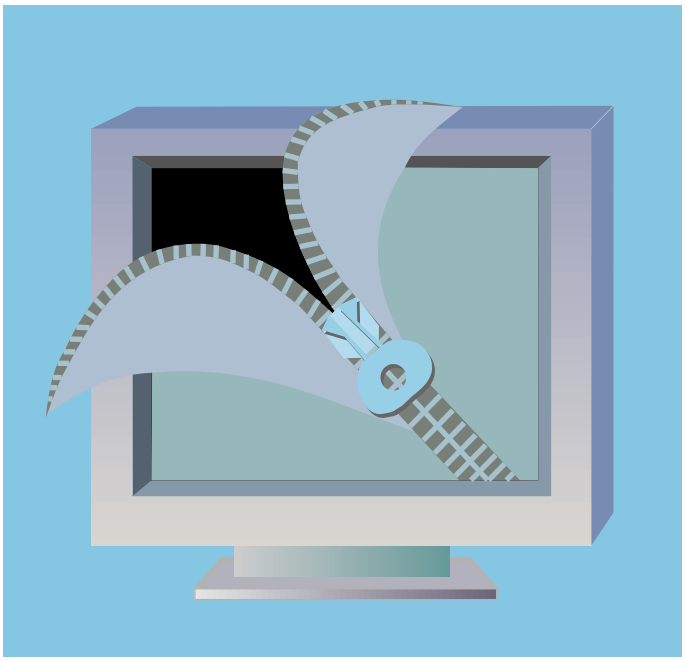
Ingo Eickmann

Screen Splitting

Bildschirmteilung bei EGA- und VGA-Adaptern

EGA- und VGA-Adapter erlauben, gleichzeitig zwei Bildschirmseiten im Videospeicher abzulegen. Dies setzt eine zum Kathodenstrahl des Monitors synchronisierte Programmierung voraus. Assembler-, Turbo-Pascal- und MS-C-Freunde finden hier konkrete Programmbeispiele.



Die Aufteilung des sichtbaren Bildschirms auf zwei unterschiedliche Bereiche des Videospeichers ist nicht neu. Diese Möglichkeit besteht seit der Markteinführung der EGA, doch bisher hat sich kaum jemand ernsthaft damit beschäftigt. Uns sind lediglich zwei Programme -und zwar EGA-/VGA-Demoprogramme - bekannt, in denen diese Eigenschaft benutzt wird. Die geringe Bedeutung des Screen-Splittings ist wohl darauf zurückzuführen, daß erst wenige Programmierer hiervon Kenntnis haben. Anwendungsmöglich-keiten für eine hardwaremäßige Aufteilung des sichtbaren Bildschirms gibt es ja genug (siehe auch [1]). Die beiden Schlüsselregister für das Screen Splitting sind das Start-Adreßregister (Index 0Ch und 0Dh) und das Line Compare Register (Index 18h) des Cathode Ray Tube Controllers, abgekürzt CRTC. Das Start-Adreßregister enthält den Offset des Bytes, das in der linken oberen Ecke des Bildschirms dargestellt wird. Beispielsweise beginnt der Speicherbereich der ersten Bildschirmseite im Videomodus 3 (80x25 Color) bei B800:0000h, der Inhalt des Start-Adreßregisters ist 0000h. Wird der Inhalt der zweiten Bildschirmseite, der bei B800:1000h beginnt, dargestellt, muß das Start-Adreßregister auf 1000h gesetzt werden. Das Start-Adreßregister ist aufgeteilt auf den Index 0Ch des CRTC für das MSB und auf den Index 0Dh für das LSB. Die CRTC-Register werden programmiert, indem man den Index in Port 3D4h - oder in den monochromen Modi - in Port 3B4h schreibt und den neuen Registerinhalt in Port 3D5h beziehungsweise 3B5h. Bei EGA-Karten kann der adressierte Wert in Port 3D5h/3B5h nur geschrieben werden, bei VGA-Adaptern sind die Register auch lesbar.
Der CRTC enthält einen internen Adreßzähler, der während des Bildaufbaus automatisch mitgezählt wird und immer auf den gerade dargestellten Speicherinhalt zeigt. Gleichzeitig wird die Nummer der vom Kathodenstrahl gezeichneten Rasterzeile mitgezählt.

