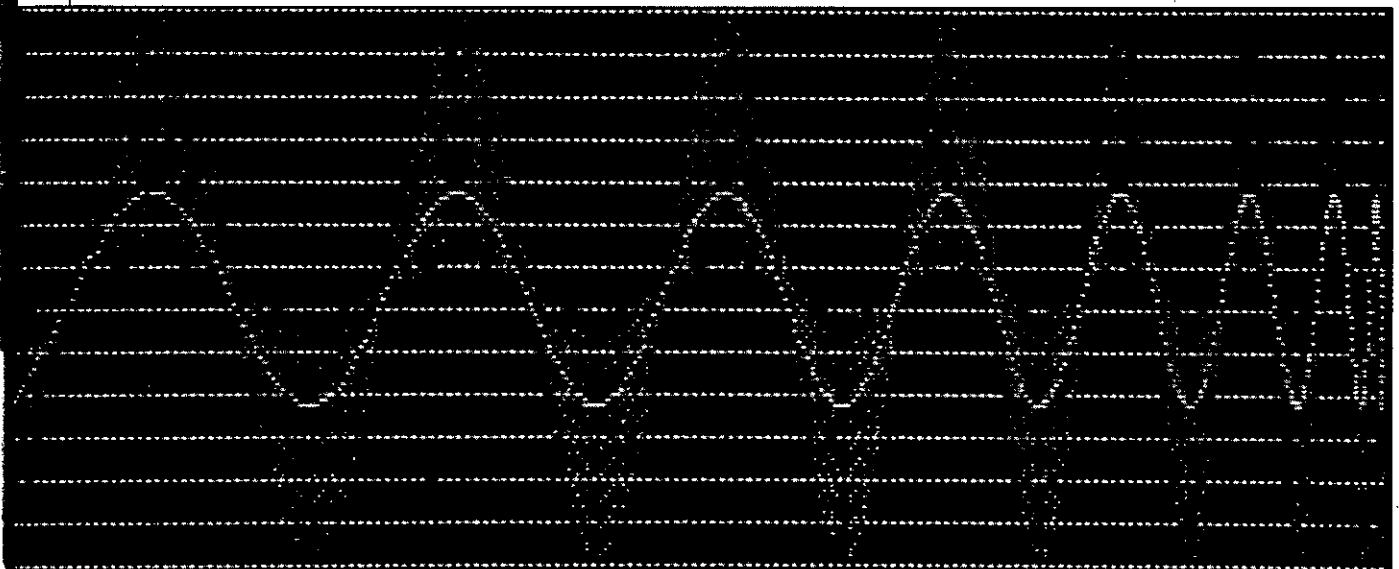


Plotting Data

An assembly language program for screen display of digitized waveforms

PETER G. AITKEN



The digital computer has become a common sight in scientific and engineering laboratories. The IBM PC is a good candidate for laboratory use (see "Passing the Lab Test," Peter G. Aitken, *PC Tech Journal*, January 1984, p. 74). One common application is to interface a PC to an analog-to-digital converter (ADC) so that electrical signals can be digitized and stored in the computer's memory.

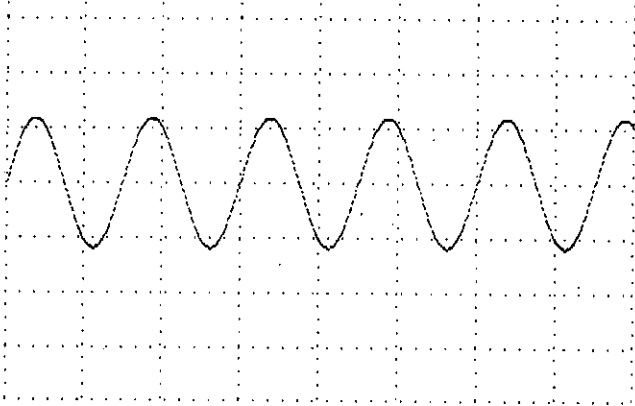
The result of such a process is usually a data array in which successive array elements contain digital numbers representing the measured signal voltage at successive time intervals. This data can be stored on disk, processed mathematically, or manipulated in other esoteric ways, such as to provide a graphic representation in which the digitized data are converted back to some sort of analog form closely resembling the original waveform. One way to do this is to use the computer monitor somewhat like an oscilloscope, where

each datum is displayed as a point on the screen with the time dimension represented horizontally and the voltage represented vertically. If the computer's graphics resolution is sufficient, that is, if the points can be placed sufficiently close together, the resulting display can be virtually indistinguishable from the original signal.

