

```

;*****;
;*                                     V M O N O                                     *;
;*-----*
;* Task           : Makes some elementary functions available for *;
;*                 access to the monochrome display screen.        *;
;*-----*
;* Info           : All functions subdivide the screen             *;
;*                 into columns 0 to 79 and lines 0 to 24           *;
;*-----*
;* Author          : Michael Tischer                                *;
;* Developed on    : 8/11/87                                         *;
;* Last update     : 3/02/92                                         *;
;*-----*
;* Assembly        : MASM VMONO;                                     *;
;*                 LINK VMONO;                                       *;
;*-----*
;* Call            : VMONO                                           *;
;*****;

;== Constants =====

CONTROL_REG = 03B8h           ;Control register port address
ADDRESS_6845 = 04B4h          ;6845 address register
DATA_6845    = 03B5h          ;6845 data register
VIO_SEG      = 0B000h         ;Segment address of video RAM
CUR_START    = 10             ;Register # CRTC: Starting cursor line
CUR_END      = 11             ;Register # CRTC: Ending cursor line
CURPOS_HI    = 14             ;Register # CRTC: Cursor pos. high byte
CURPOS_LO    = 15             ;Register # CRTC: Cursor pos. low byte

DELAY        = 20000          ;Counter for delay loop

;== Stack =====

stack        segment para stack ;Definition of stack segment

              dw 256 dup (?)     ;256-word stack

stack        ends              ;End of stack segment

;== Data =====

data         segment para 'DATA' ;Define data segment

;== the Data for the Demo-Program =====

str1         db "a",0
str2         db ">PC INTERN<",0
str3         db "  window 1  ",0
str4         db "  window 2  ",0
str5         db "          the program is stopped by "
              db " pressing a Key....          ",0

initm        db 13,10,"VMONO (c) 1987 by Michael Tischer",13,10,13,10
              db "This demonstration program only runs with "
              db " a monochrome",13,10,"display card. If your PC "
              db "has another type of display card,",13,10
              db "please enter <s> to stop the "
              db " program.",13,10,"Otherwise press any "
              db "key to start ",13,10
              db "the program ...",13,10,"$"

;== Data =====

linen        dw 0*160,1*160,2*160 ;Start addresses of the lines as
              dw 3*160,4*160,5*160 ;offset addresses in the video RAM
              dw 6*160,7*160,8*160
              dw 9*160,10*160,11*160,12*160,13*160,14*160,15*160,16*160
              dw 17*160,18*160,19*160,20*160,21*160,22*160,23*160,24*160

data         ends              ;End of data segment

;== Code =====

code         segment para 'CODE' ;Definition of the CODE segment

```

```

        assume cs:code, ds:data, es:data, ss:stack

;== this is the Demo-Program =====

demo    proc far

        mov ax,data           ;Get segment address of data segment
        mov ds,ax             ;and load into DS
        mov es,ax             ;as well as ES

        ;-- Display initial msg./wait for input -----

        mov ah,9              ;String output function
        mov dx,offset initm    ;Address of initial message
        int 21h                ;Call DOS interrupt 21H

        xor ah,ah              ;Get function number for key
        int 16h                ;Call BIOS keyboard interrupt
        cmp al,"s"              ;was <s> entered?
        je ende                ;YES --> end program
        cmp al,"S"              ;was <S> entered?
