Jörg Matthaei

Speicherengpaß entschärft

64 KByte mehr Speicher für den AT

AT-Anwender verschenken kostbaren Arbeitsspeicher; denn über einen Treiber kann die 80286-CPU unter DOS zusätzliche 64 KByte adressieren. Von Microsoft wurde dieser Speicher als high memory area (HMA) spezifiziert. Auch selbstgeschriebene Programme können von HMA profitieren.

Die Prozessoren 8086 und 8088 von Intel können über ihre 20 Adreßleitungen bis zu 1 MByte Arbeitsspeicher adressieren. Dagegen kann eine 80286-CPU auf einen wesentlich größeren Speicher zugreifen: 16 MByte - allerdings nicht unter DOS. Hier bleibt es immer noch bei dem einen Megabyte (*Bild 1*). Aber mit einem Trick adressiert die 80286-CPU auch unter DOS mehr als 1 MByte RAM.

Die letzte Adresse, auf die eine 8086-CPU oder 8088 zugreifen kann, ist FFFFFh. In der Zwei-Komponentenform, bestehend aus Segment und Offset, heißt genau dieselbe Adresse FFFF:000Fh. Würde der Adreßzeiger um eins erhöht, kommt es zu einem Überlauf auf Adresse 0000:0000h, den Anfang des Arbeitsspeichers. Dieser Überlauf wird auch als address wrap around bezeichnet. Wenn man bei einem 80286-Prozessor die 21. Adreßleitung (A20) aktivieren würde, käme es nicht zu einem Überlauf und der Speicher von FFFF:0010h bis FFFF:FFFFh könnte für Daten und Programme von DOS genutzt werden. Durch diesen kleinen Trick erhält man einen zusätzlichen Speicherbereich - nur 16 Byte kleiner als 64 KByte - der als HMA (high memory area) bezeichnet wird. 1988 hat Microsoft die Extended-Memory-Spezifikation (XMS) entwickelt, die genau festlegt, wie Programmierer extended memory und HMA nutzen sollen. Der XMS-Treiber von Microsoft HIMEM.SYS wird mit der Version 4.0 von DOS oder mit MS-Windows geliefert. Wenn man den HMA-Treiber mit der Anweisung DEVICE = HIMEM.SYS in die Datei CONFIG.SYS eingebunden hat, kann man diesen Speicher über bestimmte Funktionen anfordern oder freigeben.

Das Programm HMA.ASM (*Bild 2*) gibt einige Beispiele, wie man HMA-Funktionen anwendet. Zuerst prüft es, ob bereits ein XMS-Treiber installiert ist. Anschließend holt es sich die Einsprungadresse für die XMS-Funktionen. Danach wird der HMA-Treiber aufgerufen, wobei im Register DX die Anzahl der anzufordernden Bytes stehen muß. Zum Schluß aktiviert das Programm die Adreßleitung A20. Nun kann man den zusätzlichen Speicherbereich verwenden. Es ist sinnvoll, möglichst viel Programmcode und Daten in diesem Bereich unterzubringen, um die unteren 640 KByte zu entlasten. Wie wird ein schon vorhandenes speicherresidentes Programm in den HMA-Bereich gelegt? Ganz einfach - der Programmcode, die Daten und die Installationsroutine bleiben bis auf die Funktion terminate and stay resident (TSR) unverändert (*Bild 3*). Der Interrupt 27h oder die DOS-Funktion 31h wird durch die Programmende-Funktion (DOS-Funktion 4Ch) ersetzt. Sodann wird die HMA-Laderoutine (*Bild 4*) an den bestehenden Quellcode angefügt und vor dem ersten Befehl ein direkter Sprung auf die HMA-Laderoutine eingefügt. Die HMA-Laderoutine, im Programm mit HMA_LOADER bezeichnet, fordert zuerst 64 KByte HMA an und verschiebt das gesamte Programm in diesen Speicherbereich. Anschließend wird das Programm ab dem ursprünglichen Beginn der alten speicherresidenten Routine ausgeführt, und die Interruptvektoren werden verbogen. Sie zeigen jetzt auf das Programm im hohen Speicherbereich.

Übersicht der HMA-Funktionen

AH-Register	HMA-Funktionen
00h	XMS-Versionsnummer
01h	HMA anfordern
02h	HMA freigeben
03h	A20 global ermöglichen
04h	A20 global sperren
05h	A20 lokal ermöglichen
06h	A20 lokal sperren
07h	A20 abfragen

Tips für den Umgang mit HMA

In Programmen, die im hohen Speicherbereich laufen, müssen alle Segmentregister auf das gleiche Segment zeigen. Bei Programmen, die mit dem Speicherüberlauf bei 1 MByte arbeiten, muß vor Verlassen der speicherresidenten Routine die Adreßleitung A20 deaktiviert werden. Programme, die den unteren Bereich des extended memory wie alte VDISK-Versionen benutzen, dürfen nicht gleichzeitig verwendet werden. Wer keinen AT mit extended memory besitzt, kann auch den RAM-Bereich einer EGA oder VGA im Textmodus nutzen. Doch dann darf kein Programm mehr die Videoschnittstelle in den Grafikmodus umschalten. Beim HMA-Lader muß die Adresse, zu der die speicherresidente Routine verschoben werden soll, geändert werden. Bei EGA und VGA ist das Adresse B900h.

