Frank Oettle und Thomas Reichler

Der mc-CP/M-Plus-Computer

Teil 1: Das Betriebssystem

Wer kennt es nicht, das bekannte CP/M-2.2, das nach wie vor meist genutzte Betriebssystem. Digital Research hat nun eine völlig neu überarbeitete Version, das CP/M-Plus, Ver. 3.0 herausgebracht, das die Vorgänger-Version 2.2 ablösen soll. mc wird in den folgenden Ausgaben einen professionellen CP/M-Computer zum Nachbau vorstellen, auf dem das neue Betriebssystem CP/M-Plus, Ver. 3.0 implementiert ist. Die Entwickler von CP/M-Plus geben an, daß diese neue Version bis zu viermal mehr leistet als CP/M-2.2. Doch wo liegen nun die wesentlichen Verbesserungen von CP/M-Plus gegenüber seiner Vorgängerversion 2.2? Das soll in der folgenden Einführung zum mc-CP/M-Plus-Computer ergänzend zu mc 8/1984 (S. 44) geklärt werden.

Wie schon der große Abstand der CP/M-Version-Nummern „2.2" und „3.0" andeutet, handelt es sich bei CP/M-Plus nicht nur um ein etwas umgekrempeltes CP/M-2.2-System mit einigen Verbesserungen in Details. Es wurde ein völlig neues Betriebssystem geschaffen. Die Schwächen der Vorgängerversion sind jetzt auf sehr elegante Weise aus der Welt geschafft worden. Dennoch gewährleistet CP/M-Plus volle Aufwärtsverträglichkeit für Programme, die unter CP/M-2.2 und MP/M-II-Systemen erstellt wurden. Anwenderprogramme die unter CP/M-2.2 laufen, können also ohne Änderung unter CP/M-Plus eingesetzt werden. Somit steht für jeden CP/M-Plus-Benutzer auch weiterhin das größte Softwareangebot für Mikrocomputer bereit. Eine Ausnahme bilden, auf Grund einer neuartigen Bank-Umschaltung, Programme, die direkt das BIOS des Betriebssystems ansprechen. Doch derartige Programme sind Gott sei Dank selten anzutreffen.

Die Anforderungen an ein modernes Betriebssystem

Ein Betriebssystem kann man grob an zwei Kriterien messen: An seiner Leistungsfähigkeit und an seinem Bedienungskomfort. Zur Leistungsfähigkeit gehört zum Beispiel wie schnell Diskettenoperationen durchgeführt werden können, wieviele I/O-Geräte bedient werden können oder wieviel Rechenarbeit das Betriebssystem durch entsprechend standardisierte Unterprogramme dem aufrufenden Anwenderprogramm abnehmen kann. In Bezug auf den Bedienungskomfort hat jeder CP/M-2.2-Anwender wohl schon die äußerst aussagekräftigen „BDOS Error on A..."-Meldungen mit nachfolgendem Programmabbruch verwünscht. Oder die spartanischen Dienstprogramme, die einer Vielzahl von Utility-Programmpaketen zu starker Nachfrage verholfen hatten. Doch um eben diese „Leistungsfähigkeit" eines Betriebssystems zu steigern, muß man eine ganze Menge wertvollen Speicherplatz opfern und mit Systemroutinen belegen. Und genau das übersehen die Kritiker des CP/M-2.2-Systems oft. CP/M-2.2 ist also ein kleines Betriebssystem großer Leistung. CP/M-Plus ist ein großes Betriebssystem mit Komfort.

16 Bit kontra 8 Bit

Wertvoll ist Speicherplatz heutzutage in Mikrocomputern nicht deshalb, weil die Speicherchips noch sehr teuer wären, sondern weil der Adreßraum eines 8-Bit-Prozessors in der Regel auf 64 KByte beschränkt ist. Man stelle sich ein Betriebssystem vor, das ausführlichste Fehlermeldungen hervorzaubert (die dabei immer noch genauso ärgerlich sind), dafür aber die Hälfte des verfügbaren Speicherplatzes in Anspruch nimmt. Dann bleiben für Anwenderprogramme gerade noch lächerliche 32 KByte Platz. Jeder erfahrene Programmierer weiß, daß dies viel zu wenig ist. Nun stellt sich sofort die Frage, „weshalb nicht auf einen 16-Bit-Prozessor ausweichen", dessen Adreßraum einige MByte beträgt. Zwei gute Gründe können dagegen sprechen: Der Hardwareaufwand und damit auch die Kosten eines echten 16-Bit-Rechners sind beträchtlich. Zudem sind die Geschwindigkeitsvorteile im Vergleich zu modernen 8-Bit-Computern nicht immer groß. Ein starkes Entscheidungskriterium liefert auch das Softwareproblem: Wer will schon seine in mühevoller Kleinarbeit erstellten 8-Bit-Rechner-Programme in den Papierkorb werfen, nur weil sie auf dem neuen Prozessor nicht mehr laufen. Wer will schon auf das reichhaltige und damit kostengünstige Angebot an fertiger CP/M-Software verzichten?

Die CP/M-Plus-Philosophie

Genau diese Diskrepanz zwischen Leistungsfähigkeit und Bedienungskomfort einerseits, Kompatibilität und geringem Speicherbedarf andererseits überbrückt das neue Betriebssystem. Durch eine neuartige Bankumschaltungs-Technik wird das Problem des begrenzten Adreßraumes bei 8-Bit-Prozessoren auf sehr elegante Weise gelöst. CP/M-Plus ist in der Lage, mehrere Speicherbänke zu adressieren und zu verwalten, so daß auch ein Z80-System bis zu 1 MByte RAM-Speicher verwalten kann. Außerdem besitzt CP/M-Plus eine Zeit- und Datumsmarkierung, Fehlersuche und Fehlerkorrektur, automatisches Disk-Login, „hashed" Directory-Zugriff, Record-Pufferung und Multi-Sektor-Ein-Ausgabe. CP/M-Plus kann bis zu 16 einzelne RAM-Arbeitsspeicherbereiche (Bänke) und bis zu 16 Disketten-Plattenspeichereinheiten von jeweils 512 MByte Kapazität verwalten. Besonders Anwender, die Winchester-Laufwerke einsetzen, werden sich über diese Neuerung freuen, da unter CP/M-2.2 bisher nur Laufwerke mit einer Maximalkapazität von 8 MByte verwaltet werden konnten. Als logische Ein/Ausgabe-Einheiten werden bis zu 16 verschiedene Einheiten verwaltet. Nicht zu vergessen die hilfreiche Unterstützung durch zahlreiche Dienstprogramme, die die tägliche Arbeit mit dem Computer sehr erleichtern.

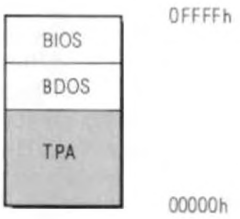


Bild 1. Die Speicheraufteilung eines CP/M-Plus-Systems ohne Bankumschaltung

Verwaltet bis zu 1 MByte Systemspeicher durch „Banking"

CP/M-Plus wendet eine interessante neue Technik der Speicherverwaltung an, die Bankumschaltung. Der Adreßraum des Prozessors wird (durch entsprechende Hardware-Unterstützung) in bis zu 16 verschiedene 64-KByte-Bänke unterteilt. Das BDOS schaltet je nach Bedarf die richtige Bank in den Adreßraum des Prozessors. Wichtig dabei ist, daß ein Teil des Adreßraums von jeder Bank aus erreichbar ist. Dieser „Computer-Memory-Teil", der in jede Speicherbank eingeblendet ist, belegt je nach Hardware die obersten 4...16 KByte jeder Bank. Von diesem Common-Bereich aus geschieht die Adreßumschaltung der einzelnen Bänke und werden die Betriebssystem-Aufrufe getätigt.

Nonbanked CP/M-Plus

Findet das Banking keine Hardwareunterstützung, so kann CP/M Plus auch als „nonbanked"-Version betrieben werden. Der Speicherbereich besteht dann aus einer einzigen, durchgehend adressierbaren 64-KByte-Bank. Im oberen Bereich der Bank ist, wie unter CP/M-2.2, das BDOS und BIOS eingelagert (*Bild 1*). Natürlich beanspruchen beide Teile mehr Speicherplatz als unter CP/M-2.2. Daher wurden verschiedene Funktionen der nonbanked-CP/M-Plus-Version gestrichen, um den TPA-Bereich nicht zu stark herabzusetzen (z. B. kein Password-Schutz möglich, eingeschränkte Editiermöglichkeiten wie unter CP/M 2.2...).

Banked CP/M-Plus

In der gebankten Version gliedert sich das CP/M-Betriebssystem in zwei Teile: Dem residenten Teil und dem gebankten Teil (*Bild 2*). Der residente Teil befindet sich im Common Memory und kann von jeder Bank aus angesprochen werden. Er dient zum Aufruf des Betriebssystems, zum Umschalten zwischen den verschiedenen Bänken und zur Abwicklung einfacher Operationen wie „Console In", „Console Out" und so fort. Der gebankte Teil des CP/M-Plus-Systems ist unterhalb des Common Memorys in Bank 0 eingelagert. Für Anwendungsprogramme stehen die gesammte Bank 1 und der noch freie Teil des Common Memory zur Verfügung. In Bild 2 wird die Speicheraufteilung des in den folgenden Ausgaben vorgestellte mc-CP/M-Plus-Computers dargestellt.

Die Speicheraufteilung des mc-CP/M-Plus-Computers

Ein gebanktes CP/M-Plus-System benötigt mindestens 128 KByte RAM-Speicher, also zwei 64-KByte-Bänke. Der Common-Memory-Bereich beträgt 16 KByte. Also: der Bereich von 0C000h bis 0FFFFh ist von jeder Bank aus erreichbar. Der residente Teil von CP/M-Plus beträgt 4 KByte. Er erstreckt sich von 0F000h bis 0FFFFh. Das residente BDOS startet bei 0F000h und ist 1,5 KByte lang. Bei 0F600h startet der residente BIOS-Teil mit einer Länge von 2,5 KByte. Der Rest des Common Memorys von 0C000h bis 0EFFFh zählt zur TPA und ist somit frei für Anwenderprogramme.

Bank 0:

Der gebankte Teil des CP/M-Systems liegt in Bank 0 und startet mit dem BDOS bei 07900h. Das gebankte BDOS belegt 11 KByte Speicherplatz, somit erstreckt sich der gebankte Teil der BIOS von 0A700h (mit einer Länge von 6,25 KByte) bis 0BFFFh, dem Ende der Bank 0. Der Rest der Bank 0 von ca. 31 KByte Länge wird zum Anlegen der Cache-Puffer für Daten und Directorys, Hash-Tabellen, dem Bios-Trackpuffer und zum Zwischenspeicher des CCPs (Console Command Prozessor) verwendet.

TPA-Bank 1:

Die gesamte Bank 1 mit 48 KByte von 00000h bis 0BFFFh ist frei für Anwenderprogramme. Doch zur TPA zählt auch noch der verbleibende Teil des Common Memorys mit 12 KByte. Somit steht dem Benutzer ein TPA-Bereich von 60 KByte zur Verfügung, bei einer Betriebssystemlänge von 21 KByte, die zahlreichen Puffer noch nicht mitgerechnet!

RAM-Floppy-Bank 2...15:

Die restlichen Bänke 2...15 werden zum Betrieb des Systems nicht unbedingt benötigt. Dieser Bereich kann je nach Speicherausbau des mc-CP/M-Plus-Computers dem System zum Anlegen weitere Puffer zugeteilt werden, oder er dient als Speicher einer sehr schnell arbeitenden RAM-Floppy. Dazu wird RAM-Speicher wie ein normales physikalisches Laufwerk angesprochen, aber ohne zeitraubende Seek-, Transfer- und Kopfladezeiten herkömmlicher Laufwerke abwarten zu müssen.

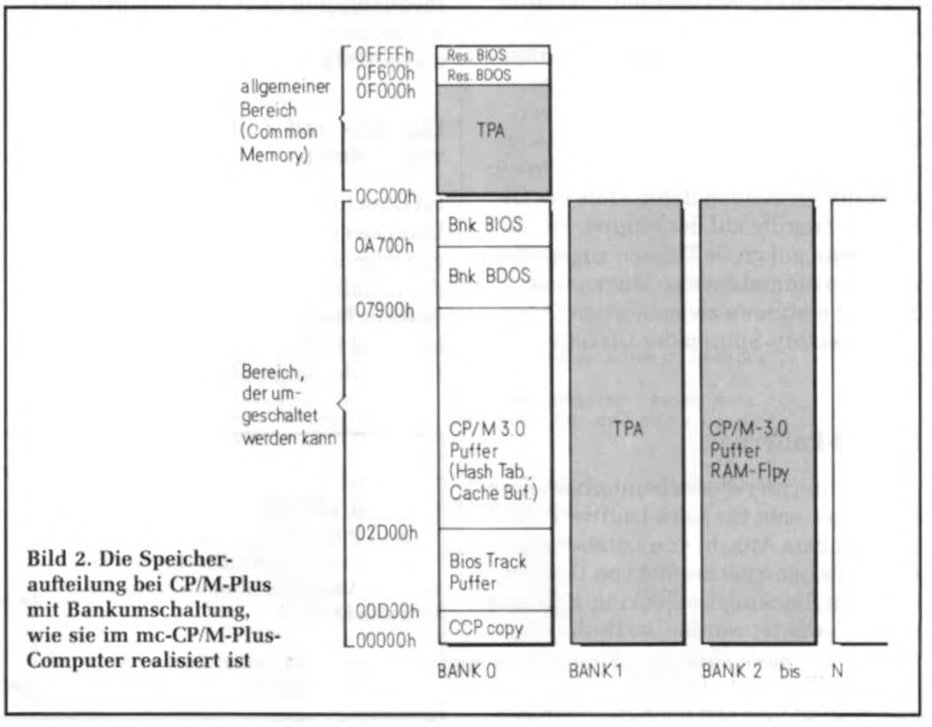


Bild 2. Die Speicheraufteilung bei CP/M-Plus mit Bankumschaltung, wie sie im mc-CP/M-Plus-Computer realisiert ist

Leistungsfähige Dateiverwaltung

Disk-Zugriffe setzen die Geschwindigkeit von Mikrocomputer-Systemen stark herab, wenn sie zahlreich auftreten, da mechanische Laufwerke leider sehr langsam arbeiten. Moderne Laufwerke weisen zwar immer kürzere Zugriffszeiten auf, im Vergleich zu Prozessor-Operationen dauert ein Disk-Transfer aber auch sehr, sehr lang. Ein Ausweg aus dieser Situation wäre eine Geschwindigkeitssteigerung der mechanischen Laufwerke, wie es z. B. Festplatten-Laufwerken mit Übertragungsraten von 10 MBit/s schon vorweisen können. Leider hat diese Lösung Grenzen und ist mit hohen Kosten verbunden. Eine viel elegantere Lösung zeigt CP/M-Plus auf: Files werden im RAM-Bereich zwischengespeichert und so Diskoperationen auf ein absolutes Minimum reduziert und das Gesamttempo deshalb erheblich beschleunigt. Unter CP/M-Plus werden einige völlig neue Techniken der Dateiverwaltung eingeschlagen.

Directory-„Hashing" und „Caching"

Hash-Tabellen verkürzen die Suche nach bestimmten Directory-Einträgen (das Inhaltsverzeichnis einer Diskette) erheblich. Mit Hilfe der Hash-Tabelle kann das BDOS die Stelle (Sektor, Track) des gewünschten Directory-Eintrags aus einem numerischen Index (der bei der Erzeugung des Eintrags gebildet wurde) sofort bestimmen. Dadurch entfällt das Durchsuchen des oft mehrere KByte großen Directorys nach bestimmten Einträgen. Je nach verfügbarem Systemspeicher werden große Teile des Directorys eines jeden Laufwerks in diesen Speicher geladen (Directory Caching), von wo aus sie sehr schnell verarbeitet werden können. Hash-Tabellen und Directory-Puffer vermeiden daher viele der Directory-Zugriffe auf der Floppy, besonders wenn auf große Dateien zugegriffen wird, wo normalerweise langwierige Seek-Operationen zwischen den Daten- und Directory-Spuren der Diskette auftreten.

Record-Pufferung

Neben Directory-Puffern unterhält das BDOS, getrennt für jedes Laufwerk, eine bestimmbare Anzahl von Daten-Puffer zur Zwischenspeicherung von Dateien und zum Blocking/Deblocking. Soll eine Datei bearbeitet werden, so findet dies meist ohne jeden Disk-Zugriff in „Record-Puffern" statt. Die Datei wird dort manipuliert und erst nach Beenden der Operationen (Close File, Disk Reset, Warm-Boot) wieder auf der Diskette zurückgeschrieben. Wird von einem Anwenderprogramm auf eine Datei zugegriffen, so untersucht das BDOS zunächst, ob sich der gewünschte Record nicht schon in einem der Daten-Puffer befindet. Dann könnte ein Disk-Zugriff vermieden werden. Reichen die verfügbaren Zwischenpuffer zur Bearbeitung nicht mehr aus, so schreibt BDOS denjenigen Puffer auf die Diskette, der die längste Zeit nicht mehr beschrieben wurde. Damit wird Platz für öfter angesprochene Teile der Datei frei. Diese Technik heißt Least Recently Used Record Pufferung oder kurz LRU.

Multi-Sektor-Ein-Ausgabe

Oft kommt es vor, daß große Teile einer Datei nicht verstreut auf der Diskette liegen, sondern sequentiell aufgezeichnet sind, also so wie sie später im RAM zu finden sind. Das BDOS von CP/M-Plus erkennt dies und teilt dem BIOS mit, daß in einem Laufwerkszugriff und in einer Umdrehung der Diskette mehrere Sektoren sequentiell gelesen oder geschrieben werden sollen. In einem einzigen Zugriff lassen sich dabei bis zu 16 KByte lange Programmteile sehr schnell transferieren. Unter CP/M-2.2 wären dazu immerhin 128 einzelne Diskzugriffe nötig. Dabei versucht das BDOS von CP/M-Plus beim Schreiben von Dateien nicht, wie unter CP/M-2.2, die Lücken auf einer Diskette von vorne her aufzufüllen, sondern möglichst große zusammenhängende Teile zu schreiben, um die Multi-Sektor-Fähigkeit des BIOS voll zu unterstützen.

Datums- und Zeitmarkierung von Dateien

Im Directory einer Diskette können jeder Datei zwei Datums- und Zeitmarkierungen zugeordnet werden. Der Zeitpunkt der Erstellung, der letzten Modifikation und des letzten Zugriffs auf eine Datei kann mit Hilfe des Dienstprogramms DIR wieder angezeigt werden.

Paßwort-Schutz

Neben der Datums- und Zeitmarkierung von Dateien besteht unter CP/M-Plus die Möglichkeit des wirkungsvollen Schutzes einzelner Files oder einer ganzen Diskette vor fremdem Zugriff mit Hilfe von Passwords. Jeder Datei kann ein bestimmtes Password zugeordnet werden, ohne dessen Eingabe bestimmte Zugriffsrechte verwehrt bleiben. Der Inhalt kann vor unbefugtem Lesen, Schreiben oder Löschen geschützt werden. Die selben Schutzmöglichkeiten sind auch für eine gesamte Diskette möglich.

Automatisches Disk-Login

Das automatische Disk-Login macht es überflüssig, jedesmal mit CTRL-C zu operieren, wenn man die Disk wechselt. Das BDOS erkennt einen Disk-Wechsel automatisch. Nach einem Disk-Login wird die Diskette automatisch in das System eingebunden.

User 0 Directory

Eine weitere Verbesserung bei CP/M-Plus ist, daß häufig benutzte Programme unter „User 0 Directory" abgelegt werden können. Dadurch kann von jedem beliebigen der insgesamt 16 User-Bereiche auf dieses Programm zugegriffen werden. Befindet sich dieses Programm in Laufwerk A, User 0 Directory, so ist sogar ein Zugriff von jedem anderen Laufwerk aus unter einer beliebigen User Nummer möglich.

Ausführliche Fehlermeldung

Treten Fehler im System auf, so gibt CP/M-Plus ausführliche Meldungen über Art und Behebbarkeit der Fehler aus. *Bild 3* zeigt ein Beispiel für einen Schreibversuch auf eine schreibgeschützte Diskette. Das BIOS erkennt den Fehler und fragt den Benutzer ob ein neuer Schreibversuch gestartet werden soll. Wird dies verneint so gibt das BIOS den Fehler an das die Funktion aufrufende BDOS weiter, das eine ordentliche Fehlermeldung an den Benutzer abgibt.

```
BIOS Error on D:

     Track - 00126, Sector - 00015, Write
     Not Writable !
     Retry (Y) ? N

CP/M Error on D: Disk I/O

     BDOS Funktion = 19   File = TEST    .COM

ERROR: Bad close
```

Bild 3. Eine Fehlermeldung bei CP/M-Plus, die stufenweise vom BIOS über das BDOS zum Anwenderprogramm durchgereicht wird

Cache-Speicher, was ist das?

Immer dort, wo in der Computerei Speicher verschiedener Technologie oder Organisation auftreten, entsteht das Problem, wie man Einträge aus dem meist mehr peripheren Massenspeicher schnell in den Hauptspeicher bekommt. Dazu muß im Hauptspeicher (oder oft in einem superschnellen Sonderspeicher) notiert sein, wo bestimmte Einträge im Massenspeicher abgelegt sind. Ein solcher Speicher heißt im Slang Cache-Memory. In ihm wird die Assoziation zwischen Namen oder Inhalt des gesuchten Eintrages mit seinem Speicherort festgehalten.

BDOS und BIOS

Ähnlich wie unter CP/M-2.2 arbeitet das BDOS (Basic Disk-Operating System) des CP/M-Plus-Systems. Der Funktionsumfang ist allerdings wesentlich erweitert worden, um die vielfältigen Möglichkeiten des Betriebssystems den Anwenderprogrammen voll zugänglich zu machen. So wurden die aufrufbaren Funktionen 0...37 unter CP/M-2.2 auf 0...152 unter CP/M-Plus erweitert! Hardwareabhängige Ein-/Ausgabe-Operationen werden wie bei CP/M-2.2 über ein spezielles Dienstprogrammsystem abgewickelt, BIOS genannt (Basic Input Output System). Das BIOS stellt die Schnittstelle zum logischen, hardware-unabhängigen Teil des Betriebssystems dar und muß an das verwendete Rechnersystem angepaßt sein. Unter CP/M-2.2 genügten noch 17 BIOS-Funktionen um das Betriebssystem an die Hardware anzupassen, unter CP/M-Plus müssen schon 33 BIOS-Funktionen implementiert werden. In einer der folgenden mc-Ausgaben wird BDOS und BIOS des mc-CP/M-Plus-Computers noch eingehender beschrieben.

Der Console Command Prozessor

Der Console Command Prozessor CCP (ein 4 KByte langes Programm) wird bei einem Warmstart des Systems wie ein normales Anwenderprogramm (CCP.COM) in den TPA-Bereich bei 0100h geladen und ausgeführt. Dieser CCP ist wesentlich komfortabler als der unter CP/M-2.2. Beim mc-CP/M-Plus-Computer besteht noch eine kleine Besonderheit. Beim Kaltstart des Systems wird der CCP in den Systemspeicher Bank 0 geladen. Von dort aus wird er bei jedem Warm-Boot mit Hilfe einer schnellen Interbank-Move-Funktion des BIOS in die TPA kopiert und ausgeführt. Dadurch wird der Diskzugriff zum Neuladen des CCP nach Verlassen von Anwenderprogrammen eingespart.

RSX-Module

Wem diese Fülle an neuen Funktionen immer noch nicht ausreicht, dem stehen mit CP/M-Plus sogenannte RSX-Betriebssystemerweiterungen zur Verfügung (RSX: Resident System eXtension). Die RSX-Moduln bilden eine Zwischenstufe zwischen Anwendungsprogrammen und CP/M-System. Sie fangen die Betriebssystemaufrufe ab und führen entsprechende Funktionen selbst aus oder übergeben die Aufrufe an das BDOS weiter. Bekannt ist diese Technik in ähnlicher Form von der CP/M-2.2-Funktion XSUB. Die RSX-Module können nach Aufruf permanent im Speicher verbleiben oder nach Beendigung der Funktion wieder aus dem Speicher entfernt werden.

Die CP/M-Plus-Dienstprogramme

Unter CP/M-2.2 gab es eine kleine Anzahl leicht überblickbarer und schnell erlernbarer Dienstprogramme wie PIP, STAT, SUBMIT und LOAD. Unter CP/M-Plus wuchs die Zahl dieser Utilities auf über 24 an. Zu jedem einzelnem

```
A>HELP DEVICE

  DEVICE

  Syntax:
          DEVICE < NAMES ( VALUES ) ( physical-dev ) logical-dev>
          DEVICE logical-dev=physical-dev <option> <,physical-dev <option>,...>
          DEVICE logical-dev = NULL
          DEVICE physical-dev <option>
          DEVICE CONSOLE < PAGE / COLUMNS = columns / LINES = lines>

  Explanation:

  DEVICE displays current logical device assignments and physical
  device names. DEVICE assigns logical devices to peripheral
  devices attached to the computer. DEVICE also sets the
  communications protocol and speed of a peripheral device, and
  displays or sets the current console screen size.

  ENTER .subtopic FOR INFORMATION ON THE FOLLOWING SUBTOPICS:

OPTIONS      EXAMPLES

HELP> .OPTIONS

  DEVICE
    OPTIONS

              < XON / NOXON / baud-rate >

  XON         refers to the XON/XOFF communications protocol.

  NOXON       indicates no protocol and the computer sends data to
              the device whether or not the device is ready to
              receive it.

  baud-rate   is the speed of the device. The system
              accepts the following baud rates:

                      50      75      110     134
                      150     300     600     1200
                      1800    2400    3600    4800
                      7200    9600    19200
```

Bild 4. So wirkt HELP bei CP/M-Plus. Gezeigt ist die Erklärung zum Kommando DEVICE

```
A>DEVICE


Physical Devices:
I=Input,O=Output,S=Serial,X=Xon-Xoff

CRT    9600  IOS    V24-A  9600  IOSX   V24-B  1200  IOS
CENTR  NONE  O      PARA   NONE  IO     TTLSER 9600  IOS
DIABLO 1200  IOS

Current Assignments:
CONIN:  = CRT
CONOUT: = CRT
AUXIN:  = V24-A
AUXOUT: = V24-A
LST:    = CENTR,V24-B

Enter new assignment or hit RETURN

CONIN:=CRT,V24-B
```

Bild 5. Das zeigt sich auf dem Bildschirm bei einem Aufruf von DEVICE

Kommando können zahlreiche Optionen angegeben werden. Das PIP-Kommando unter CP/M-Plus verfügt allein über 20 verschiedene Optionen die einzeln oder kombiniert miteinander verwendet werden können. Teure Zusatz-Programmpakete unter CP/M-2.2 werden unter CP/M-Plus überflüssig. Im folgenden soll eine kleine Auswahl der wichtigsten Funktionen gezeigt werden.

HELP

Um in dieser Vielfalt an Dienstprogrammen und Optionen nicht den Überblick zu verlieren oder ständig in entsprechenden Handbüchern nachschlagen zu müssen, gibt es die Hilfsfunktion HELP. Sie erklärt dem Benutzer die verschiedenen Systemfunktionen und gibt Beispiel zu deren Gebrauch (*Bild 4*).

DEVICE

Das DEVICE-Kommando setzt Protokolle und Baud-Raten für die 16 physikalischen Ein-/Ausgabe-Einheiten fest. Außerdem kann deren Zuordnung zu den logischen Ein-/Ausgabe Einheiten CONSOLE, LIST und AUXILIARY neu definiert werden (*Bild 5*).

DIR

Das DIR-Kommando mit seinen 19 Optionen dient der übersichtlichen Darstellung des Inhaltsverzeichnisses einer Diskette (*Bild 6*). Dabei kann zwischen alphabetisierter Anzeige bestimmter Dateigruppen bis hin zur Anzeige über mehrere Disketten und Userbereiche hinweg gewählt werden.

```
A>DIR <ATT>


Scanning Directory...

Sorting  Directory...

Directory For Drive E:  User 12

    Name      Bytes   Recs   Attributes   Prot      Update          Access
------------ ------ ------ ------------ ------ --------------  --------------

CCP      COM     4k     25 Dir RO  1    Delete 01/24/84 23:50  04/25/84 13:50
CPM3     LIB     4k     32 Sys RW   23  None   04/16/84 18:22  04/25/84 14:47
CPM3     SYS    22k    174 Sys RO  1234 Delete 02/01/84 09:01  05/07/84 12:05
DATE     COM     4k     22 Dir RW  1    Write  04/17/84 13:33  05/07/84 12:00
DEVICE   COM     8k     58 Dir RW       Write  01/04/84 08:02  05/07/84 12:04
DIR      COM    16k    114 Dir RW   3  Write  04/17/84 18:05  05/07/84 11:59
HELP             0k      0 Dir RW       None   04/15/84 23:30  05/20/84 15:41
PUT      COM     8k     55 Dir RW  1  4 Write  01/14/84 15:34  05/20/84 15:48
RMAC     COM    14k    106 Dir RO  12   Read   04/15/84 23:24  05/07/84 12:00
SET      COM    12k     81 Dir RW  1    Write  04/16/84 18:12  05/20/84 13:49
SETDEF   COM     4k     32 Dir RW  1    Write  04/25/84 13:52  05/21/84 13:52
WS       COM    16k    124 Dir RO  1 3  Read   05/07/84 11:33  05/07/84 12:03
Z80      LIB     6k     47 Sys RW   23  None   04/25/84 13:53  05/15/84 10:53

Total Bytes     =    118k  Total Records =     870  Files Found =   13
Total 1k Blocks =    114   Used/Max Dir Entries For Drive E:   33/  64
```

Bild 6. Ein Beispiel für das DIR-Kommando, angewendet auf eine System-Diskette mit vielen Kommandos

GET/PUT

Das GET-Kommando steuert die Console-Eingabe so um, daß sie von der Disk her kommen kann. Eine Datei kann also, ähnlich wie beim Submit-Befehl, Steuerdaten oder Anwendungsprogramm-Eingabedaten enthalten. Das PUT-Kommando ergänzt das GET-Kommando: Ausgaben auf die Console oder den Drucker können mit PUT auf die Diskette in eine Datei geschrieben werden.