        jne startdemo          ;NO --> start demo

ende:    mov ax,4c00h           ;Function number for program end
        int 21h                ;Call DOS interrupt 21H

startdemo label near
        mov cx,0d00h           ;Enable full cursor
        call cdef
        call cls                ;Clear screen

        ;-- Fill screen with ASCII characters -----

        xor di,di              ;Start in upper left corner
        mov si,offset str1      ;Offset address of string1
        mov cx,2000             ;2,000 characters fit on the screen
        mov al,07h              ;white letters on black background
demo1:   call print              ;Display string
        inc str1                ;Increment character in test string
        jne demo2               ;NUL code suppressed
        inc str1
demo2:   loop demo1             ;Repeat output

        ;-- Create window 1 and window 2 -----

        mov bx,0508h           ;Upper left corner of window 1
        mov dx,1316h           ;Lower right corner of window 1
        mov ah,07h              ;White letters, black background
        call clear              ;Clear window 1
        mov bx,3C02h           ;Upper left corner of window 2
        mov dx,4A10h           ;Lower right corner window 2
        call clear              ;Clear window 2
        mov bx,0508h           ;Upper left corner of window 1
        call calo               ;Convert to offset address
        mov si,offset str3      ;Offset address string 3
        mov ah,70h              ;Black characters, white background
        call print              ;Display string 3
        mov bx,3C02h           ;Upper left corner of window 2
        call calo               ;Convert to offset address
        mov si,offset str4      ;Offset address string 4
        call print              ;Display string 4
        xor di,di              ;Upper left display corner
        mov si,offset str5      ;Offset address string 5
        call print              ;Display string 5

        ;-- Display program logo -----

        mov bx,1E0Ch           ;Column 30, line 12
        call calo               ;Convert offset address
        mov si,offset str2      ;Offset address string 2
        mov ah,0F0h             ;Inverse blinking
        call print              ;Display string 2

        ;-- Fill window with arrows -----

        xor ch,ch                ;high-byte of the counter to 0

```

```

arrow:    mov    bl,1                ;Asterisk
arrow0:   push   bx                  ;Push BX on the stack
        mov    di,offset str3       ;Draw arrow line in string 3
        mov    cl,15                ;Total of 15 characters in a line
        sub    cl,bl                ;Calculate number of spaces
        shr    cl,1                 ;Divide by 2 (for left half)
        or     cl,cl                ;No blanks ?
        je     arrow1               ;YES --> ARROW1
        mov    al," "
        rep    stosb                 ;Draw blanks in string 3
arrow1:   mov    cl,bl                ;Number of asterisks in counter
        mov    al,"*"
        rep    stosb                 ;Draw stars in string 3
        mov    cl,15                ;Total of 15 characters in a line
        sub    cl,bl                ;Calculate number of blanks
        shr    cl,1                 ;Divide by 2 (for right half)
        or     cl,cl                ;No blanks?
        je     arrow2               ;YES --> ARROW2
        mov    al," "
        rep    stosb                 ;Draw blanks in string 3
arrow2:   mov    bx,0509h            ;below the first line of window 1
        call   calo                  ;Convert to offset address
        mov    si,offset str3       ;Offset address string 3
        mov    ah,07h               ;White characters, black background
        call   print                 ;Display string 3
        mov    bx,3C10h             ;into the lowest line of window 2
        call   calo                  ;Convert offset address
        call   print                 ;Display string 3

        ;-- Brief pause -----

waitlp:   mov    cx,DELAY             ;Loop counter
        loop   waitlp               ;Count loop to 0

        ;-- Scroll window 1 line down -----

        mov    bx,0509h             ;Upper left corner of window 1
        mov    dx,1316h             ;Lower right corner window 1
        mov    cl,1                 ;Scroll down
        call   scrolldn              ;one line

        ;-- Scroll window 2 one line up -----

        mov    bx,3C03h             ;Upper left corner window 2
        mov    dx,4A10h             ;Lower right corner window 2
        call   scrollup              ;Scroll up

        ;-- Was a key pressed? (end program) -----

        mov    ah,1                 ;Function number for testing key
        int    16h                  ;Call BIOS keyboard interrupt
        jne    end_it                ;Keypress -> goto end of program

        ;-- NO, display next arrow -----

        pop    bx                    ;Pop BX from stack again
        add    bl,2                  ;2 more stars in next line
        cmp    bl,17                ;Reached 17 ?