Duke University Medical Center in Durham, North Carolina, has two PCs, equipped with Tecmar Lab Master boards, that are used to digitize and store waveforms from biomedical experiments. The IBM color/graphics adapter, with its 640-by-200 high-resolution graphics mode, is well-suited for this purpose (although 400-point vertical resolution would be even better). The programs used, written in BASIC, could use the PSET statement in a FOR-NEXT loop to plot data points that were stored in an array. It takes about seven seconds to plot 640 points on the screen using interpreted BASIC; compil-

FIGURE 1: *Waveform With Grid*

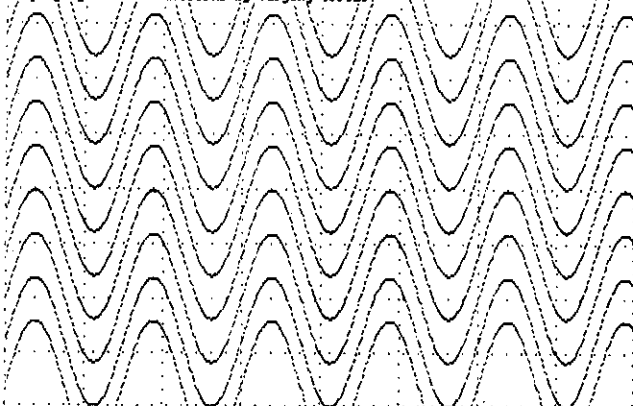
Entire waveform with grid..



This display is produced by PLOTTEST.BAS with the grid% parameter set to a nonzero value; if it is set to zero, no grid is plotted.

FIGURE 2: *Waveforms With Offset Variations*

Displaying several waveforms by varying OFFSET



These waveforms were produced using PLOTTEST.BAS and varying the offset% parameter, which may be set to any integer value.

ing the program reduces this time to about two seconds. Some of the applications, however, would benefit from even faster processing. To obtain maximum speed, an assembly language routine was written to do the plotting.

The first routine was written to take advantage of one of the PC's system resources, namely the type 10H video I/O interrupt. Interrupt 10H makes writing the routine quite straightforward, because it allows access to the system's bit-mapped graphics capability from assembly language without having to deal with all of the details of the graphics board's memory. This first attempt resulted in a tenfold speed improvement over compiled BASIC.

Yet, the source listing for the type 10 interrupt in IBM's *Technical Reference Manual* shows that the code is quite inefficient when used only to clear the screen or to write a dot in high-resolution mode. By incorporating customized routines for these two functions into the plotting program, an even greater speed advantage is realized.

The subroutine PLOT.ASM, shown in listing 1, is intended to be called from a BASIC program, but it could be easily modified for use with any language. Listing 2, PLOTTEST.BAS, is a BASIC program that demonstrates the use of the PLOT subroutine.

The video mode must be set to 640-by-200 graphics by the calling program. In addition, the routine requires that six parameters, which control various aspects of the display, be passed to it by the calling program. The parameters are all integer variables, and the call is as follows:

```
CALL PLOT (grid%,clr%,offset%,
          scale%, addr%,stp%)
```

The parameters that are passed to PLOT are described below.

grid%. If grid% is set to any nonzero value, a grid of vertical and horizontal lines is plotted along with the waveform as shown in figure 1. If it is set to zero, no grid is plotted.

clr%. If clr% is set to any nonzero value the screen is cleared. If it is set to zero the screen is left undisturbed.

The subroutine PLOT.ASM, though intended to be called from BASIC, can be modified for use with any language.

This option is used to superimpose several waveforms as illustrated in figures 2, 3, and 4.

offset%. This parameter specifies the offset of the vertical zero point and is normally set in the range 0-199, but may be set to any integer value. For example, if offset% is set to 100, data values of 0 will appear centered vertically on the screen; if set to 0, data values of 0 will appear at the top of the screen. Figure 2 illustrates the effects of varying this parameter.

scale%. The data values are divided by scale% before being plotted. This is necessary because integer data values can range between -32,768 and +32,767, while the graphics card uses a vertical plotting range of 0-199. In addition, since the vertical coordinates of the IBM board are inverted, with larger values being plotted toward the bottom of the screen, the program inverts the data back, so that

