; read mode 0
          outwrd  dx,ax           ; set read mode 0

          push    ds              ; save caller's ds
          mov     ax,0a000h       ; regen buffer segment
          mov     ds,ax           ; into our data segment

          mov     ah,3            ; start with map )plane) 3
          mov     al,READMAP      ; read map select

planes:   outwrd  dx,ax           ; select plane
          mov     cl,[si]         ; get regen byte, this plane
          shl     bh,1            ; make a spot for new bit
          test    cl,bl           ; this bit on ?
          jz      continue        ; nope
          or      bh,1            ; set result bit

continue: dec     ah              ; more pixels to read ?
          jns     planes          ; yes

          mov     al,bh           ; get result
          cbw                     ; word return value

          pop     ds              ; restore caller's ds
```

6.5 Enabling Vertical Retrace Interrupts

Sometimes events must be synchronized with vertical retrace, such as when performing a smooth pan or scroll.

- Use DOS function calls to get and set interrupt 0ah
- Two bits in one of the CRTC registers control the vertical retrace interrupt in the EGA and VGA, the others must be preserved, so look up the value in the BIOS. Note the testing for optional modes.
- Save the value above, because it will be needed in the interrupt service routine.
- Note that bit 5 of the vertical retrace control bits in CR11 is an interrupt enable. If it is set, interrupts are enabled (the interrupt output will be driven to an active state corresponding to the state of the interrupt flip-flop). If it is clear, interrupts are disabled and other devices can use IRQ2.
- Note that bit 4 of the vertical retrace control bits in CR11 is an interrupt flip-flop clear control. Clearing bit-4 holds the interrupt flip-flop in a cleared (inactive) state. Setting bit-4 enables the next vertical retrace to set the interrupt flip/flop.
- Finish with the normal PC hardware interrupt enable, i.e. 8259 interrupt controller

```
;
; get the old vector
;
     mov     sves[bp],es      ; save caller's es
     mov     ax,350ah         ; get interrupt vector
     int     21h              ; DOS function call
     mov     oldoff[bp],bx    ; save the offset
     mov     oldseg[bp],es    ; save the segment
     mov     es,sves[bp]      ; restore es
;
; set the new vector
;
     mov     sves[bp],ds      ; save caller's ds
     mov     dx,isr[bp]       ; new isr offset
     mov     ax,cs            ; new isr segment
     mov     ds,ax            ;    into ds
     mov     ah,250ah         ; set interrupt vector
     int     21h              ; DOS function call
```

```
;
; enable vertical retrace interrupt on the CRTC.  We only want to
; change two bits of the vertical retrace end register on the CRTC
; and leave the rest alone.  Since the register was write-only in the
; EGA, we look up the current value in the BIOS parameter table.
;
     mov    ax,40h            ; BIOS data segment
     mov    ds,ax             ; address BIOS data segment

     mov    si,087h           ; offset of info
     mov    bl,[si]           ; get "info" byte

     mov    cx,vmod[bp]       ; video mode
     test   bl,01100000b      ; any optional memory ?
     jz     get_parm          ; nope, 64k card
     cmp    cl,3              ; hi-res alternates ?
     ja     chk_f10           ; nope
     add    cl,13h            ; alternate tables
     jmp    short get_parm    ; get the crtc parameter
chk_f10:
     cmp    cl,0fh            ; mode f or 10 ?
     jb     get_parm          ; nope
     add    cl,2              ; use alternate mode f, 10

get_parm:
     mov    ax,type vid_prm   ; sizeof(video parm struc)
     mul    cx                ; base of our mode table

     mov    si,0a8h           ; offset of save_ptr
     lds    si,[si]           ; ds:si -> save table
     lds    si,[si]           ; ds:si -> video parameters
     add    si,ax             ; ds:si -> vid_prm, our mode
     lea    si,r_crtc[si]     ; ds:si -> crtc registers
     add    si,011h           ; ds:si -> vertical retrace
                              ; end value
     mov    ah,[si]           ; value into ah
     and    ah,00001111b      ; enable + clear
     mov    ds,sves[bp]       ; restore ds
     mov    vr_vre,ah         ; save vertical retrace
                              ; end value

     mov    dx,CRTC           ; crtc address
     mov    al,VRE            ; vertical retrace end
     or     ah,10h            ; enable vertical retrace
                              ; flip/flop
     cli                      ; don't want interrupts now
     outwrd dx,ax             ; enabled on crtc
;
; enable IRQ2 on the 8259 interrupt controller
;
     in     al,21h            ; get enabled interrupts
     mov    si,icmask[bp]     ; -> to old 8259 mask
     mov    [si],al           ; save old mask
     and    al,not 04h        ; IRQ2 enable mask
     out    21h,al            ; and back out

     sti                      ; vertical retrace enabled
```

6.6 Servicing Vertical Retrace Interrupts

Servicing the vertical retrace interrupt is a integral part of smooth scrolling and smooth panning. The following information and code example discusses proper servicing of the vertical retrace interrupt.

- The msb of status register zero is supposed to tell if it was the V7VGA that produced the interrupt, but since it is tied to IRQ2 on the bus all it shows is that an IRQ2 occurred, which we know since the code is running.
- Clearing bit 4 of the vertical retrace end register clears the interrupt condition (brings IRQ2 down).
- Clear the interrupt condition on the interrupt controller chip (8259)
- Re-enable vertical retrace interrupts by setting bit 4 of the vertical retrace end register.
- Test and set a reentrancy lock.
- Test a countdown timer.
- Establish the proper environment for the update routine and call it.
- Clear the re-entrancy lock.
- See if someone else had vertical retrace interrupts enabled.
- Return from the interrupt.

```
    push        ax              ; save ax
    push        ds              ; save ds
    mov         ax,cs:dataseg   ; value for our ds
    mov         ds,ax           ; ds -> to our data now
    assume      ds:dgroup       ; and tell the assembler


;
; verify that this is our interrupt.  A PC/AT can have
; multiple hardware devices on IRQ2 (the cascaded 8259).
;
    push    dx              ; save dx
    mov     dx,STATUS_0     ; status register zero
    in      al,dx           ; get status
    test    al,80h          ; CRTC interrupt?
                            ; This is supposed to be the bit the VGA put
                            ; onto IRQ2, but its actually IRQ2 itself,
                            ; so we'll never take this branch since this
                            ; code is running because of an IRQ2
;   jz      exit
```

```
     mov     dx,CRTC          ; crtc address
     mov     al,VRE           ; vertical retrace end
     mov     ah,vr_vre        ; get vert. retrace end value
     out     dx,ax            ; this brings IRQ2 down

     mov     al,20h           ; end of interrupt
     out     20h,al           ; to the 8259

     mov     al,VRE           ; vertical retrace end
     or      ah,10h           ; set vertical retrace f/f
     out     dx,ax            ; and interrupt enabled

     test    lock,0ffh        ; already running ?
     jnz     exit             ; yes

     dec     vr_cnt           ; count down yet ?
     jnz     exit             ; nope


     inc     lock             ; set entrancy lock
     sti                      ; enable int's on iAPX
     cld                      ; strings forward
     push    bx
     push    cx
     push    si
     push    di

     call    vr_upd

     dec     lock             ; ok to be reentered now
     pop     di
     pop     si
     pop     cx
     pop     bx
;
; see if someone else had IRQ2 enabled.
; If so, pass through to them.
;
exit:pop     dx

     test    irq_mask,04h     ; was IRQ2 enabled before?
     jnz     eoi              ; nope

     pushf
     call    old_0a           ; next isr in chain

eoi: pop     ds               ; restore interrupted ds
     pop     ax               ; and ax
     iret                     ; and return
```

6.7 Smooth Scrolling and Panning

Programming Notes:

1. The "smooth" in the following sections comes from the fact that these operations are performed under vertical retrace interrupts. Thus, they occur while the display is in vertical blanking, and at a rate that is fast enough to avoid visual perception.

2. The following sections assume text mode, such as mode 3.

3. When using either the preset row scan register or the pixel panning register, there will be more than the normal 80 by 25 characters displayed, i.e. part of the first character, 24 whole rows (or 79 whole columns), and part of the last character.

Smooth Scrolling

The preset row scan register can be used to scroll the screen by a single scan line, even though the CRT controller is in text mode. The screen can be scrolled as much as one less than the character height, i.e. 13 in high resolution text mode. Setting the preset row scan register to zero causes the display to begin with the topmost scan line of a character. Setting the preset row scan register to one causes the display to begin with the second scan line of the topmost character row and end with the topmost scan line of the 26th character row.

```
mov     dx,CRTC          ; crt controller
mov     al,PRS           ; preset row scan register
mov     ah,n[bp]         ; preset row scan value
outwrd  dx,ax            ; set it
```

Smooth Panning

```
mov     dx,STATUS        ; status one
cli                      ; don't want attr ctlr int'd
in      al,dx            ; set attr address flip/flop

mov     dx,ATTR          ; attr controller
mov     al,20h + PAN     ; enable palette + pel panning
out     dx,al            ; select panning register
mov     al,pels[bp]      ; panning value
out     dx,al            ; select panning value
sti                      ; ok to be interrupted now
```

Smooth Panning and Scrolling (Start Address High/Low)

The start address high/low in the CRTC determine the offset into the display memory buffer of the upper left corner of the display. This is how whole rows and columns of characters are scrolled.

```
mov     bx,buf[bp]   ; starting offset in bytes
shr     bx,1         ; starting address (words)

mov     dx,CRTC      ; crt controller
mov     al,LO_STRT   ; start address low
mov     ah,bl        ; lo byte of start address
outwrd  dx,ax        ; set it

mov     al,HI_STRT   ; start address high
mov     ah,bh        ; hi byte of start address
outwrd  dx,ax        ; set it
```

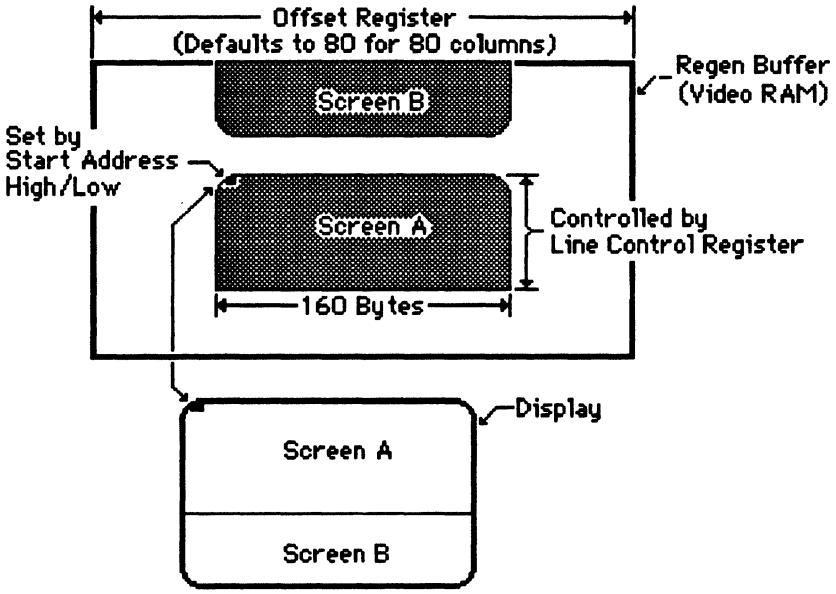


Figure 6-6. Split Screen Mapping.

When the scan line reaches the value in the line compare register, the row scan counter is cleared to zero. This the lower screen (on the display) begins at offset zero in the display memory buffer (plus any panning in effect).

```
            mov     bx,n[bp]        ; line compare value

            mov     dx,CRTC         ; crt controller
            mov     al,LCR          ; line compare register
            mov     ah,bl           ; lo byte of start address
            outwrd  dx,ax           ; set it

            mov     al,OVFL         ; overflow register
            mov     ah,0fh          ; default value
            test    bh,01h          ; more than 255 scan lines?
            jz      st_ovfl         ; nope
            or      ah,10h          ; set high bit
st_ovfl:    outwrd  dx,ax           ; out to the CRTC
```

6.9 Performance Improvement Extensions

V7VGA Extensions

V7VGA-based graphics adapters are fully compatible with the IBM VGA--and then some. The architectural enhancements and extended features of the V7VGA make it an extremely fast VGA, reducing wait states by as much as 90%. Thanks to reduced wait states, the V7VGA considerably speeds the performance of existing VGA code.

Peak performance comes with software that takes full advantage of the extended features of the V7VGA, however. These features include a hardware graphics pointer, the ability to modify selected bits in memory without a read cycle (masked writes), color text expansion capabilities, dithering support, extended underlining, and extended 256-color modes.

If you're only developing generic VGA software, you won't need to know about or use the extended features, since the V7VGA runs standard VGA software just as an IBM VGA does (except much faster). Even if this is the case, though, we suggest you read farther; you may well find that the extended features can greatly enhance your software's performance.

The baseline performance of the V7VGA--that is, the performance in VGA-compatible operation--far surpasses the IBM VGA. 16-bit memory, I/O, and ROM interfaces, a near-zero-wait-state mechanism on CPU writes, and more efficient timing states combine to improve CPU access to display memory by 350% or more in all VGA modes, with standard 120 ns D-RAMs. The V7VGA also supports V-RAMs transparently to all VGA software; with V-RAMs, CPU access to display memory is improved 700% or more over the IBM VGA.

The color and resolution capabilities of the V7VGA also far exceed the IBM VGA. 720x540 and 800x600 16-color modes are supported with 120 ns D-RAMs, 640x400 and 640x480 256-color modes are supported with 100 ns D-RAMs, and all the above modes plus 720x540 256-color and 1024x768 2- and 16-color modes are supported with V-RAMs.

We won't tell you how to program a VGA in this chapter; that's a book's worth and more. We're just going to tell you how to use the V7VGA's extended features. If you need more information about VGA programming, take a look at IBM's *Display Adapter Technical Reference*. (The same material is also found in a slightly different form in the technical reference manuals for the Model 50/60 and for the Model 80.) You might also want to look at *Programmer's Journal*, issues 5.1-5.4 and 6.1-6.2, which contain articles which cover a variety of VGA programming issues.

We will tell you how to take advantage of the V7VGA's performance-enhancing features. We'll cover: Data path enhancements, identifying the V7VGA and enabling the extended features, using masked writes, dithering, color text expansion, the hardware cursor, and 256 color mode.

A Note About 16-Bit Outs

You can use 16-bit outs--out dx,ax--if you want. The problem is that this won't work in some PC clones, notably some of those made by Olivetti and sold by AT&T. If you know your code won't end up in such a machine, go for the faster and more compact 16-bit form.

Data Path Enhancements

The data path followed from the CPU to display memory--the write data path---in the V7VGA is a superset of the IBM VGA's data path. (The read data path is functionally identical in the two chips.) Figure 6-7 below shows one plane's data path in the V7VGA.

In Figure 6-7, standard IBM VGA data paths and registers are shown with bold lines, and V7VGA extensions are shown with normal lines. (By the way, Figure 6-7 is quite useful for understanding normal VGA programming, since it shows all components of the VGA's write data path as well.) The following explanations will refer to Figure 6-7 frequently.

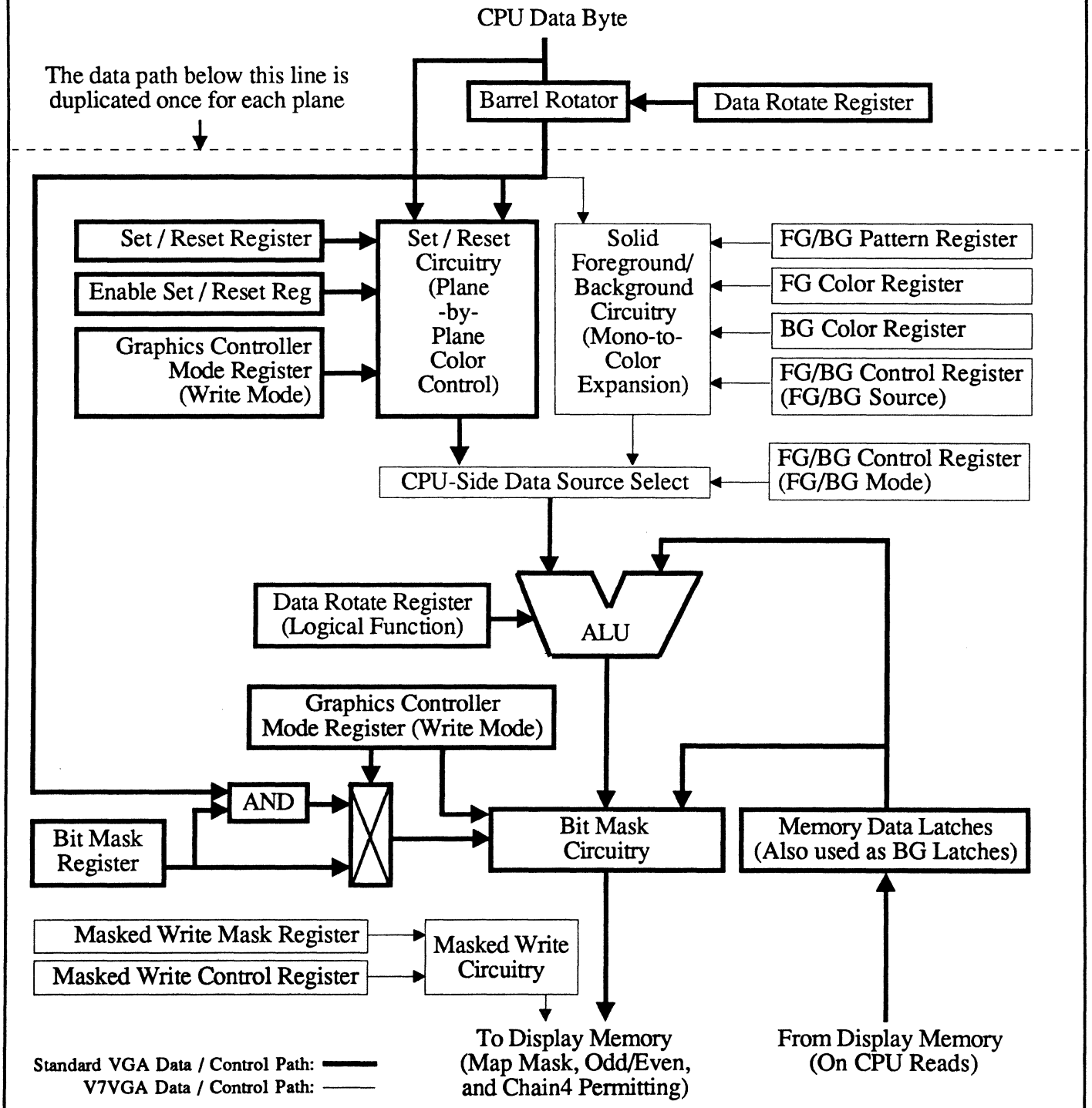


Figure 6-7. V7VGA Write Data Path

Identifying the V7VGA and Enabling Extensions

Before you can use any of the V7VGA's extended features, you must enable access to the extension registers. Before you do that, you should check to make sure you're running on a Video Seven adapter.

Positively identifying the V7VGA is a two-stage process. First, see whether you're running on a Video Seven VGA. Then, see what Video Seven chip is installed. Both of these steps can be taken with extended BIOS calls. The following function does all the necessary identification.

V7VGA Identification & Information Code