SET

Mit dem SET-Kommando können verschiedene Dateiattribute (Read Only, System, Directory...), die Art der gewünschten Datums- und Zeitmarkierungen und der Password-Schutz festgelegt werden.

SETDEF

Das SETDEF-Kommando definiert und zeigt die Disk-Such-Reihenfolge an. Mit diesem Kommando kann man CP/M-Plus veranlassen, mehr als ein Laufwerk nach der gewünschten COM- oder SUB-Datei abzusuchen.

SHOW

Das SHOW-Kommando zeigt Informationen über die Daten eines logischen Laufwerks an, wie Kapazität, Anzahl der möglichen Directory-Einträge, reservierte Systemspuren usw.

Hilfsmittel zur Softwareentwicklung

Zur Softwareentwicklung auf Maschinenebene gehört zu CP/M-Plus ein verbesserter Editor „ED", der symbolische Debugger „SID", der Macroassembler „MAC" mit Z80.LIB, der relokative Macroassembler „RMAC", der Linker „LINK-80", ein Symbolprogramm „XREF" und „LIB-80" zum Verarbeiten von Assembler-Librarys.

Ein neuer mc-Computer

Daß dieses neue Betriebssystem zum Verwirklichen aller Funktionen hohe Anforderungen an die Hardware und die BIOS-Implementation stellt, ist offen-

sichtlich. mc stellt in der nächsten Ausgabe einen CP/M-Plus-Einplatinen-Rechner zum Nachbau vor, der alle Anforderungen, die CP/M-Plus stellt, auf einer einzigen Platine erfüllt. Nur so viel im voraus: Z80B-CBU mit 6 MHz, 128 KByte RAM, 4 KByte EPROM, Floppy-Schnittstelle zum direkten Anschluß von vier 8-, 5¼- oder 3-Zoll-Laufwerken, zwei RS-232-Schnittstellen, ein serieller TTL-Vollduplex-Kanal, eine Centronics-Schnittstelle, Banking-Verwaltung für bis zu 1 MByte und eine akkugepufferte Echtzeituhr. Auch das BIOS kann sich sehen lassen: Es können vier verschiedene Laufwerke angeschlossen werden, es erkennt selbständig 32 verschiedene Diskformate, verwaltet die I/O-Geräte mit wählbaren Protokollarten und Baudraten, steuert die RGB-Farbgrafik-Platine mit 3 × 512 × 512 Punkten Auflösung und verwaltet 1 MByte Arbeitsspeicher durch sehr schnellen Interbank-Move und eine RAM-Floppy-Implementation.

Was ist Hashing?

Immer dort, wo in der Computerei eine größere Anzahl von „Einträgen" gleichberechtigter Natur im Speicher abgelegt sind, tritt das Problem des Wiederfindens eines bestimmten Eintrages auf. Man kann einmal die Einträge völlig willkürlich in einer Tabelle ablegen. Dann muß man, wenn man einen bestimmten Eintrag finden will, diese Tabelle solange durchsuchen, bis man auf den Eintrag gestoßen ist. Zum anderen kann man aber auch aus dem Inhalt eines Eintrages rechnerisch ableiten, wo der Eintrag abgespeichert werden soll und wo er wiedergefunden werden kann. Zum Beispiel könnte man ein verrücktes Lexikon aufbauen, indem man jedem Wort diejenige Zahl zuordnet, die seine in ASCII entstehende Bitfolge als Dualzahl darstellt. Diese Zahl könnte man als die Seitenzahl des Lexikons hernehmen, auf der das Wort mit seiner Erläuterung gespeichert ist. So könnte man aus dem Wort tatsächlich errechnen, wo sein Speicherort ist. Ein Verfahren, das extrem schnell, dafür aber aufwendig ist. Man kann nun zwischen dem planlosen Suchen und dem vollständigen errechnen des Speicherortes noch Zwischenstufen einführen. Zum Beispiel kann man beim verrückten Lexikon nicht die ganzen Wörter in voller Länge hernehmen, sondern den Speicherort nur aus den ersten beiden Buchstaben errechnen. Dann kommt es zwar bei gleichlautenden Wortanfängen zu Konflikten, die man aber dadurch auflösen kann, daß man in solchen Fällen Untertabellen anlegt, die wiederum zum Beispiel planlos durchsucht werden können. Die letztgenannte Speicherorganisation heißt Hash-Tabellentechnik und das Aufsuchen von Einträgen dort heißt Hashing.

F. Oettle und Th. Reichler

Der Einplatinen-computer für CP/M-Plus

Der mc-CP/M-Plus-Computer, Teil 2

In der letzten Ausgabe von mc wurde das Betriebssystem CP/M-Plus vorgestellt. Solche Hochleistungs-Software fordert natürlich auch Hochleistungen von der Hardware. Der im folgenden beschriebene Einplatinencomputer realisiert auf einer einzigen Einfach-Europa-Karte sämtliche Hardware-Voraussetzungen für ein professionelles und äußerst leistungsfähiges CP/M-3.0-Computersystem.

Die Bezeichnung Einplatinencomputer erweckt sicherlich bei so manchem erfahrenen Anwender die Befürchtung, daß solche Einkarten-Lösungen gar nicht mehr so kostengünstig sind, wenn man sie erweitern will. Oft ist ein Ausbau sogar gänzlich unmöglich. Um den Anwender vor solchen Überraschungen zu verschonen, wurde bei der Entwicklung des mc-CP/M-Plus-Computers besonderer Wert auf einen von der Industrie genormten Bus-Anschluß gelegt: Es wird der ECB-Bus verwendet. Damit kann man den Computer in Verbindung mit vielen ECB-Karten, z. B. einer Farbgrafik-Karte (wird in einer der nächsten Ausgaben vorgestellt), betreiben. Ohne Änderungen läßt sich der Computer auch als neuer „Turbo-Motor" in ein bestehendes mc-CP/M-Computer-System einsetzen. Auch die bereits erstellte Software läßt sich unverändert übernehmen.

Ein Stand-alone-System

Der mc-CP/M-Plus-Computer stellt sämtliche Funktionen für ein komplettes CP/M-Plus-Computersystem zur Verfügung. An Peripherie werden nur ein handelsübliches Terminal und mindestens ein Floppy-Laufwerk benötigt. *Bild 1* zeigt das schematisch.

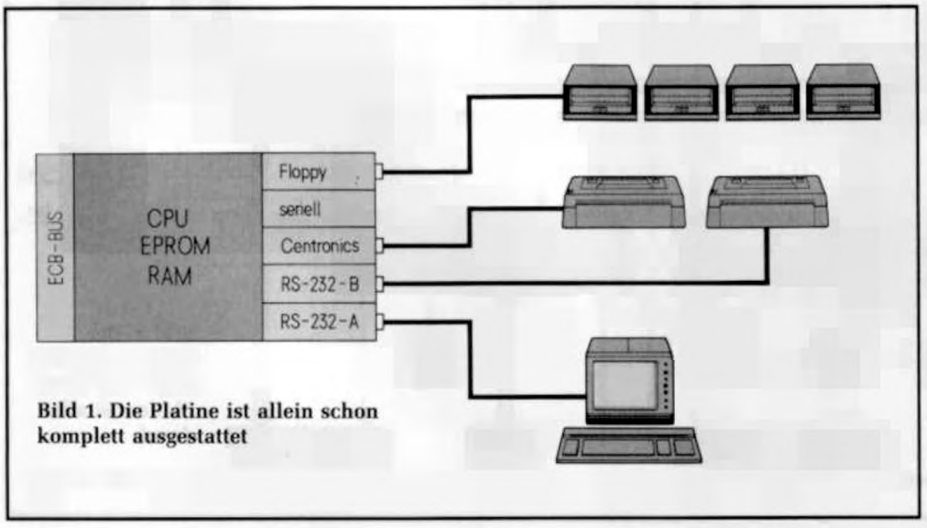


Bild 1. Die Platine ist allein schon komplett ausgestattet

Master-Baugruppe

Über die ECB-Bus-Schnittstelle kann der mc-CP/M-Plus-Computer mit einer reichhaltigen Palette von Zusatzkarten erweitert werden. Zusätzlich notwendig sind dann eine ECB-Bus-Rückwandverdrahtung und ein Einschubgehäuse. Es kann wahlweise ein Terminal über RS-232 oder über den ECB-Bus eine Video-Karte mit separatem Monitor und Tastatur angeschlossen werden.

I/O-Baugruppe

Sämtliche auf der Platine befindlichen Funktionseinheiten, wie RAM, Floppy und I/O, können auch von einer externen CPU angesprochen werden. Die Platine läßt sich in ein bereits bestehendes ECB-System einstecken und als I/O- und RAM-Karte ausnutzen. Dies ermöglicht zum Beispiel einen gezielten Test der Baugruppe beim Aufbau. Ebenso kann auf diese Weise spezielle Software für den mc-CP/M-Plus-Computer in einem bestehenden System entwickelt und getestet werden.

Multi-Prozessor-Anwendungen

Der mc-CP/M-Plus-Computer kann in einem Multiprozessor-System als Slave parallel zu anderen Prozessoren arbeiten, ohne die Aktivitäten des Masters auf dem System-Bus zu stören. Der Datenaustausch geschieht dabei mit der vollen Geschwindigkeit über den Bus.
Eine dritte serielle Schnittstelle mit TTL-Pegel eignet sich ebenfalls für asynchrone Vollduplex-Datenübertragung.

Die Echtzeituhr

Ein akku-gepufferter Uhrenbaustein stellt dem System Echtzeit zur Verfügung. Diese kann unter CP/M-Plus in Dateien eingebunden werden. Folgende Angaben stehen zur Verfügung: (Schalt-) Jahr, Monat, Wochentag, Tag, Stunde, Minute und Sekunde.

Die Stromversorgung

Die Karte (siehe Schaltplan, *Bild 4*) benötigt zwei Betriebsspannungen: +5 V und +12 V, wobei die +12 V nur für die RS-232C-Schnittstelle erforderlich sind. −12 V werden von einem Spannungsinverter ICL 7660 (IC 33) erzeugt. Dem RS-232C-Treiber MC-1488 (IC 36) werden somit etwa ±9 V bereitgestellt.

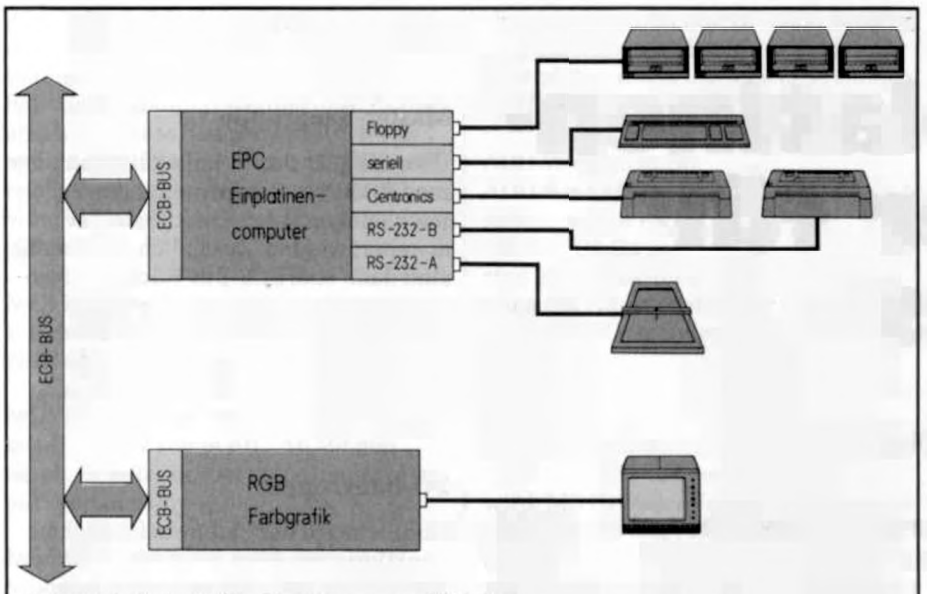


Bild 2. Ein Beispiel für die Nutzung am ECB-Bus

Die Takterzeugung

Auf der Karte befinden sich zwei Quarzoszillatoren, aufgebaut aus einem 74LS626-Baustein und zwei Quarzen. Der Systemtakt (6 MHz) wird direkt aus dem Oszillator gewonnen, wobei eine Transistor-Treiberstufe das Fan-Out verbessert. IC 05 (74LS194) teilt den Systemtakt durch den Faktor zwei und versorgt damit den Timer-Eingang TCLK des STI-Bausteins (IC 26). Der zweite Oszillator erzeugt den Takt (16 MHz) für den Baustein FDC-9229 (IC 23).

Der Reset

Ein Power-up-Reset wird durch ein RC-Glied und zwei 74LS14-Gatter (IC 19) erzeugt und über eine Open-Collector-Stufe auf PWCLR (Pin 26c am Bus-Stekker) geführt. Dieses Signal wird zum Zurücksetzen der CPU, STI, DART, FDC und des 74LS173 (Banking) benutzt. Der Eingang Reset am Bus-Stecker (31c) kann zum Anschluß eines Reset-Schalters verwendet werden.

Wait

Die Wait-Schaltung kann bei I/O- und Speicheroperationen einen Wait-Zyklus einfügen. Der Eingang (Pin 10) des 74LS194 (IC 05) ist mit dem Wait-Request-Signal des I/O- und des Memory-PROMs beschaltet. Wird dieses aktiv, wechselt IC 05 vom Load- in den Shift-Zustand über. Mit der nächsten steigenden Flanke des Systemtaktes wird ein Wait-Puls ausgegeben.
Durch Programmieren des I/O-PROMs kann festgelegt werden, bei welcher der insgesamt 256 I/O-Adressen ein Wait-Zyklus generiert werden soll. Das Memory-PROM erlaubt es jedem 16-KByte-Speicherblock, innerhalb des 1-MByte-Speicherbereichs einen Wait-Zyklus-

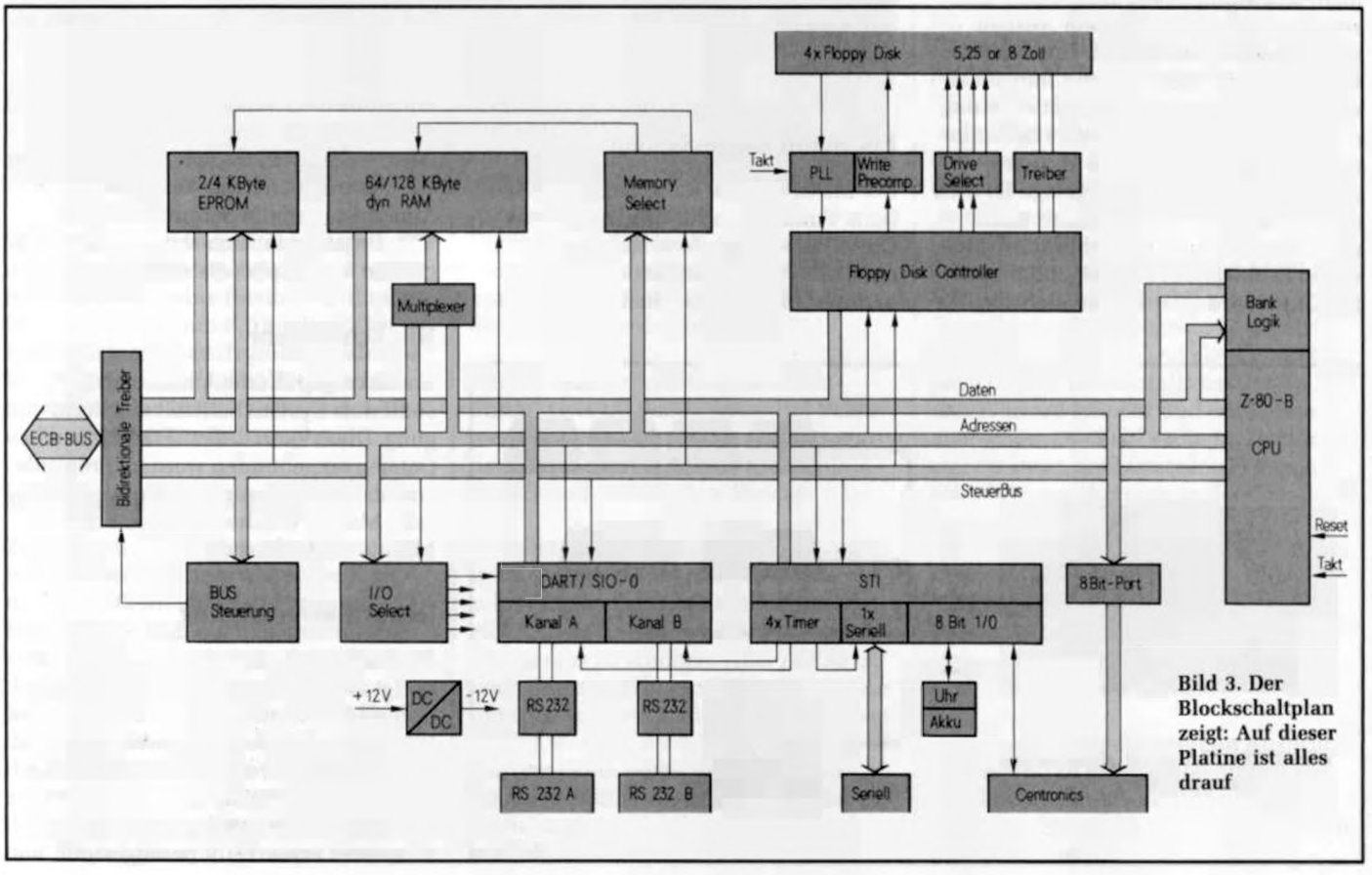


Bild 3. Der Blockschaltplan zeigt: Auf dieser Platine ist alles drauf

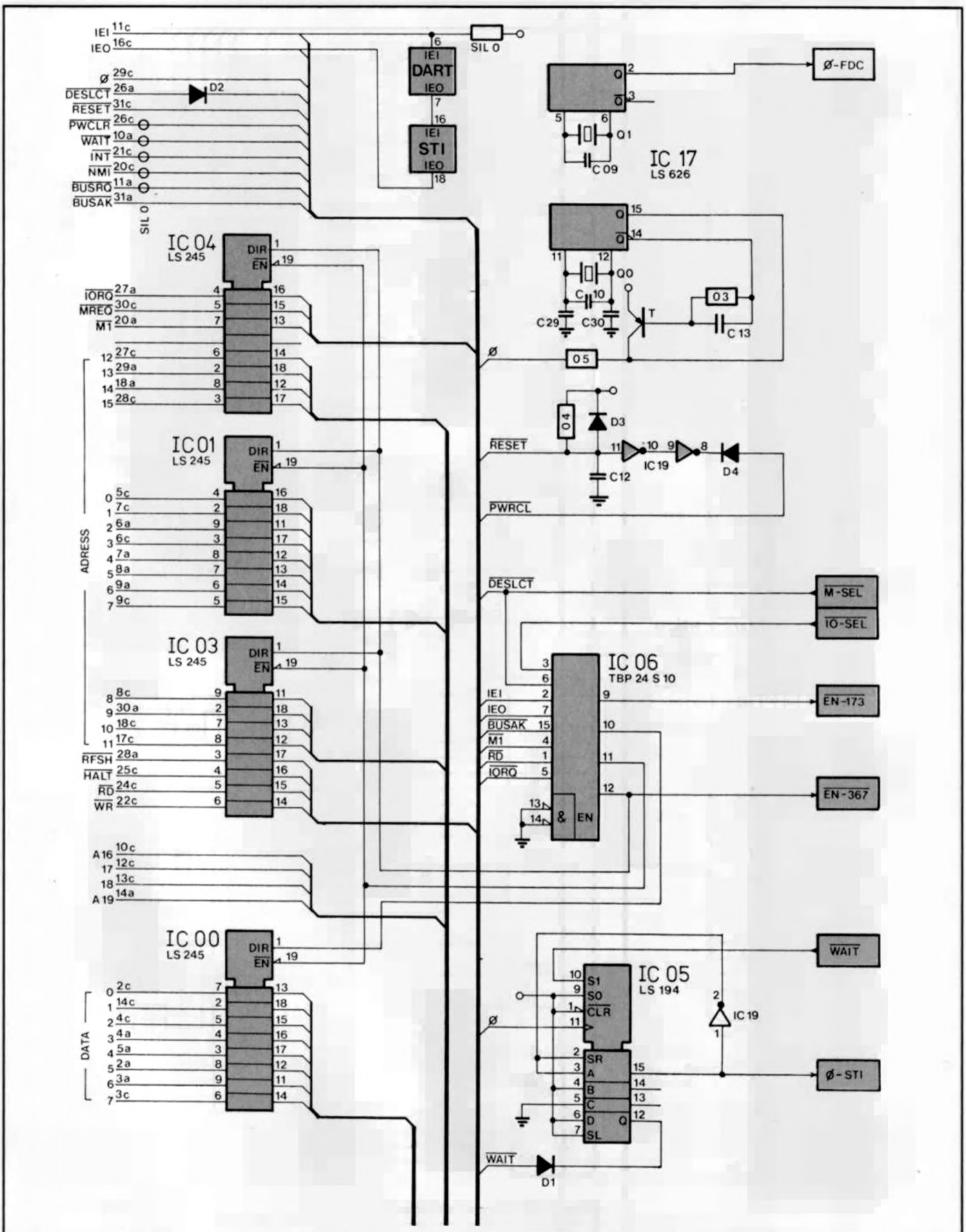


Bild 4a. Der Schaltplan ist umfangreich. Teil a zeigt Reset und Takterzeugung sowie die Bus-Pufferung und die Bus-Steuerung über PROM

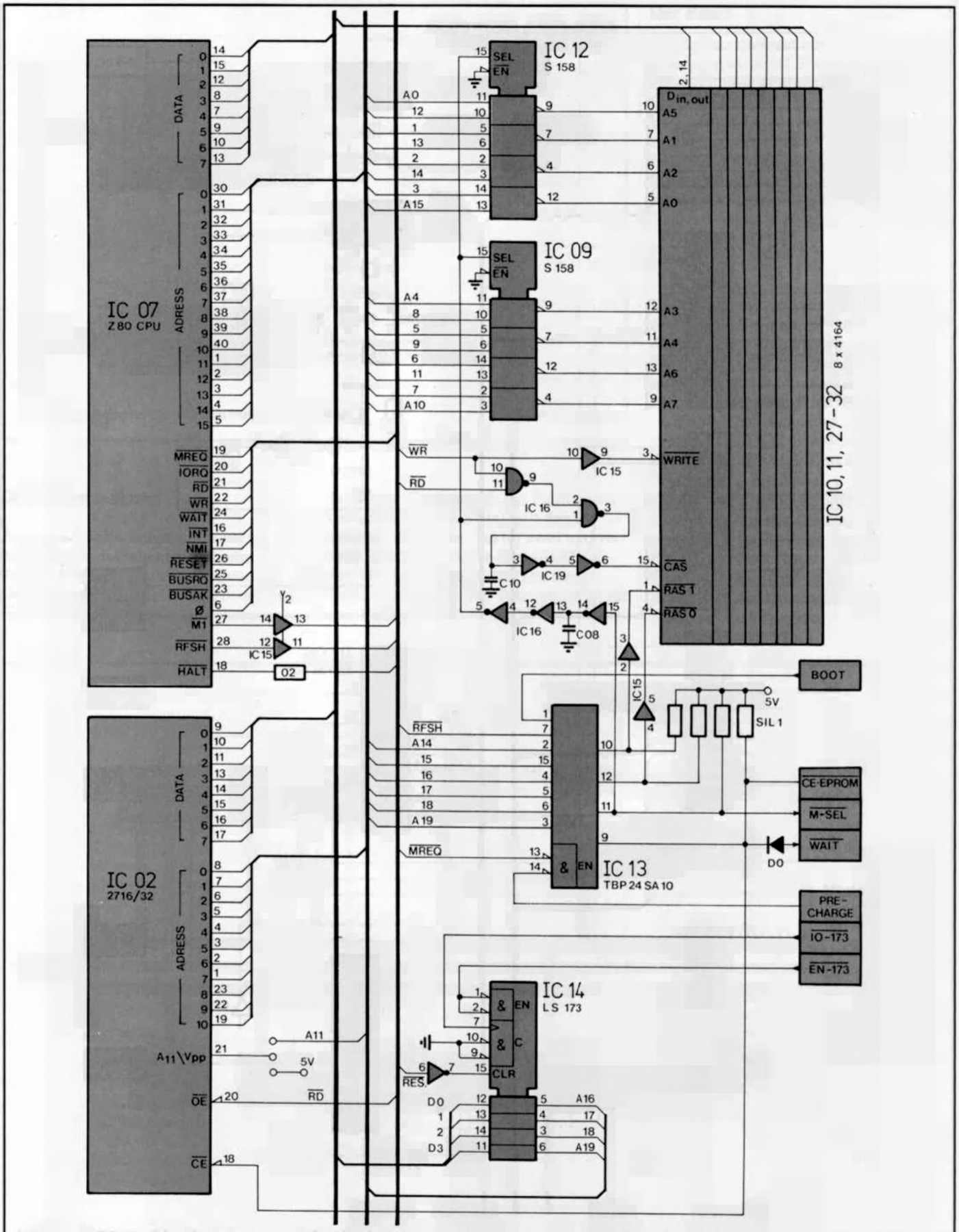


Bild 4b. CPU, RAM, Ansteuerlogik für den Speicher mit PROM sowie EPROM-Boot-Baustein

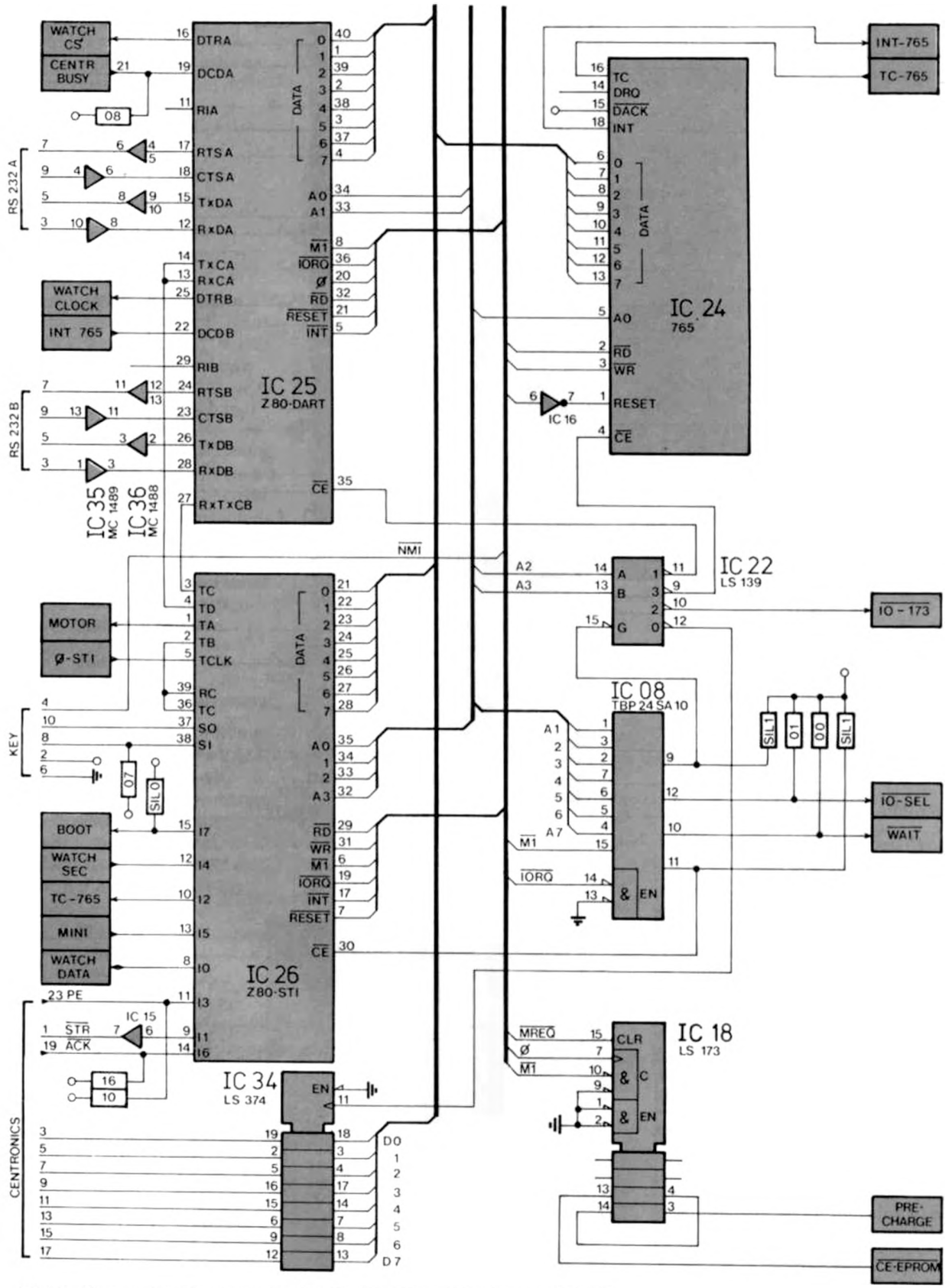


Bild 4c. Die Schnittstellenbausteine Z80-DART, -STI und der Floppy-Controller

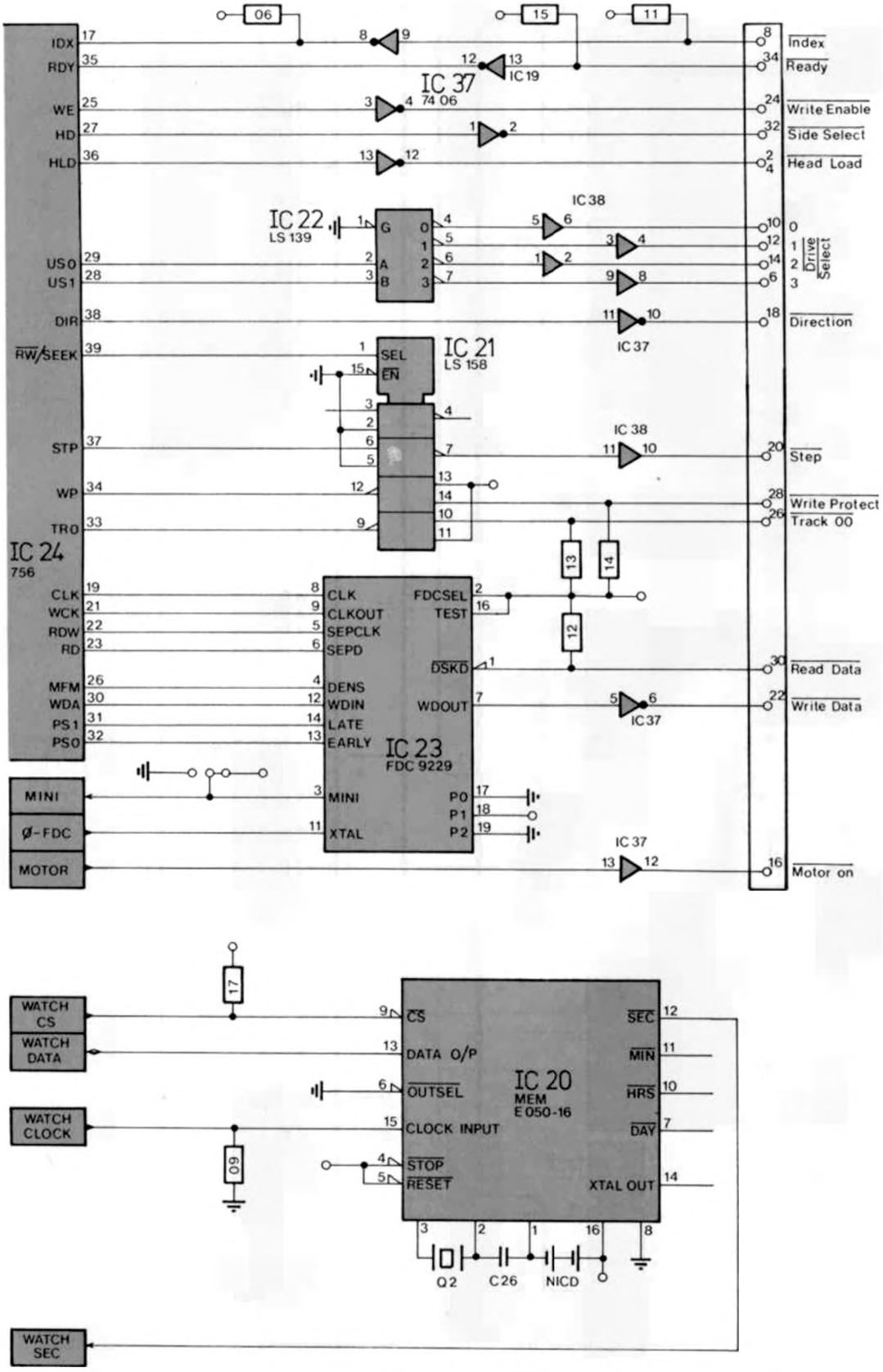


Bild 4d. Elemente der Floppy-Steuerung und Uhrenbaustein

Puls zuzuordnen. Standardmäßig wird bei jedem I/O-Zugriff (Ausnahme: µPD-765) und bei Speicherzugriffen auf das BOOT-EPROM jeweils ein Wait-Puls eingefügt. Speicherzugriffe auf externe Speicherbaugruppen und auf den internen dynamischen RAM-Speicher geschehen mit voller Geschwindigkeit.