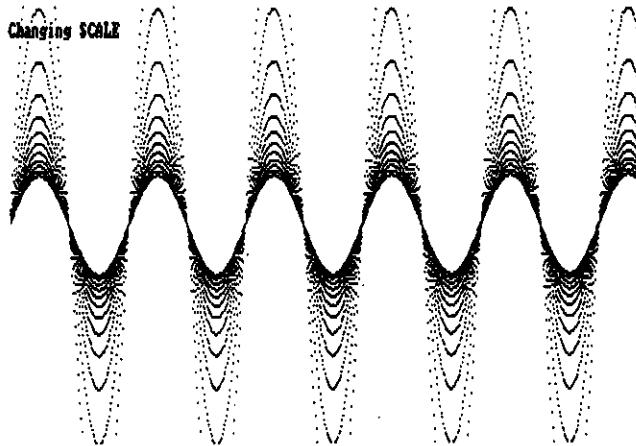
larger (positive) values are plotted at the top of the screen. If scale% is set to 0, the routine does not invert, divide, or offset the data. This is useful when the data in the array have already been converted to graphics coordinates: omitting the division step saves considerable time. Figure 3 shows the effects that can be obtained by varying scale%.

addr%. This parameter is the starting address of the data array to be plotted. The statement immediately preceding the CALL statement should be `addr%=VARPTR(x%(i,j,...))`, in which `x%(i,j,...)` is the first data array element to be plotted. For example, if a data array is dimensioned as `x%(2000,2,2)`, then setting addr% equal to `VARPTR(x%(200,2,2))` will result in elements `x%(200,2,2)` through `x%(840,2,2)` being plotted.

stp%. The horizontal resolution of the color/graphics adapter is 640 points—the maximum number of data points from one waveform that can be displayed at one time. The program allows the option of displaying fewer points, however; 320 or 160 data points may be displayed using the entire width of the screen. Stp% must be set equal to 1, 2, or 4; this value is not range-checked by the program. If set to 1, then 640 data points starting at addr% are plotted; if set to 2, then 320 are plotted; if set to 4, then 160 are plotted. In each case the plot fills the horizontal extent of the screen.

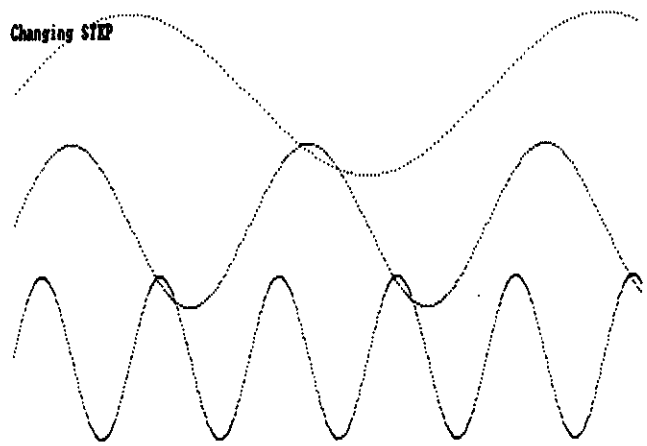
In addition to displaying waveforms collected with an ADC, the PLOT subroutine can be used to display data obtained via an RS-232 or IEEE-488 interface or data generated by some mathematical process such as process simulation. A graphics screen-dump

FIGURE 3: Waveform With Scale Variation



Waveforms like these can be produced using PLOTTEST.BAS and varying the value of the scale% parameter passed to PLOT.

FIGURE 4: Waveform With Step Variation




By setting the stp% parameter to 1, 2, or 4, waveforms with a varying number of data points can be produced with PLOTTEST.BAS.

program, such as GRAPHICS.COM supplied with PC-DOS 2.0, can be used to obtain a hard copy of the screen displays that are created by PLOT.

Execution speed could be slightly enhanced by placing the code that actually plots a point ("put_point") inside the plotting loop rather than being called as a subroutine. A call to a near procedure requires 19 clock cycles (byte operand) or 23 clock cycles

(word operand), and an intrasegment return requires 20 clock cycles if the stack is not popped. Since this procedure is called 640 times, eliminating these statements could save up to 5.8 milliseconds per plot (43 clocks times 640 calls times 210 nanoseconds/clock = 5.8 ms). The range-checking of Y could also be dispensed with, if the user is confident that the data will not go outside this range, for an additional

savings of up to 2.15 ms per plot. PLOT is, however, a reliable and useful program that potentially may find as much use in other circumstances and for other disciplines as it has already found in the laboratory at Duke University. 

Peter G. Aitken, Ph.D., works in the department of physiology at Duke University Medical Center in Durham, North Carolina. He is a computer consultant and has written several articles for this magazine.

