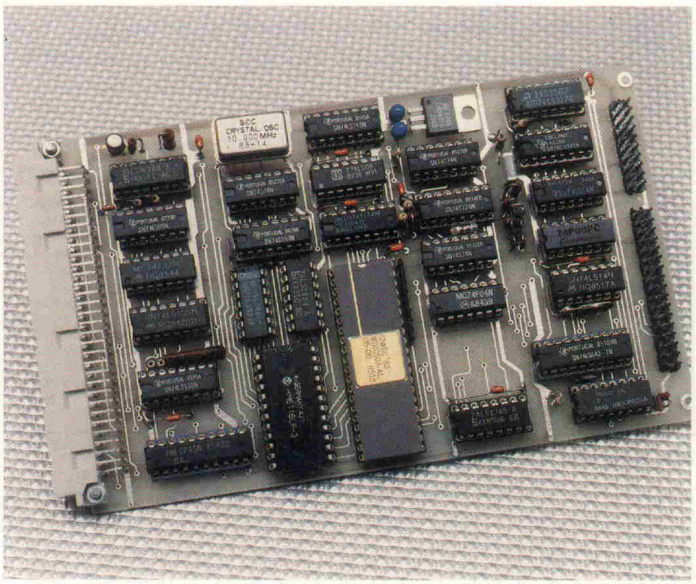

Hart, schnell und sicher

Hard-Disk-Controller mit ST-506-Schnittstelle für ECB-Rechner

Teil 3

Andreas Zippel

Das Opus neigt sich seinem vorläufigen Ende zu. Nach allgemeiner Theorie im ersten, Hardware-Praxis und Testprogrammen im zweiten Akt geht es im vorläufigen Schlußakt um die konkrete Einbindung der Treiber-Software in das Betriebssystem CP/M 3.0, mit ein paar Seitenblicken auf CP/M 2.2. Vorhang auf.

Außer der reinen Treiber-Software gibt es aber noch ein anderes kleines Problem zu lösen, das nicht nur für den Hard-Disk-Liebhaber von Interesse ist: die von Digital Research veröffentlichten Makros zur Erstellung von Disk-Parameter-Blöcken funktionieren nicht unter allen Bedingungen, so daß ein paar Korrekturen angebracht sind.

Teilen und herrschen

Wie schon im ersten Artikel angedeutet, ist die Software eigentlich sehr einfach, einfacher zumindest als etwa Treiber-Software zur Diskettenansteuerung. Aber man kann sehr viel Zeit mit falschen Strategien vertun, was wenig Freude an der Winchester aufkommen lassen würde – von der verplemperten Zeit, das Rad wieder und wieder zu erfinden, ganz zu schweigen.

Zunächst etwas Seltsames. Bisher wurde die ganze Zeit von der Riesenmenge Disk-Speicherplatz geschwärmt, und jetzt wird der Ratschlag folgen, kleine Häppchen daraus zu machen. Warum? Ganz einfach: CP/M kennt keine hierarchischen Inhaltsverzeichnisse, bei MS-DOS etwa auch als Subdirectories bekannt. Damit fehlt CP/M ein Mechanismus, der eine erhebliche Erleichterung für den Anwender darstellt, die Übersicht über seine möglicherweise recht umfangreichen Datenbestände zu behalten.

Bei einer 25-MByte-Platte zum Beispiel sollte man rund 0,6 Prozent für das Directory vorsehen – das sind immerhin fast 5000 Einträge. Da werden selbst gut sortierte Inhaltsverzeichnisse zu Labyrinthen. Auch genügt ein defekter Sektor im Directory – und es heißt Abschied nehmen von den Dateien, deren Einträge in nachfolgenden Sektoren stehen.

Es existiert jedoch eine Möglichkeit, wenigstens etwas Ordnung zu schaffen und doch noch eine gewisse Hierarchie in Form einer Baumstruktur einzuführen. Dazu unterteilt man den Speicherplatz der Winchester in sogenannte Partitionen, was letztlich nichts anderes als 'Teile' bedeutet. Die Hard-Disk wird dabei aus Sicht des Betriebssystems in mehrere logische Laufwerke unterteilt.

Jedes logische Laufwerk hat sein eigenes Directory. Bei der Implementation wird das durch trickreichen Umgang mit der Angabe für die Anzahl der Systemspuren (System-Offset) bewerkstelligt. So besitzt zum Beispiel das erste logische Laufwerk den Offset 1, das zweite den Offset 1667 und das dritte den Offset 3780. Damit liegt auch der Anfang der zugehörigen Directories fest.

Durch diese Methode verfügt man über drei 'Subdirectories' auf derselben Platte, und mit den verbleibenden sieben bis acht MByte pro Inhaltsverzeichnis kann man der Datenflut schon Herr werden. Als weitere Hilfe zur Untergliederung gibt es ja die beim Floppy-Betrieb unter CP/M kaum beachteten USER-Bereiche.

Drei Stufen eines baumstrukturierten Inhaltsverzeichnisses hat man auf diese Art geschaffen: Die 'Wurzel' ist der Benutzer selbst. Er sollte sich merken, welche Partition was enthält. Die nächste Stufe ist dann die Partition. Und als dritte Ebene kann man die USER-Nummer von CP/M heranziehen. Für UNIX-Kenner liest sich das so:

```
/⟨partition⟩/⟨user#⟩/⟨file⟩
```

Schnell? Schnell!

Wie immer wieder angerissen und im ersten Teil der Artikelreihe sehr deutlich dargelegt, muß man seine Optimierungsversuche in Richtung auf höchste Transfer-Geschwindigkeit mit Hinblick auf die jeweilige Anwendung und System-Konfiguration vornehmen. Eine Beispiel-Konfiguration:

- 8 KByte pro Spur
- 16 Sektoren pro Spur
- Sektorlänge 512 Byte
- 6-MHz-DMA mit 7 Zyklen pro Transfer

Der Host-Rechner kann darauf basierend den Sektor-Puffer in rund 600 µs füllen beziehungsweise leeren. Da während dieser Transfer-Zeit der Controller vom Sektor-Puffer abgehängt ist, kann er nicht bereits den nächsten Sektor von der Platte holen oder dorthin zurückschreiben.

