



Mit eigenem Format

Flexibles Formatierprogramm (nicht nur) für ATs

Volker Bührmann

Im allgemeinen lassen sich Disketten mit FORMAT von DOS recht unproblematisch formatieren, wenn auch der Parameter-Rattenschwanz zur Formatwahl lästig sein kann. FORMAT weigert sich aber, 720-KB-Disketten im Multifunktionslaufwerk des AT zu formatieren, und in XTs ist es nur über DRIVER.SYS möglich, 720-KB-Disketten zu erstellen. Eine universelle Formatter-Unit hat aber noch diverse andere Vorzüge.

[Unterthema: Gängige DOS- und Atari-Formate](#)

[Unterthema: Definierte FAT-Zeiger](#)

[Unterthema: Media-Bytes nach IBM](#)

[Unterthema: Disk-Status-Byte](#)

[Unterthema: Disk Drive Parameter Table \(DDPT\)](#)

[Unterthema: Interrupt 13h des ROM-BIOS](#)

[Unterthema: Gegen Double-Stepping](#)

[Unterthema: program format](#)

[Unterthema: unit Formunit](#)

[Bitte beachten Sie auch diese Ergänzung!](#)

[Bitte beachten Sie auch diese Ergänzung!](#)

Das hier vorgestellte Formatierprogramm unterstützt die am weitesten verbreiteten Formate in der DOS-Welt, also

```
360 KByte (5,25")
720 KByte (3,5 und 5,25")
1,2 MByte (5,25" MF)
1,44 MByte (3,5" MF)
```

Vielleicht vermißt jemand die einseitigen Formate (Option /1) beziehungsweise die 8-Sektor-Formate (Option /8), mit denen man Disketten mit Kapazitäten von 160, 180 und 320 KByte erzeugen kann. Bedeutung hat davon allenfalls noch das einseitige Format, weil einige Ur-PCs tatsächlich mit einseitigen Laufwerken ausgestattet waren. Diese älteren Formate unterstützt unser Formatierer jedoch nicht, denn zu den seltenen Anlässen, wo man sie braucht, kann man auch auf FORMAT von DOS zurückgreifen.

Wozu eigentlich?

Am interessantesten ist unser Formatierprogramm für AT-Besitzer, weil bei deren Multifunktionslaufwerken (MF-Laufwerke) unter DOS nur in den Formaten 360 KB (40 Spuren, Double Density) und 1,2 MB (High Density) formatiert werden kann. Ein MF-Laufwerk kann auf einem AT aber sehr wohl auch 720 KB (80 Spuren, Double Density) schreiben und lesen, wenn auch nur über einen kleinen Trick ([1], siehe Kasten). Alte AT-BIOS-Versionen (vor 2.x) ließen dabei auch das Formatieren von 720 KB über DRIVER.SYS zu, neuere verhindern das aber strikt. Hier hilft nur noch ein eigener Formatierer weiter.

Das ROM-BIOS eines einfachen XT kennt kein Format mit größerer Kapazität als 360 KByte. Dennoch kann DOS ab Version 3.2 ohne Probleme Disketten mit 80 Spuren einwandfrei schreiben und lesen, denn das wird vom BIOS nicht beeinflusst. Jedoch läßt sich ein solches Format nicht formatieren, denn dazu müßte das XT-ROM-BIOS Daten liefern, die es nicht hat. Erst mit Hilfe eines Treibers DRIVER.SYS läßt sich auch hier Abhilfe schaffen. Es ist aber unschön, nur für das relativ seltene Formatieren Speicher für einen Treiber verschenken zu müssen, so daß auch hier unser Formatierer gut eingesetzt ist.

Weiterführende Möglichkeiten betreffen die Erzeugung eigener Formate. Solche unterstützt unser Beispielprogramm zwar nicht explizit, anhand des Formataufbaus der DOS-Diskette, der noch beschrieben wird, können Sie das aber selbst implementieren. Weniger ratsam sind Privat-Formate mit 10 Sektoren pro Spur (400 und 800 KByte statt 360/720), denn irgendwo gibt's bestimmt mal Kompatibilitätsprobleme. Allerdings ist es mit DOS-FORMAT nicht möglich, 80-Spur-Disketten einseitig zu formatieren. Das kann aber zum Beispiel wichtig werden, wenn man Disketten für einen Atari ST aufbereiten will, der mit einem einseitigen Laufwerk ausgestattet ist.

Manchmal ist auch ein batch-fähiges Formatierprogramm wünschenswert, das also den Ablauf einer Batch-Datei nicht immerzu mit Nachfragen aufhält. Neben seiner standardmäßigen Menüführung läßt unser Formatter auch das zu. Nebenbei finden es viele Anwender, an deren Rechnern sich auch mal Neulinge tummeln, sehr unangenehm, daß man mit dem DOS-FORMAT jederzeit auch die Festplatte 'runderneuern' kann - diese Gefahr besteht mit unserem Formatter nicht.

Tief unten

Um einen universellen Formatierer für DOS-Disketten schreiben zu können, muß man zum einen wissen, was man dabei eigentlich auf die Diskette schreiben muß, zum andern, wie das geht.

Eine 'jungfräuliche', also nicht formatierte Diskette liefert als Information dasselbe wie eine unbespielte Tonbandkassette, nämlich Rauschen. Zwar könnte ein Programm da jetzt einfach Daten draufschreiben, jedoch empfehlen sich einige Konventionen, damit zum Beispiel verschiedene Programme nicht dieselben Bereiche nutzen. Auch ist eine Katalogisierung wünschenswert, anhand der man die gespeicherten Dateien, ihre Größe und den 'Füllstand' der Diskette erkennen kann und so weiter.

Man überläßt diese Organisation den 'unteren Schichten' des Betriebssystems. Diese wiederum gliedern den gesamten Diskettenbereich in eine definierte Struktur, schon auf physikalischer Ebene. Bei dieser physikalischen Formatierung schreibt der Floppy-Controller Kennungen auf die Scheibe, anhand derer er später alle markierten Datenbereiche wieder auffinden kann.

Daran schließt sich eine logische Formatierung an, die in der Regel auf das verwendete Betriebssystem abgestimmt ist. Dazu werden üblicherweise zu Beginn der Diskette Datenbereiche mit Katalog-Informationen (Dateinamen, Zeiger auf Dateiblöcke) beschrieben, die direkt auf die Verwaltungsmechanismen des Betriebssystems zugeschnitten sind.

Die physikalische Unterteilung einer Diskette gliedert sich in Seiten, Spuren und Sektoren, wobei die beiden ersten Parameter durch die mechanische Konstruktion des Floppy-Laufwerks selbst vorgegeben sind. Die zweiseitige Nutzung einer Diskette ist halt nur möglich, wenn das Laufwerk einen Schreib-/Lesekopf sowohl für die Ober- als auch die Unterseite der Diskette hat. Die Anzahl der Spuren besagt, in wie vielen Schritten die Köpfe radial über der Scheibe positioniert werden können. Bei einer Umdrehung der Scheibe überstreicht der Kopf (bei fester Position) eine Spur. In der IBM-Welt gibt es zur Zeit nur Laufwerke mit 40 oder 80 Spuren.

Die Unterteilung der Spuren in Sektoren ist hingegen von der Programmierung des Floppy-Controller-Chips abhängig. Die Sektorgröße kann zwar variiert werden, die PC-Welt benutzt aber grundsätzlich Sektoren von 512 Byte Länge. Je nach Drehzahl der Diskette, Kodierungsverfahren und Höhe der Datenrate paßt nur eine bestimmte Anzahl Sektoren (9 bis 18 bei PCs) in eine Spur hinein. IBM füllt die Spuren allerdings immer sehr vorsichtig, meistens würde noch ein Sektor mehr hineinpassen. Wie viele

jeweils benutzt werden, können Sie der Tabelle der logischen Disk-Parameter entnehmen.

Damit ist die erste Aufgabe eines jeden Formatierprogrammes für DOS-Disketten umrissen, nämlich den Controller-Chip zu veranlassen, alle Spuren einer Diskette einmal mit der maximalen Anzahl Sektoren vollzuschreiben. Das wichtigste dabei ist, daß danach controller-spezifische 'Randinformationen' wie Spur- und Sektornummern, aber auch etwa Synchronisationsinformationen auf der Scheibe stehen. Diese Informationen bleiben im weiteren Gebrauch der Diskette stets unverändert als eine Art Rahmen vor und hinter dem eigentlichen Datenbereich eines Sektors. Überschreibt man Daten durch neue, so betrifft das immer nur den reinen Datenbereich innerhalb des Rahmens. Beim Formatieren von DOS-Disketten werden die Datenbereiche mit einem Füllmuster vollgeschrieben, das nur aus einer Folge von Bytes mit dem Wert F6h besteht.

Jetzt muß sich aber noch eine logische Formatierung anschließen, damit DOS zum Beispiel bei einem Zugriff auf die Diskette gleich deren Format erfährt. Gäbe es eine solche nicht, müßte DOS die gesamte Disk abklappern, um die Anzahl der Seiten, Spuren und Sektoren quasi durch Ausprobieren zu ermitteln. Damit diese wichtige Basisinformation schnell verfügbar ist, legt man sie im ersten Sektor der ersten (äußersten) Spur auf der ersten Seite der Disk ab, dem Boot-Sektor.

Logik

Direkt an den Boot-Sektor schließen sich Sektoren an, die eine sogenannte FAT (File Allocation Table) enthalten. Dies ist eine verzeigerte Liste, anhand der DOS die Belegung des Diskettenspeichers vornimmt. Meist werden zwei FATs gleichen Inhalts benutzt, wobei die eine als Backup (Sicherheitskopie) für die andere dient, falls mal ein FAT-Sektor beschädigt wird. Danach folgt das Directory, wo unter anderem Dateinamen, -längen, Datum, Uhrzeit und Start-Sektor der Datei vermerkt sind.

An dieser Stelle beschränken wir die Betrachtungen der Disk-Strukturen auf die minimal benötigten Kenntnisse, um auch eigene Formate zu implementieren. Größeren Wissensdurst können Sie zum Beispiel anhand von [2] stillen.