SOFTWARE DEVELOPERS

Save thousands of dollars! Save hundreds of hours!
by using our assembly language sub-systems

B-TREE SUB-ROUTINES

FABS

Internationally known and used in many best selling application programs . . . Rapid access and maintenance of large files with fixed-length records . . . Versions available for CP/M-80, CP/M-86, MP/MII, MS DOS, PC DOS, Microsoft BASIC(s), COBOL, FORTRAN, PASCAL, PL/I, CBASIC, CB80, CB86, CBASIC 86, LATTICE C.

RETAILS FOR \$150 DEALER/OEM PRICES AVAILABLE

FABS PLUS

Expanded version of our FABS products . . . Up to millions of records DEP on Key Size . . . Extremely fast on unlimited number of keys . . . Re-indexing program included . . . Can be used on files as large as your system can hold.

RETAILS FOR \$195 DEALER/OEM PRICES AVAILABLE

SORT/MERGE SUB-ROUTINES

AUTOSORT

Optimized for very large files; stand-alone or callable sub-routine, extremely fast . . . Versions available for CP/M 80, CP/M 86, MP/MII, MS DOS, PC DOS running Microsoft BASIC(s), FORTRAN, PASCAL, CBASIC, CBASIC 86, CB80, CB 86, LATTICE C.

RETAILS FOR \$150 DEALER/OEM PRICES AVAILABLE

DATA, SCREEN, REPORT MANAGER

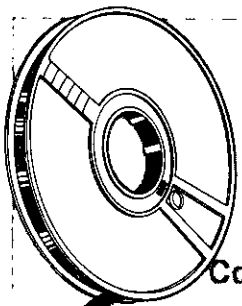
DB-FABS

A highly capable DATA BASE package designed to perform for everyone from the novice user in the Stand-Alone mode to the professional programmer in the Run-Time mode . . . Creates files, forms, reports, handles screening . . . B-Tree Indexing, high speed sorting capabilities . . . Run-Time mode use with BASIC INTERPRETER/COMPILER . . . For MS DOS, PC DOS on IBM PC/XT, DEC Rainbow, Victor 9000, Sanyo, Fujitsu, etc.

RETAILS FOR \$295 DEALER/OEM PRICES AVAILABLE

For more detailed information concerning any of our products, please contact us:

COMPUTER CONTROL SYSTEMS, INC
298 - 21st Terrace, S.E., Largo, Florida 33541 (813) 586-1886



9 TRACK TAPE CONTROLLER

New 1/2" Tape Controller for the IBM-PC

TC-PC is a high performance tape controller for the IBM-PC with these important features:

- Capable of reading and writing industry standard 1/2" tape
- Comprehensive software tools supplied
- 8 bit parallel recording with parity and read-after-write verification of data
- Compatible with most nine track formatted tape drives
- Operates with tape drive speeds up to 120 inches per second; allows data transfer rates of up to 192,000 bytes per second
- Economically priced at \$880

For more information on the TC-PC, call or write today.

Dealer/Distributor inquiries invited.

OVERLAND DATA, INC.
5644 Kearny Mesa Rd., Suite A
San Diego, CA 92111
Tel. (619) 571-5555

CIRCLE NO. 185 ON READER SERVICE CARD

Fortran Scientific Subroutine Package

Contains Approx. 100 Fortran Subroutines Covering:

- | | |
|----------------------------------|-----------------------------|
| 1. Matrix Storage and Operations | 7. Time Series |
| 2. Correlation and Regression | 8. Nonparametric Statistics |
| 3. Design Analysis | 9. Distribution Functions |
| 4. Discriminant Analysis | 10. Linear Analysis |
| 5. Factor Analysis | 11. Polynomial Solutions |
| 6. Eigen Analysis | 12. Data Screening |

Sources Included

\$295.00

FORLIB-PLUS™

Contains three assembly coded LIBRARIES plus support, FORTRAN coded subroutines and DEMO programs.

The three LIBRARIES contain support for GRAPHICS, COMMUNICATION, and FILE HANDLING/DISK SUPPORT. An additional feature within the graphics library is the capability of one fortran program calling another and passing data to it. Within the communication library, there are routines which will permit interrupt driven, buffered data to be received. With this capability, 9600 BAUD communication is possible. The file handling library contains all the required software to be DOS 3.0 PATHNAME compatible

STRINGS & THINGS™

Support for CHARACTER MANIPULATION (string support), SHELL, BATCH, MUSIC, CMD LINE, and ENVIRON CTRL.

