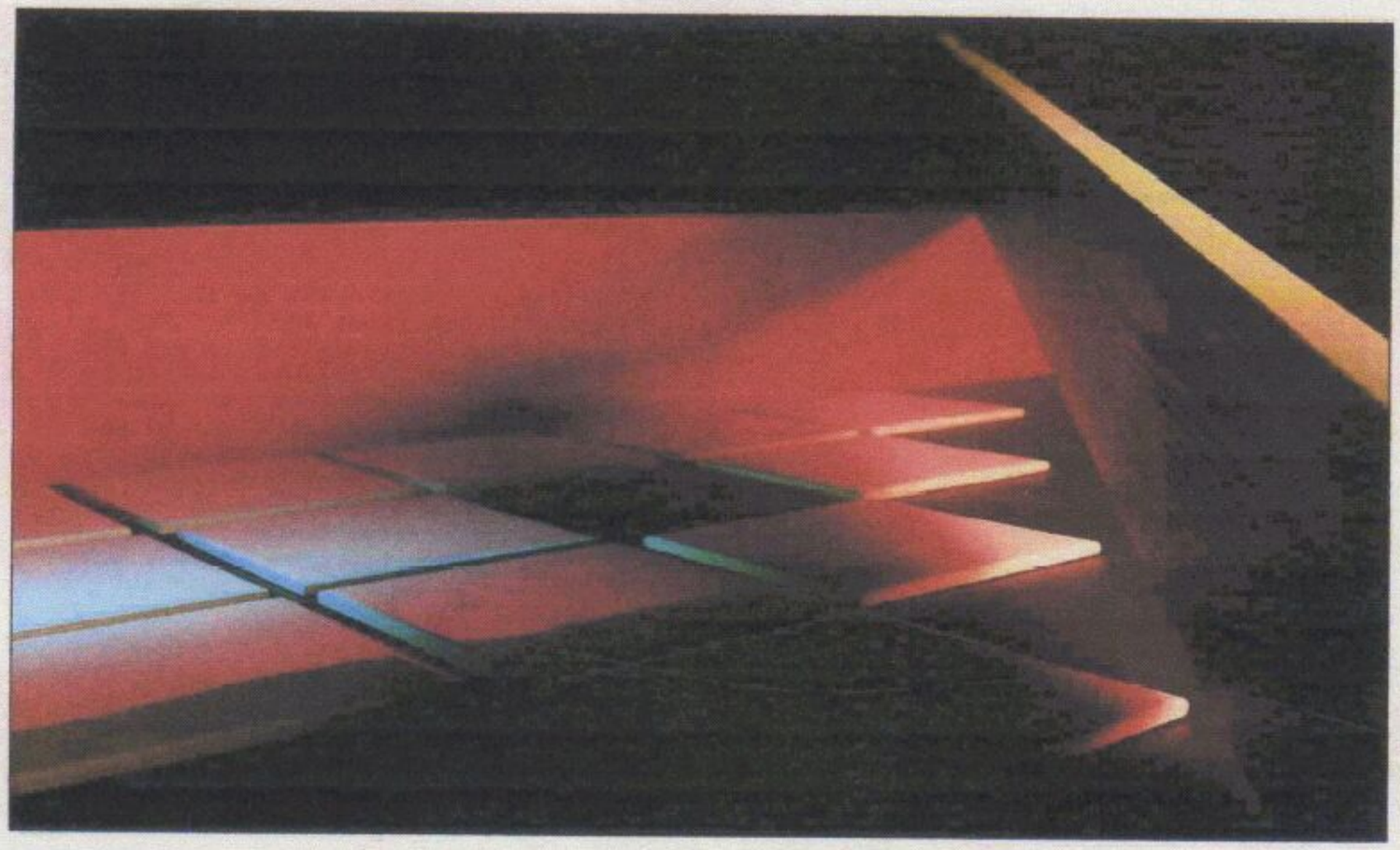


Byte-Hirte

RAM-Disk-Treiber für des CPCs 512 KByte

Gabor Herr
Hubert Schröer

Hat Ihr CPC die Aufrüstung zum CPC 6512 heil überstanden, sind Sie sicherlich schon gespannt, was man mit den 512 KB außer ein Paar bewegten Bilderchen sonst noch so anstellen kann.

Es ist leider nicht möglich, mit dem zusätzlichen Speicher den Arbeitsspeicher zu erweitern. CP/M Plus macht dem Benutzer nämlich nur Bank 1 zugänglich, wodurch die Größe der TPA beim CPC 6128 auf 61 KB begrenzt bleibt. Also müssen die zusätzlichen 384 KB für andere Aufgaben genutzt werden.

Einen Teil des Speichers kann man dem BDOS als Sektorenpuffer für die Floppy-Laufwerke zur Verfügung stellen, damit die lahmen Dreizöller mal richtig auf Trab gebracht werden. Der restliche, größere Teil steht für eine resetfeste RAM-Disk zur Verfügung, die die eingebaute Floppy dann endgültig in den Schatten stellt, was Kapazität und Geschwindigkeit anbelangt.

Bei der Entwicklung des neuen Banking-PALs wurde insbesondere darauf geachtet, daß die Selektierung der Speichererweiterung auch dem Banking-Konzept von CP/M Plus entspricht. Betrachtet man die Selektierungstabelle (c't 10/87, Seite 157) etwas genauer, erkennt man, wie die Speicherblöcke mit den Nummern 8–31 im System eingebunden sind: sie werden in die Systembank eingeblendet, und zwar auf den Adreßbereich des Bildschirmspeichers (4000h–7FFFh). Somit bilden sie die CP/M-Speicherbänke 7–30 mit der nutzbaren Größe von jeweils 16 KB (Tabelle 1).