CRTC-Register

Stimmt die Anzahl der gezeichneten Rasterzeilen mit dem Wert im Line-Compare-Register (CRTC Index 18h) überein, so wird für die nächste darzustellende Rasterzeile der interne Adreßzähler auf 0000h zurückgesetzt. Von nun ab wird im verbleibenden Bildschirmbereich der Anfang des Videobuffers dargestellt. Bei gesplittetem Bildschirm wird im unteren Teil also immer der Anfang des Videobuffers gezeigt. Dies verdeutlicht *Bild 1*.
Das Screen Splitting arbeitet völlig unabhängig vom Inhalt des Videospeichers, hier sind keine Modifikationen notwendig. Es ist ebenfalls unabhängig vom Videomodus: Sowohl Text- als auch Grafikbildschirme können geteilt werden. Lediglich die Wahl der Startadresse für den oberen Bildschirmteil muß abhängig vom Videomodus und dem, was man darstellen will, gebildet werden. Im unteren Teil ist jedoch immer der Anfang des Videospeichers zu sehen. Die Voreinstellung für den Line-Compare-Vorgang ist die bezüglich der Registergröße maximal mögliche: Bei der EGA 1FFh, bei der VGA 3FFh. Über Index 18h des CRTC lassen sich lediglich die untersten 8 Bit - Bit 0 bis 7 - des Line-Compare-Registers adressieren. Das Bit 8 ist in das Overflow-Register (CRTC Index 7) als Bit 4 eingebaut worden. Bei der VGA mußte auch noch Platz für Bit 9 geschaffen werden, es ist als Bit 6 an das Max-Scan-Register (CRTC Index 9) angehängt worden. Das schaut sehr zusammengeflickt aus, war aber aus Kompatibilitätsgründen leider nicht anders möglich. Die Werte, mit denen man das Line-Compare-Register belegt,

sollten sehr sorgfältig gewählt werden. Grundsätzlich sind alle Werte von 0 bis zum Start des Vertical Retrace zulässig. Der maximale Wert (1FFh oder 3FFh) sollte verwendet werden, um das Splitting aufzuheben. Rasterzeilennummern zwischen dem Vertical Retrace und den Vertical Total Lines (CRTC Index 6) sind zu vermeiden. In den 200-Zeilen-Modi der VGA sind nur gerade Werte für das Line-Compare-Register sinnvoll, da jede Rasterzeile dupliziert wird. Darüber hinaus gibt es eine Anomalie, die man im Zweifelsfall selbst ausprobieren sollte: Bei einigen EGA-Karten beginnt der Controller in den Grafikmodi erst bei der doppelten Zahl von Rasterzeilen zu splitten.

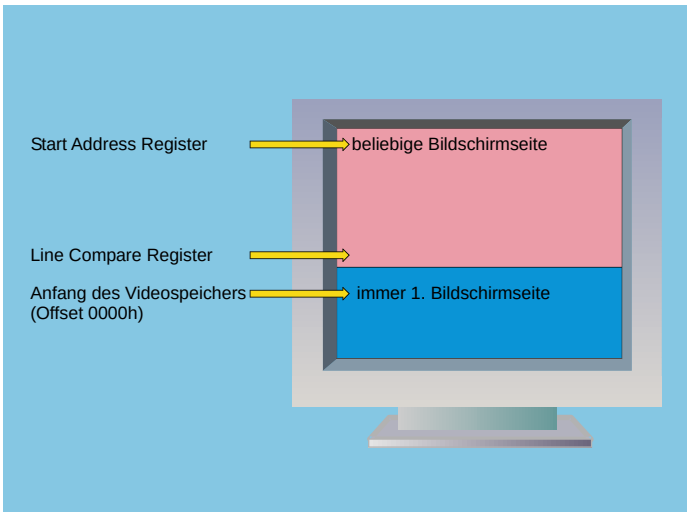


Bild 1. Der obere Bildschirmteil beginnt bei der Speicheradresse im Start-Adreßregister. Nachdem so viele Rasterzeilen dargestellt sind, wie im Line-Compare-Register vereinbart wurden, beginnt die Ausgabe vom Anfang des Videospeichers

