even in monochrome modes. The monochrome display has only two inputs and three useful combinations, but the palette is still useful for achieving additional attribute/color combinations.

The attribute controller generates underlining whenever the background RGB bits are 000 and the foreground RGB bits are 001. By properly setting the palette registers for colors 0 and 1 (or 8 and 9), it is possible to get black characters on a white background with underlining. The original monochrome adapter could not do this.

The attribute controller has a couple of other neat tricks up its sleeve. Bit 3 of the attribute mode control register (3C0.10) enables blinking for characters where attribute bit 7 is set. This is similar to the function provided by bit 5 of register 3x8 in the old adapters. When blinking is enabled, attribute bit 7 is treated as a 0 for the purposes of background color selection, so the background choice is limited to one of eight colors. However, unlike the old adapters, the EGA palette can be set for any color combination for any index. This removes the restriction against mixing bright and dim background colors when blinking is enabled.

The monochrome graphics function supports 640 pixels by 350 lines on a high-resolution monochrome display. Bit planes 0 and 2 are used, both residing at segment A000H. Two bits, one for each plane, define each pixel as follows:

Plane		
2	0	
0	0	Black
0	1	Video
1	0	Blinking video
1	1	Intensified video

Within each plane, the first byte defines the first eight pixels, the next byte the next eight, and so on sequentially for 28KB (640 pixels divided by 8 pixels per byte times 350 lines = 28,000 bytes). The high-order bit in each byte defines the first (leftmost) pixel, the low-order bit defines the last (rightmost). This is similar to the memory organization for the standard 640-by-200, one-color CGA mode, but there is no 8KB offset between even and odd scan lines. In monochrome graphics (as in all the new graphics modes) the memory organization is simply sequential.

The EGA has a feature that allows attribute bit 3 (normally the foreground intensity) to select an alternate character font table in the RAM character generator. This function is supported by BIOS, so it is not necessary to manipulate the hardware registers to achieve

TABLE 4: *BIOS Save Area Table Layout*

OFFSET	MEANING	
Dword 1	**Video Parameter Table Pointer** This pointer is initialized to BIOS EGA parameter table, and it must exist.	
Dword 2	**Dynamic Save Area Pointer** Initially 0000:0000, when nonzero this pointer is used to address a RAM area in which certain dynamic values are stored. For the Enhanced Graphics Adapter, the first 17 locations in the save area will contain the values of the 16 palette registers and the overscan color register. This save area must be at least 256 bytes long.	
Dword 3	**Alpha Mode Auxiliary Character Generator Pointer** Initially 0000:0000, when nonzero this points to a table with the following structure:	
	Byte	Bytes per character
	Byte	Char generator block to load (normally zero)
	Word	Number of character patterns to store (normally 256)
	Word	Character offset (normally 0)
	Dword	Pointer to a font table
	Byte	Displayable rows (FFH means display maximum possible)
	Byte	Consecutive bytes of mode values for which this font description is to be used, terminated with FFH.
Dword 4	**Graphics Mode Auxiliary Pointer** Initially 0000:0000, when nonzero this points to a table with the following structure:	
	Byte	Displayable rows
	Word	Bytes per character
	Dword	Pointer to a font table
	Byte	Consecutive bytes of mode values for which this font description is to be used, terminated with FFH.
Dword 5	Reserved, initially 0000:0000	
Dword 6	Reserved, initially 0000:0000	
Dword 7	Reserved, initially 0000:0000	
Dword 8	Reserved, initially 0000:0000	

The EGA BIOS's save area pointer points to various user-selected save areas in which BIOS stores the current values of additional parameters such as the palette registers. This allows programs to work with the current values used by BIOS.

