Ingo Eickmann

Zeichensatz verdoppeln

512 verschiedene Zeichen im Textmodus

Statt der üblichen 256 Zeichen können EGA- und VGA-Adapter auch doppelt so viele Zeichen darstellen. Dies ist durch eine raffinierte Doppelbelegung des sogenannten Intensity Bits möglich. Selbstdefinierte Zeichensätze können so problemlos in eigene Programme eingebunden werden.

Der erweiterte ASCII-Zeichensatz setzt dem PC-Benutzer enge Grenzen, wenn es um mathematische Sonderzeichen, das griechische Alphabet, fremdländische Schriften oder generell um nicht alltägliche Zeichen geht. Das Zuschalten eines zweiten Zeichensatzes mit 256 Zeichen schafft hier Abhilfe. *Bild 1* gibt den Eintrag eines angezeigten Zeichens im Bildschirmspeicher wieder. Der Eintrag hat immer Wortgröße: Das erste Byte enthält den Character Code, mit dem sich 256 verschiedene Zeichen ansprechen lassen. Das zweite ist das Attribute Byte und legt die Vorder- und Hintergrundfarbe fest, die Intensität der Vordergrundfarbe (I) und ob das Zeichen blinken soll (Bl). Die Bildschirmseite 0 beginnt im Textmodus bei der Adresse B800:0000h, im Monochrom-Modus 7 bei B000:0000h. Dort steht der Eintrag des Zeichens aus der linken oberen Bildschirmecke. Von links nach rechts und von oben nach unten folgen die übrigen Zeichen, im Textmodus 3 sind es 80×25 = 2000 Worte. Die Character Codes stehen an den geraden Speicherstellen, die Attribute Bytes an den ungeraden.

Grundlegendes

Was die CPU nicht sieht, ist die Aufteilung der sequentiell geschriebenen Worte auf zwei unterschiedliche parallel liegende Speicherbänke. Diese sogenannte Odd/Even-Adressierung erfolgt direkt auf der Grafikkarte und verwaltet alle Character Codes in Map 0, alle Attribute Bytes in Map 1. Darüber hinaus gibt es noch zwei weitere Maps: Map 2 und Map 3, die ebenfalls je 64 KByte groß sind (*Bild 2*). Map 2 ist besonders interessant, hier sind die Bitmasken aller Zeichen abgelegt. Dieser Speicherbereich heißt auch Character Generator RAM und ist in acht Bereiche für Zeichensätze mit 256 Zeichen unterteilt, die sogenannten Definition Tables. Die genaue Einteilung von Map 2 zeigt *Bild 3.* VGA-Karten bieten genügend Speicherplatz für alle acht Tables, EGA-Karten mit 256 KByte verfügen nur über Table 0 bis 3. Alte EGA-Karten mit weniger Speicher engen die Anzahl der Tables weiter ein. Auf einem EGA-Adapter mit 64 KByte steht nur Table 0 zur Verfügung. Ein Table besteht aus 256 Einträgen mit jeweils 32 aufeinanderfolgenden Bytes. Hier sind für die Character Codes 0 bis 0FFh die Bitmasken der Rasterzeilen von oben nach unten abgelegt: Bei EGA-Karten im Textmodus 80×25 Color 14 Byte, bei VGAs 16 Byte. Jedes Byte repräsentiert eine Rasterzeile des Zeichens (*Bild 4*). Die übrigen Bytes bis zum nächsten Eintrag nach insgesamt 32 Bytes bleiben unberücksichtigt (*Bild 5*). Übrigens, die 9 Bit breite Darstellung der VGA wird später durch das Anfügen einer leeren Spalte beziehungsweise durch Verdopplung der 8. Spalte bei den Grafiksymbolen 0C0h bis 0DFh erzeugt. Die interne Adressierung der Map 2 auf der Grafikkarte muß sequentiell erfolgen, nicht nach dem Odd/Even-Prinzip, wie bei Map 0 und 1.