Die Bankumschaltungs-Logik

Diese Schaltung erlaubt es der CPU, 1 MByte Speicherraum direkt zu adressieren. Ein 4-Bit-Ausgabeport, IC 14 (74LS173), erzeugt die vier Adressen A16...A19 (Pageadressen). Über I/O-Ausgabe-Befehle (D0 = A16, D3 = A19) läßt sich der Inhalt dieses Ports und damit die Bank-Adresse bestimmen.
Das PROM (IC 13), das die Speicheradressen-Decodierung enthält, ist standardmäßig so ausgelegt, daß die obersten 16 KByte der ersten Speicherbank im 64-KByte-Adreßraum der CPU unabhängig von den Adressen A16...A19 eingeblendet sind. Beim Wechsel auf eine andere Bank werden also nur die untersten 48 KByte ausgetauscht. Bei einem Reset wird der 74LS173-Baustein zurückgesetzt und die erste 64-KByte-Bank wird angesprochen.

Die Speicheradressen-Decodierung

Zur Freischaltung des 24poligen Byte-Wide-Sockels und der 128 KByte RAM auf der Platine wird ein 256×4-Bit-PROM (IC 13) eingesetzt. Durch entsprechende Programmierung kann die Adreßlage der beiden Speicherarten bestimmt werden. Die Adreßlage läßt sich dabei in 16-KByte-Schritten frei im maximal möglichen Adreßraum von 1 MByte bestimmen. Der BOOT-Eingang am PROM erlaubt das Ein- und Ausschalten einzelner Speicherbereiche über die System-Software. Das „Memory-PROM" wird normalerweise für den Bausatz so programmiert, daß das auf der Karte befindliche RAM die unteren 128 KByte belegt. Das EPROM ist dabei den untersten 16 KByte überlagert. Führt der BOOT-Eingang am PROM High-Pegel, ist das EPROM eingeblendet. Wird I7 des STI-Bausteins auf Low-Pegel programmiert, wird das BOOT-EPROM abgeschaltet und der adreßgleiche 16-KByte-RAM-Speicher aktiviert. Das oberhalb von 04000H liegende RAM ist ständig aktiv.
Abhängig von den acht Eingangssignalen und den beiden $\overline{CE}$-Signalen werden die vier Ausgänge $\overline{RAS1}$, $\overline{RAS2}$, $\overline{MEM\text{-}SEL}$ und $\overline{CE\text{-}EPROM}$ aktiv (Low) geschaltet: Zwei der Ausgänge werden direkt als $\overline{RAS}$-Signale für die beiden 64-KByte-RAM-Blöcke benutzt. Diese sind aktiv, wenn ein Speicherzugriff auf die entsprechende Adreßlage oder ein Refresh erfolgt.
Der Ausgang $\overline{M\text{-}SEL}$ führt Low-Pegel, wenn auf einen internen Speicher der Platine zugegriffen wird und dient zur Bus-Steuerung. Dieses Signal wird als Open-Collector-Ausgang auch an den Bus-Stecker DESLCT (Pin 26a) geführt.

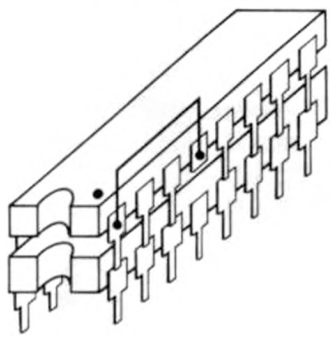


Bild 5. Die Speicherbausteine müssen huckepack montiert werden, wenn 128 KByte Platz finden sollen. Deshalb sind auch nur ICs 4164 verwendbar, deren Pin 1 im Datenblatt mit NC gekennzeichnet ist. Der Refresh muß über 7 Bit laufen

Die Stückliste zum mc-CP/M-Plus-Computer

ICs		IC 36	MC 1488	*Kondensatoren*	
IC 00, 01	74LS245	IC 37	7406	C 00	4,7 µF, 10 V
IC 02	2716/32	IC 38	7407	C 01	47 nF
IC 03, 04	74LS245			C 02, 03	4,7 µF, 10 V
IC 05	74LS194	*Dioden*		C 04	47 nF
IC 06	TBP 24 S 10	D 00...02	AA 143	C 05, 06	4,7 µF, 10 V
IC 07	Z80-B-CPU	D 03	1N4148	C 07	47 nF
IC 08	TBP SA 10	D 04	AA 143	C 08	470 pF
IC 09	74S158	D 05	1N4148	C 09, 10	entfällt
IC 10, 11	4164, 150 ns	D 06	Z 2, 7	C 11	4,7 µF, 10 V
IC 12	74S158			C 12	10 µF, 6,3 V
IC 13	TBP 24 SA 10	*Transistor*		C 13	12 pF
IC 14	74LS173	T 00	2N2907A	C 14...20	47 nF
IC 15	74LS367			C 21...23	10 µF, 16 V
IC 16	4929	*Widerstände*		C 24	4,7 µF, 10 V
IC 17	74LS626	R 00	2,2 kΩ	C 25, 26	47 nF
IC 18	74LS173	R 01	680 Ω	C 27	3...12 pF, Trimmer
IC 19	74LS14	R 02	2,2 kΩ	C 28	47 nF
IC 20	MEM E050-16	R 03	82 kΩ	C 29, 30	10 pF
IC 21	74LS158	R 04	47 kΩ		
IC 22	74LS139	R 05	22 Ω	*Quarze*	
IC 23	FDC 9229 BT	R 06	680 Ω	Q0	6 MHz
IC 24	µPD-765	R 07...10	4,7 kΩ	Q1	16 MHz
IC 25	Z80-B-DART	R 11...15	330 Ω	Q2	32,768 kHz
IC 26	Z80-A-STI	R 16, 17	4,7 kΩ		
IC 27...32	4164, 150 ns			*Akku*	
IC 33	ICL 7660	*Widerstands-Netzwerke*		NICD	2× 20 DK-F
IC 34	74LS374	SIL 0	7× 4,7 kΩ		
IC 35	MC 1489	SIL 1	7× 680 Ω		

Das $\overline{CAS}$-Signal für den dynamischen Speicher wird ebenfalls aus diesem Signal abgeleitet.
Das $\overline{CE}$-Signal für den 24poligen Sockel wird an die Wait-Schaltung weitergeleitet, welche einen Wait-Puls generiert.

Der dynamische Speicher

Als Massenspeicher werden dynamische 64-KBit-Speicher-Bausteine eingesetzt. Wichtig ist, daß nur RAMs mit 7-Bit-Refresh (128 Refresh-Zyklen) und 150 ns Zykluszeit verwendet werden. Wahlweise können acht (64 KByte) oder 16 (128 KByte) Bausteine eingebaut werden.
Um 128 KByte RAM auf diesem kleinen Raum unterzubringen, werden jeweils zwei der 64-KBit-Bausteine übereinander montiert (*Bild* 5). Pin 4 ($\overline{RAS}$) des oberen Speicher-ICs ist nicht mit dem des unteren verbunden, sondern wird getrennt über den unbelegten Pin 1 herausgeführt. Also sind nur Bausteine ohne Auto-Refresh verwendbar. Alle anderen Anschlüsse werden eins zu eins mit denen des unteren ICs verbunden.

Die Ansteuerung des Speichers

Die dynamischen Speicher werden im Early-Write-Modus betrieben. Das heißt, bei einem Schreibzugriff ist das Write-

Signal bereits mit der fallenden Flanke von $\overline{CAS}$ aktiv (Low). Dies hat den Vorteil, daß die Datentreiber (DO) beim Schreiben im Tri-State-Zustand bleiben. DI und DO müssen deshalb nicht getrennt gepuffert werden und lassen sich so einfach verbinden.
Die Adressen A0...A15 werden über zwei invertierende Adreßmultiplexer 74S158 auf die Speicher geschaltet. Wichtig ist, daß hier unbedingt Schottky-Bausteine eingesetzt werden (ICs 07 und 09).

Die Precharge-Schaltung

Um bei 6 MHz Systemfrequenz einen einwandfreien Betrieb mit den dynamischen Speichern zu gewährleisten, ist eine Precharge-Extension-Schaltung erforderlich. Diese hat zur Aufgabe, die Precharge-Zeit beim M1-Zyklus zwischen Speicherzugriff (Opcode Fetch) und Refreshzugriff während T3 zu verlängern. Wird auf den dynamischen Speicher zugegriffen, beendet IC 18 (74LS173) den Opcode-Fetch-Zyklus mit der steigenden Flanke von T3 vorzeitig.

Festwertspeicher

Als Festwertspeicher können 2- bzw. 4-KByte-EPROMs eingesetzt werden. Der 24polige Sockel ist dabei nach JEDEC-Norm ausgeführt. Pin 21 wird wahlweise mit +5 V (2 KByte) oder A11 (4 KByte) beschaltet (am Pin 21 auf der Lötseite, Standardmäßig ist Pin 21 mit +5 V verbunden).

Der Floppy-Controller

Als Floppy-Controller wurde der µPD-765 von NEC ausgewählt. Er ist in der Lage, die unter CP/M-3.0 notwendigen Multi-Sektor- und Multi-Track-Operationen zu unterstützen. Seine hohe Intelligenz entlastet die Software erheblich. So ist er in der Lage, selbständig Disketten zu formatieren und Informationen über den Zustand von vier Laufwerken zwischenzuspeichern. Das Einstellen von irgendwelchen Trimmern entfällt vollständig.
Während des Transfers von und zur Diskette wird der µPD-765 im Polling-Verfahren betrieben. Der Systemtakt von 6 MHz erlaubt es, 8-Zoll-Double-Density-Disketten mit diesem Verfahren einwandfrei zu bearbeiten.

Der Interface-Baustein

Den einwandfreien Datentransfer von und zur Diskette übernimmt der Baustein FDC-9229 (IC 23). Beim Lesen von der Diskette sorgt eine integrierte Phase-Lock-Loop-Schaltung für die Aufbereitung des seriellen Datenstroms. Beim Schreiben gewährleistet eine Write-Precompensations-Schaltung die sichere Aufzeichnung. Der Precompensations-Wert beträgt bei Mini-Laufwerken 250 ns und bei Standard-Laufwerken 125 ns. Prinzipiell können über die Eingänge P0...P2 auch andere Werte am Baustein eingestellt werden.
Der Eingang MINI am 9229 erlaubt es, zwischen 5,25-Zoll- und 8-Zoll-Laufwerken umzuschalten. Im wesentlichen wird dabei die Datentransferrate von 250 KBit/s bei 5,25 Zoll auf 500 KBit bei 8 Zoll umgeschaltet. Der Zustand des MINI-Eingangs läßt sich über eine Lötbrücke (unter dem Akku auf der Lötseite) bestimmen und über den Eingang I5 des STI-Bausteins von der Software ab-

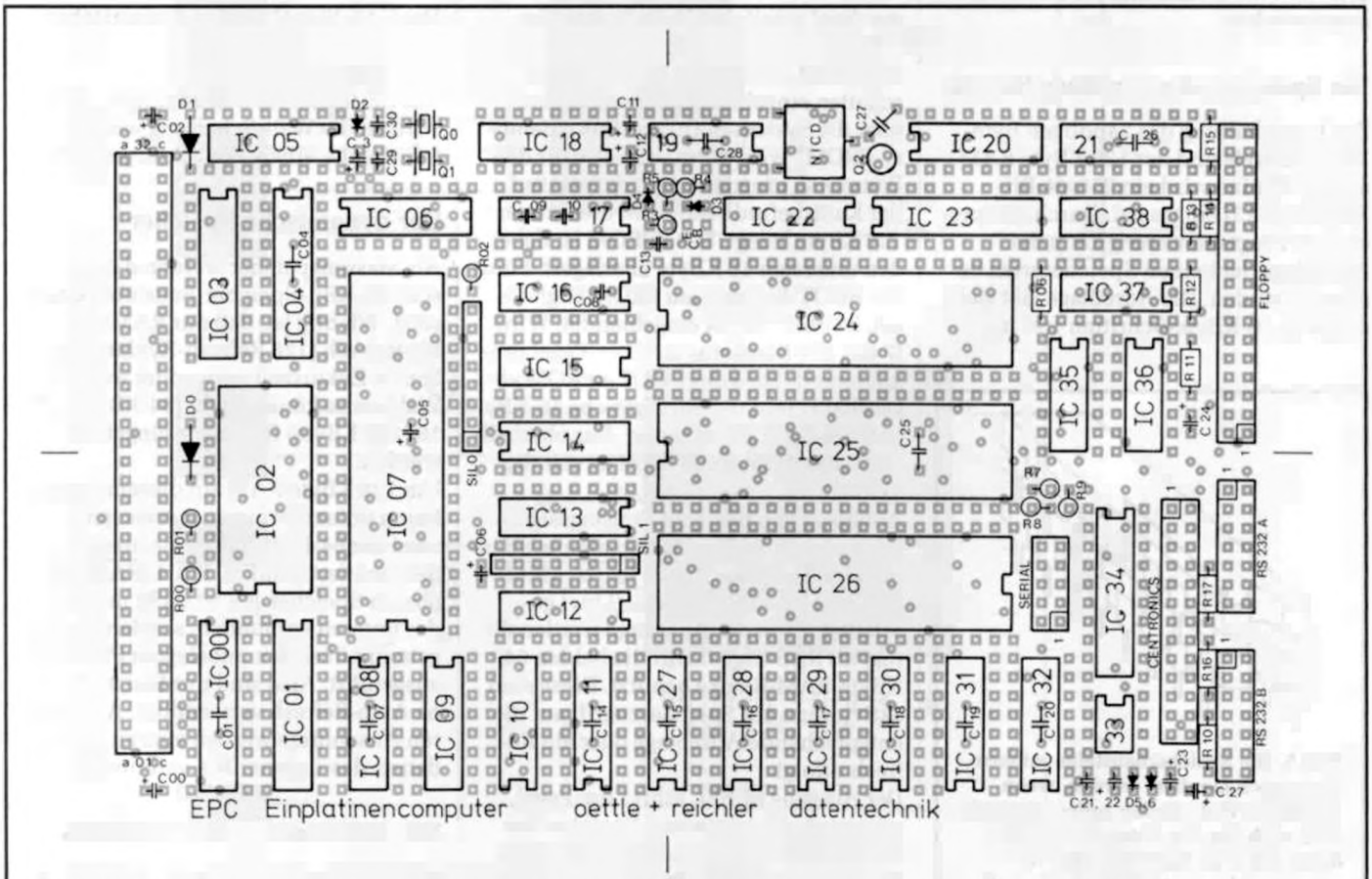


Bild 6. Der Bestückungsplan. Je zwei RAM-ICs müssen bei 128 KByte übereinander gelötet werden

fragen. Standardmäßig liegt der MINI-Eingang auf High-Potential, womit 3-Zoll- und 5,25-Zoll-Laufwerke angesprochen werden können.
Bild 7 zeigt die Belegung des Floppy-Steckers. Sie entspricht dem Industrie-Standard für Mini-Laufwerke.

I/O-Select

Die Decodierung der I/O-Bausteine übernimmt ein 256×4-Bit-PROM (IC 08) und ein 74LS139-TTL-Baustein (IC 22). Durch Programmierung des PROMs lassen sich die I/O-Bausteine beliebig im maximalen I/O-Adreßbereich (max. 256) anordnen. In Abhängigkeit von acht Eingangssignalen M1 und A1...A7 werden vier Ausgangssignale erzeugt: Das Signal $\overline{\text{IO-SEL}}$ führt Low-Pegel, wenn auf einen internen I/O-Baustein zugegriffen wird, und dient zur Bus-Steuerung. Ein Wait-Request-Signal, das an die Wait-Schaltung geführt ist, gestattet jeder der 256 I/O-Adressen, gezielt einen Wait-Puls zuzuordnen. Die beiden verbleibenden Ausgänge dienen als Chip-Enable-Signale für die I/O-Bausteine.

Ein $\overline{\text{CE}}$ wird direkt an den STI-Baustein geführt, der 16 I/O-Adressen belegt. Die oberen vier Adreßbits A4...A7 können frei durch Programmierung des PROMs bestimmt werden. Die Z80-DART, der FDC, das Centronics-Datenport und die Bank-Logik belegen ebenfalls 16 I/O-Adressen. Die oberen vier Adressen dieses Blocks können über das PROM festgelegt werden. Die Decodierung der Adressen A2 und A3 übernimmt IC 22 (74LS139). Innerhalb des 16-Byte-Blocks haben die vier I/O-Baugruppen folgende Adreßlage:

XXXX 00xx	Centronics-Datenport
XXXX 01xx	Z80-DART
XXXX 10xx	Banking-Port (74LS173)
XXXX 11xx	FDC-765

Die I/O-Bausteine belegen bei Verwendung des Standard-PROMs folgende Adressen:

00...0F	Z80-STI
10	Centronics
14...17	Z80-DART/-SIO
18	Banking-Output (74LS173)
1C, 1D	FDC-Controller µPD-765

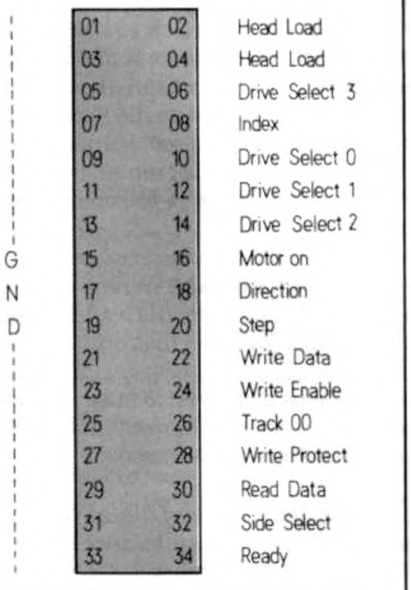


Bild 7. Die Belegung des Floppy-Stekkers. Jede gängige Mini-Floppy kann daran angeschlossen werden

Die Centronics-Schnittstelle

Die Centronics-Schnittstelle erlaubt es, auf einfache Weise einen Drucker anzuschließen. Der 8-Bit-Datenstrom zum Drucker wird über acht D-Flipflops eines 74LS374-Bausteins (IC 34) ausgeworfen. Der Strobe-Puls wird von dem STI-Baustein erzeugt und über ein Gatter (74LS367) gepuffert. Die Eingänge Paper Empty und Acknowledge sind ebenfalls mit der STI verbunden. Busy ist an den DART-Baustein geführt. Alle Signale können somit Interrupts auslösen. Sämtliche Signale sind auf einen 26poligen Stecker geführt. *Bild 8* zeigt die Belegung.

Die RS-232C-Schnittstellen

Es stehen zwei serielle asynchrone Vollduplex-Kanäle zur Verfügung (Alternativ ist durch Austausch der Z80-DART durch eine Z80-SIO-O auch synchrone Datenübertragung möglich). Die Baudrate ist softwaremäßig für beide Kanäle getrennt einstellbar. Kanal A der DART wird vom Timer D (STI) und Kanal B vom Timer C mit dem Baudraten-Takt versorgt. Receive- und Transmit-Data, Clear to Send und Request to Send sind bei beiden Kanälen mit RS-232-Treibern gepuffert und getrennt für jeden Kanal auf zwei 14polige Pfostenstecker geführt. Die Belegung dieser Stecker entspricht der Norm. Über ein Kabel in Schneid-Klemm-Technik lassen sie sich direkt mit einem 25poligen D-Stecker verbinden. *Bild 9* zeigt die Belegung.

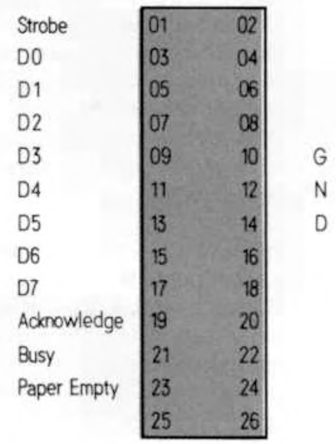


Bild 8. Das ist die Belegung der Centronics-Schnittstelle. Mit Schneid-Klemm-Technik kann man das Kabel eins zu eins an einen Centronics-Druckeranschluß führen

Seriell, 5-V-TTL

Eine weitere serielle asynchrone Vollduplex-Schnittstelle wird auf einem 10poligen Stecker bereitgestellt. Sie läßt sich zum Beispiel zum Anschluß einer Tastatur verwenden. Die Baudrate ist auch hier softwaremäßig einstellbar (Timer B, STI). Der Pegel ist TTL. *Bild 10* zeigt die Belegung.

Die Echtzeituhr

Der Uhrenbaustein E-050-16 erlaubt die Angabe von Sekunden, Minuten, Stunden, Tag, Wochentag, Monat und (Schalt-)Jahr. Das geschieht seriell in BCD. Ein Akku ermöglicht den Betrieb auch bei ausgeschaltetem Computer. Über den Trimmer C27 läßt sich die Quarzfrequenz exakt auf 32,768 kHz abgleichen. Die Steuersignale des Uhrenbausteins werden über die Z80-DART und STI beschaltet. Der Datenaustausch geschieht dabei seriell über die Leitung DATA O/P. Ein Sekundentakt ist an den Eingang I4 der STI geführt. I4 dient Timer A als Eingang im Event Count Mode. Entsprechend programmiert erzeugt der Ausgang TA des Timers das Motor-On/Off-Signal für die Floppy-Laufwerke, indem er den an I4 anliegenden Sekundenpuls herabzählt und bei Time-Out den Motor ausschaltet.

Die Bus-Schnittstelle

Große Bedeutung wurde dem Bus-Anschluß zugemessen. So sind Daten, Adressen und Steuersignale über vier

Treiber-Bausteine (74LS245) bidirektional gepuffert. Das heißt, es können externe Baugruppen, wie eine CPU oder eine DMA, auf den internen Speicher oder I/O-Baugruppen zugreifen. Der DART- und STI-Baustein sind sogar in der Lage, einen Vektor-Interrupt an eine externe CPU weiterzuleiten.

Die Steuerung des Daten- und Adreßbus wird wieder von einem 256×4-Bit-PROM (IC 06) durchgeführt. In Abhängigkeit der acht Eingangssignale werden vier Ausgangssignale erzeugt, welche die Richtung und den Tri-State-Zustand der Daten- und Adreß-Treiber bestimmen. Die vier Treiber-Bausteine sind grundsätzlich aktiv geschaltet. Wird die Karte als Slave eingesetzt, besteht die Möglichkeit, die Treiber auch in den Tri-State-Zustand zu versetzen.

Das Signal $\overline{\text{BUSAK}}$ bestimmt die Richtung der Adreß- und Steuerleitungen. Ist $\overline{\text{BUSAK}}$ inaktiv (High), sind die Treiber nach außen auf den Bus geschaltet. Die oberen vier Adressen A16...A19 werden dabei getrennt angesprochen. Diese können so bei aktivem $\overline{\text{BUSAK}}$ wahlweise in den Tri-State-Zustand versetzt werden oder nicht (standardmäßig Tri-State).

Um die Richtung des Datenstromes festzulegen, sind eine Reihe von Informationen notwendig. Hierbei muß unterschieden werden, ob die interne CPU aktiv ist oder nicht, ob auf einen internen oder externen Speicher oder eine I/O-Baugruppe zugegriffen wird und ob ein Vektor-Interrupt vorliegt.

Grundsätzlich ist der Datenstrom nach außen geschaltet. Ausnahmen sind Lesebefehle, die auf externe Baugruppen zugreifen, und Interrupt-Acknowlege-Zyklen, die von externen I/O-Baugruppen ausgelöst wurden.

Ist die interne CPU inaktiv, geht der Datenstrom zum Karteninneren. Ausnahmen sind Lesebefehle von einer externen Einheit auf Baugruppen im Karteninneren.

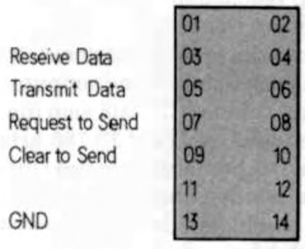


Bild 9. RS-232C-Steckerbelegung. Es sind nicht alle Handshake-Signale herausgeführt

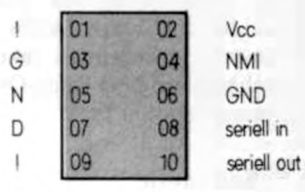


Bild 10. Eine Schnittstelle mit TTL-Signal-Pegeln

Einsatz als I/O-Karte

Sämtliche auf der CPU-Karte befindlichen Baugruppen können auch von einer externen CPU angesprochen werden. Die Baugruppe könnte zum Beispiel auch in einem bestehenden System als I/O-Karte eingesetzt werden. Es sind dann folgende Punkte zu beachten:

1. Der Systemtakt ist extern zuzuführen. Der interne Taktgenerator muß durch Entfernen des Widerstandes R05 deaktiviert werden.

2. Der $\overline{\text{BUSAK}}$-Ausgang ist vom Bus abzutrennen.

3. Die CPU muß aus der Fassung genommen und durch eine Brücke von GND (Pin 29) nach $\overline{\text{BUSAK}}$ (Pin 23) ersetzt werden.

4. Es ist zu überprüfen, ob das $\overline{\text{PWRCLR}}$-Signal auf dem Systembus der Mutter als Open-Collector ausgeführt wurde. Wenn nicht, ist es vom Bus abzutrennen.

Aufbauhinweise und Test

Grundsätzlich sind alle ICs mit Sockeln zu versehen. Unter einigen ICs müssen Blockkondensatoren eingebaut werden. Hierzu sind mechanisch geeignete Fassungen und Kondensatoren auszuwählen. Die Fassungen sind evtl. erhöht einzubauen. Teuer, aber optimal sind da Fassungen mit bereits eingebauten Blockkondensatoren.