TABLE 5: *Color Palette*

Palette register bit	7	6	5	4	3	2	1	0
Monochrome display	—	—	—	I	V	—	—	—
Standard color display	—	—	—	I	—	R	G	B
Enhanced color display	—	—	R′	G′	B′	R	G	B
Pin number in direct drive monitor connector			2*	6	7	3	4	5

** Pin 2 of direct drive monitor connector must be ground for standard color and IBM monochrome displays. Jumper P1 on the EGA selects R′ (1-2) or GND (2-3).*

The bits in the palette register are arranged as shown above for the three IBM monitors that can be used with the EGA. The last line of the table shows the corresponding pin positions on the direct drive monitor connector.

the effect. Listing 4 illustrates how to set up this particular mode.

COMPATIBILITY ISSUES

The EGA is highly compatible with the earlier display adapters and associated software, as long as certain rules are obeyed. But what are the rules?

The most important one is "Render unto BIOS that which is BIOS's." Anything that BIOS can do, let it do. Even simple tasks, such as setting palette registers, ought to be done through BIOS if future compatability is important. A fundamental purpose of BIOS is to give hardware designers the freedom to implement old functions in new ways. The color palette is an excellent example. Though the color register at port 3D9 no longer exists, BIOS calls to select a palette or border color in the compatible modes are emulated by the EGA BIOS and turned into the proper palette register settings.

IBM has a stated policy that memory mappings (from the processor's viewpoint) will remain the same for any given BIOS mode number. This does not mean that all modes will live forever in all future hardware, but it does guarantee consistency among like-numbered modes in different environments. The alpha mode mapping of characters in the even bytes and attributes in the odd bytes will surely be with us for many years to come.

The flip side of this mapping policy is that identical display appearances may be produced by different modes and mapping. For example, the PC*jr* supports medium-resolution, 16-color graphics as mode 9 with two pixels per byte. The EGA achieves the same pixel and color resolution with mode D, but with eight pixels per byte in each of four parallel bit planes. Each mode uses the same amount of physical memory (32KB), but the EGA is more economical of address space, using only 8KB.

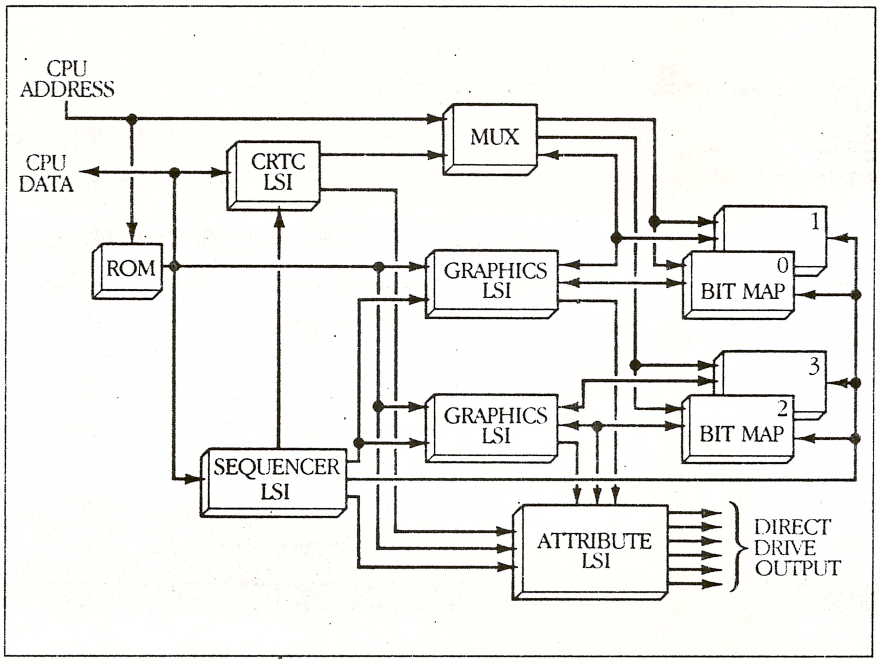
Besides the memory mapping for previously existing modes, the EGA makes some other concessions to compatibility. The light-pen hardware interface is identical, so programs that do not use BIOS to read the light pen should still work on the EGA. Also, the four low-order bits of the status register at 3xA have the same functions as in previous adapters, so programs that sense vertical or horizontal retrace can still work. It is no longer necessary to wait until retrace time to update the display memory without causing snow.