Manchen Lesern wird diese Methode sicherlich bekannt vorkommen: Bank 2, in der der CCP und der LOADER abgelegt sind, wird auf die gleiche Weise angesprochen. Diese Adressierungsart hat außerdem noch folgende Vorteile:

– Das BIOS enthält alle nötigen Routinen für die Verwaltung von solchen Bänken (Bankselektierung, bankübergreifendes Kopieren).
– Die in diesem Bereich abgelegten Programme können ohne Bankumschaltung auf alle Teile des Systems zugreifen (z. B. Aufruf von ROM-Routinen über die Kernal-Vektoren).

Diskettenpuffer

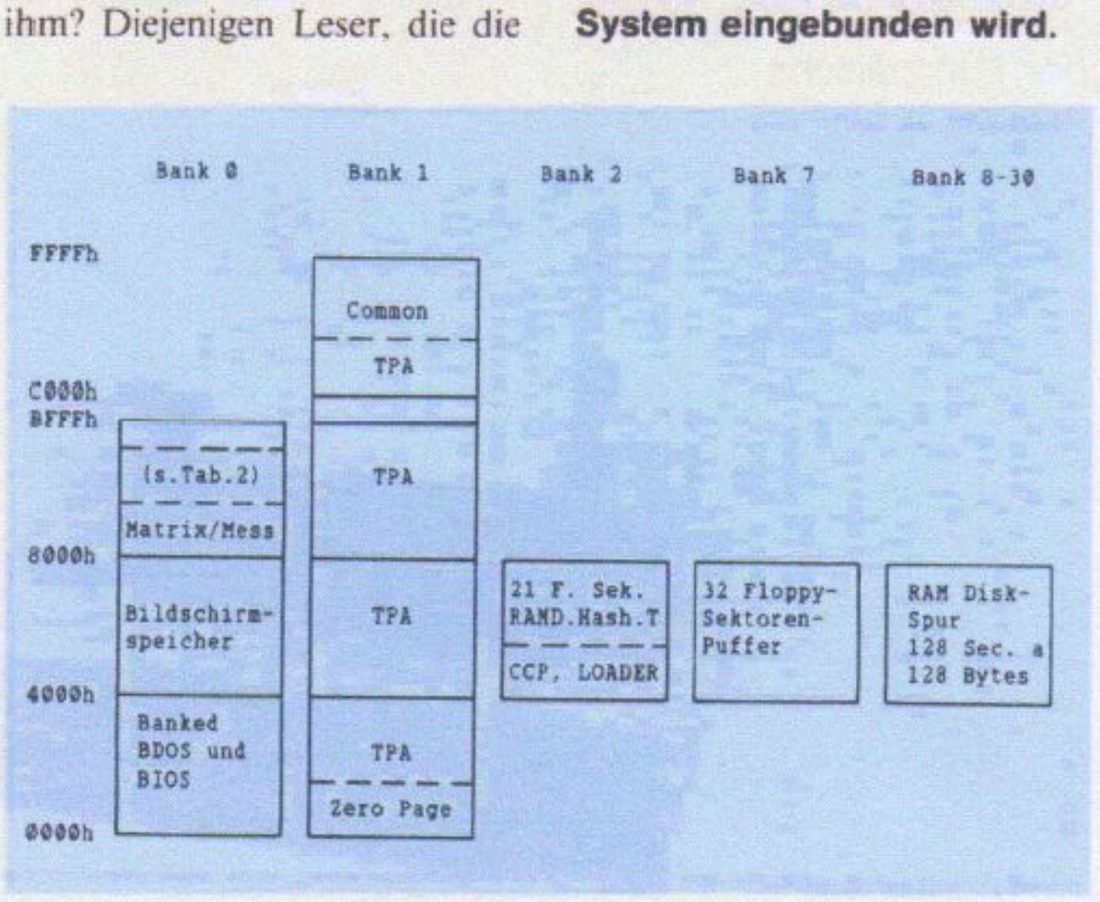
Eine RAM-Disk braucht einen Treiber, und für den wiederum muß sich ein sicheres Plätzchen im RAM finden. Wohin mit ihm? Diejenigen Leser, die die Artikel über 80-Track-Laufwerke am Schneider (c't 5, 6 und 9/87) verfolgt haben, wissen bestimmt eine Antwort. Hier wird die Einquartierung auf ähnliche Weise gelöst.

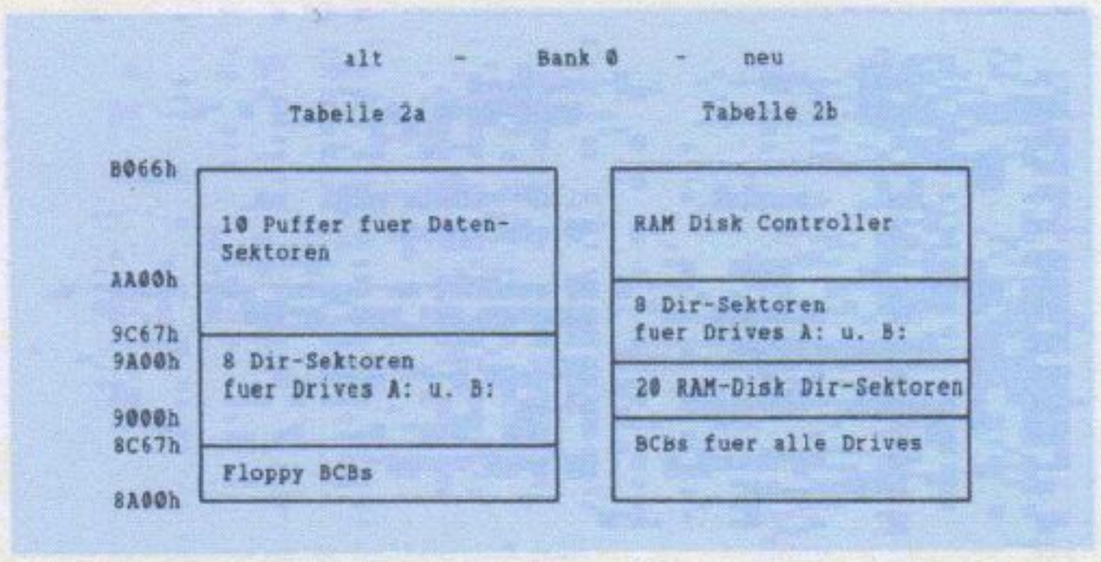
Im Originalsystem sind die Diskettenpuffer wie folgt verteilt: 8 Sektoren für das Inhaltsverzeichnis und 10 Sektoren für Daten in Bank 0 (Tabelle 2a) sowie 23 Sektoren Datenpuffer in Bank 2. Die dazugehörigen BCBs (Buffer Control Blocks) sind ebenfalls in Bank 0 ab 8A00h untergebracht.

