Buletinul Ştiinţific al Universităţii din Baia Mare
Seria B,Matematică-Informatică,vol.XI(1995),111-120

Dedicated to the Centenary of "Gazeta Matematică"

A SCREEN PAGING FACILITY FOR IBM PC COMPATIBLE COMPUTERS

Ovidiu COSMA

This paper presents a way to benefit from the excess of video memory in text modes, allowing the selection of the current video page that DOS will use. Up to eight text screens can be stored in the video buffer, with any one of them visible at any time. Normally DOS uses only page 0.

The video buffer

In text modes, each character on the screen occupies 2 bytes in the screen buffer. The first byte contains the ASCII code of the character, and the second contains the display attributes. Thus the 80 x 25 character modes screen page takes 4000 bytes for representation, and in the 40 x 25 character modes, each page occupies 2000 bytes. Therefore the maximum number of pages available for each type of video interface depends on the capacity of the video buffer and on the video display mode.

The monochrome adapter has 4K bytes of on board memory, starting from address B000:0000H. This amount of memory provides enough space for only one 80 column page of text.

The CGA adapter has 16K of memory starting from address B800:0000H. This is enough for 4 to 8 text screens depending whether they are displayed in 40 or 80 columns.

The EGA / VGA adapters may be equipped with 64K, 128K, 256K, ... of RAM. Besides serving as the video buffer, this memory also holds the data for the patterns of up to 1024 characters. The starting address of the buffer is also programmable so that it begins at A000:0000H for the advanced graphics modes and at B000:0000H and B800:000H for compatibility with the older adapters. At most, the EGA occupies the 2 segments from A000H to BFFFH, even when 256K of memory is present. This is possible because in some modes two or more bytes of video memory are accessed by the same memory address.

Switching video pages

There are two ways for selecting the page currently displayed. The first one is by using the BIOS INT 10H, function 5. The number of the new page is passed threw the AL register.

```
MOV  AH,5           ;function number
MOV  AL,page_no     ;number of new page
INT  10H            ;set the new page
```

All the BIOS interrupts that write on the screen (functions of INT 10), requires the number of the page to write in. This parameter is passed threw one of the CPU registers. On the other hand, all the DOS screen interrupts write in the page currently in view.

The other way for selecting the displayed video page is by

changing the start address in the video buffer. The start address is the address in the video buffer from where begins the display of data on the screen. Registers number 12 and 13 of the 6845 video controller chip keep the start address. The text on the display can be scrolled up and down without moving it around in the buffer, by changing the start address of the displayed area. This technique is known as hardware scrolling. The following section sets the start address:

```
MOV   BX, start_adr
MOV   DX,304      ;output to the address register
MOV   AL,12
OUT   DX,AL       ;selects register number 12
INC   DX          ;next output to command registers
MOV   AL,BH       ;high portion of the start address
OUT   DX,AL
DEC   DX
MOV   AL,13       ;selects register number 13
OUT   DX,AL
INC   DX
MOV   AL,BL       ;low portion of the start address
OUT   DX,AL
```

The keyboard interrupt

Each time a key is pressed, the keyboard processor deposits a scan code in port A of the 8255 chip (at address 60H). Than the 8255 chip invokes INT 9. The job of this interrupt is to convert the scan code into a character code, and to place the character code in the keyboard buffer, located at 0000:041EH. When the code is for SHIFT, CTRL or ALT keys no character is send to the buffer, but the interrupt makes changes in two status bytes located in the BIOS data area. The BIOS and DOS keyboard interrupts read the contents of the keyboard buffer maintained by the INT 9.

Overriding the keyboard int 9

This technique is used at the installation of resident programs that will be activated by specific combinations of keys. The int 9 is redirected to the new code that detects the specific combinations of keys and performs the required operations. On exit the control is passed to the original int 9 service routine. The following example modifies the original int 9 vector:

```
old_int9: dd   ?
    . . .
    MOV  AX,3509H              ;DOS function 35h and int 9 vector
    INT  21H                   ;get int 9 vector
    MOV  WORD old_int9, BX
    MOV  WORD old_int9[2], ES      ;save old interrupt vector
    PUSH CS
    POP  DS
    MOV  DX, offset start_driver  ;DS:DX pair contains the new int
                                  ;9 service routine
    MOV  AX,2509H                 ;DOS function 25h and int 9
    INT  21H                      ;set  new int 9 vector
```

Writing a TSR program

Terminating a program with int 20H causes it to stay resident. In the normal termination of an EXE program, the final RET instruction initializes the IP with the first values pushed in the stack: DS, 0. Since DS initially points to the bottom of the PSP, when this values are popped, the instruction pointer is directed to offset 0 in the PSP, which is initialized to contain the code INT 20H. INT 20H is the standard function for terminating programs and returning control to DOS. There is an interesting way of making INT 20H work with EXE programs by putting 27H in the second byte of the

PSP (the first contains the machine code for INT) and end the program with RET. In this way , on exit INT 27H will be executed, that will leave the program code resident in memory. The DX register must contain the offset of the end of the program, starting from the beginning of the PSP.

