

```

/*****
*                               L O G O C . C                               *
**-----**
*   Task                       : Demonstrates custom character definition for  *
*                               EGA and VGA cards, for use as a logo design.   *
**-----**
*   Author                     : Michael Tischer                             *
*   Developed on                : 08/06/90                                   *
*   Last update                 : 03/02/92                                   *
**-----**
*   (MICROSOFT C)
*   Compilation                 : CL /AS /c /W0 LOGOC.C                     *
*                               LINK LOGOC LOGOCA;                           *
**-----**
*   (BORLAND TURBO C)
*   Compilation                 : Create a project file containing the following: *
*                               logoc.c                                       *
*                               logoca.asm                                    *
**-----**
*   Call                        : LOGOC                                       *
**-----**
*****/

```

```

#include <dos.h>                                /* Add include files */
#include <stdarg.h>
#include <stdio.h>
#include <string.h>
#include <conio.h>

#ifdef __TURBOC__                                /* Compiling with Turbo C? */
#define CLI()          disable()
#define STI()          enable()
#define outpw( p, w ) outport( p, w )

```

```

#ifdef inp
#define outp( p, b ) outportb( p, b )
#define inp( p ) inportb( p )
#endif
#else
/* No --> QuickC 2.0 or MSC */
#include <conio.h>
#define MK_FP(seg,ofs) ((void far *)\
(((unsigned long)(seg) << 16) | (ofs)))

#define CLI() _disable()
#define STI() _enable()
#endif

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

#define EGAVGA_SEQUENCER 0x3C4 /* Sequencer address/data port */
#define EGAVGA_MONCTR 0x3D4 /* Monitor controller */
#define EGAVGA_GRAPHCTR 0x3CE /* Graphics controller addr./data port */

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

typedef unsigned char BYTE;

extern void defchar( BYTE ascii, BYTE table, BYTE height, /*Assembler*/
BYTE numchar, void far * buf ); /*routine */

/*****
* SetCursor : Specifies screen cursor position. *
**-----**
* Input parameters : CURCOL = New cursor column (0-79) *
* CURROW = New cursor row (0-24) *
*****/

```

```

*   Return values       : None
*****/

void SetCursor( BYTE curcol, BYTE curow )
{
    union REGS regs;                /* Processor regs. for interrupt call */

    regs.h.ah = 2;                  /* Function number: Set cursor */
    regs.h.bh = 0;                  /* Access screen page 0 */
    regs.h.dh = curow;              /* Set row */
    regs.h.dl = curcol;             /* Set column */
    int86(0x10, &regs, &regs);     /* Call BIOS video interrupt */
}

/*****
*   PrintfAt : Displays a formatted string anywhere on the screen.
**-----**
*   Input parameters: COLUMN = Column
*                   SCROW  = Row
*                   CHCOL  = Character attribute
*                   STRING = Pointer to string
*   Return values   : None
*   Info            : This function should only be called if the
*                   system running this program contains an EGA card*
*                   or a VGA card.
*****/

void PrintfAt( BYTE column, BYTE scrow, BYTE chcol, char * string, ... )
{
    va_list parameter;              /* Parameter list for VA_... macros */
    char outbufr[255],             /* Buffer for formatted string */
        *outptr;

```

```

BYTE far *vptr;                                /* Pointer to video RAM */

va_start( parameter, string );                  /* Convert parameters */
vsprintf( outbufr, string, parameter );         /* Format */

vptr = (BYTE far *) MK_FP( 0xB800, column*2+scrow*160 );

for ( outptr = outbufr; *outptr ; )             /* Execute string */
{
    *vptr++ = *(outptr++);                      /* Write character to video RAM */
    *vptr++ = chcol;                            /* Write attribute to video RAM */
}

/*****
*   ClrScr: Clears the screen.
*-----**
*   Input parameters: CHATT  = Character attribute
*   Return values   : None
*****/

void ClrScr( BYTE chatt )
{
    BYTE far *vptr;                            /* Pointer to video RAM */
    int  count = 2000;                         /* Number of characters to be cleared */

    vptr = (BYTE far *) MK_FP( 0xB800, 0 ); /* Set pointer to video RAM */

    for ( ; count--; )                         /* Execute video RAM */
    {
        *vptr++ = ' ';                        /* Write character to video RAM */
        *vptr++ = chatt;                      /* Write attribute to video RAM */
    }
}

```

```

    }
}

```

```

/*****
*   SetCharWidth: Sets VGA character width to 8 or 9 pixels.
*   ****
*   Input      : HWIDTH = Character width (8 or 9)
*   ****
*****/