Man kann die Sektorenpuffer aus der Systembank in eine andere Bank auslagern, um dadurch dem RAM-Disk-Treiber Platz zu schaffen. Da man mit dem Speicher jetzt etwas großzügiger umgehen kann, nimmt man gleich eine ganze Bank. In der neu hinzugekommenen Bank 7 lassen sich statt bisheriger 10 jetzt 32 Sektoren puffern. Zusammen mit den 21 Sektoren in Bank 2 ergibt das eine Pufferkapazität von knapp 27 KB. Zwei Sektoren oder 1K mußten in Bank 2 der Hashing-Table weichen.

Beim Versuch, auch die Directory-Puffer auszulagern, weigerte sich das BDOS, Diskettenoperationen durchzuführen. Tabelle 2b zeigt die neue Aufteilung in der Systembank mit dem RAM-Disk-Treiber und dem dazugehörigen Directory-Puffer.

Tabelle 1 zeigt noch einmal anschaulich, wie das zusätzliche RAM ins System eingebunden wird.





Der Block für die Sektor-Puffer in Bank 0 wurde umorganisiert; hier residiert jetzt auch der RAM-Disk-Treiber.

Wer sich mit dieser Pufferkapazität noch nicht zufriedengeben kann, hat die Möglichkeit, weitere Bänke für die Pufferung heranzuziehen. Dazu sind lediglich die Variablen am Anfang des Patch-Programms zu ändern. Dabei bezeichnet fddbnkx die Speicherbank, fddatbx die Anfangsadresse und fddatsx die Anzahl der gewünschten Sektoren. Bei der Planung der Sektorenpuffer sollte man beachten, daß pro Sektor ein BCB (Länge: 15 Bytes) in der Systembank benötigt wird.

Dazu ein Beispiel: In der nächsten Bank sollen weitere 32 Sektoren zwischengespeichert werden. Dazu müßte man folgende Werte angeben:

```
fddbnk3  = 8
fddatb3  = 4000h
fddats3  = 32
```

Ließe man nun das so gepatchte CP/M booten, käme man in Teufels Küche. Die BCBs der neuen Puffer müßten ihren Platz mit den Directory-Puffern teilen; sie würden mit ihrem Streit um die Plätze viel Unheil auf der Diskette anrichten. Um den Inhalt der Disketten nicht zu gefährden, muß der über den BCBs liegende Puffer entsprechend verkleinert werden. Die Berechnung für die Variablenwerte im Listing sieht folgendermaßen aus:

```
Anzahl der BCBs:
    16  rddirs
+    8  fddirs
+   23  fddats1
+   32  fddats2
+   32  fddats3
-------------------
=  111
```

Ende des BCB-Bereiches:
8A00h + 110 * 15 = 9072h

Als Anfangsadresse über dem BCB-Bereich ergibt sich für rddirb 9080h, und für rddirs nur noch 15 statt 16 Sektoren.

Bei der Verwendung von Laufwerken mit größeren Sektoren kann mit der Variablen fdsecsz die Sektorlänge eingestellt werden.

Floppy – schnell und leise

Große Speichermengen werden in ein Betriebssystem am einfachsten als RAM-Floppy eingebunden, da die vorhandenen Disk-Routinen den größten Teil der Verwaltungsarbeit abnehmen.

Als erstes muß man sich über das 'Diskettenformat' Gedanken machen. Der verfügbare Speicher wird, ähnlich wie bei einer Diskette, in Spuren und Sektoren aufgeteilt. Hierbei zeigt die verwendete Speicherorganisation ihre Vorteile: jede der verfügbaren Bänke hat die gleiche Größe und ist durchnumeriert. Es ist also zweckmäßig, diese als Spuren der RAM-Floppy zu definieren.

Dadurch läßt sich auch die Kapazität, dem Speicherausbau und der Belegung der Bänke entsprechend, problemlos variieren. Wie viele Bänke die RAM-Disk benutzen soll, hängt von den beiden Variablen TRACK0 und TRACKS ab. Erstere enthält die Nummer der Bank, die als Spur 0 dienen soll, letztere gibt die Anzahl der Spuren und somit die Größe dieses Mediums an. Beim vollem Speicherausbau mit Bank 8 als Spur 0 und mit 23 Spuren beträgt die Kapazität:

23 * 16 KB = 368 KB.

Anders ausgedrückt, kann man Turbo, WS, M80 und L80 und und und . . . gleichzeitig im Speicher haben und innerhalb von Sekunden lautlos, quasi wie ein nicht-transientes Programm, starten.

Die Spuren teilt man am besten in Sektoren zu 128 Byte auf, weil CP/M intern auch mit dieser Sektorlänge arbeitet. Dadurch erspart man sich das zeitaufwendige Deblocking.