Während dieser Zeit laufen knapp 400 Bytes unter dem Kopf der Winchester davon (die Datenrate bei Hard-Disks beträgt üblicherweise 5 MBit/s beziehungsweise 625 KByte/s). Damit läßt sich also nur jeweils jeder zweite Sektor in einer Plattenumdrehung einlesen (beziehungsweise schreiben). Oder anders ausgedrückt: Hier ist eine Platten-Formatierung mit einem Interleave-Faktor von zwei optimal.

Das bedeutet, die 8 KByte einer Spur können in rund 34 ms eingelesen werden, was einer Transfer-Rate von rund 235 KByte/s entspricht. Allerdings immer unter der Voraussetzung, daß das Betriebssystem in der verbleibenden Zeit, in der die restlichen 100 Bytes bis zum nächsten Sektoranfang durchlaufen, alle organisatorischen Probleme gelöst hat, also Sektoranforderungen und ähnliches.

Die ermittelte Datenrate von 235 KByte/s ist damit rund 7,5mal so groß wie bei 5,25-Zoll-Floppies mit doppelter Schreibdichte (250 KBit/s beziehungsweise 31,25 KByte/s). Um möglichst effizient zu arbeiten und das Betriebssystem etwas zu entlasten, empfiehlt es sich, Platz für einen Spurpuffer

```
; Korrigierte Macro Definition für DPBs
; bei Laufwerkskapzitäten viel größer als 300 KByte


        ; dpb    physical$sector$size,    - disk parameter block
        ;       physical$sectors$per$track,
        ;       number$tracks,
        ;       block$size,
        ;       number$dir$entries,
        ;       track$offset,
        ;       checksum$vec$size               (optional)
        ;       ex16                            (optional)


dpb macro ?psize,?pspt,?trks,?bls,?ndirs,?off,?ncks,ex16,?nocheck
    local ?spt,?bsh,?blm,?exm,?dsm,?drm,?al0,?al1,?cks,?psh,?psm
    local ?n
;; physical sector mask and physical sector shift
    ?psh        set 0
    ?n          set ?psize/128
    ?psm        set ?n-1
        rept 8
        ?n      set ?n/2
            if ?n = 0
            exitm
            endif
        ?psh    set ?psh + 1
        endm
    ?spt        set ?pspt*(?psize/128)
    ?bsh        set 3
    ?n          set ?bls/1024
        rept 8
        ?n      set ?n/2
            if ?n = 0
            exitm
            endif
        ?bsh    set ?bsh + 1
        endm
    ?blm        set ?bls/128-1
    if ?psize ) ?bls
      ?n        set ?psize/?bls
      ?dsm      set (((?trks - ?off) * ?pspt) * ?n) - 1
    endif
    if ?psize (= ?bls
      ?n        set ?bls / ?psize
      ?dsm      set (((?trks - ?off) * ?pspt) / ?n) - 1
    endif
;;    ?size     set (?trks-?off)*?spt
;;    ?dsm      set ?size/(?bls/128)-1

    ?exm        set ?bls/1024
        if ?dsm ) 255
            if ?bls = 1024
            exitm
            endif
        ?exm    set ?exm/2
        endif
    ?exm        set ?exm-1
    if not nul ex16
        ?exm    set ex16
    endif
    ?all        set 0
    ?n          set (?ndirs + (?bls/32)-1)/(?bls/32)
;;    ?n                set (?ndirs*32+?bls-1)/?bls
    if ?n = 0
    endif
    if ?n ) 16
    endif
        rept ?n
        ?all    set (?all shr 1) or 8000h
        endm
    ?al0        set high ?all
    ?al1        set low ?all
    ?drm        set ?ndirs-1
    if not nul ?ncks
        ?cks    set ?ncks
    else
        ?cks    set ?ndirs/4
    endif
    if not nul ?nocheck
        ?cks    set ?cks or 8000h
    endif
        dw      ?spt            ; 128 byte records per track
        db      ?bsh,?blm       ; block shift and mask
        db      ?exm            ; extent mask
        dw      ?dsm            ; maximum block number
        dw      ?drm            ; maximum directory entry number
        db      ?al0,?all       ; alloc vector for directory
        dw      ?cks            ; checksum size
        dw      ?off            ; offset for system tracks
        db      ?psh,?psm       ; physical sector size shift
                                ; and mask
    endm


Bild 1.
```

Bild 1. Der Original-Makro, den Digital Research (DRI) in seinen Applikationen zur Berechnung von DPBs vorschlägt, funktioniert nicht sicher bei hohen Plattenkapazitäten. Deshalb hier eine korrigierte Version, die Änderungen gegenüber dem Original sind fett gedruckt.

im Arbeitsspeicher zu spendieren (CP/M 3.0 unterstützt das).

Achtung, Fehler!

Die meisten CP/M-Anpasser kennen und schätzen die von Digital Research angegebenen Makros zum Generieren der verschiedenen Betriebssystemtabellen, so vermutlich auch den Makro 'DPB'. Leider wird man beim Generieren eines Systemes mit den Dimensionen einer Winchester gelegentlich verladen, weil der Makro unter gewissen Umständen versagt. In Bild 1 finden Sie daher eine korrigierte Version, mit deren Hilfe auch das korrekte Generieren von Disk-Parametern für Drives mit mehr als 8 MByte möglich ist.

Allerdings sollte man diesen Makro nicht unbedingt für Disk-Formate mit weniger als 300 KByte einsetzen. Da versagt wiederum die für große Kapazitäten korrigierte Version manchmal. Das liegt nicht etwa an schlecht programmierten Makros, sondern an der verflixten 16-Bit-Arithmetik des RMAC und anderer Assembler. Aber vielleicht findet ja einer unserer Leser den ultimaten Supermakro . . .