In der Bestückungsliste sind einige TTL-Bausteine als Schottky bzw. Low-Power-Schottky ausgewiesen. Das ist ernst zu nehmen, es sind grundsätzlich nur die dort angegebenen Typen einzusetzen. Grundsätzlich ist die Karte nach dem Bestücken optisch zu kontrollieren. Ungewollte Verbindungen sowie kalte Lötstellen und falsch eingesetzte Bauteile sind mit Abstand die häufigsten Fehlerursachen. Vor Einsetzen der ICs wird die Karte an die Versorgungsspannung angeschlossen. Es lassen sich so Kurzschlüsse und falsch eingebaute Kondensatoren entdecken. Die Karte kann danach vollständig bestückt und an die Peripherieeinheiten angeschlossen werden. Läuft der mc-CP/M-Plus-Computer nicht auf Anhieb, empfiehlt sich folgende Vorgehensweise:

Mit einem Oszilloskop werden die Versorgungsspannungen, die beiden Taktsignale sowie der Reset überprüft. Ebenso dürfen sämtliche Signale an der CPU weder Zwischen-Pegel (Kurzschluß zwischen zwei Signalleitungen) noch GND- oder Vcc-Pegel aufweisen.

Läßt sich der Fehler auf diese Weise auch nicht finden, sollte die Platine als I/O-Karte in ein bestehendes System eingesteckt werden und weitergetestet werden. Hier kann dann gezielt der Speicher und die I/O-Baugruppen betrachtet werden.

Hardware-Steckbrief

Zentraleinheit
- 6-MHz-Z80-B-CPU
- 1 MByte Adreßraum

Speicher
- 4-KByte-Boot-EPROM
- 128 KByte RAM

Floppy-Disk-Steuerung
- Anschluß von vier Laufwerken
- 3 und 5,25 oder 8 Zoll
- PLL-Datenseperator
- Write-Precompensation

I/O-Schnittstellen
- 1× Centronics parallel
- 2× RS-232C, vollduplex
- 1× TTL, seriell, vollduplex
- Baudraten programmierbar

Echtzeituhr
- Akkugepuffert
- Angabe von: (Schalt-)Jahr, Monat, Wochentag, Tag, Stunde, Minute und Sekunde

Bus-Schnittstelle
- ECB-Bus, alle Signale bidirektional gepuffert

Frank Oettle, Thomas Reichler

Der mc-CP/M-Plus-Computer

Teil 3: Software

Um in unserem Computer ein professionelles CP/M-Plus-System hochzuziehen, benötigen wir genaue Kenntnis seiner Hard- und Software-Eigenschaften. Diesmal wird die Programmierung der einzelnen System-Komponenten gezeigt und die Eigenschaften eines CP/M-Plus-BIOS erklärt. Natürlich existiert auch schon ein sehr komfortables BIOS für den mc-CP/M-Plus-Computer. Welche Schnittstellen es bedient und was man für Laufwerke daran anschließen kann, wird anschließend besprochen.

Für all diejenigen, die sich ihr BIOS selbst erstellen wollen oder den EPC-Computer für andere Zwecke einsetzen wollen, diene das in *Bild 1* gezeigte Programmbeispiel als Hilfe. Es zeigt die Initialisierung der I/O-Bausteine DART, STI, FDC, des Uhrenbausteins MEM und wie man ihre Funktion in Programme umsetzt. Wagt man sich an eine Modifikation der Routinen, so ist es ratsam, sich die Datenblätter der Bausteine zu besorgen und genau zu studieren. Aus Platzgründen werden nur die grundlegendsten, völlig hardware-abhängigen Routinen gezeigt, doch mit etwas Phantasie läßt sich daraus schon ein einfaches BIOS erstellen.

Das BIOS unter CP/M-Plus

Das BIOS des CP/M-Plus Systems besteht wie unter CP/M-2.2 aus einer Sprungleiste am Anfang des BIOS, die zu den einzelnen Funktionen verzweigt. Im Unterschied zu CP/M-2.2 müssen jetzt 33 Funktionen bedient werden. Die ersten 17 Funktionen decken sich weitgehend mit den von CP/M-2.2 bekannten Funktionen. Die BIOS-Funktionen lassen sich grob in fünf Gruppen einteilen, die an Hand unserer BIOS-Implemantation besprochen werden.

System-Initialisierung

0 BOOT
Boot initialisiert alle Hardeware-Komponenten des Systems (STI, DART, Baudraten einstellen...) und gibt eine Bereitschaftsanzeige aus, die Daten wie verfügbarer Speicherplatz, Uhrzeit und Datum enthält. Danach wird eine Kopie des CCP in Bank 0 angelegt, indem die Datei CCP.COM von Diskette gelesen wird. Dann wird die Kontrolle an die Warm-Boot-Funktion weitergeleitet.

1 WBOOT
Warm-Boot wird immer nach Verlassen eines Anwenderprogramms aufgerufen. Diese Funktion selektiert die TPA-Bank 1, setzt wie unter CP/M-2.2 zwei Sprünge zum BIOS (0000h) und BDOS (0005h) in Bank 1 und lädt schließlich die Kopie des CCPs in Bank 1 ab 100h. Danach erfolgt ein Sprung nach 100h auf den CCP.

20 DEVTBL
Sinn der Funktion DEVTBL ist es, dem BDOS die Adresse der sog. Character-Tabelle zu übergeben. Diese Tabelle enthält alle Informationen über die angeschlossen I/O-Einheiten wie Baud-Rate, Protokoll, Name usw.

21 DEVINI
DEVINI initialisiert eine bestimmte I/O-Einheit neu. Das aufrufende Programm setzt in der zugehörigen Character-Tabelle bestimmte Parameter neu, die DEVINI entsprechend verarbeitet. So difiniert z. B. das BDOS die Baud-Rate für eine serielle Schnittstelle neu.

22 DRVTBL
DRVTBL (Drive-Table) übergibt ähnlich wie DEVTBL die Adresse einer Tabelle mit Informationen über die Anzahl und Art der angeschlossenen Laufwerke. Der erste Tabelleneintrag gehört zu Laufwerk A, der zweite zu Laufwerk B usw. Die Einträge zeigen auf den zugehörigen Disk-Parameter-Header (DPH) des Laufwerks.

Zeichen-Ein- und Ausgabe

In CP/M-Plus gibt es drei logische Ein-/Ausgabeeinheiten: die Konsole (3: CONIN, 4: CONOUT, 2: CONST, 17: CONOST), den Drucker (5: LIST, 15: LISTST) und einen Hilfskanal (7: AUXIN, 6: AUXOUT, 18: AUXIST, 19: AUXOST), der die Reader- und Punch-Einheiten unter CP/M-2.2 ersetzt. Zu jeder der fünf logischen Ein- und Ausgabeeinheiten (CON-IN/OUT, AUX-IN/OUT, LIST) existiert ein 16 Bit breiter Umleitungs-Vektor (I/O Redirection Vector), der das I/O-Byte unter CP/M-2.2 ablöst. Dieser Vektor bildet sozusagen die fünf logischen Ein- und Ausgabeeinheiten auf maximal 16 physikalische Ein- und Ausgabe-Geräte ab. Jedem der 16 Bits ist eine physikalische Einheit zugeordnet. Ein gesetztes Bit des Vektors besagt, daß Zeichen von oder zu der zugehörigen Einheit transferiert werden sollen. Es können auch mehrere Bits gesetzt werden, um gleichzeitig mehrere I/O-Einheiten anzusprechen. Sind zum Beispiel mehrere Konsolen an den Computer angeschlossen, so kann durch Setzen der entsprechenden Bits im Vektor die Ein- und Ausgabe von einer beliebigen Konsole aus erfolgen und die Zeichen werden automatisch auf alle Einheiten ausgegeben. Dabei ist es gleichgültig, ob die Ausgabeeinheit ein Drucker, ein Bildschirm oder ein Kassettenlaufwerk ist. Da für Ein- und Ausgabe getrennte Umleitungs-Vektoren existieren, kann die Zeicheneingabe auch von einer ganz anderen Einheit aus erfolgen als die Ausgabe. In Verbindung mit der DEVINI-Funktion, mit der sich Protokolle und Baudraten im Betrieb ändern lassen, bedeuten diese neuen BIOS-Funktionen eine gewaltige Flexibilitäts-Steigerung gegenüber dem alten CP/M-2.2. Voraussetzung dazu ist natürlich, daß all diese Funktionen auch vollständig im BIOS implementiert sind.

Die Disk-Ein- und -Ausgabe

Die einfachen Disk-Funktionen unterscheiden sich nur unwesentlich von den bekannten Funktionen des 2.2-Systems, da es nicht unbedingt notwendig ist, spezielle Funktionen wie Multi-I/O, Flush usw. zu verwirklichen. Doch gera-

```
CP/M MACRO ASSEM 2.0    #001    EPC-Modul utility subroutines

                        MACLIB Z80
 0100                   ORG 100H

                ; CHIP BASE PORTADRESS EQUATES
                ; ============================
 0000 =         P$STI           EQU     00H
 0010 =         P$CENT          EQU     10H
 0014 =         P$SIO           EQU     14H
 0018 =         P$BANK          EQU     18H
 001C =         P$765$CMD       EQU     1CH
 001D =         P$765$DAT       EQU     1DH

                ;       DART CHANNELS:
                ;       ==============
 0014 =         P$SIOA$DAT      EQU     P$SIO+0H        ;DART A DATA
 0015 =         P$SIOB$DAT      EQU     P$SIO+1H        ;DART B DATA
 0016 =         P$SIOA$CTR      EQU     P$SIO+2H        ;DART A CTR.
 0017 =         P$SIOB$CTR      EQU     P$SIO+3H        ;DART B CTR.

                ;       STI CHANNELS:
                ;       =============
 0000 =         P$IDR           EQU     P$STI+0         ;INDIRECT DATA REG
 0001 =         P$GPIP          EQU     P$STI+1         ;I/O LINES
 0002 =         P$IPRB          EQU     P$STI+2         ;INT PENDING REG B
 0003 =         P$IPRA          EQU     P$STI+3         ;INT PENDING REG A
 0004 =         P$ISRB          EQU     P$STI+4         ;INT SERVICE REG B
 0005 =         P$ISRA          EQU     P$STI+5         ;INT SERVICE REG A
 0006 =         P$IMRB          EQU     P$STI+6         ;INT    MASK REG B
 0007 =         P$IMRA          EQU     P$STI+7         ;INT    MASK REG A
 0008 =         P$PVR           EQU     P$STI+8         ;POINTER/VECTOR REG
 0009 =         P$TABCR         EQU     P$STI+9         ;TIMER A&B CTR REG
 000A =         P$TBDR          EQU     P$STI+10        ;TIMER B DATA
 000B =         P$TADR          EQU     P$STI+11        ;TIMER A DATA
 000C =         P$UCR           EQU     P$STI+12        ;USART CTR
 000D =         P$RSR           EQU     P$STI+13        ;RECEIVER STATUS
 000E =         P$TSR           EQU     P$STI+14        ;TRANSMITTER STATUS
 000F =         P$UDR           EQU     P$STI+15        ;USART DATA


                ;***************************************
                ;*       INIT TABLE DART               *
                ;***************************************

                INIDART:
 0100 0E16              MVI C,P$SIOA$CTR                ; DART A
 0102 0607              MVI B,7
 0104 211001            LXI H,V24ATAB
 0107 EDB3              OTIR

 0109 0E17              MVI C,P$SIOB$CTR                ; DART B
 010B 0607              MVI B,7
 010D EDB3              OTIR
 010F C9                RET

                V24ATAB:
 0110 18                DB 18H          ; CHANNEL RESET
 0111 03C1              DB 3,0C1H       ; WR3: RXENABLE, 8 BITS
 0113 0444              DB 4,44H        ; WR4: X16 CLOCK MODE, 1 STOP, NO PAR.
 0115 05                DB 5
 0116 6A        AWR5:   DB 06AH         ; WR5: 8 BITS, TXENABLE, RTS
                                        ; DTR = 0 -> OUTPUT=1 -> WATCH CS = 1
                V24BTAB:
 0117 18                DB 19H
 0118 0341              DB 3,041H       ; WR3: RXENABLE, 7 BITS
 011A 044F              DB 4,04FH       ; WR4: X16 CLOCK MODE, 2 STOP, PARITY EV
EN
 011C 05                DB 5
 011D AA        BWR5:   DB 0AAH         ; WR5: RTS, 7 BITS, TXENABLE
                                        ; DTR = 1 => OUTP. = 0 -> WATCH CLOCK =
LOW

                ;*************** RS 232 A ******************
                ;INPUT/OUTPUT A CHARACTER VIA DART CHANNEL A
                ;***********************************************

                V24A$OUST:
 011E DB16              IN P$SIOA$CTR                   ; TEST OUTPUT STATE
 0120 E604C8F6FF        ANI 04H ! RZ ! ORI -1 ! RET     ; RET Z=1 IF NOT READY

                V24A$INST:
 0126 DB16              IN P$SIOA$CTR                   ; TEST INPUT STATE
 0128 E601C8F6FF        ANI 1 ! RZ ! ORI -1 ! RET       ; RET Z=1 IF NOT READY

                V24A$IN:
 012E CD2601            CALL V24A$INST                  ; TEST INPUT STATA
 0131 28FB              JRZ V24A$IN                     ; LOOP BACK IF NOT RDY
 0133 DB140000          IN P$SIOA$DAT ! NOP ! NOP       ; ROOM TO CLEAR PARITY
 0137 C9                RET

                V24A$OUT:
 0138 CD1E01            CALL V24A$OUST                  ; TEST OUTPUT STATE
 013B 28FB              JRZ V24A$OUT                    ; WAIT IF BUSY
 013D 79D314C9          MOV A,C ! OUT P$SIOA$DAT ! RET  ; OUTPUT CHAR. IN (C)

                ;****** INIT SERIAL TIMER INTERRUPT CONTROLER STI **********
                ;***********************************************************

                STIINIT:
 0141 216101            LXI H,STITAB                    ; POINTER TO TABLE
 0144 0E01060F          MVI C,P$GPIP ! MVI B,15         ; 15 DIRECT REGISTERS
 0148 7EED79    STIDIR: MOV A,M ! OUTP A                ; OUT DATA
 014B 230C              INX H ! INR C                   ; NEXT DATA
 014D 10F9              DJNZ STIDIR
 014F 0608              MVI B,8                         ;8 INDIRECT REGISTERS
 0151 DB08E6F8  STIINDR:IN P$PVR ! ANI 0F8H
 0155 050004            DCR B ! ORA B ! INR B
 0158 D308              OUT P$PVR                       ;INDIRECT ADDRESS TO PVR
 015A 7ED308            MOV A,M ! OUT P$IDR             ;CONSTANT IN INDIRECT RE
G
 015D 2310F1            INX H ! DJNZ STIINDR            ;NEXT INDIRECT REGISTER
 0160 C9                RET


                ;*************** INIT TABLE STI ****************
                ;***********************************************

                        ; 15 DIRECT ADRESSABLE REGISTER STARTING WITH GPIP
 0161 73        STITAB: DB 0111$0011B   ;GPIB BOOT AUS, TC = 0, WATCH DATA = 1
 0162 0000              DB 0,0          ;IPRB, IPRA CLEAR PENDING INTERRUPTS
 0164 0000              DB 0,0          ;ISRB, ISRA
 0166 0000              DB 0,0          ;IMRB, IMRA
 0168 08                DB 08           ;PVR, SOFTWARE END OF INT
 0169 02                DB 0000$0010B   ;TABCR-TA STOP, TB DELAY PRESC./10
 016A 01                DB 1            ;TBDR->TC= 3.0 MHZ / (2X10X16X9600)
 016B 0A                DB 10           ;TADR -> INPUT = 1 HZ -> MOTO = 10 S
 016C 88                DB 1000$1000B   ;UCR /16; 8 BITS; 1 STOP; NO PAR./DMA
 016D 01                DB 01           ;RCR RECEIVER ENABLE
 016E 05                DB 0000$0101B   ;TSR TRANSMITTER ENABLE
 016F 11                DB 011H         ;UDR -> XON

                        ; 8 INDIRECT ADR. REGISTERS STARTING WITH TCDR
 0170 A2                DB 1010$0010B   ;TCDCR- DELAY MODE,PRESCALE /10, TA RES.
 0171 87                DB 1000$0111B   ;DDR - IN = BIT 6,5,4,3
 0172 60                DB 0110$0000B   ;IERA TIMER A, ACKN. CENTR.
 0173 02                DB 0000$0010B   ;IERB
 0174 00                DB 0000$0000B   ;AER-ACTIVE EDGE REGISTER ALL FALLING
 0175 08                DB 8            ;TCDR -> TC= 3.0 MHZ / (2X16X10X1200)
 0176 01                DB 1            ;TDDR -> TC= 3.0 MHZ / (2X16X10X9600)
 0177 AA                DB 0AAH         ;SCR-SYNCH CHARACTER
```

Bild 1. Programmbeispiele zur Initialisierung des CP/M-Plus-Systems. Die einzelnen Routinen sollen Ihnen als Grundlage für die eigene Verwendung der I/O-Prozessoren dienen

```
                ; ******* TEST KEYBOARD STATUS **********
                ; SWITCH MOTOR FDC OFF IF INT FROM TIMER
                ; ****************************************

 0178 DB03      CSTS:   IN P$IPRA               ; TIME OUT FROM TIMER A ?
 017A CB6F              BIT 5,A
 017C 280A              JRZ KEY$STAT
 017E CBAF              RES 5,A
 0180 D303              OUT P$IPRA              ; RESET PENDING INTERRUPT
 0182 DB09              IN P$TABCR              ; IF TIME OUT STOP TIMER
 0184 E60FD309          ANI 0FH ' OUT P$TABCR

                ;********** KEYBOARD *************
                ; INPUT/OUTPUT CHARACTERS TO KEY
                ; USE REVERSE XON-PROTOCOLL
                ;*********************************

                KEY$STAT:                       ; TEST RECEIVER STATE
 0188 DB0DE680C8        IN P$RSR ' ANI 80H ' RZ ; RET Z=1 IF NO CHAR.
 018D F6FFC9            ORI -1 ' RET            ; ELSE RET Z=0

 0190 DB0DE6F8  KEYINP: IN P$RSR ' ANI 0F0H     ; CHARACTER AVIABLE ?
 0194 28FA              JRZ KEYINP              ; NO -> LOOP BACK
 0196 CB7F              BIT 7,A                 ; BUFFER FULL ?
 0198 200B              JRNZ KEYREC             ; SKIP IF SO
 019A 3E01D30D          MVI A,01H ' OUT P$RSR   ; ELSE ERRORS, RESET IT
 019E 0E11CDAF01        MVI C,11H ' CALL KEYOUTP ; OUTPUT XON
 01A3 18EB              JR KEYINP               ; NEW RETRY

 01A5 DB0F47    KEYREC: IN P$UDR ' MOV B,A      ; SAVE CHARACTER
 01A8 0E11CDAF01        MVI C,11H ' CALL KEYOUTP ; SEND XON TO KEY
 01AD 78C9              MOV A,B ' RET           ; RET CHARACTER IN (A)

 01AF DB0ECB7F  KEYOUTP:IN P$TSR ' BIT 7,A      ; TRANSMITTER EMPTY ?
 01B3 28FA              JRZ KEY$OUTP            ; NOT EMPTY ? -> LOOP BA
CK
 01B5 79D30FC9          MOV A,C ' OUT P$UDR ! RET ; EMPTY - OUTPUT CHARACT
ER


                ; ****************** CENTRONICS ******************
                ; OUTPUT A CHARACTER IN A TO CENTRONICS PRINTER
                ;************************************************

 01B9 3E10D316  CENTOUT:MVI A,10H ' OUT P$SIOA$CTR ; RESET EXT. STATUS INT.
 01BD CDDD0128F7        CALL CENT$STAT ' JRZ CENT$OUT ; CENTRONICS READY ?
 01C2 79D310            MOV A,C' OUT P$CENT     ; SEND CHAR
 01C5 DB01CB8F          IN P$GPIP ' RES 1,A
 01CC D301              OUT P$GPIP              ; GIVE STROBE
 01CE CBCF              SETB 1,A
 01D0 D301              OUT P$GPIP              ; RESET STROBE
 01D5 C9                RET

                CENT$STAT:
 01DD DB16E608          IN P$SIOA$CTR ! ANI 1000B ; BUSY ?  -> DCD = LOW
 01E1 C8                RZ
 01E2 F6FFC9            ORI -1 ' RET            ; READY IF DCD = 1


                ;*********************************
                ; READ/WRITE TIME ENTRY POINTS
                ; INTERFACE TO MEM E 050 - 16
                ;*********************************

 01E5 CD3802    RDTIME: CALL CONTCMD            ; GIVE CONT. CMD WORD
 01E8 CD4B02            CALL CLKTIME            ; ONE MORE CLOCK RDATA
 01EB DB08E6F8          IN P$PUR ' ANI 0F8H     ; MAKE DATA PIN = INPUT
 01EF F606D308          ORI 06H ' OUT P$PUR     ; SELECT IDR 6 => DDR
 01F3 DB00CB87          IN P$IDR ' RES 0,A      ; BIT 0 DDR = 0 -> INPUT
 01F7 D300              OUT P$IDR

 01F9 CD4B02            CALL CLKTIME            ; ONE MORE CLOCK 5
 01FC 0607              MVI B,7                 ; 7 BYTES TO READ
 01FE 213102            LXI H,DATE
 0201 0E011600  RDTIM0: MVI C,1 ' MVI D,0       ; START WITH LSB
 0205 DB01CB47  RDTIM1: IN P$GPIP ' BIT 0,A     ; TEST FOR ZERO
 0209 2803              JRZ RDTIM2              ; SKIP OUT IF SO
 020B 7AB157            MOV A,D ' ORA C ' MOV D,A ; ELSE SET BIT POSITION
 020E CD4B02    RDTIM2: CALL CLKTIME            ; NEXT BIT
 0211 CB01              RLCR C                  ; NEXT POSITION
 0213 30F0              JRNC RDTIM1             ; LOOP BACK
 0215 77                MOV M,A                 ; SAVE BYTE
 0216 23                INX H                   ; NEXT POSITION
 0217 10E0              DJNZ RDTIM0

                ; MAKE DATA PIN TO OUTPUT
 0219 DB00CBC7          IN P$IDR ' SETB 0,A     ; BIT 0 DDR = 1 => OUTPU
T
 021D D300              OUT P$IDR

                ; RESET CS
 021F 3E05D316  PREMEM: MVI A,5 ' OUT P$SIOA$CTR
 0223 3A1601            LDA AWR5                ; GET WR 5 SIO A
 0226 CBBF              RES 7,A
 0228 D316              OUT P$SIOA$CTR          ; RESET CS
 022A DB01CBC7          IN P$GPIP ' SETB 0,A    ; DATA PIN = 1
 022E D301C9            OUT P$GPIP ' RET

 0231           DATE:   DS 7

                CONTCMD:        ; GIVE CONTINUOUS COMMAND TO MEM
                                ; ACTIVATE CS FORM MEM
 0238 3E05D316          MVI A,5 ' OUT P$SIOA$CTR ; SELECT WR5 SIOA
 023C 3A1601            LDA AWR5                ; GET WR5 SIOA
 023F CBFF              SETB 7,A                ; ACTIVATE DTR -> CS
 0241 D316              OUT P$SIOA$CTR          ; ACTIVATE IT
 0243 0603              MVI B,3
 0245 CD4B02    CONTLOP:CALL CLKTIME            ; 3 TIMES CLOCKS
 248 10FB               DJNZ CONTLOP            ; -> CONTINUOUS COMMAND
 024A C9                RET

                CLKTIME:        ; GIVE ONE POSITIVE GOING CLOCK PULSE
 0248 F5C5              PUSH PSW ' PUSH B
 024D 0607              MVI B,7                 ; DELAY
 024F 10FE      CLKTIM1:DJNZ CLKTIM1
 0251 3E05D317          MVI A,5 ' OUT P$SIOB$CTR ; SELECT WR5
 0255 3A1D01CBBF        LDA BWR5 ' RES 7,A
 025A D317              OUT P$SIOB$CTR          ; CLOCK = HIGH
 025C 0607              MVI B,7                 ; DELAY
 025E 10FE      CLKTIM2:DJNZ CLKTIM2
 0260 3E05D317          MVI A,5 ' OUT P$SIOB$CTR ; SELECT WR5
 0264 3A1D01CBFF        LDA BWR5 ' SETB 7,A
 0269 D317              OUT P$SIOB$CTR          ; CLOCK = LOW
 026B C1F1C9            POP B ' POP PSW ' RET

 026E CD3802    WRTIME: CALL CONTCMD
 0271 DB01CB87          IN P$GPIP ' RES 0,A     ; SELECT WRITE
 0275 D301CD4B02        OUT P$GPIP ' CALL CLKTIME
 027A 0607213102        MVI B,7 ' LXI H,DATE    ; 7 BYTES
 027F C50608    WRTIM0: PUSH B ' MVI B,8
 0282 7E                MOV A,M                 ; 8 BITS, GET BYTE
 0283 07F5      WRTIM1: RLC ' PUSH PSW          ; MSB INTO CARRY, SAVE P
ATT.
 0285 DB01CB87          IN P$GPIP ' RES 0,A     ; ASSUME BIT IS ZERO
 0289 3002              JRNC WRTIM2             ; SKIP IF SO
 028B CBC7              SETB 0,A                ; ELSE SET BIT
 028D D301CD4B02WRTIM2: OUT P$GPIP ' CALL CLKTIME ' POP PSW
 0293 10EE              DJNZ WRTIM1             ; NEXT BIT
 0295 C123              POP B ' INX H           ; NEXT BYTE
 0297 10E6              DJNZ WRTIM0
 0299 C31F02            JMP PREMEM
```

```
                ;*****************************************************
                ;FLOPPY DISK INTERFACE ROUTINES USING UPD 765 CTRL.
                ;*****************************************************

029C 0600       INI765: MVI B,0
029E 10FE               DJNZ INI765+2                   ; DELAY
02A0 DB1CFE80           IN P$765$CMD ' CPI 80H          ; REQUEST FOR MASTER ?
02A4 2804               JRZ SPECIFY                     ; NO -> SKIP
02A6 DB1D               IN P$765$DAT                    ; ELSE READ DATA
02A8 18F2               JR INI765                       ; AND TRY IT AGAIN

02AA 21CC02     SPECIFY:LXI H,SPEC5-1                   ; SPECIFY PARAMETER
02AD 010303             LXI B,0303H                     ; SPECIFY CMD. 3 BYTES
02B0 CD2A03             CALL CMDFD1                     ; SEND COMMAND

                ; RECALIBRATE DRIVE TO TRACK ZERO
02B3 010702     RECAL:  LXI B,0207H                     ; RECAL.CMD =07 - 2 BYTE
S
02B6 CDF802             CALL MOTO

                ; SENSE DRIVE INTERUPT STATUS
02B9 010801     SENSE:  LXI B,0108H                     ; COMMAND = 08 - 1 BYTE
02BC CD2703             CALL CMDFD
02BF CD3A03             CALL NEXT                       ; FETCH RESULT
02C2 47                 MOV     B,A                     ; SAVE IT
02C3 FE80               CPI     80H                     ; INVALID CMD.?
02C5 C43A03             CNZ     NEXT                    ; NO -> NEXT RESULT
02C8 CB68               BIT     5,B                     ; SEEK FINISHED ?
02CA 28ED               JRZ     SENSE                   ; NO -> LOOP BACK
02CC C9                 RET                             ; YES -> ALL DONE


                ;************ SPECIFY PARAMETER FOR TEAC FD-55B ****************
                ;*****************************************************************

02CD DF         SPEC5   DB      11011111B
02CE 13                 DB      00011011B       ;SPECIFY PARAM. 5 1/4 TEAC B
                                                ;HLT = 36 MS, HUT = 480 MS
                                                ;SRT = 6 MS, NON-DMA MODE

                ;********** COMMAND TABLE FOR UPD 765 ***************
                ;****************************************************

02CF 46         CMDTAB: DB      046H    ; MFM READ COMMAND
02D0 00         UNIT:   DB      0       ; HEAD0/UNIT A
02D1 00         TRACK:  DB      0       ; TRACK 0
02D2 00         HEAD:   DB      0       ; HEAD 0
02D3 01         SECTOR: DB      1       ; SECTOR 1
02D4 02         N:      DB      2       ; N=2 512BYTES
02D5 0A         EOT:    DB      10      ; EOT, 10 SECTORS PER TRACK
02D6 10         GPL:    DB      10H     ; GPL GAP LENGTH
02D7 FF         DTL:    DB      -1      ; DTL DATA TERMINAL LENGTH

02D8            RESLTB: DS 7            ; RESULT TABLE FOR FDC
02DF            CMDCODE:DS 1            ; COMMAND CODE FOR 765
02E0            ZDMA:   DS 2            ; CURRENT DMA ADRESS

                ;************ SENSE DRIVE STATUS ****************
                ;************************************************

02E2 010402     SDRS:   LXI B,0204H             ; SDRS.CMD = 04 - 2 BYTES
02E5 CD2703             CALL CMDFD
02E8 CD3A03C9           CALL NEXT ! RET

                ;************** SEEK ****************
                ; SEEK TO TRACK IN A, DRIVE IN UNIT
                ;************************************

02EC 32D102     SEEK:   STA TRACK
02EF 010F03             LXI B,030FH             ; COMMAND SEEK = 0F - 2 BYTES
02F2 CDF802             CALL MOTO
02F5 C3B902             JMP SENSE

                ;******************** MOTOR ON ******************************
                ; WAIT UNTIL DISK READY, THEN TRANSMIT CMD. IN BC TO FDD
                ;************************************************************

02F8 C5         MOTO:   PUSH    B               ; SAVE CMD.
02F9 DB09       MOTO1:  IN P$TABCR              ; GET TIMER AB CTR
02FB E60F               ANI 0FH                 ; STOP TIMER A
02FD D309               OUT P$TABCR
02FF DB08               IN P$PVR                ; RESET TIMER A
0301 F607D308           ORI 07H ' OUT P$PVR     ; ACCESS IDR NR. 7
0305 DB00               IN P$IDR                ; GET TCDCTR
0307 CBFF               SETB 7,A
0309 D300               OUT P$IDR               ; RESET TIMER A -> MOTOR GOES ON
030B CB8F               RES 7,A
030D D300               OUT P$IDR               ; RUN TIMER A
030F 3E0A               MVI A,10                ; 10 SECONDS FOR MOTOR ON
0311 D30B               OUT P$TADR              ; SET TIMER A
0313 DB09               IN P$TABCR              ; GET TIMER AB CTR. REGISTER
0315 F680               ORI 80H                 ; SET TIMER A TO EVENT COUNT MOD
E
0317 D309               OUT P$TABCR             ; ENABLE TIMER A
0319 DB03               IN P$IPRA               ; PENDING INT. REG A
031B CBAF               RES 5,A
031D D303               OUT P$IPRA              ; RESET INT TIMER A

031F CDE202CB6F         CALL SDRS ' BIT 5,A     ; DRIVE READY ?
0324 28D3               JRZ MOTO1               ; NO->WAIT
0326 C1                 POP B

                ; ************** COMMAND TO UPD 765 ****************
                ; TRANSMITT COMAND IN BC TO FLOPPY DISK CONTROLER
                ; B=NUMBER OF BYTES TO TRANSMITT, C = COMMAND
                ; ************************************************

0327 21CF02     CMDFD:  LXI H,CMDTAB
032A DB1C       CMDFD1: IN P$765$CMD
032C E6C0FE80           ANI 0C0H' CPI 80H
0330 20F8               JRNZ CMDFD1             ; REQUEST FOR MASTER ?
0332 79D31D             MOV A,C' OUT P$765$DAT  ; SEND COMMAND
0335 234E               INX H ' MOV C,M
0337 10F1               DJNZ CMDFD1             ; NEXT COMMAND
0339 C9                 RET


                ;************ NEXT RESULT *************
                ; READ NEXT RESULT BYTE FROM FDC
                ;**************************************

033A DB1C       NEXT:   IN P$765$CMD
033C E6C0FEC0           ANI 0C0H' CPI 0C0H
0349 20F8               JRNZ NEXT               ; REQUEST FOR MASTER ?
0342 DB1D               IN P$765$DAT
0344 C9                 RET


                ;********** FETCH RESULTS FROM FDC ************
                ;       CHECK FOR R/W ERRORS, IF ANY Z=0
                ;**********************************************

0345 0606       RESULT: MVI B,6         ; 7 RESULTS
0347 CD3A03             CALL NEXT       ; FETCH IT
034A 21D802             LXI H,RESLTB
034D 77                 MOV M,A         ; STORE IT
034E E6C0               ANI 0C0H        ; ANY ERROR IN STATUS 0
0350 4F                 MOV C,A
0351 CD3A03     RESLOP: CALL NEXT
0354 2377               INX H ' MOV M,A
0356 10F9               DJNZ RESLOP
0358 79                 MOV A,C         ; ERROR REPORT STATUS 0
0359 B7C9               ORA A ' RET     ; RET WITH ERROR IN A
```