Damit das BIOS auf diese Sektoren auch zugreifen kann, werden noch zwei Treiberroutinen benötigt: READ zum Lesen und WRITE zum Schreiben eines Sektors. Als Parameter stehen in den Systemvariablen TRK, SEC, DMA, DMABNK die Angaben über Spur-, Sektornummer, DMA-Adresse und DMA-Bank zur Verfügung.

Die READ-Routine hat die Aufgabe, einen mit TRK und SEC bezeichneten Sektor in die DMA-Bank an die DMA-Adresse zu kopieren. Dazu muß sie lediglich Quell- und Zieladresse berechnen und an die BIOS-Funktionen XMOVE und MOVE übergeben. Diese erledigen dann die schwierigste Aufgabe, das bankübergreifende Kopieren. WRITE unterscheidet sich von READ nur darin, daß Quell- und Zieladresse vor dem Kopieren vertauscht werden.

Die Adressen dieser Treiberroutinen müssen in den gleichnamigen Vektoren am Anfang des XDPH eingetragen sein.

RAM-Disks müssen nach einem Kaltstart initialisiert werden, das heißt, das Inhaltsverzeichnis ist zu löschen. Speziell für diese Aufgabe gibt es einen XDPH-Vektor mit dem Namen INIT, der bei jedem Kaltstart aufgerufen wird. Nach einigen Tests stellte sich aber heraus, daß die Amstrad-Entwickler diesen Vektor anscheinend vergessen haben.

Doch mit einem kleinen Trick läßt sich dieses Problem lösen. Die INIT-Routine wird über den LOGIN-Vektor bei jedem Zugriff auf das Laufwerk aufgerufen. Ein Flag zeigt an, ob es sich um den ersten Zugriff nach einem Kaltstart handelt.

Um die RAM-Floppy auch resetfest zu machen, wird bei der Initialisierung zuerst überprüft, ob das Inhaltsverzeichnis etwas enthält. Dazu untersucht die Routine, ob die Bytes 1 bis 8 des ersten Eintrages einen Dateinamen enthalten. Ist dies der Fall, tastet sie die RAM-Disk nicht an.

DPBchen, wechsel' dich

Ein weiteres Problem ist, daß das BDOS den DPB unbedingt im Common-Bereich haben will. Es soll seinen Willen haben. Dort oben sind schon zwei Plätze reserviert für die DPBs der Drives A und B. Der Platz für B wird nun kurzerhand für alle in Frage kommenden Drives (außer A) benutzt. Die entsprechenden DPBs liegen hinter dem RAM-Disk-Treiber in Bank 0 und werden je bei Bedarf nach oben kopiert.

Um der RAM-Floppy einen schnellen Zugriff auf die Datensektoren zu ermöglichen, erhielt sie eine Hashing-Tabelle. Außerdem ist das Inhaltsverzeichnis gepuffert.

Als letzten Schritt der Einbindung trägt das Patch-Programm die Adresse des DPH in die Drive-Table ein. Mit welcher Drive-Kennung die RAM-Disk angesprochen werden soll, bestimmt die Variable rddrv. Zulässig sind die Werte C bis P.

Um den Treiber mit den unterschiedlichsten Konfigurationen verträglich zu machen, ist auch dieses Programm möglichst variabel gestaltet. Ob Sie eine oder zwei externe Floppies, ob 80-Track-Drives oder doppelseitige einbinden wollen, das steuern die ersten beiden Variablen im Listing. Zusätzlich müssen Sie die jeweils benötigten DPBs für B und/oder C am Ende eintragen.

Will man das CP/M Plus seines Rechners ändern oder erweitern, bieten sich zwei Alternativen: man kann das System nach jedem Kaltstart mit Hilfe eines Programms im RAM patchen oder die Änderungen direkt an der Systemdatei vornehmen. Wir haben uns mit dem hier abgedruckten Patch-Programm für die zweite Lösung entschieden.