Existing software products were run on the EGA to test for compatibility at the highest possible level. The results were mixed.

BASIC 3.0 works correctly in all the compatible modes, but develops some odd characteristics in the new modes. In high-resolution color on the enhanced display, the cursor becomes a blinking *u* with an umlaut. This has been known to happen before when BASIC gets confused after switching display adapters from color to monochrome. In 43-line mode the cursor goes away completely, probably because BASIC initializes it to its own default values. BASIC thinks there are 25 lines, and it scrolls only the first part of the screen. The CLS statement gives four divide overflow messages, but Ctrl-Home clears the screen just fine. The BASIC graphics statements (LINE, CIRCLE, etc.) work only in the standard compatible modes.

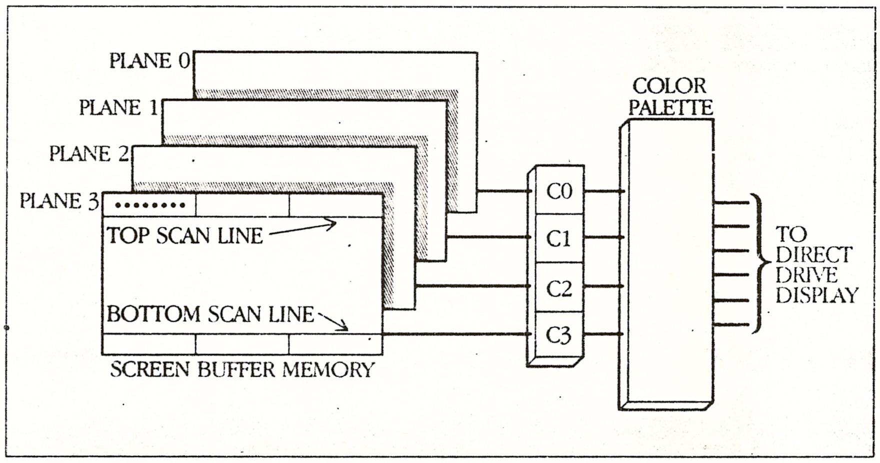
IBM has not announced, but we can still hope for, a version of BASIC that can make full use of new graphics

FIGURE 1: *Block Diagram of EGA*



The EGA has five custom large-scale integrated (LSI) circuits, reducing the chip count to 52 plus one crystal. The CGA, on the other hand, has 69 chips and no crystal, and the monochrome adapter has 66 chips and one crystal.

FIGURE 2: *Memory Organization for 16-color Graphics*



All planes are at the same processor address. Each plane is organized in the same manner: one bit per pixel, eight pixels per byte. Four bits (one from each plane) are combined in parallel to produce the color index into the palette registers. The output from the palette goes directly to the display monitor.

modes. Perhaps this will come with next DOS release. A version of the BASIC Compiler that supports the same functions, plus the enhancements added in revisions 2.0 and 3.0, is overdue.

Microsoft Flight Simulator made the screen go bonkers, probably because it was trying to set up the hardware directly to allow centering of the picture. PC Paint from Mouse Systems worked well, but it apparently does direct output to port 3D9H to change the color palette. Since the EGA has no port 3D9H, the colors never change.

Since software does not come with a "We Followed the Rules" certification, and the rules may change, the user should try his most favored combinations before committing to purchase.

IBM already has announced some graphics software packages that support the EGA and other IBM display adapters as well as graphics printers and plotters (see table 7 for prices). The Graphics Development Toolkit provides a device-independent facility for graphics applications development. It includes a Virtual Device Interface (VDI), which provides the device independence, and device drivers for many different graphic input and output devices.