de diese komplexen Funktionen steigern die Leistung des neuen Betriebssystems erheblich. Daher wurden sie auch im mc-CP/M-Plus-BIOS sorgfältig verwirklicht. Dazu kommt noch eine Auto-Disk-Select-Funktion, die 40 verschiedene Diskformate automatisch erkennt (!). Es besteht auch die Möglichkeit, ein spezielles Disk-Format mit all den zugehörigen Parametern selbst zu definieren und so sein eigenes neues BIOS zu schaffen (wie, das wird im nächsten Heft beschrieben). Eine zusätzliche Geschwindigkeitssteigerung bringt neben dem Deblocking-Algorithmus, der es erlaubt, Sektorgrößen von 128 Byte aufwärts zu verarbeiten, der sog. BIOS-Track-Puffer, der eine ganze Spur einer Diskette zwischenspeichern kann, um beim Schreiben oder Lesen die Daten in einem Durchgang sehr schnell transferieren zu können.

```
                ; ************************************************
                ; ****** SIMPLE READ & WRITE ENTRY POINTS **********
                ; R/W IN NON DMA MODE A 512 BYTE SECTOR
                ; R/W ONE SECTOR IN (SECTOR),
                ; R/W TRACK IN (TRACK), SEEK-COMMAND GIVEN
                ; R/W HEAD IN (HEAD) AND (UNIT)
                ; R/W DRIVE NUMBER IN (UNIT)
                ; DESTINATION/SOURCE IN (ZDMA)
                ; ************************************************


                ;******* WRITE SELECTED SECTOR TO FDD *******
                ;************************************************

0358 11         WRITEIT:DB 11H          ; LXI D,...
035C EDA3               OUTI            ; WRITE DATA
035E 3E05               MVI A,05H       ; 765 WRITE COMMAND IN A
0360 1805               JR RWIT

                ;******* READ SELECTED RECORD FROM FDD ******
                ;************************************************

0362 11         READIT: DB 11H
0363 EDA2               INI             ; READ DATA
0365 3E06               MVI A,6         ; 765 READ COMMAND IN A

0367 32DF02     RWIT:   STA CMDCODE             ; COMAND CODE FOR READ/WRITE
036A ED53C003           SDED INIMFM1            ; SET INI / OUTI OP-CODES
036E ED53CD03           SDED INIMFM2
0372 060A       RWLOP:  MVI B,10                ; RETRY-COUNTER
0374 C5         RWOP:   PUSH B
0375 F3                 DI
0376 119E03D5           LXI D,TERMI ' PUSH D    ; SET TERMI ONTO STACK
037A DD21B603           LXIX RW765M             ; START ADRESS
037E FD219703           LXIY ENDRW              ; RETURN ADRESS
0382 0609               MVI B,9                 ; 9 COMMANDS
0384 3ADF02             LDA CMDCODE
0387 4F                 MOV C,A                 ; COMMAND IN C
0388 3ACF02B14F         LDA CMDTAB ' ORA C ' MOV C,A    ; COMMAND READ/WRITE IN
C
038D CDF802             CALL MOTO               ; TRANSFER COMMAND
0390 2AE002             LHLD ZDMA               ; DMA ADRESS IN (HL)
0393 0E1D               MVI C,P$765$DAT
0395 DDE9               PCIX                    ; JUMP TO READ/WRITE 765

0397 DB81F604   ENDRW:  IN P$GPIP ' ORI 01000B
039B D301               OUT P$GPIP              ; TERMINATE COUNT
039D D1                 POP D                   ; RESET STACK ADRESS IF NOT USED
039E DB81F604   TERMI:  IN P$GPIP ' ORI 01000B  ; ENTRY POINT TO ABORT R/W
03A2 D301               OUT P$GPIP              ; GIVE TERMINATE COUNT
03A4 CB97               RES 2,A
03A6 000000             NOP' NOP' NOP           ; DELAY
03A9 D301               OUT P$GPIP              ; RESET TC
03AB FB                 EI                      ; ENABLE INT.
03AC CD4503             CALL RESULT             ; ENTER RESULT PHASE
03AF C1C8               POP B ' RZ              ; NO ERRORS -> RET
03B1 10C1               DJNZ RWOP               ; TRY IT AGAIN IF ERRORS
03B3 3E01C9             MVI A,1 ' RET           ; UNRECOVERABLE ERROR

                ;*****************************************
                ; PHYSICAL READ/WRITE ENTRYS NON-DMA MODE
                ;*****************************************

                ;*****************************************
                RW765M: ; HL = DESTINATION, C = PORT,
                ;*****************************************

03B6 0600               MVI B,0         ; 256 BYTES
03B8 DB1C       RWMFM:  IN P$765$CMD
03BA 07                 RLC             ; BIT 7 IN CARRY
03BB 30FB               JRNC RWMFM      ; REQUEST FOR MASTER ?
03BD 0707               RLC ' RLC
03BF D0                 RNC             ; EXECUTION MODE ?
03C0 EDA2       INIMFM1:INI             ; INI IF READ DATA, OUTI IF WRITE DATA
03C2 C2B803             JNZ RWMFM       ; REPEAT TIL DONE
03C5 DB1C       RWMFM2: IN P$765$CMD    ; NEXT 256 BYTES, REG B IS ZERO
03C7 07                 RLC             ; BIT 7 IN CARRY
03C8 30FB               JRNC RWMFM2     ; REQUEST FOR MASTER ?
03CA 0707               RLC ' RLC
03CC D0                 RNC             ; EXECUTION MODE ?
03CD EDA2       INIMFM2:INI
03CF C2C503             JNZ RWMFM2      ; REPEAT TIL DONE
03D2 FDE9               PCIY            ; JUMP TO ENDRW

03D4                    END
```

Die einzelnen Disk-Funktionen

8 HOME
Die Home-Funktion ist identisch mit der Anwahl von Track 0 über die SETTRK-Funktion.

9 SELDSK
Wie unter dem 2.2-System dient SELDSK der Anwahl des gewünschten Laufwerks. Das aufrufendes BDOS gibt hier jedoch noch einen Hinweis, ob es sich um einen Erst-AUFRUF des Laufwerks handelt. Dann wird das Format der eingelegten Diskette ermittelt. In diesem Fall wird eine LOGIN-Prozedur aufgerufen die das Format bestimmt.

10 SETTRK
Set-Track übernimmt einfach die gewünschte Spur und legt den Wert ab.

11 SETSEC
Ähnlich wie SETTRK übernimmt Set-Sector den gewünschten Sektor und speichert ihn ab.

12 SETDMA
Set-DMA speichert nur die DMA-Adresse ab.

13 READ
Hier beginnt das BIOS, den gewünschten Sektor zu übertragen. Zuerst wird geprüft, ob der Sektor sich nicht schon im Track-Puffer befindet. Ist dies der Fall, so findet eine Übertragung ohne jeden Disk-Zugriff statt. Andernfalls wird geprüft, ob sich der Laufwerk-Kopf nicht schon auf der gewünschten Spur befindet, um unnötige Such-Operationen zu vermeiden.

14 WRITE
Die Write-Funktion durchläuft die gleichen Routinen wie READ unter Vertauschung der Datenrichtung.

16 SECTRN
Diese Funktion ist schon vom CP/M-2.2-System her bekannt und übernimmt die Umrechnung der logischen in die physikalische Sektornummer mit Hilfe einer Übersetzungstabelle.

23 MULTIO
MULTIO gibt an, daß die nachfolgenden Read/Write-Befehle mehrere Sektoren umfassen, die sequentiell zwischen Speicher und Diskette transferiert werden.

24 FLUSH
Flush dient zum Schreiben noch nicht gesicherter Zwischenspeicherbereiche auf Diskette. Diese Puffer dienen in der Regel dem Blocking und Deblocking, doch auch im Track-Puffer können sich noch nicht gesicherte Daten befinden, die von vorhergegangenen Write-Befehlen zwischengespeichert wurden. Flush bewirkt beim mc-CP/M-Plus-Computer, daß der Track-Puffer – falls beschrieben – auf Diskette gesichert und sein RAM-Inhalt für ungültig erklärt wird.

Speicherselektion und Transfers

Die folgenden vier BIOS-Funktionen sind für den CP/M-2.2-Anwender völlig neu. Sie dienen der Anwahl einer Speicherbank oder dem schnellen Datentransfer zwischen verschiedenen Speicherblöcken und stellen somit einen erheblichen Teil des Fortschritts beim neuen CP/M-Plus dar.

25 MOVE
MOVE wird aufgerufen, um größere Datenmengen im Speicher zu transferieren. Der Transfer kann dabei innerhalb ein und derselben Speicherbank oder zwischen verschiedenen Speicherbänken stattfinden. Bei unserem Computer wird dazu der Z80-Block-Move-Befehl LDIR eingesetzt. Muß zwischen verschiedenen Banks transferiert werden, so werden die Daten in Blöcken zu je 128 Byte über einen Zwischenpuffer kopiert, indem zwischen Start- und Zielbank umgeschaltet wird. Besonders wirkungsvoll ist dieser Befehl, falls der Transfer per DMA erfolgt, wobei es möglich ist, direkt zwischen verschiedenen Banks zu transferieren.

27 SELMEM
Diese Funktion dient zum Umschalten zwischen verschiedenen Speicherbänken um dort Programme auszuführen.

28 SETBNK
Beim Aufruf von Set Bank wird eine Speicherbank definiert, in die Daten beim Lesen vom Diskettenlaufwerk geschrieben werden oder aus der beim Schreiben gelesen werden soll. Die durch SELMEM spezifizierte Speicherbank für die Programmausführung wird von diesem Befehl nicht beeinflußt.

29 XMOVE
XMOVE wird aufgerufen, falls die MOVE-Funktion zwischen verschiedenen Speicherbänken transferieren soll (Interbank Move). Es werden dabei Start- und Zielbank spezifiziert. Wird die MOVE-Funktion ohne vorhergehendes XMOVE aufgerufen, so sollen die Daten innerhalb der derzeitig selektierten Bank transferiert werden.

Uhrzeit und Datum

Die BIOS-Funktion 26 dient zum Setzen und Lesen von Uhrzeit und Datum. Das Datum wird dabei in Tagen seit dem 1. Januar 1978 gezählt. Da der auf unserem Computer eingesetzte Uhrenbaustein auch das Datum getrennt nach Jahr, Monat und Tag verarbeitet, müssen diese Angaben von der Software noch entsprechend umgerechnet werden.

Die residenten BIOS-Funktionen

Auf Grund der Aufteilung des Speichers und des CP/M in „banked"- und „non-banked"-Bereiche ist es Anwenderprogrammen nicht mehr ohne weiteres möglich, die BIOS-Funktionen direkt aufzurufen. Der BIOS-Sprung-Vektor liegt ab 0F600h im Common-Bereich und ist von jeder Bank aus erreichbar; doch die Routinen selbst liegen nur teilweise im Common-Bereich. Folgende BIOS-Funktionen können jederzeit von Anwenderprogrammen aufgerufen werden, da sie im Common-Bereich liegen oder selbständig auf die Systembank 0 umschalten:

WBOOT, DEVTBL, DEVINI, CONST, CONIN, CONOUT, LIST, LISTST, CONOST, AUXOUT, AUXIN, AUXIST, AUXOST, MOVE, SELMEM, TIME.

Alle anderen BIOS-Funktionen dürfen von der TPA-Bank 1 nicht aufgerufen werden. Ist dies dennoch notwendig, so steht dem Programmierer dafür die BDOS-Funktion 50 (Direkt BIOS-Call) zur Verfügung.

Interrupts

Um mit Interrupts in einem „gebankten" System arbeiten zu können gibt es folgende Regeln: Die Z80-CPU arbeitet unter unserem BIOS in der Vektor-Interrupt-Betriebsart IM 2. Der Interrupt-Vektor der CPU zeigt dabei auf die oberste Bank 0FFh. Wichtig ist, daß die Interrupt-Vektor-Tabelle und die Interrupt-Service-Routine im Common-Bereich des Speichers stehen. Sie müssen also von jeder Bank aus erreichbar sein. Handelt es sich um eine längere Interrupt-Routine, so kann deren Hauptteil auch im „gebankten" Speicherbereich stehen. Der Einsprungpunkt des Interrupts jedoch muß im Common-Bereich liegen, und es muß dann auf die „Interrupt-Bank" umgeschaltet werden. Nach dem Verlassen der Routine muß wieder auf die ursprüngliche Bank zurückgeschaltet werden. Der ursprüngliche Wert des Stackpointers muß beim Verlassen der Routine restauriert werden.

Device	Name	Gerät	Protokoll	Baud	Bit	Stop	Parity	Dir
0	CRT	RGB-Term.	rev. XON	9600	8	1	none	I/O
1	V24-A	RS 232 A	XON/XOFF	9600	8	1	none	I/O
2	V24-B	RS 232 B	XON/XOFF	1200	7	2	even	I/O
3	CENTR	Centronic	none	none	8	0	none	O
4	reserviert							
5	TTLSER	STI-ser.	XON/XOFF	9600	8	1	none	I/O
6	DIABLO	RS 232 B	ETX/ACK	1200	7	2	even	I/O

Bild 2. Die Ein-/Ausgabe-Geräte des mc-CP/M-Plus-Computers

Die I/O-Geräte des mc-BIOS

Das BIOS verwaltet sechs Ein-/Ausgabeeinheiten (*Bild 2*). Beim Kaltstart des Systems sind natürlich bestimmte Voreinstellungen für die Zuordnung zu den logischen CP/M-Einheiten, Baudraten und Protokolle getroffen. Im Betrieb lassen sich diese Einstellungen mit dem DEVICE-Kommando jederzeit ändern. Sollen jedoch schon beim Kaltstart andere Einstellungen wirksam sein, so muß das USRDEF0-File des BIOS entsprechend modifiziert und eine neue CPM3.SYS-Datei geschaffen werden. Wie das geht, zeigen wir in der nächsten Ausgabe von mc.

Die seriellen Schnittstellen

Der EPC-Computer stellt drei bidirektionale serielle Schnittstellen zur Verfügung: Zwei serielle RS-232/V.24-Schnittstellen dienen zum Anschluß eines Terminals, Druckers oder anderer externer

I/O-Einheiten. Die RS232-Schnittstelle (Teil A des DART) arbeitet normalerweise mit einer Baudrate von 9600 Baud. Es werden acht gültige Datenbits ohne Parität und ein Stopbit übertragen. Die Zeichen XON und XOFF werden berücksichtigt. Damit eignet sich dieser Kanal zum Anschluß der meisten modernen RS 232 – Geräte, so zum Beispiel auch der mc-Terminals.
Die RS232-Schnittstelle B ist für den Anschluß langsamer Einheiten gedacht. Die Baudrate ist auf 1200 Baud beschränkt. Es werden 7 Datenbits und 2 Stopp-Bits mit gerader Parität übertragen. Als Protokoll werden XON/XOFF (V.24-B) oder ETX/ACK (Diablo) verwendet. Beim Lesen von der Schnittstelle wird das oberste Datenbit (Bit 7) auf Null gesetzt.

Die dritte serielle Schnittstelle kann zum Beispiel für den Anschluß einer seriellen Tastatur mit TTL-Pegel genutzt werden. Dabei werden 8 Datenbits und 1 Stopp-Bit ohne Parität übertragen. Die Übertragungsrate beträgt 9600 Baud. Dabei wird ein spezielles Protokoll benutzt: Wird ein Zeichen von der Schnittstelle empfangen, so wird ein XON-Zeichen zurückgesendet, sobald die Empfangsbereitschaft für ein neues Zeichen wieder gegeben ist. Wird dieses Protokoll nicht benutzt, so ist die Leitung TxD (Transmit Data) zum I/O-Gerät nicht anzuschließen. Das Protokoll ist nur bei sehr schnell arbeitenden Tastaturen erforderlich, die ganze Zeichenketten auf einmal ausgeben können.

Es gibt ein RGB-Terminal

Wird das RGB-Grafik-Terminal, das noch geschildert wird, an den Computer angeschlossen, so wird es als Konsole über den Namen CRT angesprochen. Die Steuer-Software ist dabei schon in das BIOS eingebunden. Auszugebende Zeichen werden an das RGB-Terminal über den ECB-Bus übergeben. Der Empfang der Zeichen läuft über den seriellen TTL-Kanal.

Die Voreinstellung der I/O-Geräte

Beim Kaltstart des Systems prüft das BIOS, ob das RGB-Terminal an den EPC-Computer angeschlossen ist. Verläuft der Test positiv, so erfolgt automatisch folgende Voreinstellung:

CON = CRT (RGB-Grafik-Terminal)
AUX = V.24-A (RS-232 über ext. Terminal)
LST = CENTR (Paralleler Centronics-Drucker)

Erkennt das BIOS das RGB-Grafik-Terminal nicht, so wird folgende Voreinstellung getroffen
CON = V.24-A (RS-232 über ext. Terminal)
AUX = V.24-B (RS-232 zweites ext. Terminal oder Drucker)
LST = CENTR (Paralleler Centronics-Drucker)
Ist diese automatische Voreinstellung unerwünscht, so kann man selbst eine definieren (das wird in der nächsten Ausgabe besprochen).

Die Baudraten

Den drei seriellen Schnittstellen können jeweils getrennte Baudraten zwischen 110 und 19600 Baud zugeordnet werden. (1800, 3600 und 7200 Baud sind dabei Näherungswerte.) Device 0 und 5 und Device 2 und 6 unterscheiden sich nur in der Art des Protokolls, ihre Baudrate muß immer identisch gewählt werden, da es sich um dieselben physikalischen I/O-Einheiten handelt. Bei Device 6 (Diablo) wird von einer Puffergröße des Empfangskanals von 159 Zeichen ausgegangen.

Die Laufwerks-Routinen

Bei der Auslegung der Disk-I/O-Routinen wurde besonders Wert auf eine rasche Abwicklung des Datentransfers von und zum Laufwerk gelegt. Dazu sind die Kopfladezeiten, die Motorsteuerung und die Suchkommandos genau aufeinander abgestimmt, so daß unnötiges Warten auf die Bereitschaft der Laufwerke weitgehend vermieden wird. Das BIOS kennt stets von jedem der vier Laufwerke die Position der Schreib-Lese-Köpfe und ihren Ladezustand. Unnötige Seek-Kommandos mit entsprechend notwendigen Zeitverzögerungen zum Absenken und zur Beruhigung des Schreib-Lese-Kopfes werden also vermieden. Das BIOS unterstützt die Funktion MULTIO vollständig; auch Disketten mit einem Interleaving-Faktor größer als 1 sind davon nicht ausgenommen. Nach dem Aufruf der Funktion und einem entsprechenden Read/Write-Kommando werden in einem Diskettenzugriff und einer Laufwerksumdrehung mehrere Sektoren (maximal eine Spur) transferiert. Dies beschleunigt die Disketten-Operationen erheblich.
Sehr viel Wert wurde beim Erstellen des BIOS auf eine komfortable Auto-Disk-Selekt-Funktion gelegt. Man kann an jede der vier Laufwerksnummern verschiedene Laufwerke anschließen und verschieden formatierte Disketten einlegen. Parameter wie Aufzeichnungsdichte, Spurdichte, Anzahl der Seiten und Sektoren werden für jedes Laufwerk getrennt ermittelt und verarbeitet! Zudem besteht die Möglichkeit, auf einem Laufwerkstyp Formate anderer Laufwerke zu lesen oder zu schreiben.

Diskettenwechsel

Wird auf ein Laufwerk zum ersten Mal zugegriffen, so führt das BIOS für dieses Laufwerk eine LOGIN-Prozedur durch, in der das Format der eingelegten Diskette ermittelt wird. Ein Diskettenwechsel ist unter CP/M-Plus jederzeit auch ohne CTRL-C möglich, wenn das Format nicht gewechselt wird. Wird ein anderes Format eingelegt, so kann es natürlich zu Schreib-Lese-Fehlern kommen: Erkennt das BIOS einen Fehler, so untersucht es, ob der Fehler auf neues Format zurückzuführen ist. Dann wird automatisch ein neues LOGIN für dieses Laufwerk durchgeführt. Um jedoch diesen unsicheren Betriebszustand zu vermeiden, sollte nach dem Wechsel auf ein neues Diskettenformat CTRL-C betätigt werden. Dies führt dazu, daß für alle Laufwerke ein LOGOUT durchgeführt wird. Beim ersten Zugriff auf das Laufwerk wird das Format neu ermittelt.

Disk-Anwahl

Die BIOS-Funktion 9 (SELDSK) ist für das LOGIN der Laufwerke verantwortlich. Zuerst wird geprüft, ob das angewählte Laufwerk angeschlossen ist. Laufwerksnummern zwischen A und D werden akzeptiert. Ist eine RAM-Floppy implementiert, so kann noch ein fünftes „Laufwerk" E angesprochen werden. Danach wird geprüft, ob für das gewählte Laufwerk bereits ein LOGIN durchgeführt wurde. Ist dies der Fall, so werden die bereits bekannten Werte für die Kopfposition als aktuelle Positionsparameter verwendet. Ist das Laufwerk nicht bekannt, so wird ein LOGIN für diese Laufwerksnummer durchgeführt.

Die LOGIN-Routine

Die LOGIN-Routine verwendet zehn im AUTODPBO-File vordefinierte Disk-Formate, die nach entsprechender Modifikation auf 40 Formate ausgeweitet werden können. Eines dieser Formate muß dem aktuellen Format entsprechen. Die vordefinierten Formate können in einem speziellen BIOS-Modul geändert werden. Entspricht keines dem aktuellen Format, so erzeugt dies folgende Fehlermeldung:

BIOS Error on X, Login
Track-0000, Sector-0000
Retry ‹Y› ?
Die Routine erwartet nun die Eingabe von Y, um einen neuen LOGIN-Versuch zu starten. Alle anderen Eingaben brechen das LOGIN ab und geben den Fehler an das die SELDSK-Funktion aufrufende Programm weiter. Die Angaben für Track und Sector haben hierbei keine Bedeutung und stehen immer auf Null. Findet die LOGIN-Routine das entsprechende Format, so werden die zugehörigen Parameter als aktuell übernommen; entsprechende Werte für die Disk-Parameter-Base des CP/M-Systems werden angeglichen.

Disk-Fehlermeldungen

Beim Lesen oder Schreiben von und zum Laufwerk kann es zu Übertragungsfehlern kommen, die behebbar sind. Daher startet das BIOS automatisch bei Fehlern zehn weitere Schreib-/Lese-Versuche. Tritt der Fehler immer noch auf, so erscheint z. B. folgende Meldung:
BIOS Error on D, Write
Track-0053, Sector-0014
Not Writable,
Retry ‹Y› ?
Wird nach dieser Meldung Y eingegeben, so werden zehn weitere Versuche gestartet. Andere Eingaben führen zum Abbruch des Datentransfers und einer Fehlermittlung an das aufrufende Programm. Im Beispiel könnte der Benutzer den Fehler beheben, indem er den Schreibschutz von der Diskette entfernt. Die Angaben zu Track und Sektor dienen der Lokalisierung des Fehlers. „Track" gibt die logische (!) Position des Schreib-/Lese-Kopfes an. Um die physikalische Position des Kopfes zu ermitteln, ist die Track-Angabe bei zweiseitigen Laufwerken durch zwei zu dividieren. „Sektor" entspricht dem physikalischen Sektor, der gelesen oder beschrieben werden sollte. Beide Angaben erfolgen dezimal.
Folgende Status-Meldungen, die auch gemischt auftreten können geben Hinweise auf die vorliegende Fehlerart:

End of Cylinder
Es wurde eine Sektornummer angewählt, die größer als die Zahl tatsächlich vorhandener Sektoren war.

Data Error
Ein Übertragungsfehler im Daten- oder ID-Feld ist aufgetreten, da die gelesenen CRC-Bytes nicht mit den neu errechneten übereinstimmen. Die CRC-Bytes (Cyclic redundancy check) werden bei einer Schreib-Operation aufgezeichnet. Ursache für diesen Fehler ist meist ein schlechtes Diskettenmaterial, zu lange Übertragungswege, unsauber positionierter Kopf oder eine von fremden Laufwerken beschriebene Diskette.

Over Run
Die zu übertragenden Bytes können nicht rechtzeitig vom CBIOS-3 verarbeitet werden. Ursachen sind zu niedrige Taktfrequenzen des Systems oder gesperrte Interrupts durch Anwenderprogramme.

No Data
Dieser (schwerwiegende!) Fehler tritt auf, falls die gewünschte Sektornummer auf der Diskette nicht gefunden werden kann.

Not Writable
Es wurde versucht, auf eine mit Schreibschutz versehene Diskette zu schreiben.

No Adress Mark
Der Anfang eines Sektors wird nicht gefunden, da eine fremde oder defekte Diskette gelesen oder beschrieben werden soll.

Control Mark
Eine Sektor mit einer sog. „deleted data adress mark" (siehe mc 7/1984, S. 42) sollte gelesen werden. Diese Adress-Marke dient zum Markieren defekter Sektoren.

In Data Field
Der Fehler trat im Daten-Feld des Sektors auf. Fehlt diese Meldung, so trat der Fehler im ID-Feld des Sektors auf.

Wrong Cylinder
Dieser Fehler tritt im Zusammenhang mit „No Data" auf. Die gewünschte Spurnummer stimmt nicht mit der tatsächlichen Kopfposition überein. Dieser sog. Seek-Fehler ist auf eine falsch eingestellte Step-Rate oder dejustierte Laufwerke zurückzuführen.

Bad Cylinder
Es wurde versucht, eine nicht existierende Spur anzuwählen. Diese Meldung tritt zusammen mit dem No-Data-Fehler auf.

Zehn Standard-Formate

In unserem BIOS sind zehn gebräuchliche Standard-Formate definiert, die nach jeweiliger Modifikation im BIOS auf 40 verschiedene Formate ausgeweitet werden. In der *Tabelle* ist eine Übersicht über diese Standard-Formate zusammengestellt. All diese Formate können gleichzeitig auf beliebigen Laufwerken (A–D) benutzt werden. Und hier noch eine kleine Besonderheit unseres BIOS. Erkennt das BIOS keines der zehn Formate, so leitet es automatisch 40 weitere Versuche ein, wobei die Standard-Formate jeweils etwas abgeändert werden: Alle Formate, die als zweiseitig definiert wurden (Double Sided), werden nur einseitig gelesen. Neue Tabellen sonst gleicher Formate für einseitige Laufwerkstypen werden nicht definiert. Die zweite automatische Formatänderung im BIOS ist das sogenannte „double-stepping". Dabei wird einfach die doppelte Spur der gewünschten Spurnummer angewählt. Damit kann man 40-Spur-Disketten (z. B. ECMA-70) auf den modernen 80-Track-Laufwerken bearbeiten.

Mini-Formate 80 Tracks

Format 1 ist das Standard-Format für moderne 5,25- und 3,5-Zoll-Laufwerke mit 80 Spuren. Wie bei allen 80-Spur-Laufwerken ist Diskettenmaterial für doppelte Spurdichte (96 Tpi) zu verwenden. Besonders interessant ist Format 3, das für die genannten Laufwerkstypen zu verwenden ist, jedoch einen schnelleren Zugriff durch große Sektoren (1024 Bytes) erreicht und die Diskette optimal ausnutzt.

8"-kompatible Mini-Formate

Diese Formate sind für den Einsatz auf 8-Zoll-kompatiblen Mini-Laufwerken gedacht. Allerdings sind dabei sehr hochwertige Disketten einzusetzen, da eine erhöhte Spurdichte (96 Tpi) und eine größere Aufzeichnungsdichte (9646 Bpi) vorliegt. Geeignet sind z. B. Maxell-MD2-HD (Double Sided, Double Track, High Density). Format 4 zeigt die höchste Diskettenkapazität mit 1,4 MByte, und das auf einem Mini-Laufwerk!