Literatur

[1] *Chip, Anderson:* 64 KByte mehr Speicher adressieren unter DOS, MS-System-Journal 1989, März/April.
[2] NCR-DOS Programmers Guide 1985.

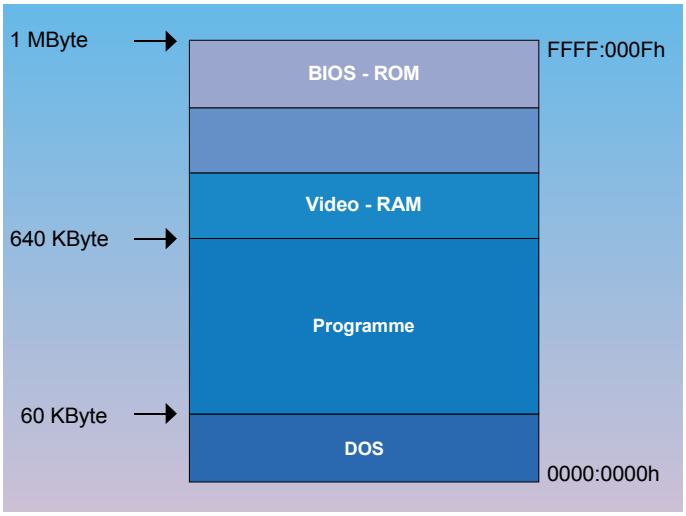

Bild 1. Speicherbelegung eines AT

```
;-------------------------------------------------
;  HMA.ASM
;
;  Programm zur Veranschaulichung der HMA-Funktionen
;  des Microsoft Himem.sys Drivers.
;
;  Datum: 06.06.89
;  Autor: Jörg Matthaei
;-------------------------------------------------
;  Macros
display macro   string
        lea     dx,string
        mov     ah,09
        int     21h
endm

cseg    segment 'code'
        assume cs:cseg,ds:cseg,es:cseg

        org     80h
```

```
parms   equ     this byte

        org     100h

begin:
        mov     ax,4300h
        int     2Fh                     ; ist ein HMA Treiber instaliert ?
        cmp     al,80h
        jz      hmada

        display msgnohma                ; nein !
fin:    mov     ah,4Ch                  ; Programmende
        int     21h
                                        ; ja hole Einsprungadresse für HMA-
hmada:  push    es                      ; Funktionen.
        mov     ax,4310h
        int     2Fh
        mov     word ptr[hmaofs],bx     ; speichere Einsprungadressen
        mov     word ptr[hmaseg],es
        pop     es

        cmp     byte ptr [parms],0      ; prüfe auf Parameter
        jne     anf0
        display msgsyntx                ; Anzeige von gültigen Parametern
        jmp     fin

anf0:   cmp     byte ptr [parms+3],'a'  ; verzweige je nach Parametern
        je      alloc
        cmp     byte ptr [parms+3],'f'
        je      free
        cmp     byte ptr [parms+3],'s'
        je      status
        jmp     fin

; Die Funktion alloc reserviert 64KB HMA

alloc:  mov     dx,0FFFFh               ; reserviere FFFF Bytes HMA
        mov     ah,01
        call    dword ptr hmaofs
        dec     ax
        jz      enab_himem              ; HMA reseviert
        display msgused                 ; Reservierung nicht möglich !
        jmp     fin
enab_himem:
        mov     ah,03                   ; A20 global ermöglichen
        call    dword ptr hmaofs
        display msgresv
        jmp     fin
```

```
; Die Funktion free gibt den mit alloc reservierten
; HMA Bereich wieder frei

free:   mov     ah,04                   ; A20 global sperren
        call    dword ptr hmaofs
        mov     ah,02                   ; HMA freigeben
        call    dword ptr hmaofs
        display msgfree
        jmp     fin

; Die Funktion status versucht HMA anzufordern,
; gelingt dies gibt sie die Meldung HMA wird nicht benutzt aus.
; Ist eine Reservierung nicht möglich, so ist der HMA Bereich
; bereits von einem anderen Programm benutzt.

status:
        mov     dx,0FFFFh               ; versuche FFFFh Bytes
        mov     ah,01                   ; HMA anzufordern
        call    dword ptr hmaofs
        dec     ax
        jz      stat1

        display msgused                 ; HMA bereits genutzt
        jmp     fin

stat1:  mov     ah,02                   ; von Funktion status reservierte
        call    dword ptr hmaofs        ; HMA wieder freigeben
        display msgnouse
        jmp     fin

hmaofs    dw   0                        ; HMA-DRIVER OFFSET
hmaseg    dw   0                        ; HMA-DRIVER SEGMENT

msgnohma  db   13,10,'Himem Driver ist nicht installiert',13,10,'$'
msgnouse  db   13,10,'HMA wird nicht benutzt',13,10,'$'
msgresv   db   13,10,'64KB HMA sind reserviert.',13,10,'$'
msgused   db   13,10,'HMA wird schon benutzt',13,10,'$'
msgfree   db   13,10,'HMA wieder freigegeben.',13,10,'$'
msgsyntx  db   13,10,'Die Syntax für HMA lautet :',13,10
          db   'hma /a  für anfordern von 64KB HMA',13,10
          db   'hma /f  für freigeben von belegter HMA',13,10
          db   'hma /s  für Status HMA',13,10,'$'

cseg   ends
       end    begin
```