The VDI support for the EGA does not take full advantage of the colors available, nor is it especially fast (the demo program does not seem especially quick at fills, for example). The point of the VDI, however, is not speed but device independence. Programs using VDI do not even have to be recompiled for new graphics devices or modes: the drivers are implemented as DOS 2.1 loadable device drivers. Unlike the BIOS interface, the VDI interface is well specified, frozen, and oriented toward graphics applications. Applications developers should find this a very attractive way to insure compatibility with a variety of devices, present and future.

A Graphical File System, Graphical Kernel System, Plotting System, and Graphics Terminal Emulator are also available from IBM.

The Enhanced Graphics Adapter is a powerful device, made with state-of-the-art technology. It offers good performance, flexibility, and value. It is more capable than competing graphics cards, but a little more expensive (in comparable configurations), and a little less compatible with the original CGA.

Third-party hardware manufacturers are not likely to run out and start producing EGA clones. First of all, IBM has a big investment in the custom LSI chips that smaller independent companies would have a tough time matching.

TABLE 6: *Color Mixing*

	STANDARD					ENHANCED					
COLOR	**I**	**–**	**R**	**G**	**B**	**R′**	**G′**	**B′**	**R**	**G**	**B**
Black	0	0	0	0	0	0	0	0	0	0	0
Blue	0	0	0	0	1	0	0	0	0	0	1
Green	0	0	0	1	0	0	0	0	0	1	0
Cyan	0	0	0	1	1	0	0	0	0	1	1
Red	0	0	1	0	0	0	0	0	1	0	0
Magenta	0	0	1	0	1	0	0	0	1	0	1
Brown*	0	0	1	1	0	0	1	0	1	0	0
White	0	0	1	1	1	0	0	0	1	1	1
Dark gray	1	0	0	0	0	1	1	1	0	0	0
Light blue	1	0	0	0	1	1	1	1	0	0	1
Light green	1	0	0	1	0	1	1	1	0	1	0
Light cyan	1	0	0	1	1	1	1	1	0	1	1
Light red	1	0	1	0	0	1	1	1	1	0	0
Light magenta	1	0	1	0	1	1	1	1	1	0	1
Yellow	1	0	1	1	0	1	1	1	1	1	0
Intensified white	1	0	1	1	1	1	1	1	1	1	1

* *Notice that in enhanced color mode, brown is composed of dark green and bright red.*

This table shows the palette register settings for the standard 16 IRGB colors in standard and enhanced color modes. When the PC is in the enhanced color mode, R'G'B' signals produce dim colors, RGB signals produce brighter colors, and a combination of the two produces the brightest colors.

TABLE 7: *Pricing and Related Products*

HARDWARE	
Basic EGA with 64KB	$524
Graphics Memory Expansion Card (64KB)	$199
Graphics Memory Module Kit (128KB)	$259
Fully loaded EGA with 256KB memory	$982
Enhanced Color Display	$849
Complete EGA System with 256KB and ECD	$1,831
SOFTWARE	
Graphics Development Toolkit	$350
Graphical File System	$175
Graphical Kernel System (GKS)	$295
Plotting System	$225
Graphics Terminal Emulator	$295

It costs more, but the EGA also offers more. Memory can be added in 64KB chips up to 256KB. IBM also has new graphics software for the EGA.

Second, the EGA is quite a bit more complex, and cloning will be a lot harder and take proportionately longer, by which time the clone may be obsolete.

The more likely course of events is for a continuing divergence of hardware implementations, supported by software interface standards such as the GKS, VDI, and even BIOS.

At $1,831 (see table 7), the complete package is expensive, mostly due to the cost of the Enhanced Color Display. Perhaps third-party display manufacturers will offer compatible displays for significantly lower cost, but even if they do not, the EGA has a lot to offer for users with standard color or monochrome displays.

Thomas Hoffmann is director of advanced systems development for General Instrument Corporation. He is a consulting editor whose last major article was a technical review of the PC/AT in the December 1984 issue.

SIDEBAR 1

EGA BIOS FUNCTIONS

EGA BIOS functions are accessed through interrupt 10H, with desired function code in AH, and other registers as detailed below.

AH = 0 Set mode
AL = mode (see table 2)
If bit 7 is set, video buffer will not be cleared (EGA only).