Die DOS-Verwaltung vergibt Diskettenplatz nicht direkt als physikalische Sektoren, sondern in logischen Einheiten (Clustern), die aus Vielfachen eines physikalischen Sektors bestehen. Anders als bei den physikalischen Sektoren, zu deren Auffinden Seite, Spurnummer und Sektornummer innerhalb der Spur nötig sind, werden Cluster fortlaufend numeriert.

In der FAT befinden sich Zeiger auf jeweils einen solchen Cluster, wenn er durch eine Datei belegt ist, allerdings steht der Zeiger auf den Start-Cluster im Directory, wo auch der zugehörige Dateiname und andere dateispezifische Informationen zu finden sind. Jeder FAT-Zeiger (auch der Zeiger auf den Start-Cluster im Directory) hat dabei eine Doppelfunktion: zum einen markiert er einen von einer Datei belegten Cluster auf der Diskette, zum andern weist er innerhalb der FAT auf den Zeiger, der den Folge-Cluster definiert (verkettete Liste).

Bei DOS-Disketten beträgt die Cluster-Größe üblicherweise ein oder zwei physikalische Sektoren; möchte man größere Werte wählen (üblich bei Festplatten), so dürfen diese nur glatten Zweierpotenzen entsprechen. DOS führt hier keinen Bereichstest aus und macht bei falschen Cluster-Größen bis zur DOS-Version 4.0 allerlei Unfug.

Man kann leicht erkennen, daß die Cluster-Zahl selbst bei einer 1,44-MB-Disk (2880 Cluster zu 512 Bytes) keine 16-Bit-Zeiger erfordert. Bei DOS-Disketten ist es daher üblich, 12-Bit-Zeiger (1,5 Byte) zu verwenden, um den Speicherbedarf für die FAT möglichst gering zu halten. Einige der FAT-Zeigerwerte haben eine spezielle Bedeutung (siehe Tabelle). Bei größerer Cluster-Zahl (etwa auf Harddisk-Partitionen größer 10 MByte) werden FATs mit 16-Bit-Zeigern benutzt, seit DOS 4.0 sind für Partitionen größer 32 MByte auch 32-Bit-Zeiger möglich.

Je größer man einen Cluster macht, desto weniger Cluster-Nummern (kleinere FAT) muß DOS auf der Diskette zur Verwaltung von Dateiblöcken benutzen. Auf der anderen Seite belegt aber auch die kleinste Datei (ein Byte Länge) immer einen ganzen Cluster. Eine 360-KByte-Diskette ist also aus der Sicht von DOS voll, wenn man 354 Dateien von einem Byte Länge draufschreibt (354 mal 1024 Bytes ist das exakte Fassungsvermögen im Datenbereich).

Für den Formatierer braucht man eigentlich nur zu wissen, daß es feste Konventionen für die von IBM benutzten Disketten-Formate gibt, die die Größe der FAT und des Directory in Vielfachen von physikalischen Sektoren festlegen. Ihre jeweilige Größe bestimmt auch unmittelbar deren Lage auf der Diskette, weil grundsätzlich Boot-Sektor, FAT- und Directory-Sektoren 'nahtlos' aufeinanderfolgen, wobei aber der Wechsel von erster zu zweiter FAT und von zweiter FAT zu Directory prinzipiell auf Sektorgrenzen erfolgt. Auch der Bereich hinter dem Directory, der zum Speichern von Dateien zur Verfügung steht - nur dieser Bereich wird über FAT-Zeiger verwaltet -, beginnt mit einem neuen Sektor.

Weiterhin existiert die Konvention, zu Beginn jeder FAT das Media-Byte (siehe Parameter-Tabelle) gefolgt von zweimal FFh zu schreiben und den gesamten Rest mit 00h zu initialisieren. Ebenso werden auch die Directory-Bereiche mit 00h gefüllt, wohingegen das F6h von der physikalischen Formatierung in den Datenbereichen belassen wird.

Boot-Sektor

Der stets als erstes gelesene Boot-Sektor hat neben der Speicherung der Formatinformationen - wie sein Name vermuten läßt - noch weitere Aufgaben. Sein weitaus größter Teil enthält das Umlade-Programm für DOS, den Booter. Beim Kaltstart des Rechners lädt das ROM-BIOS dieses kleine Maschinenprogramm in den Speicher und führt es aus. Der Lader versucht nun, die PCDOS-Dateien IBMBIO.COM und IBMDOS.COM (bei MSDOS IO.SYS und MSDOS.SYS) von der Diskette zu laden. Sind die Systemdateien nicht auf der Scheibe, gibt er eine entsprechende Meldung aus.

In unserem Formatierer haben wir der Einfachheit halber die Option für eine Übertragung der Systemdateien weggelassen. Durch die Batch-Fähigkeit unseres Formatters lassen sich leicht Batch-Dateien schreiben, die einen Aufruf des Systemübertragungsprogramms SYS.COM enthalten. Da SYS auch gleich einen neuen Boot-Sektor anlegt, konnte weiterhin der Boot-Code im Boot-Sektor auf die Ausgabe der Meldung 'Keine Systemdiskette' verkürzt werden. Übrigens haben wir auch auf die Namensgebung für Disketten verzichtet, weil dies - zumindest ab DOS 3.3 - über LABEL.COM möglich ist.

Parameter

In den Anfangstagen des DOS gab es nur die eingangs aufgeführten Formate bis maximal 360 KByte Kapazität. Zur Unterscheidung wurde - bis DOS 3.1 einschließlich - nicht der komplette Formatparametersatz, sondern nur das sogenannte Media-Byte zu Beginn der FATs benutzt (siehe Tabelle). Dieser Mechanismus funktioniert auch bei neuen DOS-Versionen weiterhin, 'alte' Disketten, selbst wenn sie keinen Parametersatz enthalten, werden anhand des Media-Bytes korrekt verwaltet. Umgekehrt kennt DOS bis Version 3.1 nur in Ausnahmefällen (einige Laptops) bereits das 80-Spur-Format mit 720 KByte. Wer also mit älteren DOS-Versionen arbeitet, kann zwar unseren Formatierer benutzen, später aber einige Formate weder lesen noch schreiben.

Ab DOS 3.2 hingegen wird die Formatparametertabelle im Boot-Sektor ausgewertet, vor der allerdings noch ein paar andere Bytes liegen. Ganz am Anfang des Boot-Sektors steht ein Sprungbefehl (JMP) über alle Tabellen hinweg ins eigentliche Ladeprogramm. Nach diesen drei Bytes folgt ein 8 Byte langer Kennungs-String, meist ein Hersteller-Name nebst DOS-Version.

Erst in den darauffolgenden Feldern steht die Formatparametertabelle, eine Mischung aus physikalischen und logischen Daten. Zu den physikalischen zählen Anzahl der Bytes pro Sektor, Gesamtzahl der Sektoren, Sektoren pro Spur und Anzahl der Köpfe (Seiten). Die logischen Angaben beziehen sich auf

DOS-Konventionen: Sektoren pro Cluster, reservierte Sektoren (Boot-Sektor), Anzahl der FATs und so weiter, wie in der Tabelle erläutert.

Bei der Schaffung eines 'eigenen' Formates (oder dem Aufbau eines bereits bekannten Zielformates, etwa für Atari) ist das wichtigste, daß der Formatierer die Angaben im Parameterblock wirklich exakt bei der Initialisierung der Datenfelder von FAT und Directory umsetzt. Außerdem muß natürlich auch die Größe der nutzbaren Datenbereiche aufgehen, damit man weder Platz verschenkt noch der Fall eintritt, daß DOS Platz belegen will, der gar nicht mehr auf der Scheibe liegt.

Auf die Formatparameter folgen einige Null-Bytes (bisher noch nicht belegt). Es gibt bei DOS 3.2 und 3.3 noch eine weitere Tabelle im Boot-Sektor, die Angaben über laufwerksspezifische Parameter (Steprate, Motorhochlaufzeit und ähnliches) enthält, die 'Diskette Drive Parameter Table' (DDPT, dazu später noch mehr). Damit konnten die vom ROM-BIOS beim Kaltstart vorgegebenen Parameter während des Bootens überschrieben werden. Diese Tabelle liegt von 2Bh bis 35h, jede eingetragene 00h bedeutet, daß der Original-Parameter aus dem BIOS erhalten bleiben soll. Das ist auch die Default-Einstellung des Boot-Sektors, den unser Formatierer anlegt (obwohl ohnehin der Boot-Lader, der diese Tabelle auswertet, in unserem Boot-Sektor fehlt).

Ab DOS 4.0 und bei der neuesten OS/2-Version hat es im gesamten Tabellenbereich einige Änderungen gegeben. In der Diskparametertabelle zum Beispiel ist der letzte Eintrag ('hidden Sectors') jetzt als Doppelwort (32 Bit) definiert. Dieser Eintrag hat allerdings nur für Festplatten Bedeutung, denn über diese Sektoren 'verstecken' die Partitionen ihre Partitionsinformationen voreinander. Auf DOS-Disketten ist dieser Eintrag stets 00.

Die laufwerksspezifischen Daten lassen sich ab DOS 4.0 auch nicht mehr über Änderungen in einer disk-basierten DDPT ändern. Das ist eigentlich vernünftig, denn die Rechnerhersteller sollten schließlich die optimalen Daten der eingebauten Laufwerke am besten kennen und gleich im ROM-BIOS einbauen. Leider benutzen die meisten fernöstlichen Board-Hersteller zur Sicherheit die ungefährlichsten Parameter, wodurch sich oft genug 'schreckliche Geräusche' bei der Kopfbewegung in den Laufwerken ergeben [3].

Im DOS-4.0-Boot-Sektor befindet sich ab Adresse 27h im ehemaligen DDPT-Bereich statt dessen die verschlüsselte Diskettenbezeichnung, die 'Datenträgernummer'. Um all diese Neuerungen braucht man sich aber beim Formatieren keine Sorgen zu machen. Wenn DOS 4.0 auf 'alte' Boot-Sektoren stößt, behandelt es diese ebenfalls korrekt und leitet aus den DDPT-Werten nicht etwa eine Datenträgernummer ab.

Um unser Formatierprogramm auf die wesentlichen Punkte zu beschränken, haben wir stets den einfachsten, universell funktionierenden Weg gewählt und folglich auch keine Datenträgernummer-Erzeugung vorgesehen. Sollte Ihnen an einer Nachrüstung gelegen sein, dann können Sie in [4] nachlesen, wie diese Nummer erzeugt werden muß.

