

```

;*****;
;*                               V C O L                               *;
;*-----*
;*   Task           : Makes some basic functions available for      *;
;*                   access to the Color Graphics Adapter (CGA).      *;
;*-----*
;*   Info           : All functions subdivide the screen              *;
;*                   into columns 0 to 79 and lines 0 to 24            *;
;*                   in text mode and into columns 0 to 719 and        *;
;*                   the lines 0 to 347 in graphic mode.               *;
;*                   the 40 column text mode is not supported !        *;
;*                   A high resolution graphic screen should appear*;*;
;*                   first, followed by a text screen. If the high    *;
;*                   res screen doesn't appear, try running the       *;
;*                   program a few times in succession.               *;
;*-----*
;*   Author          : Michael Tischer                                *;
;*   Developed on     : 08/13/87                                        *;
;*   Last update      : 04/12/95                                        *;
;*-----*
;*   Assembly        : MASM VCOL;                                     *;
;*                   LINK VCOL;                                       *;
;*-----*
;*   Call            : VCOL                                           *;
;*****;

```

```

;== Constants =====

```

```

CONTROL_REG  = 03D8h           ;Control register port address
CCHOICE_REG  = 03D9h           ;Color select register port address
ADDRESS_6845 = 03D4h           ;6845 address register
DATA_6845    = 03D5h           ;6845 data register

```

```

VIO_SEG      = 0B800h      ;Video RAM segment address
CUR_START    = 10          ;Reg # for CRTC: Cursor start line
CUR_END      = 11          ;Reg # for CRTC: Cursor end line
CURPG_HI     = 12          ;Page address (high byte)
CURPG_LO     = 13          ;Page address (low byte)
CURPOS_HI    = 14          ;Reg # for CRTC: Cursor pos high byte
CURPOS_LO    = 15          ;Reg # for CRTC: Cursor pos low byte
DELAY        = 20000       ;Counter for delay loop

;== Macros =====

;-- SETMODE : Macro for configuring screen control register -----

setmode      macro modus

        mov  dx,CONTROL_REG    ;Address of the display control register
        mov  al,modus          ;New mode into the AL register
        out  dx,al             ;Send mode to control register

        endm

;-- WAITRET: waits until display is completed -----

waitret      macro

local        wr1                ;Local label

        mov  dx,3DAh           ;Address of the display status register
wr1:        in  al,dx            ;Get content
        test al,8               ;Vertical retrace?
        je   wr1                ;No --> waitforit

        endm

```

```

;== 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'     ;Definition of data segment

;== Data required for demo program =====

initm      db 13,10
           db "VCOL (c) 1988,1989 by Michael Tischer "
           db 13,10,13,10
           db "This demo program only runs with a Color/Graphics",13,10
           db "Adapter ( CGA ).  If your PC uses another type of",13,10
           db "video card press the <s> key to stop the program.",13,10
           db "Press any other key to start the program...",13,10,"$"

str1       db 1,0

;== Table of offset addresses of line beginnings =====
lines      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

```

```

graphict  db 38h, 28h, 2Dh, 0Ah, 7Fh, 06h  ;register values for the
          db 64h, 70h, 02h, 01h, 06h, 07h  ;graphic-modes

texttt    db 71h, 50h, 5Ah, 0Ah, 1Fh, 06h  ;register values for the
          db 19h, 1Ch, 02h, 07h, 06h, 07h  ;graphic modes

waitforit db 0                               ;TRUE (<>0) when caller uses the
                                           ;/F switch

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 only the Demo-Program =====

demo      proc far

          ;-- Look for /F from DOS prompt -----

          mov  cl,ds:128                    ;Get number of bytes from prompt
          or   cl,cl                        ;No parameters given?
          je   switch1                     ;No --> Ignore
          mov  bx,129                       ;BX points to first byte in prompt
          mov  ch,bh                        ;Set loop high byte to 0

switch:   cmp  [bx],"F/"                    ;Switch in this position?
          je   switch1                     ;Yes --> Switch found
          cmp  [bx],"f/"                    ;Switch in this position?