Trickreiches

Wer sich den Inhalt des Videospeichers im Textmodus einmal näher angesehen hat, stellt zurecht die Frage: Wo können weitere Informationen abgelegt werden, die den Zeichensatz bestimmen? Alle Bits sind doch belegt! Dies ist richtig, aber ein Bit hat eine doppelte Bedeutung. Die Auswahl des dargestellten Zeichensatzes aus dem Character Generator RAM erfolgt durch das Character Map Select Register des Sequencers. Den Aufbau dieses Registers enthält Bild 6. Die Bits a0...a2 beziehungsweise b0...b2 geben die Nummer des Definition Table für Zeichensatz A oder B wieder. Die Auswahl, ob A oder B für das jeweilige Zeichen verwendet wird, erfolgt durch das Intensity Bit des Attribute Bytes (Bit 3):

Intensity Bit = 0 bewirkt Zeichensatz A
Intensity Bit = 1 bewirkt Zeichensatz B

Jedes dargestellte Zeichen enthält in seinem Attribute Byte also die Information, ob Zeichensatz A oder B verwendet werden soll, die Intensität der Vordergrundfarbe ist entscheidend. Normalerweise merkt man von der Doppelbelegung dieses Bits - Zeichensatzauswahl und Intensität der Vordergrundfarbe - nichts. Das Video-BIOS setzt bei der Initialisierung sowohl Zeichensatz A als auch B auf Table 0. Dorthin (Offset 0000h in Map 2) wird der übliche Zeichensatz geladen. Das Intensity Bit wählt also erst dann wirklich verschiedene Zeichensätze aus, wenn man im Character Map Select Register verschiedene Tables adressiert.

Erstaunliches

Das Assemblermodul in Bild 7 programmiert die Register des Sequencers und des Grafik-Controllers für das Zuschalten der zweiten 256 Zeichen. Zu Demonstrationszwecken wurden im zusätzlichen Zeichensatz alle bekannten Zeichen um 180 Grad gedreht, der normale Zeichensatz ist jetzt auch kopfstehend verfügbar. Die Bilder und Listings beziehen sich auf die Daten der VGA, alle Änderungen für die Programmierung von EGA-Karten sind vermerkt. Jedem Fan benutzerdefinierter Zeichensätze stehen hier Tür und Tor offen, um mit neuen Bitmasken eigenen Phantasien nachzugehen. Vorsicht ist jedoch bei Programmen geboten, die das Intensity Bit setzen, ohne sich um die neue Bedeutung zu kümmern. Daher sind die neuen Darstellungsmöglichkeiten in erster Linie bei selbstgeschriebenen Programmen sinnvoll anzuwenden. Der Gemeinde der Turbo Pascal Programmierer ist die Unit in Bild 8 gewidmet. Der neue Zeichensatz wird mit dem Kommando ExtCharInit initialisiert und bei der Anwahl heller Vordergrundfarben aktiviert. Diese Zeichen erscheinen jedoch auf dem Bildschirm in normaler Helligkeit. Die Unterdrückung der Intensity-Auswertung erfolgt gegen Ende des Assemblermoduls und kann - je nach Geschmack - auch entfallen. Das Beispielprogramm in Bild 9 stellt zwei Sätze auf dem Bildschirm dar. Der erste Satz erscheint normal, der zweite steht auf dem Kopf. Hier bleibt es jedem selbst überlassen, ob er zum Lesen lieber den Bildschirm dreht oder seinen Kopf.

Literatur

[1] *Cebulla, H.*: So funktioniert die VGA.
Im Sonderheft PC und Hardware, mc-Extra 1/89.
[2] *Eickmann, I.*: Zwei Videokarten im PC.
mc 6/89.
[3] *Smode, D.*: Das große DOS-Profi-Arbeitsbuch.
Franzis-Verlag, München, 1987.
[4] W*ilton, R.*: The Programmer's Guide to PC and PS/2 Video Systems.
Microsoft Press, 1987.

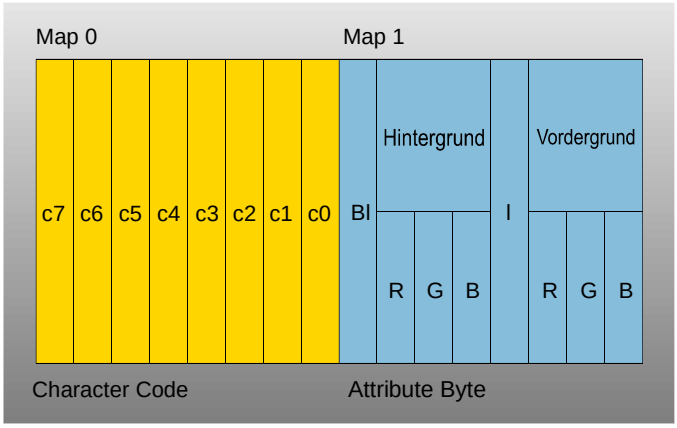

Bild 1. Ein Zeichen wird durch seinen Character Code und sein Attribute Byte im Videospeicher repräsentiert