```
;
; Function to determine whether a Video Seven VGA is installed,
; and if so what its capabilities are.
;
; Input:    none
;
; Output:   AL = 1 if a Video Seven VGA is installed, 0 otherwise
;           If AL = 1:
;            AH = bits 6-0: # of 256Kb banks of RAM (1-4)
;                 bit 7: 1 if V-RAM is installed, 0 if D-RAM is installed
;            BH = chip revision (S/C chip rev # on VEGA VGA)
;            BL = chip revision (G/A chip rev # on VEGA VGA)
;            CX = 0 (reserved for future use)
;
VGA_ID PROC NEAR
       mov  ax,6f00h
       int  10h
       sub  al,al
       cmp  bx,'V7'
       jnz  Done            ;is this a Video Seven VGA?
       mov  ax,6f07h        ;yes, check what kind
       int  10h             ;get revision code info to return to user
       mov  al,1            ;mark that this is a Video Seven VGA
Done:  ret
VGA_ID ENDP
```

BIOS function 6Fh, subfunction 0, returns the character pair 'V7' in BX if a Video Seven VGA is installed.

On return from extended BIOS function 6Fh, subfunction 7, AH contains the number of 256K RAM banks installed (1-4), with bit 7 of AH set to 1 if this is a V-RAM board and 0 if this is a D-RAM board, and BH and BL both contain the revision number of the VGA chip. V7VGA chip revision numbers start at 70h, while VEGA VGA revision numbers are all greater than 80h. Practically speaking, if the chip revision number is in the range 70h-7Fh, the chip is a V7VGA. (BH and BL are both used because the VEGA VGA chipset has two chips, and hence two revision numbers).

CX is set to zero because it is "reserved for future use;" this is our way of saying, "we're going to trash CX now, so later when we decide to return some useful value in it, your code isn't going to blow up."

So, after you've called VGA_ID, you know that if AL = 1, 70h <= BL < 80h, and bit 7 of AH is 1, then you're running on a V7VGA-based graphics adapter. Once you know that, you can enable access to the extensions registers (Sequencer registers 80h-FFh, also known as ER80-ERFF) by writing the value 0EAh to Video Seven Extensions Enable register, SR6. Once access to the extensions registers is enabled, SR6 reads back as 1. You can disable access to the extensions registers by writing 0AEh to SR6. When access to the extensions registers is disabled, SR6 reads back as 0.

Here's how to turn extensions on and off:

Extension Enable & Disable Code

```
        mov   dx,3c4h
        mov   al,6
        inc   dx
        mov   al,0eah
        out   dx,al
        :
;   Extensions are now enabled.
        :
        mov   dx,3c4h
        mov   al,6
        inc   dx
        mov   al,0aeh
        out   dx,al
        :
;   Extensions are now disabled.
```

We suggest that you keep access to the extensions registers disabled as much as possible, to guard against the possibility of code accidentally writing to the extensions registers and wreaking havoc. Also, be aware that many BIOS calls turn off access to the extensions registers.

The extensions registers that you, as a graphics application/driver (as opposed to BIOS) programmer may need to use are:

94 - Pointer Pattern Address
9C - Pointer Horizontal Position High
9D - Pointer Horizontal Position Low
9E - Pointer Vertical Position High
9F - Pointer Vertical Position Low
A0 - Graphics Controller Latch Plane 0
A1 - Graphics Controller Latch Plane 1
A2 - Graphics Controller Latch Plane 2
A3 - Graphics Controller Latch Plane 3
A5 - Cursor Attributes

F1 - Fast Latch Load State
F2 - Fast Background Latch Load
F3 - Masked Write Control
F4 - Masked Write Mask
F5 - Foreground/Background Pattern
F6 - 1 Mb DRAM Bank Select
F9 - Extended Page Select
FA - Extended Foreground Color
FB - Extended Background Color
FC - Compatibility Control
FE - Foreground/Background Control

See Table 6-3 for a list of all bits in these extended registers. Remember, access to the extensions registers must be enabled by SR6 before you can read or write any of the above registers.

There are many other extended registers in the V7VGA. Don't touch them! The BIOS takes care of controlling the other extended registers, which are of no use to any other software (but which can easily scramble the screen if misprogrammed).

A final note: Please, please, please put any extensions registers you modify back to zero when your program ends. (Or, better yet, back to the state you found them in--they are readable, you know.) The BIOS does not force the state of the extensions registers on mode sets, so if you leave masked writes or color expansion on, the next program that runs is going to look mighty strange.

Masked Writes (Write-Per-Bit)

On the IBM VGA, the Bit Mask register and the latches are used to make it possible to selectively alter bits within a byte of display memory. Unfortunately, the use of the Bit Mask always requires a read as well as a write, since the latches must be loaded from the target memory byte. This not only adds instructions, but also introduces many wait states; since CPU memory access slots are spaced evenly, the second (write) access of a paired read/write operation always ends up waiting the maximum possible time.

(To put this another way, the read access suffers the average access latency, which is 1/2 the time between successive CPU memory access slots--but the write access suffers the maximum access latency, since it always occurs immediately after the CPU read access has just been completed. If this makes no sense to you, don't worry about it--just take our word for it that a read/write access takes 3 times, not 2 times, as long as a standalone write access.)

There's no way around this limitation with D-RAMs. V-RAMs, however, support the modification of selected bits within a byte *without any reads whatsoever*. This is known as a *masked write*, or *write-per-bit*, access. The V7VGA fully supports masked writes when V-RAMs are installed, with the masked write mask value coming either from the rotated CPU byte (think of this as being similar to write mode 3), or from the Masked Write Mask register (extension register F4). The Masked Write Control register (extension register F3) enables masked write operation as follows:

Extension Register F3 bit 0:	1 to enable masked writes, 0 to disable masked writes.
Extension Register F3 bit 1:	1 to select the rotated CPU byte as the source for the masked write mask 0 to select the Masked Write Masked register (extension register F4) as the source for the masked write mask.

The masked write can be used like the bit mask, to keep selected bits in the target memory byte from being changed. Here's the D-RAM code for setting bit 4 of the byte at A000:0000 to 1, leaving all other bits unchanged (without masked writes):

DRAM (Non-Masked Write) Bit Setting Code

```
mov   ax,0a000h
mov   ds,ax         ;point DS to display memory
sub   bx,bx         ;point to offset 0
mov   dx,3ceh       ;GC Index register
mov   al,8          ;index of Bit Mask register
out   dx,al         ;point GC Index to Bit Mask register
inc   dx            ;GC Data register
mov   al,010h       ;Bit Mask pattern to allow CPU to modify only bit 4
out   dx,al         ;set Bit Mask register to preserve all bits but bit 4
mov   al,[bx]       ;read memory in order to load latches
mov   [bx],0ffh     ;write to the target byte-Bit Mask register preserves
                    ; all bits but bit 4, which is set to 1
```

Here's the equivalent V-RAM code, using a masked write controlled by the Masked Write Mask register (assuming extensions are enabled):

VRAM (Masked Write Based on Masked Write Mask Reg) Bit Setting Code

```
mov   ax,0a000h
mov   ds,ax         ;point DS to display memory
sub   bx,bx         ;point to offset 0
mov   dx,3c4h       ;Sequencer Index register
mov   al,0f3h       ;index of Masked Write Control register
out   dx,al         ;point Sequencer Index to Masked Write Control register
inc   dx            ;Sequencer Data register
mov   al,1          ;bit pattern to select masked writes sourced
                    ;from Masked Write Mask register
out   dx,al         ;enable masked writes sourced from the Masked Write Mask
dec   dx            ;Sequencer Index register
mov   al,0f4h       ;index of Masked Write Mask register
out   dx,al         ;point Sequencer Index to Masked Write Mask register
inc   dx            ;Sequencer Data register
mov   al,010h       ;pattern to preserve all bits but bit 4
out   dx,al         ;set Masked Write Mask to preserve all bits but bit 4
mov   [bx],0ffh     ;write to the target register-masked write preserves
                    ; all bits but bit 4, which is set to 1. No read is needed
```

and here's the same thing with a masked write controlled by the CPU byte:

VRAM (Masked Write Based on CPU Byte) Bit Setting Code

```
mov   ax,0a000h
mov   ds,ax         ;point DS to display memory
sub   bx,bx         ;point to offset 0
mov   dx,3c4h       ;Sequencer Index register
mov   al,0f3h       ;index of Masked Write Control register
out   dx,al         ;point Sequencer Index to Masked Write Control register
inc   dx            ;Sequencer Data register
mov   al,3          ;pattern to enable masked writes, source from the CPU byte
out   dx,al         ;enable masked writes sourced from the CPU byte
mov   dx,3ceh       ;Graphics Controller Index register
mov   al,0          ;index of Set/Reset register
out   dx,al         ;point Graphics Controller Index to Set/Reset register
inc   dx            ;Graphics Controller Data
mov   al,0fh        ;pattern for color 15, white
out   dx,al         ;set set/reset color to white
dec   dx            ;Graphics Controller Index
mov   al,1          ;index of Enable Set/Reset register
out   dx,al         ;point Graphics Controller Index to Enable Set/Reset register
inc   dx            ;Graphics Controller Data
mov   al,0fh        ;pattern to enable set/reset for all planes
out   dx,al         ;enable set/reset for all planes
mov   [bx],010h     ;write to the target register-CPU byte becomes the
                    ; masked write mask & preserves all bits but bit 4,
                    ; which is set to 1. No read is needed
```

Sure, I know this doesn't look *real* convenient, but we're only changing one bit here. Once you've set up the masked write mode, and enabled Set/Reset and so forth, you can just whiz along without doing any outs at all--and without doing any of those slow, slow, slow reads.

A particularly nifty aspect of masked writes is that while they can serve the same function as the bit mask, they don't keep the bit mask from working, if you need both the bit mask and masked writes at once. This means that you can, for example, use the masked write mask for clipping and the bit mask as a selector between a foreground color and a background pattern.

Here's a scenario: Suppose you're writing red text on a background dither pattern of blue and green, and you need to clip the memory access so that only the upper nibble of the display memory byte is changed. You can set the masked write mask to 0F0h to preserve the lower nibble of display memory, fill the latches with the dither pattern (we'll cover how to do this later), put the V7VGA into write mode 3, load the Set/Reset register with red, and write the character. That single write will do everything you need: Clip, mix foreground and dithered background, and modify display memory.

Here's the code (as usual, extensions must be enabled):

Masked Write-Based Clipped Text on a Dithered Background

```
;
;     Enable masked writes, with the mask coming from the Masked Write Mask register
;
      mov   dx,3c4h
      mov   al,0fch
      out   dx,al
      inc   dx
      mov   al,1
      out   dx,al
      dec   dx
;
;     Set the Masked Write Mask reg to preserve the lower nibble of the target byte
;
      mov   al,0fdh
      out   dx,al
      inc   dx
      mov   al,0f0h
      out   dx,al
      dec   dx
;
;     Load the latches with the dither pattern
;     (note that the V7VGA lets you load the latches directly).
;
      mov   al,0f2h
      out   dx,al
      inc   dx
      in    al,dx
      mov   al,55h        ;blue plane's part of the dither pattern
      out   dx,al         ;set blue plane latch
      mov   al,0aah       ;green plane's part of the dither pattern
      out   dx,al         ;set green plane latch
      sub   al,al         ;red and intensity planes don't contribute to dither pattern
      out   dx,al         ;set red plane latch
      out   dx,al         ;set intensity plane latch
```

(continued)

```
;
;     Set to write mode 3
;
      mov   dx,3ceh
      mov   al,5
      out   dx,al
      inc   dx
      in    al,dx       ;get current Graphics Mode setting
      or    al,3        ;set the write mode field to 3
      out   dx,al       ;now we're in write mode 3
      dec   dx
;
;     Set the Set/Reset color to red
;
      mov   al,0
      out   dx,al
      inc   dx
      mov   al,4        ;red is color 4
      out   dx,al       ;the Set/Reset color is now red
;
;     Write the character
;
      mov   ax,0a000h
      mov   es,ax                       ;point ES to display memory
      mov   di,[CharacterLocation]      ;get the character's starting offset
                                        ; in display memory
      mov   si,[CharacterFont]          ;get the character's font pattern offset
      mov   cx,[CharacterHeight]        ;get the character's height in scan lines
      cld                               ;make string instructions count up

CharacterLoop:
      lodsb                             ;get the next character scan line
      stosb                             ;write the character scan line to display
                                        ; memory, with the Masked Write Mask
                                        ; clipping, the latches providing the
                                        ; background dither pattern, the Set/Reset
                                        ; register providing the text color, and
                                        ; the CPU byte selecting between foreground
                                        ; and background on a bit-by-bit basis
      add   di,SCREEN_WIDTH-1           ;point to the display memory offset for the
                                        ; next scan line of the character
      loop  CharacterLoop               ;do next character scan line
```

Once again, this seems like a lot of trouble to go to just to write a single clipped character, but it's even more trouble (and slower) to do the same thing on a standard VGA. Besides, you'll want to do a whole column of clipped characters at a time for speed, and then the masked write approach *really* flies!

This is far from the only way to mix dithers, colors, and clips. If you spend some time understanding Figure 6-7, you'll see that Bullet's data path provides a remarkably rich set of possible data sources and combinations.

Dithering

Dithering is the process of drawing with a pattern made of pixels of two or more colors, in order to approximate a color that the hardware does not support. Dither patterns are often used as backgrounds for windows and other objects, and sometimes for foregrounds (such as halftone text) as well.

The problem with dither patterns on a normal VGA is that, by definition, a dither pattern must be a full 32 bits wide--that is, 8 bits per plane, so each of the 8 pixels at a given address can be any color. Unfortunately, this means that there's no way to write an arbitrary dither pattern directly from the CPU (although a two-color dither pattern can be controlled directly by the CPU using the color text expansion feature of the V7VGA, discussed below).

On the other hand, it ***is*** possible to write a full 32-bit (8x4) dither pattern from the latches, since there's an 8-bit latch for each plane. Getting information into the latches is a real bear on a standard VGA, though, since there's no way for the CPU to write directly to the latches. You have to write each plane's dither pattern separately to an off-screen address, then read the address to latch the pattern.

The V7VGA's latches are (hallelujah!) directly readable and writable, at extension registers A0-A3. The latches can be accessed just like any other extension register. Better yet, all four latches can be loaded at a single I/O port, the Fast Background Latch Load register (extension register F2).

The way the Fast Background Latch register works is this: Each write to the Fast Background Latch register goes to the latch for the plane selected by bits 1 and 0 of the Fast Latch Load State register (extension register F1). If bits 1 & 0 of ERF1 are 0, the plane 0 (blue plane) latch is loaded, if bits 1 & 0 are 1, the plane 1 (red plane) latch is loaded, and so on. Each time the Fast Background Latch register is written to, the field made up of bits 1 & 0 of the Fast Latch Load State register is incremented by 1, modulo 4. When you ***read*** the Fast Background Latch register, bits 1 & 0 of the Fast Latch Load State register are reset to 0.

The normal way to load a dither pattern is to read the Fast Background Latch register, then write the blue, green, red, and intensity plane dither bytes to the Fast Background Latch register in succession. Here's a function to load the dither pattern in CX:BX on the IBM VGA:

IBM VGA Dither Pattern Load

```
;     Load a dither pattern.
;
;     Input: CH = intensity plane dither byte
;              CL = red plane dither byte
;              BH = green plane dither byte
;              BL = blue plane dither byte
;
;     Output:  none
;
LoadDither  proc        near
     push   ds
     mov    ax,0a000h
     mov    ds,ax       ;point DS to display memory
     mov    si,0ffffh   ;point to the last byte of display memory
     mov    dx,3c4h     ; (which is presumably unused)
     mov    al,02h
     out    dx,al
     inc    dx
     mov    al,1        ;map mask pattern to select blue plane only
     out    dx,al       ;set map mask to allow writes to blue plane only
     mov    [si],bl     ;write blue plane dither byte to blue plane
     mov    al,2        ;map mask pattern to select green plane only
     out    dx,al       ;set map mask to allow writes to green plane only
     mov    [si],bh     ;write red plane dither byte to green plane
```

```
      mov    al,4          ;map mask pattern to select red plane only
      out    dx,al         ;set map mask to allow writes to red plane only
      mov    [si],cl       ;write red plane dither byte to red plane
      mov    al,8          ;map mask pattern to select intensity plane only
      out    dx,al         ;set map mask to allow writes to intensity plane only
      mov    [si],ch       ;write intensity plane dither byte to intensity plane
      mov    al,[bx]       ;load the dither pattern into the latches
      pop    ds
      ret
LoadDither  endp
```

and here's the code to do the same thing on the V7VGA, using the fast latch load capability:

V7VGA Dither Pattern Load

