

```

***** V 3 2 4 0 P A . A S M *****
;*
;*-----*
;* Task : Contains routines for operating in 320x400 *
;* 256 color mode on a VGA card. *
;*-----*
;* Author : Michael Tischer *
;* Developed on : 09/05/90 *
;* Last update : 02/24/92 *
;*-----*
;* Assembly : MASM /mx V3240PA; or TASM -mx V3240PA *
;* ... link to V3240P.PAS *
;*****;

;== Constants =====

SC_INDEX = 3c4h ;Index register for sequencer ctrl.
SC_MAP_MASK = 2 ;Number of map mask register
SC_MEM_MODE = 4 ;Number of memory mode register

GC_INDEX = 3ceh ;Index register for graphics ctrl.
GC_READ_MAP = 4 ;Number of read map register
GC_GRAPH_MODE = 5 ;Number of graphics mode register
GC_MISCELL = 6 ;Number of miscellaneous register

CRTC_INDEX = 3d4h ;Index register for CRT controllers
CC_MAX_SCAN = 9 ;Number of maximum scan line reg.
CC_START_HI = 0Ch ;Number of high start register
CC_UNDERLINE = 14h ;Number of underline register
CC_MODE_CTRL = 17h ;Number of mode control register

DAC_WRITE_ADR = 3C8h ;DAC write address
DAC_READ_ADR = 3C7h ;DAC read address
DAC_DATA = 3C9h ;DAC data register

VERT_RESCAN = 3DAh ;Input status register #1

PIXX = 320 ;Horizontal resolution

;== Data segment =====

DATA segment word public

vio_seg dw (?) ;Video segment with current page
;must be initialized during runtime

DATA ends

;== Program =====

CODE segment byte public ;Program segment

assume cs:code, ds:data

;-- Public declarations -----

public init320400 ;Initialize 320x400 mode
public setpix ;Set pixel
public getpix ;Get pixel color
public showpage ;Display page 0 or 1
public setpage ;Set page for setpix and getpix

;-----
;-- INIT320400: Initializes 320x400 pixel mode
;-- Call from TP: init320400;

init320400 proc near

;-- Sets mode 13H, since BIOS uses this mostly for
;-- initialization.

mov ax,0013h ;Set normal mode 13H
int 10h

mov dx,GC_INDEX ;Memory division
mov al,GC_GRAPH_MODE ;Disable bit 4 of
out dx,al ;graphic mode register
inc dx ;in graphics controller
in al,dx
and al,11101111b
out dx,al
dec dx

mov al,GC_MISCELL ;And change bit 1
out dx,al ;in the miscellaneous
inc dx ;register

```

```

in      dx
and     al,11111101b
out     dx,al

mov     dx,SC_INDEX      ;Modify memory mode register in
mov     al,SC_MEM_MODE   ;sequencer controller so no further
out     dx,al            ;address division follows in
inc     dx               ;bitplanes, and set the bitplane
in      al,dx            ;currently in the
and     al,11110111b     ;bit mask register
or      al,4
out     dx,al

mov     ax,0A000h        ;Fill all four bitplanes with color
mov     vio_seg,ax       ;code 00H and clear the screen
mov     es,ax
xor     di,di
mov     ax,di
mov     cx,8000h
rep     stosw

mov     dx,CRTC_INDEX    ;Double pixel rows in maximum
mov     al,CC_MAX_SCAN   ;scan line register of the CRT
out     dx,al            ;controller by disabling bit 7,
inc     dx               ;while setting bits 0-4 to 1 to
in      al,dx            ;change the character
and     al,01110000b     ;height
out     dx,al
dec     dx               ;Return DX to CRT index register

mov     al,CC_UNDERLINE  ;Set double word mode using bit 6
out     dx,al            ;in underline register of
inc     dx               ;CRT controller
in      al,dx
and     al,10111111b
out     dx,al
dec     dx

mov     al,CC_MODE_CTRL  ;Using bit 6 of mode control reg.
out     dx,al            ;of CRT controller, change
inc     dx               ;from word mode to byte mode
in      al,dx
or      al,01000000b
out     dx,al

ret                                     ;Return to caller

init320400 endp                    ;End of procedure

;-----
;-- SETPIX: Changes a pixel to a specific color
;-- Call from TP: setpix( x , y : integer; pcolor : byte );

setpix    proc near

sframe    struc                ;Structure for stack access
bp0        dw ?                ;Gets BP
ret_adr0    dw ?                ;Return address to caller
pcolor      dw ?                ;Color
y0          dw ?                ;Y-coordinate
x0          dw ?                ;X-coordinate
sframe     ends                ;End of structure

frame     equ [ bp - bp0 ]      ;Address structure elements

push     bp                    ;Prepare for parameter addressing
mov      bp,sp                 ;through BP register

mov      ax,PIXX / 4            ;Compute offset in video RAM
mul      frame.y0               ;and pass to DI
mov      cx,frame.x0
mov      bx,cx
shr      bx,1
shr      bx,1
add      ax,bx
mov      di,ax

and      cl,3                   ;Compute bit mask for map to be
mov      ah,1                   ;addressed, move to AH
shl      ah,cl
mov      al,SC_MAP_MASK        ;Register number to AL
mov      dx,SC_INDEX           ;Load sequencer index address
out      dx,ax                 ;Load bit mask register

mov      ax,vio_seg             ;Set ES to video RAM
mov      es,ax