Kathodenstrahlsynchronisation

Das Listing in *Bild 2* enthält die Programmierung der beiden CRTC-Register als Makro. Die Synchronisation der Lese- und Schreibvorgänge ist wichtig, um mit dem videokarteninternen Zugriff auf diese Register nicht zu kollidieren. Das Start-Adreßregister wird während der Vertical-Retrace-Phase, der vertikalen Rückführung des Elektronenstrahls, in den internen Adreßzähler übertragen. Daher wird für die Programmierung dieses Registers auf die nächste fallende Flanke des Vertical-Retrace-Bits im Input-Status-Register 1 gewartet. Das Line-Compa-re-Register wird dagegen während des Bildschirmaufbaus nach jeder Rasterzeile zum Vergleich gelesen, es kann nur während der vertikalen Rückführungsphase des Kathodenstrahls neu geschrieben werden. Im Makro Split_Screen wird daher auf die darauffolgende steigende Flanke des Vertical-Retrace-Signals gewartet, bevor das Line-Compare-Register seinen neuen Wert erhält. Das Listing in *Bild 2* ist für den Betrieb einer VGA-Karte ausgelegt. Dies gilt auch für *Bild 3*. Anhand der Kommentare lassen sich die wenigen Änderungen für die Benutzung einer EGA-Farbkarte leicht durchführen. Die Schnappschuß-Funktion, die im Juni [1] für zwei Videokarten im PC vorgestellt wurde, ist in *Bild 3* für das Screen Splitting neu geschrieben worden. So kommen auch die Benutzer eines EGA- oder VGA-Adapters in den Genuß, Momentaufnahmen von ihrem Bildschirm machen zu können. Die aktive Bildschirmseite 1 wird hierzu mit der Tastenkombination ALT S auf die 2. Bildschirmseite kopiert. Sie kann mit ALT U vor der 2. Bildschirmseite hoch und runter oder ALT D gefahren werden. Das Schnappschuß-Programm ist nur für den meistbenutzten Textmodus 3 (80x25 Color) ausgelegt. Wenn der Bereich und die Größe des Bildschirmbuffers vom Videomodus abhängig gemacht werden, läßt sich das Programm an weitere Text- und Grafikmodi anpassen. Zu beachten ist allerdings, daß man im gesamten Bereich des virtuellen Bildschirms nicht das Ende des Bildschirmspeichers oder das Ende des 64-KByte-Segments überschreitet. Anhänger von Turbo-Pascal mögen ihren Blick auf *Bild 4* lenken. Dort ist Split_Screen als Prozedur in einer Unit deklariert. MS-C-Benutzer können die gleiche Deklaration verwenden, lediglich der Segmentname code ist durch _TEXT zu ersetzen und die Parameterübergabe der zwei Integer-Werte an das gewünschte Speichermodell anzupassen. Erster Parameter ist jeweils die neue Startadresse für die Bildschirmausgabe, zweiter Parameter ist die maximale Rasterzeile des oberen Bildschirmbereichs.

Literatur

[1] *Eickmann, I.*: Zwei Videokarten im PC.
mc 6/89, S. 99.