```

```

        je    switch1          ;Yes --> Switch found
        inc   bl               ;Set BX to next character
        loop  switch          ;Check next character

switch1: mov   ax,data         ;Get segment addr. of data segment
        mov   ds,ax           ;and load into DS
        mov   es,ax           ;and ES

        mov   waitforit,cl     ;Set WAIT flag

;-- Display init message and wait for input -----

        mov   ah,9             ;Function number for string display
        mov   dx,offset initm  ;Address of initial message
        int   21h             ;Call DOS interrupt 21H

        xor   ah,ah           ;Function number: get key
        int   16h             ;Call BIOS keyboard interrupt
        cmp   al,"s"          ;<s> key pressed?
        je    ende            ;Yes --> End program
        cmp   al,"S"          ;<S> key pressed?
        jne   startdemo       ;No --> Start demo

ende:    mov   ax,4C00h        ;Function number: End program
        int   21h             ;Call DOS interrupt 21H

startdemo label near
        call  grafhi           ;switch on 320x200 pixel graphics
        xor   al,al            ;Clear graphic display
        call  cgr

        xor   bx,bx            ;Column 0

```

	xor dx,dx	;Line 0
	mov ax,199	;Pixels-vertical
	mov cx,639	;Pixels-horizontal
gr1:	push cx	;Record horizontal pixels
	mov cx,ax	;Vertical pixels to counter
	push ax	;Record vertical pixels on the stack
	mov al,1	
gr2:	call pixhi	;Set pixel
	inc dx	;Increment line
	loop gr2	;Draw line
	pop ax	;Get vertical pixels from the stack
	sub ax,3	;Next line 3 pixels less
	pop cx	;Get horizontal pixels from the stack
	push cx	;Record horizontal pixels
	push ax	;Record vertical pixels on the stack
	mov al,1	
gr3:	call pixhi	;Set pixel
	inc bx	;Increment column
	loop gr3	;Draw line
	pop ax	;Get vertical pixels from stack
	pop cx	;Get horizontal pixels from stack
	sub cx,6	;Next line 6 pixels less
	push cx	;Record horizontal pixels
	mov cx,ax	;Vertical pixels to counter
	push ax	;Record vertical pixels on the stack
	mov al,1	
gr4:	call pixhi	;Set pixel
	dec dx	;Decrement line
	loop gr4	;Draw line
	pop ax	;Get vertical pixels from stack
	sub ax,3	;Next line 3 pixels less
	pop cx	;Get horizontal pixels from stack

```

        push cx                ;Record horizontal pixels
        push ax                ;Record vertical pixels on the stack
gr5:    mov  al,1
        call pixhi             ;Set pixel
        dec  bx                ;Increment column
        loop gr5               ;Draw line
        pop  ax                ;Get vertical pixels from the stack
        pop  cx                ;Get horizontal pixels from the stack
        sub  cx,6               ;Next line 6 pixels less
        cmp  ax,5               ;Is the vertical line longer than 5
        ja   gr1               ;YES--> continue

        xor  ah,ah              ;Wait for function number of key wait
        int  16h               ;Call BIOS keyboard interrupt

        call text               ;Switch on 80x25 character text mode
demo1:  xor  bp,bp               ;Process screen page 0 first
        mov  al,30h             ;ASCII code "0"
        or   ax,bp              ;Convert page number to ASCII
        mov  str1,al            ;Store in string
        call setcol             ;Set color
        call setpage            ;Activate screen page in BP
        call cls                ;Clear screen page
        xor  bx,bx              ;Begin in the upper left
        call calo               ;Screen corner with output
        mov  cx,2000            ;A page contains 2,000 characters
        xor  ah,ah              ;Start with color code 0
        mov  si,offset str1     ;Offset address of string 1
demo2:  inc  ah                  ;Increment color value
        call print              ;Output string 1
        loop demo2              ;Repeat until screen is full

```

```

        xor     ah,ah                ;Wait for key
        int     16h                 ;Call BIOS keyboard interrupt
        inc     bp                  ;Increment page number
        cmp     bp,4                ;All 4 pages processed ?
        jne     demol              ;No --> then next page

        xor     bp,bp                ;Activate page 0 again
        call    setpage
        jmp     ende
demo     endp                      ;Goto program end

;== The actual functions follow =====

;-- TEXT: switches the text display on -----
;-- Input      : none
;-- Output     : none
;-- Register   : AX, SI, BH, DX and FLAGS are changed

text     proc near

        mov     si,offset textt     ;Offset address of the register table
        mov     bl,00100001b        ;80x25 text mode,blinking
        jmp     short vcprog        ;Program video controller again

text     endp

;-- GRAFHI: switches the 640*200 pixel graphic mode on -----
;-- Input      : none
;-- Output     : none
;-- Register   : AX, SI, BH, DX and FLAGS are changed

grafhi   proc near

