COMPUTER TECHNOLOGY

A high-resolution color graphic display four-channel software tachistoscope

EUGEN DIESCH

Technical University of Berlin, Berlin, Germany

A software tachistoscope for IBM-PC-compatible personal computers equipped with Tseng-ET4000-based VGA boards is described. It is shown how up to four bit-mapped color images with a resolution of up to 800×600 pixels may be stored in video display memory and alternately be selected for screen display in synchrony with the vertical retrace signal of the video controller.

Frequent research methods in perceptual and cognitive psychology require that stimuli be presented under precise temporal control, for short durations, and often in immediate temporal succession. These demands can be satisfied by implementations of such "tachistoscopic" displays on IBM-PC-compatible microcomputers if multiple virtual screens are made available, and if screen image onset, switching between screen images, and blanking of the screen is virtually instantaneous and synchronized with the vertical retrace signal of the video display hardware. Over several years, advances in the development of micro-computer graphic display technology have enabled contributions to this journal to address these problems in an increasingly satisfactory manner.

In the world of IBM PC compatibles, screen output is either character based or graphic, with character output being less demanding of the hardware than graphic output. Therefore, earlier software tachistoscopes for IBM PC compatibles (Bührer, Sparrer, & Weitkunat, 1987; Finley, 1989, 1991; Graves & Bradley, 1987, 1988; Heathcote, 1988; Segalowitz, 1987, 1988) were restricted to the presentation of character stimuli. However, there are research problems that require the tachistoscopic presentation of pictorial stimuli. More recently, Paredes, Miller, and Creeger (1990) and Haussmann (1992) have pointed out that enhanced graphics adapters (EGAs) and video graphics arrays (VGAs) support page switching between virtual screens defined in graphic display memory. The widely used Turbo family of compilers (Turbo Pascal, Turbo C), marketed by Borland International (Scotts Valley, CA), provides procedures (SetActivePage, SetVisual-

Page) to direct output to and to select among two or more virtual graphic screens. Unfortunately, this approach suffers from limitations both in the number of virtual screens and in the image resolution that it offers. There is a tradeoff between the number of virtual screens and image resolution. Even with only two virtual screens, the graphic resolution of the standard VGA is restricted to 640×350 pixels (video modes \$0F and \$10). At this degree of resolution, the line structure of the display screen is highly visible. Furthermore, tachistoscopes with only two fields cannot handle typical experimental trials with a fixation stimulus, a test stimulus, a visual masker, and a screen display for the intertrial interval immediately following each other. The latter problem may be circumvented with the use of four virtual screens, but this solution is bought at a price, because with four graphic pages the maximal image resolution attainable decreases to a maximum of 640×200 pixels (video mode \$0E).

Essentially, the problem is due to a restriction in the amount of memory that the standard VGA controller can access. However, many video controllers compatible with the VGA can access more memory and thus offer higher image resolution modes. The larger memory space available with these "super VGAs" (SVGA) can be used for the definition of a higher number of virtual graphic mode screens with higher image resolution. Unfortunately, there is no industry standard for such higher resolution display modes. Therefore, the software tachistoscope described here will run on computers with only one particular kind of graphics controller, the ET4000 produced by Tseng Laboratories (Taipei, Taiwan).

ET4000 Segmented Plane Configuration

The graphics controller chip ET4000 was introduced by Tseng Laboratories in 1989. Subsequently, it has been incorporated into a large number of video boards by third-party manufacturers. For example, the Turbo Mega Eva/1024 from Tseng Laboratories, the Genoa 7800 from the Genoa Systems Corporation, and the Orchid Prodesigner

The author gratefully acknowledges the helpful comments and suggestions made by the reviewers of an earlier version of this paper. Correspondence should be addressed to E. Diesch, Institute of Psychology, Technical University of Berlin, Dovestrasse 1-5, 10587 Berlin, Germany (e-mail: diesch@mailszrz.zrz.tu-berlin.de).

—Accepted by previous editor, N. John Castellan, Jr.

from Orchid Technology are all based on the ET4000. While being compatible with the IBM VGA controller, the ET4000 can access 1 MB of video display memory and offers additional video modes not available with the IBM VGA, some of them with high graphic resolution. The ET4000 allows for a number of color schemes and several configurations of video display memory. The memory configuration considered here is that of a segmented plane system with 16 colors.