Wo ansetzen?

Das physikalische Formatieren kann man auf PCs oder ATs auf drei Arten bewerkstelligen. Die hardwarenächste Methode ist die direkte Programmierung des Floppy-Controller-Chips μ PD765, die allerdings eine Menge recht unangenehmer 'Register-Popelei' erfordert (auf langsamen Rechnern ist dabei sogar DMA-Programmierung nötig). Allerdings ist man bei dieser Methode 'völlig frei' von allen IBM- und DOS-Formatzwängen.

Seit Version 3.2 stellt DOS über Funktion 44h (IOCTL, Subfunktion 0Dh) einen Mechanismus bereit, mit dem man unter anderem auch Disketten formatieren kann. Der Aufwand für den Programmierer ist allerdings auch hier ganz beträchtlich, denn es muß ein recht komplexer Datensatz an (schlecht dokumentierten) Geräteinformationen aufgebaut werden. Diese Funktion ist außerdem schon wieder so komplex und mit Plausibilitätsprüfungen durchsetzt, daß man ein 720-KB-Format für MF-Laufwerke daran nicht 'vorbeimogeln' kann.

Der günstige Mittelweg liegt bei den Funktionen des ROM-BIOS-Interrupt 13h. Damit liegt man recht komfortabel 'oberhalb' der reinen Controller-Programmierung, hat aber bezüglich der gewünschten Formate unterhalb von DOS noch ausreichend Freiheiten zur Umgehung der typischen IBM- und DOS-Repressalien. Allerdings hat man in der BIOS-Dokumentation durch eine 'geschickte Weglassung' eine kleine Falle gestellt, auf die wir noch ausführlich zu sprechen kommen. Die INT-13h-Routinen, die unser Formatierprogramm benutzt, sind alle ausführlich in einer Tabelle beschrieben.

Gründlich vorbereiten

Nach dem Start des Programms wird als erstes das Hauptprogramm der Unit 'formunit' ausgeführt. Dieses speichert in den Variablen 'laufwerka' und 'laufwerkb' den Typ des jeweiligen Laufwerks ab, den die Funktion 'config' via Interrupt 13h, Funktion 08h ermittelt. Diese Funktion findet man grundsätzlich in ATs, auch schon in ganz neuen PC/XTs und wohl auch in den meisten ROM-BIOS-Erweiterungen von XT-Controllern, die auch 1,2- und 1,44-MByte-Formate unterstützen. Wenn kein 'Response' auf Funktion 8 erfolgt (ältere XTs), wird der Laufwerkstyp 360 KByte zugrunde gelegt, der aber eine Bedienung von 720-KB-Typen ebenfalls zuläßt. Die Anzahl der vorhandenen Laufwerke muß in diesem Fall separat über INT 11h (Read Equipment List) ermittelt werden.

Damit stehen Anzahl und Beschaffenheit der Laufwerke fest, aus denen man auch die jeweils möglichen Formate ableiten könnte. Es erfolgt hier jedoch noch keine Prüfung auf Stimmigkeit, so daß auch durchaus unsinnige Kombinationen möglich sind, aber dazu später mehr. Die Formatwahl wird dem Unterprogramm 'diskformat' mit Anzahl der zu formatierenden Spuren und Sektoren pro Spur übergeben. 'diskformat' richtet nun als erstes einen dynamischen Sektorenspeicher ein, der die Größe der zu formatierenden Sektoren pro Spur umfaßt. Der maximale Wert wurde dabei auf 18 Sektoren pro Spur begrenzt. Der weitere Ablauf gliedert sich in folgende Blöcke:

- Schreibrate und physikalische Parameter auf das gewünschte Format setzen
- 11 Byte lange Laufwerkstabelle einrichten (DDPT)
- Spuren formatieren
- Spuren verifizieren (abschaltbar)
- BOOT-, FAT- und ROOT-Datenstrukturen schreiben

Der erste große Block ist das Einstellen des Controllers auf die gewünschte Schreibrate, die bei AT-Controllern 250, 300 und 500 kBit/s betragen kann (je nach Laufwerksdrehzahl und Schreibdichte). Zusammen mit dieser Information, die in den sogenannten Disk-Status-Bytes an Adresse 40:90h und 40:91h (Laufwerk A: und B:) abgelegt sind, wird auch ein sogenanntes Doppel-Step-Bit übergeben (siehe Tabelle). Es veranlaßt das BIOS, im 80spurigen MF-Laufwerk bei der Bearbeitung von 360-KB-Disketten (40spurig) immer eine Spur zu überspringen.

Normale XT-Controller arbeiten immer mit 250 KBit/s, XT-Spezial-Controller für AT-Formate (mit eigenem ROM-BIOS) verhalten sich jedoch weitgehend wie AT-Controller.

Die Einstellung der Schreibrate nimmt das BIOS an sich automatisch vor, wenn man die Funktion 18h mit Laufwerksparemtern aufruft, die das BIOS mit einem der Standardformate in Einklang bringen kann. Ein Aufruf von Funktion 17h bewirkt im Prinzip dasselbe, diese ältere Funktion kennt aber noch nicht die 1,44-MB-Disketten. Wenn sich jedoch bei mehrfachen Aufrufen der Funktion 18h durch 'schreibrate' zeigt, daß keine fehlerfreie Antwort zurückkommt, dann wird ersatzweise auf Funktion 17h ausgewichen. Wenn diese Funktionen (bei älteren XTs) beide nicht vorhanden sind, dann werden sie auch nicht benötigt, denn in diesen Fällen gibt es keine unterschiedlichen Schreibraten einzustellen, und es kann einfach so formatiert werden.

Wenn man jedoch das Format '720 KByte auf MF-Laufwerken' realisieren will, muß man nach diesem

Aufruf die Status-Zellen selbst 'korrigieren'. Denn bei normaler Schreibdichte stellt das BIOS für das MF-Laufwerk grundsätzlich Doppel-Stepping ein. Ansonsten sind alle Parameter (bis auf die Spürzahl) identisch zum Format '360 KByte im MF-Laufwerk', und ein direkter Patch der Zellen 40:90h beziehungsweise 40:91h (Prozedur 'einzelstep') ermöglicht den Zugriff auf 80 Spuren - aber nur auf ATs mit Kombi-Controller. Die meisten XT-Spezial-Controller ignorieren nämlich leider den Inhalt dieser Speicherzellen, womit einigen Anwendern dann das 720-KB-Format im MF-Laufwerk vorenthalten bleibt.

Sollten Formatierversuche bei einigen Sektoren fehlschlagen, so werden diese einige Male wiederholt (sogenannte Retries), wobei vor jedem neuen Versuch ein Disk-Reset ausgeführt wird. Dann muß jedesmal auch wieder die Schreibrate korrigiert werden, denn nach einem Disk-Reset stellt das ROM-BIOS 'seine' Standard-Werte ein.

Der 'Vergessene'

Der Floppy-Controller-Chip braucht zum physikalischen Formatieren noch einige sehr wichtige Informationen, die von Format zu Format deutlich variieren (etwa die Länge der Gaps, das sind spezielle Füll- und Synchronisationsstrecken im Sektor-Rahmen). Auch diese finden sich in der bereits erwähnten Disk Drive Parameter Table, der DDPT. Die aktuelle DDTP wird über einen Zeiger lokalisiert, nämlich den Interrupt-Vektor 1Eh, liegt also ab der Speicherstelle 0000:0078h (es handelt sich nicht etwa um die Adresse von ausführbarem Code einer Interrupt-Routine).

Dieser Vektor weist im Normalzustand auf Adresse 0000:0522h, also in den sogenannten DOS-Datenbereich. Dort unterhält DOS einen Satz Standard-Parameter, typischerweise die Default-Werte für das Boot-Format, die während des Zugriffs auf andere Datenträger meist nur in einigen Bytes kurzzeitig umgepatcht werden.

Das Problem beim Schreiben eines Formatierers für beliebige Formate liegt nun darin, daß in der BIOS-Beschreibung zwar Mittel zur Erlangung eines Zeigers auf diese formatabhängigen Tabellen beschrieben sind, aber nicht, wann sie wie benutzt werden sollen. Der Aufruf der eigentlichen Formatierfunktion (AH=5) verlangt nämlich nicht danach.

Ignoriert man den DDPT-Zeiger, so wird stets der Standard-Parameterblock bei 0:522h benutzt, dessen aktueller Zustand zum Teil von vorangegangenen Zugriffen verfälscht sein kann, in den meisten Fällen aber für die Erstellung von 360-KB-Disketten paßt. Damit eignet er sich gleichzeitig für das in dieser Hinsicht identische 720-KB-Format, weshalb auch Spezial-Formatierer für 720 KByte in MF-Laufwerken, die den DDPT-Zeiger nicht definiert setzen, meistens funktionieren.

Die korrekten Parameter-Blöcke stehen im ROM-BIOS. In neuen ATs mit BIOS-Versionen größer 3.0 liefert die Funktion 18h einen Zeiger in den Registern ES:DI darauf zurück, allerdings nur dann, wenn ihr Parameter übergeben werden, die mit einem korrekten Standard-Format harmonieren. Dieser Zeiger sollte dann während des Formatierens (Funktion 5h) für den Interrupt-Vektor 1Eh eingetragen werden.

Eine Debugger-Überwachung dieses Zeigers offenbart, daß FORMAT von DOS 3.3 zum Beispiel so vorgeht. Ältere DOS-Versionen begnügen sich mit dem (offiziell statthaften) Umpatchen des DDPT bei 0:522. Leider patchen aber auch modernste Versionen von DOS und/oder Formatter immer noch gern im DDPT herum, selbst wenn er im ROM steht. Darüber konnte man lange Zeit amüsiert hinwegsehen, aber bei einem ROM-BIOS im Shadow-RAM, das sich aus irgendwelchen Gründen nicht schreibschützen läßt, richten solche Patches bleibende Schäden an. Die daraus resultierenden Fehler sind 'sporadisch teuflisch', und meist sucht man die Fehler beim Diskzugriff verzweifelt in Programmen, die damit überhaupt nichts zu tun haben.

