2025/12/18 16:26 1/4 Teil 2

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:

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
wir brauchen damit 1600h/256 = 22 Tracks f. Systemspur
insg. 1024 tracks → 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
                         ;sec per track
    db
           4
                     ;block shift
    db
           15
                     ;block mask
    db
                     ;extnt mask
    dw
           1011
                        ;disk size-1
    dw
           127
                       ;directory max
    db
          C<sub>0</sub>h
                       ;alloc0
    db
           0
                     ;alloc1
    dw
           0
                     ;check size
                      ;offset
          22
    dw
alv00:
        ds
               007Fh
csv00:
               0000
        ds
```

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

```
; Lesen von Diskette
READ:
        CALL
                 ADRE
        LDIR
READ1:
    0UT
            (READDI), A
    0UT
            (RAMDI), A
    X0R
           Α
    RET
; Schreiben auf Diskette
WRITE:
        CALL
                 ADRE
          DE, HL
    EX
    JR
          READ1-#
; Berechnung Adr.
ADRE:
        0UT
                (RAMEN), A
    0UT
            (READEN), A
    LD
          HL, (TRACK)
    0UT
            (LDAH), L
                          ; hi-adr.
```

2025/12/18 16:26 3/4 Teil 2

```
0UT
           (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
          H,80h
ADREa:
               DE, (DMAAD)
        LD
    LD
          BC,128
    RET
```

Variante 2

Um eine kleinere Blockgröße nutzen zu können, muss die Anzahl der Spuren ≤ 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→ 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 → Adr. f. RAM-Disk ist aufwendiger!

```
;DISKDEF 1,1,16,,1024,252,64,0,3
dpb01:
         dw
               16
                           ;sec per track
           3
                     ;block shift
    db
    db
           7
                     ;block mask
    db
                     ;extnt mask
           0
    dw
           251
                        :disk size-1
    dw
           63
                      ;directory max
           C<sub>0</sub>H
                        ;alloc0
    db
    db
                     :alloc1
           0
           0
                     ;check size
    dw
           3
                     ;offset
    dw
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
```

```
; HL = Track * 10h (SPT)
ADD
       HL, HL
      DE, (SECTOR)
LD
DEC
       DE
                 ; wg. CP/A
ADD
       HL, DE
                   ; HL := HL + Sector
X0R
                ; A = 0, Cy = 0
       Α
RR
      Н
RR
               ; HL := HL/2 ( da 2 Sektoren/Fenster )
              ; L Lo-Bit in A Hi-Bit ( A = 0 oder 80h)
RR
0UT
      (LDAH), L ; hi-adr.
0UT
       (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=1331274891