Chamäleon

CP/M bietet einen unschätzbaren Vorzug: Man kann es allen möglichen Anforderungen und Wünschen anpassen. Um das aber korrekt über die Bühne zu kriegen, sollte man sich vorab erst einmal in den 'System Alteration Guide' von Digital Research einlesen. Denn all das darzulegen, was bei einer Implementation berücksichtigt werden muß, würde den Rahmen dieses Artikels ganz gewaltig sprengen. Aber es bleiben natürlich ein paar Tips, die man unter Umständen nicht im Handbuch findet. Zunächst für CP/M 3.0.

CP/M Plus

Benutzer dieses Betriebssystems kennen vermutlich bereits dessen modulare Struktur, die von DRI vorgeschlagen wurde. Hier werden dem DPH (Disk Parameter Header) des 'alten' CP/M 2.2 einfach ein paar Bytes

```
        DW      HD$WRITE        ; Schreiben auf Harddisk
        DW      HD$READ         ; Lesen von Harddisk
        DW      HD$LOGIN        ; Ein"loggen" der Harddisk
        DW      HD$INIT0        ; Initialisierung (erfolgt im BOOT)
                                ; fuer die gesamte HD
        DB      0               ; relative LW-Nummer
        DB      0               ; Disk-Modus
HDSK0:  DW      0               ; SKEW-Table (0 fuer Noskew)
        DB      0,0,0,0,0,0,0,0,0,0
                                ; BDOS SCRATCH AREA
        DB      0               ; MEDIA FLAG (0, falls kein
                                ; Plattenwechsel moeglich)
        DW      DPB0            ; DISK PARAMETER BLOCK
        DW      0               ; CHECKSUM VECTOR (0, falls keine
                                ; Checksum-Ueberpruefung)
        DW      0FFFEH          ; ALLOCation VECTOR ALLOCATED
                                ; (wird von GENCPM versorgt)
        DW      0FFFEH,0FFFEH,0FFFEH
                                ; DIRBCB, DTABCB, HASH ALLOC'D
                                ; (ebenfalls von GENCPM gesetzt)
        DB      0               ; HASH BANK
        DW      0               ; CHECKSUM VECTOR

Bild 2.
```

Bild 2. Die Disk-Parameter-Header (DPH) sind bei CP/M 3.0 gegenüber CP/M 2.2 zu sogenannten XDPHs (eXtended DPH) erweitert worden.

vorangestellt. Über den Laufwerksbuchstaben ('A:' bis 'P:') findet das BIOS aus der Drivetable – meist mit dem Label DRVTBL gekennzeichnet – den aktuellen DPH. Bild 2 zeigt nochmals die einzelnen Teile des XDPH, also des erweiterten (eXtended) DPH aus CP/M 2.2.

Ausgehend vom als bekannt vorausgesetzten Standard-DPH

werden die vorangestellten Extension-Bytes 'rückwärts' besprochen. Als erstes ist dann die relative Laufwerksnummer zu erwähnen. Mit ihr ist es möglich, von einem Treiber (so etwas wie dem hier vorgestellten Programm) mehrere Laufwerke bedienen zu lassen. Mit unserer Hard-Disk-Controller-Karte ist jedoch nur ein Laufwerk ansprechbar, wodurch dieser Parameter hier seine Bedeutung einbüßt.

Der nächste Wert, das sogenannte 'Mode Byte', ist für unseren Treiber ebenfalls nicht von Belang, jedoch ist es immer sicherer, auch unbenutzte Bytes definiert auf 00h zu setzen – man weiß ja nie . . .

Alles Routinen

Es folgt die Adresse (Wort) der Init-Routine. Nur ein einziger XDPH zeigt auf eine echte Routine, die anderen führen auf eine einfache Return-Anweisung. (Wenn man den Empfehlungen folgt.) Die Init-Routine stellt nur fest, ob die Winchester anspricht, also im System ist, und merkt sich diese Tatsache.

Das nächste Wort enthält die Adresse einer Login-Routine. Diese wird bei der ersten Selektion eines Laufwerkes nach einem Ctrl-C angesprungen. Sie holt nur den Merker der Init-Routine und meldet die Disk als 'invalid' (ungültig), wenn von der Init-Routine ein Fehler vermerkt wurde.

Es verbleiben die Vektoren für die Read- und Write-Routinen. Diese besetzen den Vektor für Fehlermeldungen vor. Anschließend setzen sie mit Hilfe der Routine UPTASK das Task File des WD1010 (siehe hierzu die vorausgegangenen Teile des Artikels).

Beide Routinen starten dann ihre eigentlichen Aktivitäten, indem sie das entsprechende Kommando an den WD1010 schicken. Die Routine READ wartet ab, bis der Controller fertig ist, und transferiert die Daten mit der Routine HDTRANS1 zum Rechner.

Die Write-Routine wartet auf die Anforderung des Controllers (DRQ-Bit im Status-Register). Sobald diese Anforderung erfolgt, startet WRITE den Transfer der Daten zum Controller mit Hilfe von HDTRANS2. Danach wartet sie, bis der Controller fertig ist. Dieses Warten in der Routine BUSY ist sehr wichtig, da der Controller-Chip während seiner Plattenzugriffe vom Rechner-Bus abgehängt ist und dadurch kein Zugriff auf sein internes Status-Register möglich ist. Während dieser Aktivitäten des Controllers stellt ein per Zusatz-Hardware simuliertes Busy-Bit die einzige definierte Verbindung zum Host-Rechner dar (siehe Hardware-Beschreibung im zweiten Teil des Artikels).