Ältere BIOS-Versionen stellen nur die Funktion 17h zur Verfügung. Dann kann über Funktion 8h der Zeiger auf den Block ermittelt werden, der für die maximale Kapazität des unterstützten Laufwerks gilt. Die einzigen Laufwerke, bei denen es mehr als eine Kapazität gibt, sind bei solchen BIOS-Versionen die

1,2-MB-MF-Laufwerke. Wenn BIOS und DOS immer dafür sorgen, daß sich bei 0:522h die Default-Werte für das 360-KB-Format befinden, müssen nur für die abweichenden Formate Tabellen ermittelt werden. Dennoch unterstützt Funktion 8h neuerer BIOS-Versionen natürlich auch die 1,44-MB-Laufwerke und liefert auch hier den Blockzeiger für das größtmögliche Format. Üblicherweise benutzt man aber nur Funktion 18h, weil sie für jedes Format den erforderlichen Tabellenzeiger liefert.

Unser Formatter enthält diese Tabellen der Einfachheit halber direkt, so daß man von eventuellen BIOS-Abweichungen und DOS-Versionen unabhängig wird. Die Funktion 'laufwerkstabneu' sorgt für das vorübergehende Umsetzen des Tabellenzeigers, wobei für 360 und 720 KByte dieselbe Tabelle gilt. Das ist vor allem im Hinblick auf die Formatierung 'exotischer' Formate der einzig gangbare Weg; der BIOS-Funktion 18h setzt man dann nur irgendwas Passendes vor, damit sie die gewünschte Schreibrate einstellt.

Physikalisch formatieren

Eine winzige Kleinigkeit fehlt noch, bevor man endlich Funktion 5 zum Formatieren einer Spur aufrufen kann (Prozedur 'spurformat'), denn dieser Aufruf erwartet noch die Übergabe eines Zeigers auf die sogenannte Address Field List, deren Aufbau Sie der Funktionsbeschreibung des INT 13h entnehmen können. Unmittelbar nach dem Formatieren einer Spur wird je nach Inhalt der Variablen 'verify' die Spur auf Korrektheit geprüft, allerdings nur anhand der CRC-Prüfsumme (siehe Funktionsbeschreibung). Dazu dient das Unterprogramm 'readwriteverify' in der Unit, mit dem man je nach Parameterübergabe eine Spur (ganz oder teilweise) lesen, schreiben oder verifizieren kann. Dieser Vorgang wiederholt sich nun, bis alle Spuren der Diskette formatiert sind.

FORMAT von DOS schließt grundsätzlich einen Verify-Durchgang an jede formatierte Spur an. Treten in den ersten Spuren auch nach mehreren Versuchen Fehler auf (Boot-Sektor, FAT, Directory), so bricht der Formatter mit 'falsches Medium oder Diskette defekt ab'. Können erst später einige Sektoren auch nach mehreren Versuchen (Retries) nicht sauber formatiert werden, so merkt sich der DOS-Formatierer diese Sektoren und markiert sie später beim Schreiben der FAT als defekt, damit DOS sie nicht benutzt.

Unser Formatierer bricht bei aktiviertem Verify grundsätzlich nach Überschreiten der Maximalzahl von Fehlversuchen (Konstante 'versuch') mit der Meldung ab, daß diese Diskette unbrauchbar sei - was sie unserer Meinung nach dann auch ist.

Die Verify-Option läßt sich bei unserem Formatierer zwar abschalten, davon sollte man aber *nicht* deshalb Gebrauch machen, weil das den Formatiervorgang beschleunigt. Vielmehr läßt sich auf diese Art eine normale 360-KB-Diskette 'mal eben' komplett löschen, indem man sie mit 1,2 MByte und erst danach mit 360 KByte formatiert. Macht man das mit dem Formatierer von DOS, wird das sehr langwierig, weil er ja dauernd defekte Sektoren notiert.

Solches Löschen ist empfehlenswert, wenn man mit MF-Laufwerken Disketten für PCs aufbereiten muß. Wenn beispielsweise eine vom PC mit einem 40-Spur-Laufwerk (breite Spuren) formatierte Disk mit einem MF-Laufwerk (80 Spuren, diese sind schmaler) beschrieben wird, kann der PC diese Daten meist nicht lesen. Dem breiten 40-Spur-Kopf präsentiert sich dabei ein untrennbares Gemisch aus 'Resten' der 40-Spur-Information und der 80-Spur-Information.

Für PCs bestimmte, aber im AT zu erstellende Disketten sollten daher nur im MF-Laufwerk formatiert und beschrieben worden sein. Kann man das nicht gewährleisten oder hat man keine noch nie formatierten Disketten, ist eine Formatierung in High Density zum Löschen (alle Spuren, kein Doppel-Step) und eine anschließende 360-KB-Formatierung eine ziemlich sichere Methode.

Logisches Format

Die physikalisch formatierte Diskette würde DOS noch nicht akzeptieren, sondern bei einem Zugriff darauf mit derselben Fehlermeldung abbrechen, als wäre sie überhaupt nicht formatiert. Erst wenn Boot-Sektor, FATs und Directory eingerichtet und initialisiert sind, hat man eine unter DOS brauchbare

Diskette erzeugt.

Als Informationsquelle, wie diese Datenstrukturen aufgebaut sind, dient die Tabelle 'formtab', in der jeweils das formatabhängige Media-Byte, Zahl der Sektoren pro Cluster, die Einträge im Hauptverzeichnis und die Sektoren pro FAT enthalten sind. Alle anderen Daten lassen sich daraus errechnen (etwa Sektoren gesamt) beziehungsweise sind bei den vier unterstützten Formaten gleich. Bei eigenen Formaten kann es nötig sein, einige Parameter mehr zu variieren.

Diese Strukturen werden dann mit Hilfe des dynamischen Pufferspeichers 'puffer' innerhalb der Prozedur 'diskprep' organisiert. Zuerst wird der Boot-Sektor im Puffer angelegt (seine letzten Bytes lauten immer 55 AAh) und auf Disk geschrieben. Dann wird der Sektorpuffer gelöscht (00h), und die FAT- und Directory-Sektoren werden aufgebaut und geschrieben.

Allgemeines

Das Format-Programm ist in Turbo-Pascal 5.0 geschrieben und realisiert alle minimal benötigten Dinge zum Erzeugen einer für DOS akzeptablen Diskette. Nach dem Aufruf ohne oder mit falschen Parametern meldet sich ein einfaches Menü, das die verfügbaren Formate anzeigt, die über Zifferneingabe von 1 bis 4 ausgewählt werden.

Darüber hinaus zeigt das Menü das aktuelle Laufwerk und den Verify-Zustand. Standardmäßig ist 'Laufwerk A:' und 'Verify ON' eingestellt; die Eingabe von 'L' toggelt zwischen A: und B:, ein Druck auf die V-Taste zwischen Verify ON und OFF. Sollte nur das Laufwerk A: vorhanden sein, ist die Änderung des Laufwerkes nicht möglich. Mit ESC lassen sich alle begonnenen Vorgänge und das Programm selbst abbrechen.

Verlangt man die Formatierung einer Diskette in einem vom eingebauten Laufwerk nicht realisierbaren Format, legt der Formatierer erst mal los und versucht, das Gewünschte zu vollbringen. Er merkt dann in der Regel erst am fehlgeschlagenen Verifizieren, daß irgendwas nicht stimmt. Damit man eventuell unbeabsichtigte Formatierungen frühzeitig bemerkt, sollte also normalerweise die Verify-Option aktiv bleiben.

Einige 'Fehlformate' kann der Formatierer dennoch nicht bemerken. Ein einfacher XT-Controller kann technisch keine High-Density-Formate bewältigen, so daß also nur die Formate 360 und 720 KByte verbleiben. Die 720 KByte wiederum kann man nur nutzen, wenn ein doppelseitiges 80-Spur-Laufwerk eingebaut ist. Der Formatierer kann nicht erkennen, wie das Laufwerk physikalisch gestaltet ist. Er wird folglich auch auf einem 80-Spur-Laufwerk das 360-KByte-Format nicht verweigern und dabei dann nur die ersten 40 Spuren beschreiben.

Er weigert sich aber auch nicht, auf einem 40-Spur-Laufwerk 80 Spuren zu formatieren. In diesem Fall hängt der Schreib-Lesekopf dann 40 Spuren lang 'an der Bande', und es wird immer wieder die letzte Spur formatiert (und korrekt verifiziert). Anschließend legt er - unwissentlich falsch - Boot-Sektor, FAT und Directory für eine 720-KB-Disk an. Auch CHKDSK wird einer solchen Diskette nicht anmerken, daß sie völlig falsch formatiert ist und weit weniger Kapazität hat. Erst wenn Schreibzugriffe oberhalb 40 Spuren fällig werden, hagelt es Schreibfehler.

Man könnte natürlich - auf jeden Fall bei ATs - schon vorab sehr genau festlegen, welches Format man auf welchem Laufwerkstyp zulassen möchte. Genau diese Restriktionen des DOS-FORMAT ('ungültiger Parameter') wollten wir aber mit unserem Formatierer ja überwinden. Wen das stört, der kann entsprechende Prüfungen einfügen.

Startet man das Programm über die Kommandozeile oder per Batch-Datei mit Parametern, so ist etwas Vorsicht geboten. Denn jetzt legt der Formatierer ohne jedwede Rückfrage los, und wenn man - in alter Gewohnheit - noch keine Diskette eingelegt hat, hängt man erstmal an einer Fehlermeldung fest, der man 'manuell' abhelfen muß. Solche Rückfragemöglichkeiten ('Disk eingelegt?') kann man zum Beispiel in

einer Batch-Datei über ECHO- und PAUSE-Befehl einfügen.

Maximal drei Kommandozeilen-Parameter sind möglich, die beiden ersten sind dabei obligatorisch:

FORMAT drv fmt vf

Mit `drv` gibt man das Ziellaufwerk A: oder B: an, mit `fmt` das gewünschte Format: 360, 720, 1200 und 1440. `vf` dient zum *Abschalten* des Verifizierens, indem man ein `n` als dritten Parameter angibt. Läßt man das `n` weg, wird immer verifiziert. Bei falschen oder fehlenden Parametern erfolgt eine Ausgabe der korrekten Kommandosyntax, und es wird in das normale Menü verzweigt. Bei Formatierfehlern im Kommandozeilenmodus bricht das Programm mit Exit-Code 01 ab, so daß man zum Beispiel einen längeren Batch-Job über eine Error-Level-Abfrage abbrechen kann.