```

```

        mov     al,byte ptr frame.pcolor ;Load pixel color and
        stosb                                ;write to selected bitmap

        pop     bp                            ;Get registers from stack

        ret     6                            ;Return to caller, remove
                                           ;arguments from stack

setpix    endp                                ;End of procedure

;-----
;-- GETPIX: Returns a pixel color
;-- Call from TP: x := getpix( x , y : integer );

getpix    proc near

sframe1   struc                                ;Structure for stack access
bp1        dw ?                                ;Gets BP
ret_adr1   dw ?                                ;Return address to caller
y1         dw ?                                ;Y-coordinate
x1         dw ?                                ;X-coordinate
sframe1    ends                                ;End of structure

frame     equ [ bp - bp1 ]                    ;Address structure elements

        push   bp                            ;Prepare for parameter addressing
        mov    bp,sp                          ;through BP register

        mov    ax,PIXX / 4                    ;Compute offset in video RAM
        mul    frame.y1                       ;and pass to SI
        mov    si,frame.x1
        mov    cx,si
        shr    si,1
        shr    si,1
        add    si,ax

        and    cl,3                            ;Compute bit mask for map to be
        mov    ah,cl                          ;addressed in AH
        mov    al,GC_READ_MAP                 ;Move register number to AL
        mov    dx,GC_INDEX                    ;Load graphics controller index
        out    dx,ax                          ;Load Read Map register

        mov    ax,vio_seg                     ;Set ES to video RAM
        mov    es,ax
        mov    al,es:[si]                     ;Load pixel color

        pop    bp                            ;Pop register from stack

        ret    4                            ;Return to caller, remove
                                           ;arguments from stack

getpix    endp                                ;End of procedure

;-----
;-- SETPAGE: Sets page for access from setpix and getpix
;-- Call from TP: setpage( page : byte );

setpage    proc near

        pop    ax                            ;Pop return address from stack
        pop    cx                            ;Pop argument from stack

        push   ax                            ;Push this back on

        mov    bl,0a0h                        ;Check for page 0
        or     cl,cl                          ;Is this page 0?
        je     sp1                            ;Yes --> Store segment address

        mov    bl,0a8h                        ;No --> Page 1

sp1:       mov    byte ptr vio_seg + 1,bl ;Move new segment address

        ret                                ;Return to caller, remove
                                           ;arguments from stack

setpage    endp                                ;End of procedure

;-----
;-- SHOWPAGE: Display one of the two screen pages
;-- Call from TP: showpage( page : byte );

showpage    proc near

        pop    bx                            ;Pop return address from stack
        pop    ax                            ;Pop argument from stack

```

```

push    bx                ;Push these back in
or      al,al             ;Is this page 0?
je      sp2               ;Yes --> Number is also
                        ; high-byte of offset

mov     al,80h            ;No --> Page 1 with offset 8000H

sp2:    mov     dx,CRTC_INDEX ;Address CRT controller
        mov     ah,al        ;Move high byte of offset to AH
        mov     al,CC_START_HI ;Move register number to AL
        out     dx,ax        ;and exit

        ;-- Wait to return to starting screen design -----

sp3:    mov     dx,VERT_RESCAN ;Wait for end of
        in      al,dx        ;vertical rescan
        test    al,8
        jne     sp3

sp4:    in      al,dx        ;Go to start of rescan
        test    al,8
        je      sp4

        ret                ;Return to caller, remove
                        ;arguments from stack

showpage    endp          ;End of procedure

;== End =====
CODE        ends          ;End of code segment
end         end           ;End program

```