Als Parameter beim Aufruf von RAMDISK wird der Dateiname des System-Files angegeben, das man patchen möchte, in der Regel A:C10CPM3.EMS. Nach dem Booten des gepatchten CP/M Plus erscheinen in der Einschaltmeldung dann die Angaben für Laufwerksbezeichnung und Kapazität der RAM-Floppy. Zum Schluß noch einmal den bekannten Spruch: Bitte nicht im Original herumpatchen! (bb)

```
;       Version     2.0 / gh87

;Dieses Programm "patcht" die CP/M Plus Systemdatei

;Fuer Besitzer von drei Laufwerken, oder 80 Track Drives wurden
;die BIOS-Patches aus dem Artikel von Holger Merk "CPC ruft Lauf-
;werk C:" (c't 6/87) uebernommen. Im Listing sind diese Aende-
;rungen mit einem "*" markiert.

;Fuer die einzelnen Laufwerke muessen die DPBs, am Ende des
;Listings angepasst werden: dpb_b fuer Drive B:, dpb_fd3 fuer's
;dritte Laufwerk.

;Floppy Parameter
;-----------------------------------------------------------------
bigdrives equ   1           ;80 Track Laufwerke B: (und 3. Laufw.)
floppy3   equ   1           ;3 Floppies am CPC

;nur wichtig, wenn bigdrives <> 0:
fd3drv    equ   'C'         ;3. Floppy log. Bezeichnung
Step_B    equ   12          ;Steprate B:
Step_C    equ   12          ;Steprate vom 3. Laufwerk (hier C:)

;RAM Disk Parameter
;-----------------------------------------------------------------
rddrv     equ   'D'         ;logische Drive Bezeichnung
tracks    equ   23          ;Anzahl der Spuren (Baenke),
                            ;Kapazitaet=tracks*16 KB
track0    equ   8           ;CP/M Bank Nr. des 1. Tracks
sector0   equ   4000h       ;Adresse des 1. Sektors
dirblks   equ   2           ;Anzahl der Directory-Bloecke
al0       equ   11000000b   ;Position der Dir-Bloecke auf der Disk
al1       equ   00000000b

rddirb    equ   9000h       ;RAM Disk Directory-Puffer Adresse
rddirs    equ   20          ;und Laenge in Sektoren

;Puffer Parameter
;-----------------------------------------------------------------
fdsecsz   equ   512         ;Floppy Sektorengroesse
fddirb    equ   9a00h       ;Floppy Directory-Puffer Adresse
fddirs    equ   8           ;und Laenge

fddbnk1   equ   2           ;Floppy Datenpuffer 1 Bank
fddatb1   equ   5480h       ;Adresse
fddats1   equ   21          ;und Laenge

fddbnk2   equ   7           ;Floppy Datenpuffer 2 Bank
fddatb2   equ   4000h       ;Adresse
fddats2   equ   32          ;und Laenge

fddbnk3   equ   0           ;Floppy Datenpuffer 3 Bank
fddatb3   equ   0000h       ;Adresse
fddats3   equ   00          ;und Laenge

;BIOS Variablen/ Vektoren:
;-----------------------------------------------------------------
bios      equ   0FC00h      ;Adresse der BIOS Sprungleiste
selbnk    equ   bios+3*27   ;F 27: Bank selektieren
xmove     equ   bios+3*29   ;F 29: extended Move
bmove     equ   0FCCBh      ;F 25: Move (nicht ueber SySTEM Call)
system    equ   0FD1Ch      ;Amstrad Call System Vektor
adrv      equ   0BEF0h      ;abs. Laufwerknr.
trk       equ   0BEF2h      ;Spurnr.
sec       equ   0BEF4h      ;Sektornr.
dma       equ   0BEF6h      ;DMA Adresse
dmabnk    equ   0BEF9h      ;DMA Bank
sendmsg   equ   00572h      ;BIOS Einschaltmeldung ausgeben

;Parameter des Installationsprogramms
;-----------------------------------------------------------------
ramdsk    equ   6c00h       ;Adr des RAM-Disk-Treibers im Sys-File
boot      equ   0           ;CP/M Warm
bdos      equ   5           ;BDOS
fcb1      equ   5ch         ;File Control Block 1
lf        equ   10          ;ASCII Konstanten:
cr        equ   13

patch macro ident           ;ein Makro zum Patchen des Sys-Files
      ld    hl,ident&src
      ld    de,ident&dst
      ld    bc,ident&end-ident&src
      ldir
      endm

;RAM Disk Installationsprogramm:
;-----------------------------------------------------------------
instal:
      ld    (oldsp),sp      ;Stack einrichten
      ld    sp,stack
      ld    de,hallo        ;Begruessung
      call  print
      call  open            ;System-File oeffnen
      call  sysread         ;lesen
      call  sysptch         ;patchen
      call  open
      call  syswrte         ;schreiben
      call  close           ;und dann schliessen

exit:
      ld    sp,(oldsp)      ;Stack restaurieren
      jp    boot            ;und fertig...

sysread:                    ;System File an die Adr c00h einlesen:
      ld    de,0c00h        ;also DMA auf c00h setzen
      call  setdma
      ld    e,128           ;die ersten 128 Records
      call  multis          ;lesen
      xor   a
      ld    (fcb1+32),a     ;Current Record=0
      call  readsq
      ld    de,0c00h+128*128
      call  setdma          ;dann die restlichen 72
      ld    e,72
      call  multis
      jp    readsq

syswrte:                    ;gepatchten Systemfile speichern:
      ld    de,0c00h        ;DMA=c00h
      call  setdma
      ld    e,128           ;zuerst 128 Records
      call  multis
      xor   a
      ld    (fcb1+32),a     ;Current Record=0
      call  writesq         ;schreiben
      ld    de,0c00h+128*128
      call  setdma          ;dann den Rest
      ld    e,72
      call  multis
      jp    writesq

sysptch:                    ;System File patchen:
      patch lodr            ;CP/M Plus Loader
      patch dtrd            ;RAM Disk ins Drive Table
      patch seld            ;Select Disk
      patch rdc             ;RAM Disk Controller
      patch buffr           ;BCBs
      patch mesg            ;Einschaltmeldung...

   if bigdrives             ;*
      patch ptch1           ;2. Head Check
      patch ptch2           ;Seek Track
   endif

   if floppy3               ;*
      patch dtfd3           ;Floppy3 ins Drive Table
   endif

      ret

;BDOS Funktionsaufrufe:
;-----------------------------------------------------------------
print:
      ld    c,9             ;Print String
      jp    bdos

open:
      ld    de,fcb1         ;Open File
      xor   a
      ld    (fcb1+12),a     ;extent=0
      ld    c,15
      jp    exec

close:
      ld    de,fcb1         ;Close File
      ld    c,16
      jp    exec

readsq:
      ld    de,fcb1         ;Read Sequential
      ld    c,20
      jp    exec

writesq:
      ld    de,fcb1         ;Write Sequential
      ld    c,21
exec:
      call  bdos
      or    a
      jp    nz,exit         ;bei Error raus...
      ret

setdma:
      ld    c,26            ;Set DMA Address
      jp    bdos

multis:
      ld    c,44            ;Set Multi-Sector Count
      jp    bdos

hallo:db    cr,lf,'RAM Disc Installer V 2.0/  gh`87',cr,lf,lf
      db    'RAM disc drive: ',rddrv,':',cr,lf,lf,'$'

oldsp:                      ;Stackbereich
```

```
197         ds    64
198   stack:
199
200   ;CP/M 3 System File Patches
201   ;---------------------------------------------------------------
202   lodrsrc:                    ;die CPC Versions-Pruefroutine
203   lodrdst     equ   0cd5h     ;muss dem RAM Disk Loader Platz machen
204         .phase      lodrdst
205         ld    hl,ramdsk       ;Der RAM Disk Controller wird an die
206         ld    de,0aa00h       ;Adr. AA00h geschoben...
207         ld    bc,rdcend-rdcsrc
208         ldir
209         ret
210         .dephase
211   lodrend:
212
213   dtrddst  aset 262fh+2*(rddrv-'A')     ;RAM Disk DPH ins
214   dtrdsrc:                    ;Drive-Table eintragen
215         dw    dph_rd
216   dtrdend:
217
218      if floppy3
219                                ;wenn vorhanden
220   dtfd3dst aset  262fh+2*(fd3drv-'A')   ;Floppy 3 ins DPH eintragen
221   dtfd3src:
222         dw    dph_fd3
223   dtfd3end:
224
225      endif
226
227      if bigdrives
228
229   ptch1dst aset     1a1ah     ;*Patch in der Resultphasen-
230   ptch1src:                   ;auswertung
231         call  TwoInOne
232   ptch1end:
233
234
235   ptch2dst aset     18c7h     ;*Patch in der Seek Track
236   ptch2src:                   ;Routine
237         jp    Steprate
238   ptch2end:
239
240      endif
241
242
243   selddst     aset  3057h
244   seldsrc:                    ;SelDisk umbiegen
245         call  seldsk
246   seldend:
247
248   buffrdst aset     0e5ah     ;Init BCB umbiegen
249   buffrsrc:
250         .phase      buffrdst
251         jp    initbuf
252         .dephase
253   buffrend:
254
255   mesgsrc:                    ;Zusatz bei der Einschaltmeldung
256   mesgdst     equ   1d5fh     ;als 31. Eintrag in der Message-Table
257         db    'RAM disc drive '
258         db    0fdh,' '        ;FDh= Inh. von Reg C ausgeben(DriveNr)
259         db    0f9h,' KB'      ;F9h= Inh. von DE ausgeben(Kapazitaet)
260         db    cr,lf,lf,0ffh
261   mesgend:
262
263   ;RAM Disk:
264   ;---------------------------------------------------------------
265   rdcsrc:
266   rdcdst      aset  ramdsk
267         .phase      0aa00h
268
269   read:                       ;Sektor lesen:
270         call  gettrk          ;Spurnummer holen und
271         call  xmove           ;bankuebergreifendes Kopieren vorber.
272         call  getsec          ;Basisadr. aus der Sektornr. berechnen
273         call  bmove           ;und Sektor kopieren...
274         xor   a               ;A=0 d.h. alles O.K.
275         ret
276
277   write:                      ;Sektor schreiben:
278         call  gettrk          ;Spurnr. holen
279         ld    a,c             ;wie bei read:, nur Ziel und Quelle
280         ld    c,b             ;werden vertauscht
281         ld    b,a
282         call  xmove           ;Werte an XMOVE uebergeben
283         call  getsec          ;Basisadr. berechnen
284         ex    de,hl           ;auch tauschen
285         call  bmove           ;Sektor kopieren
286         xor   a               ;alles O.K.
287         ret
288
289   login:                      ;Laufwerk einloggen:
290         ex    de,hl           ;DPH-Adresse in hl
291         ld    a,(bootflg)     ;erster Zugriff nach
292         or    a               ;einem Kaltstart?
293         ret   z               ;nein - fertig
294         xor   a               ;sonst RAM Disk initialisieren
295         ld    (bootflg),a     ;Flag loeschen
296         push  hl              ;Regs retten
297         push  de
298         push  bc
299         ld    a,track0        ;Directory ab 4000h einblenden
300         call  selbnk
301         ld    hl,sector0+1    ;ersten Eintrag ueberpruefen
302         ld    b,8             ;testen ob Filename
303
304   checkd:     ld    a,(hl)    ;aus ASCII Grossbuchstaben,
305         cp    ' '             ;SPACE
306         jr    z,nxtchk
307         cp    '0'             ;oder Zahlen besteht.
308         jr    c,newdir        ;wenn nicht - Directory loeschen
309         cp    '9'+1
310         jr    c,nxtchk
311         cp    'A'
312         jr    c,newdir
313         cp    'Z'+1
314         jr    nc,newdir
315   nxtchk:     inc   hl
316         djnz  checkd          ;naechses Zeichen
317         jr    exlogn          ;alles klar, Directory bleibt...
318
319   newdir:                     ;Directory loeschen:
320         ld    hl,sector0      ;ab Track 0/Sektor 0
321         ld    d,h             ;mit E5h fuellen
322         ld    e,l
323         inc   de
324         ld    bc,dirblks*2048;dirblks Bloecke
325         ld    (hl),0E5h
326         ldir
327
328   exlogn:                     ;login beenden:
329         xor   a               ;Systembank ein
330         call  selbnk
331         pop   bc              ;Regs restaurieren
332         pop   de
333         pop   hl
334         ret
335
336   gettrk:                     ;Quell- und Zielbank aus trk und
337                               ;dmabnk berechnen:
338         ld    a,(trk)         ;BIOS Spurnr. ins Akku
339         add   a,track0        ;Banknummer der 1. Spur dazu
340         ld    c,a             ;und in Reg c ablegen,
341         ld    a,(dmabnk)      ;Reg b enthaelt die BankNr. des DMA
342         ld    b,a
343         ret
344
345   getsec:                     ;Quell und Zieladr. bereitstellen:
346         ld    hl,(sec-1)      ;BIOS Sektornr. ins h
347         ld    l,0             ;l=0 => hl=sec*256
348         srl   h               ;hl/2 = sec*128
349         rr    l
350         ld    de,sector0      ;Sektoradr.=sec*128+sector0
351         add   hl,de
352         ex    de,hl           ;Sektoradr in de
353         ld    hl,(dma)        ;DMA in hl
354         ld    bc,80h          ;Sektorlaenge in bc
355         ret
356
357
358   ;BIOS Aenderungen:
359   ;---------------------------------------------------------------
360
361   seldsk:                     ;die neue Select Disk Routine:
362         ld    (adrv),a        ;urspruengliche Funktion
363         ld    hl,dpbidnt      ;Drive schon selektiert?
364         cp    (hl)
365         ret   z               ;ja - fertig
366         push  de              ;de schon mal retten...
367         ld    de,dpb_b        ;DPB B: ins de
368         cp    1               ;ist es B:?
369         jr    z,copydpb
370         ld    de,dpb_rd       ;sonst DPB RamD: laden
371         cp    rddrv-'A'       ;ist es die RAM Disk?
372         jr    z,copydpb       ;ja - kopieren
373
374      if floppy3               ;wenn vorhanden
375         ld    de,dpb_fd3      ;genauso mit 3. Laufwerk
376         cp    fd3drv-'A'      ;verfahren
377         jr    z,copydpb
378      endif
379
380         pop   de
381         ret                   ;war nicht dabei - raus...
382
383   copydpb:
384         push  af              ;Regs retten und
385         push  bc
386         ld    (hl),a          ;als akt. Laufwerk speichern und
387         ld    hl,0ff7fh       ;DPB nach ff7fh kopieren
388         ex    de,hl
389         ld    bc,27
390         ldir
391         pop   bc              ;Regs vom Stack
392         pop   af
```

```
393         pop   de              ;und O.K.
394         ret
395
396   initbuf:                    ;die neue Init BCB Routine:
397         ld    bc,rddirs*256   ;RAM Disk Directory Buffer
398         ld    de,rddirb       ;muss in Bank0 liegen
399         ld    hl,8a00h        ;BCB Adresse
400         ld    a,1             ;128 Byte Sektoren
401         ld    (dirbcbv),hl    ;Adresse des BCB speichern
402         call  initbcb         ;BCB aufbauen
403         dec   hl
404         ld    (hl),b          ;letzter Eintrag (hl)=0
405         dec   hl
406         ld    (hl),b          ;(hl-1)=0
407         inc   hl
408         inc   hl
409
410         ld    bc,fddirs*256   ;DirBCB fuer A: und B: Bank 0
411         ld    de,fddirb
412         ld    a,fdsecsz/80h   ;Sektorengroesse in a
413         push  hl              ;Adresse auf'n Stack
414         call  initbcb         ;und aufbauen
415         dec   hl
416         ld    (hl),b
417         dec   hl
418         ld    (hl),b          ;letzter Eintrag
419         inc   hl
420         inc   hl
421
422         push  hl
423         ld    bc,fddats1*256+fddbnk1 ;DatBCB fuer die Floppies
424         ld    de,fddatb1             ;Nr 1
425         call  initbcb
426         ld    bc,fddats2*256+fddbnk2 ;Nr. 2
427         ld    de,fddatb2
428         call  initbcb
429         ld    bc,fddats3*256+fddbnk3 ;und Nr. 3
430         ld    de,fddatb3
431         call  initbcb
432         dec   hl                     ;letzter Eintrag
433         ld    (hl),b
434         dec   hl
435         ld    (hl),b
436
437         ld    hl,bcbpars      ;BCBPars nach FFE8h kopieren
438         ld    de,0ffe8h
439         ld    bc,bcbpend-bcbpars
440         ldir
441         call  initmsg         ;RAM Disk Init Message ausgeben
442
443   if  bigdrives               ;*Patch 4 & 5
444         ld    a,255           ;wird nach jedem Kaltstart aufgerufen
445         ld    (0be40h),a      ;Flag fuer 2 Drives
446         ld    a,1
447         ld    (3fb5h),a       ;Drive B:
448   endif
449
450         pop   hl              ;hl = Adr. DirBCB der Floppies
451         pop   de              ;de = Adr. DatBCB  "      "
452         ret
453
454   initbcb:
455   ; Buffer Control Block aufbauen --  Parameter:
456   ; a =Sektorengroesse/128:  b =Puffergroesse in Sektoren
457   ; c =Pufferbank          ;  de=Pufferadresse
458   ; hl=Adresse des BCBs
459
460         inc   b               ;Laenge=0?
461         dec   b
462         ret   z               ;ja - Abbruch...
463         push  hl              ;
464         ld    h,a             ;Sektorlaenge in Bytes
465         srl   h               ;umrechnen
466         ld    l,0             ;
467         rr    l               ;
468         ex    (sp),hl         ;Sektorlaenge --> Stack
469
470   bcbloop:
471         push      de
472         ld    (hl),0ffh       ;drivenr. = 0ffh
473         ld    de,0fh          ;naechses BCB 15 Bytes weiter
474         add   hl,de
475         ld    d,h
476         ld    e,l
477         dec   hl
478         ld    (hl),d          ;Addresse eintragen
479         dec   hl
480         ld    (hl),e
481         dec   hl
482         ld    (hl),c          ;Speicherbank des Puffers
483         ex    de,hl
484         ex    (sp),hl
485         ex    de,hl
486         dec   hl
487         ld    (hl),d          ;Pufferadresse speichern
488         dec   hl
489         ld    (hl),e
490         pop   hl
491         ex    (sp),hl
492         ex    de,hl
493         add   hl,de           ;naechste Pufferadresse berechnen
494         ex    de,hl
495         ex    (sp),hl
496         djnz  bcbloop