Die eingebaute Schleife für Wiederholungen im Fehlerfalle ist eigentlich unnötig (der Controller besorgt schon einiges davon selbst), dennoch erhöht es ein wenig die Datensicherheit. Wenn Fehler auftreten, münden beide Routinen in eine gemeinsame Fehlerbehandlung. Ansonsten geht es in eine kleine Schlußroutine, die die Winchester deselektiert (bei UPTASK wurde sie ja selektiert). Die Fehlerroutine meldet den jeweiligen Fehler auf dem Bildschirm und haucht ihr Leben dann in der gleichen Schlußroutine aus.

Noch ein Wort zum Banking, einer Methode von CP/M 3.0 zur Speicherplatzverwaltung, die CP/M 2.2 nicht unterstützt. Der gesamte Treiber mit Ausnahme der Transferroutine HDTRANS1 und HDTRANS2 kann im 'gebankten' Teil liegen. Bitte beachten Sie aber, daß die Reihenfolge zwischen Laden der DMA-Adresse und dem Bankumschalten nicht vertauscht wird. Manche Systeme verstecken nämlich die DMA-Adresse in einer Bank (und nicht im Common-Bereich). Und mit einer 'weggeswitchten' DMA-Adresse bekommen sie dann meist einen fürchterlichen Anfall von Selbstzerstörung.

Zur Assembler-Terminologie ist vielleicht auch noch ein Hinweis angebracht. Routinen, die mit einem Fragezeichen versehen sind, gelten global und müssen vom Rest des Systems bereitgestellt werden. Im 'System Guide' von DRI ist deren Funktion genau beschrieben, wobei einige Routinen allerdings nur in den Beispieltreibern zu finden sind.

Hard-Disk unter CP/M 2.2?

Im Prinzip ja, konkret ausprobiert haben wir es nicht. Für kreative Tüftler ein paar Denkanstöße, wie eine Änderung des 3.0-Treibers den 2.2-Betrieb ermöglichen sollte.

Zunächst entfallen die Zusätze zu den DPHs. Statt dessen sind einfach die Einsprünge zu den Routinen, die vorher in den DPHs standen, als globale Namen zu vereinbaren. Ein Verteiler vor den schon vorhandenen Floppy-Routinen kann dann dafür sorgen, daß die Unterprogramme für die Hard-Disk-Steuerung angesprungen werden.

Für den bei CP/M 2.2 benötigten Blocking-Deblocking-Algorithmus kann man die zusätzlichen Bytes am Ende der DPBs nutzen. Diese Bytes sollten dann die Sektorlänge in einer Form enthalten, die es einfach macht, die einzelnen logischen Sektoren aus dem physikalischen Sektor herauszuarbeiten.

Abzuraten ist davon, die Sektoren der Einfachheit halber zu 128 Byte Länge zu wählen, da damit Geschwindigkeit und Speicherplatz verlorengehen. Aber wer die Schwierigkeiten im Umgang mit dem Blocking-Deblocking scheut, braucht nicht zu verzweifeln: Der Treiber erlaubt auch Sektorgrößen von 128 Bytes.