\$69.95 each

P.O. Box 2517
Cypress, CA 90630



(714) 894-6808

California residents, please add 6% sales tax
Versions available for IBM Professional Fortran
or MICROSOFT 3.2 Fortran

PLOT

LISTING 1: PLOT.ASM

```

comment *
                                PLOT
                                Copyright (C) 1984 Dr. Peter G. Aitken
                                Department of Physiology
                                Duke University Medical Center, Durham, NC 27710

This subroutine uses the IBM color graphics board in the
640 X 200 high resolution mode to display analog waveforms
that are stored as a series of values in an integer array.
Format:  call plot (grid%,clr%,offset%,scale%,addr%,stp%)
if grid%<>0 a screen grid is plotted;
if grid%=0 no grid is plotted.

if clr%<>0 the screen is cleared;
if clr%=0 the screen is not cleared.

offset% sets the vertical offset of the display;
this is normally set in the range 0-199.

scale% sets the vertical gain of the display, with
smaller values of scale% giving more gain.

addr% is the address (i.e., offset within the
data segment) of the first array element to be
plotted.

stp% must equal 1, 2, or 4: it determines if
640 (step%=1), 320 (step%=2) or 160 (step%=4)
data points are plotted. Value not range checked.
*
;*****
sseg      segment stack          ;set up stack
          dw      50 DUP (?)
dseg      ends
          segment
          ;set up data segment with re-
          ;served memory locations for
          ;the 6 parameters to be
          ;passed by the calling program.
array     dw      (?)
divisor   dw      (?)
step      dw      (?)
off_set   dw      (?)
clear     dw      (?)
grid      dw      (?)
dseg      ends

video     segment at 08B00H
video     ends

cseg      segment public 'CODE'

          assume cs:cseg,ss:sseg,ds:dseg,es:video

          public PLOT          ;declare "PLOT" public
                                ;so it can be called
                                ;from another program,
                                ;save register

PLOT      proc far
          push bp

;the next block of code gets the 6 parameters passed
;by the calling program and stores them in the proper locations

          mov     bp,sp
          mov     si,[bp]+16
          mov     dx,[si]
          mov     grid,dx          ;1st parameter in GRID
          mov     si,[bp]+14
          mov     dx,[si]
          mov     clear,dx        ;2nd parameter in CLEAR
          mov     si,[bp]+12
          mov     dx,[si]
          mov     off_set,dx      ;3rd parameter in OFF_SET
          mov     si,[bp]+10
          mov     dx,[si]
          mov     divisor,dx      ;4th parameter in DIVISOR
          mov     si,[bp]+8
          mov     dx,[si]
          mov     array,dx        ;5th parameter in ARRAY
          mov     si,[bp]+6
          mov     dx,[si]
          mov     step,dx         ;6th parameter in STEP

          push   es              ;save es value
          mov    dx,08B00H      ;point es at video ram
          mov    es,dx

;if CLEAR = 0 jump ahead, if CLEAR <> 0 clear screen.

```

```

mov dx,clear
cmp dx,0
je no_clear

;the next 5 lines clear the screen by using the STOSW instruction
;to place 0 in all words of video memory.

mov cx,2000H ;video ram word count
mov ax,0
mov di,0 ;start at offset 0
cld ;set forward direction
rep stosw

;if GRID <=0 put a grid on the screen - else jump
no_clear: mov dx,grid
cmp dx,0
je no_grid

;first do horizontal grid lines

mov ax,199 ;plot line in row 199
call horiz
mov ax,175 ;plot line in row 175,
M1: call horiz ;150, etc.
sub ax,25
jns M1 ;if dx not < 0, do another

;now do vertical grid lines

mov cx,639 ;plot line in column 639
call vert
mov cx,560 ;plot line in column 560,
M2: call vert ;480, etc.
sub cx,80
jns M2 ;if cx not < 0, do another

;the next block of code initializes cx, which will count loops,
;and si, which is used as an offset pointer within the data array
no_grid: mov cx,640 ;count in cx
mov ax,1280 ;offset for start of plot
cld ;will be 1278 for step=1,
idiv step ;638 for step=2, and 318
mov si,ax ;for step=4.
sub si,2

;now we start to plot

mov bx,array ;put array base address
;in bx
plot_loop: sub cx,step ;subtract STEP from cx;
jcxz done ;if the result is less
;than 0, we're done.
mov ax,[bx][si] ;get array element
cmp divisor,0 ;if divisor=0 jump ahead
jz no_div
neg ax ;negate it
cld ;convert to double word
idiv divisor ;divide by scaling factor
add ax,off_set ;add the offset

;now ax has the row number, which can range from 0 to 199.
;The next 4 lines check to see if ax is within this range;
;if ax less than 0 or greater than 199 the point is not plotted.
no_div: cmp ax,0
jl skip
cmp ax,199
jg skip

call put_point

skip: sub si,2 ;decrement data pointer
;to point at next array
jmp plot_loop ;element go back and
;do another

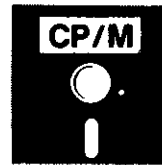
done: pop es
pop bp
ret 12

PLOT endp

;*****
;subroutines

; "horiz" places a row of dots across the screen, placing a