[2] *Wilton, R.*: The programmer's guide to PC and PS/2 video Systems.
Microsoft Press, 1987.

```
;------------------------------------------------------------------------
;     Split Screen - Macros                                  release 1.0
;     Copyright (c) Ingo Eickmann, 1989                        06/14/89
;------------------------------------------------------------------------

IRP com,<push,pop>                    ; Macros zum push-en und pop-en von
   REG_&com macro paras               ; Registern
      IRP n,<paras>
         &com &n
      endm
   endm
endm

Vertical_retrace macro n              ; Vertical Retrace (Bit 3) im Input
local Loco                            ; Status Register 1 abfragen und
Loco: in al,dx                        ; auf 0 bzw. 1 warten
      test al,00001000b
      j&n Loco
endm

Split_Screen macro Start,Line         ; Teilung des Bildschirms:
  local LC1,LC2                       ; Start: Bildschirmspeicheroffset
     mov  ax,40h                      ;        für oberen Bildschirmteil
     mov  es,ax                       ; Line: letzte Scan Line für oberen Teil
     mov  dx,es:[63h]                 ; dx: CRTC address port (3B4h oder 3D4h)
     add  dx,6                        ; dx: CRT Status port (3BAh oder 3DAh)
     Vertical_retrace z               ; Warten, bis Vertical Retrace aktiv
     Vertical_retrace nz              ; Marten, bis Vert. Retrace nicht aktiv
     cli
     sub  dl,6
     mov  ax,Start                    ; Startadresse für oberen Bildteil
     shr  ax,1
     mov  bl,al
     mov  al,0Ch
     out  dx,ax                       ; MSB in CRTC - Register 0Ch
     inc  al
     mov  ah,bl
     out  dx,ax                       ; LSB in CRTC - Register 0Dh
     sti
     Vertical_retrace z               ; Warten, bis Vertical Retrace aktiv
     cli
     mov  ax,Line                     ; letzte Scan Line für oberen Bildteil
     mov  bh,ah
     mov  bl,ah
     and  bx,0201h                    ; Bit 8 und 9 trennen und aufbereiten
     mov  cl,4
     shl  bx,cl
     shl  bh,1
     mov  ah,18h
     xchg ah,al
     out  dx,ax                      ; Bits 7-0 ins Line Compare Register
     mov  al,7                       ; schreiben
     out  dx,al                      ; Overflow - Register adressieren
     inc  dx
     in   al,dx                      ; ! EGA (350 Zeilen-Mode): mov al,1Fh !
                                     ; ! EGA (200 Zeilen-Mode): mov al,11h !
     and  al,11101111b               ; Bit 8 des Line Compare Wertes in
     or   al,bl                      ; Bit 4 des Overflow - Registers
     out  dx,al                      ; schreiben
     dec  dx

     mov  al,9                       ; ! entfällt für EGA:                 !
     out  dx,al                      ; !                                   !
     inc  dx                         ; ! Bit 9 des Line Compare Wertes     !
     in   al,dx                      ; ! in Bit 6 des Max Scan Line        !
     and  al,10111111b               ; ! Registers schreiben               !
     or   al,bh                      ; !                                   !
     out  dx,ax                      ; ! bis hier entfällt es für die EGA  !
     sti
endm
```