In 16-color systems, four bits are needed for the representation of each pixel. Three bits encode the "primary colors" red, green, and blue. The remaining bit encodes intensity. In plane configurations, an (m,n) pixel array is composed of four superimposed (m,n) bit arrays or "bit planes," in such a way that the color of a pixel $p(x,y)$ is determined by the values of the bits $r(x,y)$, $g(x,y)$, $b(x,y)$, and $i(x,y)$ of the bit planes for red, green, blue, and intensity. Within each of the bit planes, each byte represents color information for a screen row segment eight pixels in length. The bit planes are read simultaneously by the video hardware in order to display their contents on the screen. Things are different for a program reading data from or writing data to video memory. Since the CPU cannot access the planes in parallel, the program must execute a plane select command before it can read from or write to a particular bit plane. A plane may be selected for reading by copying the integer value 4 to I/O port \$3CE and the integer code of the plane (0...3) to I/O port \$3CF. For writing, the plane is selected by copying the integer value 2 to I/O port \$3C4 and the integer value 2^{plane} (plane = 0...3) to I/O port \$3C5. One way to do this in Turbo Pascal is to use the predeclared array variable Port. Port is an array of type byte; its index variable is of type word. Input from an I/O port p is read by assigning the array element Port[p] to a variable. The Pascal statement for this is " $v := \text{Port}[p];$ ", where v is a variable of type byte. A value v is written to the I/O address (I/O channel) p by assigning v to Port[p] using the Pascal statement " $\text{Port}[i] := v;$ ".

One way of exchanging data between main memory and video display memory is to map video memory onto the address space of the CPU. In IBM-PC-compatible microcomputers equipped with VGA cards, a memory segment 64K in size and starting at memory location \$A0000 is usually reserved for this purpose. However, with a sum total of 1 MB of video display memory, each of four bit planes comprises 256K. For this reason, the planes are partitioned into 64K segments, and as a consequence of this, a program that reads from or writes to video memory not only has to select among planes, but also among segments. The segment selected for writing is determined by bits 0-3 of the video controller register accessible via I/O port \$3CD, and the segment for reading by bits 4-7.

In most video modes, the contents of only a relatively small proportion of the video display memory is displayed on the screen. The display window—the address range within video memory displayed at any one time—is defined by its start address and the size of the screen image

associated with the current video mode. In the graphic video mode \$12, with its image resolution of 640×480 pixels, 37.5K are required of each bit plane for the representation of one 16-color screen image. In video mode \$29, which offers a resolution of 800×600 pixels, this value increases to 58.6K. Thus, one bit plane of images with a resolution of either 640×480 pixels or 800×600 pixels can be represented in less than one segment of video memory. Since there are four segments per plane available in ET4000 plane configurations with four planes, it is an easy matter to set up four virtual screens. In fact, 1 MB of video memory is sufficient to support six virtual screens with an image resolution of 640×480 pixels each, but writing to those virtual screens would be complicated by multiple segment select operations. A virtual screen may be selected for actual display simply by changing the start address of the display area. One way to put this is to talk about the display window's being moved across video display memory. The way in which virtual screen switching may be synchronized with the vertical retrace signal of the video controller has been described by Segalowitz (1987, 1988), for example.

A linear addressing scheme is used for determining the position of the display window relative to the video display memory of the ET4000. The start address of the display window is determined by an 18-bit pointer. The display window can easily be moved across the available address range by changing its start address. This is done in stages. The start address is broken down into a low part for bits 0-7, a middle part for bits 8-15, and a high or extended part for bits 16 and 17. The low and the middle parts are copied to I/O port \$3D5 after copying \$0D and \$0C, respectively, to an index register accessible via I/O port \$3D4. Because bytes are the smallest unit of information directly handled by Intel CPUs, the high part of the starting address is determined by setting bits 0 and 1 of a variable (or constant) of type byte and copying it to I/O port \$3D5, this time after writing \$33 to the index register at I/O port \$3D4.

The Turbo Pascal Unit TSCOPE4

Code for a four-field tachistoscope using the segmented plane configuration of the ET4000 with 16 colors was packaged into a Turbo Pascal unit, *tscope4.pas*, which compiles under Turbo Pascal Version 5.5, 6.0, or 7.0. Turbo Pascal units are self-contained modules that allow for the exportation of procedures, functions, and data structures. The items exported can be referenced in a program that uses the unit. *Tscope4.pas* contains a complete set of procedures for the tachistoscopic display of bit-mapped images. Bit maps for display may be generated using a drawing program like Micrografx designer (Micrografx, Inc.) or a scanning device capable of producing image files in the widely used PCX file format. Together with procedures and functions implementing a software timer with millisecond resolution (e.g., Heathcote, 1988) or with the even more precise procedures of Bovens and Brysbaert (1990), which offer a resolution as good as