```



**Disk-to-Disk Transfer
with**

MEDIA MASTER™

MEDIA MASTER gives you instant access to CP/M and MS-DOS programs and data by allowing your IBM PC or XT to **READ, WRITE, and FORMAT** over 70 different computer formats.

The disk formats supported by the program include: DEC VT180, Osborne (DD), Morrow, IBM PC (CP/M-86, PC-DOS 1.0 & 2.0), Heath Z100, Heath w/ Magnolia CP/M, TI Professional CP/M, TRS-80 IV CP/M+, Xerox 820 II, NEC PC-8001A, Actrix, Kaypro II/2, Zenith Z90, and dozens more!

MEDIA MASTER allows you to make CP/M to MS-DOS and MS-DOS to CP/M file conversions, copy files, display or print directories, erase files, and type or print files using easy menu steps!

Ask for **MEDIA MASTER** at your local IBM dealer or send \$39.95+\$2.50 S/H. (CA residents please add 6%). To order COD (we pay all COD fees!) call our **24-HOUR TOLL FREE NUMBER: 800-824-7888**, and ask for Operator 251.



Dealer Inquiries Invited

4573 Heatherglen Ct, Moorpark, CA 93021
Technical Questions: 805-529-5073

CIRCLE NO. 162 ON READER SERVICE CARD

Multi-Tasking – Real Time Software System for IBM-PC ANDROMEDA™

with utility programs **PC-Debug™** and **C-Chain™**

- Designed for real time automation, data acquisition and monitoring applications.
- Offers full DOS compatibility or can run completely independent of DOS.
- Virtually unlimited in the number of tasks that can be run concurrently.
- Supports extensive split-screen/windowing for both console and serial connected terminals.
- Provides customizable modules for specific serial terminals, printers and for adding your own executive calls.
- May be ROM resident — requires only 12K of memory.
- Tasks are written in C language.

Special Introductory Price

Andromeda™ with manual and demonstration programs.....\$ 295.00
 Demonstration program only.....18.00
 PC-DEBUG™ optional module and manual.....95.00
 C-CHAIN™ optional module and manual.....95.00

Order by phone, ask for Sales Dept., Visa & MC accepted.



7392 Trade St. • San Diego, CA 92121 • (619) 566-7515

;dot in every 15th column. The calling program must pass the
;desired row number (0-199) in ax.

```

horiz      proc near
            mov     cx,630          ;starting column number
H1:        call    put_point
            sub     cx,15          ;decrement column number
            cmp     cx,0           ;if cx>=0 do another
            jge     H1
            ret
horiz      endp
    
```

;"vert" places a column of dots down the screen, placing a dot
;in every 6th row. The calling program must pass the desired
;column number (0-639) in cx

```

vert      proc near
            mov     ax,198         ;starting row number
V1:        call    put_point
            sub     ax,6          ;decrement row number
            cmp     ax,0           ;if ax>=0 do another
            jge     V1
            ret
vert      endp
    
```

;This subroutine puts a point on the high resolution graphics
;screen. Enter with Y (0-199) in ax, X (0-639) in cx, and es
;pointing to video ram (8000). All registers are preserved.

```

put_point proc near
            push    si             ;save all registers used
            push    bx
            push    cx
    
```

```

            push    ax
            push    cx             ;save cx and ax again for
            push    ax             ;program use

            and     ax,0FEh        ;strip odd/even bit of Y
            sal     ax,1
            sal     ax,1
            sal     ax,1           ;ax=ax times 8
            mov     bx,ax
            sal     ax,1
            sal     ax,1           ;ax=ax times 32
            add     ax,bx          ;ax=ax times 40
            pop     bx             ;now bx has original ax
            sar     bx,1           ;low bit into CF
            jnb     even           ;if low bit was 0, y is
            add     ax,2000H        ;even. if odd, adjust
            ;addresses
    
```

