

## Teil 2

weiter geht es mit dem **DPB**

## Beispiel 2

Wir wollen ein RAM-Floppy ansteuern. Die **RAM-Floppy (NANOS)** hat folgende Eigenschaften:

- 256 K Gesamtkapazität
  - die RAM-Floppy kann einen Speicherbereich von 256 Byte in den Hauptspeicher einblenden

Ein Byte mit Adresse A17..A0 in der RAM-Floppy wird so angesprochen:

1. Ausgabe A17..A16 auf Port „Bank“,
  2. Ausgabe A15..A8 auf Port „HiAdr“,
  3. Einblenden in den Hauptspeicher (auf Adresse „Window“ bis „Window“+255,
  4. Zugriff auf „Window“+A7..A0

Für die Nutzung im CP/M soll außerdem eine Kopie von CCP+BDOS (5 KByte) auf der RAM-Disk gehalten werden, sinnvollerweise in Systemspuren.

Eine RAM-Floppy hat keine physischen Spuren, deshalb kann man die Aufteilung in virtuelle Spuren und Sektoren nach eigenen Ideen vornehmen.

## Die Ansteuerung als Übersicht:

Diagramm zur Darstellung von RAM-Adressen (RAF 256K) und deren Zuordnung zu TRACK, SECTOR und RECORD.

Die obere Zeile zeigt die **Vollständige 18 bit RAM-Adresse (RAF 256K)** mit den Bitnummern von 17 bis 0 von links nach rechts. Unter der Adresse sind zwei Varianten von 18-bit-Adressen dargestellt:

- Variante 1:** Bank | | Hi-Adr. | | Window |
- Variante 2:** TRACK - SECTOR - RECORD

Unter den Varianten sind die Bitnummern von 9 bis 0 von links nach rechts angegeben. In Variante 1 sind die Bits 9 bis 0 unter **Bank**, **Hi-Adr.** und **Window** aufgetrennt. In Variante 2 sind die Bits 9 bis 0 unter **TRACK**, **SECTOR** und **RECORD** aufgetrennt.

## **Variante 1**

Die FenstergröÙe von 256 Byte bietet es an, die SpurgröÙe als 256 Byte zu wählen. Hi-Byte und Lo-

Byte der Tracknummer sind dann direkt „Bank“ und „HiAdr“. Das macht die Ansteuerung besonders einfach.

also:

1 Track = 256 Byte (Fenstergröße)

d.h. 2 Records/track

wir brauchen damit  $1600\text{h}/256 = 22$  Tracks f. Systemspur

insg. 1024 tracks  $\rightarrow$  DSM = 1023-22

wir wählen die kleinstmögliche Blockgröße 2k (1k gehen nicht wg. EXM, da wir mehr als 256 Spuren haben)

und z.B. 128 Dir-Einträge (d.h. 2 Dir-Blöcke)

Für den DPB ergibt sich damit:

```
;DISKDEF 0,1,2,,2048,1012,128,0,22
dpb00: dw      2      ;sec per track
      db      4      ;block shift
      db      15     ;block mask
      db      0      ;extnt mask
      dw      1011    ;disk size-1
      dw      127     ;directory max
      db      C0h    ;alloc0
      db      0      ;alloc1
      dw      0      ;check size
      dw      22     ;offset
;
alv00: ds      007Fh
csv00: ds      0000
```

Die BIOS-Routinen zum Blocklesen und -schreiben verweisen auf folgende Routinen:

```
; Lesen von Diskette
READ:  CALL    ADRE
READ1: LDIR
      OUT    (READDI), A
      OUT    (RAMDI), A
      XOR    A
      RET

; Schreiben auf Diskette
WRITE: CALL    ADRE
      EX     DE,HL
      JR    READ1-# 

; Berechnung Adr.
ADRE:  OUT    (RAMEN), A
      OUT    (READEN), A
;
      LD     HL,(TRACK)
      OUT    (LDAH), L      ; hi-adr.
```

```

OUT    (LDBB), H      ; Bank
LD     HL, WINDOW      ; das ist eine xx00h-Adr.
LD     a, (SECTOR)     ; 1 oder 2 (in CP/A wg. SECTRAN)
CP     2
jr    nz, ADRE0a
LD     L,80h
ADREa: LD    DE,(DMAAD)
LD    BC,128
RET

```

## Variante 2

Um eine kleinere Blockgröße nutzen zu können, muss die Anzahl der Spuren  $\leq 256$  werden. Da geht z.B. mit einer Spurgröße von 2 KByte.

1 Track = 2048 Byte  
d.h. 16 Records/Track  
wir brauchen damit  $1600h/2048 = 3$  Tracks f. Systemspur  
insg. 256 Tracks  $\rightarrow$  DSM = 255-3  
kleinste Blockgröße 1k  
und z.B. 64 Dir-Einträge (d.h. 2 Dir-Blöcke)

Diese Aufteilung ist wg. der kleineren Blockgröße günstiger, wenn viele kleine Dateien auf der RAM-Disk gehalten werden sollen. Auch wird weniger Platz für den Allocation Vektor ALVxx benötigt. Aber die Umrechnung logischer Track-Sektor  $\rightarrow$  Adr. f. RAM-Disk ist aufwendiger!

```

;DISKDEF 1,1,16,,1024,252,64,0,3
dpb01: dw    16      ;sec per track
      db    3      ;block shift
      db    7      ;block mask
      db    0      ;extnt mask
      dw    251     ;disk size-1
      dw    63      ;directory max
      db    C0H     ;alloc0
      db    0      ;alloc1
      dw    0      ;check size
      dw    3      ;offset
alv01: ds    0020h
csv01: ds    0000h

```

Read und Write sind wie oben implementiert, die Adressierung ist jetzt umfangreicher:

```

ADRE: OUT    (RAMEN), A
      OUT    (READEN), A
;
;Adr. Fenster = (track*16+sector)/2
LD    HL,(TRACK)
ADD   HL,HL
ADD   HL,HL
ADD   HL,HL

```

```
ADD  HL,HL      ; HL = Track * 10h (SPT)
LD   DE,(SECTOR)
DEC  DE      ; wg. CP/A
ADD  HL,DE      ; HL := HL + Sector
XOR  A      ; A = 0, Cy = 0
RR   H
RR   L      ; HL := HL/2   ( da 2 Sektoren/Fenster )
RR   A      ; L Bit0 nach A Bit7 ( A = 0 oder 80h)
OUT  (LDAH), L      ; hi-adr.
OUT  (LDBB), H      ; Bank
LD   H, Hi(WINDOW)
LD   L,A
LD   DE,(DMA)
LD   BC,128
RET
```

From:

<https://hc-ddr.hucki.net/wiki/> - **Homecomputer DDR**



Permanent link:

[https://hc-ddr.hucki.net/wiki/doku.php/cpm/write\\_a\\_bios/teil\\_2?rev=1331889909](https://hc-ddr.hucki.net/wiki/doku.php/cpm/write_a_bios/teil_2?rev=1331889909)

Last update: **2012/03/16 09:25**