

```

/*****
*                               S O F T S C R C . C                               *
**-----**
* Task                        : Demonstrates soft scrolling on EGA & VGA cards. *
**-----**
* Author                     : Michael Tischer                               *
* Developed on               : 08/26/90                                       *
* Last update                : 02/21/92                                       *
**-----**
* (MICROSOFT C)                                                       *
* Compilation                : CL /AS /c /W0 softscrc.c                     *
*                               LINK softscrc softscca;                       *
**-----**
* (BORLAND TURBO C)                                                  *
* Compilation                : Create a project file containing the following: *
*                               softscrc.c                                     *
*                               softscca.obj                                   *
**-----**
* Call                      : softscrc                                       *
*****/

```

```

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

#ifdef __TURBOC__                                /* Compiling with Turbo C? */
#define CLI()          disable()
#define STI()          enable()
#define outpw( p, w )  outputport( p, w )
#define inp
#define outpb( p, b )  outputportb( 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 FAST          2                  /* Speed constant for ShowScrlText() */
#define MEDIUM       1
#define SLOW          0

#define PCOLR         0x5E              /* Yellow from lilac palette */
#define PCOLR1        0x5F              /* White from lilac palette */
#define CWIDTH        8                  /* Character width in pixels */
#define CHEIGHT       14                 /* Character height in scan lines */
#define COLUMNS       216               /* Columns per row in video RAM */
#define BANDSIZE      10800              /* Band size */
#define BANDNUM        3                  /* Number of bands */
#define MAXLEN        61                 /* Maximum number of characters */
#define STARTR        5                  /* Starting character row on screen */

#define CrtAttr        0x3C0             /* CRT attribute controller register */
#define CrtStatus      0x3da             /* Status port */
#define CrtAdr         0x3d4             /* Monitor address port */

#define TRUE           ( 0 == 0 )
#define FALSE          ( 0 == 1 )

#define EGA            0                  /* Card types */

```

```

#define VGA 1
#define NEITHERNOR 2

typedef unsigned char BYTE; /* Create a BYTE */
typedef unsigned int WORD; /* Create a WORD */
typedef BYTE BOOL;

typedef WORD VRAM[BANDNUM][25][COLUMNS]; /* Video RAM definition */
typedef VRAM far *VPTR; /* FAR pointer to video RAM */

typedef BYTE FDEF[256][14]; /* Font array */
typedef FDEF far *TPTR; /* Pointer to font */

/*-- External functions -----*/
extern void far * getfontptr( void ); /* Assembler function */

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

VPTR vp; /* Pointer to video RAM */

/*****
* SetOrigin : Specifies the visible part of video RAM for
* programming the video controller.
*-----*/
* Input : BAND = Number of band to be displayed (1-5)
* COLUMN, = Number of columns and rows displayed in the
* SCROW upper-left corner of the screen (origin=0/0)
* PIXX, = Pixel offsets
* PIXY
* Output : None
*****/

```

```

void SetOrigin( BYTE band, BYTE column, BYTE scrow,
               BYTE pixx, BYTE pixy)
{
    int offset;                                /* Offset of video RAM start */

    offset = ( BANDSIZE >> 1) * band + scrow * COLUMNS + column;

    /*- Execute vertical rescan and wait for end -----*/
    while ( !(( inp(CrtStatus) & 8 ) == 8 ))
        ;
    while ( !(( inp(CrtStatus) & 8 ) == 0 ))
        ;

    /* Write offset for start of video RAM to registers 0CH and 0DH,    */
    /* after ensuring that next screen layout is valid                  */
    CLI();                                    /* Disable interrupts */
    outpw( CrtAdr, ( offset & 0xFF00 ) + 0x0c );
    outpw( CrtAdr, ( (BYTE) offset << 8 ) + 0x0d );
    STI();                                    /* Enable interrupts */

    /*- While waiting for next rescan, write new pixel offset -----*/
    /*- and set up new starting address                                -----*/

    while ( !(( inp( CrtStatus ) & 8 ) == 8 ))
        ;

    /*- Write pixel offsets in register 08H or -----*/
    /*- register 13H of the attribute controller -----*/

```

```

CLI();                                     /* Disable interrupts */
outpw( CrtAdr, ( pixy << 8 ) + 0x08 );
outp( CrtAttr, 0x13 | 0x20 );
outp( CrtAttr, pixx );
STI();                                     /* Enable interrupts */
}

```

```

/*****
*   PrintChar : Creates a character within the visible range of
*               video RAM.
*-----*
*   Input      :   THECHAR = Character to be created
*                  BAND    = Band number (0-4)
*                  COLUMN   = Column in video RAM at which the column
*                           should begin
*   Info       :   The character can be moved in the visible range of
*                   the screen by calling SmoothLeft.
*                   The character is created from a 14x8 matrix, based on
*                   the EGA/VGA ROM font.
*****/

```

```

void PrintChar( char thechar, BYTE band, BYTE column )
{
    char ch;                                     /* Character pixels */
    BYTE i, k,                                  /* Loop counter */
        BMask;                                  /* Bit mask for character design */
    static TPTR fptr = (TPTR) 0;                /* Pointer to ROM font */

    if ( fptr == (TPTR) 0 )                     /* Pointer to font already set? */
        fptr = getfontptr();                   /* No --> Get assembly language function */

    /*- Generate character line by line -----*/

```

```

for ( i = 0; i < CHEIGHT; ++i )
{
    BMask = (*fptr)[thechar][i];          /* Get bit pattern for one line */
    for ( k = 0; k < CWIDTH; ++k )        /* Set individual bits */
    {
        ch = ( BMask & 128 ) ? 219 : 32;
        (*vp)[band][STARTR+i][column*CWIDTH+k] = (BYTE) ch+( PCOLR << 8 );
        BMask <= 1;                       /* Process next bit */
    }
}
}