Ausblick

Als Anregung für eigene Verfeinerungen kann man durch Umsetzen der Konstanten `seitenzahl` auf `1` erreichen, daß über den (dann nicht ganz zutreffenden) Menüpunkt `720 KB` eine einseitige 80-Spur-Diskette erzeugt wird. Das Programm benutzt dann allerdings weitgehend dieselben Parameter wie beim doppelseitigen Format (jedoch mit korrekter Kapazität), es ist daher weder ein echtes Atari- noch ein echtes PC-Format. Trotzdem geben sich beide Rechner damit zufrieden - aber wie gesagt, Verfeinerungen anhand der Tabellen sind ja leicht möglich.

Da in der Unit `Formunit` alle Funktionen enthalten sind, die man für Diskettenzugriffe benötigt, stellt sie eine allgemeine Grundlage für eigene Diskettenanwendungen dar. In einer der nächsten Ausgaben stellen wir ein Programm ähnlich DISKCOPY (inklusive DISKCOMP) vor, das auch große Disketten, die größer als der freie Arbeitsspeicher sind, in einem Rutsch - also ohne Diskwechsel - kopieren und verifizieren kann. (gr)

Literatur

- [1] M. Ernst, D. Grell, Damit die Scheibe spurt, c't 11/87, S. 216 ff.
- [2] W. Haaf, F. Middel, Daten auf Scheiben, File- und Diskettenstrukturen unter CP/M, DOS und TOS, c't 11/87, S. 241 ff.
- [3] S. Gipp, M. Wilde, Jetzt schnurrt die alte Säge, Einblicke ins Booten und Steppen, c't 11/87, S. 142 ff.
- [4] Marcus Gröber, DOS 4.0 durchleuchtet, c't 4/89, S. 170 ff.

Kasten 1

Gängige DOS- und Atari-Formate

<i>Atari</i>	<i>Atari</i>	<i>IBM PS/2</i>	<i>IBM PS/2-HD</i>	<i>IBM AT-HD</i>	<i>IBM</i>	<i>IBM</i>	<i>IBM</i>	<i>IBM</i>	<i>Erklärung</i>
80T/	80T/	80T/	80T/	80T/	40T/	40T/	40T/	40T/	
9S/SS	9S/DS	9S/DS	18S/DS	15S/DS	8S/SS	8S/DS	9S/SS	9S/DS	
00 02	00 02	00 02	00 02	00 02	00 02	00 02	00 02	00 02	Bytes pro Sektor (unt. DOS immer 512)
02	02	02	01	01	01	02	01	02	Sektoren pro Cluster (hier 1 oder 2)

01 00	01 00	01 00	01 00	01 00	01 00	01 00	01 00	01 00	reservierte Sektoren (hier immer 1)
02	02	02	02	02	02	02	02	02	Anzahl FATs (hier immer 2)
70 00	70 00	70 00	E0 00	E0 00	40 00	70 00	40 00	70 00	Directory-Einträge zu 32 Byte
D0 02	A0 05	A0 05	40 0B	60 09	40 01	80 02	68 01	D0 02	Anzahl Sektoren total
F8	F9	F9	F0	F9	FE	FF	FC	FD	Media-Byte
05 00	05 00	03 00	09 00	07 00	01 00	01 00	02 00	02 00	Sektoren pro FAT
09 00	09 00	09 00	12 00	0F 00	08 00	08 00	09 00	09 00	Sektoren pro Seite (8, 9, 15, 18)
01 00	02 00	02 00	02 00	02 00	01 00	02 00	01 00	02 00	Anzahl Köpfe (Seiten)
00 00	00 00	00 00	00 00	00 00	00 00	00 00	00 00	00 00	versteckte Sektoren (nicht bei Floppy)

Sämtliche Zwei-Byte-Angaben sind in der Reihenfolge ihres Auftretens im Boot-Sektor ab dem 12. Byte (gezählt ab 1) aufgeführt. Der reale 16-Bit-Zahlenwert, den sie repräsentieren, ergibt sich erst nach Vertauschen der beiden Bytes. Es sei noch angemerkt, daß auch die Formatierer der neuen DOS-Versionen bei der Erstellung von Disketten im alten 8-Sektor-Format keinen Disk-Parameter-Block in den Boot-Sektor eintragen, hier also stets die Formaterkennung über das Media-Byte läuft.

Kasten 2

Definierte FAT-Zeiger

Code	Bedeutung
(0)000h	Cluster frei
(F)FF7h	Cluster fehlerhaft, wird nicht benutzt
(F)FF8h - (F)FFFh	letzter Cluster einer Datei
(X)XXXh	Cluster einer Datei

Werte in Klammern gelten für 16-Bit-FAT

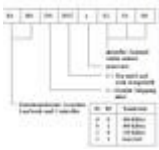
Kasten 3

Media-Bytes nach IBM

Byte	Format
F0h	2 Seiten, 80 Spuren zu 18 Sektoren
	(auch für künftige Formate)
F8h	Festplatte
F9h	2 Seiten, 80 Spuren zu 15 Sektoren
	2 Seiten, 80 Spuren zu 9 Sektoren
FCh	1 Seite, 40 Spuren zu 9 Sektoren
FDh	2 Seiten, 40 Spuren zu 9 Sektoren
FEh	1 Seite, 40 Spuren zu 8 Sektoren

FFh	2 Seiten, 40 Spuren zu 8 Sektoren
-----	-----------------------------------

Kasten 4



S2	S1	S0	aktueller Zustand
0	0	0	360-KB-Disk in 40-Spur-Laufwerk, Status ungeprüft
0	0	1	360-KB-Disk in MF-Laufwerk, Status ungeprüft
0	1	0	1,2-MB-Disk in MF-Laufwerk, Status ungeprüft
0	1	1	360-KB-Disk in 40-Spur-Laufwerk, Status geprüft
1	0	0	360-KB-Disk in MF-Laufwerk, Status geprüft
1	0	1	1,2-MB-Disk in AT-Laufwerk, Status geprüft
1	1	0	reserviert
1	1	1	720-KB-Disk in 720-KB- oder 1,44-MB-Laufwerk oder 1,44 MB-Disk in 1,44-MB-Laufwerk, Status geprüft

Kasten 5

Disk Drive Parameter Table (DDPT)

Byte	Funktion
00h	oberes Nibble: Steprate (‘rückwärts’ gezählt)
	Aktuelle Step-Rate ist abhängig von Controller-Hardware. Bei 8 MHz Controller-Takt: Fh = 1 ms, Eh = 2 ms usw., bei 4 MHz Takt verdoppelt sich die Zeit. Der optimale Wert ist laufwerksspezifisch (Geräuscentwicklung kontra Positionier-Fehler). Default gemäß IBM-Festlegung: Dh
	unteres Nibble: Head Unload Time
	Raster: 16 ms; Default gemäß IBM-Festlegung: Fh = 240 ms (sicherer Maximalwert, eigentlich laufwerksspezifisch)
01h	Bit 0:
	Non-DMA-Mode-Flag
	Default bei PC/AT und PS/2: 0 = DMA-Mode
	Bit 1-7:
	Head Load Time
	Raster: 2 ms, Default gemäß IBM-Festlegung: 01h (minimaler Wert, weil Köpfe beim Motor-On geladen werden und die viel größere Motor-Hochlaufzeit eh gewartet wird)
02h	Nachlaufzeit des Diskettenmotors
	Raster: 1/18,2 s, Default gemäß IBM-Festlegung: 25h
03h	Bytes pro Sektor:
	0 = 128 Bytes
	1 = 256 Bytes

	2 = 512 Bytes (Default gemäß DOS/IBM)
	3 = 1024 Bytes
04h	Maximale Sektorzahl pro Spur
	Defaults gemäß DOS/IBM: formatabhängig 8 bis 18 (12h)
05h	‘Länge’ von Gap3 beim Lesen/Schreiben
	Defaults gemäß Controller-Chip-Hersteller: 1Bh bei High-, 2Ah bei Double-Density (keine echte Gap-Länge, sondern chipinterne Steueranweisung)
06h	DTL (Data Length)
	Default gemäß Chip-Hersteller: FFh, wenn Sektorgröße ungleich 128 Bytes
07h	Längenfestlegung von Gap3 beim Formatieren
	Default gemäß Chip-Hersteller: 50h bei Double-, 54h bis 6Ch bei High-Density
08h	Wert des Füll-Bytes
	Wird beim Formatieren in alle Daten-Bytes des Sektors eingetragen. Default gemäß DOS/IBM: F6h
09h	Head Settle Time
	Wert für die Arbeitspause des Schreib-/Lesekopfes nach einem Spurwechsel. Raster: 1 ms, Default gemäß IBM: 0Fh (eigentlich laufwerksspezifisch, bei älteren Drives 25 ms)
0Ah	Hochlaufzeit des Motors auf Nenndrehzahl
	Raster: 0,125 s, Default gemäß IBM: 08h (laufwerksspezifisch, wird meist beim Lesen auf die Hälfte herabgesetzt, weil Fehler dabei durch weitere Versuche kompensiert werden können)

Die Disk Drive Parameter Tabelle (DDPT) enthält eine Mischung aus System-Daten (für das BIOS) und direkt an den Controller-Chip weitergereichten Daten, die dieser zum Formatieren benötigt.

Kasten 6

Interrupt 13h des ROM-BIOS

Der Interrupt 13h ist eine umfangreiche Unterprogrammbibliothek des BIOS. Alle Disketten- und Festplattenoperationen werden mit diesem System-Interrupt ausgeführt. Die gewünschte Unterfunktion wird durch den Inhalt des AH-Registers spezifiziert. Im folgenden sind nur die im Rahmen des hier vorgestellten Formatierprogrammes relevanten Funktionen aufgeführt.

Funktion 00h, Disk-Reset

Der Aufruf dieser Funktion löst ein Reset sowohl der Disketten- und Festplattenlaufwerke als auch des Controllers aus. Vor allem nach jeder fehlgeschlagenen Diskettenoperation sollte ein Reset durchgeführt werden, der eine Initialisierung der genannten Komponenten bewirkt.