```
;     Load a dither pattern
;
;     Input: CH = intensity plane dither byte
;            CL = red plane dither byte
;            BH = green plane dither byte
;            BL = blue plane dither byte
;
;     Output:  none
;
;     Assumes extensions are enabled.
;
LoadDither  proc  near
      mov  dx,3c4h
      mov  al,0f2h
      out  dx,al  ;point Sequencer Index to Fast Background Latch Load register
      inc  dx
      in   al,dx  ;set background latch load state to select latch 0
      mov  al,bl  ;get blue plane dither byte
      out  dx,al  ;set blue plane dither byte
      mov  al,bh  ;get green plane dither byte
      out  dx,al  ;set green plane dither byte
      mov  al,cl  ;get red plane dither byte
      out  dx,al  ;set red plane dither byte
      mov  al,ch  ;get intensity plane dither byte
      out  dx,al  ;set intensity plane dither byte
      ret
LoadDither  endp
```

Enough said. (Well, almost enough--note that because the V7VGA version doesn't access memory, no wait states are incurred, while wait states occur on each of the five memory accesses in the standard VGA version.)

Color Text Expansion

One of the most annoying tasks on a standard VGA is simply turning a binary font pattern into colored text on an opaque, colored background. Most software handles this case by first drawing the character box with the background color, then drawing the text on top of the new background. This causes flicker and requires three memory accesses per byte of font pattern: One write to draw the background and a read/write pair to load the background into the latches, then mix in the foreground (text) pixels and write the final result to memory.

Astute programmers may immediately notice that we could load the background color into the latches, just as we did with the dither pattern in the last example, set the set/reset color to the desired foreground color and enable set/reset for all planes, and use write mode 3, and then we'd only have to do one write per font byte. Very ingenious...but the V7VGA has a better solution yet.

The V7VGA can expand 1-bits in either the CPU byte or the Foreground / Background Pattern (extension register F5) to one color, and 0-bits to another color. It's sort of like having one set/reset color for 1 bits and another set/reset color for 0 bits. The result: You can produce colored text on an opaque, colored background with a single write--and you can change either or both of the colors very easily.

Color text expansion is controlled by the Foreground/Background Control register (extension register FE). The bits in this register are:

Bits 3 & 2: 00 = normal IBM VGA operation
01 = color expansion mode
10 = reserved
11 = reserved

Bit 1: 1 = select the rotated CPU byte as the data to be expanded
0 = select the contents of the Foreground / Background register as the data to be expanded

Take another look at Figure 6-7, paying special attention to the foreground / background section toward the upper right. The box labeled "CPU-Side Data Source Select Circuitry" selects between the output of the set/reset circuitry (which may be the rotated CPU data byte, the set/reset value, or a byte-expanded bit of the CPU data, depending on the mode) and the output of the foreground / background circuitry (which for each bit is either the foreground color--for 1 bits--or the background color--for 0 bits). When bits 3 & 2 of extension register FE are 00, the set/reset output is selected; when bits 3 & 2 are 01, the foreground / background output is selected.

Fine. Now, where do the foreground and background colors come from, and how is one or the other selected? That's easy enough: The foreground color is stored in the Extended Foreground Color register (extension register FA), and the background color is stored in the Extended Background Color register (extension register FB). These two colors are selected between on a bit-by-bit basis by a byte from one of two sources: The rotated CPU byte or the Foreground / Background Pattern register (extension register F5). 1 bits in the selector byte choose the foreground color, and 0 bits choose the background color.

Here's code that draws yellow characters on a blue background, using the V7VGA's color expansion capabilities (assuming access to the extension registers is enabled):

An Example of Color Expansion

```
;       Set to color expansion mode
;
        mov     dx,3c4h
        mov     al,0feh
        out     dx,al
        inc     dx
        mov     al,6
```

```
        out     dx,al           ;enable color expansion mode, sourced by the CPU byte
        dec     dx
;
;       Set the foreground color to yellow
;
        mov     al,0fah
        out     dx,al
        inc     dx
        mov     al,14
        out     dx,al
        dec     dx
;
;       Set the background color to blue
;
        mov     al,0fbh
        out     dx,al
        inc     dx
        mov     al,1
        out     dx,al
;
;       Write the character
;
        mov     ax,0a000h
        mov     es,ax                   ;point ES to display memory
        mov     di,[CharacterLocation]  ;get the character's starting offset in display memory
        mov     si,[CharacterFont]      ;get the character's font pattern offset
        mov     cx,[CharacterHeight]    ;get the character's height in scan lines
        cld                             ;make string instructions count up

CharacterLoop:
        lodsb                           ;get the next character scan line
        stosb                           ;write the character scan line to display memory,
                                        ; with 1 bits converted to yellow and 0 bits converted to blue.
        add     di,SCREEN_WIDTH-1       ;point to display memory offset for next scan line of character
        loop    CharacterLoop           ;do next character scan line
```

As always, masked writes are still available to clip the data written to display memory; in fact, the bit mask is available should you need to mix the foreground / background output with yet a third color or even a dither pattern. You could, for example, put a pattern in the Foreground / Background Pattern register, and set up the foreground and background color registers to the two colors you want associated with that pattern. Then you could put a dither pattern in the latches (don't forget that neat fast latch load capability!), put the V7VGA in write mode 3, and use the rotated CPU byte to select the mixing of the two colors from the color expansion with the dither pattern; in fact, since in write mode 3 the rotated CPU byte is ANDed with the Bit Mask register before serving as the bit mask, you could set the Bit Mask register to further adjust the mixing. Finally, you can use the Masked Write Mask register to clip the final result.

Or, you could route the CPU byte to the masked write circuitry, and use the Bit Mask register to control the dither mixing with the color expansion. Or, you could route the CPU byte to the color expansion...or you could use write mode 2 and route the CPU byte through the set/reset; and or course you can use set/reset too, if you wish. And don't forget that you can rotate the CPU byte.

The point? The V7VGA's data paths give you incredible data selection and mixing capabilities. The CPU data byte can go to five different places in the data path, performing a different function in each place. Whatever you want to do with color expansion, color mixing, rotation, and clipping, odds are you can come up with a clever way to get the V7VGA's data path to do it.

Extended Underlining

One nuisance with the IBM VGA is that only text with attribute X000X001 can be underlined; this makes it impossible to support bold, blinking, underlined text (that is, text with arbitrary attributes). The V7VGA solves this by letting you store a specific underline pattern for each character in plane 3, the intensity plane, which is normally unused in text mode. This feature is known as extended attributes, although it might just as well be called extended underlining. (It's called extended attributes instead of extended underlining because the underline pattern resides at the same address as the normal attribute, but in plane 3, and because any texture of underline, rather than just solid, is supported.

Here's how extended attributes work. Normally, a character and its attribute are fetched from a given address in planes 0 & 1, and the character is used to look up font data in plane 2. However, you can enable extended attributes by setting bit 0 of extension register FC (the Compatibility Control register) to 1. Now, when each character and attribute is fetched from planes 0 and 1, the byte at the same address is fetched from plane 3. If the current row scan (character scan line within the character row) is the underline scan line, as selected by CRT Controller register 14, then the byte fetched from plane 3 is used as the font data in place of the byte looked up in the font plane. Basically, this means that at the underline scan line, the byte at the character's address in plane 3 provides the video data.

This means that you can solidly underline a character by putting 0FFh at the same address as the character's attribute, but in plane 3; you can also get a dotted underline by putting 55h in plane 3, or a half-underline by putting 0Fh in plane 3. Whatever 8-bit pattern you want to see for an underline for a given character, you can get it by putting that pattern in plane 3. (By the way, extended attribute pixels show up in the foreground and background colors specified by the normal attribute, just like any other character data.)

How do you write to plane 3? Just set the Map Mask register to 08h, and then write to the same address as you would write to if you were changing the attribute of the character that you're underlining. The Map Mask setting of 8 will make sure that the write ends up in plane 3. When you're done modifying the extended attributes in plane 3, put the Map Mask back to a value of 3 (or, better yet, its original value, which you read out before starting all this. Didn't you?)

Important safety tip: Always write to odd locations in plane 3 only, and only set the Map Mask to 8, not 0Ch or anything like that. Otherwise, you could wind up trashing your fonts, which is not a good idea. Also--you can't use extended attributes when you're in V-RAM 1:4 text mode (132-column on a 31.5 KHz monitor), because that mode requires the font to be duplicated in plane 3.

When extended attributes are enabled, normal underlining is disabled, so the fabled X000X001 attribute just gives you some sort of blue, and nothing more.

I realize that all this may not be 100% clear, so here's an example. (Not a very practical example, but it does illustrate all the actions needed to use extended attributes.)

An Example of Extended Attributes

```
;     Enables extended attributes, then puts a green "A" at column 0, row 0,
;     with a dashed underline.
;
;     Assumes extensions are enabled and that mode 3 is set.
;
;     Set the underline scan line to 8,
;     so we can see the extended attribute underline.
;
      mov   dx,3d4h       ;CRTC Index
      mov   al,14h        ;Underline register index
      out   dx,al         ;point CRTC Index to Compatibility Control register
      inc   dx            ;CRTC Data
      mov   al,8          ;underline setting of scan line 8
```

```
    out   dx,al       ;set underline to show up at scan line 8
;
;   Enable extended attributes
;
    mov   dx,3c4h     ;Sequencer Index
    mov   al,0fch     ;Compatibility Control register index
    out   dx,al       ;point Sequencer Index to Compatibility Control register
    inc   dx          ;Sequencer Data
    in    al,dx       ;get extension register FC setting
    or    al,1        ;set extended attributes enable bit
    out   dx,al       ;turn on extended attributes
;
;   Write the character & attribute
;
    mov   ax,0b800h
    mov   ds,ax       ;point to display memory (mode 3)
    mov   bx,0
    mov   al,'A'
    mov   [bx],al     ;write character "A"
    inc   bx
    mov   al,2        ;color green
    mov   [bx],al     ;set attribute for this "A" to green
;
;   Write the extended attribute of dashed underline
;
    mov   dx,3c4h     ;Sequencer Index
    mov   al,2        :Map Mask register index
    out   dx,al       ;point Sequencer Index to Map Mask register
    inc   dx          ;Sequencer Data
    in    al,dx       ;get Map Mask setting
    push  ax          ;save Map Mask setting to restore it later
    mov   al,08h      ;Map Mask setting to enable only plane 3
    out   dx,al       ;set Map Mask to allow writes only to plane 3
    mov   al,55h      ;dashed underline pattern
    mov   [bx],al     ;write dashed underline extended attribute
    pop   ax          ;get back original Map Mask setting
    out   dx,al       ;restore original Map Mask setting
```

Extended 256-Color Modes

To set a pixel in the 256 color graphics modes the software must compute the address of that pixel. For the 640x400, 640x480, and 720x540 resolution 256-color modes video memory is greater than 64 KBytes. The V7VGA allows the software to access the video memory in 64 KByte banks at A000:0. In order to get the proper 64K bank into the memory window at A000:0 the software must set the bank. The bank is usually determined by computing a 19-bit address into video memory and using the 3 msb's as the bank select. The 640x400 256-color mode only requires the use of 2 bits for the bank select, so some efficiency can be gained by ignoring the 19th bit.

These bank bits currently translate into the following register locations:

bit 0 - Extended page select bit - bit 0 of extension register F9
bit 1 - Page select bit - bit 5 of the Miscellaneous Output Register
bit 2 - CPU bank select bit 0 - Extension register F6 bit-0 (W) and bit-2 (R)

The following code illustrates what must be done to set the correct bank:

```
;
;       Set Bank for 256 color modes
;
;       Routine: set_256_bank
;       Entry:   DX = bank to set and DS = CS.
;       Exit:    Correct bank set, AX,BX,DX are
;                destroyed. SC_INDEX has been changed.
;
;       Assume:  Extensions are enabled.  The active page
;                variable has been properly initialized
;                with a call to this routine while dx = 0.
;
active_page     db      1
SC_INDEX        equ     3c4h
ER_PAGE_SEL     equ     0f9h
MISC_INPUT      equ     3cch
MISC_OUTPUT     equ     3c2h
ER_BANK_SEL     equ     0f6h
```

(continued on following page)

```
set_256_bank    proc    near
      push      ds

      cmp       active_page,dl          ;Is this the currently selected bank?
      je        sp_ext_256_support_10   ;yes, do nothing.

      mov       active_page,dl          ;no, set bank.
      mov       bl,dl

      mov       dx,SC_INDEX             ;bank bit 0
      mov       ah,bl
      and       ah,1
      mov       al,ER_PAGE_SEL
      out       dx,ax

      mov       ah,bl                   ;bank bit 1
      and       ah,2
      shl       ah,1
      shl       ah,1
      shl       ah,1
      shl       ah,1
      mov       dx,MISC_INPUT
      in        al,dx
      and       al, not 20h
      mov       dx,MISC_OUTPUT
      or        al,ah
      out       dx,al

IFDEF  3_BANK_BITS
      mov       dx,SC_INDEX             ;bank bit 2.
      mov       al,ER_BANK_SEL
      out       dx,al                   ;get ER_BANK_SEL register
      inc       dx
      in        al,dx
      mov       ah,al
; duplicate bit 2 into bit 0 (set read and write bank equal)
      shr       bl,1
      shr       bl,1
      add       bl,7
      not       bl
      and       bl,5
      and       ah,0f0h                 ;clear bank select bits
      or        ah,bl
      mov       al,ah
      out       dx,al
ENDIF

sp_ext_256_support_10:
      pop       ds
      ret
set_256_bank    endp
```

Hardware Graphics Cursor (Pointer)

This section shows how to program the V7VGA hardware cursor (also called the pointer). Routines are shown to turn the pointer on and off, read and write the current pointer pattern number (up to 256 patterns may be stored simultaneously in display memory), read and write the current pointer XY screen position, and load pointer patterns to display memory. In addition, a number of example pointer data patterns are listed at the end of the pointer pattern load routine.

Note that the pointer XY position on the screen is the position of the upper left corner of the pointer pattern. If the program requires that the pointer be located partially off the screen to the left or at the top, the programmer must adjust the pointer pattern in memory accordingly. No code is given in this section to handle these cases (this is left as an exercise for the reader). Since the XY position indicates the upper left of the pointer pattern, the V7VGA chip automatically handles cases where the cursor is located partially off the screen at the bottom or at the right side.

The following definitions are used in the code examples:

```
;
;       Display Memory Definitions
;
RAMSEG  EQU     0A000H          ;Display memory start address
RAMSIZ  EQU     64*1024         ;Display memory size
;
;       3Cx I/O Port Address Definitions
;
ATR     EQU     03C0H           ;Attribute controller
FEAT    EQU     03C2H           ;Feature read register
MISC    EQU     03C2H           ;Misc output register write
SEQ     EQU     03C4H           ;Sequencer index reg (seq data is SEQ+1)
;       EQU     03C6H           ;Color Palette
;       EQU     03C8H           ;Color Palette
FCRD    EQU     03CAH           ;Feature Control register read
MSRD    EQU     03CCH           ;Misc output register read
GRC     EQU     03CEH           ;Graphics Controller
;
;       V7VGA Extension Register Index Definitions
;
GRL0    EQU     0A0H            ;Graphics Controller Data Latch 0
GRL1    EQU     0A1H            ;Graphics Controller Data Latch 1
GRL2    EQU     0A2H            ;Graphics Controller Data Latch 2
GRL3    EQU     0A3H            ;Graphics Controller Data Latch 3

CURS    EQU     0A5H            ;Cursor Attributes
PPA     EQU     094H            ;Pointer Pattern Address

PXH     EQU     09CH            ;Pointer X Position High
PXL     EQU     09DH            ;Pointer X Position Low
PYH     EQU     09EH            ;Pointer Y Position High
PYL     EQU     09FH            ;Pointer Y Position Low
```

The following code sequence should be executed before accessing any of the extension registers to enable access to the extension registers:

```
        ...
        MOV     DX,SEQ
        MOV     AL,6
        OUT     DX,AL           ;Point at SR6
        INC     DL
        MOV     AL,0EAH
        OUT     DX,AL           ;Enable access to extensions
        ...
```

The following routine turns the pointer on so that it will be visible on the screen:

```
PTRON   PROC    NEAR
        MOV     DX,SEQ
        MOV     AL,CURS
        OUT     DX,AL           ;Point at the Cursor extension register
        INC     DL
        IN      AL,DX           ;Read the current contents
        OR      AL,80H          ;Turn the pointer on
        OUT     DX,AL           ;Write it back
        RET
PTRON   ENDP
```

The following routine turns the pointer off so that it will not be visible on the screen:

```
PTROFF  PROC    NEAR
        MOV     DX,SEQ
        MOV     AL,CURS
        OUT     DX,AL           ;Point at the Cursor extension register
        INC     DL
        IN      AL,DX           ;Read the current contents
        AND     AL,7FH          ;Turn the pointer off
        OUT     DX,AL           ;Write it back
        RET
PTROFF  ENDP
```

The following code reads the current pointer number and returns with it in AL:

```
GETPPA  PROC    NEAR
        MOV     DX,SEQ
        MOV     AL,PPA
        OUT     DX,AL           ;Point at Pointer Pattern Address register
        INC     DL
        IN      AL,DX           ;Read current pointer pattern number
        RET
GETPPA  ENDP
```

The following code loads a new pointer number from a value in AH. This should be a valid pointer number corresponding to a pattern that has been loaded into display memory (or will be before the pointer is enabled). Valid pointer numbers for the example pointer load routine at the end of this section are between 'MAXCNT' (255) and 'MAXCNT-PTRCNT' (PTRCNT is the number of pointers loaded at the end of display memory).