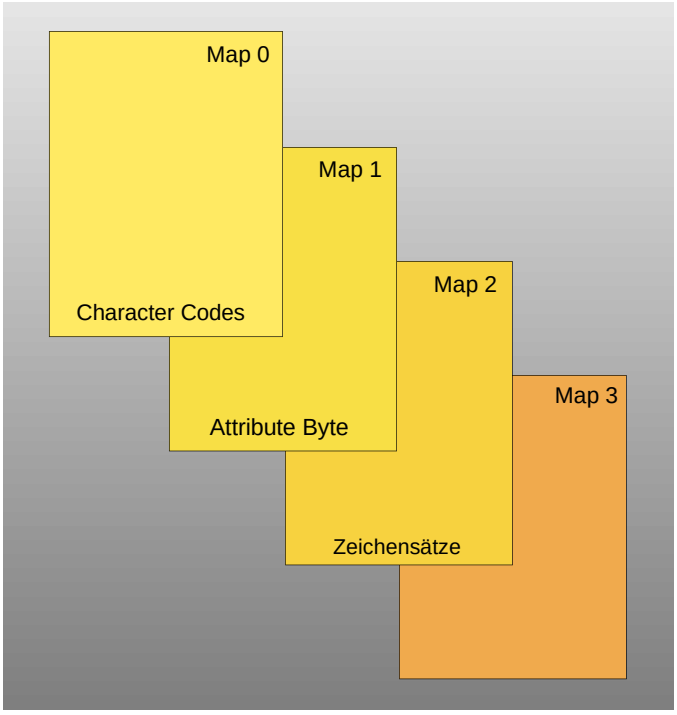

Bild 2. Parallel zum eigentlichen Videospeicher - Map 0 und 1 - liegen weitere Speicherbänke

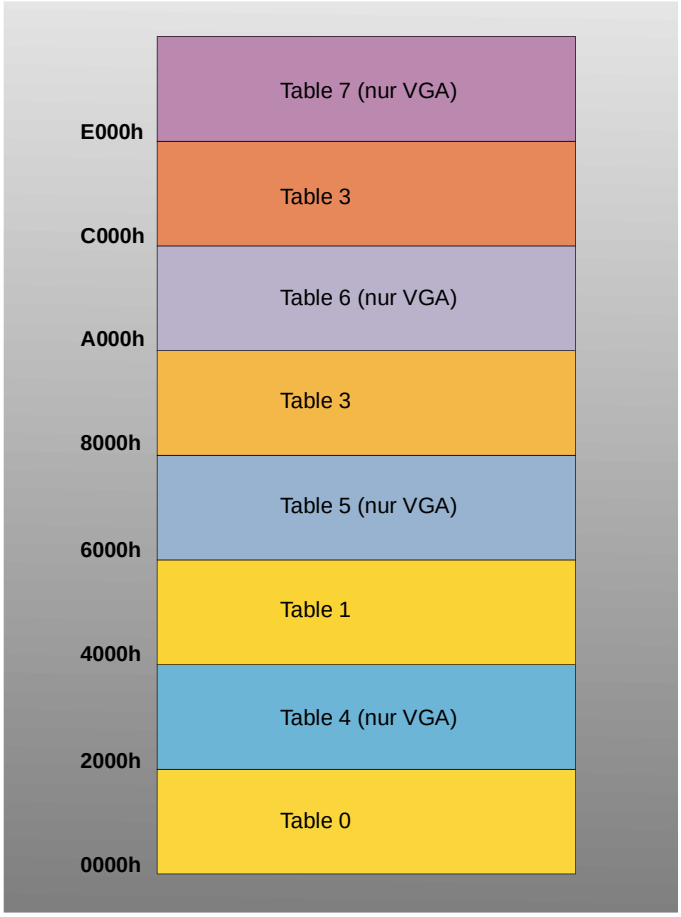

Bild 3. Das Character Generator RAM kann bis zu acht verschiedene Zeichensätze enthalten