Eingabe : AH = 00h
 DL = 0 oder 1 (Laufwerk A: oder B:)
 Ausgabe : Carry-Flag = 0: Operation erfolgreich, AH = 0
 Carry-Flag = 1: AH = Fehlercode (siehe unten)

Bemerkung: Der Reset wird bei allen verfügbaren Laufwerken ausgelöst, so daß der Wert in DL eigentlich nicht gebraucht wird. Mit Werten oberhalb 80h in DL spricht man die Festplatten an.

Die wichtigsten Fehlercodes:

01h:	nicht erlaubte Funktionsnummer
------	--------------------------------

02h:	keine Adreß-Markierung
03h:	Diskette ist schreibgeschützt
04h:	Sektor nicht gefunden
08h:	DMA-Überlauf
09h:	DMA-Transfer über Segmentgrenze
10h:	Lesefehler
20h:	Fehler des Disk-Controllers
40h:	Spur nicht gefunden
80h:	Timeout-Fehler, Laufwerk reagiert nicht

Der Inhalt aller Register außer BX, CX, DX, SI, DI, BP und den Segmentregistern kann durch diese Funktion verändert werden.

Funktionen 02h, 03h, 04h: Lesen, Schreiben, Verifizieren

Mit Hilfe dieser drei Funktionen können ein oder mehrere Sektoren einer Spur jeweils gelesen, geschrieben oder verifiziert werden. Bei den Funktionen muß ein Puffer für die Sektor-Daten übergeben werden. Die Verify-Funktion vergleicht nicht die Daten aus dem Pufferspeicher mit denen auf der Diskette, sondern prüft lediglich die Lesbarkeit und Korrektheit anhand einer Art Sektor-Prüfsumme (CRC, Cyclic Redundancy Check). Im folgenden sind nur die Angaben für den Diskettenbetrieb aufgeführt.

```

Eingabe:AH      =      02h für Lesen von Sektoren
          AH      =      03h für Schreiben von Sektoren
          AH      =      04h für Verifizieren von Sektoren
          AL      =      Anzahl der zu bearbeitenden Sektoren
          CH      =      Nummer der Spur
          CL      =      Nummer des ersten Sektors
          DL      =      Nummer des Diskettenlaufwerks (0 oder 1)
          ES      =      Segmentadresse des Puffers
          BX      =      Offset-Adresse des Puffers
Ausgabe:Carry-Flag = 0: Operation erfolgreich, AH = 0
          Carry-Flag = 1: AH = Fehlercode wie Funktion 00h
          AL      =      Anzahl der bearbeiteten Sektoren

```

Bemerkungen: Die Anzahl der zu bearbeitenden Sektoren im AL-Register ist in der Hinsicht eingeschränkt, daß mit einem Funktionsaufruf immer nur die logisch aufeinanderfolgenden Sektoren einer Spur auf einer Seite bearbeitet werden können. Die Anzahl der Spuren kann auf Festplatten bis 1024 reichen, die beiden 'fehlenden' Bits für diesen 10-Bit-Wert repräsentieren die obersten beiden Bits in CL. Es ist daher sinnvoll, diese beiden Bits in CL auch bei allen anderen Funktionen, die in DH eine Spur-Zahl erwarten, bei Diskettenzugriffen definiert auf 0 zu halten.

Der Inhalt aller Register außer BX, CX, DX, SI, DI, BP und den Segmentregistern kann verändert worden sein.

Funktion 05h, Diskette formatieren

Durch den Aufruf dieser Funktion wird jeweils eine komplette Spur einer Diskettenseite formatiert. Dazu muß der Funktion ein Zeiger auf eine sogenannte Address Field List (AFL) übergeben werden (Aufbau siehe unten).

Eingabe: AH = 05h

```

          AL      =      Anzahl der zu formatierenden Sektoren
          CH      =      Nummer der Spur
          DL      =      Nummer des Diskettenlaufwerks (0 oder 1)
          DH      =      Nummer der Diskettenseite (0 oder 1)
          ES      =      Segmentadresse AFL

```

```

BX      =      Offset-Adresse AFL
Ausgabe:Carry-Flag = 0: Operation erfolgreich, AH = 0
        Carry-Flag = 1: AH = Fehlercode wie bei Funktion 00h

```

Bemerkung: Der in ES:BX übergebene Puffer enthält für jeden der zu formatierenden Sektoren einen Eintrag, der aus vier aufeinanderfolgenden Bytes besteht.

1. Nummer der Spur
2. Nummer der Seite
3. logische Nummer des Sektors
4. Anzahl der Bytes in diesem Sektor
 - 0: 128 Bytes
 - 1: 256 Bytes
 - 2: 512 Bytes (Standard)
 - 3: 1024 Bytes

Während Spur- und Seitennummern innerhalb der Tabelle konstant sein müssen, erhöht man die Sektornummern üblicherweise von Eintrag zu Eintrag um `1`. Die Sektornummern dürfen aber auch in beliebiger Reihenfolge vergeben werden (Interleave).

Der Inhalt aller Register außer BX, CX, DX, SI, DI, BP und den Segmentregistern kann verändert worden sein.

Funktion 08h, Feststellung der Laufwerksdaten

Über diese Funktion lassen sich diverse laufwerksspezifische Parameter ermitteln, auch von Festplatten. Hier werden nur die Floppy-Parameter aufgeschlüsselt.

```

Eingabe:AH      =      08h
        DL      =      Nummer des Diskettenlaufwerks (0 oder 1)
Ausgabe:Carry-Flag = 1: AH = Fehlercode wie Funktion 00h
        Carry-Flag = 0: Operation erfolgreich, dann weiter
        AX      =      0
        BL      =      Typ wie in CMOS-RAM
                0: Kein Laufwerk angeschlossen
                1: 5,25"-Laufwerk mit 40 Spuren und 360 KByte
                2: 5,25"-MF-Laufwerk mit 80 Spuren und 1,2 MB
                3: 3,5- oder 5,25"-Laufwerk mit 80 Spuren und 720 KB
                4: 3,5"-MF-Laufwerk mit 80 Spuren und 1,44 MB
        BH      =      0
        CL      =      maximale Sektorzahl pro Spur
                    (plus 2 High-Bits für Spurzahl)
        CH      =      maximale Spurzahl (plus High-Bits in CL)
        DL      =      Anzahl vorhandener Diskettenlaufwerke
        DH      =      Anzahl der Seiten (Köpfe)
        ES:DI   =      Zeiger auf DDPT (siehe Extra-Tabelle)

```

Bemerkung: Wenn ein Laufwerk mehr als ein Format zulässt, werden stets die Daten für das Format mit der größten Kapazität geliefert. Wenn BL = 0 zurückgeliefert wird, obwohl die anderen Parameter korrekt sind, so sind Fehler im CMOS-RAM der Uhr (Batterien leer oder ähnlich) aufgetreten.

Der Inhalt aller Register außer BX, CX, DX, SI, DI, BP und den Segmentregistern kann verändert worden sein.

Der Inhalt aller Register außer BX, CX, DX, SI, DI, BP und den Segmentregistern kann verändert worden sein.

Funktion 17h, Diskettenformat/Datenrate festlegen

Veralteter Vorgänger der vielseitigeren Funktion 18h, um vor Aufruf von Funktion 05h die korrekten Disk-Status-Parameter (Datentransferrate, Single-/Double-Stepping) einzustellen.

```

Eingabe:AH      =      17h
        AL      =      1: 360 KByte in einem 360-KB-Laufwerk

```

```

                2: 360 KByte in einem 1,2-MB-Laufwerk
                3: 1,2 MByte in einem 1,2-MB-Laufwerk
                4: 720 KByte in einem 720-KB-Laufwerk
Ausgabe:Carry-Flag = 0: Operation erfolgreich, AH = 0
        Carry-Flag = 1: AH = Fehlercode wie Funktion 00h

```

Bemerkung: AL = 4 erst bei ATs ab Juni 1985 zulässig. Diese Funktion ist nur für Floppy-Laufwerke einsetzbar. Der Zeiger auf DDPT muß über Funktion 08h ermittelt werden.

Funktion 18h, Diskettenformat/Datenrate festlegen

Nach der Einführung der 1,44-MByte-Laufwerke war die Formatfestlegung der Funktion 17h nicht mehr ausreichend. Zwar kann auch eine 1,44-MByte-Diskette mit der Formatfestlegung der Funktion 17h/3 formatiert werden, doch ist die dafür verwendete DDPT nicht gleichwertig. Funktion 18h liefert auch einen Zeiger für die DDPT jedes unterstützten Formates (nicht nur Maximalwerte).

```

Eingabe:AH      =      18h
        CL      =      Sektoren pro Spur
        CH      =      Anzahl der Spuren
        DL      =      Nummer des Laufwerks
Ausgabe:Carry-Flag = 0: Operation erfolgreich, AH = 0
        Carry-Flag = 1: AH = Fehlercode wie Funktion 00h
        ES      =      Segmentadresse DDPT
        BX      =      Offset-Adresse DDPT

```

Bemerkung : Der zurückgelieferte Zeiger zeigt auf eine 11 Byte lange Tabelle im ROM-BIOS. Es können deshalb nur Formate unterstützt werden, für die eine Tabelle im ROM vorhanden ist. Passen die übergebenen Parameter (BIOS-abhängig in unterschiedlichen Grenzen) nicht zu den implementierten ROM-Tabellen, bricht die Funktion mit Fehler ab. Die DDPT ist in einer eigenen Tabelle erklärt.

Der Inhalt aller Register außer BX, CX, DX, SI, DI, BP und den Segmentregistern kann verändert worden sein.

Kasten 7

Gegen Double-Stepping

Das Formatieren auf 80 Spuren in MF-Laufwerken ist nur die halbe Miete, will man damit das 720-KB-Format nutzen. Denn um Disketten in diesem Format auch schreiben und lesen zu können, muß ebenfalls wieder dafür gesorgt werden, daß das Double-Stepping unterbleibt. Einige ROM-BIOS-Versionen (erstaunlicherweise die der meisten IBM-Rechner) machen das völlig selbständig, nur das Formatieren geht nicht (ohne Spezial-Formatierer). Bei fast allen Clone-BIOSsen muß man jedoch selbst für die richtige Schrittweite sorgen.