```
GETPPA  PROC    NEAR
        MOV     DX,SEQ
        MOV     AL,PPA
        OUT     DX,AL           ;Point at Pointer Pattern Address register
        INC     DL
        MOV     AL,AH           ;Get pointer pattern number
NXTPAT: OUT     DX,AL           ;Load it
        RET
GETPPA  ENDP
```

The following routine reads the current location of the pointer on the screen:

```
;-------------------------------------------------------------------------
;
;       Get Pointer Position
;
;       Call with:      AL = Start Index (PXH)
;                       DX = I/O Address of Sequencer Index Reg
;
;       Returns with:   BX = X Position
;                       CX = Y Position
;                       AX,DX preserved
;
GETPTR  PROC    NEAR
        PUSH    AX

        OUT     DX,AL           ;Point at XH
        INC     DL              ;Bump I/O address
        MOV     AH,AL           ;Save for next reg
        IN      AL,DX           ;Program XH
        MOV     BH,AL           ;Save XH value
        DEC     DL              ;Reset I/O address to index reg

        INC     AH              ;Bump index
        MOV     AL,AH           ;Get index
        OUT     DX,AL           ;Point at XL
        INC     DL              ;Bump I/O address
        IN      AL,DX           ;Program XL
        MOV     BL,AL           ;Save XL value
        DEC     DL              ;Reset I/O address to index reg

        INC     AH              ;Bump index
        MOV     AL,AH           ;Get index
        OUT     DX,AL           ;Point at YH
        INC     DL              ;Bump I/O address
        IN      AL,DX           ;Program YH
        MOV     CH,AL           ;Save YH value
        DEC     DL              ;Reset I/O address to index reg

        INC     AH              ;Bump index
        MOV     AL,AH           ;Get index
        OUT     DX,AL           ;Point at YL
        INC     DL              ;Bump I/O address
        IN      AL,DX           ;Program YL
        MOV     CL,AL           ;Save YL value
        DEC     DL              ;Reset I/O address to index reg

        POP     AX
        RET
GETPTR  ENDP
```

The following routine updates the current location of the pointer on the screen:

```
;-------------------------------------------------------------------------
;
;       Update Pointer Position
;
;       Call with:      AL = Start Index (PXH)
;                       BX = X Position
;                       CX = Y Position
;                       DX = I/O Address of Sequencer Index Reg
;
;       Returns with:   All registers preserved
;
PUTPTR  PROC    NEAR
        PUSH    AX

        OUT     DX,AL           ;Point at XH
        INC     DL              ;Bump I/O address
        MOV     AH,AL           ;Save for next reg
        MOV     AL,BH           ;Get XH value
        OUT     DX,AL           ;Program XH
        DEC     DL              ;Reset I/O address to index reg

        INC     AH              ;Bump index
        MOV     AL,AH           ;Get index
        OUT     DX,AL           ;Point at XL
        INC     DL              ;Bump I/O address
        MOV     AL,BL           ;Get XL value
        OUT     DX,AL           ;Program XL
        DEC     DL              ;Reset I/O address to index reg

        INC     AH              ;Bump index
        MOV     AL,AH           ;Get index
        OUT     DX,AL           ;Point at YH
        INC     DL              ;Bump I/O address
        MOV     AL,CH           ;Get YH value
        OUT     DX,AL           ;Program YH
        DEC     DL              ;Reset I/O address to index reg

        INC     AH              ;Bump index
        MOV     AL,AH           ;Get index
        OUT     DX,AL           ;Point at YL
        INC     DL              ;Bump I/O address
        MOV     AL,CL           ;Get YL value
        OUT     DX,AL           ;Program YL & update screen position
        DEC     DL              ;Reset I/O address to index reg

        POP     AX
        RET
PUTPTR  ENDP
```

For example, a code sequence to move the pointer right one pixel would be:

```
        ...
        MOV     DX,SEQ          ;Point at sequencer index register
        MOV     AL,PXH          ;Point at pointer position register group
        CALL    GETPTR          ;Get current X and Y location
        INC     BX              ;Bump X to move the pointer right one pixel
        CALL    PUTPTR          ;Load new location
        ...
```

All four position registers should be updated as a group as the write of the last register of the group actually updates the position on the screen. The V7VGA chip pipelines the actual screen update like this so that the cursor does not appear at spurious positions on the screen during the update sequence.

The next three pages list a routine that may be used to load the pointer patterns into display memory. The first page (this page) is the start of the routine and shows code required to save the appropriate V7VGA registers and reprogram them appropriately for the actual pointer load routine which is listed on the next page. The page after the pointer load code lists the code required to put everything back the way it was before the routine is called.

The seven pages following the code for the routine list the data files used by the routine (a number of example pointer patterns).

This routine will work completely independent of what mode the V7VGA is in (text, graphics, monochrome, 4-plane color, byte-per-pixel 256-color, etc, makes no difference).

```
;-------------------------------------------------------------------------
;
;       Load Pointer Patterns to Memory
;
;       Entry:  none
;       Exit:   AX, DX modified
;
LOADPTR PROC    NEAR
        MOV     DX,GRC
        MOV     AL,6
        OUT     DX,AL           ;Point at GR6
        INC     DL
        IN      AL,DX           ;Read current value
        PUSH    AX              ;Save for later
        AND     AL,1            ;Mask all except text/graphics bit
        OR      AL,04H          ;No chaining, mem map = A0000-AFFFF
        OUT     DX,AL           ;and put it back

        MOV     DX,GRC
        MOV     AL,5
        OUT     DX,AL           ;Point at GR5
        INC     DL
        IN      AL,DX           ;Read current value
        PUSH    AX              ;Save for later
        MOV     AL,1            ;Turn off odd/even bit & set to write mode 1
        OUT     DX,AL           ;and put it back

        MOV     DX,SEQ
        MOV     AL,4
        OUT     DX,AL           ;Point at SR4
        INC     DL
        IN      AL,DX           ;Read current value
        PUSH    AX              ;Save for later
        MOV     AL,7            ;Turn off odd/even and double odd/even
        OUT     DX,AL           ;and put it back

        MOV     DX,SEQ
        MOV     AL,2
        OUT     DX,AL           ;Point at SR2
        INC     DL
        IN      AL,DX           ;Read current value
        PUSH    AX              ;Save for later
        MOV     AL,0FH
        OUT     DX,AL           ;Set to write all planes
```

Now that the appropriate registers are saved away on the stack and everything set up for the pointer load, the following code is used to actually load the pointer patterns into display memory. The pointer patterns and a pointer table used to load the patterns are listed later in this section.

Note: The pointer patterns are loaded at the end of display memory (PPA register values starting from 255 and below). The routine loads pointer data until all pointers are loaded, then exits. So the pointers take up as much display memory as there are pointers in the table, starting from the end of display memory. More pointers can be added to or deleted from the table without changing any of the code. Also, in order to make the data patterns easy to edit, the data is entered as binary formatted double words. This places the data bytes in the 'wrong' order in memory, so the following code just loads them in the reverse order, starting with byte 3 instead of byte 0.

```
;
;       Now load the pointers
;
        MOV     AX,RAMSEG
        MOV     ES,AX           ;Point at display memory
        MOV     AH,256-PTRCNT   ;Calculate where to start in display memory
        MOV     AL,0
        SAR     AX,1
        SAR     AX,1
        MOV     DI,AX           ;Initialize display memory index

        MOV     SI,OFFSET PTRS  ;Pointer Table Index
        MOV     CH,PTRCNT       ;This many pointers

LPOLUP: MOV     CL,64           ;64 * 4 (256) bytes per pointer

LPLOOP: MOV     AL,GRL3
        CALL    LDGRL           ;Load data latch 3
        MOV     AL,GRL2
        CALL    LDGRL           ;Load data latch 2
        MOV     AL,GRL1
        CALL    LDGRL           ;Load data latch 1
        MOV     AL,GRL0
        CALL    LDGRL           ;Load data latch 0

        MOV     ES:[DI],AL      ;Write 4 bytes to display memory

        INC     DI              ;Bump display memory index
        DEC     CL              ;Decrement pointer byte count
        JNZ     LPLOOP          ;Loop over this pointer

        DEC     CH              ;Decrement pointer count
        JNZ     LPOLUP          ;Loop over all pointers

        JMP     SHORT LDEXIT    ;Done
;-------------------------------------------------------------------------------
;
;       Support Subroutine for Pointer Load
;
LDGRL   PROC    NEAR
        MOV     DX,SEQ
        OUT     DX,AL
        INC     DX
        MOV     AL,DS:[SI]      ;Get pointer byte
        OUT     DX,AL           ;Write to data latch
        INC     SI
        RET
LDGRL   ENDP
```

```
;
;       Restore original data values
;
LDEXIT: MOV     DX,SEQ
        MOV     AL,2
        OUT     DX,AL           ;Point at SR2
        INC     DL
        POP     AX
        OUT     DX,AL           ;Restore original value

        MOV     DX,SEQ
        MOV     AL,4
        OUT     DX,AL           ;Point at SR4
        INC     DL
        POP     AX
        OUT     DX,AL           ;Restore original value

        MOV     DX,GRC
        MOV     AL,5
        OUT     DX,AL           ;Point at GR5
        INC     DL
        POP     AX
        OUT     DX,AL           ;Restore original value

        MOV     DX,GRC
        MOV     AL,6
        OUT     DX,AL           ;Point at GR6
        INC     DL
        POP     AX
        OUT     DX,AL           ;Restore original value
        RET

LOADPTR ENDP
```

```
;       Graphics Cursor (Pointer) Table
;
PTRTAB: DW      OFFSET ARROW    ;Type 00 - Arrow         (16x16)
        DW      OFFSET BOX      ;Type 01 - Box           (32x32)
        DW      OFFSET BOX2     ;Type 02 - Box #2        (32x32)
        DW      OFFSET CROSS    ;Type 03 - Cross         (32x32)
        DW      OFFSET INVBLK   ;Type 04 - Inverse Block (32x32)
        DW      OFFSET WHTBLK   ;Type 05 - White   Block (32x32)
        DW      OFFSET BLKBLK   ;Type 06 - Black   Block (32x32)
PTRSIZ  EQU     ($-PTRTAB)/2
PTRS:
;       Graphics Cursor Pattern:  Arrow (black with white border)
;
ARROW:  DD      00111111111111111111111111111111b       ;AND 00  AND XOR Result
        DD      00011111111111111111111111111111b       ;AND 01  --- --- ------
        DD      00001111111111111111111111111111b       ;AND 02   0   0  Black
        DD      00000111111111111111111111111111b       ;AND 03   0   1  White
        DD      00000011111111111111111111111111b       ;AND 04   1   0  Clear
        DD      00000001111111111111111111111111b       ;AND 05   1   1  Invers
        DD      00000000111111111111111111111111b       ;AND 06
        DD      00000000011111111111111111111111b       ;AND 07
        DD      00000000001111111111111111111111b       ;AND 08
        DD      00000000000111111111111111111111b       ;AND 09
        DD      00000001111111111111111111111111b       ;AND 10
        DD      00010000111111111111111111111111b       ;AND 11
        DD      00110000111111111111111111111111b       ;AND 12
        DD      01111000011111111111111111111111b       ;AND 13
        DD      11111000011111111111111111111111b       ;AND 14
        DD      11111100011111111111111111111111b       ;AND 15
        DD      11111111111111111111111111111111b       ;AND 16
        DD      11111111111111111111111111111111b       ;AND 17
        DD      11111111111111111111111111111111b       ;AND 18
        DD      11111111111111111111111111111111b       ;AND 19
        DD      11111111111111111111111111111111b       ;AND 20
        DD      11111111111111111111111111111111b       ;AND 21
        DD      11111111111111111111111111111111b       ;AND 22
        DD      11111111111111111111111111111111b       ;AND 23
        DD      11111111111111111111111111111111b       ;AND 24
        DD      11111111111111111111111111111111b       ;AND 25
        DD      11111111111111111111111111111111b       ;AND 26
        DD      11111111111111111111111111111111b       ;AND 27
        DD      11111111111111111111111111111111b       ;AND 28
        DD      11111111111111111111111111111111b       ;AND 29
        DD      11111111111111111111111111111111b       ;AND 30
        DD      11111111111111111111111111111111b       ;AND 31
ARROWX: DD      11000000000000000000000000000000b       ;XOR 00  AND XOR Result
        DD      10100000000000000000000000000000b       ;XOR 01  --- --- ------
        DD      10010000000000000000000000000000b       ;XOR 02   0   0  Black
        DD      10001000000000000000000000000000b       ;XOR 03   0   1  White
        DD      10000100000000000000000000000000b       ;XOR 04   1   0  Clear
        DD      10000010000000000000000000000000b       ;XOR 05   1   1  Revers
        DD      10000001000000000000000000000000b       ;XOR 06
        DD      10000000100000000000000000000000b       ;XOR 07
        DD      10000000010000000000000000000000b       ;XOR 08
        DD      10000001111000000000000000000000b       ;XOR 09
        DD      10001001000000000000000000000000b       ;XOR 10
        DD      10010010010000000000000000000000b       ;XOR 11
        DD      10100100100000000000000000000000b       ;XOR 12
        DD      11000010010000000000000000000000b       ;XOR 13
        DD      10000010010000000000000000000000b       ;XOR 14
        DD      00000001110000000000000000000000b       ;XOR 15
        DD      00000000000000000000000000000000b       ;XOR 16
        DD      00000000000000000000000000000000b       ;XOR 17
        DD      00000000000000000000000000000000b       ;XOR 18
        DD      00000000000000000000000000000000b       ;XOR 19
        DD      00000000000000000000000000000000b       ;XOR 20
        DD      00000000000000000000000000000000b       ;XOR 21
        DD      00000000000000000000000000000000b       ;XOR 22
        DD      00000000000000000000000000000000b       ;XOR 23
        DD      00000000000000000000000000000000b       ;XOR 24
        DD      00000000000000000000000000000000b       ;XOR 25
        DD      00000000000000000000000000000000b       ;XOR 26
        DD      00000000000000000000000000000000b       ;XOR 27
        DD      00000000000000000000000000000000b       ;XOR 28
        DD      00000000000000000000000000000000b       ;XOR 29
        DD      00000000000000000000000000000000b       ;XOR 30
        DD      00000000000000000000000000000000b       ;XOR 31
```

```
;-------------------------------------------------------------------------
;
;       Graphics Cursor Pattern:  Box (32x32)
;
BOX:    DD      00000000000000000000000000000000b       ;AND 00  AND XOR Result
        DD      00111111111111111111111111111100b       ;AND 01  --- --- ------
        DD      01011111111111111111111111111010b       ;AND 02   0   0  Black
        DD      01101111111111111111111111110110b       ;AND 03   0   1  White
        DD      01110111111111111111111111101110b       ;AND 04   1   0  Clear
        DD      01111011111111111111111111011110b       ;AND 05   1   1  Invers
        DD      01111101111111111111111101111110b       ;AND 06
        DD      01111110111111111111111011111110b       ;AND 07
        DD      01111111011111111111110111111110b       ;AND 08
        DD      01111111101111111111101111111110b       ;AND 09
        DD      01111111110111111111011111111110b       ;AND 10
        DD      01111111111011111110111111111110b       ;AND 11
        DD      01111111111101111101111111111110b       ;AND 12
        DD      01111111111110110111111111111110b       ;AND 13
        DD      01111111111111011011111111111110b       ;AND 14
        DD      01111111111111111111111111111110b       ;AND 15
        DD      01111111111111111111111111111110b       ;AND 16
        DD      01111111111111011011111111111110b       ;AND 17
        DD      01111111111110111101111111111110b       ;AND 18
        DD      01111111111101111110111111111110b       ;AND 19
        DD      01111111111011111111011111111110b       ;AND 20
        DD      01111111110111111111101111111110b       ;AND 21
        DD      01111111101111111111110111111110b       ;AND 22
        DD      01111111011111111111111011111110b       ;AND 23
        DD      01111110111111111111111101111110b       ;AND 24
        DD      01111101111111111111111110111110b       ;AND 25
        DD      01111011111111111111111111011110b       ;AND 26
        DD      01110111111111111111111111101110b       ;AND 27
        DD      01101111111111111111111111110110b       ;AND 28
        DD      01011111111111111111111111111010b       ;AND 29
        DD      00111111111111111111111111111100b       ;AND 30
        DD      00000000000000000000000000000000b       ;AND 31

BOXX:   DD      11111111111111111111111111111111b       ;XOR 00  AND XOR Result
        DD      11000000000000000000000000000011b       ;XOR 01  --- --- ------
        DD      10100000000000000000000000000101b       ;XOR 02   0   0  Black
        DD      10010000000000000000000000001001b       ;XOR 03   0   1  White
        DD      10001000000000000000000000010001b       ;XOR 04   1   0  Clear
        DD      10000100000000000000000000100001b       ;XOR 05   1   1  Invers
        DD      10000010000000000000000001000001b       ;XOR 06
        DD      10000001000000000000000010000001b       ;XOR 07
        DD      10000000100000000000000100000001b       ;XOR 08
        DD      10000000010000000000001000000001b       ;XOR 09
        DD      10000000001000000000010000000001b       ;XOR 10
        DD      10000000000100000000100000000001b       ;XOR 11
        DD      10000000000010000001000000000001b       ;XOR 12
        DD      10000000000001000010000000000001b       ;XOR 13
        DD      10000000000000100100000000000001b       ;XOR 14
        DD      10000000000000011000000000000001b       ;XOR 15
        DD      10000000000000011000000000000001b       ;XOR 16
        DD      10000000000000100100000000000001b       ;XOR 17
        DD      10000000000001000010000000000001b       ;XOR 18
        DD      10000000000010000001000000000001b       ;XOR 19
        DD      10000000000100000000100000000001b       ;XOR 20
        DD      10000000001000000000010000000001b       ;XOR 21
        DD      10000000010000000000001000000001b       ;XOR 22
        DD      10000000100000000000000100000001b       ;XOR 23
        DD      10000001000000000000000010000001b       ;XOR 24
        DD      10000010000000000000000001000001b       ;XOR 25
        DD      10000100000000000000000000100001b       ;XOR 26
        DD      10001000000000000000000000010001b       ;XOR 27
        DD      10010000000000000000000000001001b       ;XOR 28
        DD      10100000000000000000000000000101b       ;XOR 29
        DD      11000000000000000000000000000011b       ;XOR 30
        DD      11111111111111111111111111111111b       ;XOR 31
```

