

```

{***** V 3 2 2 0 P . P A S *****)
*-----*
* Task      : Demonstrates programming in 320x200 VGA      *
*            graphic mode, using 256 colors and four screen *
*            pages. This program requires the V3220PA.ASM    *
*            assembly language module.                      *
*-----*
* Author    : Michael Tischer                               *
* Developed on : 09/08/90                                    *
* Last update  : 02/13/92                                    *
*****}

program V3220P;

uses dos, crt;

{-- Type declarations -----}

type BPTR = ^byte;

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

{$L v3220pa}                                { Link assembler module }

procedure init320200; external;
procedure setpix( x, y : integer; pcolor : byte ); external;
function getpix( x, y : integer ) : byte ; external;
procedure setpage( page : byte ); external;
procedure showpage( page : byte ); external;

{-- Constants -----}

const MAXX = 319;                            { Maximum X- and Y-coordinates }
      MAXY = 199;

{*****
* 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;
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;

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

        setpix( x, y, lpcol ); { Set first pixel }
        for y:=y1+1 to y2 do { Execute line on Y-axes }
        begin
            if ( d >= 0 ) then
            begin
                inc( x, xincr );
                inc( d, aincr );
            end
            else
                inc( d, bincr );
            setpix( x, y, lpcol);
        end;
    end;
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 );
    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;
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;

begin
    for i:=1 to length( strt ) do
    begin
        printchar( strt[i], x, y, fg, bk );
        inc( x, 8 );
    end;
end;

{ *****
* ColorBox: Draws a rectangle and fills it with a line pattern.
* *****
* Input      : X1, Y1 = Upper-left coordinates of window
*              X2, Y2 = Lower-right coordinates of window
*              COLMAX = Greatest color value
* Info       : Line colors are selected in a cycle of 0-COLMAX.
* ***** }

procedure ColorBox( x1, y1, x2, y2 : integer; colmax : byte );

var x, y,
    sx, sy : integer;

begin
    Line( x1, y1, x1, y2, 15 );
    Line( x1, y2, x2, y2, 15 );
    Line( x2, y2, x2, y1, 15 );
    Line( x2, y1, x1, y1, 15 );

    for y := y2-1 downto y1+1 do
    Line( x1+1, y2-1, x2-1, y, y mod colmax );

    for y := y2-1 downto y1+1 do
    Line( x2-1, y2-1, x1+1, y, y mod colmax );

    {-- From center of box to top border -----}
    sx := x1+ (x2-x1) div 2;

```

```

sy := y1 + (y2 - y1) div 2;
for x := x1 + 1 to x2 - 1 do
    Line( sx, sy, x, y1 + 1, x mod colmax );
end;

{*****}
* DrawAxis: Draws axes from left and top borders on the screen. *
{*****}
* Input      : XSTEP = Increment for X-axis *
*             YSTEP = 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 }
    lpcoords : 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, lpcoords );
            if ( x < 100 ) then
                GrfxPrint( x - 8, 8, fg, bk, lpcoords )
            else
                GrfxPrint( x - 12, 8, fg, bk, lpcoords );
            inc( x, stepx );
        end;

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

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

procedure Demo;

const PAUSE = 100;                  { Pause counter in milliseconds }

var page : byte;                    { Page counter }

begin
    for page := 0 to 4 do
        begin
            setpage( page );        { Process page }
            ColorBox( 80, 25, 308, 175, ( page + 1 ) * 16 ); { Paint box }
            DrawAxis( 30, 20, 15, 255 ); { Draw axes }
            GrfxPrint( 46, MAXY - 10, 15, 255,
                'V3220P - (c) by Michael Tischer' );
        end;

    {-- Display four graphic pages in sequences-----}

    for page := 0 to 50 do          { 50 executions }
        begin
            showpage( page mod 4 ); { Display page }
            delay( PAUSE );          { Brief pause }
        end;
    end;

{*****}
* MAIN PROGRAM *
{*****}

begin
    if IsVga then                    { VGA card installed? }
        begin
            init320200;              { Yes --> Go ahead }
            Demo;                    { Initialize graphic mode }
            repeat until keypressed;
        end;
end;

```

```
Textmode( C080 );                                { Shift into text mode }
end
else
writeln( 'V3220P - (c) 1992 by Michael Tischer'#13#10#10 +
        'This program requires a VGA card', #13#10);
end.
```