





### Variante 1

Die Fenstergröße von 256 Byte bietet es an, die Spurgröße 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 1..2
- wir brauchen damit 1600h/256 = 22 Tracks f. Systemspur
- insg. 1024 tracks → DSM = 1024-22 = 1012
- wir wählen die kleinstmögliche Blockgröße 2k (1k gehen nicht wg. EXM, da DSM > 255)
- 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 ;SPT sectors per track
db 4 ;BSF block shift factor
db 15 ;BLM block mask
db 0 ;EXM null mask
dw 1011 ;DSM disk size-1
dw 127 ;DRM directory max
db C0h ;AL0 alloc 0
db 0 ;Al1 alloc 1
dw 0 ;CKS check size
dw 22 ;OFS track offset
;
alv00: ds 007Fh ;allocation vector
csv00: ds 0000 ;check vector
    
```

Die BIOS-Routinen zum Blocklesen und -schreiben verweisen auf folgende Routinen. Wegen der Spurgröße von 256 Byte = 2 Records muss ein Blocking/Deblocking erfolgen. Glücklicherweise ist das bei einer RAM-Disk nicht weiter schwierig umzusetzen, da innerhalb des Zugriffsfensters nur der angesprochene Bereich von 128 Byte gelesen bzw. verändert wird.

```

; 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 Spurgöße von 2 KByte.

1 Track = 2048 Byte  
d.h. 16 Records/Track 1..16  
wir brauchen damit  $1600h/2048 = 3$  Tracks f. Systemspur  
insg. 128 Tracks  $\rightarrow$  DSM = 128-3  
kleinste Blockgröße 1k  
und z.B. 64 Dir-Einträge (d.h. 2 Dir-Blöcke)

Diese Aufteilung ist aufgrund 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 und Sektor  $\rightarrow$  Adr. f. RAM-Disk ist aufwendiger!

```

;DISKDEF 1,1,16,,1024,125,64,0,3
dpb01:  dw    16          ;SPT sectors per track
        db    3          ;BSF block shift factor
        db    7          ;BLM block mask
        db    0          ;EXM null mask
        dw    124        ;DSM disk size-1
        dw    63        ;DRM directory max
        db    C0H        ;AL0 alloc 0
        db    0          ;Al1 alloc 1

```

```

    dw    0      ;CKS check size
    dw    3      ;OFS track offset
;
alv01:  ds     0010h      ;allocation vector
csv01:  ds     0000h      ;check vector

```

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

```

### Beispiel 3

Ein Floppy-Laufwerk: 800K, 2 Seiten, 1K phys. Sektorgröße, 5 phys. Sektoren pro Spur und Seite

- log. Sektoren pro Spur 1..40
- Blockgröße BLS = 2048
- Diskgröße = 800K/BLS = 400
- 192 Directory-Einträge
- keine Systemspuren

```

        ;DISKDEF 0,1,40,,2048,400,192,192,0
dpba:   dw     40      ;SPT sectors per track
        db     4      ;BSF block shift factor
        db     15     ;BLM block mask
        db     0      ;EXM null mask
        dw     399    ;DSM disk size-1
        dw     191    ;DRM directory max

```

```

db    0E0H        ;AL0 alloc 0
db    0           ;All alloc 1
dw    48          ;CKS check size
dw    0           ;OFS track offset
alva: ds    0032h
csva: ds    0030h

```

CP/M zählt die logischen Recordnummern pro Spur von 0..39. SECTTRAN übersetzt diese Recordnummern in 1..40 (CP/A-Umrechnung) und übergibt diese berechnete Recordnummer mit SETSEC ans BIOS.

CP/M ermittelt anhand DSM, ob 16Bit- oder 8-Bit-Blocknummern genutzt werden:  $DSM > 255 \rightarrow$  16Bit-Blocknummern.

Die max. Spurnummer berechnet sich als  $DSM * BLS / SPT / 128 - OFS$ . CP/M arbeitet aber intern nicht mit einer maximalen Spurnummer, sondern testet auf Überschreiten von DSM.

Der DPB wird vom CP/M wie folgt angezeigt:

```

A>stat dsk:
  A: Drive Characteristics
6400: 128 Byte Record Capacity
 800: Kilobyte Drive Capacity
 192: 32 Byte Directory Entries
 192: Checked Directory Entries
 128: Records/ Extent
  16: Records/ Block
  40: Sectors/ Track
   0: Reserved Tracks

```

POWER gibt ein paar mehr Informationen aus:

```

          POWER 3.03 on CP/M 2.22 1/2
A=disk
disk capacity:      800K
tracks:            160    0 system
sectors/track:     40    40 last
sectors/system:    0     48 dir
dir entries:       192    6K
sectors/group:     16    2K 18FH groups
kbytes/extent:     16K

```

Bislang wurde noch nicht darauf eingegangen, dass eine phys. Diskette 2 Seiten hat. Die Adressierung von Diskettenseite/Spur/phys.Sektor incl. Blocking/Deblocking ist Aufgabe des BIOS.

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=1538722171](https://hc-ddr.hucki.net/wiki/doku.php/cpm/write_a_bios/teil_2?rev=1538722171)



Last update: **2018/10/05 06:49**