```
;-------------------------------------------------------------------------------
;
;       Graphics Cursor Pattern:  Box #2 (32x32)
;
BOX2:   DD      00000000000000000000000000000000b       ;AND 00  AND XOR Result
        DD      00000000000000000000000000000000b       ;AND 01  --- --- ------
        DD      00000000000000000000000000000000b       ;AND 02   0   0  Black
        DD      00000000000000000000000000000000b       ;AND 03   0   1  White
        DD      00000000000000000000000000000000b       ;AND 04   1   0  Clear
        DD      00000000000000000000000000000000b       ;AND 05   1   1  Invers
        DD      00000000000000000000000000000000b       ;AND 06
        DD      00000000000000000000000000000000b       ;AND 07
        DD      00000000000000000000000000000000b       ;AND 08
        DD      00000000000000000000000000000000b       ;AND 09
        DD      00000000000000000000000000000000b       ;AND 10
        DD      00000000000000000000000000000000b       ;AND 11
        DD      00000000000000000000000000000000b       ;AND 12
        DD      00000000000000000000000000000000b       ;AND 13
        DD      00000000000000000000000000000000b       ;AND 14
        DD      00000000000000000000000000000000b       ;AND 15
        DD      00000000000000000000000000000000b       ;AND 16
        DD      00111111111111110011111111111100b       ;AND 17
        DD      00111111111111110011111111111100b       ;AND 18
        DD      00111111111111110011111111111100b       ;AND 19
        DD      00111111111111110011111111111100b       ;AND 20
        DD      00111111111111110011111111111100b       ;AND 21
        DD      00111111111111110011111111111100b       ;AND 22
        DD      00111111111111110011111111111100b       ;AND 23
        DD      00111111111111110011111111111100b       ;AND 24
        DD      00111111111111110011111111111100b       ;AND 25
        DD      00111111111111110011111111111100b       ;AND 26
        DD      00111111111111110011111111111100b       ;AND 27
        DD      00111111111111110011111111111100b       ;AND 28
        DD      00111111111111110011111111111100b       ;AND 29
        DD      00000000000000000000000000000000b       ;AND 30
        DD      00000000000000000000000000000000b       ;AND 31

BOX2X:  DD      11111111111111111111111111111111b       ;XOR 00  AND XOR Result
        DD      11111111111111111111111111111111b       ;XOR 01  --- --- ------
        DD      11000000000000001111111111111111b       ;XOR 02   0   0  Black
        DD      11000000000000001111111111111111b       ;XOR 03   0   1  White
        DD      11000000000000001111111111111111b       ;XOR 04   1   0  Clear
        DD      11000000000000001111111111111111b       ;XOR 05   1   1  Invers
        DD      11000000000000001111111111111111b       ;XOR 06
        DD      11000000000000001111111111111111b       ;XOR 07
        DD      11000000000000001111111111111111b       ;XOR 08
        DD      11000000000000001111111111111111b       ;XOR 09
        DD      11000000000000001111111111111111b       ;XOR 10
        DD      11000000000000001111111111111111b       ;XOR 11
        DD      11000000000000001111111111111111b       ;XOR 12
        DD      11000000000000001111111111111111b       ;XOR 13
        DD      11000000000000001111111111111111b       ;XOR 14
        DD      11111111111111111111111111111111b       ;XOR 15
        DD      11111111111111111111111111111111b       ;XOR 16
        DD      11000000000000001111111111111111b       ;XOR 17
        DD      11000000000000001111111111111111b       ;XOR 18
        DD      11000000000000001111111111111111b       ;XOR 19
        DD      11000000000000001111111111111111b       ;XOR 20
        DD      11000000000000001111111111111111b       ;XOR 21
        DD      11000000000000001111111111111111b       ;XOR 22
        DD      11000000000000001111111111111111b       ;XOR 23
        DD      11000000000000001111111111111111b       ;XOR 24
        DD      11000000000000001111111111111111b       ;XOR 25
        DD      11000000000000001111111111111111b       ;XOR 26
        DD      11000000000000001111111111111111b       ;XOR 27
        DD      11000000000000001111111111111111b       ;XOR 28
        DD      11000000000000001111111111111111b       ;XOR 29
        DD      11111111111111111111111111111111b       ;XOR 30
        DD      11111111111111111111111111111111b       ;XOR 31
```

```
;-------------------------------------------------------------------------
;
;       Graphics Cursor Pattern:  Cross (32x32)
;
CROSS:  DD      1111111111111110011111111111111b        ;AND 00  AND XOR Result
        DD      1111111111111110011111111111111b        ;AND 01  --- --- ------
        DD      1111111111111110011111111111111b        ;AND 02   0   0  Black
        DD      1111111111111110011111111111111b        ;AND 03   0   1  White
        DD      1111111111111110011111111111111b        ;AND 04   1   0  Clear
        DD      1111111111111110011111111111111b        ;AND 05   1   1  Invers
        DD      1111111111111110011111111111111b        ;AND 06
        DD      1111111111111110011111111111111b        ;AND 07
        DD      1111111111111110011111111111111b        ;AND 08
        DD      1111111111111110011111111111111b        ;AND 09
        DD      1111111111111110011111111111111b        ;AND 10
        DD      1111111111111110011111111111111b        ;AND 11
        DD      1111111111111110011111111111111b        ;AND 12
        DD      1111111111111110011111111111111b        ;AND 13
        DD      1111111111111110011111111111111b        ;AND 14
        DD      0000000000000001100000000000000b        ;AND 15
        DD      0000000000000001100000000000000b        ;AND 16
        DD      1111111111111110011111111111111b        ;AND 17
        DD      1111111111111110011111111111111b        ;AND 18
        DD      1111111111111110011111111111111b        ;AND 19
        DD      1111111111111110011111111111111b        ;AND 20
        DD      1111111111111110011111111111111b        ;AND 21
        DD      1111111111111110011111111111111b        ;AND 22
        DD      1111111111111110011111111111111b        ;AND 23
        DD      1111111111111110011111111111111b        ;AND 24
        DD      1111111111111110011111111111111b        ;AND 25
        DD      1111111111111110011111111111111b        ;AND 26
        DD      1111111111111110011111111111111b        ;AND 27
        DD      1111111111111110011111111111111b        ;AND 28
        DD      1111111111111110011111111111111b        ;AND 29
        DD      1111111111111110011111111111111b        ;AND 30
        DD      1111111111111110011111111111111b        ;AND 31

CROSS2: DD      0000000000000001100000000000000b        ;XOR 00  AND XOR Result
        DD      0000000000000001100000000000000b        ;XOR 01  --- --- ------
        DD      0000000000000001100000000000000b        ;XOR 02   0   0  Black
        DD      0000000000000001100000000000000b        ;XOR 03   0   1  White
        DD      0000000000000001100000000000000b        ;XOR 04   1   0  Clear
        DD      0000000000000001100000000000000b        ;XOR 05   1   1  Invers
        DD      0000000000000001100000000000000b        ;XOR 06
        DD      0000000000000001100000000000000b        ;XOR 07
        DD      0000000000000001100000000000000b        ;XOR 08
        DD      0000000000000001100000000000000b        ;XOR 09
        DD      0000000000000001100000000000000b        ;XOR 10
        DD      0000000000000001100000000000000b        ;XOR 11
        DD      0000000000000001100000000000000b        ;XOR 12
        DD      0000000000000001100000000000000b        ;XOR 13
        DD      0000000000000001100000000000000b        ;XOR 14
        DD      1111111111111111111111111111111b        ;XOR 15
        DD      1111111111111111111111111111111b        ;XOR 16
        DD      0000000000000001100000000000000b        ;XOR 17
        DD      0000000000000001100000000000000b        ;XOR 18
        DD      0000000000000001100000000000000b        ;XOR 19
        DD      0000000000000001100000000000000b        ;XOR 20
        DD      0000000000000001100000000000000b        ;XOR 21
        DD      0000000000000001100000000000000b        ;XOR 22
        DD      0000000000000001100000000000000b        ;XOR 23
        DD      0000000000000001100000000000000b        ;XOR 24
        DD      0000000000000001100000000000000b        ;XOR 25
        DD      0000000000000001100000000000000b        ;XOR 26
        DD      0000000000000001100000000000000b        ;XOR 27
        DD      0000000000000001100000000000000b        ;XOR 28
        DD      0000000000000001100000000000000b        ;XOR 29
        DD      0000000000000001100000000000000b        ;XOR 30
        DD      0000000000000001100000000000000b        ;XOR 31
```

```
;-------------------------------------------------------------------------------
;
;       Graphics Cursor Pattern:  Inverse Block
;
INVBLK: DD      11111111111111111111111111111111b       ;AND 00  AND XOR Result
        DD      11111111111111111111111111111111b       ;AND 01  --- --- ------
        DD      11111111111111111111111111111111b       ;AND 02   0   0  Black
        DD      11111111111111111111111111111111b       ;AND 03   0   1  White
        DD      11111111111111111111111111111111b       ;AND 04   1   0  Clear
        DD      11111111111111111111111111111111b       ;AND 05   1   1  Invers
        DD      11111111111111111111111111111111b       ;AND 06
        DD      11111111111111111111111111111111b       ;AND 07
        DD      11111111111111111111111111111111b       ;AND 08
        DD      11111111111111111111111111111111b       ;AND 09
        DD      11111111111111111111111111111111b       ;AND 10
        DD      11111111111111111111111111111111b       ;AND 11
        DD      11111111111111111111111111111111b       ;AND 12
        DD      11111111111111111111111111111111b       ;AND 13
        DD      11111111111111111111111111111111b       ;AND 14
        DD      11111111111111111111111111111111b       ;AND 15
        DD      11111111111111111111111111111111b       ;AND 16
        DD      11111111111111111111111111111111b       ;AND 17
        DD      11111111111111111111111111111111b       ;AND 18
        DD      11111111111111111111111111111111b       ;AND 19
        DD      11111111111111111111111111111111b       ;AND 20
        DD      11111111111111111111111111111111b       ;AND 21
        DD      11111111111111111111111111111111b       ;AND 22
        DD      11111111111111111111111111111111b       ;AND 23
        DD      11111111111111111111111111111111b       ;AND 24
        DD      11111111111111111111111111111111b       ;AND 25
        DD      11111111111111111111111111111111b       ;AND 26
        DD      11111111111111111111111111111111b       ;AND 27
        DD      11111111111111111111111111111111b       ;AND 28
        DD      11111111111111111111111111111111b       ;AND 29
        DD      11111111111111111111111111111111b       ;AND 30
        DD      11111111111111111111111111111111b       ;AND 31

INVXOR: DD      11111111111111111111111111111111b       ;XOR 00  AND XOR Result
        DD      11111111111111111111111111111111b       ;XOR 01  --- --- ------
        DD      11111111111111111111111111111111b       ;XOR 02   0   0  Black
        DD      11111111111111111111111111111111b       ;XOR 03   0   1  White
        DD      11111111111111111111111111111111b       ;XOR 04   1   0  Clear
        DD      11111111111111111111111111111111b       ;XOR 05   1   1  Invers
        DD      11111111111111111111111111111111b       ;XOR 06
        DD      11111111111111111111111111111111b       ;XOR 07
        DD      11111111111111111111111111111111b       ;XOR 08
        DD      11111111111111111111111111111111b       ;XOR 09
        DD      11111111111111111111111111111111b       ;XOR 10
        DD      11111111111111111111111111111111b       ;XOR 11
        DD      11111111111111111111111111111111b       ;XOR 12
        DD      11111111111111111111111111111111b       ;XOR 13
        DD      11111111111111111111111111111111b       ;XOR 14
        DD      11111111111111111111111111111111b       ;XOR 15
        DD      11111111111111111111111111111111b       ;XOR 16
        DD      11111111111111111111111111111111b       ;XOR 17
        DD      11111111111111111111111111111111b       ;XOR 18
        DD      11111111111111111111111111111111b       ;XOR 19
        DD      11111111111111111111111111111111b       ;XOR 20
        DD      11111111111111111111111111111111b       ;XOR 21
        DD      11111111111111111111111111111111b       ;XOR 22
        DD      11111111111111111111111111111111b       ;XOR 23
        DD      11111111111111111111111111111111b       ;XOR 24
        DD      11111111111111111111111111111111b       ;XOR 25
        DD      11111111111111111111111111111111b       ;XOR 26
        DD      11111111111111111111111111111111b       ;XOR 27
        DD      11111111111111111111111111111111b       ;XOR 28
        DD      11111111111111111111111111111111b       ;XOR 29
        DD      11111111111111111111111111111111b       ;XOR 30
        DD      11111111111111111111111111111111b       ;XOR 31
```

```
;-------------------------------------------------------------------------
;
;       Graphics Cursor Pattern:  White Block
;
WHTBLK: DD      00000000000000000000000000000000b       ;AND 00  AND XOR Result
        DD      00000000000000000000000000000000b       ;AND 01  --- --- ------
        DD      00000000000000000000000000000000b       ;AND 02   0   0  Black
        DD      00000000000000000000000000000000b       ;AND 03   0   1  White
        DD      00000000000000000000000000000000b       ;AND 04   1   0  Clear
        DD      00000000000000000000000000000000b       ;AND 05   1   1  Invers
        DD      00000000000000000000000000000000b       ;AND 06
        DD      00000000000000000000000000000000b       ;AND 07
        DD      00000000000000000000000000000000b       ;AND 08
        DD      00000000000000000000000000000000b       ;AND 09
        DD      00000000000000000000000000000000b       ;AND 10
        DD      00000000000000000000000000000000b       ;AND 11
        DD      00000000000000000000000000000000b       ;AND 12
        DD      00000000000000000000000000000000b       ;AND 13
        DD      00000000000000000000000000000000b       ;AND 14
        DD      00000000000000000000000000000000b       ;AND 15
        DD      00000000000000000000000000000000b       ;AND 16
        DD      00000000000000000000000000000000b       ;AND 17
        DD      00000000000000000000000000000000b       ;AND 18
        DD      00000000000000000000000000000000b       ;AND 19
        DD      00000000000000000000000000000000b       ;AND 20
        DD      00000000000000000000000000000000b       ;AND 21
        DD      00000000000000000000000000000000b       ;AND 22
        DD      00000000000000000000000000000000b       ;AND 23
        DD      00000000000000000000000000000000b       ;AND 24
        DD      00000000000000000000000000000000b       ;AND 25
        DD      00000000000000000000000000000000b       ;AND 26
        DD      00000000000000000000000000000000b       ;AND 27
        DD      00000000000000000000000000000000b       ;AND 28
        DD      00000000000000000000000000000000b       ;AND 29
        DD      00000000000000000000000000000000b       ;AND 30
        DD      00000000000000000000000000000000b       ;AND 31

WHTXOR: DD      11111111111111111111111111111111b       ;XOR 00  AND XOR Result
        DD      11111111111111111111111111111111b       ;XOR 01  --- --- ------
        DD      11111111111111111111111111111111b       ;XOR 02   0   0  Black
        DD      11111111111111111111111111111111b       ;XOR 03   0   1  White
        DD      11111111111111111111111111111111b       ;XOR 04   1   0  Clear
        DD      11111111111111111111111111111111b       ;XOR 05   1   1  Invers
        DD      11111111111111111111111111111111b       ;XOR 06
        DD      11111111111111111111111111111111b       ;XOR 07
        DD      11111111111111111111111111111111b       ;XOR 08
        DD      11111111111111111111111111111111b       ;XOR 09
        DD      11111111111111111111111111111111b       ;XOR 10
        DD      11111111111111111111111111111111b       ;XOR 11
        DD      11111111111111111111111111111111b       ;XOR 12
        DD      11111111111111111111111111111111b       ;XOR 13
        DD      11111111111111111111111111111111b       ;XOR 14
        DD      11111111111111111111111111111111b       ;XOR 15
        DD      11111111111111111111111111111111b       ;XOR 16
        DD      11111111111111111111111111111111b       ;XOR 17
        DD      11111111111111111111111111111111b       ;XOR 18
        DD      11111111111111111111111111111111b       ;XOR 19
        DD      11111111111111111111111111111111b       ;XOR 20
        DD      11111111111111111111111111111111b       ;XOR 21
        DD      11111111111111111111111111111111b       ;XOR 22
        DD      11111111111111111111111111111111b       ;XOR 23
        DD      11111111111111111111111111111111b       ;XOR 24
        DD      11111111111111111111111111111111b       ;XOR 25
        DD      11111111111111111111111111111111b       ;XOR 26
        DD      11111111111111111111111111111111b       ;XOR 27
        DD      11111111111111111111111111111111b       ;XOR 28
        DD      11111111111111111111111111111111b       ;XOR 29
        DD      11111111111111111111111111111111b       ;XOR 30
        DD      11111111111111111111111111111111b       ;XOR 31
```