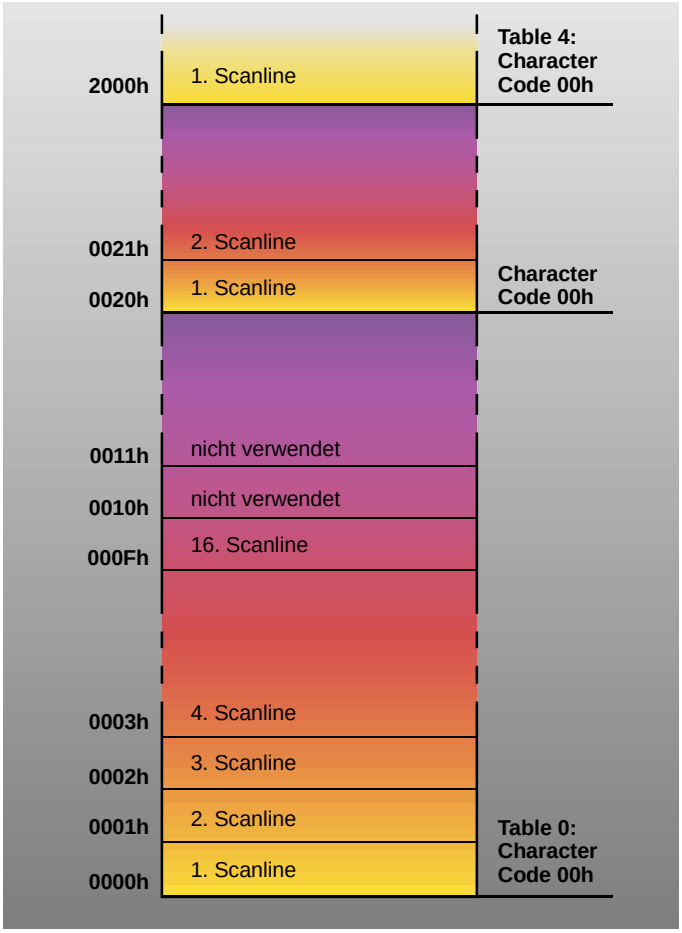

Bild 4. Jeder Definition Table enthält 256 Bitmasken a 32 Byte, von denen bei der VGA 16, bei der EGA 14 belegt sind

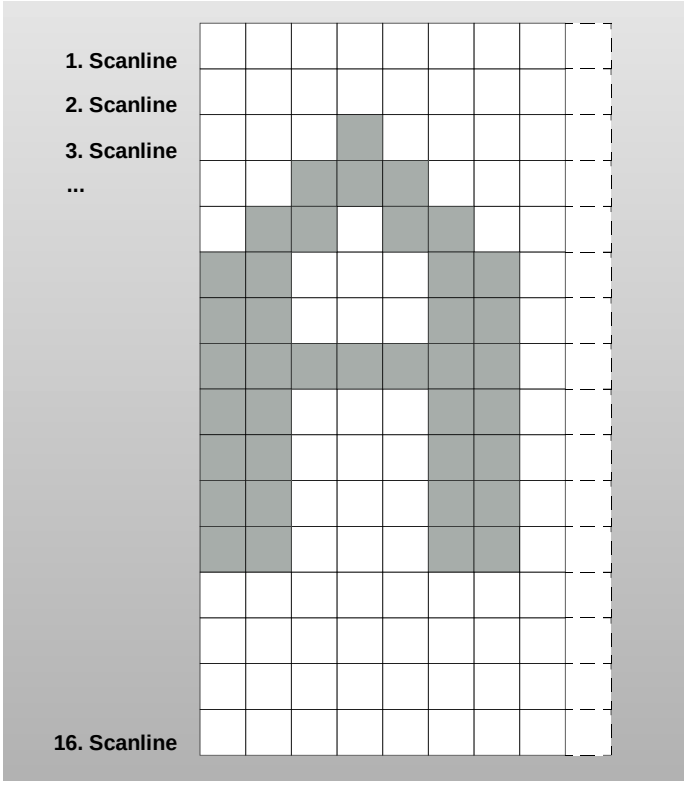

Bild 5. Die Bitmaske eines Zeichens ist 8×16 Bit, bei EGA 8×14 Bit, groß. Die VGA hängt eine 9. Spalte an