Maxi-Formate

Dies sind beides Formate für echte 8-Zoller, die sehr häufig anzutreffen sind. Format 8 stellt das bekannte IBM-3740-Format dar, das das einzige wirkliche genormte Diskettenformat ist. Es gewährleistet eine sehr sichere Aufzeichnung, da es große Toleranzen im Kopfpositionier-System zuläßt. Daher wird es besonders gern beim Kauf neuer Software eingesetzt. Format 5 ist auch sehr häufig auf 8-Zoll-Maschinen anzutreffen, zeigt dabei eine vernünftige Kapazität von ca. 1,1 MByte. Beide Formate sind

natürlich auch auf den 8-Zoll-kompatiblen Mini-Laufwerken einzusetzen.

Mini-Formate mit 40 Tracks

Diese Mini-Formate sind für 40-Track-Laufwerke geeignet. Format 6 besitzt eine hohe Kapazität von 400 KByte. Format 7 dürfte wohl jedem aufmerksamen mc-Leser bekannt sein. Es ist das sogenannte ECMA-70-Standard-Format und ist sehr häufig anzutreffen, doch es besitzt nur eine Kapazität von 160 KByte. Format 9 ist z. B. auf den Rechnern von NCR DM-V anzutreffen. All diese Formate können Dank des automatischen ‚double-Stepping' des BIOS ohne jede Änderung auch auf modernen 80-Spur-Laufwerken gelesen und geschrieben (!) werden.

Directory-Einträge und Blockgröße

Gerade bei Disketten hoher Kapazität unter CP/M-Plus ist die Anzahl der Directory-Einträge sehr entscheidend, da die Zeit- und Datumsmarkierungen zusätzliche Directory-Einträge erzeugen. Dabei kann es leicht passieren, daß, obwohl die Diskette noch gar nicht ‚voll' ist, kein Platz mehr für neue Directory-Einträge vorhanden ist. Daher sollte die Anzahl dieser Einträge nicht zu knapp bemessen sein (minimal 128). Doch das BDOS muß oft das Directory nach bestimmten Einträgen durchsuchen, was natürlich mehr Zeit bei vielen Einträgen in Anspruch nimmt. Ein Ausweg aus dieser Zwickmühle stellt die Blockgröße dar. Die Blockgröße gibt an, wie viele Bytes der Diskette pro Eintrag reserviert werden. Erhöht man die Blockgröße z. B. auf 4 KByte, so kann mehr Speicher auf der Diskette adressiert werden und die Anzahl der Directory-Einträge kann wieder gesenkt werden. Doch Vorsicht – bearbeitet man nur sehr kleine Dateien dann wird auf Grund zu großer Blockgrößen sehr viel Platz auf der Diskette vergeudet, da der Eintrag viel mehr Platz reserviert, als für diese kleine Datei überhaupt notwendig wäre. Also: Zum Bearbeiten großer Dateien (z. B. WordStar, DBASE, PASCAL, lange Texte...) ist eine große Blockgröße zu wählen, die Anzahl der Directory-Einträge kann gesenkt werden und ein schneller Zugriff auf diese Dateien ist gewährleistet. Arbeitet man mit sehr vielen kleinen Dateien, so ist die Blockgröße zu senken und dafür die Anzahl der Directory-Einträge zu erhöhen. Man kann beim mc-CP/M-Plus-Computer natürlich gemischt verschiedene Formate einsetzen, die auf die jeweilige Dateigrößen optimal abgestimmt sind.

Die zehn Standard-Formate des mc-CP/M-Plus-BIOS

Format Nummer	1	2	3	4	5	6	7	8	9	10
Mini = 0 Maxi = 1	0	1	0	1	1	0	0	1	0	1
Double/Single Sided	DS	DS	DS	DS	DS	DS	SS	SS	DS	DS
Double/Single Density	DD	DD	DD	DD	DD	DD	DD	SD	DD	DD
Sektorgröße in Bytes	512	512	1024	1024	512	512	256	128	512	512
Sektoren/ Spur	9	16	5	9	15	10	16	26	8	14
Anzahl der Spuren	80	80	80	80	77	40	40	77	40	80
Anzahl der Systemspuren	4	4	4	4	4	4	4	2	3	4
Skew-Faktor	3	3	6	5	5	3	1	6	1	3
Blockgröße in KByte	2	2	4	2	2	2	1	1	2	4
Directory-Einträge	128	256	128	256	128	64	64	64	128	128
Speicherkapazität formatiert KByte	720	1280	800	1440	1155	400	160	250	320	1120

Eigene Formate

Für die meisten Anwender dürften die Standard-Formate wohl mehr als ausreichend sein. Doch eine Anpassung an Formate fremder exotischer Rechner ist sehr einfach möglich. Man kann entweder zusätzlich zu den zehn Formaten praktisch beliebig viele neue hinzufügen oder die bestehenden abändern. Wie das geht zeigen wir im nächsten Heft.

Die RAM-Floppy-Erweiterung

Nach einer RAM-Floppy-Erweiterung steht dem Anwender ein fünftes „Laufwerk" zur Verfügung. Voraussetzung dazu ist eine entsprechende Speichererweiterung des Systems über 128 KByte hinaus. Bis zu 1 MByte RAM wird zur Abwicklung aller herkömmlichen Disk-Operationen unter enormen Geschwindigkeitssteigerungen verwendet. Die Leistungssteigerung kommt durch die hohe Daten-Transferrate zustande, ohne daß zeitraubende Suchoperationen, Kopfladezeiten, Beruhigungszeiten usw. abgewartet werden müssen.
Beim Kaltstart das Systems führt das BIOS einen Speichertest durch, der alle Pages (Speicherseiten) umfaßt. Der vorhandene Speicherplatz wird ermittelt und alle dafür notwendigen Parameter werden automatisch angeglichen. Jetzt werden alle Directory-Einträge im RAM gelöscht. Der RAM-Bereich muß dabei kontinuierlich ab Bank 2 angeordnet sein. (Es ist nicht zulässig, eine Bank auszulassen und eine nachfolgende Bank wieder zu bestücken!)
Alle Disk-Operationen, z. B. das Kopieren von Files zwischen RAM-Floppy und physikalischen Laufwerken, sind möglich. Die RAM-Floppy wird unter der Laufwerksnummer E angesprochen. Übertragen werden jeweils 128 Bytes pro Sektor. Die CP/M-Data-Allocation-Size beträgt 2048 Byte pro Block. Dabei sind bis zu 64 Directory-Einträge möglich.

Ausblick

Im nächsten Heft zeigen wir, wie das BIOS des mc-CP/M-Plus-Computers aufgebaut ist und wie man sich sein ganz „persönliches" BIOS durch Abändern der Standardversion schaffen kann.

Frank Oettle, Thomas Reichler

Der mc-CP/M-Plus-Computer

Teil 4: CBIOS-3

In den vorhergehenden Folgen zum mc-CP/M-Plus-Computer wurde geschildert, welche Anforderungen ein CP/M-Plus-System an die Hardware stellt und wie unser neuer Rechner konstruiert ist. Nun geht es nur noch darum, die Schnittstelle „Software-Hardware" das CBIOS-3, für den mc-CP/M-Plus-Computer zu beschreiben. Die Implementation des BIOS ist von entscheidender Bedeutung für den gesamten Rechner.

Die BIOS-Variante, die in einem Plus-System eingesetzt wird, beeinflußt die Leistung des Systems erheblich. Daher wurde dem mc-CP/M-Plus-Computer ein sehr komfortables BIOS mitgegeben, das die Fähigkeiten des CP/M-Systems auch wirklich zur Geltung bringt. Im folgenden Artikel wird aufgezeigt, wie unser CBIOS-3 aufgebaut ist, welche Besonderheiten es zeigt und wie es sich an die individuellen Bedürfnisse des Benutzers anpassen läßt.

Die CPM3.SYS-Datei

Im ersten Teil dieser Serie wurde die Speicheraufteilung des Betriebssystems CP/M-Plus beschrieben. Der größte Teil des Betriebssystems befindet sich demnach in der Systembank 0 und ist für Anwenderprogramme transparent. Beim mc-CP/M-Plus-Computer steht das gesamte CP/M-System nicht, wie man meinen sollte, auf den Systemspuren einer Diskette, sondern auf den Datenspuren in einer Datei mit dem Namen CPM3.SYS. Es gibt vier Betriebssystem-Teile: gebanktes BDOS und BIOS sowie residentes BDOS und BIOS. Beim Start des Systems werden diese vier Teile aus der CPM3.SYS-Datei an ihre entsprechenden Positionen in den Systemspeicher geladen. Danach wird diese Datei auf der Diskette nicht mehr angesprochen bis der Reset-Knopf einen neuen Boot-Vorgang einleitet.

Die CCP.COM-Datei

Eine Kopie des Consol-Command-Prozessors (CCP), also des CP/M-Programmes, das Befehle entgegennimmt und die entsprechenden Programme aufruft, befindet sich im untersten Teil der Systembank 0. Da der Kommandoteil des neuen Betriebssystems sehr groß ist, ist er – anders als unter CP/M-2.2 – ein ganz normales, lauffähiges Programm (CCP.COM-Datei), das in der TPA ab 100h abläuft. Wird ein Programm aufgerufen, so überschreibt dies CCP. CCP muß also nach Beendigung eines Programms wieder neu in die TPA nachgeladen werden. Diese Aufgabe übernimmt das BIOS (Warmboot), das ohne jeden Diskzugriff die Kopie des CCPs von Bank 0 in die TPA-Bank 1 bringt und anspringt. Ähnlich wie bei CPM3.SYS wird daher CCP.COM nach dem System-Startup nicht mehr auf der Diskette angesprochen.

Der Kaltstart

Das Laden des CP/M-Systems in den Speicher vollzieht sich in folgenden vier Schritten:

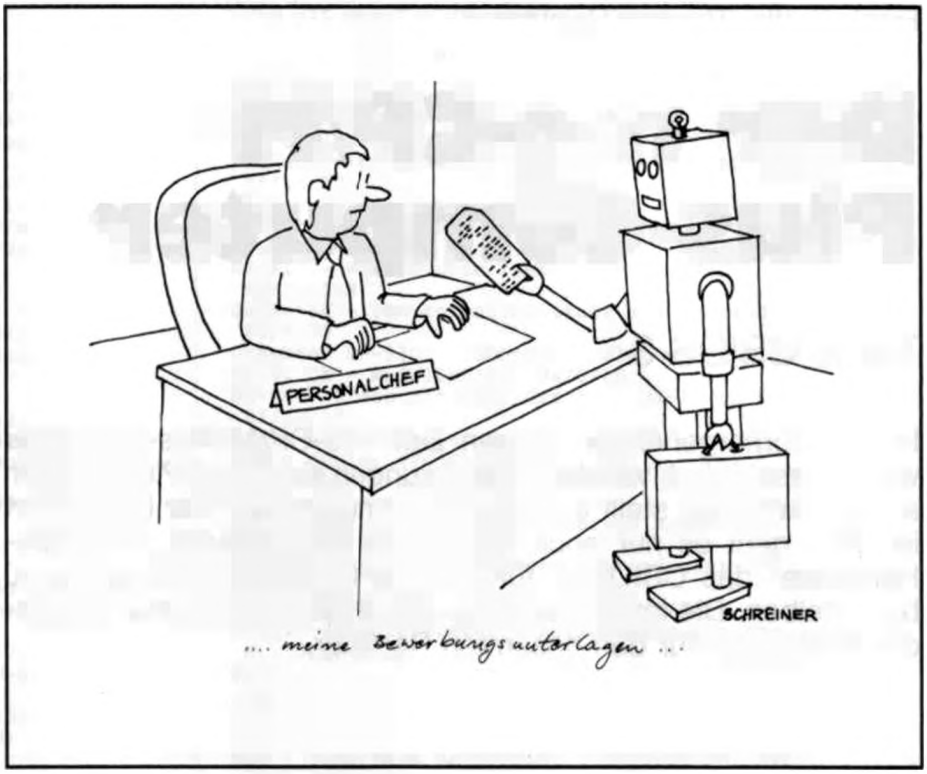
1. Der „Boot-Loader", ein im Boot-EPROM befindliches Programm, sorgt nach dem Einschalten des Systems dafür, daß das CP/M-Lade-Programm auf den Systemspuren der Diskette in den Speicher geladen und ausgeführt wird.
2. Die Aufgabe des CP/M-Lade-Programms besteht darin, das eigentliche CP/M-Plus-System von den Datenspuren der Systemdiskette richtig in die entsprechenden Speicherbänke zu bringen und das BIOS zur Initialisierung zu veranlassen.
3. Die Kaltstart-Routine im BIOS lädt den CCP von den Datenspuren in Bank 0 (0000h–0D00h), nimmt die restlichen Initialisierungen vor und springt zum Schluß die BIOS-Warmstart-Routine an. Die Warmstart-Routine wird immer auch nach Verlassen eines Anwenderprogrammes aufgerufen. Sie dient hauptsächlich dem Laden des CCPs in den TPA-Bereich bei 100h.
4. Bei einem Warmstart wird einfach die Kopie des CCPs aus Bank 0, die ja die Kaltstart-Routine angelegt hat, in den TPA-Bereich geschrieben und der CCP dann angesprungen. Nun endlich erscheint die bekannte Botschaft A⟩ auf dem Bildschirm, der mc-CP/M-Plus-Computer ist bereit.

Der System-Start ist deshalb so kompliziert, weil zum Beispiel auf älteren kleinen Diskettenformaten nicht einmal genug Platz wäre, um das BDOS und BIOS auf den Systemspuren abzuspeichern, vom CCP ganz zu schweigen. Daher mußten diese Teile auf die normalen Datenspuren der Systemdiskette ausgelagert werden. Dies nimmt zwar Speicherplatz in Anspruch, hat aber einen entscheidenden Vorteil: Das gesamte CP/M-System (CPM3.SYS) wird wie jede andere normale Datei behandelt; das System kann also beliebig modifiziert und kopiert werden, ohne daß man auf seine Länge achten muß. Änderungen im BIOS können jederzeit vorgenommen werden ohne, wie noch z. B. beim mc-Computer, ein Boot-EPROM auswechseln zu müssen.

Der Urlader

Nach dem Einschalten der Versorgungsspannung sorgt die Power-On-Reset-Schaltung für ein ordnungsgemäßes Rücksetzen des Systems (Reset). Ein PROM auf der Platine sorgt dafür, daß das Boot-EPROM des Computers im Bereich von 0–3FFFh in den Adreßraum der CPU geschaltet wird und daß die entsprechenden Adressen des dynamischen RAMs ausgeblendet werden. Die CPU startet nach dem Reset bei 0000h. Die Aufgabe des Boot-EPROMs besteht nun darin, den mc-CP/M-Plus-Computer

zu initialisieren (STI, DART, RGB-Farbgrafik...), einen RAM-Test durchzuführen, um gegebenenfalls eine RAM-Floppy einrichten zu können, die RAM-Floppy – falls mehr als 128 KByte Speicher vorhanden sind – zu initialisieren und schließlich eine Mitteilung auszugeben. Danach wartet das Programm auf eine Bestätigung von der Konsole, um den Ladevorgang einleiten zu können. Und schon hier besteht eine kleine Besonderheit des Systems. Das Programm im EPROM untersucht, ob zum Beispiel das RGB-Grafik-Terminal oder ein anderes externes Terminal über die RS-232-Schnittstelle als Konsole angeschlossen ist. Je nachdem lenkt es die entsprechenden Ein- und Ausgaben um. Erfolgt also eine bestätigende Eingabe von der Konsole, so wird das Format der in Laufwerk A befindlichen Systemdiskette automatisch festgestellt. Es werden schon beim Booten die 40 Formate erkannt, die auch das BIOS erkennen kann. Nach der Formaterkennung lädt der Urlader die Systemspuren (Track 0,1, bei Double Sided Laufwerken beide Seiten) in den Speicher ab 4000h aufwärts. Auf den Systemspuren befindet sich normalerweise ein CP/M-Ladeprogramm oder irgendein anderes Programm (z. B. ein Monitor). Nach dem Laden übergibt der Urlader die Kontrolle an das neu eingelesene Programm.



Das CP/M-Ladeprogramm

Das CP/M-Ladeprogramm liest das CP/M-Plus-System aus der CPM3.SYS-Datei und lädt die entsprechenden Teile in den residenten und gebankten Teil des Speichers. Die CPM3.SYS-Datei enthält dazu auch die Informationen, wie groß die vier Teile des Systems sind und wohin sie geladen werden müssen. Da die CPM3.SYS-Datei wie jede andere normale Datei auf den Datenspuren der Diskette steht, muß deren Position auf der Diskette über die Directory-Einträge errechnet werden. Deshalb ist das Ladeprogramm schon ein einfaches nichtgebanktes CP/M-System, das aus zwei Teilen, dem CPMLDR und dem LDRBIOS (vgl. BDOS und BIOS), besteht. Das LDRBIOS enthält alle hardwarespezifischen Funktionen für das Laden von CPM3.SYS und besitzt, wie das normale BIOS, eine Sprungleiste am Anfang, über die die dazugehörigen Routinen erreicht werden können. Im Loader-BIOS sind nur die wichtigsten BIOS-Funktionen zum Laden einer Datei implementiert (BOOT, CONOUT, SETSEC, SETDMA, SETTRK, SECTRAN, READ, MOVE...). Genauso wie beim eigentlichen BIOS müssen hier natürlich wiederum alle der 40 möglichen Diskettenformate erkannt werden, also genaue Kenntnisse über deren Datenstruktur eingearbeitet sein. Nach dem Plazieren von CPM3.SYS im Systemspeicher geschieht der Sprung auf die BIOS-BOOT-Funktion zum eigentlichen Start des Systems. Soll das BIOS des mc-CP/M-Plus-Computers verändert werden oder ist eine andere Speicheraufteilung erwünscht, so muß CPM3.SYS neu zusammengestellt werden. Dazu gibt es das Programm GENCPM, das CPM3.SYS durch Zusammenbinden von BNKBDOS3.SPR (gebankter Teil des BDOS), RESBDOS3.SPR (resistenter Teil des BDOS) und dem BNKBIOS3.SPR (unserem CBIOS-3) erzeugt. Den doch recht mühevollen Weg bis zu einem regelrechten CPM3.SYS-File zeigen wir im folgenden.

BNKBIOS3.SPR

Wie der Name schon andeutet, handelt es sich bei diesem Programm um den BIOS-Teil, der mit dem BDOS zu einem fertigen CPM3.SYS zusammengebunden wird. BNKBIOS3 bedeutet, daß es sich um die gebankte BIOS-Version handelt (es gibt auch eine nichtgebankte Version des CP/M-Plus-Systems). Die Extension SPR besagt, daß dies ein System Page Relocatable File ist. Das Programm ist also noch nicht auf eine bestimmte Speicheradresse fixiert, sondern um das Vielfache von 256 Byte (eine Page) verschiebbar. Zudem besteht das Programm aus zwei Teilen, dem gebankten und dem residenten BIOS-Teil.

Die Teile des BIOS

Das BIOS des mc-CP/M-Plus-Computers besteht aus 12 verschiedenen Modulen, die völlig unabhängig voneinander bearbeitet werden können.

BIOSKRN0.ASM stellt das noch weitgehend hardwareunabhängige Grundgerüst des BIOS dar (BIOS-Kernel). Es enthält die BIOS-Sprung-Vektoren. Von ihm aus werden die anderen BIOS-Module angesteuert. *Bild 1* zeigt den Anfang dieses Teiles. Man sieht dort neben den Tabellen, die dem Informationsaustausch zwischen den verschiedenen Modulen dienen, die bekannte BIOS-Sprung-Tabelle.

XMOVE0.ASM enthält alle Funktionen zur Speicherselektion und zum Speichertransfer.

TIME0.ASM stellt die BIOS-Funktion 26 (Time) zur Verfügung und steuert den Uhrenbaustein des Computers an.

BOOT0.ASM: Erstellen und Nachladen des CCP.

```
        ; Ver 2.0  Rev 15. Apr 84
        ; by Thomas Reichler, 8900 Augsburg

        MACLIB PORTS2     ; port equates for EPC
        MACLIB Z80        ; z80 macros
        MACLIB MODEBAUD   ; definitions for char. I/O

SYSTEMNR        EQU 23H

        cseg              ; GENCPM puts CSEG stuff in common memory

    ; variables in system data page

        extrn $covec,$civec,$aovec,$aivec,$lovec ; I/O redirection vectors
        extrn $mxtpa                             ; addr of system entry point
        extrn $bnkbf                             ; 128 byte scratch buffer

    ; initialization

        extrn ?init                     ; general initialization and signon
        extrn ?ldccp,?rlccp             ; load & reload CCP for BOOT & WBOOT

        public va$tab,vbtab             ; init tables for i/o
        public awr5,bwr5

    ; user defined character I/O routines

        extrn ?ci,?co,?cist,?cost       ; each take device in <B>
        extrn ?cinit                    ; (re)initialize device in <C>
        extrn $ctbl                     ; physical character device table
        extrn co                        ; gdp out
        extrn vainst,vaoust,vain,vaout
        extrn vbinst,vboust,vbin,vbout
        public rtrmflag
        public duplex

    ; disk communication data items

        extrn $dtbl                     ; table of pointers to XDPHs
        public $dbnk                    ;    ''       ''  ''  ''
        public $adrv,$trk,$sect,$dma

        extrn $sekdsk,$seksec,$sektrk,$sekdma,$sekbnk,flush
        extrn fdwrite,fdread,fdlogin,multio
        extrn dpb1,dpb2,dpb3,dpb4
        extrn dpb5,dpb6,dpb7,dpb8

        public biosflag                 ; controls bios functions
        extrn hstac1                    ; track buffer control

    ; memory control

        public $cbnk                    ; current bank
        extrn ?xmove,?move              ; select move bank, and block move
        extrn bnksel                    ; select CPU bank

    ; clock support

        extrn ?time                     ; signal time operation
        public datum
```

Bild 1. BIOSKRN0.ASM fängt so an

```
    ; general utility routines
        public ?pmsg,?pdec      ; print message, print number from 0 to 65535
        public ?pderr           ; print BIOS disk error message header

    ; External names for BIOS entry points

        public ?boot,?wboot,?const,?conin,?cono,?list,?auxo,?auxi
        public ?home,?sldsk,?sttrk,?stsec,?stdma,?read,?write
        public ?lists,?sctrn
        public ?conos,?auxis,?auxos,?dvtbl,?devin,?drtbl
        public ?mltio,?flush,?mov,?tim,?bnksl,?stbnk,?xmov

        ; External names for user definable routines

        extrn zboot,zwbot,zconst,zconin,zconou,zlist,zauxou
        extrn zauxin,zlstst,zconos,zaxist,zaxost,zcinit
        extrn zhome,zselds,zsettr,zsetsc,zsetdm,zread,zwrite
        extrn zsectr,ztime

    ; BIOS Jump vector.

                ; All BIOS routines are invoked by calling these
                ;       entry points.

?boot:  jmp boot        ; initial entry on cold start
?wboot: jmp wboot       ; reentry on program exit, warm start

?const: jmp const       ; return console input status
?conin: jmp conin       ; return console input character
?cono:  jmp conout      ; send console output character
?list:  jmp list        ; send list output character
?auxo:  jmp auxout      ; send auxilliary output character
?auxi:  jmp auxin       ; return auxilliary input character

?home:  jmp home        ; set disks to logical home
?sldsk: jmp seldsk      ; select disk drive, return disk parameter info
?sttrk: jmp settrk      ; set disk track
?stsec: jmp setsec      ; set disk sector
?stdma: jmp setdma      ; set disk I/O memory address
?read:  jmp fdread      ; read physical block(s)
?write: jmp fdwrite     ; write physical block(s)

?lists: jmp listst      ; return list device status
?sctrn: jmp sectrn      ; translate logical to physical sector

?conos: jmp conost      ; return console output status
?auxis: jmp auxist      ; return aux input status
?auxos: jmp auxost      ; return aux output status
?dvtbl: jmp devtbl      ; return address of device def table
?devin: jmp ?cinit      ; change baud rate of device

?drtbl: jmp getdrv      ; return address of disk drive table
?mltio: jmp multio      ; set multiple record count for disk I/O
?flush: jmp flush       ; flush BIOS maintained disk caching

?mov:   jmp ?move       ; block move memory to memory
?tim:   jmp ?time       ; Signal Time and Date operation
?bnksl: jmp bnksel      ; select bank for code execution and default DMA
?stbnk: jmp setbnk      ; select different bank for disk I/O DMA operations.
?xmov:  jmp ?xmove      ; set source and destination banks for one operation

        jmp 0           ; reserved for future expansion
        jmp 0           ; reserved for future expansion
        jmp 0           ; reserved for future expansion
```

```
        ; init table dart:

vatab:  db      p$sioa$ctr,7
        db      18h             ; channel reset
        db      3,0C1h          ; wr3: rxenable, 8 Bits
        db      4,044h          ; wr4: x16 clock mode, 1 Stop, No parity
        db      5
awr5:   db      06AH            ; wr5: no DTR, 8 Bits, ten, RTS

vbtab:  db      p$siob$ctr,7
        db      18h
        db      3,041h          ; wr3: rxenable, 7 Bits
        db      4,04Fh          ; wr4: x16 clock mode, 2 Stop, Parity even
        db      5
bwr5:   DB      0AAh            ; wr5: DTR, RTS, 7 Bits, txenable

        ;*****************************************
        ; BIOS ADRESS TABLES *********************
        ;*****************************************

        ; biosflag meaning
        ; bit 0 set to 1 -> multi read/write operation
        ; bit 1 set to 1 -> login track 0,1 else track 4,5
        ; bit 2 set to 1 -> double sided seek force

biosflag db 0           ; yet normal operation
         dw 0           ; free for future expansions

        ; disk communication data items

$trk:   ds      2               ; selected track
$sect:  ds      2               ; selected sektor
$dma:   ds      2               ; selected dma-adress
$dbnk:  db      0               ; bank for DMA operations
$cbnk:  db      0               ; bank for processor operations
$adrv:  db      0               ; selected drive
curdpb: ds      2               ; adress of current dpb
curdph: ds      2               ; adress of current dph
        ds      3               ; space for extensions
datum:  ds      6               ; date and time
year:   db      84h
        db      systemnr
        dw      0               ; free

        ; Adress of eight disk-parameter-blocks

        dw      dpb1, dpb2, dpb3, dpb4
        dw      dpb5, dpb6, dpb7, dpb8
        ds 3                    ; free
rtrmflag:
        db 0                    ; 0 if rgb-terminal
                                ; -1 if ext. terminal

        ; Adress of direkt rs-232 i/o-routines

        dw vain, vaout, vainst, vaoust
        dw vbin, vbout, vbinst, vboust
```

AUTODPB0.ASM (*Bild 2*) enthält alle Informationen, die zum Verarbeiten der verschiedenen Disk-Formate notwendig sind. Es enthält die Drive-Tabelle, die Disk-Parameter-Header, die Disk-Parameter-Bases (vgl. AUTODPB) und die Umrechnungstabellen der logischen in die physikalischen Sektornummern (Skew-Tabellen).

AUTODSK0.ASM leitet sich vom Auto-Disk-Select ab. Dieser Modul enthält die wichtigen Funktionen Disk-Read/Write und Login. Alle Laufwerke werden von hier aus gesteuert.

TVI950.ASM enthält die Definition der Steuerzeichen und Escape-Sequenzen für das RGB-Terminal.

CHARIO0.ASM (Character I/O) enthält alle Sequenzen für die Zeichen Ein-/Ausgabe. Es wickelt die verschiedenen Protokollarten ab und setzt die Baudraten fest.

SCB.ASM ist der sogenannte System Control Block (*Bild 3*). Er stellt eine Tabelle dar, über die Daten zwischen dem BDOS und dem BIOS ausgetauscht werden. Der SCB dient zum Beispiel zum Übertragen von Datum und Uhrzeit; er enthält die fünf I/O-Redirection-Vektoren, gibt Informationen über die Art der Fehleranzeige des BIOS an und enthält die Eintrittsadresse in das BDOS.

USRDEF0.ASM enthält die Device-Tabelle, die Aufschluß über die möglichen Zeichen-Ein-/Ausgabe-Einheiten gibt (*Bild 4*).

RGBTERM0.ASM ist notwendig, wenn der Computer mit dem RGB-Grafik-Terminal ausgerüstet ist. Die alphanumerische Zeichen-Ausgabe über die Grafik-Karte wird darüber abgewickelt. Es werden 84 × 24 Zeichen in einer 8×12-Punkt-Matrix dargestellt. Zahlreiche Attribute, wie unterstreichen, invers, deutscher/internationaler Zeichensatz, Kursivschrift usw., sind in diesem Teil enthalten. Die Funktionen sind weitgehend mit den Funktionen des TVI950-Terminals kompatibel (die Definition der Control- und Escapesequenzen geschieht im Modul TVI950).

USRROUT.ASM dient zur einfachen Erweiterung des BIOS. Jede BIOS-Funktion ruft vor ihrer normalen Abarbeitung eine zugehörige Routine in diesem Modul auf. Soll das BIOS erweitert werden, so kann man auf einfache Weise durch Hinzufügen eigener Routinen die jeweilige BIOS-Funktion erweitern oder abändern.