```
;-------------------------------------------------------------------------
;
;       Graphics Cursor Pattern:  Black Block
;
BLKBLK: DD      00000000000000000000000000000000b       ;AND 00  AND XOR Result
        DD      00000000000000000000000000000000b       ;AND 01  --- --- ------
        DD      00000000000000000000000000000000b       ;AND 02   0   0  Black
        DD      00000000000000000000000000000000b       ;AND 03   0   1  White
        DD      00000000000000000000000000000000b       ;AND 04   1   0  Clear
        DD      00000000000000000000000000000000b       ;AND 05   1   1  Invers
        DD      00000000000000000000000000000000b       ;AND 06
        DD      00000000000000000000000000000000b       ;AND 07
        DD      00000000000000000000000000000000b       ;AND 08
        DD      00000000000000000000000000000000b       ;AND 09
        DD      00000000000000000000000000000000b       ;AND 10
        DD      00000000000000000000000000000000b       ;AND 11
        DD      00000000000000000000000000000000b       ;AND 12
        DD      00000000000000000000000000000000b       ;AND 13
        DD      00000000000000000000000000000000b       ;AND 14
        DD      00000000000000000000000000000000b       ;AND 15
        DD      00000000000000000000000000000000b       ;AND 16
        DD      00000000000000000000000000000000b       ;AND 17
        DD      00000000000000000000000000000000b       ;AND 18
        DD      00000000000000000000000000000000b       ;AND 19
        DD      00000000000000000000000000000000b       ;AND 20
        DD      00000000000000000000000000000000b       ;AND 21
        DD      00000000000000000000000000000000b       ;AND 22
        DD      00000000000000000000000000000000b       ;AND 23
        DD      00000000000000000000000000000000b       ;AND 24
        DD      00000000000000000000000000000000b       ;AND 25
        DD      00000000000000000000000000000000b       ;AND 26
        DD      00000000000000000000000000000000b       ;AND 27
        DD      00000000000000000000000000000000b       ;AND 28
        DD      00000000000000000000000000000000b       ;AND 29
        DD      00000000000000000000000000000000b       ;AND 30
        DD      00000000000000000000000000000000b       ;AND 31

BLKXOR: DD      00000000000000000000000000000000b       ;XOR 00  AND XOR Result
        DD      00000000000000000000000000000000b       ;XOR 01  --- --- ------
        DD      00000000000000000000000000000000b       ;XOR 02   0   0  Black
        DD      00000000000000000000000000000000b       ;XOR 03   0   1  White
        DD      00000000000000000000000000000000b       ;XOR 04   1   0  Clear
        DD      00000000000000000000000000000000b       ;XOR 05   1   1  Invers
        DD      00000000000000000000000000000000b       ;XOR 06
        DD      00000000000000000000000000000000b       ;XOR 07
        DD      00000000000000000000000000000000b       ;XOR 08
        DD      00000000000000000000000000000000b       ;XOR 09
        DD      00000000000000000000000000000000b       ;XOR 10
        DD      00000000000000000000000000000000b       ;XOR 11
        DD      00000000000000000000000000000000b       ;XOR 12
        DD      00000000000000000000000000000000b       ;XOR 13
        DD      00000000000000000000000000000000b       ;XOR 14
        DD      00000000000000000000000000000000b       ;XOR 15
        DD      00000000000000000000000000000000b       ;XOR 16
        DD      00000000000000000000000000000000b       ;XOR 17
        DD      00000000000000000000000000000000b       ;XOR 18
        DD      00000000000000000000000000000000b       ;XOR 19
        DD      00000000000000000000000000000000b       ;XOR 20
        DD      00000000000000000000000000000000b       ;XOR 21
        DD      00000000000000000000000000000000b       ;XOR 22
        DD      00000000000000000000000000000000b       ;XOR 23
        DD      00000000000000000000000000000000b       ;XOR 24
        DD      00000000000000000000000000000000b       ;XOR 25
        DD      00000000000000000000000000000000b       ;XOR 26
        DD      00000000000000000000000000000000b       ;XOR 27
        DD      00000000000000000000000000000000b       ;XOR 28
        DD      00000000000000000000000000000000b       ;XOR 29
        DD      00000000000000000000000000000000b       ;XOR 30
        DD      00000000000000000000000000000000b       ;XOR 31

PTRCNT  EQU     ($-PTRS)/256    ;Number of pointers
MAXCNT  EQU     255             ;Max pointer pattern #
```

Table 6-3: V7VGA User-Programmable Extension Register Summary

Index	Description
94	**Pointer Pattern Address**: bits 13-6 of the address at which the currently displayed pointer pattern (hardware graphics cursor) is stored. Bits 15 & 14 are always 0, and each pointer pattern is 64 addresses long (accounting for bits 5-0).
9C	**Pointer Horizontal Position Low**: bits 7-0 of the X coordinate of the hardware graphics cursor (in pixels). The value in this register does not actually take effect on the screen until extension register 9F is written.
9D	**Pointer Horizontal Position High**: bits 10-8 of the X coordinate of the hardware graphics cursor (in pixels). The value in this register does not actually take effect on the screen until extension register 9F is written.
9E	**Pointer Vertical Position Low**: bits 7-0 of the Y coordinate of the hardware graphics cursor (in pixels). The value in this register does not actually take effect on the screen until extension register 9F is written.
9F	**Pointer Vertical Position High**: bits 9 & 8 of the Y coordinate of the hardware graphics cursor (in pixels). Writing this register updates the actual screen position.
A0-A3	**Graphics Controller Latches 0-3**: the blue, green, red, and intensity plane latches, respectively.
A5	**Cursor Attributes**: Bit 7: 1 to enable the hardware graphics cursor 0 to disable Bit 3: 1 to select XOR text cursor 0 to select normal (overwrite) cursor Bit 0: 1 to disable cursor blink, 0 to enable cursor block (normal)
F1	**Fast Latch Load State**: Bits 1 & 0: the latch register (0-3 correspsonds to A0-A3) written to by the next write to the Fast Background Latch Load register.
F2	**Fast Background Latch Load**: on reads, resets bits 1 & 0 of the Fast Latch Load State register to 0. On writes, data written to this register goes to the one of four latch registers (A0-A3) selected by bits 1 & 0 of the Fast Latch Load State register, and the field made up of bits 1 & 0 of the Fast Latch Load State register is incremented after the write is complete.
F3	**Masked Write Control**: Bit 1: 1 to select the rotated CPU byte as the masked write mask source 0 to select the Masked Write Mask register as the masked write mask source (No effect if bit 0 is 0.) Bit 0: 1 to enable masked writes (write-per-bit), 0 to disable masked writes. Masked writes only work with V-RAM.
F4	**Masked Write Mask**: the bit pattern used for the masked write mask when bit 1 of the Masked Write Control register is 0 and bit 1 of that register is a 1.
F5	**Foreground/Background Pattern**: the bit pattern to be expanded to a two-color pattern when bits 3 & 2 of the Foreground/Background Control register are 01 and bit 1 of that register is 0. 1 bits are expanded to the color in the Extended Foreground Color register, and 0 bits are expanded to the color in the Extended Background Color register.

(continued)

Table 6-3: V7VGA User-Programmable Extension Register Summary (continued)

F6 **1 Mb DRAM Bank Select**

Bit 7: 1 to reset the bank when line compare occurs
0 to load the bank select from bits 5 & 4 of this register when line compare occurs
Bit 6: 1 to let banks turn over (scan into the next bank)
0 to keep the bank from turning over
Bits 5 & 4: Bank from which CRTC starts fetching video data at the start of a frame
Bits 3 & 2: Bank from which the CPU reads
Bits 1 & 0: Bank to which the CPU writes

F9 **Extended Page Select**

Bit 0: 1 to select extended page 1
0 to select extended page 0
(No effect when bit-2 of the Compatibility Control register is 0)

Exact action depends on the settings of bits 5, 4, and 1 of the Compatibility Control register.

FA **Extended Foreground Color**: Bits 3-0: color 1 bits are expanded to when bits 3 & 2 of the Foreground/Background Control register are 01.

FB **Extended Background Color**: Bits 3-0: color 0 bits are expanded to when bits 3 & 2 of the Foreground/Background Control register are 01.

FC **Compatibility Control**:

Bit 5: 1 to select sequential chain 4 mode (useful for hi-res V-RAM 256-color modes)
Bit 4: 1 to select sequential chain mode (useful for hi-res V-RAM 256-color modes)
Bit 2: 1 to enable extended 256-color modes
0 to select normal 256-color operation
Bit 1: 1 to select 128K extended 256-color mode
0 to select 64K extended 256-color modes
Bit 0: 1 to enable extended attributes (extended underlining).

FE **Foreground/Background Control**:

Bits 3 & 2: Foreground/background mode selection. 00 for normal IBM VGA operation, 01 for color text expansion, 10 and 11 reserved.
Bit 1: 1 to select the rotated CPU byte as the color expansion source
0 to select the FG / BG Pattern register as the color expansion source.

FF **16-Bit Interface Control**:

Bits 6 & 5: select the bank in which the pointer pattern (hardware graphics cursor masks) resides.
Bit 4: 1 to select banked operation with 256K of V-RAM or D-RAM
0 to select banked operation with 1 Mbit D-RAM chips

Important Do not change any unused bits in any registers, since some of those bits are used by the BIOS. To change selected bits, read the register, change just the bits of interest, and write the result back to the register. Failure to observe this could result in unreadable displays or reduced performance.

APPENDIX A

Register Initialization Tables

This appendix shows the default values for the registers when operating the V7VGA board in all modes. There are two tables: 1) VGA-Standard modes and 2) Video Seven proprietary modes.

Standard Modes				Displays
0,1	40x25	Color Text	(8x8 font)	AD, MS
0,1*	40x25	Color Text	(8x14 font)	AD, MS
0,1+	40x25	Color Text	(9x16 font)	AD, MS
2,3	80x25	Color Text	(8x8 font)	AD, MS
2,3*	80x25	Color Text	(8x14 font)	AD, MS
2,3+	80x25	Color Text	(9x16 font)	AD, MS
7	80x25	Mono Text	(9x14 font)	AD, MS
7+	80x25	Mono Text	(9x16 font)	AD, MS
6	640x200	Mono Graphics	(2-color)	AD, MS
F	640x350	Mono Graphics	(4-color)	AD, MS
11	640x480	Mono Graphics	(2-color)	AD, MS
4,5	320x200	Color Graphics	(4-color)	AD, MS
D	320x200	Color Graphics	(16-color)	AD, MS
E	640x200	Color Graphics	(16-color)	AD, MS
10	640x350	Color Graphics	(16-color)	AD, MS
12	640x480	Color Graphics	(16-color)	AD, MS
13	320x200	Color Graphics	(256-color)	AD, MS

Note: MS = Multi-Frequency Display

AD = PS/2-Type Fixed-Frequency Analog Display (Color or Monochrome) (IBM 8512,8513,8514 Color or IBM 8503 Mono or equivalent)

XL = MultiSync XL (19"), Nanao 9070S (16"), or equivalent

BIOS Setmode:

```
mov ax,mode
int  10h
```

Proprietary Modes				Displays
40	80x43	Color Text	(8x8 font)	AD, MS
41	132x25	Color Text	(8x14 font)	AD, MS
42	132x43	Color Text	(8x8 font)	AD, MS
43	80x60	Color Text	(8x8 font)	AD, MS
44	100x60	Color Text	(8x8 font)	AD, MS
45	132x28	Color Text	(8x14 font)	AD, MS
46-5F	(reserved)			
60	752x410	Color Graphics	(16-color)	AD, MS
61	720x540	Color Graphics	(16-color)	XL, MS
62	800x600	Color Graphics	(16-color)	XL, MS
63	1024x768	Color Graphics	(2-color)	XL
64	1024x768	Color Graphics	(4-color)	XL
65	1024x768	Color Graphics	(16-color)	XL
66	640x400	Color Graphics	(256-color)	AD, MS
67	640x480	Color Graphics	(256-color)	AD, MS
68	720x540	Color Graphics	(256-color)	XL, MS
69 (future)	800x600	Color Graphics	(256-color)	XL
6A-7F	(reserved)			

BIOS Setmode:

```
mov ax,6f05h
mov bl,mode
int  10h
```

6/27/88

6/27/88

Table A - 1. Mode of Operation (PS/2-Type Analog Displays)

	Characteristics	0,1	2,3	4,5	6	D	E	7	F	10	0,1*	2,3*	0/1+	2/3+	7+	11	12	13
	Text / Graphics	Text	Text	Gr	Gr	Gr	Gr	Text	Gr	Gr	Text	Text	Text	Text	Text	Gr	Gr	Gr
	Colors	16	16	4	2	16	16	4*	2	16	16	16	16	16	4*	16	16	256
Pixels	H Resolution	320	640	320	640	320	640	720	640	640	320	640	360	720	720	640	640	320
Pixels	V Resolution	200	200	200	200	200	200	350	350	350	350	350	400	400	400	480	480	200
Chars	Text Columns	40	80	-	-	-	-	80	-	-	40	80	40	80	80	-	-	-
Rows	Text Rows	25	25	-	-	-	-	25	-	-	25	25	25	25	25	-	-	-
Pixels	Character Width	8	8	8	8	8	8	9	8	8	8	8	8	9	9	8	8	8
Pixels	Character Height	8	8	1	1	1	1	14	1	1	14	14	16	16	16	1	1	2
CPU:CRT	Bandwidth (D-Ram)	3:2	1:4	3:2	1:4	3:2	1:4	1:4	1:4	1:4	3:2	1:4	3:2	1:4	1:4	1:4	1:4	1:4
CPU:CRT	Bandwidth (V-Ram)	1:1	1:1	1:1	1:1	1:1	1:1	1:1	1:1	1:1	1:1	1:1	1:1	1:1	1:1	1:1	1:1	1:1
MHz	Pixel Clock	12	25	12	25	12	25	28	25	25	12	25	14	28	28	25	25	25
KHz	H Sync Rate	31.5	31.5	31.5	31.5	31.5	31.5	31.5	31.5	31.5	31.5	31.5	31.5	31.5	31.5	31.5	31.5	31.5
Hz	V Sync Rate	70	70	70	70	70	70	70	70	70	70	70	70	70	70	60	60	70
Chars	H Displayed	40	80	40	80	40	80	80	80	80	40	80	40	80	80	80	80	80
Chars	H Total	57	100	57	100	57	100	100	100	100	57	100	57	100	100	100	100	100
Lines	V Displayed	400	400	400	400	400	400	350	350	350	350	350	400	400	400	480	480	400
Lines	V Total	449	449	449	449	449	449	449	449	449	449	449	451	449	449	524	524	449

	Display Timing	0,1	2,3	4,5	6	D	E	7	F	10	0,1*	2,3*	0/1+	2/3+	7+	11	12	13
	Display Type	AD	AD	AD	AD	AD	AD	AD	AD	AD	AD	AD	AD	AD	AD	AD	AD	AD
usec	H Sync Delay	4.5	1.9	4.5	1.9	4.5	1.9	1.9	1.9	1.9	4.5	1.9	4.5	1.9	1.9	1.9	1.9	1.9
usec	H Sync Width	3.9	3.8	3.9	3.8	3.9	3.8	3.8	3.8	3.8	3.9	3.8	3.9	3.8	3.8	3.8	3.8	3.8
usec	V Sync Delay	414	413	414	413	414	413	1207	1208	1208	1210	1208	414	413	413	254	254	413
usec	V Sync Width	64	64	64	64	64	64	64	64	64	64	64	64	64	64	381	381	64

	Extensions	0,1	2,3	4,5	6	D	E	7	F	10	0,1*	2,3*	0/1+	2/3+	7+	11	12	13
FD	Timing State (D/V)	22/82	22/82	22/82	22/82	22/82	22/82	22/82	22/82	22/82	22/82	22/82	22/82	22/82	22/82	22/82	22/82	22/82
A4	Clock	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
--	Sony Fixup	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
FC	Compatibility Ctrl	08	08	08	08	08	08	08	08	08	08	08	08	08	08	08	08	08
F6	Bank Select	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
F8	Extended Clock	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
FF	16-Bit Interface	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

	Misc Info	0,1	2,3	4,5	6	D	E	7	F	10	0,1*	2,3*	0/1+	2/3+	7+	11	12	13
	Text Columns	40	80	40	80	40	80	80	80	80	40	80	40	80	80	80	80	40
	Text Rows - 1	24	24	24	24	24	24	24	24	24	24	24	24	24	24	29	29	24
	Font Height	8	8	8	8	8	8	14	14	14	14	14	16	16	16	16	16	8
	Page Size	0800	1000	4000	4000	2000	4000	1000	8000	8000	0800	1000	0800	1000	1000	9600	9600	FA00

	Sequencer	0,1	2,3	4,5	6	D	E	7	F	10	0,1*	2,3*	0/1+	2/3+	7+	11	12	13
SR1	Clocking Mode	9	1	9	1	9	1	0	5/1	5/1	9	1	8	0	0	1	1	1
SR2	Plane Mask	3	3	3	1	F	F	3	F	F	3	3	3	3	3	F	F	F
SR3	Char Map Select	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SR4	Memory Mode	2	2	2	6	6	6	3	0/6	0/6	2	2	2	2	2	6	6	E

	External Regs	0,1	2,3	4,5	6	D	E	7	F	10	0,1*	2,3*	0/1+	2/3+	7+	11	12	13
MISC	Miscellaneous	63	63	63	63	63	63	A6	A2	A3	A3	A3	67	67	66	E3	E3	63

Table A - 1. (continued) Mode of Operation (PS/2-Type Analog Displays)

	CRT Ctrlr	0,1	2,3	4,5	6	D	E	7	F	10	0,1*	2,3*	0/1+	2/3+	7+	11	12	13
CR0	H Total - 5	2D	5F	2D	5F	2D	5F	5F	5F	5F	2D	5F	2D	5F	5F	5F	5F	5F
CR1	H Display End	27	4F	27	4F	27	4F	4F	4F	4F	27	4F	27	4F	4F	4F	4F	4F
CR2	H Blanking Start	28	50	28	50	28	50	50	50	50	28	50	28	50	50	50	50	50
CR3	H Blanking End	90	82	90	82	90	82	82	1A/82	17/82	90	82	90	82	82	82	82	82
CR4	H Retrace Start	2B	55	2B	54	2B	54	55	54	64	2B	55	2B	55	55	54	54	54
CR5	H Retrace End	A0	81	80	80	80	80	81	E0/80	BA/80	A0	81	A0	81	81	80	80	80
CR6	V Total - 2	BF	BF	BF	BF	BF	BF	BF	BF	BF	BF	BF	BF	BF	BF	0B	0B	BF
CR7	Overflow	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F	3E	3E	1F
CR8	Preset Row Scan	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
CR9	Char Cell Height	C7	C7	C1	C1	C0	C0	4D	40	40	4D	4D	4F	4F	4F	40	40	41
CRA	Cursor Start	06	06	00	00	00	00	0B	00	00	0B	0B	0D	0D	0D	00	00	00
CRB	Cursor End	07	07	00	00	00	00	0C	00	00	0C	0C	0E	0E	0E	00	00	00
CRC	Start Address H	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
CRD	Start Address L	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
CRE	Cursor Location H	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
CRF	Cursor Location L	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
CR10 (R)	Light Pen H	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
CR11 (R)	Light Pen L	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
CR10 (W)	V Retrace Start	9C	9C	9C	9C	9C	9C	83	83	83	83	83	9C	9C	9C	EA	EA	9C
CR11 (W)	V Retrace End	8E	8E	8E	8E	8E	8E	85	85	85	85	85	8E	8E	8E	8C	8C	8E
CR12	V Display End	8F	8F	8F	8F	8F	8F	5D	5D	5D	5D	5D	8F	8F	8F	DF	DF	8F
CR13	Offset	14	28	14	28	14	28	28	14/28	14/28	14	28	14	28	28	28	28	28
CR14	Underline Row	1F	1F	00	00	00	00	0D	00	00	1F	1F	1F	1F	0F	00	00	40
CR15	V Blanking Start	96	96	96	96	96	96	63	63	63	63	63	96	96	96	E7	E7	96
CR16	V Blanking End	B9	B9	B9	B9	B9	B9	BA	BA	BA	BA	BA	B9	B9	B9	04	04	B9
CR17	Mode Control	A3	A3	A2	C2	E3	E3	A3	8B/E3	8B/E3	A3	A3	A3	A3	A3	C3	E3	A3
CR18	Line Compare	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

	Attribute Ctrlr	0,1	2,3	4,5	6	D	E	7	F	10	0,1*	2,3*	0/1+	2/3+	7+	11	12	13
AR0	Palette 00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
AR1	Palette 01	01	01	13	17	01	01	08	08	01	01	01	01	01	08	3F	01	01
AR2	Palette 02	02	02	15	17	02	02	08	00	02	02	02	02	02	08	3F	02	02
AR3	Palette 03	03	03	17	17	03	03	08	00	03	03	03	03	03	08	3F	03	03
AR4	Palette 04	04	04	02	17	04	04	08	18	04	04	04	04	04	08	3F	04	04
AR5	Palette 05	05	05	04	17	05	05	08	18	05	05	05	05	05	08	3F	05	05
AR6	Palette 06	06	06	06	17	06	06	08	00	14	14	14	14	14	08	3F	14	06
AR7	Palette 07	07	07	07	17	07	07	08	00	07	07	07	07	07	08	3F	07	07
AR8	Palette 08	10	10	10	17	10	10	10	00	38	38	38	38	38	10	3F	38	08
AR9	Palette 09	11	11	11	17	11	11	18	08	39	39	39	39	39	18	3F	39	09
ARA	Palette 0A	12	12	12	17	12	12	18	00	3A	3A	3A	3A	3A	18	3F	3A	0A
ARB	Palette 0B	13	13	13	17	13	13	18	00	3B	3B	3B	3B	3B	18	3F	3B	0B
ARC	Palette 0C	14	14	14	17	14	14	18	00	3C	3C	3C	3C	3C	18	3F	3C	0C
ARD	Palette 0D	15	15	15	17	15	15	18	18	3D	3D	3D	3D	3D	18	3F	3D	0D
ARE	Palette 0E	16	16	16	17	16	16	18	00	3E	3E	3E	3E	3E	18	3F	3E	0E
ARF	Palette 0F	17	17	17	17	17	17	18	00	3F	3F	3F	3F	3F	18	3F	3F	0F
AR10	Mode Control	08	08	01	01	01	01	0E	0B	01	08	08	0C	0C	0E	01	01	41
AR11	Overscan Color	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
AR12	Color Plane Ena	0F	0F	03	01	0F	0F	0F	05	0F	0F	0F	0F	0F	0F	0F	0F	0F
AR13	H Pixel Pan	00	00	00	00	00	00	08	00	00	00	00	08	08	08	00	00	00

	Graphics Ctrlr	0,1	2,3	4,5	6	D	E	7	F	10	0,1*	2,3*	0/1+	2/3+	7+	11	12	13
GR0	Set/Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GR1	Enable Set/Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GR2	Color Compare	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GR3	Data Rotate	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GR4	Read Map Select	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GR5	Mode	10	10	30	00	00	00	10	10/00	10/00	10	10	10	10	10	00	00	40
GR6	Miscellaneous	0E	0E	0F	0D	05	05	0A	7/5	7/5	0E	0E	0E	0E	0A	05	05	05
GR7	Color Don't Care	00	00	00	00	0F	0F	00	0F	0F	00	00	00	00	00	01	0F	0F
GR8	Bit Mask	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

Table A - 2. Mode of Operation (Video Seven Proprietary Text Modes)

	Characteristics	40	41	42	43	44	45	46
	Text / Graphics	Text	Text	Text	Text	Text	Text	Text
	Colors	16	16	16	16	16	16	16
Pixels	H Resolution	640	1056	1056	640	800	1056	720
Pixels	V Resolution	350	350	350	480	480	400	540
Chars	Text Columns	80	132	132	80	100	132	80
Rows	Text Rows	43	25	43	60	60	28	67
Pixels	Character Width	8	8	8	8	8	8	9
Pixels	Character Height	8	14	8	8	8	8	8
CPU:CRT	Bandwidth (D-Ram)							
CPU:CRT	Bandwidth (V-Ram)							
MHz	Pixel Clock	25	40	40	25	40	40	
KHz	H Sync Rate	31.5	31.5	31.5	31.5	31.5	31.5	
Hz	V Sync Rate	70	70	70	60	60	70	
Chars	H Displayed	80	132	132	80	100	132	80
Chars	H Total							
Lines	V Displayed	344	350	344	480	480	400	540
Lines	V Total							

	Display Timing	40	41	42	43	44	45	46
	Display Type	AD	AD	AD	AD	AD	AD	MS
usec	H Sync Delay							
usec	H Sync Width							
usec	V Sync Delay							
usec	V Sync Width							

	Extensions	40	41	42	43	44	45	46
FD	Timing State (D/V)	22	22	22	22	22	22	22
A4	Clock	00	00	00	00	00	00	00
--	Sony Fixup	00	00	00	00	00	00	00
FC	Compatibility Ctrl	08	08	08	08	08	08	08
F6	Bank Select	00	00	00	00	00	00	00
F8	Extended Clock	00	00	00	00	00	00	00
FF	16-Bit Interface	00	00	00	00	00	00	00

	Misc Info	40	41	42	43	44	45	46
	Text Columns	80	132	132	80	100	132	80
	Text Rows - 1	42	24	42	59	59	27	66
	Font Height	8	14	8	8	8	14	8
	Page Size	1B00	1A00	2D00	2600	2F00	1D00	2A00

	Sequencer	40	41	42	43	44	45	46
SR1	Clocking Mode	1	1	1	1	1	1	0
SR2	Plane Mask	3	3	3	3	3	3	3
SR3	Char Map Select	0	0	0	0	0	0	0
SR4	Memory Mode	2	2	2	2	2	2	2

	External Regs	40	41	42	43	44	45	46
MISC	Miscellaneous	A3	AF	AF	E3	E7	6F	

Table A - 2. (continued) Mode of Operation (Video Seven Proprietary Text Modes)

	CRT Ctrlr	40	41	42	43	44	45	46
CR0	H Total - 5	5F	9C	9C	5F	7C	9C	
CR1	H Display End	4F	83	83	4F	63	83	
CR2	H Blanking Start	50	84	84	50	64	84	
CR3	H Blanking End	82	9F	9F	82	9F	9F	
CR4	H Retrace Start	55	89	89	55	6E	89	
CR5	H Retrace End	81	1C	1C	81	9D	1C	
CR6	V Total - 2	BF	BF	BF	0B	0B	BF	
CR7	Overflow	1F	1F	1F	3E	3E	1F	
CR8	Preset Row Scan	00	00	00	00	00	00	00
CR9	Char Cell Height	47	4D	47	47	47	4D	47
CRA	Cursor Start	06	0B	06	06	06	0B	06
CRB	Cursor End	07	0C	07	07	07	0C	07
CRC	Start Address H	--	--	--	--	--	--	--
CRD	Start Address L	--	--	--	--	--	--	--
CRE	Cursor Location H	--	--	--	--	--	--	--
CRF	Cursor Location L	--	--	--	--	--	--	--
CR10 (R)	Light Pen H	--	--	--	--	--	--	--
CR11 (R)	Light Pen L	--	--	--	--	--	--	--
CR10 (W)	V Retrace Start	83	83	83	EA	EA	9C	
CR11 (W)	V Retrace End	85	85	85	8C	8C	8E	
CR12	V Display End	57	5D	57	DF	DF	87	
CR13	Offset	28	42	42	28	32	42	
CR14	Underline Row	1F	1F	1F	1F	1F	1F	1F
CR15	V Blanking Start	63	63	63	E7	E7	90	
CR16	V Blanking End	BA	BA	BA	04	04	B9	
CR17	Mode Control	A3	E3	E3	A3	E3	E3	
CR18	Line Compare	FF	FF	FF	FF	FF	FF	FF

	Attribute Ctrlr	40	41	42	43	44	45	46
AR0	Palette 00	00	00	00	00	00	00	00
AR1	Palette 01	01	01	01	01	01	01	01
AR2	Palette 02	02	02	02	02	02	02	02
AR3	Palette 03	03	03	03	03	03	03	03
AR4	Palette 04	04	04	04	04	04	04	04
AR5	Palette 05	05	05	05	05	05	05	05
AR6	Palette 06	14	14	14	14	14	14	14
AR7	Palette 07	07	07	07	07	07	07	07
AR8	Palette 08	38	38	38	38	38	38	38
AR9	Palette 09	39	39	39	39	39	39	39
ARA	Palette 0A	3A	3A	3A	3A	3A	3A	3A
ARB	Palette 0B	3B	3B	3B	3B	3B	3B	3B
ARC	Palette 0C	3C	3C	3C	3C	3C	3C	3C
ARD	Palette 0D	3D	3D	3D	3D	3D	3D	3D
ARE	Palette 0E	3E	3E	3E	3E	3E	3E	3E
ARF	Palette 0F	3F	3F	3F	3F	3F	3F	3F
AR10	Mode Control	08	08	08	08	08	08	08
AR11	Overscan Color	00	00	00	00	00	00	00
AR12	Color Plane Ena	0F	0F	0F	0F	0F	0F	0F
AR13	H Pixel Pan	00	00	00	00	00	00	08

	Graphics Ctrlr	40	41	42	43	44	45	46
GR0	Set/Reset	0	0	0	0	0	0	0
GR1	Enable Set/Reset	0	0	0	0	0	0	0
GR2	Color Compare	0	0	0	0	0	0	0
GR3	Data Rotate	0	0	0	0	0	0	0
GR4	Read Map Select	0	0	0	0	0	0	0
GR5	Mode	10	10	10	10	10	10	10
GR6	Miscellaneous	0E	0E	0E	0E	0E	0E	0E
GR7	Color Don't Care	00	00	00	00	00	00	00
GR8	Bit Mask	FF	FF	FF	FF	FF	FF	FF

Table A - 3. Mode of Operation (Video Seven Proprietary Graphics Modes)

	Characteristics	60	61	62	63	64	65	66	67	68	69
	Text / Graphics	Gr	Gr	Gr	Gr	Gr	Gr	Gr	Gr	Gr	Gr
	Colors	16	16	16	2	4	16	256	256	256	256
Pixels	H Resolution	752	720	800	1024	1024	1024	640	640	720	800
Pixels	V Resolution	410	540	600	768	768	768	400	480	540	600
Chars	Text Columns	-	-	-	-	-	-	-	-	-	-
Rows	Text Rows	-	-	-	-	-	-	-	-	-	-
Pixels	Character Width	8	8	8	8	8	8	8	8	8	8
Pixels	Character Height	1	1	1	1	1	1	1	1	1	1
CPU:CRT	Bandwidth (D-Ram)	1:4	1:4	p1:4	p1:4	p1:4	p1:4	p1:4	p1:4	p1:4	p1:4
CPU:CRT	Bandwidth (V-Ram)	1:1	1:1	1:2	1:4	1:4	1:4	1:4	1:4	1:4	1:4
MHz	Pixel Clock	28.3	32.5	40	65	65	65	50.4	50.4	65	65