Im Prinzip genügt es, jedesmal 'zu Fuß' - etwa mit dem Debugger - das Disk-Status-Byte an Adresse 40:90h (oder 40:91h für Laufwerk B:) auf 54h umzupatchen. Allerdings wird nach jedem Öffnen des Laufwerkes beim nächsten Disk-Zugriff wieder der Standard-Wert für 360-KB-Disketten dort eingetragen. Das kann bei häufigem Gebrauch sehr lästig werden, und auch eine Batch-Datei mit DEBUG-Aufruf oder ein einfaches Umschaltprogramm hält immer nur bis zum nächsten Disk-Wechsel vor - billiges BACKUP ade.

Das hier abgedruckte kleine Assembler-Programm bringt Abhilfe. Es legt sich resident in den Interrupt 13h und fängt alle Aufrufe der Funktion 02h und 03h ab (Sektor lesen und schreiben). Wenn bei versuchten Lese- oder Schreibzugriffen die Fehlermeldung 04h in AH zurückkommt (Sektor nicht gefunden), dann liegt der Verdacht nahe, daß eine 80-Spur-Scheibe eingelegt wurde.

DOS macht im Fehlerfalle fünf Versuche. Nach zwei Fehlversuchen schaltet unser Treiber auf 80 Spuren um, wenn auch hier zwei Fehlversuche auftreten, wird es noch ein letztes Mal wieder im 360-KB-Format

versucht und dann mit Fehlermeldung abgebrochen.

Das Programm wird mit MASM ab Version 4.0 (oder TASM) assembliert, gelinkt und mit EXE2BIN (bei TLINK von TASM reicht die Option /t) in eine COM-Datei verwandelt. Wir haben es INT13.ASM genannt. Die Aufrufe lauten dann:

```
MASM INT13;
LINK INT13;
EXE2BIN INT13.EXE          \
INT13.COM
```

Anschließend kann INT13.COM gestartet werden. Das Programm bedient in der Standardeinstellung immer nur ein Laufwerk, und zwar A:. Wenn das 1,2-MB-Laufwerk bei Ihnen als Laufwerk B: eingebaut ist, müssen Sie das Programm an den entsprechend kommentierten Stellen ändern.

```
1  ; residentes Programm für AT-Rechner, um 720-KB-Disketten
2  ; in MF-Laufwerken schreiben und lesen zu können
3
4  code          segment
5                assume cs:code
6                org      100h
7  start:
8                jmp      init
9
10 newint13:
11                cli          ;Interrupts sperren
12                cmp         ah, 0      ;Disk-Reset?
13                jne         next      ;Nein, dann weiterprüfen
14                mov         cs:reset, 1 ;Resetflag setzen
15                jmp         wpruef
16 next:
17                mov         cs:reset, 0 ;Resetflag löschen
18 wpruef:
19                cmp         ah, 3      ;Sektoren schreiben
20                jg          oldint13   ;Nein, dann alter Int 13h
21                cmp         ah, 2      ;Sektoren lesen
22                jb          oldint13   ;Nein
23                cmp         dl, 0      ;Laufwerk A
24                ;Für Laufwerk B cmp dl, 1 einfügen
25                jne         oldint13   ;Nein
26                cmp         ch, 0      ;Spur größer als Null
27                je          oldint13   ;Nein
28                sti          ;Interrupts zulassen
29                pushf
30                call        dword ptr cs:[int13]
31                ;alte Int 13h Routine aufrufen
32                jnc         ende        ;Kein Fehler aufgetreten
33                cmp         ah, 4      ;Sektor nicht gefunden
34                jne         weiter      ;Nein
35                dec         cs:zaehler ;decrement Fehlerzähler
36                cmp         cs:zaehler, 0 ;schon alle Versuche?
37                jne         weiter      ;Nein
38                mov         cs:zaehler, 2 ;Fehlerzähler neu setzen
39                cmp         cs:einstep, 1 ;schon Einzelstep?
40                je          umschalt    ;Ja
41                mov         cs:einstep, 1 ;Einzelstepflag
42                call        step        ;Umschalten auf Einzelstep
43                jmp         weiter
44 umschalt:
45                mov         cs:einstep, 0 ;Umschalten auf Doppelstep
46 weiter:
47                stc          ;Fehlerkennung setzen
48                jmp         ende
62                call        step        ;Einzelstep einschalten
63 schluss:
64                popf
65 ende:
```

```

66         push    ax
67         pushf
68         pop     ax             ;Flagregister nach AX laden
69         push    bp
70         mov     bp, sp
71         mov     ss:[bp+8], ax
72         ;Flagregister aktualisieren
73         pop     bp
74         pop     ax
75
76         iret
77
78 step     proc     near
79         push    ax
80         push    ds
81         mov     ax, 0040h
82         mov     ds, ax         ;Segmentregister
83         and byte ptr ds:[0090h], 11011111b
84 ;Für Laufwerk B and byte ptr ds:[0091h], 11011111b einsetzen
85         ;Umschalten auf Einzelstep
86         pop     ds
87         pop     ax
88         ret
89 step     endp
90
91 oldint13:
92         sti                     ;Interrupt wieder zulassen
93         pushf
94         call    dword ptr cs:[int13] ;Int 13h aufrufen
95         pushf                     ;Flags retten
96         cmp     dl, 0             ;nur Diskzugriffe auf A
97 ;Für Laufwerk B cmp dl, 1 einfügen
98         jne     schluss          ;kein Diskzugriff
99         cmp     cs:reset, 1       ;Resetflag gesetzt
100        jne     schluss          ;Nein
101        cmp     cs:einstep, 1     ;und Einzelstep
102        jne     schluss          ;Nein
103
104 int13     dd      ?              ;alter 13h Vektor
105 zaehler   db      ?              ;Fehlerzähler
106 einstep   db      ?              ;Einzelschrittflag
107 reset     db      ?              ;Resetflag
108
109 init:
110         mov     zaehler, 2        ;Fehlerzähler laden
111         mov     einstep, 0        ;kein Einzelstep
112         mov     reset, 0          ;kein Reset
113         mov     ah, 35h           ;lesen des alten 13h Vektors
114         mov     al, 13h
115         int     21h
116         mov     word ptr int13, bx
117         ;Offset des Vektors
118         mov     word ptr int13+2, es
119         ;Segment des Vektors
120         mov     dx, offset newint13 ;neuer 13h Vektor
121         mov     ah, 25h           ;Int 13h Vektor setzen
122         mov     al, 13h
123         int     21h
124         mov     dx, offset ausgabe
125         ;Zeichenkette ausgeben
126         mov     ah, 9
127         int     21h
128         mov     ax, 3100h
129         mov     dx, ((offset init-offset start)+10fh) shr 4
130         int     21h              ;Programm resident beenden
131
132 ausgabe db 13,10,'5,25-720 KB Treiber für 80286 und 386',13,10,'$'
133
134 code     ends

```

122

end

start

Kasten 8

Das Programm FORMAT.PAS (Turbo-Pascal 5.0) bildet die Oberfläche des Formatierprogrammes und greift auf die Programmteile der Unit FORMUNIT.PAS zurück.

```

1  program format;  {c't/Bührmann, 1/1990}
2
3  uses crt, formunit;
4
5  const
6      seitenzahl = 2;
7      {Anzahl der zu formatierenden Seiten}
8      {Bei Atari-Format 80 Spuren einseitig hier}
9      {seitenzahl=1 eintragen und 720 KB wählen}
10 bootsek  : array[1..108] of byte =
11  ($eb,$34,$90,$56,$42,32,$2d,32,$63,$27,$54,
12   0,2,0,1,0,2,0,0,0,0,0,0,0,0,0,0,seitenzahl,
13   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
14   0,0,0,0,0,0,$fa,$b8,$30,0,$8e,$d0,$bc,$fc,
15   0,$fb,$e,$1f,$bb,7,0,$be,$6c,$7c,$90,$8a,
16   4,$46,$3c,0,$74,8,$b4,$e,$56,$cd,$10,$5e,
17   $eb,$f1,$b4,1,$cd,$16,$74,6,$b4,0,$cd,$16,
18   $eb,$f4,$b4,0,$cd,$16,$33,$d2,$cd,$19);
19 boottext1 : array[1..39] of char =
20   #13#10'Diskette hat keine Systemdateien !'#13#10#10;
21
22   { !!!! Boottext2 und Boottext3 müssen jeweils in einer Zeile
23     !!!! stehen, damit der Compiler zufrieden ist }
24
25   boottext2 : array[1..83] of char =
26     'Um die Systemdateien zu übertragen muß die Diskette mit SYS A:
27     behandelt werden.'#13#10#10;
28   boottext3 : array[1..59] of char =
29     'Bitte Diskette wechseln und weiter mit beliebiger
30     aste.'#13#10#10;
31
32 type
33     sektor = array[1..512] of byte;
34     sekzeig = array[1..18] of ^sektor; {Max. 18 Sektoren Puffer}
35
36 var
37     param          : string[3]; {Kommandozeilenparameter}
38     taste          : char;
39     laufwerk       : string[1]; {Gewähltes Laufwerk bei Batch}
40     formart        : string[4]; {Gewähltes Format bei Batch}
41     verify         : string[1]; {Verify bei Batch}
42     artform        : integer;   {Format als Realzahl}
43     drive          : shortint;  {Formatlaufwerk}
44     code           : integer;
45     wahl           : shortint;  {Formatauswahl}
46     art            : byte;      {Art des Laufwerks (s. Formunit)}
47     verifyflag     : boolean;   {Verifizieren ?}
48     batch          : boolean;   {Aufruf mit Kommandozeile ?}
49     fparam         : boolean;   {Fehler bei Parametereingabe}
50     puffer         : sekzeig;   {Platz für max. 18 Sektoren}
51
52 {*****Unterprogramme*****}
53 procedure ausgang(sekzahl : byte);
54 var
55     sek : byte;
56 begin
57     laufwerkstabalt;
58     for sek:=1 to sekzahl do dispose(puffer[sek]);
59 end;
60 {*****}