Mit diesem vorerst letzten Beitrag ist der Winchester-Controller aber noch nicht aus dem Blickfeld verschwunden. Die nächste Monitorversion für den c't86 wird als eine ihrer Neuerungen die ganze Treiber-Software für die Controller-Karte bereits enthalten. Die neue Monitorversion ist jedoch in ihrer Gesamtheit für eine Freigabe noch nicht ausreichend erprobt, so daß c't86-Besitzer noch ein paar Hefte Geduld aufbringen müssen.

```
        PAGE 70

        TITLE 'WD1010 WINCHESTER CONTROLLER DRIVER [08.08.85] fuer PROF-80'

        ;   CP/M-80 VERSION 3     --  MODULAR BIOS
        ;
        ;   Version 0.03 (C) by A.Zippel 29.4.86
        ;   Deutsche Kommentierung (P) by A.Zinser 12.6.86
        ;
        ;   Alle verwendeten Parameter sind fuer eine Winchester
        ;   MicroScience HH-725, 4 Koepfe, ausgelegt

        ; Laufwerk-Tabellen (fuer BIOS-I/O & DRVTBL.ASM)

        PUBLIC  HDSK0,HDSK1,HDSK2

        ; verwendete/benoetigte Parameter aus dem BDOS/BIOS

        EXTRN   @ADRV,@rdrv             ; absolute LW-Nummer
        EXTRN   @DMA                    ; DMA-Adresse
        EXTRN   @TRK,@SECT              ; Sektor-Adresse
        EXTRN   @DBNK,@CBNK             ; Daten- und Common-Bank-Nummern

        ; SYSTEM CONTROL BLOCK VARIABLES

        EXTRN   @ERMDE                  ; BDOS Fehlermeldungs-Modus

        ; Utility-Routinen aus dem Standard-BIOS

        EXTRN   ?WBOOT                  ; Warm-Boot-Adresse
        EXTRN   ?PMSG                   ; Print Message
        EXTRN   ?PDERR                  ; Print BIOS DISK ERROR HEADER
        EXTRN   ?BANK                   ; BANK SELECT-Routine

        ; CP/M 3 DISK DEFINITION MACROS

        MACLIB CPM3N                    ; modifiziertes Makro fuer DPB

        ; Z80 MACRO LIBRARY INSTRUCTION DEFINITIONS

        MACLIB Z80

        ; Port-Adressen der Hard-Disk-Karte/Controller
        ; In der c't-86-Version wird Bit 0 durch Bit 3 ersetzt
                ct86    equ     7
                ecb80   equ     0
        ; Dadurch gibt es bei den betroffenen Adressleitungen einen
        ; Offset von 7.

jumper  equ     ct86

hdcbase equ     0E0h                    ; basis addresse des controllers
hdcdat  equ     hdcbase                 ; datenregister
hdcerr  equ     hdcbase+1+jumper        ; (r) error register
hdcwpc  equ     hdcbase+1+jumper        ; (w) write-precompensation start
hdcscn  equ     hdcbase+2               ; sector-zaeher fuer format
hdcsek  equ     hdcbase+3+jumper        ; sector register
hdccyl  equ     hdcbase+4               ; low byte des cylinders
hdccyh  equ     hdcbase+5+jumper        ; high byte des cylinders
```

```
hdcsdh  equ     hdcbase+6               ; sdh-register
hdccmd  equ     hdcbase+7+jumper        ; (w) command register
hdcsta  equ     hdcbase+7+jumper        ; (r) status register

        ; Allgemeine EQUals
CR      EQU     0dh
LF      EQU     0ah
BELL    EQU     07h

        ; Controller-Kommandos

c$rest  EQU     10H                     ; Restore Drive (Recalibrieren)
c$read  EQU     20H                     ; Lesekommando (mit Retry)
c$write EQU     30H                     ; Schreibkommando (mit Retry)

;*****************************************************************************

        ; physikalische Definitionen der Harddisk. Diese Parameter mueesen an das    ***
        ; gewaehlte Format und Laufwerk angepasst werden.                             ***
                                        ;                                           ***
WPVALUE EQU     32                      ; Begin der Precompensation (s.Text)        ***
SECSIZ  EQU     1024                    ; Sektorgroesse [Bytes]                     ***
secmask EQU     0100$0000B              ; Codierung der Sektorgroesse (s.Text)      ***
SECPTRK EQU     9                       ; Sektoren pro Spur und Kopf                ***
heads   EQU     4                       ; Anzahl Koepfe (muss fuer diese
                                        ;   Implementation unbedingt ein            *
                                        ;   Vielfaches von 2 sein)                  ***
cylIND  EQU     612                     ; Anzahl Zylinder                           *

;*****************************************************************************

        ; STATUSBIT MASKEN (siehe auch c't 8/86)

DRQFLG  EQU     08H                     ; Data Request
BUSYFLG EQU     80H                     ; Controller Busy
ERRFLG  EQU     01H                     ; Error
RDYFLG  EQU     40H                     ; DRIVE READY

        ; Select-Deselect-Maskierung (1=select)

selmask equ     00h                     ; LW Selektieren
desmask equ     00h                     ; und deselektieren

        PAGE

        ; EXTENDED DISK PARAMETER HEADERS (XPDHS)
        ; fuer jedes logische Laufwerk getrennt,
        ; (Partition = log. LW)

        dseg

        DW      HD$WRITE                ; Schreiben auf Harddisk
        DW      HD$READ                 ; Lesen von Harddisk
        DW      HD$LOGIN                ; Ein"loggen" der Harddisk
        DW      HD$INIT0                ; Initialisierung (erfolgt im BOOT) fuer die gesamte HD
        DB      0                       ; relative LW-Nummer
        DB      0                       ; Disk-Modus
HDSK0:  DW      0                       ; SKEW-Table (0 fuer Noskew)
        DB      0,0,0,0,0,0,0,0,0       ; BDOS SCRATCH AREA
        DB      0                       ; MEDIA FLAG (0, falls kein Plattenwechsel moeglich)
        DW      DPB0                    ; DISK PARAMETER BLOCK
        DW      0                       ; CHECKSUM VECTOR (0, falls keine Checksumueberpruefung)
        DW      0FFFEH                  ; ALLOCation VECTOR ALLOCATED (wird von GENCPM versorgt)
        DW      0FFFEH,0FFFEH,0FFFEH    ; DIRBCB, DTABCB, HASH ALLOC'D (ebenfalls von GENCPM gesetzt)