```



```

        mov     bl,00010010b        ;Graphic mode with 640x200 pixels
        jmp     short graphic        ;Program video controller again

grafhi    endp

;-- GRAFLO: switches the 320x200 pixel graphic mode on -----
;-- Input    : none
;-- Output   : none
;-- Register : AX, SI, BH, DX and FLAGS are changed

graflo    proc near

        mov     bl,00100010b        ;Graphic mode with 320x200 pixels
graphic:   mov     si,offset graphict ;Offset address of the register table

graflo    endp

;-- VCPROG: programs the video controller -----
;-- Input    : SI = Address of a register table
;--          : BL = Value for display control register
;-- Output   : none
;-- Register : AX, SI, BH, DX and FLAGS are changed

vcprog    proc near

        setmode bl                  ;Bit 3 = 0: screen off

        mov     cx,12                ;12 registers are set
        xor     bh,bh                ;Start with register 0
vcpl:      lodsb                     ;Get register value from table
        mov     ah,al                ;Register value to AH

```

```

mov    al,bh                ;Number of the register to AL
call   setvk                ;Transmit value to controller
inc    bh                   ;Address next register
loop   vcp1                 ;Set additional registers

or     bl,8                  ;Bit 3 = 1: screen on
setmode bl                   ;Set new mode
ret                                ;Back to caller

```

```
vcprog    endp
```

```

;-- SETCOL    : Sets the color of the display frame and Background -----
;-- Input     : AL = color value
;-- Output    : none
;-- register  : AX and DX are changed
;-- Info      : in text mode the lowest 4 bits indicate the frame color
;--            in graphic mode the lowest 4 bits indicate the frame
;--            and background color, bit 5 selects the color palette

```

```
setcol    proc near
```

```

mov     dx,CCHOICE_REG      ;Address of the color selection register
out     dx,al               ;Output color value
ret                                ;Back to caller

```

```
setcol    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

;-- SETPAGE : sets the screen page -----
;-- Input   : BP = Number of the screen page (0 to 3)
;-- Output  : none
;-- register : BX, AX, CX and DX are changed
;-- Info    : in the Graphic modes the first screen page has the
;--          number 0, the second the number 2

setpage    proc near

    mov     bx,bp             ;Screen page to BX
    mov     cl,5              ;Multiply by 2,048
    ror     bx,cl
    mov     al,CURPG_HI       ;Register 12: High byte page address
    mov     ah,bh             ;High byte of the screen page to AH
    call    setvk             ;Transmit to video controller
    mov     al,CURPG_LO       ;Register 13: Low byte page address
    mov     ah,bl             ;Low byte of the screen page to AH
    jmp     short setvk       ;Transmit to video controller

setpage    endp

```

```

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

```

```

setblink  proc near

```

```

    mov  bx,di          ;Move offset to BX
    mov  al,CURPOS_HI   ;High byte of the cursor offset
    mov  ah,bh          ;High byte of the offset
    call setvk          ;Transmit to video controller
    mov  al,CURPOS_LO   ;Low byte of the cursor offset
    mov  ah,bl          ;Low byte of the offset

```

```

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

```

```

setblink  endp

```

```

;-- SETVK    : sets a byte in one register 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       ;Short 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    : gets a byte from one register of the video controller -
;-- Input    : AL = Number of the register
;-- Output   : AL = Contents of 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
        inc  dx              ;Index register address
        jmp  short $+2       ;Short io pause
        in   al,dx           ;Set new contents
        ret                ;Back to caller

getvk   endp

;-- SCROLLUP: scrolls a window N lines upward -----
;-- Input    : BL = line upper left
;--          : BH = column upper left
;--          : DL = line below right
;--          : DH = column below right
;--          : CL = Number of lines, to be scrolled
;--          : BP = Number of the screen page (0 to 3)
;-- Output   : none
;-- register : only FLAGS are changed
;-- Info     : the display lines liberated are cleared

