

```

/*****
*
*          V 3 2 4 0 C . C
*
**-----**
* Task      : Demonstrates programming in 320x400 VGA
*            graphic mode, using 256 colors and four screen
*            pages. This program requires the V3240CA.ASM
*            assembly language module.
**-----**
* Author    : Michael Tischer
* Developed on : 09/04/90
* Last update : 02/26/92
**-----**
* Memory model : SMALL
**-----**
* (MICROSOFT C)
* Compilation  : CL /AS /c /W0 V3240C.C
*              : LINK V3240C V3240CA;
**-----**
* (BORLAND TURBO C)
* Compilation  : Create a project file using the following:
*              v3240c.c
*              v3240ca.asm
**-----**
* Call       : v3240c
*****/

```

```

#include <dos.h>
#include <stdarg.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>

```

```

/*-- Type delarations -----*/

```

```

typedef unsigned char BYTE;

```

```

/*-- External references to the assembler routines -----*/

```

```

extern void init320400( void );
extern void setpix( int x, int y, unsigned char pcolor);
extern BYTE getpix( int x, int y );
extern void setpage( BYTE page );
extern void showpage( BYTE page );
extern void far * getfontptr( void );

```

```

/*-- Constants -----*/

```

```

#define MAXX 319          /* Maximum X- and Y-coordinates */
#define MAXY 399

```

```

/*****
* IsVga: Determines whether a VGA card is installed.
**-----**
* Input   : None
* Output  : 0 If no VGA exists, otherwise < 0
*****/

```

```

BYTE IsVga( void )
{
    union REGS Regs;          /* Processor registers for interrupt call */

    Regs.x.ax = 0x1a00;        /* Function 1AH applies to VGA only */
    int86( 0x10, &Regs, &Regs );
    return ( Regs.h.al == 0x1a ); /* Is the function available? */
}