        DB      0                       ; HASH BANK
        DW      0                       ; CHECKSUM VECTOR

        DW      HD$WRITE                ; Schreiben auf Harddisk
        DW      HD$READ                 ; Lesen von Harddisk
        DW      HD$LOGIN                ; Ein"loggen" der Harddisk
        DW      HD$INIT1                ; Initialisierung erfolgte bereits mit INIT$0
        DB      0                       ; relative LW-Nummer
        DB      0                       ; Disk-Modus
HDSK1:  DW      0                       ; SKEW-Table (0 fuer Noskew)
        DB      0,0,0,0,0,0,0,0,0       ; BDOS SCRATCH AREA
        DB      0                       ; MEDIA FLAG (0, falls kein Plattenwechsel moeglich)
        DW      DPB1                    ; DISK PARAMETER BLOCK
        DW      0                       ; CHECKSUM VECTOR (0, falls keine Checksumueberpruefung)
        DW      0FFFEH                  ; ALLOCation VECTOR ALLOCATED (wird von GENCPM versorgt)
        DW      0FFFEH,0FFFEH,0FFFEH    ; DIRBCB, DTABCB, HASH ALLOC'D (ebenfalls von GENCPM gesetzt)
        DB      0                       ; HASH BANK
        DW      0                       ; CHECKSUM VECTOR

        DW      HD$WRITE                ; Schreiben auf Harddisk
        DW      HD$READ                 ; Lesen von Harddisk
        DW      HD$LOGIN                ; Ein"loggen" der Harddisk
        DW      HD$INIT1                ; Initialisierung erfolgte bereits mit INIT$0
        DB      0                       ; relative LW-Nummer
        DB      0                       ; Disk-Modus
HDSK2:  DW      0                       ; SKEW-Table (0 fuer Noskew)
        DB      0,0,0,0,0,0,0,0,0       ; BDOS SCRATCH AREA
        DB      0                       ; MEDIA FLAG (0, falls kein Plattenwechsel moeglich)
        DW      DPB2                    ; DISK PARAMETER BLOCK
        DW      0                       ; CHECKSUM VECTOR (0, falls keine Checksumueberpruefung)
        DW      0FFFEH                  ; ALLOCation VECTOR ALLOCATED (wird von GENCPM versorgt)
        DW      0FFFEH,0FFFEH,0FFFEH    ; DIRBCB, DTABCB, HASH ALLOC'D (ebenfalls von GENCPM gesetzt)
        DB      0                       ; HASH BANK
        DW      0                       ; CHECKSUM VECTOR

        CSEG

        ; Die Disk-Parameter-Bloecke muessen entweder mit dem
        ; modifizierten DPB-Makro CPM3N.LIB oder ebenfalls per
        ; Hand erstellt und an das jeweilige LW/Format
        ; angepasst werden.

DPB0    ;DPB 1024,9,777,4096,2048,1,0
        DW      72                      ; 128 BYTE RECORDS PER TRACK
        DB      5,1FH                   ; BLOCK SHIFT AND MASK
        DB      1                       ; EXTENT MASK
        DW      1745                    ; MAXIMUM BLOCK NUMBER
        DW      7FFH                    ; MAXIMUM DIRECTORY ENTRY NUMBER
        DW      0FFFFH                  ; ALLOC VECTOR FOR DIRECTORY
        DW      8000H                   ; CHECKSUM SIZE
        DW      1                       ; OFFSET FOR SYSTEM TRACKS
        DB      3,7                     ; PHYSICAL SECTOR SIZE SHIFT

DPB1    ;DPB 1024,9,1665,4096,2048,778,0
        DW      72                      ; 128 BYTE RECORDS PER TRACK
        DB      5,1FH                   ; BLOCK SHIFT AND MASK
        DB      1                       ; EXTENT MASK
        DW      1995                    ; MAXIMUM BLOCK NUMBER
        DW      7FFH                    ; MAXIMUM DIRECTORY ENTRY NUMBER
        DW      0FFFFH                  ; ALLOC VECTOR FOR DIRECTORY
        DW      8000H                   ; CHECKSUM SIZE
        DW      778                     ; OFFSET FOR SYSTEM TRACKS
        DB      3,7                     ; PHYSICAL SECTOR SIZE SHIFT

DPB2    ;DPB 1024,9,2451,4096,2048,1666,0
        DW      72                      ; 128 BYTE RECORDS PER TRACK
        DB      5,1FH                   ; BLOCK SHIFT AND MASK
        DB      1                       ; EXTENT MASK
        DW      1765                    ; MAXIMUM BLOCK NUMBER
        DW      7FFH                    ; MAXIMUM DIRECTORY ENTRY NUMBER
        DW      0FFFFH                  ; ALLOC VECTOR FOR DIRECTORY
        DW      8000H                   ; CHECKSUM SIZE
        DW      1666                    ; OFFSET FOR SYSTEM TRACKS
        DB      3,7                     ; PHYSIKAL SECTOR SIZE AND SHIFT
        MACE

        DSEG

        ; Hard-Disk-I/O-Routinen
        ; ----------------------

        ; Initialisierung, wird im BOOT vorgenommen
        ; Bei der Initialisierung wird ein RESTORE-DRIVE-Kommando abgesetzt
        ; und in Abhaengigkeit der Controller-(Fehler-)Meldungen
        ; ein Fehler-Bit gesetzt.

HD$INIT0:
        CALL    RESTORE                 ; RESTORE absetzen
        LXI     H,RESTERROR             ; Fehlermeldungstext
        MVI     A,0FFH                  ; und Fehler setzen
        jrc     hd$init2                ; FALLS CARRY gesetzt!
        mvi     a,0                     ; ansonsten fehlerflag zuruecksetzen
hd$init2:
        STA     HD$BLOCKED              ;
        CC      ?PMSG                   ; Nur Meldung, falls Fehler
        MVI     A,0C9H
        STA     HD$INIT0                ; Weitere Initialisierungen unterbinden
hd$init1:
        jp      hd$exit                 ; Deselect Drive

        ; LOGIN. Diese Routine wird zur Bestimmung des
        ; Diskettenformates verwendet. Ausserdem werden mit ihrer
        ; Hilfe die entsprechenden Parameter im XDPH erstellt. Da
        ; auf der Harddisk kein Formatwechsel erfolgen kann und der
        ; XDPH schon fertig vorliegt, erfolgt nur die Pruefung, ob
        ; auf die Disk zugegriffen werden kann (das Fehlerflag wird
        ; bereits von INIT gesetzt). Im Fehlerfall wird das Carry
        ; gesetzt und der aufrufenden BIOS-Routine damit der Fehler
        ; signalisiert.

HD$LOGIN:
        LDA     HD$BLOCKED