AH = 1 Set cursor mode
CH = start line for cursor (0 to 31)
CL = end line for cursor (0 to 31)

AH = 2 Set cursor position
BH = page number
DH,DL = row, column (0,0 is upper left)

AH = 3 Read cursor position
BH = page number
On exit:
DH,DL = row, column of current cursor
CH,CL = cursor mode currently set

AH = 4 Read light-pen position
On exit:
AH = 0: switch not closed, not triggered
AH = 1: valid light pen value in registers
DH,DL = row, column of LP character position
CH = raster line (0 to 199) in old graphics modes
CX = raster line (0 to nnn) in new graphics modes
BX = pixel column (0 to 319 or 639)

AH = 5 Select active display page
AL = new page value (see table 2 for max pages for each mode)

AH = 6 Scroll active page up (blank lines enter at bottom)

AH = 7 Scroll active page down (blank lines enter at top)
AL = number of lines (0 means entire window)
CH,CL = row,col of upper left corner
DH,DL = row, col of lower right corner
BH = attribute to be used on blank lines

CHARACTER HANDLING ROUTINES

For read/write character routines in CGA-compatible graphics modes (4, 5, and 6), the first 128-character code patterns are contained in system ROM, and the second 128-character code patterns are pointed to by the interrupt vector for INT 1F. The vector for INT 44 may be used to point to an alternate set of character patterns for character codes 0 to 127.

For the new graphics modes, 256-character patterns are supported in the system ROM, or through a pointer at the vector for INT 44.

AH = 8 Read attribute/character at current cursor position
BH = display page
On exit:
AL = character read
AH = attribute byte (alpha modes only)

AH = 9 Write attribute/character at current position
BH = display page
CX = count of characters to write
AL = character to write
BL = attribute (alpha) or color (graphics)
If bit 7 = 1, character is XORed onto screen

AH = A Write character only at current cursor position
BH = display page
CX = count of characters to write
AL = character to write
In graphics modes, replication count in CX works correctly only if all characters written are contained on the same row.

GRAPHICS INTERFACE

AH = B Set color palette (for use in CGA-compatible modes)
BH = palette color ID being set (0 to 127)
0 = background color (0 to 15)
1 = palette combination
value 0 = green/red/brown,
value 1 = cyan/magenta/white
BL = color value to be used
In alpha modes, the value for color ID 0 sets the border color (0 to 31) where 16 to 31 select high-intensity background color.

AH = C Write dot
BH = display page
DX,CX = row, column
AL = color value (if bit 7 set, value is XORed with current value of dot)

AH = D Read dot
BH = display page
DX,CX = row, column
On exit:
AL = color value of dot read

MISCELLANEOUS FUNCTIONS

AH = E Write character to active page (TTY emulation)
AL = character to write
BL = foreground color in graphics mode

AH = F Return current video state
On exit:
AL = mode (see table 2)
AH = number of character columns on screen
BH = current active display page

AH= 10 Set palette registers
AL = 0: set individual register
BL = palette register number
BH = value to set
AL = 1: set overscan register
BH = value to set
AL = 2: set all palette registers and overscan
ES:DX points to 17-byte array (P0 to P15, then overscan)
AL = 3: toggle intensify/blinking bit
BL = 0: enable intensity
BL = 1: enable blink

AH = 11 Character generator functions

The following functions will cause a mode set, completely resetting the video environment, but without clearing the video buffer.