497         inc   sp              ;Sektorlaenge vom Stack
498         inc   sp
499         ret
500
501     if bigdrives
502
503   TwoInOne:                   ;*
504         ld    a,c             ;ausblenden der Headadresse
505         res   2,a             ;um Kopf 2 im Result des FDC
506         or    20h             ;wie Kopf 1 zu behandeln
507         ret
508
509   Steprate:                   ;*
510         add   iy,bc           ;urspruengliche Funktion
511         push  hl              ;
512         srl   c               ;Drivenr. in C
513         ld    hl,Steptab      ;
514         add   hl,bc           ;Eintrag finden
515         ld    a,(hl)
516         ld    hl,0b0f1h       ;als akt. Steprate
517         cp    (hl)            ;gefunden?
518         jr    z,stimmt
519         ld    (hl),a          ;speichern
520         ld    hl,0ad5h        ;HL auf Headload/ -unload
521         call  0af2h           ;an FDC uebergeben
522   stimmt:
523         pop   hl
524         pop   bc
525         ret
526
527   Steptab:
528         db    12              ;Steprate A:
529         db    Step_B          ;Steprate B:
530         db    0               ;Dummy
531         db    Step_C          ;Steprate C:
532
533     endif
534
535   initmsg:                    ;RAM Disk Meldung nach dem BOOTen
536         ld    c,rddrv-'A'     ;DriveNr. und
537         ld    de,tracks*16    ;Kapazitaet
538         ld    a,31            ;31. Eintrag aus der Message Table
539         jp    sendmsg         ;ausgeben...
540
541   bcbpars:  ; Diese Bytes muessen ins Common-Bereich kopiert
542         db    0               ;DirBCB Bank
543         dw    fddirb          ;       Buffer
544         dw    2469h           ;       Länge
545
546         db    fddbnk1         ;DatBCB Bank
547         dw    fddatb1         ;       Buffer
548         dw    2f80h           ;       Länge
549         db    0ffh
550   bcbpend:
551
552   ;XDPHs und DPHs
553   ;-----------------------------------------------------------------
554
555   ;RAM Disk:
556   xdph_rd:                    ;eXtended Disk Parameter Header
557         dw    write           ;Sektor schreiben
558         dw    read            ;Sektor lesen
559         dw    login           ;Disk "einloggen"
560         dw    0               ;"Init"-Vektor (wird vom CP/M nicht
561                               ;aufgerufen...)
562         db    2               ;physikalische Drivenr.
563         db    0
564
565   dph_rd:                     ;Disk Parameter Header
566         dw    0               ;kein Skew
567         db    0,0,0           ;BDOS Scratch
568         db    0,0,0
569         db    0,0,0
570         db    0               ;kein MF, da kein Diskwechsel moeglich
571         dw    0ff7fh          ;DPB Vektor
572         dw    0               ;kein Checksum Vektor
573         dw    alv_rd          ;Allocation Vektor
574         dw    dirbcbv         ;DirBCB Vektor
575         dw    datbcbv         ;DatBCB     Vektor
576         dw    05280h          ;Hashing Table
577         db    2               ;in Bank 2
578
579   dpb_rd:
580         dw    128             ;Sektoren pro Spur
581         db    4               ;Blockgroesse 2 KB
582         db    15
583         db    0
584         dw    tracks*8-1      ;Kapazitaet in blocks
585         dw    dirblks*64-1    ;Dir-Groesse
586         db    al0,al1
587         dw    0               ;CKS/ kein Checksum
588         dw    0               ;Off/ kein Offset
```

```
589         db    0               ;PSH/ kein Sektor Deblocking.
590         db    0               ;PHM
591                                 ;Amstrad's params:
592         db    0,0,0,0,0,0,0,0,0,0
593
594   alv_rd:
595         ds    (tracks*8-1)/4+2 ;Platz fuer die Belegungstabelle
596
597   dirbcbv:
598         dw    0
599   datbcbv:
600         dw    datbcb
601
602   datbcb:
603         db    0FFh            ;Datenpuffer 1 Sektor
604         ds    3
605         db    0
606         db    0
607         dw    0
608         dw    0
609         dw    datbuf
610         db    0
611         dw    0
612
613   datbuf:
614         ds    128
615
616      if floppy3               ;nur wenn 3. Laufwerk angeschlossen
617
618   ;3. Laufwerk
619   xdph_fd3:
620         dw    03f1ch          ;Write
621         dw    03f17h          ;Read
622         dw    03ed6h          ;Login
623         dw    03ecbh          ;Init
624         db    3               ;phys. Drivebezeichnung
625         db    0
626   dph_fd3:
627         dw    0               ;kein Skew
628         db    0,0,0,0,0,0,0,0,0,0
629
630         db    0               ;MF (Mediaflag)
631         dw    0ff7fh          ;wird mit dem DPB von B: vertauscht!
632         dw    csv_fd3         ;Checksum Vector
633         dw    alv_fd3         ;Allocation Vector
634         dw    0befbh          ;DirBCB
635         dw    0befdh          ;DatBCB
636         dw    5080h           ;Hashing Table
637         db    2               ;in Bank 2
638
639   csv_fd3:ds  33              ;Checksum (DRM/4)+1
640
641   alv_fd3:ds  66              ;Allocation Tab. (DSM/4)+2
642
643   dpb_fd3:
644
645   ; **** HIER DPB FOR LAUFWERK C: EINTRAGEN **** ;
646   ; **** (und Loginflag am Ende auf 0FFh setzen) ;
647
648      endif
649
650   ;Drive B:
651   dpb_b:
652   ; **** Hier DPB für Laufwerk B: eintragen **** ;
653   ; **** (Loginflag auf 0FFh)
654
655   dpbidnt:
656         db    0
657   bootflg:
658         db    0ffh
659         .dephase
660
661   rdcend:
662
663         end
664
665
```

Alle nötigen Patches für die RAM-Disk und für eventuelle zusätzliche Laufwerke werden direkt im CPM-System-File vorgenommen. Durch Änderung von Variablen im Listing lassen sich verschiedene Konfigurationen erzeugen.

ct