Bild 2. Das Include-File SPLIT.MAC. Für EGA-Karten müssen die Änderungen in den Kommentaren berücksichtigt werden

```
;-------------------------------------------------------------------------
;     Split_Screen - SnapShot                                  release 1.0
;     Copyright (c) Ingo Eickmann, 1989                          06/14/89
;-------------------------------------------------------------------------
;     Assemblieren mit MASM 5.x :
;            MASM split;                                 I. Eickmann
;            LINK split;                                 Im Leuchterbruch 8
;            EXE2BIN split.exe split.com                 5000 Köln 80
;     Getestet unter MS-DOS 3.3
;-------------------------------------------------------------------------
      PAGE   65,80
      TITLE  Split_Screen

ALT_U       equ 1600h                 ; Scan-Code von <ALT-U>
ALT_D       equ 2000h                 ; Scan-Code von <ALT-D>
ALT_S       equ 1F00h                 ; Scan-Code von <ALT-S>
Snap_Buffer equ 1000h                 ; Offset der 2. Bildschirmseite
max_Lines   equ 400                   ; für VGA-Textmode, EGA-Textmode: 350
if1
   include  split.mac
endif
;-------------------------------------------------------------------------
KBDSEG segment at 40h                 ; Einträge im System-Segment
   org 1Ah
      HEAD   dw ?                     ; Vektor auf ältesten Eintrag im Buffer
      TAIL   dw ?                     ; Vektor auf freien Platz im Buffer
      Buffer dw 16 dup(?)             ; Tastatur-Buffer
   org 80h
      Buffer_Start dw ?               ; erste Speicherstelle des Buffers
      Buffer_End   dw ?               ; letzte Speicherstelle + 1
KBDSEG ends
;-------------------------------------------------------------------------
code segment                          ; Segment des Programms
     assume cs:code,ds:code,es:KBDSEG
     org 100h                         ; Startadresse für .com-File
start:    jmp Install
;-------------------------------------------------------------------------
oldint09h    dd 0                     ; Vektor auf alte Interruptroutine 09h
oldint1Ch    dd 0                     ; Vektor auf alte Interruptroutine 1Ch
Line_Compare dw -1                    ; Wert für Line_Compare_Register
Add_Word     dw 0
;-------------------------------------------------------------------------
INT09h proc near                      ; Erweiterung für Tastaturinterrupt
          pushf                       ; retten aller Register und Flags
          REG_PUSH <ax,bx,cx,dx,di,si,ds,es>
          pushf                       ; Aufruf der alten ISR 09h
          call  dword ptr cs:[oldint09h]
          cli
          push  cs                    ; Vorbelegung der Segmentregister
          pop   ds
          mov   ax,KBDSEG
          mov   es,ax
          cmp   byte ptr es:[49h],3   ; Text-Mode 80x25 Color?
          jne   Raus
          mov   si,[Tail]             ; Vektor auf letzten Eintrag im
          cmp   si,[Head]             ; Tastatur-Buffer
          jz    Raus                  ; Scan-Code vorhanden? - Nein => Raus
          cmp   si,[Buffer_Start]
          jg    ok2
          mov   si,[Buffer_End]
ok2:      sub   si,2                  ; Scan-Code entnehmen
          mov   ax,es:[si]            ; und in ax speichern
          cmp   ax,ALT_U              ; Ist Scan-Code ALT-U?
          jne   ok3
          sub   Add_word,1            ; JA: nach oben beschleunigen
          jmp   Ende
ok3:      cmp   ax,ALT_D              ; Ist Scan-Code ALT-D?
          jne   ok4
          add   Add_word,1            ; JA: nach unten beschleunigen
          jmp   Ende
ok4:      cmp   ax,ALT_S              ; Ist Scan-Code ALT-S?
          jne   Raus
          REG_PUSH <si,ds,es>         ; JA: Register speichern
          mov   ax,0B800h             ;     Video-Segment laden
          mov   ds,ax
          mov   es,ax
          mov   si,0                  ;     Offset der 1. Bildschirmseite
          mov   di,Snap_Buffer        ;     Offset des Zwischenspeichers
          mov   cx,1000h shr 1        ;     Zähler auf 1000h Bytes
          cld
          repnz movsw                 ;     Bildschirmseite kopieren
```

```
          REG_POP <es,ds,si>           ;     Register wiederherstellen
Ende:     mov   [Tail],si              ; Vektor auf nächsten Eintrag zurück-
Raus:     REG_POP <es,ds,si,di,dx,cx,bx,ax>                  ; schreiben
          popf                         ; alle Register wiederherstellen
          iret                         ; Rücksprung aus der ISR
INT09h endp
;--------------------------------------------------------------------------
INT1Ch proc near                       ; Erweiterung des Zeitgeber-Interrupts
          sti
          pushf                        ; Register retten
          REG_PUSH <ax,bx,cx,dx,ds,es,cs>
          pop   ds                     ; Segmentregister vorbelegen
          mov   ax,Add_word
          or    ax,ax                  ; ist Verschiebung notwendig?
          jnz   pt0
          jmp   no_mov                 ; NEIN: => Raus
pt0:      add   ax,Line_Compare        ; Neuen Wert für Line_Compare berechnen
          cmp   ax,-1                  ; oberste Zeile erreicht?
          jg    pt1
          mov   Add_word,0             ; JA: Verschiebung zurücksetzen
          mov   ax,-1
pt1:      cmp   ax,max_Lines           ; unterste Zeile erreicht?
          jl    pt2
          mov   Add_word,0             ; JA: Verschiebung zurücksetzen
          mov   ax,max_Lines
pt2:      mov   cx,Snap_Buffer         ; Offset für Start_Address_Register
          push  ax
          cmp   ax,-1                  ; oberste Zeile?
          jg    pt3
          mov   ax,3FFh                ; JA: default für Line_Compare
          mov   cx,0                   ;     Offset für 1. Bildschirmseite
pt3:      mov   Line_Compare,ax
          Split_Screen cx,Line_Compare ; s. Listing der Macros (Bild 2)
          pop   ax
          mov   Line_Compare,ax        ; aktuellen Line_Compare speichern
no_mov:   REG_POP <es,ds,dx,cx,bx,ax>  ; Register wiederherstellen
          popf
          jmp dword ptr cs:[oldint1Ch] ; Sprung in die alte ISR
INT1Ch endp
;--------------------------------------------------------------------------
Install:  push  cs                     ; ds-Segment vorbelegen
          pop   ds
          IRP n,<09h,1Ch>              ; Macro für Interrupts 09h und 1Ch
             mov  ax,35&n
             int  21h                  ; alte ISR-Vektoren retten
             mov  word ptr oldint&n,bx
             mov  word ptr oldint&n+2,es
             mov  ax,25&n              ; neue ISR-Vektoren setzen
             mov  dx,offset Int&n
             int  21h
          endm
          mov   dx,offset InstMSG      ; Meldung ausgeben
          mov   ah,9
          int   21h
          mov   ax,3100h               ; resident beenden
          mov   dx,offset Install
          mov   cl,4
          shr   dx,cl
          inc   dx
          int   21h
InstMSG   db '╔═══════════════════════════════════════════'
          db '═══════════════════════════════════════╗'
          db '║ Split_Screen - SnapShot                 '
          db '                              release 1.0 ║'
          db '║ Copyright (c) I.Eickmann, 1989          '
          db '                                06/14/89 ║'
          db '║ Press <ALT-U> to accelerate up, <ALT-D'
          db '> to acc. down, <ALT-S> for SnapShot    ║'
          db '╚═══════════════════════════════════════════'
          db '═══════════════════════════════════════╝$'
code ends
end  start
```