        ANA     A
        RZ                              ; OK, DISK AVAILABLE
        STC
        RET                             ; DISK NOT AVAILABLE

        PAGE

        ; Lese-/Schreib-Routinen
        ; ----------------------

        ; Die folgenden beiden Routinen werden mit folgenden
        ; Parametern aufgerufen:
        ; @dma    (16)  DMA-Adresse
        ; @dbnk   (8)   Banknummer des Datenbereiches
        ; @cbnk   (8)   Banknummer des Common-Bereiches
        ; @trk    (16)  Spurnummer
        ; @sect   (16)  Sektornummer

        ; Lesen von der Hard-Disk

HD$READ:
        MVI     A,2                     ; Anzahl Wiederholungen im Fehlerfall
HD$READ1:
```

```
        PUSH    PSW
        LXI     H,READ$MSG              ; Speichere Information ueber die Zugriffsart fuer
        SHLD    OPERATION$NAME          ; eine detailierte Fehlermeldung ab
        CALL    UPTASK                  ; Setze TASK-Register
        MVI     A,c$read
        OUT     HDCCMD                  ; Beginne Leseoperation
        CALL    BUSY                    ; Warte, bis Lesen zu Ende
        JrC     RD$ERROR                ; Falls C-Fleg gesetzt, trat Fehler auf
        POP     PSW
        CALL    WAITDRQ                 ; Ansonsten warte auf DATA-REQUEST
        CALL    HDTRANS1                ; Uebertrage die Daten (Sektorpuffer => (@DMA))
        JMP     HD$EXIT
RD$ERROR:
        CALL    CLEARDRQ                ; Loesche anliegende DATA-REQUESTs
        POP     PSW
        DCR     A                       ; noch ein Versuch ?
        JrNZ    HD$READ1                ; JMP, falls ja
        Jr      wr$error1               ; sonst Fehler behandeln

        ; Schreibe auf Harddisk (Ablauf aequivalent zu LESEN)

HD$WRITE:
        MVI     A,2
HD$WRITE1:
        PUSH    PSW
        LXI     H,WRITE$MSG
        SHLD    OPERATION$NAME
        CALL    UPTASK
        MVI     A,c$write
        OUT     HDCCMD
        CALL    WAITDRQ
        CALL    HDTRANS2                ; Uebertrage Daten in den Sektorpuffer
        CALL    BUSY
        JrC     WR$ERROR
        POP     PSW
        JMP     HD$EXIT
WR$ERROR:
        POP     PSW
        DCR     A
        JrNZ    HD$WRITE1
wr$error1:
        JMP     HD$ERROR

        ; Setzen des TASK-Register
        ; Dem BIOS/BDOS praesentiert sich die Harddisk als
        ; "Disklaufwerk" mit einem Kopf und einer entsprechend
        ; grossen Anzahl Spuren (s. auch Text). Daher muss
        ; in der folgenden Routine die physikalische Sektor-Kopf-Spur-
        ; information aus der logischen Sektor-Spur-Information
        ; gewonnen weden. (Dabei kann fuer alle Partitionen die
        ; gleiche Routine verwendet werden, da sie fortlaufende
        ; Spurnummern verwenden. Die erste Spur der zweiten
        ; Partition ist die auf die letzte Spur der ersten Partition
        ; folgende. Dies wurde durch einen entsprechenden
        ; SYSTEM-TRACK-OFFSET erreicht (siehe DPB)).
        ; Man kann alerdings die Selektion der verschiedenen
        ; Partitionen auch ueber @rdrv, relative drive number, die im
        ; dph angegeben ist, vornehmen. In diesem Fall besteht auch
        ; eine einfache moeglichkeit, mit z.B. 6-Koepfern zu arbeiten

UPTASK:
        CALL    CLEARDRQ                ; RESTLICHE DRQ'S ABHOLEN (WENN DA)

        ;---------------- SEKTOR

        LDA     @SECT                   ; HOLE SEKTORNUMMER
        OUT     HDCSEK                  ; SETZE REGISTER

        ;---------------- SDH-REGISTER
        ;
        ;Die unteren Bits der logischen Spurnummer repraesentieren
        ;die physikalische Kopfnummer (2 Koepfe - 1 Bit, 4 Koepfe - 2 Bit etc.)
        ;Die Kopfadresse wird zusammen mit den S = Sektor-Size- und D = Drive-
        ;select-Informationen auf das SDH-Register geschrieben

        LDA     @TRK                    ; Die unteren 1/2/3 Bit der Spur
        ANI     heads-1                 ; repraesentieren die Kopfnummer.
        ORI     secmask or selmask      ; SEKTORLAENGE (maskiert) und Select
        OUT     HDCSDH                  ; dazu und REGISTER SETZEN

        ;---------------- Zylinder
        ;
        ;Da die unteren Bytes der log.Spur den Kopf darstellen,
        ;muessen sie aus der log.Spur-Information entfernt werden,
        ;um die phys.Spur zu erhalten. Dieser 10-Bit-Wert wird in die
        ;entsprechenden WD-Register geschrieben

        LDA     @TRK                    ; LOW BYTE DER SPUR
        MOV     B,A
        LDA     @TRK+1                  ; HIGH BYTE DER SPUR
        SRLR    A                       ; Kopfinformation herausrotieren
        RARR    B                       ; Hi(@TRK)/2
        if heads)2
        SRLR    A
        RARR    B                       ; HI(@TRK)/4
        IF heads)4
        SRLR    A
        RARR    B                       ; HI(@TRK)/8
        ENDIF
        ENDIF
        OUT     HDCCYH                  ;SETZEN DES HIGH BYTES
        MOV     A,B
        OUT     HDCCYL                  ;SETZEN DES LOW BYTES
        RET

        ; Loeschen anliegender DATA-Requests

CLEARDRQ:
        IN      HDCSTA
        ANI     DRQFLG
        RZ
        IN      HDCDAT                  ;DUMMY READ
        jr      CLEARDRQ

        ; Warte, bis DRQ aktiv wird

WAITDRQ:
        IN      HDCSTA
        ANI     DRQFLG
        RNZ
        jr      WAITDRQ

        ; Warte, bis Kommando fertig abgearbeitet ist

BUSY:   IN      HDCSTA
        ANA     A
        JM      BUSY                    ; NEGATIV -) NOCH BUSY
        RAR                             ; ERROR BIT IN CARRY
        RET
        ; CONTROLLER- UND LAUFWERKS-INITIALISIERUNG.
        ; Es wird geprueft, ob der Controller "frei" ist (kein busy); ist
        ; dieses der Fall, wird das RESTORE-Kommando abgesetzt und
        ; auf korrekte Abarbeitung (kein Busy, kein Error) gewartet;
        ; geschieht dies nicht innerhalb einer angemessenen Frist,
