

```

{***** V 8 0 6 0 P . P A S ****}
*
*-----*
* Task : Demonstrates programming in 800x600 pixel
* graphics mode of Super VGA cards with 16 colors.*
* This program requires the V8060PA.ASM assembly
* language module.*
*-----*
* Author : Michael Tischer
* Developed on : 01/14/91
* Last update : 03/03/92
*****}

```

```
program V8060P;
```

```
uses dos, crt;
```

```
{-- Type declarations -----}
```

```
type BPTR = ^byte;
```

```
{-- External references to the assembler routines -----}
```

```
{$L v8060pa} { Link assembler module }
```

```
function init800600 : boolean; external;
procedure setpix( x, y : integer; pcolor : byte ); external;
function getpix( x, y: integer ) : byte ; external;
```

```
{-- Constants -----}
```

```
const MAXX = 799; { Maximum X- and Y-coordinates }
```

```

MAXY      = 599;
NUMLINES  = 2500;                                { Number of lines }
XSPACING  = 40;                                { Distance of line box from margin }
YSPACING  = 30;
X1         = ( 2 * XSPACING );                  { Coordinates of line box }
Y1         = ( 2 * YSPACING );
X2         = ( MAXX-XSPACING );
Y2         = ( MAXY-YSPACING );

{ *****
*   IsVga : Determines whether a VGA card is installed.   *
**-----**
*   Input   : None                                         *
*   Output  : TRUE or FALSE                               *
*****}

function IsVga : boolean;

var Regs : Registers;                            { Processor registers for interrupt call }

begin
  Regs.AX := $1a00;                                { Function 1AH applies to VGA only }
  Intr( $10, Regs );
  IsVga := ( Regs.AL = $1a );
end;

{ *****
*   PrintChar : Writes a character to the screen while in graphic mode.*
**-----**
*   Input   :   THECHAR = Character to be written         *
*               x, y     = X- and Y-coordinates of upper-left corner *
*               FG       = Foreground color                *
*****}

```

```

*           BK           = Background color           *
*   Info    : Character is created in an 8x8 matrix, based on the   *
*           8x8 ROM font.                                           *
*****}

procedure PrintChar( thechar : char; x, y : integer; fg, bk : byte );

type FDEF = array[0..255,0..7] of byte;                { Font array }
TPTR = ^FDEF;                                           { Pointer to font }

var  Regs   : Registers;                                { Registers for interrupt call }
     ch     : char;                                     { Individual pixels in character }
     i, k,   { Loop counter }
     BMask  : byte;                                     { Bit mask for character design }

const fptr : TPTR = NIL;                                { Pointer to font in ROM }

begin
  if fptr = NIL then                                     { Pointer to font already set? }
    begin                                               { No }
      Regs.AH := $11;                                   { Call video BIOS function 11H, }
      Regs.AL := $30;                                   { sub-function 30H }
      Regs.BH := 3;                                     { Get pointer to 8x8 font }
      intr( $10, Regs );
      fptr := ptr( Regs.ES, Regs.BP );                  { Set pointers }
    end;

  if ( bk = 255 ) then                                  { Drawing transparent characters? }
    for i := 0 to 7 do                                  { Yes --> Set foreground pixels only }
      begin
        BMask := fptr^[ord(thechar),i]; { Get bit pattern for one line }
        for k := 0 to 7 do

```

```

begin
    if ( BMask and 128 <> 0 ) then                { Pixel set? }
        setpix( x+k, y+i, fg );                    { Yes }
        BMask := BMask shl 1;
    end;
end
else
    { No --> consider background as well }
    for i := 0 to 7 do                            { Execute lines }
        begin
            BMask := fptr^[ord(thechar),i];{ Get bit pattern for one line }
            for k := 0 to 7 do
                begin
                    if ( BMask and 128 <> 0 ) then    { Foreground? }
                        setpix( x+k, y+i, fg )      { Yes }
                    else
                        setpix( x+k, y+i, bk );      { No --> Background }
                        BMask := BMask shl 1;
                    end;
                end;
            end;
        end;
    end;
end;

```

```

{*****}
* Line: Draws a line based on the Bresenham algorithm. *
*-----*
* Input      : X1, Y1 = Starting coordinates (0 - ...) *
*             X2, Y2 = Ending coordinates              *
*             LPCOL = Color of line pixels              *
{*****}

```

```

procedure Line( x1, y1, x2, y2 : integer; lpcol : byte );

```

```

var d, dx, dy,

```

```

    aincr, bincr,
    xincr, yincr,
    x, y          : integer;

{-- Procedure for swapping two integer variables -----}

procedure SwapInt( var i1, i2: integer );

var dummy : integer;

begin
    dummy := i2;
    i2     := i1;
    i1     := dummy;
end;

{-- Main procedure -----}

begin
    if ( abs(x2-x1) < abs(y2-y1) ) then          { X- or Y-axis overflow? }
    begin                                         { Check Y-axis }
        if ( y1 > y2 ) then                      { y1 > y2? }
        begin
            SwapInt( x1, x2 );                    { Yes --> Swap X1 with X2 }
            SwapInt( y1, y2 );                    { and Y1 with Y2 }
        end;

        if ( x2 > x1 ) then xincr := 1           { Set X-axis increment }
        else xincr := -1;

        dy := y2 - y1;
        dx := abs( x2-x1 );
    end;
end;

