

```

/*****
*                               V 1 6 C O L C . C                               *
**-----**
* Task                          : Demonstrates programming in EGA and VGA graphic *
*                               modes using 16 colors. This program requires      *
*                               the V16COLCA.ASM assembly language module.        *
**-----**
* Author                        : Michael Tischer                               *
* Developed on                  : 12/20/90                                       *
* Last update                   : 02/29/92                                       *
**-----**
* Memory model                  : SMALL                                           *
**-----**
* (MICROSOFT C)                                                         *
* Compilation                   : CL /AS /c /W0 v16colc.c                      *
*                               LINK v16colc.c v16colca;                        *
**-----**
* (BORLAND TURBO C)                                                     *
* Compilation                   : Create a project file containing the following: *
*                               v16colc.c                                       *
*                               v16colca.asm                                    *
**-----**
* Call                          : v16colc                                       *
**-----**
* Info                          : Turbo C warning "Unreachable code ... " is not *
*                               an error.                                       *
*****/

```

```

#include <dos.h>
#include <stdarg.h>
#include <stdlib.h>
#include <io.h>

```

```

#include <stdio.h>
#include <conio.h>

/*-- Type declarations -----*/

typedef unsigned char BYTE;

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

extern void init320200( void );
extern void init640480( void );
extern void init640350( void );
extern void init640200( void );
extern void setpix( int x, int y, unsigned char pcolor);
extern BYTE getpix( int x, int y );
extern void setpage( int page );
extern void showpage( int page );
extern void far * getfontptr( void );

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

#define A320200 1 /* Possible resolutions and modes */
#define A640200 2
#define A640350 3
#define A640480 4

#define VMODE A640350 /* Specify constants for desired mode */

#define EGA 0 /* Card types */
#define VGA 1
#define NEITHERNOR 2

```

```

/*- Global variables -----*/

int  MaxX,                      /* Maximum X- and Y-coordinates */
     MaxY;
BYTE Pages;                    /* Number of screen pages */

/*****
*  IsEgaVga : Determines whether an EGA or a VGA card is installed.  *
**-----**
*  Input   : None                                                    *
*  Output  : EGA, VGA or NEITHERNOR                                  *
*****/

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

    Regs.x.ax = 0x1a00;         /* Function 1AH applies only to VGA */
    int86( 0x10, &Regs, &Regs );
    if ( Regs.h.al == 0x1a )     /* Function available? */
        return VGA;
    else
    {
        Regs.h.ah = 0x12;       /* Call function 12H, */
        Regs.h.bl = 0x10;       /* sub-function 10H */
        int86(0x10, &Regs, &Regs ); /* Call video BIOS */
        return (BYTE) (( Regs.h.bl != 0x10 ) ? EGA : NEITHERNOR);
    }
}

/*****
*  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 array */
    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 assembler function */

    /*- Draw 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 a line */
            for ( k = 0; k < 8; ++k, BMask <= 1 ) /* Execute columns */
                if ( BMask & 128 )            /* Pixel set? */
                    setpix( x+k, y+i, fg );    /* Yes */
        }
}

```

```

else                                                    /* No --> Set every pixel */
for ( i = 0; i < 8; ++i )                               /* Execute lines */
{
    BMask = (*fptr)[thechar][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 the line pixels
*****/

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

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

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

/*-- Main section of function -----*/

void Line( int x1, int y1, int x2, int y2, BYTE lpcol )
{

```

```

int d, dx, dy,
    aincr, bincr,
    xincr, yincr,
    x, y;

if ( abs(x2-x1) < abs(y2-y1) )          /* X- or Y-axis overflow? */
{
    /* Check Y-axes */
    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, lpcol );
    for (y=y1+1; y<= y2; ++y )
    {
        if ( d >= 0 )
        {
            x += xincr;
            d += aincr;
        }
        else

```

```

        d += bincr;
        setpix(x, y, lpcol);
    }
}
else
{
    /* Check X-axes */
    if ( x1 > x2 )
    {
        /* x1 > x2? */
        SwapInt( &x1, &x2 );
        SwapInt( &y1, &y2 );
        /* Yes --> Swap X1 with X2 */
        /* and Y1 with Y2 */
    }

    yincr = ( y2 > y1 ) ? 1 : -1;
    /* Set Y-axis increment */

    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; x<=x2; ++x )
    {
        /* Set first pixel */
        /* Execute line on X-axes */
        if ( d >= 0 )
        {
            y += yincr;
            d += aincr;
        }
        else
            d += bincr;
    }
}

```

```

        setpix(x, y, lpcol);
    }
}
}

```

```

/*****
*   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 are similar to printf
*****/

```

```

void GrfxPrintf( int x, int y, BYTE fg, BYTE bk, char * string, ... )
{
    va_list parameter;          /* Parameter list for VA_... macros */
    char stngbuf[255],          /* Buffer for formatted string */
        *cp;

    va_start( parameter, string );          /* Convert parameters */
    vsprintf( stngbuf, string, parameter ); /* Format */
    for ( cp = stngbuf; *cp; ++cp, x+= 8 ) /* Display formatted */
        PrintChar( *cp, x, y, fg, bk );    /* string using PrintChar */
}

```

```

/*****
*   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, (BYTE) (y % colmax) );

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

```

[illegible]

```

BYTE pgcount;                                /* Page counter */
long delay;                                  /* Pause counter */

for ( pgcount = 1; pgcount <= Pages; ++pgcount )
{
    setpage( pgcount-1 );                    /* Process page */
    showpage( pgcount-1 );                  /* Process page */
    ColorBox( 50+pgcount*2, 40, MaxX-50+pgcount*2, MaxY-40, 16 );
    DrawAxis( 30, 20, 15, 255 );           /* Draw axes */
    GrfxPrintf( 46, MaxY-10, 15, 255,
                "V16COLC - (c) by Michael Tischer" );
}

/*-- Display graphic pages in sequences -----*/

for ( x = 0; x < 50; ++x )                  /* 50 executions */
{
    showpage( x % Pages );                  /* 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;

    printf( "V16COLC.C - (c) 1990, 92 by Michael Tischer\n\n" );

```

```

if ( VMODE == A640480 )                /* VGA mode selected? */
{
    if ( IsEgaVga() != VGA )           /* Yes */
    {                                  /* No */
        printf( "This program requires VGA 640x480 mode\n" );
        exit(1);                      /* End program */
    }
else                                   /* Yes --> Initialize mode and set parameters */
{
    MaxX = 639;
    MaxY = 479;
    Pages = 1;
    init640480();
}
}
else                                  /* Must be in an EGA mode */
{
    if ( IsEgaVga() == NEITHERNOR )    /* Neither EGA nor VGA card? */
    {                                  /* No */
        printf( "This program requires an EGA or a VGA card\n" );
        exit(1);                      /* End program */
    }
else                                   /* Yes --> Initialize mode and set parameters */
    switch( VMODE )
    {
        case A320200 : {                /* 320x200 pixels */
            MaxX = 319;
            MaxY = 199;
            Pages = 8;
            init320200();
            break;
        }
    }
}

```

```

        case A640200 : {
                                /* 640x200 pixels */
                                MaxX = 639;
                                MaxY = 199;
                                Pages = 4;
                                init640200();
                                break;
                                }
        case A640350 : {
                                /* 640x350 pixels */
                                MaxX = 639;
                                MaxY = 349;
                                Pages = 2;
                                init640350();
                                }
    }

    Demo();
    getch();
    regs.x.ax = 0x0003;
    int86( 0x10, &regs, &regs );
}

```

•