; now ax has 40*Y if Y is even and 40*Y+2000H if Y is odd -
;this is the address of the leftmost video byte in the row
;in which this point is to be put. We now must add (X/8) to
;get actual address

```

even:      sar     cx,1
            sar     cx,1
            sar     cx,1           ;cx=x/8
            add     ax,cx          ;now ax has final address
            mov     si,ax
            pop     cx             ;now cx has original x
            ;now to determine the bit to set
            and     cx,0007H      ;now bx has x mod 8
            mov     al,80H         ;start with mask 10000000B
            shr     al,cl          ;shift bit right to correct
            ;position
            or     al,es:[si]     ;put in any other bits
            ;that are already set in
            ;that byte of video ram
            mov     es:[si],al    ;put it back in video ram

            pop     ax
            pop     cx
            pop     bx
    
```

EXEC*U*STAT™

Advanced Software for Today's Executives

While you're still using your favorite spreadsheet package...
How would you like to have:

- Other Features:**
- SMOOTHING
 - PULL-DOWN CALCULATOR
 - SCRATCH PAD
 - CROSS TABULATION
 - TELEPHONE BOOK
 - NONLINEAR MODELS
 - SEASONAL ADJUSTMENT
 - ANALYSIS OF VARIANCE
 - SLIDE FACILITY
 - PLOTTERS
 - NET PRESENT VALUE
 - INTERNAL RATE OF RETURN

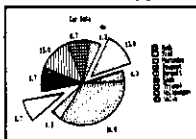
...and a lot more!

Make the executive move to

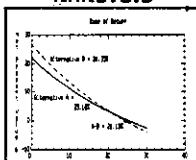
EXEC*U*STAT™
EXEC·U·STAT INC.

Research Park, 2 Wall Street Princeton, NJ 08540 (609) 924-9357

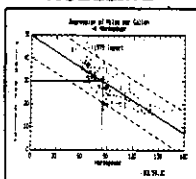
PRESENTATION GRAPHICS



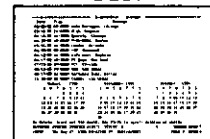
FINANCIAL ANALYSIS



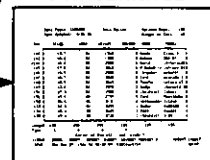
REGRESSION MODELING



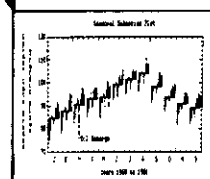
EXECUTIVE DESK



DATA MAINTENANCE



FORECASTING



```

pop si
ret

put_point endp
;*****
cseg ends
end

```

LISTING 2: PLOTTEST.BAS

```

' BASIC program to demonstrate PLOT assembly language routine.
'
' compile: BASCOM plottest/n
' link: LINK plottest+plot

40 cls : locate 15,10
input "Color/graphics adapter needed - continue (Y/N)";k$
if k$="N" or k$="n" goto 2000
if k$="Y" or k$="y" goto 50 else goto 40
50 dim x%(1280) : delay=6000 : screen 2

55 cls : locate 18,10
print "Now calculating waveform array..."
60 for i%=0 to 1280
q=sin(i%*3.14/60)
q=300*q
x%(i%)=int(q)
next

x$="Entire waveform without grid"
clr%=1 : grid%=0 : stp%=1 : scale%=10 : offset%=100
gosub 1000
gosub 1500

x$="Entire waveform with grid.."
clr%=1 : grid%=1 : stp%=1 : scale%=10 : offset%=100
gosub 1000
gosub 1500

x$="Displaying several waveforms by varying OFFSET"
clr%=1 : grid%=1 : stp%=1 : scale%=10
for offset%=10 to 190 step 20

```

```

gosub 1000 : clr%=0 : next offset%
gosub 1500

x$="Changing SCALE"
clr%=1 : grid%=0 : stp%=1 : offset%=100
for scale%= 13 to 2 step -1
gosub 1000 : clr%=0 : next scale%
gosub 1500

x$="Changing STEP"
clr%=1 : grid%=0 : scale%=8 : offset%=40
stp%=4 : gosub 1000
stp%=2 : clr%=0 : offset%=100 : gosub 1000
stp%=1 : clr%=0 : offset%=160 : gosub 1000
gosub 1500

x$="waveform scrolling with scale=0"
cls : locate 2,i : print x$
for i%=0 to 1280
x%(i%)=x%(i%)\6+100
next
gosub 1500
clr%=1 : grid%=0 : scale%=0 : offset%=100 : stp%=1
for i%=0 to 600 step 10
addr%=varptr(x%(i%))
call plot (grid%,clr%,offset%,scale%,addr%,stp%)
next

locate 24,34 : print "A=again E=exit"
900k$=inkey$ : if k$="" then goto 900
if k$="A" or k$="a" then goto 55
if k$="E" or k$="e" then goto 2000 else goto 900

1000 addr%=varptr(x%(0))
call plot (grid%,clr%,offset%,scale%,addr%,stp%)
locate 2,i : print x$ : return

1500' delay loop
for w=1 to delay
next w
return

2000 screen 0,0,0 : end