AL = 00: load user-specifid patterns

```
            ES:BP   = pointer to user table
            CX      = count of patterns to
                      store
            DX      = character offset into map
                      2 block
            BL      = block to load (in map 2)
            BH      = number of bytes per
                      character pattern

AL      = 01: load ROM monochrome patterns (8
          by 14)
            BL      = block to load

AL      = 02: load ROM 8-by-8, double-dot
          patterns
            BL      = block to load

AL      = 03: set block specifier
            BL      = block specifier
          bits [3-2]: block for attr bit 3 = 1
          bits [2-0]: block for attr bit 3 = 0
          (Recommend calling INT 10 with
          AX=1000H and BX=0712H to set
          color plane enable register to ignore
          attribute bit 3 in addressing color pal-
          ette registers.)
```

The following routines (AL=1x) are designed to be called only immediately after a mode set, and are similar to AL=0x except that:

Page 0 must be active
Bytes/character is recalculated
Max character rows is recalculated
CRT buffer length is recalculated
CRTC registers are reprogrammed

```
R09    = bytes/char-1   max scan line
         (mode 7 only)
R0A    = bytes/char-2   cursor start
R0B    = 0              cursor end
R12                     vertical display end
       = ((rows+1)*(bytes/char))-1
R14    = bytes/char      underline loc
(*** BUG: should be 1 less ***)

AL     = 10: user alpha load
ES:BP  = pointer to user table
CX     = count of patterns to store
DX     = character offset into map 2
         block
BL     = block to load in map 2
BH     = number of bytes per character

AL     = 11: ROM monochrome set
BL     = block to load

AL     = 12: ROM 8-by-8 double-dot font
BL     = block to load
```

The following functions are meant to be called only immediately after a mode set.

```
AL     = 20: user 8-by-8 graphics charac-
         ters (INT 1F)
ES:BP  = pointer to user table

AL     = 21: user graphics characters
ES:BP  = pointer to user table
CX     = bytes per character
BL     = row specifier
           0: user set—DL = number of
              rows
           1: 14 rows
           2: 25 rows
           3: 43 rows

AL     = 22: ROM 8-by-14 set
BL     = row specifier

AL     = 23: ROM 8-by-8 double dot
BL     = row specifier
AL     = 30: return information
BH     = pointer specifier
         0: INT 1F pointer
         1: INT 44 pointer
         2: ROM 8-by-14 character font
            pointer
         3: ROM 8-by-8 double-dot font
            pointer
         4: ROM 8-by-8 DD font (top
            half)
            pointer
         5: ROM alpha alternate (9-by-
            14) pointer
On exit:
ES:BP  = specified pointer value
CX     = bytes/character
DL     = character rows on screen
```

AH = 12 Alternate function select

```
BL     = 10: return EGA information
BH     = 0: color mode in effect (3Dx)
         1: mono mode in effect (3Bx)
BL     = memory installed
         0: 64KB, 1: 128KB, 2: 192KB, 3: 256KB
CH     = feature bits
CL     = switch settings

BL     = 20: select alternate print screen rou-
         tine
```

AH = 13 Write string
CR, LF, backspace, and bell (07) are treated as commands, not printable characters.

```
ES:BP  = pointer to string to be written
CX     = character count
DH,DL  = row, column position to begin writ-
         ing
BH     = page number
AL     = 0:
         string = (char, char, char, ...)
         BL = attribute
         cursor is not moved
AL     = 1:
         string = (char, char, char, ...)
         BL = attribute
         cursor is moved
AL     = 2:
         string = (char, attr, char, attr, ...)
         cursor is not moved
AL     = 3:
         string = (char, attr, char, attr, ...)
         cursor is moved
```

SIDEBAR 2

EGA I/O REGISTER SUMMARY

Register descriptions are presented in the format:
Port.Index   Register name
    [bit (s)] Function
Port.Index notation indicates that the address register for the device must first be written with the index value, then data can be read or written at the data register I/O address.