```

```

/*****
* 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;
}

/*****
*   ShowScrlText : Scrolls text on the screen.
*   -----
*   Input      :  STEXT = String of text for scrolling
*                 SPEED = Scroll speed (SLOW, MEDIUM or FAST)
*                 VC     = Video card type (EGA or VGA)
*****/

void ShowScrlText( char * stext, BYTE speed, BYTE vc )
{
    int          band,                /* Current band */
               column,               /* Current column */
               index,                /* Index to string to be created */
               len,                  /* Length of text to be displayed */
               i, k,                  /* Loop counter */
               pixx;                 /* Horizontal pan value */
    WORD far *wptr;                  /* Pointer to video RAM loop */
    union REGS Regs;                 /* Processor registers for interrupt call */

    static BYTE steptable[2][3][10] =
    {
        {
            {
                0,  1,  2,  3,  4,  5,  6,  7, 255, 255 },
            {
                0,  2,  4,  6, 255, 255, 255, 255, 255, 255 },
            {
                0,  4, 255, 255, 255, 255, 255, 255, 255, 255 }
        },
        {
            /* VGA step values */

```

```

        { 8, 0, 1, 2, 3, 4, 5, 6, 7, 255 },
        { 8, 2, 5, 255, 255, 255, 255, 255, 255, 255 },
        { 8, 3, 255, 255, 255, 255, 255, 255, 255, 255 }
    };
};

```

```

vp = MK_FP( 0xB800, 0x0000 );          /* Set pointer to video RAM */

/*- Fill entire video RAM with blank spaces -----*/

for ( index = 0; index < BANDNUM; ++index )
    for ( i = 0; i < 25; ++i )
        for ( k = 0; k < COLUMNS; ++k )
            (*vp)[index][ i ][ k ] = ( PCOLR << 8 ) + 32;

/*- Draw horizontal bands -----*/

for ( k = 0; k < BANDNUM; ++k )
    for ( i = 0; i < COLUMNS; ++i )
    {
        (*vp)[ k ][ STARTR-2 ][ i ] = (BYTE) 'Í' + ( PCOLR1 << 8 );
        (*vp)[ k ][ STARTR + CHEIGHT + 2 ][ i ] = (BYTE) 'Í' + ( PCOLR1 << 8 );
    }

/*- Remove blinking cursor from the screen -----*/

Regs.h.ah = 0x02;                      /* Function number: Set cursor */
Regs.h.bh = 0;                         /* Screen page */
Regs.x.dx = 0;                         /* coordinates */
int86( 0x10, &Regs, &Regs );

/*- Set screen border color -----*/

```



```

Regs.h.ah = 0x10;          /* Function number: Set border color */
Regs.h.al = 0x01;          /* Sub-function number */
Regs.h.bh = PCOLR >> 4;    /* Border color */
int86( 0x10, &Regs, &Regs );

/*- Place number of columns per row in video RAM from columns -----*/
outpw( CrtAdr, ( ( COLUMNS >> 1 ) << 8 ) + 0x13 );

/*-- Place scrolling text in video RAM -----*/
if ( ( len = strlen( stext ) ) > MAXLEN )      /* String too long? */
    *(stext + ( len = MAXLEN )) = '\0';        /* Yes --> Truncate */

for ( column = band = index = 0; index < len; )
{
    PrintChar( *(stext+index++), band, column++ ); /* Draw characters */
    if ( column >= COLUMNS / CWIDTH )           /* Band change? */
    {
        column = 0;                               /* Yes */
        ++band;                                    /* Restart in column 1 */
        index -= 80 / CWIDTH;                     /* Next band */
    }
    /* Character one page back */
}

/*-- Move scroll text from right to left on the screen -----*/

for ( column = band = 0 , i = (len - ( 80 / CWIDTH )) * CWIDTH;
    i > 0;
    --i )
{

```

```

    for ( k = 0; ( pixx = steptable[vc][speed][k]) != 255 ; ++k )
        SetOrigin( band, column, 0, pixx, 0 );

    if ( ++column == COLUMNS - 80 )                /* Band change? */
    {                                                  /* Yes */
        column = 0;                                /* Restart in column 0 */
        ++band;                                     /* Increment band */
    }
}

/*- Revert to 80 characters per row in video RAM -----*/
outpw( CrtAdr, ( 40 << 8 ) + 0x13 );

SetOrigin( 0, 0, 0, 8, 0 );                        /* Revert to default */

/*- Return cursor to the screen -----*/

Regs.h.ah = 0x02;                                /* Function number: Set cursor */
Regs.h.bh = 0;                                    /* Screen page */
Regs.x.dx = 0;                                    /* coordinates */
int86( 0x10, &Regs, &Regs );

/*- Reset border colors -----*/

Regs.h.ah = 0x10;                                /* Function number: Set border color */
Regs.h.al = 0x01;                                /* Sub-function number */
Regs.h.bh = 0;                                    /* Black border */
int86( 0x10, &Regs, &Regs );

/*-- Clear screen -----*/

```

```

for ( wptr = (WORD far *) vp, i = 80*25; i-- ; )
    *wptr++ = 0x0720;
}

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

void main( void )
{
    BYTE vc;                                /* Get video card type */

    if ( ( vc = IsEgaVga() ) == NEITHERNOR )
        printf( "SOFTSCRC - (c) 1992 by Michael Tischer\n" \
                "Warning: No EGA or VGA card found\n" );
    else
        ShowScrlText( "++ PC Intern.....Published by Abacus ++" \
                      "      ", FAST, vc );
}

```