```

```

scrollup  proc near

    cld                                ;On string commands count up

    push ax                            ;All changed registers to the
    push bx                            ;Secure stack
    push di                            ;In this case the sequence
    push si                            ;must be observed !

    push bx                            ;These three registers are returned
    push cx                            ;before the end of the routine
    push dx                            ;From the stack
    sub  dl,bl                         ;Calculate the number of lines
    inc  dl
    sub  dl,cl                         ;Subtract number of lines to be scrolled
    sub  bh,dh                         ;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 in offset
    xchg si,di                        ;Exchange SI and DI

    cmp  waitforit,0                  ;Flicker suppressed?
    je   sup0                         ;No --> SUP0

    waitret                            ;Yes -->Wait for retrace
    setmode 00100101b                ;Disable screen

sup0:  push ds                        ;Store segment register
       push es                        ;On the stack
       mov  ax,VIO_SEG                ;Segment address of the video RAM

```

```

        mov  ds,ax          ;To DS
        mov  es,ax          ;And ES

sup1:   mov  ax,di           ;Record DI in AX
        mov  bx,si           ;Record SI in BX
        mov  cl,dh           ;Number of columns 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  sup1           ;No --> move another line

        pop  es              ;Get segment register from
        pop  ds              ;Stack

        cmp  waitforit,0     ;Flickering suppressed?
        je   sup2           ;No --> SUP2

        setmode 00101101b    ;Yes --> Enable screen

sup2:   pop  dx              ;Get lower right corner back
        pop  cx              ;Return number of lines
        pop  bx              ;Return upper left corner
        mov  bl,dl           ;Lower line to BL
        sub  bl,cl           ;Subtract number of lines
        inc  bl
        mov  ah,07h          ;Color : black on white
        call clear           ;Clear lines

        pop  si              ;CX and DX have already been

```

```

        pop    di                ;Restored
        pop    bx
        pop    ax

        ret                    ;Back to caller

scrollup endp

;-- SCROLLDN: scrolls a window N lines down -----
;-- Input    : BL = line upper left
;--          : BH = column upper left
;--          : DL = line below right
;--          : DH = column below right
;--          : CL = number of lines to be scrolled
;--          : BP = number of the screen page (0 to 3)
;-- Output    : none
;-- register : only FLAGS are changed
;-- Info      : the display lines liberated are cleared

scrolldn proc near

        cld                    ;On string commands count up

        push   ax                ;Record all changed registers
        push   bx                ;On the 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 columns
        inc  dh
        mov  al,bl                ;Record line upper left in AL
        mov  bl,dl                ;Line below right to line below left
        call calo                 ;Convert upper left in offset
        mov  si,di                ;Record address in SI
        sub  bl,cl                ;Subtract number of characters 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
        sub  dl,cl                ;Subtract number of lines to be scrolled

        cmp  waitforit,0          ;Flicker suppressed?
        je   sdn0                 ;No --> SDN0

        waitret                   ;Yes --> Wait for retrace
        setmode 00100101b        ;Disable screen

sdn0:    push ds                  ;Store segment register on the
        push es                  ;Stack
        mov  ax,VIO_SEG          ;Segment address of the video RAM
        mov  ds,ax               ;To DS
        mov  es,ax               ;and ES

sdn1:    mov  ax,di                ;Record DI in AX
        mov  bx,si                ;Record SI in BX
        mov  cl,dh                ;Number of columns in counter
        rep movsw                 ;Move a line
        mov  di,ax                ;Restore DI from AX
        mov  si,bx                ;Restore SI from BX
        sub  di,160               ;Set into next line

```

```

        sub    si,160
        dec    dl                ;processed all lines ?
        jne    sdn1             ;No --> move another line

        pop    es                ;Return segment register from
        pop    ds                ;Stack

        cmp    waitforit,0      ;Flicker suppressed?
        je     sdn2             ;No --> SDN2

        setmode 00101101b      ;Yes --> Enable screen

sdn2:    pop    dx                ;Get lower right corner
        pop    cx                ;Return number of lines
        pop    bx                ;Return upper left corner
        mov    dl,b1            ;upper line to DL
        add    dl,cl            ;Add number of lines
        dec    dl
        mov    ah,07h           ;Color : black on white
        call   clear            ;Erase liberated lines

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

        ret                    ;Back to caller

scrollldn endp

;-- CLS: Clear the screen completely -----