EXTERNAL REGISTERS

3C2   Miscellaneous output (write only)

[0]	Select base I/O address (0=3Bx, 1=3DX)
[1]	Enable RAM (1) / Disable RAM (0)
[2-3]	Dot clock select
	00 = 14 mHz osc from I/O channel
	01 = 16 mHz on-board osc
	10 = external osc from feature
	11 = not used
[4]	Select output source (0=internal, 1=feature output)
[5]	Page bit for odd/even mode (0=low, 1=high)
[6]	Horizontal retrace polarity (0=pos, 1=neg)
[7]	Vertical retrace polarity (0=pos, 1=neg)
3C2	Status 0 (read only)
[4]	Switch sense (switch addressed by dot clock select)
[5-6]	Feature code input
[7]	CRT interrupt (if enabled: 0=vert retrace, 1=display)
3xA	Feature control (write only)
[0-1]	FC0, FC1
[2-3]	Reserved
3xA	Status 1 (read only)
	Any input 3C0 to attribute address
[0]	Display enabled (1) / vert or horiz retrace (0)
[1]	Light-pen latch (0=armed, 1=triggered)
[2]	Light-pen switch (0=close, 1=open)
[3]	Vertical retrace (1) / display active (0)
[5-6]	Diagnostic video output monitor
3xB	Clear light-pen latch (write only)
3xC	Set light-pen latch (write only)

ATTRIBUTE CONTROLLER (write only)

The attribute address register at 3C0 points to the attribute data register where data are to be written. An internal flip-flop switches between address and data registers. The flip-flop may be cleared to address mode by reading input status 1 at I/O address 3xA.

3C0	Attribute address / palette address source
[0-4]	Attribute address
[5]	Palette address source (0=processor, 1=display memory) (Must be 0 to load palette registers, 1 to enable display)
3C0.00-0F	Palette registers 0 to F (write only)
[0]	Blue
[1]	Green
[2]	Red
[3]	Blue' / mono video
[4]	Green' / intensity
[5]	Red'
3C0.10	Mode control
[0]	Graphics (1) / alphanumeric (0)
[1]	Monochrome attributes (1) / color attributes (0)
[2]	Enable line graphics codes (1) / ninth dot background (0)
[3]	Enable blink (1) / attr-7 = background intensity (0)
3C0.11	Overscan color
[0-5]	Same as palette registers
3C0.12	Color plane enable
[0-3]	Enable C0-C3 to attribute controller (1)
[4-5]	Video status mux (to status1 [4-5])
	00 = R B
	01 = B' G
	10 = R' G'
	11 = not used
3C0.13	Horizontal pixel panning
[0-3]	Number of pixels to shift video data left

SEQUENCER (write only)

3C4	Sequencer address
3C5.00	Reset (active low)
[0]	Async reset (0)
[1]	Sync reset (0)
3C5.01	Clocking mode
[0]	Dots per character (0=9, 1=8)
[1]	CRT bandwidth (0=high, 1=low)
[2]	Shift load (0=every character, 1=every second character)
[3]	Dot clock (0=normal, 1=halved)
3C5.02	Map mask
[0-3]	Enable CPU writes (1)
3C5.03	Character map select
[0-1]	Map B select (attr bit 3 = 0)
[2-3]	Map A select (attr bit 3 = 1)
	A and B must be different and extended memory installed to enable map select function for attr bit 3.
3C5.04	Memory mode
[0]	Alpha (1=enable char gen) / nonalpha (0)
[1]	Extended memory installed (1) / 64KB only (0)
[2]	Sequential addressing (1) / odd/even mode (0)

GRAPHICS CONTROLLER (write only)

3CC	Graphics 1 position (must be 00 for EGA)
3CA	Graphics 2 position (must be 01 for EGA)
3CE	Graphics 1 & 2 address
3CF.00	Set/reset
[0-3]	Values written to planes for which S/R is enabled when write mode is 0
3CF.01	Enable set/reset
3CF.02	Color compare
[0-3]	Color value for which read mode 1 returns a 1 in matching bit positions
3CF.03	Data rotate and function select for write mode 0
[0-2]	Rotate left count for write mode 0
[3-4]	Function select for write modes 0 and 2
	00 = unmodified
	01 = AND
	10 = OR
	11 = XOR
3CF.04	Read map select
[0-2]	Map number (encoded) for processor read mode 0
3CF.05	Mode register
[0-1]	Write mode
	00 = rotate processor data, apply function, write planes enabled (S/R planes get 8 bits from S/R register bit for that plane)