Table 1
Procedures Exported by Turbo Pascal Unit TSCOPE4

Procedure	Parameters	Effect
GetVideoMode	OldVideoMode	determines current video mode
SetVideoMode	VideoMode	initialization, sets a new video mode and clears all virtual screens
VideoOff		blanks the screen
VideoOn		turns the screen on
WaitForSync		synchronization with vertical retrace signal
ClearScreen	VirtualScreenNumber	selects a virtual screen for clearing
DisplayPage	VirtualScreenNumber	selects a virtual screen for display
InitImageBuffer	ImageBufferPointer, NumPlanes, NumLines, NumColumns	initializes a pointer to a data structure holding an image
DisposeImageBuffer	ImageBufferPointer, NumPlanes, NumLines, NumColumns	disposes of an image pointer
LoadImageBuffer	ImageFilename, ImageBufferPointer, NumPlanes, NumLines, NumColumns	reads an image file into the array pointed to by an image pointer variable
PutImageBuffer	ImageBufferPointer, SegmentNumber, NumPlanes, NumLines, NumColumns	moves the contents of an image buffer to video display memory
ReadPcxFile	ImageFilename, ImageBufferPointer, GrafDriver, NumPlanes, NumLines, NumColumns	reads a PCX file into the array pointed to by an image pointer variable

0.054 msec, tscope4.pas can easily be incorporated into a control program for running experiments with tachistoscopic display and response time measurement.

Table 1 lists the procedures that are declared in the interface part of the unit and therefore are visible to the user. Apart from these procedures, the unit exports a number of constants, which are summarized in Table 2. The two tables should be read and interpreted together. The constants VirtualScreen1 to VirtualScreen4 in Table 2 may be passed to ClearScreen and DisplayPage in Table 1. ClearScreen (VirtualScreen1) clears the first virtual screen. DisplayPage (VirtualScreen2) displays the contents of virtual screen number 2 on the physical screen. The VideoMode parameter passed to the SetVideoMode procedure in Table 1 may assume the values of the Pascal constants Graphics640x480, Graphics800x600, and TextScreen in Table 2. A call to SetVideoMode sets the values of the variables MaxLines and MaxColumns. Max-

Lines is scaled in units of pixels and MaxColumns in units of bytes. Calling SetVideoMode (Graphics800x600) returns MaxLines equal 600 and MaxColumns equal 100. When one is writing a program, it is sensible first to call SetVideoMode and then pass the returned values of MaxLines and MaxColumns to InitImageBuffer and, at a later stage, to DisposeImageBuffer. InitImageBuffer initializes the data structure for the storage of an image. Naturally, its size depends on the image resolution of the video mode activated by SetVideoMode. The amount of memory on the Pascal heap that is reserved by InitImageBuffer may be reclaimed by a subsequent call to DisposeImageBuffer. The procedure LoadImageBuffer reads a bit-mapped image file, places the image in the image buffer set up by InitImageBuffer, and returns the number of planes, lines (in pixels), and columns (in bytes) of which the image consists. The same thing is done for PCX files by ReadPcxFile. PutImageBuffer transfers the image buffer to one of the virtual screens in video memory. When one is calling PutImageBuffer, the values returned by LoadImageBuffer or ReadPcxFile for NumPlanes, NumLines, and NumColumns should be passed to the procedure.

Table 3 presents a simple example of how the unit may be used. At the start of the program, the screen is blanked by a call to VideoOFF. After that, a call to GetVideoMode is issued. GetVideoMode returns the integer value assigned to the video mode that was active at the time at which the program was started. This mode number is stored in the variable OldMode. Next, a high-resolution graphics mode is selected and initialized by SetVideoMode(Graphics800x600). Graphics800x600 is a constant declared in the interface part of tscope4.pas. It encodes the integer value of the segmented four plane 800×600 pixel 16-color display mode of the ET4000. SetVideoMode also clears all virtual screens by calls to ClearScreen. InitImageBuffer, which is called next, initializes the pointer variable b using the Turbo Pascal procedure Getmem. The pointer b points at an array of four pointers, each of which points at an array of byte with an array size of MaxLines × MaxColumns. This structure is used to represent the four bit planes of 16-color bit-mapped images. The first call to the procedure ReadPcxFile reads and decodes the file textons1.pcx. The resulting bit-mapped image is stored in the array structure pointed to by b. The first call to PutImageBuffer transfers this image to the first virtual screen. Because VideoOFF was called earlier on, the im-

Table 2
Constants Exported by Turbo Pascal Unit TSCOPE4

Constant	Value	Purpose
VirtualScreen1	0	parameter for DisplayPage & ClearScreen
VirtualScreen2	1	parameter for DisplayPage & ClearScreen
VirtualScreen3	2	parameter for DisplayPage & ClearScreen
VirtualScreen4	3	parameter for DisplayPage & ClearScreen
Graphics640x480	\$12	parameter for SetVideoMode
Graphics800x600	\$29	parameter for SetVideoMode
TextScreen	\$03	parameter for SetVideoMode
MaxLines	nn	determined by SetVideoMode
MaxColumns	nn	determined by SetVideoMode