```

```

;-- Input   : BP = number of the screen page (0 or 1)
;-- Output  : none
;-- register : only FLAGS are changed

cls                proc near

    mov  ah,07h                ;Color is white on black
    xor  bx,bx                 ;upper left is (0/0)
    mov  dx,4F18h              ;Lower right is (79/24)

    ;-- Execute Clear -----

cls                endp

;-- CLEAR: fills a designated display area with space characters -----
;-- Input   : AH = attribute/color
;--          BL = line upper left
;--          BH = column upper left
;--          DL = line below right
;--          DH = column below right
;--          BP = number of the screen page (0 to 3)
;-- Output  : none
;-- register : only FLAGS are changed

clear             proc near

    cld                        ;On string commands count up
    push cx                    ;Store all register which are
    push dx                    ;Changed on the stack
    push si
    push di
    push es

```

```

        sub    dl,bl           ;Calculate number of lines
        inc    dl
        sub    dh,bh          ;Calculate number of columns
        inc    dh
        call   calo           ;Offset address of the upper left corner
        mov    cx,VIO_SEG     ;Segment address of the video RAM
        mov    es,cx          ;To ES
        xor    ch,ch          ;High bytes of the counter to 0
        mov    al," "         ;Space character

        cmp    waitforit,0    ;Flickering suppressed?
        je     clear1         ;No --> CLEAR1

        push   dx             ;Store DX on the stack
        waitret               ;Retrace wait
        setmode 00100101b     ;Switch screen off
        pop    dx             ;Return DX from the stack

clear1:  mov    si,di          ;Record DI in SI
        mov    cl,dh          ;Number columns in counter
        rep    stosw          ;Store space character
        mov    di,si          ;Return DI from SI
        add    di,160         ;Set in next line
        dec    dl             ;All lines processed ?
        jne    clear1        ;No --> erase another line

        cmp    waitforit,0    ;Flicker suppressed?
        je     clear2         ;No --> CLEAR2

        setmode 00101101b     ;Enable screen

clear2:  pop    es             ;Get registers from

```

```

        pop    di                ;Stack again
        pop    si
        pop    dx
        pop    cx
        ret                    ;Back to caller

clear    endp

;-- PRINT: outputs a string on the screen -----
;-- Input   :   AH = attribute/color
;--          DI = offset address of the first character
;--          SI = offset address of the strings to DS
;--          BP = number of the screen page (0 to 3)
;-- Output  :   DI points behind the last character output
;-- register : AL, DI and FLAGS are changed
;-- Info    :   the string must be terminated by a NUL-character.
;--          other control characters are not recognized

print    proc near

        cld                    ;On string commands count up
        push    si              ;Store SI, DX and ES on the stack
        push    es
        push    cx
        push    dx
        mov     dx,VIO_SEG      ;Segment address of the video RAM
        mov     cl,waitforit    ;Get WAITFORIT flag
        mov     es,dx           ;First to DX and then to ES

        jmp     short print3     ;Get character and display it

print1    label near

```

```

        or    cl,cl                ;Flicker suppressed?
        je    print2              ;No --> PRINT2

        push ax                    ;Record characters and color
        mov   dx,3DAh              ;Address of the display-status-register
hr1:    in     al,dx                ;Get content
        test  al,1                 ;Horizontal retrace?
        jne   hr1                  ;No --> wait
        cli                     ;permit no further interrupts
hr2:    in     al,dx                ;Get content
        test  al,1                 ;Horizontal retrace?
        je    hr2                  ;Yes --> wait
        pop   ax                   ;Restore characters and color
        sti                     ;Do not suppress Interrupts any more

print2:  stosw                     ;Store attribute and color in V-RAM
print3:  lodsb                     ;Get next character from the string
        or    al,al                 ;Is it NUL
        jne   print1              ;No --> output

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

print    endp

;-- CALO: Converts line and column into offset address -----
;-- Input      : BL = line
;--            : BH = column

```

```

;--          BP = number of the screen page (0 to 3)
;-- Output   : DI = the offset address
;-- register : DI and FLAGS are changed

```

```

calo      proc near

```

```

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

    shl  bx,1        ;Column and line times 2
    mov  al,bh        ;Column to AL
    xor  bh,bh        ;High byte
    mov  di,[lines+bx] ;Get offset address of the line
    xor  ah,ah        ;HI byte for column offset
    add  di,ax        ;Add line and column offset
    mov  bx,bp        ;Screen page to BX
    mov  cl,4         ;Multiply by 4,096
    ror  bx,cl
    add  di,bx        ;Add beginning of screen page to offset
    pop  bx          ;Restore BX from stack
    pop  ax          ;Restore AX from stack
    ret              ;Back to caller