KHz	H Sync Rate	31.6	35.0	37.9	48.5	48.5	48.5	31.5	31.5	35.0	32.9
Hz	V Sync Rate	70	60	60	60	60	60	70	60	60	53
Chars	H Displayed	94	90	100	128	128	128	80	80	90	100
Chars	H Total										
Lines	V Displayed	410	540	600	768	768	768	400	480	540	600
Lines	V Total										

	Display Timing	60	61	62	63	64	65	66	67	68	69
	Display Type	AD	MS	XL	XL	XL	XL	AD	AD	MS	XL
usec	H Sync Delay										
usec	H Sync Width										
usec	V Sync Delay										
usec	V Sync Width										

	Extensions	60	61	62	63	64	65	66	67	68	69
FD	Timing State (D/V)	0/0	0/0	0/90	0/A0	0/A0	0/A0	0/A0	0/A0	0/A0	0/A0
A4	Clock	00	10	10	10	10	10	10	10	10	10
--	Sony H Skew	00	00	00	00	00	00	00	00	00	00
FC	Compatibility Ctrl	08	08	08	18	18	08	6C	6C	6C	6C
F6	Bank Select	00	00	00	00	00	C0	00	C0	C0	C0
F8	Extended Clock	00	00	10	00	00	00	00	00	00	00
FF	16-Bit Interface	00	00	00	00	00	10	00	10	10	10

	Misc Info	60	61	62	63	64	65	66	67	68	69
	Text Columns	94	90	100	128	128	128	80	80	90	100
	Text Rows - 1	50	66	74	95	95	95	24	29	66	74
	Font Height	8	8	8	8	8	8	16	16	8	8
	Page Size	9700	BE00	EB00	6000	C000	>64K	FA00	>64K	>64K	>64K

	Sequencer	60	61	62	63	64	65	66	67	68	69
SR1	Clocking Mode	1	9	1	5	5	1	1	1	1	1
SR2	Plane Mask	F	F	F	3	F	F	F	F	F	F
SR3	Char Map Select	0	0	0	0	0	0	0	0	0	0
SR4	Memory Mode	6	6	6	2	2	6	E	E	E	E

	External Regs	60	61	62	63	64	65	66	67	68	69
MISC	Miscellaneous	E7	E7	EF	E7	E7	C7	43	C3	C7	C7

6/27/88

Table A - 3. (cont) Mode of Operation (Video Seven Proprietary Graphics Modes)

	CRT Ctrlr	60	61	62	63	64	65	66	67	68	69
CR0	H Total - 5	6D	6F	7F	A3	A3	A3	C6	C3	E3	F2
CR1	H Display End	5D	59	63	7F	7F	7F	9F	9F	B3	C7
CR2	H Blanking Start	5E	5A	64	83	83	82	A3	A2	B6	CA
CR3	H Blanking End	90	92	82	A5	A5	A6	86	85	85	94
CR4	H Retrace Start	61	5C	67	8D	8D	8D	AA	A8	B8	CB
CR5	H Retrace End	8F	85	18	82	82	82	00	00	8A	8E
CR6	V Total - 2	BF	47	77	29	29	29	E0	0B	47	70
CR7	Overflow	1F	F0	F0	FD	FD	FD	10	3E	F0	F0
CR8	Preset Row Scan	00	00	00	00	00	00	00	00	00	00
CR9	Char Cell Height	40	60	60	60	60	60	40	40	60	60
CRA	Cursor Start	00	00	00	00	00	00	00	00	00	00
CRB	Cursor End	00	00	00	00	00	00	00	00	00	00
CRC	Start Address H	--	--	--	--	--	--	--	--	--	--
CRD	Start Address L	--	--	--	--	--	--	--	--	--	--
CRE	Cursor Location H	--	--	--	--	--	--	--	--	--	--
CRF	Cursor Location L	--	--	--	--	--	--	--	--	--	--
CR10 (R)	Light Pen H	--	--	--	--	--	--	--	--	--	--
CR11 (R)	Light Pen L	--	--	--	--	--	--	--	--	--	--
CR10 (W)	V Retrace Start	A2	26	5C	07	07	07	CE	EA	26	59
CR11 (W)	V Retrace End	8E	08	0E	8A	8A	8A	85	8C	08	0C
CR12	V Display End	99	1B	57	FF	FF	FF	C7	DF	1B	57
CR13	Offset	2F	2D	32	20	20	40	50	50	5A	64
CR14	Underline Row	00	00	00	00	00	00	00	00	00	00
CR15	V Blanking Start	A1	24	5C	07	07	07	CC	E7	24	59
CR16	V Blanking End	B9	3F	74	22	22	22	DC	04	3F	6A
CR17	Mode Control	E3	E3	E3	EB	EB	E3	E7	E3	E3	E3
CR18	Line Compare	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

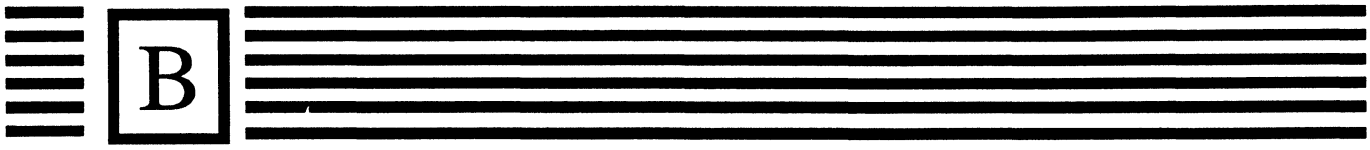
	Attribute Ctrlr	60	61	62	63	64	65	66	67	68	69
AR0	Palette 00	00	00	00	00	00	00	00	00	00	00
AR1	Palette 01	01	01	01	3F	3C	01	01	01	01	01
AR2	Palette 02	02	02	02	3F	00	02	02	02	02	02
AR3	Palette 03	03	03	03	3F	00	03	03	03	03	03
AR4	Palette 04	04	04	04	3F	3A	04	04	04	04	04
AR5	Palette 05	05	05	05	3F	3F	05	05	05	05	05
AR6	Palette 06	14	14	14	3F	00	14	06	06	06	06
AR7	Palette 07	07	07	07	3F	00	07	07	07	07	07
AR8	Palette 08	38	38	38	3F	00	38	08	08	08	08
AR9	Palette 09	39	39	39	3F	00	39	09	09	09	09
ARA	Palette 0A	3A	3A	3A	3F	00	3A	0A	0A	0A	0A
ARB	Palette 0B	3B	3B	3B	3F	00	3B	0B	0B	0B	0B
ARC	Palette 0C	3C	3C	3C	3F	00	3C	0C	0C	0C	0C
ARD	Palette 0D	3D	3D	3D	3F	00	3D	0D	0D	0D	0D
ARE	Palette 0E	3E	3E	3E	3F	00	3E	0E	0E	0E	0E
ARF	Palette 0F	3F	3F	3F	3F	00	3F	0F	0F	0F	0F
AR10	Mode Control	01	01	01	01	01	01	41	41	41	41
AR11	Overscan Color	00	00	00	00	00	00	00	00	00	00
AR12	Color Plane Ena	0F	0F	0F	01	05	0F	0F	0F	0F	0F
AR13	H Pixel Pan	00	00	00	00	00	00	00	00	00	00

	Graphics Ctrlr	60	61	62	63	64	65	66	67	68	69
GR0	Set/Reset	0	0	0	0	0	0	0	0	0	0
GR1	Enable Set/Reset	0	0	0	0	0	0	0	0	0	0
GR2	Color Compare	0	0	0	0	0	0	0	0	0	0
GR3	Data Rotate	0	0	0	0	0	0	0	0	0	0
GR4	Read Map Select	0	0	0	0	0	0	0	0	0	0
GR5	Mode	00	00	00	10	10	00	40	40	40	40
GR6	Miscellaneous	05	05	05	07	07	05	05	05	05	05
GR7	Color Don't Care	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F
GR8	Bit Mask	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

Table A - 4. CGA / MDA / Herc Register Initialization Tables (for reference only)

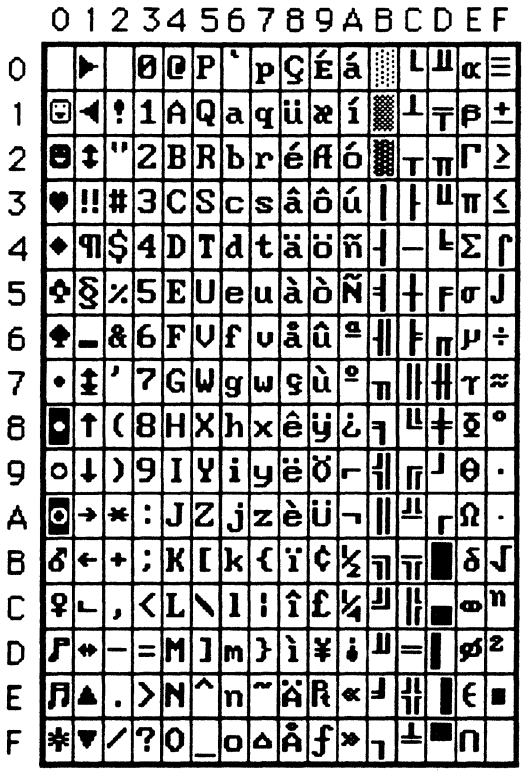
		Mode 0	Mode 1	Mode 2	Mode 3	Bricks Mode	Mode 4	Mode 5	Mode 6	Mode 7	
		CGA	CGA	CGA	CGA	CGA	CGA	CGA	CGA	MDA	Herc
		40x25 Mono Text	40x25 Color Text	80x25 Mono Text	80x25 Color Text	160x100 16-Color Graphics	320x200 Mono Graphics	320x200 4-Color Graphics	640x200 Mono Graphics	720x350 Mono Text	720x348 Mono Graphics
	Mode	2C	28	2D	29	09	0E/2A	0A/2E	1E	28/29	0A
	Color	30	30	30	30	30	30	30	3F	n/a	n/a
	Config (Full)	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	02	03
	Config (Half)	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	00	01
6845:											
R0	H Total - 1	38	38	71	71	71	38	38	38	61	35
R1	H Displayed	28	28	50	50	50	28	28	28	50	2D
R2	H Sync Pos	2D	2D	5A	5A	5A	2D	2D	2D	52	2E
R3	Sync Width	0A	0A	0A	0A	0A	0A	0A	0A	0F	07
R4	V Total - 1	1F	1F	1F	1F	7F	7F	7F	7F	19	5B
R5	V Adjust	06	06	06	06	06	06	06	06	06	02
R6	V Displayed	19	19	19	19	64	64	64	64	19	57
R7	V Sync Pos - 1	1C	1C	1C	1C	70	70	70	70	19	57
R8	Display Mode	02	02	02	02	02	02	02	02	02	02
R9	Cell Height - 1	07	07	07	07	01	01	01	01	0D	03
RA	Cursor Start	06	06	06	06	06	06	06	06	0B	00
RB	Cursor End	07	07	07	07	07	07	07	07	0C	00
RC	Start Addr H	00	00	00	00	00	00	00	00	00	00
RD	Start Addr L	00	00	00	00	00	00	00	00	00	00
RE	Cursor Addr H	00	00	00	00	00	00	00	00	00	00
RF	Cursor Addr L	00	00	00	00	00	00	00	00	00	00
	Page Size								4000	4000	
Pixels	Cell Size	8x8	8x8	8x8	8x8	8x2	8x2	8x2	16x2	9x14	16x4
Chars	H Displayed	40	40	80	80	80	40	40	40	80	45
Chars	H Total	57	57	114	114	114	57	57	57	98	54
Lines	V Displayed	200	200	200	200	200	200	200	200	343	348
Lines	V Total	262	262	262	262	262	262	262	262	370	370
MHz	Dot Clock	14.318	14.318	14.318	14.318	14.318	14.318	14.318	14.318	16.257	16.257
KHz	H Freq	15.70	15.70	15.70	15.70	15.70	15.70	15.70	15.70	18.43	18.82
Hz	V Freq	59.92	59.92	59.92	59.92	59.92	59.92	59.92	59.92	49.82	50.85

APPENDIX

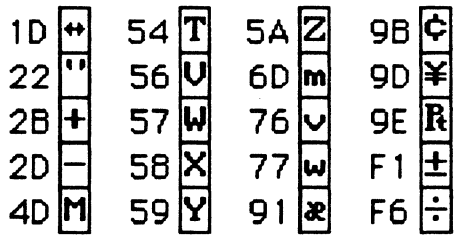


Character Fonts

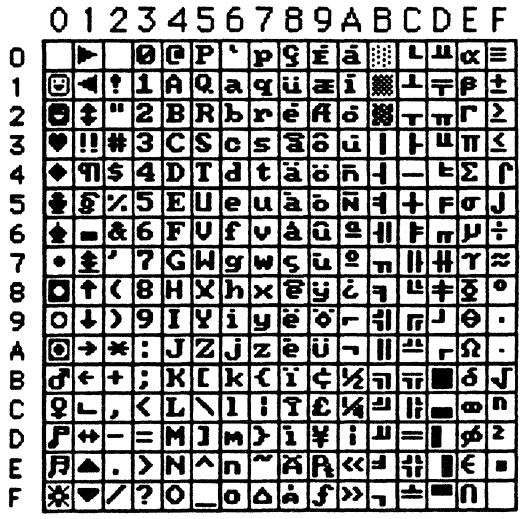
The following figure shows the 8x14 character set.



The following figure shows the monochrome (9x14) supplement to the 8x14 character set.



The following figure shows the 8x8 character set.



APPENDIX C

Connector Pinouts

Analog Video Connector

Pin	Name	Dir
1	Red (Analog 0-.7V)	Out
2	Green (Analog 0-.7V)	Out
3	Blue (Analog 0-.7V)	Out
4	MonID[2]	In
5	-reserved-	--

Pin	Name	Dir
6	Red Return	--
7	Green Return	--
8	Blue Return	--
9	(no pin)	--
10	Sync Return	--

Pin	Name	Dir
11	MonID[0]	In
12	MonID[1]	In
13	Hsync (TTL)	Out
14	Vsync (TTL)	Out
15	-reserved-	--

Feature Connector

Pin	Name	Dir
1	C0	I/O
2	C1	I/O
3	C2	I/O
4	C3	I/O
5	C4	I/O
6	C5	I/O
7	C6	I/O
8	C7	I/O
9	FEATCLK	I/O
10	BLANK*	I/O
11	HSYNC	I/O
12	VSYNC	I/O
13	Gnd	--

PCB Component Side

Pin	Name	Dir
14	Gnd	--
15	Gnd	--
16	Gnd	--
17	VIDENA	In
18	SYNCENA	In
19	CLKENA	In
20	-reserved-	--
21	Gnd	--
22	Gnd	--
23	Gnd	--
24	Gnd	--
25	-reserved-	--
26	-reserved-	--

PCB Solder Side

Note: The lowest numbered pin on each side of the board is located on the end nearest the Video Connector

PC/AT Bus Pinout Summary (shown viewed from top front of system motherboard)

Pin	Name	Direction
B1	GND	In
B2	RESET	In
B3	+5V	In
B4	IRQ2 (PC/AT IRQ9)	Out
B5	-5V	n/c
B6	DRQ2 (Floppy)	n/c
B7	-12V	In
B8	CARDSEL* (PC/AT OWS)	Out
B9	+12V	In
B10	GND	In
B11	MEMW*	In
B12	MEMR*	In
B13	IOW*	In
B14	IOR*	In
B15	DACK3*	n/c
B16	DRQ3 (Lowest Priority)	n/c
B17	DACK1*	n/c
B18	DRQ1 (Highest Priority)	n/c
B19	DACK0* (PC/AT REFRESH*)	In
B20	CLK (4.772727 MHz)	n/c
B21	IRQ7 (Lowest Priority - LPT1)	n/c
B22	IRQ6 (Floppy)	n/c
B23	IRQ5	n/c
B24	IRQ4 (Serial I/O 1)	n/c
B25	IRQ3	n/c
B26	DACK2* (Floppy)	n/c
B27	TC	n/c
B28	ALE	n/c
B29	+5V	In
B30	OSC (14.31818 MHz)	In
B31	GND	In

Pin	Name	Direction
A1	NMI*	Out
A2	D7	I/O
A3	D6	I/O
A4	D5	I/O
A5	D4	I/O
A6	D3	I/O
A7	D2	I/O
A8	D1	I/O
A9	D0	I/O
A10	WAIT*	Out
A11	AEN	In
A12	A19	In
A13	A18	In
A14	A17	In
A15	A16	In
A16	A15	In
A17	A14	In
A18	A13	In
A19	A12	In
A20	A11	In
A21	A10	In
A22	A9	In
A23	A8	In
A24	A7	In
A25	A6	In
A26	A5	In
A27	A4	In
A28	A3	In
A29	A2	In
A30	A1	In
A31	A0	In

Pin	Name	Direction
D1	MEMCS16*	Out
D2	IOCS16*	Out
D3	IRQ10	n/c
D4	IRQ11	n/c
D5	IRQ12	n/c
D6	IRQ13	n/c
D7	IRQ14	n/c
D8	DACK0*	n/c
D9	DRQ0	n/c
D10	DACK5*	n/c
D11	DRQ5	n/c
D12	DACK6*	n/c
D13	DRQ6	n/c
D14	DACK7*	n/c
D15	DRQ7	n/c
D16	+5V	In
D17	MASTER*	n/c
D18	Gnd	In

Pin	Name	Direction
C1	SHBE*	In
C2	LA23	In
C3	LA22	In
C4	LA21	In
C5	LA20	In
C6	LA19	In
C7	LA18	In
C8	LA17	In
C9	XMEMR*	n/c
C10	XMEMW*	n/c
C11	D8	I/O
C12	D9	I/O
C13	D10	I/O
C14	D11	I/O
C15	D12	I/O
C16	D13	I/O
C17	D14	I/O
C18	D15	I/O

Note: The C / D connector is only available on PC/AT-class systems (not on PC and PC/XT systems)

PS/2 (MicroChannel) Bus Pinouts (shown viewed from top front of system motherboard)

Pin	Name	Direction
B01	AUDIO GND	--
B02	AUDIO	--
B03	GND	GND
B04	14.31818MHZ	--
B05	GND	GND
B06	A23	In
B07	A22	In
B08	A21	In
B09	GND	GND
B10	A20	In
B11	A19	In
B12	A18	In
B13	GND	GND
B14	A17	In
B15	A16	In
B16	A15	In
B17	GND	GND
B18	A14	In
B19	A13	In
B20	A12	In
B21	GND	GND
B22	IRQ09*	--
B23	IRQ03*	--
B24	IRQ04*	--
B25	GND	GND
B26	IRQ05*	--
B27	IRQ06*	--
B28	IRQ07*	--
B29	GND	GND
B30	-reserved-	--
B31	-reserved-	--
B32	CHCK*	--
B33	GND	GND
B34	CMD*	In
B35	CHRDYRTN	--
B36	CDSFDBK*	Out
B37	GND	GND
B38	D01	I/O
B39	D03	I/O
B40	D04	I/O
B41	GND	GND
B42	RESET	In
B43	-reserved-	--
B44	-reserved-	--
B45	GND	GND
B46	-key-	no pin
B47	-key-	no pin
B48	D08	I/O
B49	D09	I/O
B5O	GND	GND
B51	D12	I/O
B52	D14	I/O
B53	D15	I/O
B54	GND	GND
B55	IRQ10*	--
B56	IRQ11*	--
B57	IRQ12*	--
B58	GND	GND

Pin	Name	Direction
A01	CDSETUP*	In
A02	MADE24	In
A03	GND	GND
A04	A11	In
A05	A10	In
A06	A09	In
A07	+5V	+5V
A08	A08	In
A09	A07	In
A10	A06	In
A11	+5V	+5V
A12	A05	In
A13	A04	In
A14	A03	In
A15	+5V	+5V
A16	A02	In
A17	A01	In
A18	A00	In
A19	+12V	--
A20	ADL*	In
A21	PREEMPT*	--
A22	BURST*	--
A23	-12V	--
A24	ARB00	--
A25	ARB01	--
A26	ARB02	--
A27	-12V	--
A28	ARB03	--
A29	ARB/GNT*	--
A30	TC*	--
A31	+5V	+5V
A32	S0*	In
A33	S1*	In
A34	M/IO*	In
A35	+12V	--
A36	RDY	Out
A37	D00	I/O
A38	D02	I/O
A39	+5V	+5V
A40	D05	I/O
A41	D06	I/O
A42	D07	I/O
A43	GND	GND
A44	DS16RTN*	--
A45	REFRESH*	In
A46	KEY	no pin
A47	KEY	no pin
A48	+5V	+5V
A49	D10	I/O
A50	D11	I/O
A51	D13	I/O
A52	+12V	--
A53	-reserved-	--
A54	SBHE*	In
A55	CDDS16*	Out
A56	+5V	+5V
A57	IRQ14*	--
A58	IRQ15*	--

APPENDIX

D

Jumper / Switch Settings

Dipswitches 1-3: Monitor Type (future - ignored in present BIOS revisions)

Sw1	Sw2	Sw3	Monitor	Modes Available	Max Res	Pitch	V (Hz)	H (KHz)	V/H Rtrc (usec)
-	-	-	Automatic Detect Color/Mono PS/2 Monitor	-	-	-	-	-	-
on	-	-	MultiSync (NEC),Nanao 8060S	all	800x540	.31mm	56-62 NP	15.5-35 NP	50/2 min
-	on	-	MultiScan (Sony),JVC,Thompson,Mitsubishi	all	800x540	.26mm	50-100 NP	15.5-34 NP	50/1 min
on	on	-	MultiSync Plus	all	960x720	.31mm	56-80 NP	21.8-45 NP	50/1 min
-	-	on	ACD (Analog Color Display)	0-7, D-13	640x480	.31mm	50-70 NP	31.5 NP	
on	-	on							
-	on	on							
on	on	on							

Dipswitch 4: 16-Bit I/O Enable (future - ignored in V7VGA chip revision 3 based boards)

on = Enabled **off = Disabled**

Dipswitch 5: CGA / MGA Emulation Enable

on = Enabled **off = Disabled**

Dipswitch 6: 16-Bit ROM Enable

on = Enabled **off = Disabled**

Dipswitch 7: 16-Bit Memory Enable

on = Enabled **off = Disabled**

Dipswitch 8: Pure Mode

on = Pure VGA Mode **off = V7VGA Extensions Enabled**

PCB Jumper Settings:

TST1 - DCLK / CCLK Test Connector - for factory test only, no jumper installed
Pin 1 = DCLK (dotclock), Pin 2 = Gnd, Pin 3 = CCLK (character clock)

JMP4 - Memory Enable	**1-2: Enable**	2-3: Disable	Default = 1-2
JMP3 - ROM Enable	**1-2: Enable**	2-3: Disable	Default = 1-2
JMP2 - Port 2E9 Enable	**1-2: Enable**	2-3: Disable	Default = 1-2
JMP1 - Interrupt Enable	1-2: Disable	**2-3: Enable**	Default = 2-3

Note: pin 1 is not connected for all jumpers JMP1-JMP4

P/N 700-0242