        ; wird das Laufwerk als fehlerhaft zurueckgemeldet. Bei
        ; langsam "hochfahrenden" laufwerken ist ggf. der Zaehler
        ; innerhalb der WAIT-Routine zu erhoehen.

RESTORE:
        mvi     a,selmask               ; Selektiere LW
        OUT     HDCSDH
        lxi     h,-1                    ; timeout-zaehler auf 0
restore0:
        in      hdcsta                  ; Laufwerk busy ?
        ana     a                       ; negativ -) noch busy
        jm      restore4                ; ueberspringe restore-anforderung, da
        CALL    CLEARDRQ                ;       Controller nicht bereit ist
        MVI     A,c$rest
        OUT     HDCCMD                  ; RESTORE COMMAND
restore2:
        in      hdcsta
        ana     a
        jm      restore4                ; laufwerk noch busy
        rar                             ; falls nicht busy und kein fehler:
        JrNC    RESTORE1                ; OK, DISK MELDET SICH FERTIG
restore4:
        CALL    WAIT
        LXI     D,1
        DAD     D                       ; TIMEOUT TESTEN
        JrNC    restore3                ; NOCH NICHT, WEITER VERSUCHEN
        RET                             ; FEHLER, (DISK NICHT BEREIT), CY=1
RESTORE1:
        MVI     A,WPVALUE               ; START TRACK FOR PRECOMPENSATION
        OUT     HDCWPC
        xra     a
        ret                             ; CY=0 (kein Fehler)

        ; Warteschleife (fuer RESTORE)

WAIT:   PUSH    PSW
        PUSH    H
        LXI     H,8000H
wait1:  dcx     h
        mov     a,h
        ora     l
        JrNz    WAIT1
        POP     H
        POP     PSW
        RET

        cseg

        ; Daten-Uebertragungs-Routinen
        ; ----------------------------

        ; Einlesen der Daten in den DMA-Puffer
        ; WICHTIG: Die sector-groesse muss fuer diese routine
        ; groesser 128 Byte/Sector sein, allerdings ist dieses
        ; format aus anderen Gruenden auch nicht zu empfehlen (s.Text)

HDTRANS1:
        LHLD    @DMA                    ; AUF DMA PUFFER ZEIGEN
        DI                              ; Interrupts verbieten
        LDA     @DBNK
        CALL    ?BANK                   ; BANK FUER DATEN SELECTIEREN
        lxi     b,hdcdat                ; b = 0, c = Datenregister
        rept secsiz/256
        inir
        endm
        jr      hdtrans3                ; Ende der Datenuebertragung
```

```
        ; Schreiben der Daten in den Controller-Puffer
HDTRANS2:
        LHLD    @DMA                    ; AUF DMA PUFFER ZEIGEN
        DI
        LDA     @DBNK
        CALL    ?BANK                   ; BANK FUER DATEN SELECTIEREN
        lxi     b,hdcdat                ; b = 0, c = Datenregister
        rept secsiz/256
        outir
        endm
HDtrans3:
        LDA     @CBNK
        CALL    ?BANK                   ; PROGRAMMBANK SELECTIEREN
        EI
        RET

        DSEG
        ; Fehlerbehandlung
        ; ----------------
        ; Im Fehlerfall wird, sofern das @ERMDE-Flag entsprechend
        ; gesetzt ist, eine ausfuehrliche Fehlermeldung in der Form:
        ; BIOS ERR ON D: T=NN, S=MM (OPERATION) (TYPE), RETRY ?
        ; ausgegeben. Die angabe (TYPE) enthaelt alle im Fehler-Byte
        ; des Controllers markierten Errors in Klartext.
HD$ERROR:
        LDA     @ERMDE
        CPI     0FFH
        JZ      HARD$ERROR              ; Keine ausfuehrliche Fehlermeldung
        CALL    ?PDERR                  ; PRINT MESSAGE HEADER
        LHLD    OPERATION$NAME          ; wird von READ/WRITE gesetzt
        CALL    ?PMSG                   ; LAST FUNCTION
        IN      HDCERR                  ; Hole Fehler-Byte vom Controller
        LXI     H,ERROR$TABLE           ; Zeige auf Meldungstabelle
ERRM1:
        MOV     E,M
        INX     H
        MOV     D,M
        INX     H                       ; GET NEXT MESSAGE ADDRESS
        ADD     A
        PUSH    PSW                     ; rotiere nach links und sichere
        PUSH    H                       ;       die restlichen Bits
        XCHG
        CC      ?PMSG                   ; falls Bit gesetzt war, gebe
        XCHG                            ;       Meldung aus
        POP     H
        POP     PSW
        JNZ     ERRM1                   ; Falls noch mehr Bits, rotiere...
HARD$ERROR:
        CALL    HD$EXIT                 ; Deselect drive
        MVI     A,1                     ; setze A=1 fuer BIOS/BDOS (Error occured)
HARD1$ERROR:
        RET                             ; RETURN HARD ERROR TO BDOS
HD$EXIT:
        MVI     A,desmask
        OUT     HDCSDH                  ; DESELECT DRIVE
        XRA     A                       ; NO ERROR
        RET
        PAGE

        CSEG

        ; Hilfsspeicherstellen, Tabellen & Messages ...
        ; ---------------------------------------------
HD$BLOCKED      DB      0       ; HARDDISK NICHT ANSPRECHBAR WENN () 0

        DSEG

READ$MSG        DB      ', READ',0
WRITE$MSG       DB      ', WRITE',0

OPERATION$NAME  DW      READ$MSG

        ; Tabelle der Zeigr auf die detailierten Fehlermeldungen
        ; Jede Adresse entspricht einem Fehlerbit des WD1010
        ; Start ist Bit 7.

ERROR$TABLE     DW      B7$MSG
                DW      B6$MSG
                DW      B5$MSG
                DW      B4$MSG
                DW      B3$MSG
                DW      B2$MSG
                DW      B1$MSG
                DW      B0$MSG

B7$MSG          DB      ', BAD BLOCK',0
B6$MSG          DB      ', CRC,',0
B5$MSG          DB      ' ',0
B4$MSG          DB      ', REC NOT FOUND',0
B3$MSG          DB      ' ',0
B2$MSG          DB      ', ABORTED',0
B1$MSG          DB      ', NO TR000',0
B0$MSG          DB      ', NO DAM',0

RESTERROR       DB      bell,'** Winchester Drive NOT AVAILABLE **',cr,lf,0

        END
```

Das Treiber-Programm für die Hard-Disk-Steuerung zur Einbindung in CP/M 3.0. c't