Bild 2. Demoprogramm für den Treiber Himem.sys von Microsoft

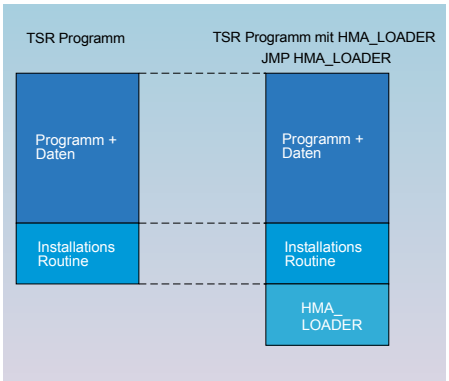

Bild 3. Speicherresidente Routinen in HMA

```
;-------------------------------------------------
; Dieses Programm wird an Ihre bestehende
; TSR-Routine angefügt und mit einem JMP-Befehl
; an erster Stelle in Ihrer TSR-Routine aktiviert.
;
; Datum : 03.06.89
; Autor : Jörg Matthaei
;-------------------------------------------------

himem   proc    near

begin:   call   hmada                   ; Teste ob Himem-Driver aktiv ist
         jnz    fin
         mov    cx,pgende-pganf         ; pgende ist die End-Adresse Ihrer
         mov    dx,-1                   ; TSR-Routine bzw.
         mov    ah,01                   ; pganf ist die Start-Adresse
                                        ; cx enthält somit die Anzahl der
                                        ; Bytes die in die HMA zu kopieren
                                        ; sind

         call   dword ptr hmaofs        ; reserviere HMA
         dec    ax
         jz     him1
         ret                            ; Reservierung nicht möglich
him1:    mov    ah,03
         call   dword ptr hmaofs        ; A20 global ermöglichen
         dec    ax
         jnz    him2
                                        ; kopiere PSP ab 10h in
         push   cx                      ; den HMA Bereich
         mov    ax,0FFFFh
         mov    es,ax
         mov    di,10h
         mov    si,10h
         mov    cx,0F0h
         rep    movsb
```

```
        pop    cx
        mov    di,offset pganf        ; verschiebe TSR Routine
        mov    si,di                  ; in den HMA Bereich
        rep    movsb
        cli
        mov    ds,ax                  ; ds=0FFFFh
        db     0EAh
        dw     103h,0FFFFh            ; JMP FAR auf TSR Routine
        sti                           ; die sich nun selbst im
        ret                           ; HMA Bereich installiert !

him2:   mov    ah,04                  ; A20 sperren
        call   dword ptr hmaofs
        mov    ah,02                  ; HMA Bereich wieder freigeben
        call   dword ptr hmaofs
        ret

hmada:  mov    ax,4300H               ; Teste ob Himem.sys aktiv ist
        int    2Fh
        cmp    al,80H
        jz     alloc
        mov    dx,offset msgins       ; Himem.sys nicht aktiv
        mov    ah,09
        int    21h
        ret

alloc:  push   es
        mov    ax,4310H               ; hole HMA Einsprungadresse
        int    2Fh
        mov    hmaofs,bx
        mov    hmaseg,es
        pop    es
        and    al,0
        ret

fin:    mov    ah,4Ch
        int    21h

msgins  db     10,13,9,'XMS - Driver nicht installiert !',10,13,'$'

hmaofs  dw     0                      ; HMA-Driver Offset
hmaseg  dw     0                      ; HMA-Driver Segment

himem   endp
```

Bild 4. HMA-Routine für speicherresidente Programme