Table 3
Example Program: Usage of TSCOPE4.PAS Procedures

```

Program Show4Pcx;

Uses Tscope4, ...;

Var b : BufPtrType; { BufPtrType is exported by tscope4 }
    i : Integer; OldMode : Byte;

begin

  VideoOFF;
  GetVideoMode (OldMode);
  SetVideoMode (Graphics800x600);

  InitImageBuffer (b, MaxLines, MaxPlanes, MaxColumns);

  ReadPcxFile ('textons1.pcx', b, ...);
  PutImageBuffer (b, 0, ...);
  ReadPcxFile ('textons2.pcx', b, ...);
  PutImageBuffer (b, 1, ...);
  ReadPcxFile ('textons3.pcx', b, ...);
  PutImageBuffer (b, 2, ...);
  ReadPcxFile ('textons4.pcx', b, ...);
  PutImageBuffer (b, 3, ...);

  DisplayPage (0); WaitForSync; VideoON; delay (3000);
  WaitForSync; DisplayPage (1); delay (3000);
  WaitForSync; DisplayPage (2); delay (3000);
  WaitForSync; DisplayPage (3); delay (3000);

  DisposeImageBuffer (b, MaxLines, MaxPlanes, MaxColumns);
  SetVideoMode (OldMode);

end.
```

age cannot yet be seen at this stage. Another three calls to ReadPcxFile and PutImageBuffer put three more PCX images onto the remaining virtual screens. As a result, four images reside in video memory. To display one of them, one need only move the display window to the appropriate virtual screen and enable screen display. DisplayPage selects one of the images, WaitForSync ascertains that display onset is synchronized with the vertical retrace signal, and the Turbo Pascal routine delay leaves the image on the screen for 3,000 msec. After all four images have been displayed in this way, the program terminates with calls to DisposeImageBuffer and SetVideoMode (OldMode). DisposeImageBuffer frees the memory assigned to the image buffer, and the call to SetVideoMode restores the original video mode.

Availability

The easiest way to obtain the source code of tscope4.pas, demonstration programs using the procedures contained therein, and rt.pas, a program for the running of experiments with randomized image presentation sequences, tachistoscopic display, and response time measurement, is via electronic mail. Please direct requests to diesch@mailszrz.zrz.tu-berlin.de. Alternatively, the code is available on floppy disk for a handling charge of \$15 (for the address, see the author's note). Please specify disk format (3.5 or 5.25 in.) and density (double or high).

REFERENCES

- BOVENS, N., & BRYSHAERT, M. (1990). IBM PC/XT/AT and PS/2 Turbo Pascal timing with extended resolution. *Behavior Research Methods, Instruments, & Computers*, **22**, 332-334.
- BÜHRER, M., SPARRER, B., & WEITKUNAT, R. (1987). Interval timing routines for the IBM PC/XT/AT microcomputer family. *Behavior Research Methods, Instruments, & Computers*, **19**, 327-334.
- ET4000 graphics controller data book (1990). Newtown, PA: Tseng Laboratories Inc.
- FINLEY, G. P. (1989). Tachistoscopic software for the Hercules display controller. *Behavior Research Methods, Instruments, & Computers*, **21**, 387-390.
- FINLEY, G. P. (1991). A VGA tachistoscope. *Behavior Research Methods, Instruments, & Computers*, **23**, 546-547.
- GRAVES, R., & BRADLEY, R. (1987). Millisecond interval timer and auditory reaction time programs for the IBM PC. *Behavior Research Methods, Instruments, & Computers*, **19**, 30-35.
- GRAVES, R., & BRADLEY, R. (1988). More on millisecond timing and tachistoscope applications for the IBM PC. *Behavior Research Methods, Instruments, & Computers*, **20**, 408-412.
- HAUSSMANN, R. E. (1992). Tachistoscopic presentation and millisecond timing on the IBM PC/XT/AT and PS/2: A Turbo Pascal unit to provide general-purpose routines for CGA, Hercules, EGA, and VGA monitors. *Behavior Research Methods, Instruments, & Computers*, **24**, 303-310.
- HEATHCOTE, A. (1988). Screen control and timing routines for the IBM microcomputer family using a high-level language. *Behavior Research Methods, Instruments, & Computers*, **20**, 289-297.
- PARADES, D. R., MILLER, K. F., & CREEGER, C. (1990). Graphic precision: Controlling stimulus displays on IBM PC-compatible computers. *Behavior Research Methods, Instruments, & Computers*, **22**, 319-322.
- SEGALOWITZ, S. J. (1987). IBM PC tachistoscope: Text stimuli. *Behavior Research Methods, Instruments, & Computers*, **19**, 383-388.
- SEGALOWITZ, S. J. (1988). IBM PC tachistoscope: II. Assembly language subroutines. *Behavior Research Methods, Instruments, & Computers*, **20**, 432.

(Manuscript received March 26, 1993;
 revision accepted for publication September 3, 1993.)