Bild 3. Das Programm SPLIT.MAC erlaubt eine Schnappschuß-Funktion auf einem Monitor

```
;------------------------------------------------------------------------
;     Split Screen für Turbo Pascal 4.0/5.x                  release 1.0
;     Copyright (c) Ingo Eickmann, 1989                        06/14/89
;------------------------------------------------------------------------
code  segment word public 'CODE'      ; MS-C: Segmentname code in _TEXT ändern
      assume cs:code                  ; MS-C: Segmentname code in _TEXT ändern
if1
      include split.mac
endif

public _Split_Screen
_Split_Screen proc far

   Lines equ [bp+6]                   ; die Parameter liegen in umgekehrter
   Start equ [bp+8]                   ; Reihenfolge auf dem Stack

      push  bp
      mov   bp,sp
      push  es
      Split_Screen Start,Lines        ; Einbindung des Macros
      pop   es
      pop   bp
      ret   4                         ; 2 Integer-Parameter belegten 4 Bytes
                                      ; zusätzlich auf dem Stack
_Split_Screen endp

code ends                             ; MS-C: Segmentname code in _TEXT ändern
end
```

Bild 4. Split_Screen-Macro für Turbo Pascal. Die Änderungen MS-C sind im Kommentar vermerkt

```
(* Unit - Deklaration für Turbo Pascal 4.0/5.x, I. Eickmann 1989 *)

unit split;

{$F+}

interface
  procedure _Split_Screen(Start,Line : Integer);

implementation
  {$L split_tp.obj}
  procedure _Split_Screen; external;

end.
```

Bild 5. So wird das Screen Splitting als Unit für Turbo Pascal aufbereitet