```

8087 AND 80287 TECHNICAL TOOLS

87FFT™ performs Forward and Inverse FFTs on real and complex arrays which occupy up to 512K bytes of RAM. Also does convolutions, auto correlations, hamming, complex vector multiplication, and complex to radial conversions. Callable from MS Fortran or 87BASIC/INLINE. \$150

87FFT-2™ performs two-dimensional FFTs. Ideal for image processing. Requires 87FFT...\$75

MATRIXPAK™ manages a MEGABYTE! Written in assembly language, our runtime package accurately manipulates large matrices at very fast speeds. Includes matrix inversion and the solution of simultaneous linear equations. Callable from MS Fortran 3.2, 87MACRO, 87BASIC/INLINE, and RTOS. each \$150

DATA ACQUISITION PACKAGE
Interactive, user-oriented language which allows the acquisition and analysis of large data streams. CALL

GRAPHICS PACKAGES
Energraphics (stand alone) 295
Grafmatic for MS Fortran or Pascal 125
Plotmatic for Grafmatic 125
Halo for Basic, C or Fortran each 150

OTHER TOOLS
Alpha Software ESP 595
Borland Sidekick, Toolbox, or Graphics 45
COSMOS Revelation 850

MAYNSTREAM
Maynard's portable streaming tape backup.
60 megabyte version 1695
Cartridge 50

87BASIC/INLINE™ converts the output of the IBM Basic Compiler into optimized 8087 inline code which executes up to seven times faster than 87BASIC. Supports separately compiled inline subroutines which are located in their own segments and can contain up to 64K bytes of code. This allows programs greater than 128K! Requires the IBM Basic Compiler and Macro Assembler. Includes 87BASIC \$200

DFixer
A disk utility which thoroughly checks PC or AT hard disks for bad sectors and updates the MS DOS file allocation table accordingly. 149

RTOS - REAL TIME OPERATING SYSTEM
RTOS is a multi-user, multi-tasking real time operating system. It includes a configured version of Intel's iRMX-86, LINK-86, LOC-86, LIB-86, OH-86, and MicroWay's 87DEBUG. Runs on the IBM-PC, XT, PC-AT and COMPAQ 400

INTEL COMPILERS™
FORTRAN-86 750
PASCAL-86 750
PL/M-86 500
87C (LATTICE/MICROWAY) 750
ASM-86 200

URS™ - Universal Run Time System™
Generates programs with the Intel compilers which run on other operating systems. MS-DOS version is included with RTOS.
Xenix-286 Version 300

SoftScope Symbolic Debugger™500
*Requires RTOS or iRMX-86. All Intel compiler names and iRMX-86 TM Intel Corp.

8087-3 5mhz..... \$149
Including DIAGNOSTICS and 180-day warranty
For IBM PC and compatibles

8087-2 8mhz..... \$275
For Wang, AT&T, DeskPro, NEC, Leading Edge

80287-3 5mhz..... \$275
For the IBM PC AT

64K RAM Set..... \$22

256K RAM Set..... \$135

128K RAM Set PC AT..... \$185

NUMBER SMASHER™...1590
9.5mhz 8087 coprocessor board for the IBM PC

FORTRAN and UTILITIES
Microsoft Fortran 3.2 229
IBM Professional Fortran 545
Intel Fortran-86¹ 750
FORLIB+ 65
STRINGS and THINGS 65

BASIC and UTILITIES
IBM Basic Compiler 270
87BASIC/INLINE 200
Summit BetterBASIC™ 175
Summit 8087 Module 87

MACRO ASSEMBLERS
IBM Assembler with Librarian 155
87MACRO 150
Microsoft Assembler V 3.0 125

PASCAL
Microsoft Pascal 3.2 209
Borland Turbo 35
Turbo with 8087 Support 85

APL
STSC APL★PLUS/PC 475
Pocket APL 85

C
Microsoft V 3.0 CALL
Lattice 299

Micro Way P.O. Box 79
Kingston, Mass.
02364 USA
(617) 746-7341

You Can Talk To Us!