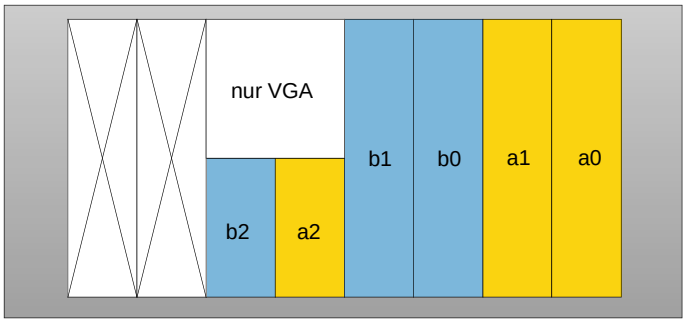

Bild 6. Die Bits des Character Map Registers

```
;-------------------------------------------------------------------------
;     Extended Character Set (VGA)                              release 1.0
;     Copyright (c) Ingo Eickmann, 1989                            07/09/89
;-------------------------------------------------------------------------
;     Assemblieren mit MASM 5.x :                       I. Eickmann
;           MASM Zeichen2;                              Im Leuchterbruch 8
;     Getestet unter MS-DOS 3.3, Turbo Pascal V4.0      5000 Köln 80
;-------------------------------------------------------------------------
      PAGE  65,80
      TITLE Extended Character Set (VGA)

Sequencer  equ 3C4h                   ; Adresse des Sequencer
GraphicsC  equ 3CEh                   ; Adresse des Graphic Controller
Scanlines  equ 16                     ; Scanlines/Char (EGA: 14)

if1
   Port_out macro low,high            ; Index und Daten in Port dx schreiben
      mov al,low
      mov ah,high
      out dx,ax
   endm
endif

code segment word public 'CODE'       ; MS-C: Segmentname code in _TEXT ändern
     assume cs:code

public ExtCharInit                    ; MS-C: Name in _ExtCharInit ändern

ExtCharInit proc far
         push es
         push si
         push di
         cli                          ; Map 2 für sequentiellen Zugriff
         mov dx,Sequencer             ; bereitstellen
         Port_out 0,1                 ; Sync. reset
         Port_out 2,100b              ; CPU beschreibt Map 2
         Port_out 3,4         ; ====>   Map Select Register: Zeichensatz B auf
                                      ; Table 1 setzen
         Port_out 4,7                 ; Sequential Addressing
         Port_out 0,3                 ; Clear Sync. reset
         sti
         mov dx,GraphicsC
         Port_out 4,10b               ; CPU liest Map 2
         Port_out 5,0                 ; Sequential Addressing
         Port_out 6,1100b             ; Miscellaneous Register
;-------------------------------------------------------------------------
         mov ax,0B800h                ; Beispiel: 2. Character Set um 180°
         mov es,ax                    ;                              gedreht
         mov bx,255*32                ;    (256 Characters - 1) * 32 Bytes
Lop2:    mov si,0
         mov di,Scanlines - 1
Lop1:    mov al,es:[si+bx]            ;    Scanline aus 1. CharSet holen
         mov cx,8                     ;    8 Bits/Byte spiegeln
Lop3:    shr al,1
         rcl ah,1
         loop Lop3
         mov es:[di+bx+4000h],ah      ;    Scanline in 2. CharSet ablegen
         inc si
         dec di
         jnl Lop1                     ;    nächste Scanline
         sub bx,32
         jnl Lop2                     ;    nächster Character
```

```
;-------------------------------------------------------------------------
        cli                             ; Anwahl der Maps 0 und 1 im Odd/Even-
        mov dx,Sequencer                ; Mode wiederherstellen
        Port_out 0,1                    ; Sync. reset
        Port_out 2,011b                 ; CPU beschreibt Map 0 and 1
        Port_out 4,11b                  ; Odd/Even Addressing
        Port_out 0,3                    ; Clear Sync. reset
        sti
        mov dx,GraphicsC
        Port_out 4,0                    ; CPU liest Map 0 und 1
        Port_out 5,10h                  ; Odd/Even Addressing
        Port_out 6,1110b                ; Miscellaneous Register

        mov ax,1000h                    ; Unterdrückung des Intensity-Bits
        mov bx,0712h                    ; durch Color Plane Enable Register,
        int 10h                         ; kann entfallen

        pop di
        pop si
        pop es
        ret
ExtCharInit endp

code ends
     end
```

Bild 7. Das Assemblermodul schaltet den zusätzlichen Zeichensatz ein

```
unit Zeichen2;

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

{$F+}

interface
  procedure ExtCharInit;

implementation
  {$L Zeichen2.obj}
  procedure ExtCharInit; external;
end.
```

Bild 8. Diese Unit bindet das Assemblermodul in Turbo Pascal ein

```
program ExtCharDemo;

(* Demo des zusätzlichen Zeichensatzes, I.Eickmann 1989 *)

uses Crt,Zeichen2;

var  i : Integer;

Begin
  TextMode(C80);
  ExtCharInit;

(* Character Set 1: Normaler Zeichensatz (256 Chars.). *)

  TextColor(LightGray);
  writeLn('Character Set 1: Normaler Zeichensatz (256 Chars.).');
  writeLn('');

(* Character Set 2: Alle Zeichen des CharSet 1 um 180° gedreht. *)

  TextColor(white);
  writeLn('!therdeg °081 mu 1 teSrahC sed nehcieZ ellA :2 teS retcarahC');
  writeLn('');

  TextColor(LightGray);
End.
```

Bild 9. Das Demoprogramm hat Zugriff auf 512 gleichzeitig darstellbare Zeichen