There is also an easier way of leaving a program resident, by using DOS function 31H. In this case, DX register must contain the size of the resident portion of the program in 16 bytes paragraphs.

```
MOV  DX, offset end_driver
ADD  DX,100H          ;length of PSP
MOV  CL,4
SHR  DX,CL            ;DX = number of 16 bytes
                      ;paragraphs
INC  DX               ;round the number of paragraphs
MOV  AH,31H
INT  21H
```

The following program is a little TSR which allows the user to flip between the video pages by holding down the ALT key and pressing F1 for the first page, F2 for the second, ...,and F8 for the eight's. This is useful in some cases when working in DOS, someone needs to keep the information on the screen for latter use. In this cases he can continue his work in another video page, and then come back to the previous page when he needs the information. This driver occupies only a couple of hundred bytes of memory.

```
COD segment
assume cs:COD, ds:COD
inceput LABEL WORD
     jmp  loader          ;the first 4 memory locations will
                          ;be used latter to keep the
     db   ?               ;original interrupt handler routine
```

```
st_alt    db   0          ;keep alt state

driver:
     pushf
     cmp  ax,0aaaah      ;test if driver already installed?
     jnz  nu_e_test      ;if not, continue with the driver
     mov  ax,0ffffh      ;answer "yes, driver already installed"
     popf
     iret

nu_e_test:                    ;begins the operation
     popf
     sti                      ;allow other interrupts
     push ax                  ;save the general registers
     push ds
     pushf

     push    cs
     pop     ds               ;initialize data segment

     in   al,60h              ;get key scan code
     cmp  al,38h              ;was the ALT key pressed?
     jz   set_alt             ;set ALT flag

     cmp  al,0b8h             ;was the ALT key released?
     jz   res_alt             ;reset alt flag

     mov  ah,cs:st_alt        ;an other key (different from ALT)
                              ;was pressed
     cmp  ah,0                ;is ALT released?
     jz   gata                ;if ALT is released do nothing
     cmp  al,3bh              ;if scan code is less than F1 code,
exit
     jl   gata
     cmp  al,42h    ;if scan code is greater than F8 code, exit
```

```
    jg   gata       ;the function keys F1 - F8 have
                    ;codes between 3bh and 42h

    sub  al,3bh                 ;F<n> pressed, n=number of page
    mov  ah,5                   ;function to switch video pages
    int  10h                    ;set page no.
    jmp  gata

res_alt:
     mov  BYTE PTR cs:st_alt,0
     jmp  gata
set_alt:
     mov  BYTE PTR cs:st_alt,1

                            ;exit
gata:
     cli
     popf
     pop  ds
     pop  ax                ;restore the general registers
     jmp  DWORD PTR cs:inceput ;to the original interrupt routine
gata_driver:                ;end of driver

;the following section will load the driver in memory
loader:
     push cs
     pop  ds

     mov  ax,0aaaah         ;test if the driver is already installed
     int  9
     cmp  ax,0ffffh         ;if the driver is installed, it returns
                            ;0ffffh in ax
     jnz  nu_e_inst         ;go to installation

     lea  dx,m_deja
     mov  ax,900h
```

```
        int  21h                ;display message "driver already
                                ;installed"

        mov  ax,4c02h
        int  21h                ;terminate program, return to DOS

;here starts the installation process
nu_e_inst:
        lea  dx,m_inst
        mov  ax,900h
        int  21h                ;display initial message

        mov  ax,3509h           ;function number 35, and int 9
        int  21h                ;get old keyboard interrupt vector

        mov  cs:inceput,bx
        mov  cs:inceput[2],es   ;save interrupt vector

        push cs
        pop  ds                 ;reinitialize data segment

        mov  dx,offset driver
        mov  ax,2509h           ;function number 25, and int 9
        int  21h                ;set new interrupt vector

        mov  dx,offset gata_driver    ;length of the driver
        add  dx,100h                  ;+100h for PSP
        mov  cl,4
        shr  dx,cl              ;number of paragraphs = length / 16
        inc  dx                 ;round number of paragraphs
        mov  ah,31h
        int  21h                ;terminate and stay resident

m_inst:
     db   "Screen page flipper driver installed"
```

```
        db    0dh,0ah
        db    "Press ALT F<n> where n = number of video page"
        db    0dh,0ah
        db    "$"

m_deja:
        db    "Driver already installed"
        db    0dh,0ah
        db    "$"

COD ends
    end
```

REFERENCES

1.INTEL, " MCS-86 Macro Assembly Language Reference Manual", 1979

2.IBM, " Personal Computer Hardware Reference Library", 1984

3.Brady Communications Company, "Programmer's Problem Solver for the IBM PC, XT & AT", Prentice Hall Press, 1986

4.MICROSOFT, "MS DOS Programmer's Reference Guide".

University of Baia Mare
Victoriei 76, 4800 Baia Mare
ROMANIA