Die Software-Werkzeuge für CP/M-3

Zum Betriebssystem CP/M-Plus gehören unter anderem auch der relokative Assembler RMAC und der Linker LINK-80. Sie werden zum Assemblieren und Binden von CBIOS-3 benötigt. Modifiziert man also einen Teil des BIOS (zum Beispiel AUTODPB0.ASM) mit Hilfe eines Editors, so muß durch RMAC nach folgendem Beispiel assembliert werden:

A)RMAC AUTODPB0.ASM

RMAC erzeugt aus dem BIOS-Teil ein beliebig verschiebbares Programm mit der File-Extension REL. Im obigen Beispiel erzeugt der Assembler neben AUTODPB0.SYM (Symbol-Datei) und der Print-Datei AUTODPB0.PRN das relokative Programm AUTODPB0.REL. Liegen

```
        ; Ver. 2.1
        ; Rev. 09.04.84
        ; by Thomas Reichler, 8900 Augsburg


    ; Disk drive dispatching tables for linked BIOS

        public  $dtbl,xdpha,xdphb
        public  dpb1,dpb2,dpb3,dpb4
        public  dpb5,dpb6,dpb7,dpb8
        public  revtrk,dpbnr
        extrn   dphr

    ; CP/M 3 Disk definition macros

        maclib cpm3

;*********************************************************
; Disk timing
; Replace with values for your disk
; Define headload, steprate and headunloadtime
;*********************************************************

; Refer to your disk drive data sheet and uPD 765 data sheet .

;*********************************************************
; Teac FD-55G drives and most other 8 Zoll drives
;*********************************************************

hltg equ 32     ; Head load time in ms for 8 Mhz clock, 2ms increments (2-254)
hutg equ 240    ; Head unload time in ms for 8 Mhz clock, 16ms increments (16-240)
srtg equ 3      ; Step rate in ms for 8 MHz clock, 1 ms increments (1-16)

specg equ ((hltg)/2)*512 + (16-srtg)*16 + (hutg+15)/16


;*********************************************************
; Teac FD-55F drives and most other 96 tpi 5 Zoll drives
;*********************************************************

hltf equ 32/2   ; Head load time in ms for 4 Mhz clock, 4ms increments (4-508)
hutf equ 480/2  ; Head unload time in ms for 4 Mhz clock, 32ms increments (32-480)
srtf equ 4/2    ; Step rate in ms for 4 MHz clock, 2 ms increments (2-32)

specf equ ((hltf)/2)*512 + (16-srtf)*16 + (hutf+15)/16

;*********************************************************
; Teac FD-55B drives and most other 48 tpi 5 Zoll drives
;*********************************************************

hltb equ 32/2   ; Head load time in ms for 4 Mhz clock, 2ms increments (4-508)
hutb equ 480/2  ; Head unload time in ms for 4 Mhz clock, 32ms increments (32-480)
srtb equ 6/2    ; Step rate in ms for 4 MHz clock, 2 ms increments (2-32)

specb equ ((hltb)/2)*512 + (16-srtb)*16 + (hutb+15)/16


; ********************************
; Type Flag Definition
; ********************************
```

Bild 2. AUTODPB0 enthält alle Parameter der Diskettenlaufwerke

```
tyskip          equ     40h
tysingle        equ     20h
tyreverse       equ     04h
ty5zoll         equ     02h
tymfm           equ     00h
tyfm            equ     01h

        dseg    ; disk para headers are switched

;********** Extended Disk Parameter Headers (XPDHs) **************
;*****************************************************************

xdpha:  ds 12                   ; do not change
DPHA:   dph     xlt4,dpb4

xdphb:  ds 12                   ; do not change
DPHB:   dph     xlt4,dpb4

xdphc:  ds 12                   ; do not change
DPHC:   dph     xlt4,dpb4

xdphd:  ds 12                   ; do not change
DPHD:   dph     xlt4,dpb4

        cseg    ; DPB & DTBL must be resident


                ;***** DRIVE TABLE *******
                ;************************

$DTBL:  DTBL    <dphA,dphB,dphC,dphD,dphR>


              ;***** DISK PARAMETER BLOCK CHAIN *****
              ;***************************************

; First some definitions ...

; Number of disk parameter blocks:
; ================================

DPBNR:  db 10   ; The std.-version contains 10 disk-parameter blocks.
                ; Please adapt this number if you use more or fewer
                ; disk formats.

; Number of tracks until the reverse seek-option will be choosen:

REVTRK: db 40   ; NCR DM-V 40 tracks, adapt it if incorrect


;*************************************************
; DRIVE 1 Layout FD-55 F 9x512 2x80
;*************************************************

sect1           equ     512     ; sector Size
mxsc1           equ     9       ; sectors per track
mxtr1           equ     80      ; tracks per surface
side1           equ     2       ; number of surfaces
xtrk1           equ     mxtr1*side1
blk1            equ     1024*2  ; number of 1024K Blocksizes
dir1            equ     128     ; number of directory entries
off1            equ     4       ; offset for system tracks
type1           equ     tymfm + ty5zoll
```

alle BIOS-Module fehlerfrei als REL-Dateien vor, so können diese durch den Linker zu einer einzigen Datei BNKBIOS3.SPR verschmolzen werden. Die Reihenfolge beim Binden (= Linken) ist nach folgendem Beispiel zwingend vorgeschrieben:

A⟩link bnkbios3[b] =
bioskrn0,xmove0,time0,boot0,auto-dpb0,autodsk0,tvi950,chario0,scb, usrdef0,rgbterm0,usrrout,...(yourfile)

Das Zeichen b in den eckigen Klammern veranlaßt den Linker die System-Page-Relocatable-Datei BNKBIOS3.SPR zu erzeugen.

BIOS: in der Bank und resident

Durch Kennzeichnung mit dem Assembler-Pseudo-Befehl CSEG (Code Section) in der Quelldatei veranlaßt man den Assembler, Programmcode mit der Adreßlage für den residenten Systemspeicher zu erzeugen. So müssen die nicht-gebankten Teile des BIOS erzeugt werden. Das Zeichen DSEG (Data Section) schaltet die Lage der Ziel-Datei vom nicht-gebankten zum gebankten Teil um. Die Länge des residenten BIOS-Teils ist begrenzt, da die Gesamtlänge des residenten Teils des Betriebssystems auf 4 KByte begrenzt ist, um einen möglichst großen TPA-Bereich von 60 KByte zu erhalten. Da die Länge des residenten BDOS mit 1,5 KByte vorgegeben ist, bleibt für

```
Xdpb1:  db      2               ; NUM
        db      mxsc1
        db      1Ah             ; Gap 3
        db      -1              ; DTL
        dw      specf           ; Specify Parameter
        db      type1           ; type-byte
        dw      xlt1

DPB1:   dpb     sect1,mxsc1,xtrk1,blk1,dir1,off1
;*****************************************
; DRIVE 2 Layout FD55-G 16x512, 2x80
;*****************************************

sect2   equ     512             ; sector Size
mxsc2   equ     16              ; sectors per track
mxtr2   equ     80              ; tracks per surface
side2   equ     2               ; number of surfaces
xtrk2   equ     mxtr2*side2
blk2    equ     1024*2          ; number of 1024K Blocksizes
dir2    equ     256             ; number of directory entries
skew2   equ     3               ; skew faktor
off2    equ     4               ; offset for system tracks

Xdpb2:  db      02              ; NUM
        db      mxsc2
        db      07h             ; Gap 3
        db      -1              ; DTL
        dw      specg           ; Specify Parameter
        db      tymfm           ; Type
        dw      xlt2
DPB2:   dpb     sect2,mxsc2,xtrk2,blk2,dir2,off2

;*****************************************
; DRIVE 3 Layout FD-55 F 5x1024 2x80
;*****************************************

sect3   equ     1024            ; sector Size
mxsc3   equ     5               ; sectors per track
mxtr3   equ     80              ; tracks per surface
side3   equ     2               ; number of surfaces
xtrk3   equ     mxtr3*side3
blk3    equ     1024*4          ; number of 1024k Blocksizes
dir3    equ     128             ; number of directory entries
off3    equ     4               ; offset for system tracks
skew3   equ     6               ; skew faktor
type3   equ     tymfm + ty5zoll

Xdpb3:  db      3               ; NUM
        db      mxsc3
        db      25h             ; Gap 3
        db      -1              ; DTL
        dw      specf           ; Specify Parameter
        db      type3           ; type-byte
        dw      xlt3

DPB3:   dpb     sect3,mxsc3,xtrk3,blk3,dir3,off3

;*****************************************
; DRIVE 4 Layout FD55-G 9x1024, 2x80
;*****************************************

        sect4           equ     1024    ; sector Size
        mxsc4           equ     9       ; sectors per track
        mxtr4           equ     80      ; tracks per surface
        side4           equ     2       ; number of surfaces
        xtrk4           equ     mxtr4*side4
        blk4            equ     1024*2  ; number of 1024K Blocksizes
        dir4            equ     256     ; number of directory entries
        skew4           equ     5       ; skew faktor
        off4            equ     4       ; offset for system tracks

Xdpb4:          db      03              ; NUM
                db      mxsc4
                db      07h             ; Gap 3
                db      -1              ; DTL
                dw      specg           ; Specify Parameter
                db      tymfm           ; Type
                dw      xlt4
DPB4:   dpb     sect4,mxsc4,xtrk4,blk4,dir4,off4
;*****************************************
; DRIVE 5 Layout FD55-G 15x512, 2x77
;*****************************************

        sect5           equ     512     ; sector Size
        mxsc5           equ     15      ; sectors per track
        mxtr5           equ     77      ; tracks per surface
        side5           equ     2       ; number of surfaces
        xtrk5           equ     mxtr5*side5
        blk5            equ     1024*2  ; number of 1024k Blocksizes
        dir5            equ     128     ; number of directory entries
        skew5           equ     5       ; skew faktor
        off5            equ     4       ; reserved tracks

Xdpb5:          db      2               ; NUM: ld (sect/128)
                db      mxsc5
                db      1Bh             ; GAP 3
                db      -1              ; DTL important if num = 0
                dw      specg           ; Specify parameter
                db      tymfm           ; Type
                dw      xlt5
DPB5:   dpb     sect5,mxsc5,xtrk5,blk5,dir5,off5

;*****************************************
; DRIVE 6 Layout FD55-B 10x512 2x40
;*****************************************

        sect6           equ     512     ; sector Size
        mxsc6           equ     10      ; sectors per track
        mxtr6           equ     40      ; tracks per surface
        side6           equ     2       ; number of surfaces
        xtrk6           equ     mxtr6*side6
        blk6            equ     1024*2  ; number of 1024K Blocksizes
        dir6            equ     64      ; number of directory entries
        off6            equ     4       ; offset for system tracks

Xdpb6:          db      2               ; NUM
                db      mxsc6
                db      10h             ; GAP 3
                db      -1              ; DTL
                dw      specb           ; specify parameter
                db      tymfm+ty5zoll   ; Type
                dw      xlt6
DPB6:   dpb     sect6,mxsc6,xtrk6,blk6,dir6,off6
```

das residente BIOS eine Maximal-Länge von 2,5 KByte (09FFh), die nicht überschritten werden darf. Der Linker gibt nach seinem Lauf unter anderem die Meldung aus, wie groß der gebankte und residente Teil des BIOS sind. Die Linker-Meldung muß auf jeden Fall folgendermaßen erscheinen:

ABSOLUTE	0000	
CODE SIZE	09XX	(0000–09XX)
DATA SIZE	...	(0A00–XXXX)
COMMON		
SIZE	0000	

Der CSEG-Bereich darf 2,5 KByte nicht überschreiten (0000–09XX), die Länge des DSEG-Bereichs ist fast beliebig und nur durch die 48 KByte lange Systembank 0 begrenzt.
Liegt nun endlich eine fehlerfreie BNKBIOS3.SPR-Datei vor, so muß diese wiederum mit dem BDOS zusammengebunden werden. Diese Aufgabe übernimmt das Programm GENCPM.

```
;**********************************************
; DRIVE 7 FD-55 B 16x256x1x40, Ecmac 70
;**********************************************

sect7   equ     256     ; sector Size
mxsc7   equ     16      ; sectors per track
mxtr7   equ     40      ; tracks per surface
side7   equ     1       ; number of surfaces
xtrk7   equ     mxtr7*side7
blk7    equ     1024/1  ; number of 1024k Blocksizes
dir7    equ     64      ; number of directory entries
skew7   equ     1       ; skew faktor
off7    equ     4       ; offset for system tracks

Xdpb7:  db      1       ; NUM
        db      mxsc7
        db      20h     ; Gap 3
        db      -1      ; DTL
        dw      specb   ; Specify Parameter
        db      tymfm+ty5zoll+tysingle
        dw      xlt7
DPB7:   dpb     sect7,mxsc7,xtrk7,blk7,dir7,off7

;**********************************************
; DRIVE 8 Layout IBM 3740 Standard
;**********************************************

sect8   equ     128     ; sector Size
mxsc8   equ     26      ; sectors per track
mxtr8   equ     77      ; tracks per surface
side8   equ     1       ; number of surfaces
xtrk8   equ     mxtr8*side8
blk8    equ     1024/1  ; number of 1024k Blocksizes
dir8    equ     64      ; number of directory entries
skew8   equ     6       ; skew faktor
off8    equ     2       ; offset for system tracks

Xdpb8:  db      0       ; NUM
        db      mxsc8
        db      07h     ; Gap 3
        db      128     ; 128
        dw      specg   ; Specify Parameter
        db      tysingle+tyfm ; Type
        dw      xlt8
DPB8:   dpb     sect8,mxsc8,xtrk8,blk8,dir8,off8

;**********************************************
; DRIVE 9 Layout NCR F3 format 2x40 Tracks reverse
;**********************************************

sect9   equ     512     ; sector Size
mxsc9   equ     8       ; sectors per track
mxtr9   equ     40      ; tracks per surface
side9   equ     2       ; number of surfaces
xtrk9   equ     mxtr9*side9
blk9    equ     1024*2  ; number of 1024k Blocksizes
dir9    equ     128     ; number of directory entries
skew9   equ     1       ; skew faktor
off9    equ     3       ; offset for system tracks

Xdpb9:  db      2               ; NUM
        db      mxsc9
        db      2Ah             ; Gap 3
        db      -1              ; DTL
        dw      specb
        db      tymfm+ty5zoll+tyreverse; Type
        dw      xlt9
DPB9:   dpb     sect9,mxsc9,xtrk9,blk9,dir9,off9

;**********************************************
; DRIVE 10 Layout 8" MFM Format 14x512x2x80
;**********************************************

sect10  equ     512     ; sector Size
mxsc10  equ     14      ; sectors per track
mxtr10  equ     80      ; tracks per surface
side10  equ     2       ; number of surfaces
xtrk10  equ     mxtr10*side10
blk10   equ     1024*4  ; number of 1024k Blocksizes
dir10   equ     128     ; number of directory entries
skew10  equ     3       ; skew faktor
off10   equ     4       ; offset for system tracks

Xdpb10: db      2               ; NUM
        db      mxsc10
        db      25h             ; Gap 3
        db      -1              ; DTL
        dw      specg           ; Specify Parameter
        db      tymfm           ; type
        dw      xlt10
DPB10:  dpb     sect10,mxsc10,xtrk10,blk10,dir10,off10

        dseg

        ;*******  SECTOR TRANSLATE TABLES  **********
        ;**********************************************

xlt1:   db      1,4,7,2,5,8,3,6,9       ; Standard FD-55 F
xlt2:   skew    mxsc2,skew2,1           ; 16x512x2x80
xlt3:   skew    mxsc3,skew3,1           ; 5x1024x2x80
xlt4:   skew    mxsc4,skew4,1           ; 9x1024x2x80
xlt5:   skew    mxsc5,skew5,1           ; Standard FD-55G/8" 512x15x2x77
xlt6:   db      1,4,7,10,3,6,9,2,5,8    ; Standard FD-55 B 10x512x2x40
xlt7:   skew    mxsc7,skew7,1           ; Ecmac 70
xlt8:   skew    mxsc8,skew8,1           ; FD-55G/8" std ibm 3740
xlt9:   skew    mxsc9,skew9,1           ; NCR F3
xlt10:  skew    mxsc10,skew10,1         ; 14x512x2x80

        end
```

Der Aufbau der BIOS-Module

Bild 4 zeigt Ausschnitte der USRDEF0-Datei. In ihr wird die Device-Tabelle aufgestellt. Man sieht dort sehr gut, wie ein BIOS-Modul aufgebaut ist. Die erste Assembleranweisung cseg bedeutet wie gesagt, daß nachfolgender Code im Common-Memory-Bereich stehen soll. Die folgende Anweisung maclib modebaud weist den Assembler an, die Macro-Library Modebaud (*Bild 5*) einzulesen. Diese Library enthält die Definitionen aller gültigen Werte, die in die Device-Tabelle eingesetzt werden dürfen. Die nächste Anweisung dient dem Binden der BIOS-Module. Die beiden Marken §intab und §ctbl sind als öffentliche (public) Marken definiert, auf die sich andere BIOS-Module beziehen können. Soll in einem anderen BIOS-Modul auf die Marke §ctbl zugegriffen werden, so muß in diesem Modul die Marke über extrn §ctbl als extern definiert werden. Eine der Aufgaben des Linkers ist zum Beispiel nun, all diese Marken, die in den verschiedenen Modulen als extern oder public definiert werden, durch absolute Adressen zu ersetzen. Der Assembler ist dazu noch nicht in der Lage, da er ja von den anderen Modulen noch nichts weiß.
Der Inhalt der AUTODPB0-Datei (Bild 2) bestimmt die Disk-Formate, die bearbeitet werden können. Hier erfolgt eine Anpassung an das Laufwerks-Timing (Steprate, Headload-Time...) und die exakte logische und physikalische Definition der Formate.

AUTODPB0: zum Disktiming

Laufwerke verschiedener Hersteller weisen oft unterschiedliche Stepraten auf, also Zeiten in denen sie den Schreib-/Lesekopf bewegen können. In AUTODPB0 können diese Zeiten exakt, getrennt für jedes Format, bestimmt werden. Die Angabe hltg equ 32 in AUTODPB0 besagt zum Beispiel, daß die Kopflade-Zeit in diesem Fall 32 ms beträgt. Die drei Angaben hltx (Head Load Time), hutx (Head Unload Time) und srtx (Steprate-Time) werden nach entsprechender Umrechnung zu einem Specify-Parameter specx umgerechnet, der zur Programmierung des FDC-Controllers µPD 765 dient.

AUTODPB0: Type-Flag

Das Type-Flag gibt dem BIOS einige physikalische Angaben über das betreffende Disk-Format:
Bit 0 = 1:
Es findet die sog. FM-Aufzeichnung Verwendung. Ist dieses Bit auf 0 gesetzt, so werden die Daten mit doppelter Aufzeichnungsdichte übertragen.
Bit 1 = 1:
Ist dieses Bit gesetzt, so liegt ein 5¼-Zoll-kompatibles Format vor, andernfalls ein 8-Zoll-Format mit doppelter Schreibtaktfrequenz (500/250 kHz).
Bit 2 = 1:
Bit 2 dient zur Kennzeichnung älterer zweiseitiger Diskettenformate. Bei solchen Formaten wird von Spur 0 bis 39 der Kopf 0 des Laufwerks adressiert, bei Spur 40 bis 79 wird Kopf 1 Track 0 bis 39 adressiert. Ist dieses Bit auf Null gesetzt, so wird bei doppelseitigen Formaten ein weitaus gebräuchlicheres Verfahren zur Spur-Anwahl eingesetzt: Bei geraden Spurnummern wird Kopf 0, bei ungeraden Spurnummern Kopf 1 adressiert, d. h. bei steigenden Spurnummern wird immer zwischen Kopf 0 und Kopf 1 abgewechselt. Vorteil dieser Methode: man benötigt um die Hälfte weniger Step-Befehle.
Bild 5 = 1:
Ist dieses Bit gesetzt, so liegt nur eine einseitig formatierte Diskette vor. Es wird nur Kopf 0 des Laufwerks angesprochen. Ist dieses Bit rückgesetzt, so werden beide Köpfe des Laufwerks angesprochen.
Bit 6 = 1:
Dieses Skip-Bit veranlaßt das BIOS beim Anwählen einer Spur die Spurnummer zu verdoppeln. Jede zweite Spur wird übersprungen. Dies ist bei Einsatz von 80-Track-Laufwerken interessant. Mit Hilfe der Skip-Funktion können auf 80-Track-Laufwerken auch ältere 40-Track-Disketten verarbeitet werden.

```
        title 'System Control Block Definition for CP/M3 BIOS'
        public $civec, $covec, $aivec, $aovec, $lovec, $bnkbf
        public $crdma, $crdsk, $vinfo, $resel, $fx, $usrcd
        public $mltio, $ermde, $erdsk, $media, $bflgs
        public $date, $hour, $min, $sec, ?erjmp, $mxtpa

scb$base equ    0FE00H          ; Base of the SCB

$CIVEC  equ     scb$base+22h    ; Console Input Redirection
                                ; Vector (word, r/w)
$COVEC  equ     scb$base+24h    ; Console Output Redirection
                                ; Vector (word, r/w)
$AIVEC  equ     scb$base+26h    ; Auxiliary Input Redirection
                                ; Vector (word, r/w)
$AOVEC  equ     scb$base+28h    ; Auxiliary Output Redirection
                                ; Vector (word, r/w)
$LOVEC  equ     scb$base+2Ah    ; List Output Redirection
                                ; Vector (word, r/w)
$BNKBF  equ     scb$base+35h    ; Address of 128 Byte Buffer
                                ; for Banked BIOS (word, r/o)
$CRDMA  equ     scb$base+3Ch    ; Current DMA Address
                                ; (word, r/o)
$CRDSK  equ     scb$base+3Eh    ; Current Disk (byte, r/o)
$VINFO  equ     scb$base+3Fh    ; BDOS Variable "INFO"
                                ; (word, r/o)
$RESEL  equ     scb$base+41h    ; FCB Flag (byte, r/o)
$FX     equ     scb$base+43h    ; BDOS Function for Error
                                ; Messages (byte, r/o)
$USRCD  equ     scb$base+44h    ; Current User Code (byte, r/o)
$MLTIO  equ     scb$base+4Ah    ; Current Multi-Sector Count
                                ; (byte,r/w)
$ERMDE  equ     scb$base+4Bh    ; BDOS Error Mode (byte, r/o)
$ERDSK  equ     scb$base+51h    ; BDOS Error Disk (byte,r/o)
$MEDIA  equ     scb$base+54h    ; Set by BIOS to indicate
                                ; open door (byte,r/w)
$BFLGS  equ     scb$base+57h    ; BDOS Message Size Flag (byte,r/o)
$DATE   equ     scb$base+58h    ; Date in Days Since 1 Jan 78
                                ; (word, r/w)
$HOUR   equ     scb$base+5Ah    ; Hour in BCD (byte, r/w)
$MIN    equ     scb$base+5Bh    ; Minute in BCD (byte, r/w)
$SEC    equ     scb$base+5Ch    ; Second in BCD (byte, r/w)
?ERJMP  equ     scb$base+5Fh    ; BDOS Error Message Jump
                                ; (word, r/w)
$MXTPA  equ     scb$base+62h    ; Top of User TPA
                                ; (address at 6,7)(word, r/o)
        end
```

Bild 3. Der System Control Block der BIOS ist eine Tabelle für den Datenaustausch zwischen BIOS und BDOS

AUTODPB0: Der Disk Parameter Header

Für jedes Laufwerk existiert unter CP/M-Plus ein Disk Parameter Header (DPH), der die aktuellen logischen Informationen über das Laufwerk enthält. Diese Tabellen wurden für unseren Computer etwas erweitert und enthalten zusätzliche physikalische Informationen für das BIOS. Das Anlegen der Tabelle übernimmt die Macro-Library dph, die unter anderem über die Assembleranweisung maclib cpm3 eingelesen wird. Als Eingangsparameter benötigt dieser Macro zwei Werte: Die Adresse eines Disk-Parameter-Blocks und die zugehörige Sektor-Übersetzungstabelle. Die Tabelle enthält unter anderem auch Informationen darüber, wie groß der Allocation und der Checksum Vektor für dieses Laufwerk sind, die an Hand des Disk-Parameter-Blocks ermittelt werden. Da unser BIOS über eine Auto-Disk-Selekt-Funktion verfügt, kann jedem Laufwerk ein beliebiges Format zugeordnet werden. Daher kann die Größe dieser Vektoren unterschiedlich sein. Um auf jeden Fall genügend Platz für diese Vektoren zu reservieren, sollte nach dem Macro-Aufruf dph derjenige Disk-Parameter-Block eingesetzt werden, der den meisten Platz für diese Vektoren in Anspruch nimmt. Im Standard-BIOS ist dies das Format 4 mit 256 möglichen Directory-Einträgen und einer Kapazität von ca. 1,4 MByte.

AUTODPB0: Die Drive-Tabelle

Nach dem Disk-Parameter-Header in AUTODPB0 steht die Laufwerks-Tabelle §DTBL, die wiederum von einem Macro, dtbl, angelegt wird. Diese Tabelle enthält die Adressen der Disk-Parameter-Header und bestimmt, wieviele Laufwerke vom BIOS verwaltet werden. Im Standard-BIOS werden vier Laufwerke (dphA–dphD) und eine RAM-Floppy (dphR) angesprochen. Sind weniger Laufwerke angeschlossen, so sind diese entsprechend aus der Drive-Table zu streichen.

AUTODPB0: Die Disk-Parameter-Blöcke

Ein Disk-Parameter-Block (DPB) gibt genaue Auskunft über die Art des verwendeten Disketten-Formats. Er enthält einerseits logische Daten, wie z. B. Anzahl der Directory-Einträge, Blockgröße usw., andererseits alle notwendigen Informationen über den physikalischen Aufbau des Formats, z. B. Anzahl der Spuren, Größe der Sektoren, Aufzeichnungsdichte usw. Jedes Format, das erkannt werden soll, benötigt seinen eigenen Disk-Parameter-Block. Standardmäßig sind im CBIOS-3 zehn Blöcke vorhanden. Diese können natürlich modifiziert werden. Man kann eigene neue Blöcke hinzufügen, die bestehenden modifizieren oder nicht benötigte Formate streichen. Unter dem mc-CP/M-Plus-BIOS wurden die normalen DPBs etwas erweitert und werden nun mit XDPB (extended DPB) gekennzeichnet. Daher bestehen die Disk-Parameter-Blöcke aus zwei Teilen, dem BIOS- und dem BDOS-spezifischen Teil. Der erste, BIOS-spezifische Teil setzt sich aus folgenden Werten zusammen, die teilweise direkt mit der Programmierung des FDC-Controllers µPD 765 (s. Datenblatt) in Verbindung stehen:

1. NUM — 0 falls 128 Bytes Sektorgröße, 1 bei 256 Byte, 2 bei 512 Bytes pro Sektor...
2. MXSC — Sektoren pro Spur
3. GAP3 — Länge von GAP3
4. DTL — −1 falls NUM > 0, sonst Bytes pro Sektor
5. SPEC — Specify Parameter für das Disk-Timing
6. TYPE — Type Flag
7. XLT — Adresse der Sektor-Übersetzungstabelle

Der zweite Teil ist der eigentliche Disk-Parameter-Block. Der Macro dpb generiert diesen automatisch aus den angegebenen Parametern. Verändert man einen Disk-Parameter-Block, so muß man darauf achten, daß die Reihenfolge und Länge der zehn Blöcke nicht durcheinander kommt. Am einfachsten ist, man verändert nur die Equate-Anweisungen am Kopf eines jeden Blockes.

So erkennt das BIOS ein Format

Beim Login, d. h. beim Identifizieren des Formates einer neuen Diskette, stehen dem BIOS die zehn Disk-Parameter-Blöcke zur Verfügung. Das BIOS startet seine Login-Prozedur mit der Marke DPB1. Daher muß diese Marke auch die erste der zehn DPB-Marken sein. Nun

```
; I/O Characteristics for mc-CP/M-Plus Computer
        cseg                    ; This part resides in common memory

        maclib modebaud         ; macro library definition
        public §intab,§ctbl     ; public data items for the linker

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;               Baudrates and Protocolls
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

; Limitations:
; Baud rates 1800, 3600 and 7200 are approximations
; Baud rate 50, 75 is not supported
; Equal baudrates are selected for device 0,5 and device 2,6

§ctbl:
        ; Device 0 CRT (RGB-Terminal)

        db 'CRT
        db mb$in$out+mb$serial+mb$softbaud
        db baud$9600

        ; Device 1 V24A (DART channel A)

        db 'V24-A '
        db mbinout+mb$serial+mb$softbaud+mb$xon$xoff
        db baud$9600

        ; Device 2 V24B (DART channel B)

        db 'V24-B '
        db mb$in$out+mb$serial+mb$softbaud
        db baud$1200

        ; Device 3 CENTR (Centronics Interface)

        db 'CENTR '
        db mb$output
        db baud$none

        ; Device 4 PARA (reserved parallel Port)

        db 'PARA  '
        db mb$input
        db baud$none
```

Bild 4. Die Datei USERDEF

übernimmt das BIOS die Daten aus dem jeweiligen Block, programmiert den FD-Controller entsprechend und startet auf bestimmten Positionen der Diskette Lese-Versuche. Verlaufen diese fehlerfrei, so untersucht BIOS noch, ob die Sektorenzahl pro Track, die Anzahl der Seiten usw. mit den Angaben des jeweiligen Disk-Parameter-Blocks übereinstimmen. Verlaufen alle Tests positiv, so übernimmt das BIOS die Daten aus dem Disk-Parameter-Block und setzt sie als aktuelle Laufwerksparameter ein und gleicht den Disk-Parameter-Header für dieses Laufwerk entsprechend an. Diese Werte haben nun Gültigkeit, bis der Anwender durch Betätigen von ↑C einen Disk-Reset durchführt oder ein Schreib-/Lesefehler auf der Diskette erkannt wird, der auf das Einlegen einer anders formatierten Diskette zurückzuführen ist. Verläuft einer der oben genannten Tests negativ, so überspringt das BIOS den derzeitigen Disk-Parameter-Block und startet einen neuen Login-Versuch beim nächsten Block. Verlaufen alle zehn möglichen Tests negativ, so leitet das BIOS automatisch eine Modifizierung der Formate ein.

Die Type-Byte-Modifikation

Das BIOS verändert dazu in vier Schritten die ursprünglichen Werte des Type-Bytes im Disk-Parameter-Block, um dennoch das eingelegte Format zu erkennen. Es geht um zwei Bits, Bit 5 (single/double-sided) und Bit 6 (skip/no-skip).