```

```

/*****
* 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.
*****/

```

```

void PrintChar( char thechar, int x, int y, BYTE FG, BYTE BK )
{
    typedef BYTE FDEF[256][8];          /* Font definition */
    typedef FDEF far *TPTR;              /* Pointer to font */

    BYTE          i, k,                  /* Loop counter */
                 BMask;                  /* Bit mask for character design */

    static TPTR fptr = (TPTR) 0;         /* Pointer to font in ROM */

    if ( fptr == (TPTR) 0 )              /* Pointer to font already set? */
        fptr = getfontptr(); /* No --> Use the assembler function to load */

    /*- Generate character pixel by pixel -----*/

    if ( BK == 255 )                    /* Drawing transparent characters? */
        for ( i = 0; i < 8; ++i )      /* Yes --> Set foreground pixels only */
        {
            BMask = (*fptr)[thearchar][i]; /* Get bit pattern for one line */
            for ( k = 0; k < 8; ++k, BMask <= 1 ) /* Execute column */
                if ( BMask & 128 ) /* Pixel set? */
                    setpix( x+k, y+i, FG ); /* Yes */
        }
    else /* No --> Consider background as well */
        for ( i = 0; i < 8; ++i ) /* Execute lines */
        {
            BMask = (*fptr)[thearchar][i]; /* Get bit pattern for one line */
            for ( k = 0; k < 8; ++k, BMask <= 1 ) /* Execute columns */
                setpix( x+k, y+i, ( BMask & 128 ) ? FG : BK );
        }
    }

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

    /*-- Function for swapping two integer variables -----*/

    void SwapInt( int *i1, int *i2 )
    {
        int dummy;

        dummy = *i2;
        *i2 = *i1;
        *i1 = dummy;
    }

    /*-- Main part of function -----*/

    void Line( int x1, int y1, int x2, int y2, BYTE pcolor )
    {
        int d, dx, dy,
            aincr, bincr,
            xincr, yincr,
            x, y;

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

            xincr = ( x2 > x1 ) ? 1 : -1; /* Set X-axis increment */

            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, pcolor );
for (y=y1+1; y<= y2; ++y )
{
    if ( d >= 0 )
    {
        x += xincr;
        d += aincr;
    }
    else
        d += bincr;
    setpix(x, y, pcolor);
}
}
else
{
    if ( x1 > x2 )
    {
        SwapInt( &x1, &x2 );
        SwapInt( &y1, &y2 );
    }
}

yincr = ( y2 > y1 ) ? 1 : -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, pcolor);
for (x=x1+1; x<=x2; ++x )
{
    if ( d >= 0 )
    {
        y += yincr;
        d += aincr;
    }
    else
        d += bincr;
    setpix(x, y, pcolor);
}
}

/*****
* GrfxPrintf: 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
*              ...       = Arguments similar to printf
*****/

void GrfxPrintf( int x, int y, BYTE FG, BYTE BK, char * string, ... )
{
    va_list parameter;
    char stngbuf[255],
    *cp;

    va_start( parameter, string );
    vsprintf( stngbuf, string, parameter );
    for ( cp = stngbuf; *cp; ++cp, x+= 8 )
        PrintChar( *cp, x, y, FG, BK );
}

/*****
* 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.           *
*****/

void ColorBox( int x1, int y1, int x2, int y2, int colmax )
{
    int x, y,                                     /* Loop variables */
        sx, sy;                                  /* Exit point for last color loop */

    Line( x1, y1, x1, y2, 15 );                  /* Draw border */
    Line( x1, y2, x2, y2, 15 );
    Line( x2, y2, x2, y1, 15 );
    Line( x2, y1, x1, y1, 15 );

    for ( y = y2-1; y > y1; --y )                /* Bottom left to right border */
        Line( x1+1, y2-1, x2-1, y, y % colmax );

    for ( y = y2-1; y > y1; --y )                /* Bottom right to left border */
        Line( x2-1, y2-1, x1+1, y, y % colmax );

    /*-- From center of box to top border -----*/

    for ( x=x1+1, sx=x1+(x2-x1)/2, sy=y1+(y2-y1)/ 2; x < x2; ++x )
        Line( sx, sy, x, y1+1, x % colmax );
}

/*****
* 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)              *
*****/

void DrawAxis( int stepx, int stepy, BYTE FG, BYTE BK )
{
    int x, y;                                     /* Loop coordinates */

    Line( 0, 0, MAXX, 0, FG );                   /* Draw X-axis */
    Line( 0, 0, 0, MAXY, FG );                   /* Draw Y-axis */

    for ( x = stepx; x < MAXX; x += stepx )      /* Scale X-axis */
    {
        Line( x, 0, x, 5, FG );
        GrfxPrintf( x < 100 ? x - 8 : x - 12, 8, FG, BK, "%d", x );
    }

    for ( y = stepy; y < MAXY; y += stepy )      /* Scale Y-axis */
    {
        Line( 0, y, 5, y, FG );
        GrfxPrintf( 8, y-4, FG, BK, "%3d", y );
    }
}

/*****
* Demo: Demonstrates the functions in this module.                       *
*****/

void Demo( void )
{
    #define PAUSE 100000                        /* Pause amount, varies with system */

    int x;                                     /* Coordinate counter */
    long delay;                               /* Pause counter */

    ColorBox( 80, 50, 308, 350, 16 );          /* Paint box */
    DrawAxis( 30, 40, 15, 255 );              /* Draw axes */
    GrfxPrintf( 46, MAXY-10, 15, 255,
        "V3240C - (c) by Michael Tischer");

    setpage( 1 );                              /* Process page */
    ColorBox( 80, 50, 308, 350, 255 );        /* Paint box */
    DrawAxis( 30, 40, 15, 255 );              /* Draw axes */
    GrfxPrintf( 46, MAXY-10, 15, 255,
        "V3240C - (c) by Michael Tischer" );
}

```

```

/*-- Display four graphic pages in sequence -----*/
for ( x = 0; x < 50; ++x )          /* 50 executions */
{
    showpage( x % 2 );              /* Display page */
    for ( delay = 1; delay < PAUSE; ++delay )    /* Brief pause */
        ;
}

/*****
**                               M A I N   P R O G R A M                               **
*****/

void main( void )
{
    union REGS regs;

    if ( IsVga() )                  /* VGA card installed? */
    {
        init320400();              /* Yes --> Go ahead */
        Demo();                    /* Initialize graphic mode */
        getch();                   /* Wait for a key */
        regs.x.ax = 0x0003;        /* Shift into text mode */
        int86( 0x10, &regs, &regs );
    }
    else
        printf( "V3240C - (c) 1992 by Michael Tischer\n\n"\
                "This program requires a VGA card\n\n" );
}

```