```

```

void SetCharWidth( BYTE hwidth )
{
    union REGS Regs;          /* Processor registers for interrupt call */
    unsigned char x;          /* Value for misc. output reg. */

    Regs.x.bx = ( hwidth == 8 ) ? 0x0001 : 0x0800;

    x = inp( 0x3CC ) & (255-12);          /* Toggle horizontal */
    if ( hwidth == 9 )                    /* resolution from */
        x |= 4;                          /* 720 to 640 pixels */
    outp( 0x3C2, x);

    CLI();                               /* Toggle sequencer from 8 to 9 pixels */
    outpw( EGAVGA_SEQUENCER, 0x0100 );
    outpw( EGAVGA_SEQUENCER, 0x01 + ( Regs.h.bl << 8 ) );
    outpw( EGAVGA_SEQUENCER, 0x0300 );
    STI();

    Regs.x.ax = 0x1000;                  /* Change horizontal screen position */
    Regs.h.bl = 0x13;
    int86( 0x10, &Regs, &Regs );
}

```

```

/*****
*   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 to VGA only */
    int86( 0x10, &Regs, &Regs );
    if ( Regs.h.al == 0x1a )          /* Is the 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 ( Regs.h.bl != 0x10 ) ? EGA : NEITHERNOR;
    }
}

```

```

/*****
*   BuildLogo : Draws a logo on the screen using custom characters   *
*               based on existing ASCII characters.
**-----**
*   Input      : COLUMN = Starting column of logo (1-80)
*               SLROW   = Starting row of logo (1-25)
*               DEPTH    = Number of logo scan lines
*               OPCOL    = Logo output color

```



```

        topb,                                /* Top border in pixels */
        bttmb;                               /* Bottom border in pixels */

videoc = IsEgaVga();                        /* Check for video card */
switch ( videoc )
{
    case NEITHERNOR :
        printf( "Warning: No EGA or VGA card found\n" );
        return;

    case EGA :
        charheight = 14;                    /* EGA: 14 scan lines per character */
        break;

    case VGA :
        SetCharWidth( 8 );                  /* Set char. width to 8 pixels */
        charheight = 16;                    /* 16 scan lines per character */
        break;
}

swidth = strlen( *bufp );                  /* Get string length and logo width */
dx = ( swidth + 7 ) / 8;                   /* Compute number of characters needed */
dy = ( depth + charheight - 1 ) / charheight;
if ( dx*dy > MAX_CHAR )
    printf( "Error: Logo in BuildLogo too large\n" );
else
{
    topb   = ( dy*charheight-depth ) / 2;    /* Compute border */
    bttmb  = depth + topb - 1;
    leftb  = ( dx*8-swidth ) / 2;
    rightb = swidth + leftb - 1;
}

```



```

for ( index = 0, i = 0; i < dy; ++ i)
{
    for ( j = 0; j < dx; ++j, ++index )      /* Execute text rows */
    {                                           /* Execute text columns */
        PrintfAt( column+j, slow+i, opcol,      /* Display character */
            "%c", UseChars[ index ] );

        /*-- Compute new character pattern for the character -----*/

        for ( k = 0; k < charheight; ++ k )    /* Execute scan lines */
        {
            bmask = 0;                          /* Bit mask orig. 0 */
            for ( l = 0; l <= 7; ++l )          /* Each char. is 8 pixels wide */
            {
                bmask <= 1;                      /* Move mask left */
                lcurow = i * charheight + k;
                lcolumn = j * 8 + l;

                if ( lcurow>=topb && lcurow<=bttmb &&          /* Pixel out */
                    lcolumn>=leftb && lcolumn<=rightb )      /* of border? */
                {
                    if ( (*(bufp+lcurow-topb)+lcolumn-leftb) != ' ' )
                        bmask |= 1;                  /* Set pixel in logo */
                }
                chardef[ k ] = bmask;                /* Bit pattern in char. buffer */
            }
        }
        defchar( UseChars[ index ], 0, charheight, 1, chardef );
    }
}
}

/*****

```


[illegible]

```

"          ****          ****          ****          ",
"          ****          ****          ****          " };

```

```

int i, j;
BYTE opcol;

```

```

static BYTE NewDef[MAX_CHAR] =          /* Chars. to be redefined */
{ 128, 130, 131, 133, 134, 135, 136, 137, 138, 139,
  140, 141, 143, 144, 145, 146, 147, 149, 150, 151,
  152, 155, 156, 157, 158, 159, 160, 161, 162, 163,
  164, 165 };

```

```

ClrScr( 0 );
for ( i = 0; i < 256; ++i )          /* Display entire character set */
{
    for ( j = 0; j < MAX_CHAR; ++ j )    /* Should chars by redefined? */
    {
        if ( NewDef[ j ] == i )          /* Corresponding chars? */
            break;                      /* Yes --> Exit loop */
    }
    opcol = ( j < MAX_CHAR ) ? 15 : 14;

    PrintfAt( ( i % 13 ) * 6 + 1, i / 13, opcol, "%3d:%c", i, i );
}

```

```

PrintfAt( 22, 22, 14, "LOGOC - (c) 1992 by Michael Tischer" );
BuildLogo( 60, 21, 32, 0x3F, MyLogo );          /* Build logo */
getch();
ResetLogo();          /* Reset logo */
ClrScr( 15 );
SetCursor( 0, 0 );
}

```

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

```
void main( void )  
{  
    Test();  
}
```

•