

Teil 2

weiter geht es mit dem **DPB**

Das BIOS erhält die Information, wo konkret auf die Datenspeicher zugegriffen werden soll, via BIOS-Funktion

- SETDSK (Laufwerk, 0..15),
- SETTRK (Track, Spur 0...x, berechnet aus Blocknummer + Systempuren),
- SECTRAN (Transformation der aktuellen log. Record-Nummer 0..BSM, Sektorversatz, skewing)
- und SETSEC (Sector, transformierte Record-Nummer).

Das BDOS ruft diese genannten BIOS-Funktionen immer vor Aufruf von READ oder WRITE in dieser Reihenfolge auf. Ein BIOS speichert daher die Werte für Laufwerk, Spur, Sektor zwischen und greift bei Lese- und Schreiboperationen darauf zu. Pro Format/Laufwerkstyp gibt es dazu üblicherweise eine Funktion zur Adressberechnung, die aus logischer Spurnummer und transformierter Record-Nummer die physisch anzusprechenden Werte ermittelt. Bei Disketten sind das Seite, Spur, phys. Sektor, Pos. innerhalb des phys. Sektors.

Das folgende Beispiel stammt aus einer [CP/A](#)-Implementation. Bei CP/A erfolgt die Sektorzählung ab 1, deshalb steht in sectran ein inc hl. Die Zählung ab 1 muss bei den direkten Zugriffen (Adressberechnung, Read, Write) beachtet werden!

```
;-----  
--  
; Uebersetzung Sektornummer in CP/A  
;-----  
--  
sectran:    ld     h, b  
            ld     l, c  
            inc    hl        ;Sektoren zaehlen in CP/A ab 1  
            ret
```

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

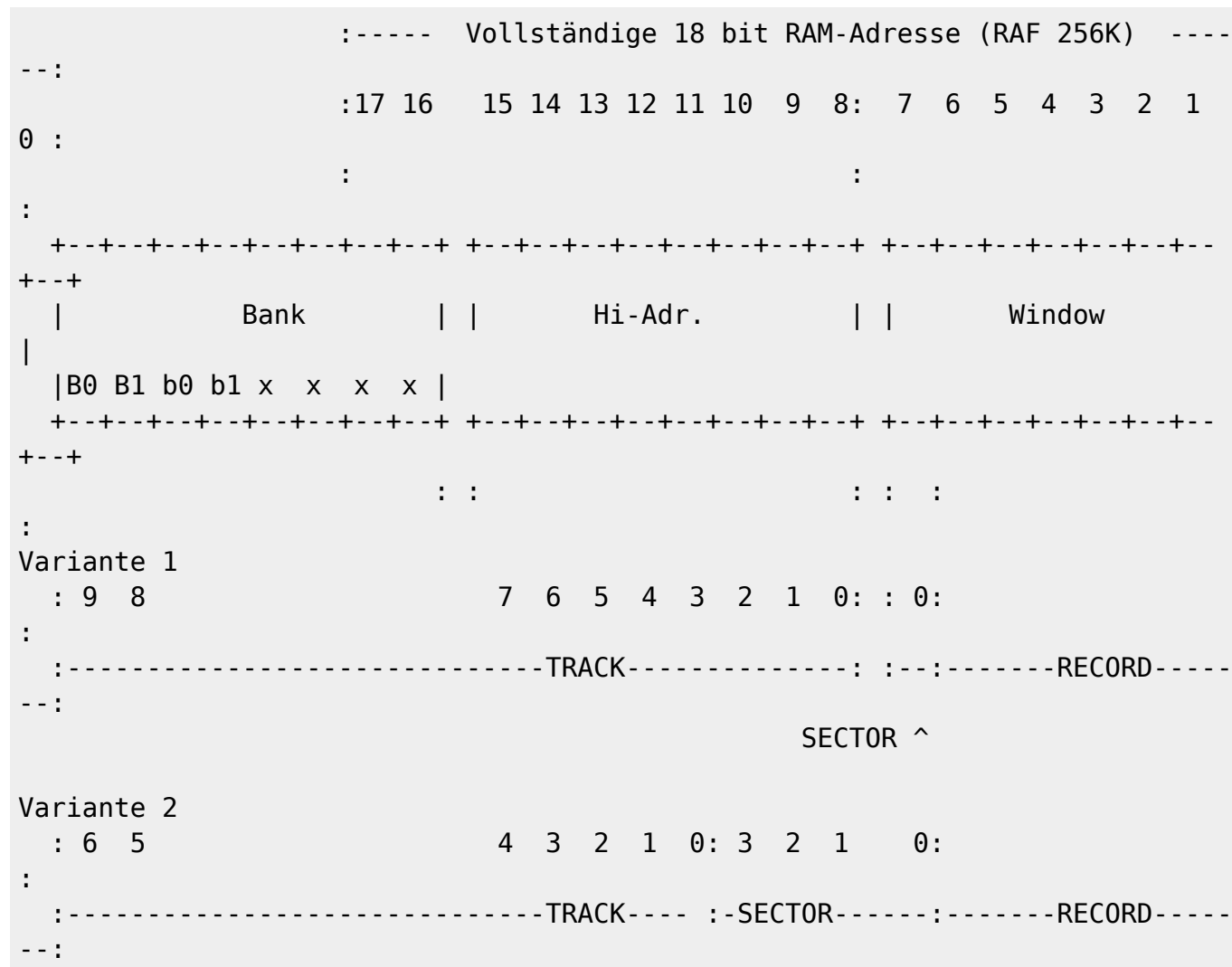
Ein Byte mit Adresse A17..A0 in der RAM-Floppy wird so angesprochen:

1. Ausgabe A17..A16 auf Port „Bank“,
2. Ausgabe A15..A8 auf Port „HiAdr“,
3. Einblenden von 256 Byte 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:



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 \rightarrow 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.
        ld      a,h
        rrca
        rrca
        OUT     (LDBB), A      ; Bank
        LD      HL, WINDOW     ; das ist eine xx00h-Adr.
        LD      a, (SECTOR)    ; 1 oder 2 (in CP/A wg. SECTAN)
        CP      2
        jr      nz, ADRE0a
        LD      L, 80h
ADREa:  LD      DE, (DMAAD)

```

```
LD    BC,128
RET
```

Variante 2

(System EPOS, EPOSRF2.MAC)

Um eine kleinere Blockgröße nutzen zu können, muss die Anzahl der Spuren $\leftarrow 256$ werden. Da geht z.B. mit einer Blockgröße von 1 KByte. Als phys. Spurgröße wählen wir 2 KByte.

1 Track = 2048 Byte

d.h. 16 Records/Track 0..15 (CPA 1..16)

wir brauchen $1600h/2048 = 3$ Tracks f. Systemspur OFF, Epos 1 Systemspur

wie wählen kleinste Blockgröße BLS 1k, 8 bit-Blocknummern (DSM < 256)

insg. 256 Tracks \rightarrow DSM = $(256-3*2)/1-1 = 249$, (epos 253)

und z.B. 64 Dir-Einträge (d.h. 2 Dir-Blöcke)

DSM = (DISKSIZE-OFF*SPURSIZE)/BLS-1

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!

```
dpbm:    dw    16            ;spt
         db    3            ;bsh
         db    7            ;blm
         db    0            ;exm
         dw    253          ;dsm
         dw    63           ;dpm
         db    0c0h         ;al0
         db    00           ;all
         dw    0            ;cks
         dw    1            ;off
         db    80h          ;dev
         ds    12,0

;
allm:    ds    33           ;alloc-map
;
rfrwoper:                ;physisches schreiben
         ld    hl,(septrk)  ;und lesen
         add   hl,hl
         add   hl,hl
         add   hl,hl        ; trk << 3
         ld    a,(seksec)
         and   1fh
         srl   a            ; sec >> 1
         push  af
         add   a,l
         di
```

```

    out    (ioa+5),a
    out    (ioa+7),a
    out    (ioa),a      ;Hi-Adr. im ramfl
    ld     a,03h
    and    h            ;Bit0+1
    rrca
    rrca          :->Bit6+7
    out    (ioa+2),a    ;bank
;
    pop    af          ;cy=off im fenster
    ld     hl,wind
    jr     nc,rf01
    ld     hl,wind+128
rf01:    ld     de,(dmaad)
    ld     a,b
    or     a            ;rd/wr
    jr     z,rf02
    ex     de,hl
rf02:    ld     bc,128
    ldir
;
    xor     a
    out    (ioa+4),a
    ei
    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            ;Al1 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.

TODO

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

Last update: **2025/04/19 15:36**