        jne    arrow0               ;NO --> next arrow
        jmp    arrow                 ;No key --> next arrow

        ;-- Get ready to end program

end_it:   xor    ah,ah                ;Get function number for key
        int    16h                  ;Call BIOS keyboard interrupt
        mov    cx,0D0Ch              ;Restore normal cursor
        call   cdef                  ;Clear screen
        call   cls                    ;Clear screen
        jmp    ende                  ;Go to end of program

demo      endp

;== Functions =====

;-- SOFF: switches the display off -----
;-- Input      : none

```

```

;-- Output      : none
;-- register    : AX and DX are changed

SOFF      proc near

        mov     dx,CONTROL_REG      ;Address of display control register
        in      al,dx               ;read its content
        and     al,11110111b        ;bit 3 = 0: display off
        out     dx,al               ;set new value (display off)

        ret                                ;back to caller

SOFF      endp

;-- SON: switches the display on -----
;-- Input       : none
;-- Output      : none
;-- register    : AX and DX are changed

SON       proc near

        mov     dx,CONTROL_REG      ;Address of display control register
        in      al,dx               ;Read its content
        or      al,8                ;Bit 3 = 1: display on
        out     dx,al               ;Set new value (display on)
        ret                                ;Back to caller

SON       endp

;-- CDEF: sets the start and end line of the cursor -----
;-- Input       : CL = Start line
;--             : CH = End line
;-- Output      : none
;-- register    : AX and DX are changed
cdef      proc near

        mov     al,CUR_START        ;Register 10: start line
        mov     ah,cl               ;Start line to AH
        call    setvk               ;Transmit to video controller
        mov     al,CUR_END          ;Register 11: end line
        mov     ah,ch               ;End line to AH
        jmp     short setvk         ;Transmit to video controller

cdef      endp

;-- SETBLINK: sets the blinking display cursor -----
;-- Input       : DI = offset address of the cursor
;-- Output      : none
;-- register    : BX, AX and DX are changed

setblink  proc near

        mov     bx,di               ;Transmit offset to BX
        mov     al,CURPOS_HI        ;Register 15:high byte of cursor offset
        mov     ah,bh               ;high-byte of the offset
        call    setvk               ;Transmit to video controller
        mov     al,CURPOS_LO        ;Register 15:Low byte of cursor offset
        mov     ah,bl               ;Low byte of the offset

        ;-- SETVK is called automatically -----

setblink  endp

;--SETVK: sets a byte in one of the registers of the video controller --
;-- Input       : AL = number of the register
;--             : AH = new content of the register
;-- Output      : none
;-- register    : DX and AL are changed

setvk     proc near

        mov     dx,ADDRESS_6845     ;Address of the index register
        out     dx,al               ;Send number of the register
        jmp     short $+2            ;Small I/O pause
        inc     dx                  ;Address of the index register
        mov     al,ah               ;Content to AL

```

```

        out  dx,al          ;Set new content
        ret                ;Back to caller

setvk    endp

;-- GETVK: reads a byte from one register of the video controllers -
;-- Input      : AL = number of the register
;-- Output     : AL = content of the register
;-- register   : DX and AL are changed

getvk    proc near

        mov  dx,ADDRESS_6845 ;Address of the index register
        out  dx,al          ;Send number of the register
        jmp  short $+2
        inc  dx             ;Address of the index register
        in   al,dx          ;Read content to AL
        ret                ;Back to caller

getvk    endp

;-- SCROLLUP: scrolls a window up by N lines -----
;-- Input   : BL = line upper left
;--          BH = column upper left
;--          DL = line lower right
;--          DH = column lower right
;--          CL = number of lines to scroll
;-- Output  : none
;-- register : only FLAGS are changed
;-- Info    : the display lines released are erased

scrollup proc near

        cld                ;Increment on string instructions

        push ax             ;Push all changed registers on the
        push bx             ;stack
        push di             ;In this case the sequence
        push si             ;must be observed!

        push bx             ;These three registers are restored
        push cx             ;from the stack before ending
        push dx
        sub  dl,bl          ;Calculate the number of lines
        inc  dl
        sub  dl,cl          ;Deduct number of lines scrolled
        sub  dh,bh          ;Calculate number of columns
        inc  dh
        call calo           ;Convert upper left in offset
        mov  si,di          ;Record Address in SI
        add  bl,cl          ;First line in scrolled window
        call calo           ;Convert first line to offset
        xchg si,di          ;Exchange SI and DI
        push ds             ;Store segment register on
        push es             ;the stack
        mov  ax,VIO_SEG     ;Segment address of the video RAM
        mov  ds,ax          ;to DS
        mov  es,ax          ;and ES
supl:    mov  ax,di          ;Record DI in AX
        mov  bx,si          ;Record SI in BX
        mov  cl,dh          ;Number of column in counter
        rep movsw           ;Move a line
        mov  di,ax          ;Restore DI from AX
        mov  si,bx          ;Restore SI from BX
        add  di,160         ;Set next line
        add  si,160
        dec  dl             ;Processed all lines ?