```

```

calo      endp

```

```

;-- CGR: Erase the complete Graphic display -----
;-- Input   : AL = 00H : erase all pixels
;--          FFH : set all pixels
;-- Output   : none
;-- register : AH, BX, CX, DI and FLAGS are changed
;-- Info     : this Function erases the Graphic display in both
;--          Graphic modes

```

```

cgr      proc near

    push es                ;Store ES on the stack
    cbw                   ;Expand AL to AH
    xor di,di              ;Offset address in video RAM
    mov bx,VIO_SEG         ;Segment address screen page
    mov es,bx              ;Segment address into segment register
    mov cx,2000h           ;One page is 8KB words
    rep stosw              ;Fill page
    pop es                 ;Return ES from stack
    ret                   ;Back to caller

cgr      endp

;-- PIXLO: sets a pixel in the 320*200 pixel graphic mode -----
;-- Input   : BP = number of the screen page (0 or 1)
;--          BX = column (0 to 319)
;--          DX = line (0 to 199)
;--          AL = color of the pixels (0 to 3)
;-- Output   : none
;-- register : AX, DI and FLAGS are changed

pixlo    proc near

    push ax                ;Secure AX on the stack
    push bx                ;Note BX on the stack
    push cx                ;Store CX on the stack
    mov cl,7
    mov ah,bl              ;Transmit column to AH
    and ah,11b             ;Column mod 4
    shl ah,1               ;Column * 2

```



```

sub    cl,ah                ;7 - 2 * (column mod 4)
mov    ah,11                ;Bit value
shl    ax,cl                ;Move to pixel position
not    ah                   ;Reverse AH
shr    bx,1                  ;Divide BX by 4 by shifting
shr    bx,1                  ;Right twice
jmp    short spix           ;Set pixel

```

pixlo      endp

```

;-- PIXHI: sets a pixel in the 640*200 pixel graphic mode -----
;-- Input   :   BP = number of the screen page (0 or 1)
;--          :   BX = column (0 to 639)
;--          :   DX = line  (0 to 199)
;--          :   AL = color of the pixels (0 or 1)
;-- Output  :   none
;-- register : AX, DI and FLAGS are changed

```

pixhi      proc near

```

push    ax                  ;Store AX on the stack
push    bx                  ;Note BX on the stack
push    cx                  ;Note CX on the stack
mov     cl,7
mov     ah,b1               ;Transmit column to AH
and     ah,111b             ;Column mod 8
sub     cl,ah               ;7 - column mod 8
mov     ah,1                ;Bit value
shl     ax,cl               ;Move pixel position
not     ah                  ;Reverse AH
mov     cl,3                ;3 shifts
shr     bx,cl               ;Divide BX by 8

```

```

        ;-- set pixel -----
pixhi          endp

;-- SPIX: sets a pixel in the graphic display -----
;-- Input      : BX = column offset
;--            : DX = line (0 to 199)
;--            : AH = Value to cancel old Bits
;--            : AL = new Bit value
;-- Output     : none
;-- register   : AX, DI and FLAGS are changed

spix          proc near

        push es                ;Secure ES on the stack
        push dx                ;Secure DX on the stack
        push ax                ;Secure AX on the stack

        xor di,di              ;Offset address in video RAM
        mov cx,VIO_SEG         ;Segment address screen page
        mov es,cx              ;Segment address into segment register
        mov ax,dx              ;Move line to AX
        shr ax,1               ;Divide line by 2
        mov cl,80              ;The factor is 90
        mul cl                 ;Multiply line by 80
        and dx,1               ;Line mod 2
        mov cl,3               ;3 shifts
        ror dx,cl              ;Rotate right (* 2000H)
        mov di,ax              ;80 * int(line/2)
        add di,dx              ;+ 2000H * (line mod 4)
        add di,bx              ;Add column offset

```

```

        pop    ax                ;Return AX from stack
        mov    bl,es:[di]        ;Get pixel
        and    bl,ah             ;Erase Bits
        or     bl,al             ;Add pixel
        mov    es:[di],bl       ;write pixel back

        pop    dx                ;Return DX from stack
        pop    es                ;Return ES from stack
        pop    cx                ;Return CX from stack
        pop    bx                ;Return BX from stack
        pop    ax                ;Return AX from stack

        ret                     ;Back to caller

spix    endp

;== end =====

code    ends                    ;End of the code segment
        end    demo

```