Schritt 1:
Rücksetzen von Bit 5 und 6. Ausschalten der Skip-Option und zweiseitiger Diskettenzugriff.
TYPE I: X00X.XXXX
Nun startet das BIOS wieder zehn Login-Versuche, bis es das entsprechende Format findet. Falls nicht, wird das Byte ein zweitesmal geändert.

Schritt 2:
Einschalten der Skip-Option und zweiseitige Aufzeichnung.
TYPE II: X10X.XXXX
Nun werden z. B. doppelseitige Formate von 40 Track-Laufwerken erkannt, die in 80 Track-Laufwerken eingelegt wurden.

Schritt 3:
Ausschaltung der Skip-Option und einseitige Aufzeichnung.
TYPE III: X01X.XXXX
Hier können die ursprünglich für zweiseitige Aufzeichnung vorgesehenen Formate auch auf einseitig formatierte Disketten angewendet werden.

Schritt 4:
Einschalten der Skip-Option und einseitige Aufzeichnung.
TYPE IV: X11X.XXXX
Schritt 1 bis 4 werden natürlich für alle zehn Disk-Parameter-Blöcke durchgeführt. Dies ergibt maximal 40 verschiedene Formate, die das Standard-BIOS nach diesem Verfahren erkennen kann. Verlaufen alle Schritte erfolglos, so setzt das BIOS eine Fehlermeldung ab und startet, falls der Bediener es wünscht, eine neue Login-Prozedur oder gibt den Fehler an das aufrufende Programm weiter.

```
        ; Device 5 TTLSER (STI serial channel)
        ; Select same baudrate as in device 0 CRT

        db 'TTLSER'
        db mbin$out+mb$serial+mb$softbaud
        db baud$9600

        ; Device 6 DIABLO (DART channel B with etx/ack Protocol)
        ; Select same baudrate as in device 2 V24B
        ; Do not implement a xon/xoff Protocol

        db 'DIABLO'
        db mb$in$out+mb$serial+mb$softbaud
        db baud$1200


;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
;               I/O - Select at Coldboot
;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

; Definitions ( dont alter these values)

crt     equ     8000h
v24a    equ     4000h
v24b    equ     2000h
centr   equ     1000h
ttlser  equ     0800h

; Here you can define the initial I/O-Redirection-Vector
; in the SCB at coldboot. Multiple I/O-Devices for the same
; logical device are allowed.
; For example: crt + v24a + ...

$intab:
        db      0               ; Flag to say autodevice select if zero.
                                ; If non-zero following values will be choosen
                                ; at system start-up.

        dw      crt             ; console input
        dw      crt             ; console output
        dw      v24a            ; auxiliary input
        dw      v24a            ; auxiliary output
        dw      centr           ; list output

        end
```

AUTODPB0: Die Sektor-Übersetzungs-Tabellen

Jedem Format kann eine eigene Übersetzungstabelle zugeordnet werden, die sich mit den anderen am Schluß der AUTODPB0-Datei befindet. Das Aufstellen der Tabelle übernimmt der Macro skew nach Angabe der Sektorzahl und dem Skew-Faktor (interleaving factor). Wenn das CBIOS richtig konfiguriert ist, dann muß aus diesem BIOS und den anderen Komponenten von CP/M-3 ein lauffähiges System zusammengebaut werden. Das geschieht mit GENCPM.

So arbeitet GENCPM

GENCPM benötigt die vorher genannten vier Teile des CP/M-Plus-Betriebssystems, um eine CPM3.SYS-Datei zu erzeugen. Die beiden BIOS-Teile liegen in der BNKBIOS3.SPR-Datei vor, die beiden BDOS-Teile werden mit dem Betriebssystem von Digital Research geliefert und tragen die Namen RESDBOS3.SPR und BNKBDOS3.SPR. Bevor GENCPM die CPM3.SYS-Datei aus diesen drei Komponenten zusammenbaut, hat der Anwender noch eine Reihe von Möglichkeiten, das Betriebssystem nach seinen Wünschen anzupassen. Die Anzahl der verfügbaren Speicherbänke, deren Nutzung, die Art der Fehlermeldun-

gen usw. können hier noch festgelegt werden. Im folgenden Beispiel wird gezeigt, wie die Fragen von GENCPM beantwortet werden.

```
A>GENCPM
CP/M 3.0 System Generation
Copyright (C) 1982, Digital Research
Default entries are shown in (parents)
Default base is Hex, precede entry with
# for decimal

Use GENCPM.DAT for defaults (Y) ? Y
Create a new GENCPM.DAT file (N) ? N
```

Nach dem ersten Lauf von GENCPM legt dieses die Datei GENCPM.DAT auf der Diskette ab. Darin sind alle Antworten enthalten, die auf die Fragen von GENCPM gegeben wurden. Wird GENCPM danach noch einmal gerufen, so werden diese sogenannten Default-Werte zur Beantwortung der Fragen herangezogen. Das angepaßte CP/M-Plus für den mc-CP/M-Plus-Computer enthält natürlich schon diese GENCPM.DAT-Datei mit den richtigen Antworten.

```
Display Load Map at Cold Boot (Y) ? Y
```

Beim Kaltstart des Systems bringt der CPM Loader eine sog. Load Map auf den Bildschirm, die Auskunft über die Speicherbelegung des Systems gibt, wenn hier mit ⟨Y⟩ geantwortet wird.

```
Number of console columns (#85) ? #85
Number of lines in console page
(#24) ? #24
Backspace echoes erased character (N) ? N
Rubout echoes erased character (N) ? N
```

Das Betriebssystem benötigt Daten über die angeschlossene Consol-Einheit. Wird die RGB-Grafik-Karte als Terminal verwendet, so sind die Werte oben richtig.

```
Initial default drive (A:) ? A
```

Diese Eingabe legt die Laufwerksnummer nach dem Coldboot des Systems fest.

```
Top page of memory (FF) ? FF
Bank switched memory (Y) ? Y
Common memory base page (C0) ? C0
```

Die oberste 256-Byte-Seite (Top Page) des mc-CP/M-Computers ist 0FFh, da jede Speicherseite 64 KByte lang ist. Die zweite Frage muß natürlich mit ⟨Y⟩ beantwortet werden, da das BIOS für die bank-switched-CP/M-Version geschrieben wurde. Die Common-Memory-Base, also die Adresse für den Beginn des nichtgebankten Speicherbereichs, liegt beim mc-Computer bei 48 KByte, also in der Page 0C0h.

```
Long error messages (Y) ? Y
```

Wird diese Frage mit ⟨Y⟩ beantwortet, so erzeugt das System sog. lange, d. h. sehr detaillierte und aufschlußreiche Meldungen, falls ein Fehler auftritt. Allerdings benötigen diese Meldungen etwas mehr Speicherplatz.

```
Accept new system definition (Y) ? Y
Setting up Allocation vector for drive
A:,B:,C:,D:,E:

Setting up Checksum vector for drive
A:,B:,C:,D:,E:
```

GENCPM deutet mit dieser Meldung (verkürzt!) an, daß es für die fünf Laufwerke entsprechende Tabellenplätze reserviert.

```
*** Bank 1 and Common are not
    included                    ***
*** in the memory segment table. ***
Number of memory segments (#1) ? 1
```

Hier fragt GENCPM nach der Anzahl der Speichersegmente. Der normale mc-CP/M-PLUS-Computer ohne Speichererweiterung hat 128 KByte Speicher (2 Speicherbänke). GENCPM rechnet die TPA-Bank 1 nicht zu den freien Systemspeicherbänken, daher muß die Frage normalerweise mit ⟨1⟩ beantwortet werden. Steht nach einem Ausbau mehr Speicher zur Verfügung, so können diese weiteren Bänke entweder der RAM-Floppy oder dem Betriebssystem zugeteilt werden. Wird eine RAM-Floppy eingesetzt, so muß die obige Frage wieder mit ⟨1⟩ beantwortet werden, da die RAM-Floppy den Speicherplatz ab Bank 2 aufwärts benutzt.

```
CP/M 3 Base,size,bank (7B,45,00)
Enter memory segment table:
Base,size,bank (31,4A,00) ? 31,4A,00
```

GENCPM zeigt in der ersten Meldung die Belegung der Systembank 0 mit dem Betriebssystem an. Das gebankte Betriebssystem hat eine Länge von knapp 32 KByte und erstreckt sich daher von 7B00h bis 0BFFFh. Der Bereich unterhalb von 7B00h ist frei zum Anlegen der verschiedenen Puffer-Bereiche für das Betriebssystem. Nun muß eingegeben werden, welcher Bereich des noch freien Teils der Bank 0 dem CP/M-Plus zugeordnet werden soll. Wie schon erwähnt, befindet sich in Bank 0 eine Kopie des CCPs und der BIOS-Track-Puffer, die standardmäßig den Bereich von 0–30FFh belegen. Die Länge des CCPs liegt fest. Er erstreckt sich von 0–0CFFh. Bei 0D00h startet der BIOS-Track-Puffer, dessen Länge sich nach der größten Tracklänge der verwendeten Formate richtet. Unsere Standard-BIOS-Version

```
        ; ************************************
        ; equates for mode byte bit fields
        ; ************************************

mb$input                equ 0000$0001b  ; device may do!input
mb$output               equ 0000$0010b  ; device may do output
mb$in$out               equ mb$input+mb$output

mb$soft$baud            equ 0000$0100b  ; software selectable
                                        ; baud rates

mb$serial               equ 0000$1000b  ; device may use protocol
mb$xon$xoff             equ 0001$0000b  ; XON/XOFF protocol
                                        ; enabled

baud$none               equ 0           ; no baud rate associated
                                        ; with this device
baud$50                 equ 1           ; 50 baud
baud$75                 equ 2           ; 75 baud
baud$110                equ 3           ; 110 baud
baud$134                equ 4           ; 134.5 baud
baud$150                equ 5           ; 150 baud
baud$300                equ 6           ; 300 baud
baud$600                equ 7           ; 600 baud
baud$1200               equ 8           ; 1200 baud
baud$1800               equ 9           ; 1800 baud
baud$2400               equ 10          ; 2400 baud
baud$3600               equ 11          ; 3600 baud
baud$4800               equ 12          ; 4800 baud
baud$7200               equ 13          ; 7200 baud
baud$9600               equ 14          ; 9600 baud
baud$19200              equ 15          ; 19.2k baud
```

Bild 5. MODEBAUD enthält die gültigen Werte für die Baudraten und die I/O-Charakteristiken

hat als größte Tracklänge das 8-Zoll-Format mit 9 × 1024 Byte pro Track, also reserviert der Puffer den 9 KByte langen Bereich von 0D00h–30FFh. Für die CP/M-Puffer verbleibt also der Bereich von 3100h bis 7AFFh mit einer Länge von 4Ah Seiten.

```
CP/M 3 Sys      7B00H 4500H Bank 00
Memseg No. 00   3100H 4A00 Bank 00
```

Accept new memory segment table entries (Y) ? Y

Da nur eine Systembank zur Verfügung steht, bricht GENCPM die Fragen nach der Speicheraufteilung ab, zeigt diese noch einmal an und vergewissert sich, ob die Fragen auch richtig beantwortet wurden.

```
Setting up directory hash tables:
Enable hashing for drive A: (Y) ? Y
Enable hashing for drive B: (Y) ? Y
Enable hashing for drive C: (N) ? N
Enable hashing for drive D: (N) ? N
Enable hashing for drive E: (N) ? N
```

GENCPM fragt, für welches der fünf Laufwerke Hash-Tabellen angelegt werden sollen. Das Directory-Hashing verkürzt den Disk-Zugriff, benötigt jedoch Speicherplatz. Daher sollte nur für diejenigen Laufwerke mit ⟨Y⟩ geantwortet werden, mit welchen man hauptsächlich arbeitet.

```
Setting up Blocking/Deblocking buffers:
The physical record size is 0400H:
```

Nun legt GENCPM Daten- und Directory-Puffer für die fünf Laufwerke an. Es erkennt aus den Disk-Parameter-Header-Einträgen, daß die maximale Sektorgröße 1024 Bytes (0400h) beträgt. Nun läßt sich für jedes Laufwerk getrennt angeben, wieviel Daten- und Directory-Puffer ihm zugeordnet werden sollen. Da jeder Puffer Speicherplatz in Anspruch nimmt, steht natürlich nur eine begrenzte Anzahl an Puffern zur Verfügung, die nun optimal aufgeteilt werden sollten. Als Regel gilt: Die Laufwerke mit denen man am meisten arbeitet, sollten auch die meisten Puffer bekommen. Für Laufwerk A sieht das folgendermaßen aus:

```
Available space in 256 byte pages:
TPA = 00F0H, Bank 0 = 0042H,
Other banks = 0000H

Number of directory buffers for drive A:
(#4) ? 4

Available space in 256 byte pages:
TPA = 00F0H, Bank 0 = 0031H,
Other Banks = 000H

Number of data buffers for drive A:
(#4) ? #4
Allocate buffers outside of
Common (Y) ? Y
```

GENCPM fragt, ob es die Datenpuffer außerhalb des Common-Memory-Bereiches installieren soll. Diese Frage sollte mit ⟨Y⟩ beantwortet werden, um den TPA-Bereich nicht unnötig herabzusetzen. Alle Puffer werden bei unserem Computer außerhalb des Common-Bereiches installiert, da das BIOS über die XMOVE-Funktion verfügt. Die folgenden GENCPM-Fragen werden aus Platzgründen etwas verkürzt dargestellt:

```
Number of directory buffers for drive B: (#4) ? 4
Number of data buffers for drive B: (£4) ? 4
Number of directory buffers for drive C: (#0) ? 0
Number of data buffers for drive C: (#0) ? 0
Share buffer(s) with which drive (B): ? B
Number of directory buffers for drive D: (#0) ? 0
Number of data buffers for drive D: (#0) ? 0
Share buffer(s) with which drive (B): ? B
Number of directory buffers for drive E: (2) ? 2
```

Da die RAM-Floppy (Laufwerk E) mit 128-Byte-Sektoren arbeitet, werden für Drive E keine Datenpuffer angelegt. Die letzte GENCPM-Speicher-Meldung muß folgendermaßen aussehen:

```
Available space in 256 byte pages:
TPA = 00F0H, Bank 0 = 0000H,
Other banks = 0000H
```

Die TPA muß auf Page 0F0h liegen, alle Speicherbänke sollten jetzt belegt sein.

```
Accept new buffer definitions (Y) ? Y
BNKBIOS3 SPR F600H 0A00H
BNKBIOS3 SPR A700H 1900H
RESBDOS3 SPR F000H 0600H
BNKBDOS3 SPR 7900H 2E00H

*** CP/M 3.0 SYSTEM GENERATION
    DONE ***
```

Nun hat GENCPM seinen Lauf beendet und eine neue CPM3.SYS-Datei auf der Diskette erstellt. Um eine neue Systemdiskette zu erhalten, muß noch die Datei CCP.COM kopiert und mit COPYSYSN der CP/M-Loader auf die Systemspuren gebracht werden.

Die Generierung des CP/M-Loaders

Der CP/M-Loader stellt ein einfaches nichtgebanktes CP/M-System dar, das auf den Systemspuren der Diskette steht.

Der CP/M-Loader besteht beim mc-CP/M-Plus-Computer aus zwei Teilen: dem von Digital Research gelieferten CPMLDR-BDOS (CPMLDR.REL) und dem spezifischen CPMLDR-BIOS. Das Loader-BIOS wiederum besteht aus dem Hauptteil LDRBIOS und dem Tabellenteil LDRDPB0, der formatspezifische Informationen ähnlich wie AUTODPB0 enthält. Ändert man das Loader-BIOS, so muß man die geänderten Teile neu assemblieren und nach folgendem Schema mit den übrigen zu einem neuen CP/M-Loader zusammenbinden:

```
A>RMAC LDRDPB0
A>LINK CPMLDR0.400[L4000]=
CPMLDR,LDRBIOS0,LDRDPB0
```

Die Anweisung L4000 in eckigen Klammern veranlaßt den Linker, ein lauffähiges Programm, den CPM-Loader mit Namen CPMLDR0.400, mit der Startadresse 4000h zu erzeugen. Das ist die Adresse, auf der auch das Boot-EPROM den CP/M-Loader ablegt. Nachdem ein neuer CPM-Loader erzeugt ist, muß dieser mit Hilfe von COPYSYSN auf die Systemspuren kopiert werden:

```
A>COPYSYSN CPMLDR0.400
```

Natürlich läuft der EPC-Computer mit einer entsprechenden Systemdiskette auch ohne die oben aufgeführte Systemgenerierung. Aber Ihnen sollen alle Möglichkeiten dieses Computers bis hin zum Eingriff ins Betriebssystem offenstehen.

Fortsetzung folgt, mit der Schilderung der Hilfsprogramme für den mc-CP/M-Plus-Computer.

Frank Oettle, Thomas Reichler

Der mc-CP/M-Plus-Computer

Einige Hilfsprogramme

Weil der mc-CP/M-Plus-Computer in bezug auf die Floppy-Disk-Routinen sehr universell ist, müssen zu den Standard-Hilfsprogrammen von CP/M-3 noch einige Routinen kommen, die auf die speziellen Eigenschaften des BIOS eingehen. Das Formatieren der Disketten, das Kopieren der Systemspuren und vieles mehr wird durch diese Programme systemgerecht unterstützt.

Leider existiert keine verbindliche Norm über die Verfahren bei der Aufzeichnung von Daten auf 5¼-Zoll Disketten. Das mc-CP/M-Plus-BIOS enthält deshalb Disk-Parameter für fast jedes Format und kann neu eingelegte Disketten automatisch identifizieren und sich nach deren Format einstellen. Dadurch entstehen aber Probleme bei der Zusammenstellung von bootfähigen Arbeitsdisketten. Das beginnt schon beim Formatieren. Damit Sie dabei keine Schwierigkeiten bekommen, gibt es einen Format-Manager, FM genannt, der fast alle Formate beherrscht. „Fast" soll hier bedeuten, daß wir alle Formate, die wir kannten eingearbeitet haben, daß aber vielleicht noch irgendein exotisches Format (zum Beispiel mit variabler Drehzahl) existiert, das unser BIOS nicht kennt. *Bild 1* zeigt, was wir können.

Der Format-Manager

Diskettenformate, die man ständig benötigt, trägt man am besten in die AUTODPB0-Datei ein und hat sie fortan immer parat. Zur Untersuchung „wildfremder" und nur gelegentlich verwendeter Formate werfe man den Format-Manager an. Es erscheinen dann auf dem Bildschirm die Parameter des Disk-Parameter-Blockes 1.

Mit der Eingabe einer der Zahlen von 1 bis 9 kann jeder der ersten 9 (im BIOS eingearbeiteten) Parameter-Blöcke zur Bearbeitung angewählt werden. Hat man seine Wahl getroffen, so steht dieser Parameterblock für Änderungen sowohl des physikalischen als auch des logischen Formates zur Verfügung. Mit „P" kann man die „Physik" anwählen.

```
------------------------------------------------------------------------
FORMAT - MANAGER                  Copyright (C) 1984 by oettle & reichler
------------------------------------------------------------------------

Current format number: 8

PHYSICAL PARAMETERS:     LOGICAL PARAMETERS:      RESULTING PARAMETERS:

Tracks:          77      Offset for system:  2    Size in kB.:   248832
Sides:            1      Directory max.:    64    Size in blocks:   243
Sectors/Track:   26      Block length:    1024    Blockshift:         3
Sector length:  128      Skew faktor:        6    Blockmask:          7
Length GAP3:      7                             Extentmask:         0
Double step:      N      SKEW TABELLE:           CheKsum size:      16
Reverse track:   40
Reverse seek:     N      1,  7, 13, 19, 25,  5, 11, 17, 23,
MINI 5.25 Zoll:   N      3,  9, 15, 21,  2,  8, 14, 20, 26,
Steprate time:    3      6, 12, 18, 24,  4, 10, 16, 22,

Format (1 to 9), 'P' (Phys.), 'L' (Log.), 'S', 'R', 'W' or 'Q' (Quit) ? W

Filname to write Diskformat ?  I8M3740
------------------------------------------------------------------------
```

Bild 1. So erscheint das Menue des Format-Managers auf dem Bildschirm.

Das physikalische Format

Unter „Tracks" gibt man die Gesamtzahl der Spuren des Laufwerkes an. Unter „Sides" wird 1 oder 2 eingegeben, unter „Sectors/Track" kann man beliebige Werte eingeben, der Sinn wird nicht geprüft, die Größe MSEC wird neu eingestellt. Unter „Sector/Length" muß einer der Werte 128, 256, 512 und 1024 eingegeben werden, andere Werte werden überlesen. Length GAP 3 ist eine Größe, die nur bei multi-I/O eine Rolle spielt. Bei Bedarf ziehe man das Datenblatt des µPD 765 zu Rate. „Double Step", dieser Eintrag spricht den Teil des BIOS an, in dem die Spurdichte festgelegt wird. Mit Reverse Track gibt man die Spurnummer ein, ab der auf Seite 1 des Laufwerkes umgeschaltet werden soll. Allerdings geht das nur für alle die Laufwerke gemeinsam, die auf diese Art die Seitenanwahl realisieren. Das sind die, die mit „Reverse Seek" arbeiten (siehe mc 1984, Heft 12). Mit Mini 5.25 Zoll wird der Grund-Typ eingestellt. Die Steprate ist ebenfalls einstellbar, nach großen oder kleinen Laufwerken getrennt. Während der Parametereingabe kann man die schon richtig eingetragenen Werte mit CR bestätigen.

Fallstricke

Auf keinen Fall sollte man das Format ändern, mit dem man gerade arbeitet. Man sägt sich dann den eigenen Ast ab. Das kann nicht geschehen, wenn man mit Laufwerk E arbeitet, der RAM-Disk. Wenn man ein Format kreiert, das so ähnlich zu einem bereits vorhandenen ist, daß das BIOS keine Unterschiede mehr erkennt, dann kann es Datensalat geben.

Das logische Format

Nachdem ein „L" dem Format-Manager gegeben wurde, kann man den System-Offset eingeben, der die Systemspuren eines Formates reserviert. Mit „Directory max." wählt man die Anzahl der möglichen Directory-Einträge (fast beliebig einstellbar). „Blocklength" legt die Anzahl der Bytes fest, die pro Directory-Eintrag adressiert werden können: 1024, 2048, 4096, 8192 und 16 384. Der Skew-Faktor steht normalerweise auf 0. Damit wird die Voreinstellung im CBIOS benutzt. Man kann eine neue Skew-Tabelle erzeugen, wenn man eine Zahl ungleich Null angibt. Mit „−1" kann man etwas Exotisches von Hand eingeben. Aus technischen Gründen kann der Formatmanager die neu erzeugte Skew-Tabelle

nur auf dem Platz des Parameter-Blockes für das zehnte Format ablegen, weshalb eben nur neun Formate mit dem Manager einstellbar sind und auch nur neun Formate nach dem Ändern des Skew-Faktors von FM aus zur Verfügung stehen. Die Länge der Tabelle ist auf 26 Einträge begrenzt, was 26 umzurechnenden Sektoren entspricht. Der Formatmanager schreibt erst auf Anforderung mit „S" die neu errechneten Werte in die zugeordneten Diskparameterblöcke. Ein Kommando W erlaubt die Speicherung in einer eigenen Datei (Extension .FMT), die vom Manager aus wieder mit R gelesen werden kann. Mit Q kann man den Manager nach S verlassen, mit CTRL C ist dann das neue Format für das aktuelle Laufwerk festgeschrieben.

Das Formatieren

Zu einem „Multi-Format-BIOS" gehört ein Disketten-Formatierprogramm, das alle möglichen Formate auch erzeugen kann. Wiederum menügeführt, findet man hier die folgenden „Standard-Formate":

★ Double oder Single Sided
★ 8 Zoll oder 5.25 Zoll
★ 77 oder 80 Spuren (8 Zoll)
★ 40 oder 80 Spuren (5.25 Zoll)
★ 26×128, 14×512, 15×512, 16×512, oder 9×1024 (8 Zoll)
★ 16×256, 08×512, 09×512, 10×512 oder 5×1024 (5.25 Zoll)

Bei allen mc-Computern ist als empfohlenes Standardformat neu festgelegt: 5¼-Zoll mit 80 Spuren auf zwei Seiten zu je 5 Sektoren mit je 1024 Byte. Der mc-CP/M-Plus-Computer kann natürlich alle anderen Formate formatieren und auch damit, soweit das System darauf existiert, booten und arbeiten.
Das Formatieren geht mit Frage und Antwort vor sich. Meist werden die Parameter schon so weit vorgeschlagen, daß man nur noch mit ja oder nein antworten muß. Auch die Formatierung von 8-Zoll-Disketten wird so durchgeführt, wobei dort zum Beispiel von 14 Sektoren zu je 512 Bytes (schnell aber geringe Kapazität) bis 9 Sektoren zu je 1024 Byte pro Sektor alles gängige gewählt werden kann, wenn nicht der Standard 26 × 128 in Frage kommt. Wenn Sie 8-Zoll-kompatible Mini-Laufwerke fahren wollen, dann sollten Sie auf höchste Disketten-Qualität achten, zum Beispiel Maxell MD2-HD, Double Sided, High-Density, Double Track oder BASF 2 HD High Density.
Generell ist zu sagen, daß Disketten beim Hersteller einer Prüfung unterzogen werden, die formatgebunden ist.

Tabelle 1. Die GAPs müssen so auf das Format eingestellt werden

	8" komp. (FD55 G)				5.25" komp. (FD55 B/F)			
	Sector	N	SC	Gap 3	Sector	N	SC	Gap 3
FM	128	00	1AH	1BH	128	00	12H	09H
FM	256	01	0FH	2AH	128	00	10H	19H
FM	512	02	08H	3AH	256	01	08H	30H
FM	1024	03	04H	8AH	512	02	04H	87H
FM	2048	04	02H	FFH	1024	03	02H	FFH
FM	4096	05	01H	FFH	2048	04	01H	FFH
MFM	256	01	1AH	36H	256	01	12H	0CH
MFM	512	02	0FH	54H	256	01	10H	32H
MFM	1024	03	08H	74H	512	02	08H	50H
MFM	2048	04	04H	FFH	1024	03	04H	F0H
MFM	4096	05	02H	FFH	2048	04	02H	FFH
MFM	8192	06	01H	FFH	4096	05	01H	FFH

Beispielsweise können Disketten in 40-Track-Laufwerken einwandfrei funktionieren, können aber bei 80 Spuren versagen, da sie in den Zwischenspuren nicht auf Fehler in der Beschichtung geprüft sind, wenn dies nicht ausdrücklich bestätigt ist (96 tpi). Beim mc-CP/M-Plus-Computer-BIOS kann ein Bit gesetzt werden, das 80-Spur-Laufwerke fähig macht, auch 40-Spur-formatierte Disketten zu lesen und darauf zu schreiben.
Beim Formatieren freier Formate muß auch die Lücke zwischen Identifikations-Bytes und Datenfeld eines Sektors eingestellt werden. Die *Tabelle 1* zeigt die Werte, mit welchen alles funktioniert. Ebenfalls muß bei freiem Format der Aufzeichnungstakt eingestellt werden, der 500 kHz bei 8-Zoll-DD beträgt, 250 kHz bei 8-Zoll-SD und bei 5¼-DD, 125 kHz bei 5¼-Zoll-SD. Er muß zur vorgewählten Dichte passen. Einer mit einem bestimmten Format eingestellten Diskette kann man auf den ersten Spuren noch ein zweites Format für das System aufprägen. Viele Urlader haben nämlich ein anderes Format als das geladene System.

Das Copysys-Programm

Das Copysys-Programm kopiert das CP/M-System von einer CP/M-Diskette auf eine andere. Die Disketten müssen dabei nicht die gleichen Formate aufweisen. Auch zwischen unterschiedlichen Laufwerken und Formaten kann kopiert werden. Das COPYSYSN-Programm ersetzt bei uns das unter CP/M-Plus von Digital Research gelieferte COPYSYS-Programm. COPYSYS von Digital Research kann nur zwischen Disketten gleichen Formats (IBM 3740) kopieren und ist deshalb nicht zu verwenden. COPYSYSN kopiert die reservierten Systemspuren von einer Diskette zu einer anderen. Es ist zu beachten, daß für diese Systemspuren ein Format gewählt wird, das von seiner Kapazität her den auf den Systemspuren befindlichen CP/M-Loader CPMLDR von ca. 6 KByte Länge abspeichern kann.
Um unter CP/M-Plus eine lauffähige CP/M-Systemdiskette zu erhalten, müssen die Files CPM3.SYS und CCP.COM kopiert werden. Dazu sollten die folgenden Fragen jeweils mit <Y> beantwortet werden, wenn COPYSYSN zunächst die Systemspuren bearbeitet hat:
Do you wish to copy CPM3.SYS ? Y
Function complete
Do you wish to copy CCP.COM ? Y
Function complete
Als Quell- und Ziellaufwerk werden hierbei die gleichen Laufwerke wie beim Kopieren der Systemspuren verwendet. Soll das Programm schon während der Laufwerksabfrage abgebrochen werden, so ist ^C zu betätigen.

Setzen des Jahr-Feldes

Zur Datums-Umrechnung benötigt das BIOS eine Jahresangabe, die mit Hilfe des Copysys-Programms geändert werden kann. Es genügt hinter den Aufruf des Programms das aktuelle Jahresdatum einzugeben. Dadurch wird das derzeitige Jahresdatum im Systemfile aktualisiert. Beispiel: Das Jahrfeld im Systemfile wird auf 1985 gesetzt. A > COPYSYSN 85

Kopieren von Anwenderfiles auf die Systemspuren

Unter COPYSYSN besteht die Möglichkeit, Anwenderprogramme auf die Systemspuren zu übertragen. Das können Monitorprogramme, modifizierte Systemfiles o. ä. sein. Hinter den Aufruf des Programms ist dazu der gewünschte Systemfile anzugeben. Beispiel: Kopieren des Files USER.PGM auf die Systemspuren: A > COPYSYSN B: USER.PGM