```

```

d := 2 * dx - dy;
aincr := 2 * (dx - dy);
bincr := 2 * dx;
x := x1;
y := y1;

setpix( x, y, lpcol );
for y:=y1+1 to y2 do
begin
    if ( d >= 0 ) then
    begin
        inc( x, xincr );
        inc( d, aincr );
    end
    else
        inc( d, bincr );
        setpix( x, y, lpcol );
    end;
end
else
begin
    if ( x1 > x2 ) then
    begin
        SwapInt( x1, x2 );
        SwapInt( y1, y2 );
    end;

    if ( y2 > y1 ) then yincr := 1
        else yincr := -1;

    dx := x2 - x1;
    dy := abs( y2-y1 );

```

```

{ Set first pixel }
{ Execute line on Y-axes }

```

```

{ Check X-axes }

```

```

{ x1 > x2? }

```

```

{ Yes --> Swap X1 with X2 }
{ and Y1 with Y2 }

```

```

{ Set Y-axis increment }

```

```

d := 2 * dy - dx;
aincr := 2 * (dy - dx);
bincr := 2 * dy;
x := x1;
y := y1;

setpix( x, y, lpcol );
for x:=x1+1 to x2 do
begin
    if ( d >= 0 ) then
    begin
        inc( y, yincr );
        inc( d, aincr );
    end
    else
        inc( d, bincr );
    setpix( x, y, lpcol );
end;
end;

{ *****
* GrfxPrint: Displays a formatted string on the graphic screen. *
*-----*
* Input      : X, Y      = Starting coordinates (0 - ...)      *
*              FG        = Foreground color                    *
*              BK        = Background color (255 = transparent) *
*              STRING    = String with format information      *
*-----*
***** }
```

```

procedure GrfxPrint( x, y : integer; fg, bk : byte; strt : string );
```

```

var i : integer;                                { Loop counter }

begin
  for i:=1 to length( strt ) do
    begin
      printchar( strt[i], x, y, fg, bk );      { Display using PrintChar }
      inc( x, 8 );                             { Move X to next character position }
    end;
  end;

  {*****}
  * DrawAxis: Draws axes from left and top borders on the screen. *
  **-----**
  * Input   : STEPX = Increment for X-axis *
  *          STEPY = Increment for Y-axis *
  *          FG    = Foreground color *
  *          BK    = Background color (255 = transparent) *
  {*****}

  procedure DrawAxis( stepx, stepy : integer; fg, bk : byte );

  var x, y      : integer;                      { Loop coordinates }
      ordinate : string[3];

  begin
    Line( 0, 0, MAXX, 0, fg );                  { Draw X-axis }
    Line( 0, 0, 0, MAXY, fg );                  { Draw Y-axis }

    x := stepx;                                  { Scale X-axis }
    while ( x < MAXX ) do
      begin
        Line( x, 0, x, 5, fg );

```



```

    str( x, ordinate );
    if ( x < 100 ) then
        GrfxPrint( x - 8 , 8, fg, bk, ordinate )
    else
        GrfxPrint( x - 12, 8, fg, bk, ordinate );
    inc( x, stepx );
end;

y := stepy;                                { Scale Y-axis }
while ( y < MAXY ) do
    begin
        Line( 0, y, 5, y, fg );
        str( y:3, ordinate );
        GrfxPrint( 8, y-4, fg, bk, ordinate );
        inc( y, stepy );
    end;
end;

{ *****
* Demo: Demonstrates the functions and procedures in this module. *
*-----*
* Input      : None *
*-----*
*-----*
*****
}

procedure Demo;

var i : integer;                            { Loop counter }

begin
    Randomize;                               { Set random number generator in motion }
    DrawAxis( 30, 20, 15, 255 );            { Draw axes }
    GrfxPrint( X1, MAXY-10, 15, 255,

```

```

      'V8060P.PAS - (c) by Michael Tischer' );

Line( X1, Y1, X1, Y2, 15 );           { Draw border around line box }
Line( X1, Y2, X2, Y2, 15 );
Line( X2, Y2, X2, Y1, 15 );
Line( X2, Y1, X1, Y1, 15 );

{-- Create random lines within line box -----}

for i := 1 to NUMLINES do
  Line( random( X2 - X1 - 1 ) + X1 + 1,
        random( Y2 - Y1 - 1 ) + Y1 + 1,
        random( X2 - X1 - 1 ) + X1 + 1,
        random( Y2 - Y1 - 1 ) + Y1 + 1,
        i mod 16 );
end;

{-----}
{--                               M A I N   P R O G R A M                               ----}
{-----}

begin
  writeln( 'V8060P.PAS - (c) 1992 by Michael Tischer'#13#10 );
  if IsVga then                                { VGA card installed? }
    begin                                     { Yes --> but can graphics mode also be initialized? }
      if init800600 then
        begin                                { Mode O.K. }
          Demo;                               { Execute Demo }
          repeat until keypressed;           { Wait for key }
          Textmode( CO80 );                  { Shift into text mode }
        end
      else
    end;
  end;
end;

```

```
        writeln( '800x600 mode could not be initialized!' );
    end
    else
        writeln( 'This program requires a VGA card', + #13#10);
    end.
•
```