        jne supl           ;NO --> move another line
        pop  es             ;Get segment register from
        pop  ds             ;stack
        pop  dx             ;Get lower right corner
        pop  cx             ;Read number of lines
        pop  bx             ;Get upper left corner
        mov  bl,dl          ;Lower line to BL
        sub  bl,cl          ;Deduct number of lines
        inc  bl

```

```

        mov     ah,07h           ;Color : black on white
        call    clear           ;Erase lines freed

        pop     si               ;CX and DX have already
        pop     di               ;been read
        pop     bx
        pop     ax

        ret                     ;Back to caller

scrollup endp

;-- SCROLLDN: scrolls a window down N lines -----
;-- Input  : BL = line upper left
;--          BH = column upper left
;--          DL = line lower right
;--          DH = column lower right
;--          CL = number of lines to scroll
;-- Output  : none
;-- register: only FLAGS are changed
;-- Info    : display lines released are erased

scrolldn proc near

        cld                     ;Increment on string instructions

        push    ax               ;Store all changed registers on the
        push    bx               ;stack
        push    di               ;In this case the sequence
        push    si               ;must be observed !

        push    bx               ;These three registers are returned
        push    cx               ;from the stack before the end
        push    dx               ;of the routine

        sub     dh,bh            ;Calculate the number of the column
        inc     dh
        mov     al,bh            ;Record line upper left in AL
        mov     bl,dl            ;Line upper right to line upper left
        call    calo             ;Convert upper left into offset
        mov     si,di            ;Record address in SI
        sub     bl,cl            ;Deduct number of lines to scroll
        call    calo             ;Convert upper left in offset
        xchg    si,di            ;Exchange SI and DI
        sub     dl,al            ;Calculate number of lines
        inc     dl               ;Deduct number
        sub     dl,cl            ;of lines to be scrolled
        push    ds               ;Push segment register onto stack
        push    es

        mov     ax,VIO_SEG       ;Segment address of video RAM
        mov     ds,ax            ;to DS
        mov     es,ax            ;and ES

sdn1:   mov     ax,di             ;Move DI to AX
        mov     bx,si            ;Move SI to BX
        mov     cl,dh            ;Number column in counter
        rep     movsw            ;Scroll one line
        mov     di,ax            ;Get DI from AX
        mov     si,bx            ;Restore SI from BX
        sub     di,160           ;Set next line
        sub     si,160
        dec     dl               ;All lines processed ?
        jne     sdn1            ;NO --> scroll another line
        pop     es               ;Get segment register from
        pop     ds               ;stack
        pop     dx               ;Return lower right corner
        pop     cx               ;Return number of lines
        pop     bx               ;Return upper left corner
        mov     dl,bh            ;Upper line to DL
        add     dl,cl            ;Add number of lines
        dec     dl
        mov     ah,07h           ;Color : black on white
        call    clear           ;Erase lines which were released

        pop     si               ;CX and DX are
        pop     di               ; already returned
        pop     bx

```



```

        push si                ;Store SI, DX and ES on the stack
        push es
        push dx
        mov dx,VIO_SEG        ;Segment address of the video RAM
        mov es,dx             ;First to DX and then to ES
        jmp print1            ;YES --> Output finished

print0:  stosw                  ;Store attribute and color in V-RAM
print1:  lodsb                  ;Get next character from the string
        or al,al               ;Is it NUL
        jne print0            ;NO --> output

printe:  pop dx                 ;Get SI, DX and ES back from stack
        pop es
        pop si
        ret                    ;Back to caller

print    endp

;- CALO: converts line and column into offset address -----
;-- Input      : BL = line
;--             BH = column
;-- Output     : DI = the offset address
;-- Registers: DI and FLAGS are changed

calo     proc near

        push ax                ;Store AX on the stack
        push bx                ;Store BX on the stack

        shl bx,1               ;Column and line times 2
        mov al,bh              ;Column to AL
        xor bh,bh              ;Get high-byte
        mov di,[linen+bx]      ;Offset address of the line
        xor ah,ah              ;high-byte for column offset
        add di,ax              ;Add line- and column offset

        pop bx                 ;Get BX from stack again
        pop ax                 ;Get AX from stack again
        ret                    ;Back to caller

calo     endp

;== End =====

code     ends                  ;End of the CODE segment
~~~~~   end demo              ;Start program execution w/ demo

```