```

```
59 procedure diskprep(sekzahl : byte; anzsek : word);
60 var
61     {Bootsektor, ROOT und FAT schreiben}
62     zaehler : byte;
63     anzahl  : byte;
64     i       : integer;
65     kopie   : integer;
66     fehler  : byte;
67 begin
68     anzahl:=formtab^[4]*2+round(formtab^[3]*32/512);
69     {Sektoren FAT und ROOT}
70     fillchar(puffer[1]^1, sekzahl*512, 0); {Puffer löschen}
71     for i:=1 to 108 do puffer[1]^i:=bootsek[i];
72     puffer[1]^14:=formtab^2; {Sektoren pro Cluster}
73     puffer[1]^18:=formtab^3;
74     puffer[1]^19:=0; {Anzahl der Einträge im Hauptverzeichnis}
75     puffer[1]^20:=lo(anzsek);
76     puffer[1]^21:=hi(anzsek); {Anzahl der Sektoren total}
77     puffer[1]^22:=formtab^1; {Mediabyte}
78     puffer[1]^23:=formtab^4;
79     puffer[1]^24:=0; {Sektoren pro FAT}
80     puffer[1]^25:=sekzahl; {Sektoren pro Seite einer Spur}
81     for i:=1 to 39 do puffer[1]^i+108:=ord(boottext1[i]);
82     for i:=1 to 83 do puffer[1]^i+147:=ord(boottext2[i]);
83     for i:=1 to 59 do puffer[1]^i+230:=ord(boottext3[i]);
84     puffer[1]^511:=$55;
85     puffer[1]^512:=$aa; {Ende des Bootsektors}
86     fehler:=readwriteverify(3, 0, 0, 1, 1, drive, puffer[1]^);
87     for i:=1 to 512 do puffer[1]^i:=0;
88     {Bootsektor in Puffer löschen}
89     puffer[1]^1:=formtab^1; {Mediabyte}
90     puffer[1]^2:=$ff;
91     puffer[1]^3:=$ff;
92     kopie:=formtab^4+1;
93     puffer[kopie]^1:=formtab^1;
94     puffer[kopie]^2:=$ff;
95     puffer[kopie]^3:=$ff;
96     fehler:=readwriteverify(3, 0, 0, 2, sekzahl-1, drive,
97                             puffer[1]^);
98     dec(anzahl,sekzahl-1);
99     fillchar(puffer[1]^1, anzahl*512, 0); {Puffer löschen}
100    fehler:=readwriteverify(3, 0, 1, 1, anzahl, drive,
101                            puffer[1]^);
102 end;
103 {*****}
104 procedure diskformat(wahl, spurzahl, sekzahl : byte);
105     {Diskette formatieren}
106 var
107     spur      : byte;    {Aktuelle Spur}
108     sek       : byte;    {Aktueller Sektorpuffer}
109     seite     : byte;    {Aktuelle Seite}
110     anzsektor : word;    {Sektoren auf Disk}
111     taste     : char;    {evt. Taste betätigt}
112     fehler    : byte;    {Fehler bei Diskoperationen}
113
114 begin
115     if not batch then
116     begin
117         writeln(spurzahl*sekzahl,' KByte Format'); writeln;
118         write('Bitte Diskette in Laufwerk ');
119         highvideo;
120         if drive=0 then write('A') else write('B');
121         lowvideo;
122         writeln(' einlegen, weiter mit Eingabetaste');
123         repeat
124             taste:=readkey;
125             until (taste=#13) or (taste=#27);
126             if taste=#27 then exit;
127         end;
```

```
128   for sek:=1 to sekzahl do new(puffer[sek]);
129   schreibrate(art, wahl, drive);
130   {Schreibrate für Format setzen}
131   laufwerkstabneu;           {DPB für Format setzen}
132   anzsektor:=spurzahl*sekzahl*seitenzahl;
133   spur:=0;
134   repeat
135     for seite:=0 to (seitenzahl-1) do
136       begin
137         if not batch then gotoxy(1,14);
138         gotoxy(1,wherey);
139         write('Kopf : ',seite,'   Zylinder : ',spur);
140         fehler:=spurformat(spur, seite, 1, sekzahl, drive);
141         if fehler=3 then
142           begin
143             if not batch then gotoxy(1,13);
144             writeln; write('Diskette ist schreibgeschützt !');
145             taste:=readkey;
146             ausgang(sekzahl);
147             exit;
148           end;
149           if verifyflag then
150             fehler:=readwriteverify(4, spur, seite, 1,
151                                     sekzahl, drive, puffer[1]^);
152           if fehler<>0 then
153             begin
154               if not batch then gotoxy(1,13);
155               writeln;
156               write('Diskette für dieses Format unbrauchbar !');
157               if not batch then taste:=readkey;
158               ausgang(sekzahl);
159               if not batch then exit
160               else halt(1); {Programm mit Fehlermeldung abbrechen}
161             end;
162             if keypressed then taste:=readkey;
163           end;
164           inc(spur);   {Nächste Spur}
165         until (taste=#27) or (spur>=spurzahl);
166         diskprep(sekzahl, anzsektor);
167         ausgang(sekzahl);
168         if taste<>#27 then
169           begin
170             if not batch then
171               begin
172                 gotoxy(1,14);
173                 write('Diskette wurde fehlerfrei formatiert, ');
174                 write('weiter mit beliebiger Taste');
175                 taste:=readkey;
176               end;
177             end;
178         end;
179   {*****Hauptprogramm*****}
180   begin
181     checkbreak:=false;
182     {kein Abbruch des Programms mit CTRL-BREAK}
183     drive:=0; verifyflag:=true; {Laufwerk A, Verify : ON}
184     art:=laufwerka; {Art des Laufwerkes s. Unit Formunit}
185     fparam:=false;
186     if paramcount=0 then
187       batch:=false {Keine Kommandozeile}
188     else
189       batch:=true; {Mit Kommandozeile aufgerufen}
190     if batch then
191       begin
192         laufwerk:=paramstr(1); {Laufwerk holen}
193         drive:=ord(uppercase(laufwerk[1]))-65;
194         {0 oder 1 für A oder B}
195         if (drive>1) or (drive<0) then
196           begin
```

```
197     batch:=false;  {Batchbetrieb abbrechen}
198     fparam:=true;  {Falsche Parameter}
199     drive:=0;      {Wieder Laufwerk A}
200 end;
201 if drive=0 then art:=laufwerka else art:=laufwerkb;
202 format:=paramstr(2);  {Gewählte Formatkapazität}
203 val(format, artform, code);
204 if (code<>0) or (artform<>360) and (artform<>720) and
205     (artform<>1200) and (artform<>1440) then
206 begin
207     batch:=false;
208     fparam:=true;
209 end;
210 verify:=paramstr(3); {Verify OFF bei Wert in Kommandozeile}
211 if verify='' then verifyflag:=true else verifyflag:=false;
212 end;
213 case artform of
214     360 : wahl:=1;
215     720 : wahl:=2;
216     1200 : wahl:=3;
217     1440 : wahl:=4;
218 end;
219 taste:=chr(wahl+48);
220 repeat
221     if not batch then
222     begin
223         clrscr;
224         gotoxy(50,1); write('(L)aufwerk : ');
225         if drive=0 then
226         begin
227             write('A'); art:=laufwerka;
228         end
229         else
230         begin
231             write('B'); art:=laufwerkb;
232         end;
233         gotoxy(50,2); write('(V)erify   : ');
234         if verifyflag then write('ON ') else write('OFF');
235         if fparam then {Fehler bei Parametereingabe}
236         begin
237             gotoxy(1,20);
238             writeln('Fehler bei der Parametereingabe !');
239             write('Korrektur Aufruf mit ');
240             writeln('"Format Laufwerk Formatwahl [n]"');
241             write('Laufwerk A oder B, ');
242             writeln('Formatwahl 360, 720, 1200 oder 1440');
243             writeln('Bei Eingabe von n kein Verify');
244         end;
245         gotoxy(1,1);
246         writeln('   Formatauswahl:');writeln;
247         writeln('1.  360  KB auf 5,25-Laufwerk');
248         writeln('2.  720  KB auf 5,25- und 3,5-Laufwerk');
249         writeln('3.  1,2  MB auf 5,25-Laufwerk');
250         writeln('4.  1,44 MB auf 3,5-Laufwerk');writeln;
251         writeln('Bitte gewünschte Nummer eingeben ? ');writeln;
252         taste:=readkey;
253         taste:=upcase(taste);
254         wahl:=ord(taste)-48; {Wahl des Diskettenformates}
255     end;
256     case taste of
257         '1' : diskformat(wahl, 40, 9);
258             {Format, Spurzahl, Sekzahl, 360 KB}
259         '2' : diskformat(wahl, 80, 9);  {720 KB}
260         '3' : diskformat(wahl, 80, 15); {1,2 MB}
261         '4' : diskformat(wahl, 80, 18); {1,44 MB}
262         'V' : if verifyflag then
263                 verifyflag:=false else verifyflag:=true;
264         'L' : if (drive=0) and (laufwerkb<>0) then
265                 drive:=1 else drive:=0;
```

```

266     end;
267     until (taste=#27) or batch;
268     clrscr;
269     {Programmende durch ESC oder Batch}
270 end.

```

Kasten 9

Das Programm FORMAT.PAS (Turbo-Pascal 5.0) bildet die Oberfläche des Formatierprogrammes und greift auf die Programmteile der Unit FORMUNIT.PAS zurück.

```

1  unit Formunit;    {Formatiererroutinen für Standardformate}
2
3  interface
4
5  const            {Laufwerkstabellen für Standardformate}
6      tab36 : array[1..11] of byte
7          = ($df,$02,$25,$02,$09,$2a,$ff,$50,$f6,$0f,$08);
8      tab12 : array[1..11] of byte
9          = ($df,$02,$25,$02,$0f,$1b,$ff,$54,$f6,$0f,$08);
10     tab14 : array[1..11] of byte
11         = ($af,$02,$25,$02,$12,$1b,$ff,$6c,$f6,$0f,$08);
12     form36 : array[1..4] of byte
13         = ($fd,2,$70,2);
14         {Mediabyte, Sektoren pro Cluster}
15         {Einträge Hauptverzeichnis, Sektoren pro FAT}
16     form72 : array[1..4] of byte
17         = ($f9,2,$70,3);
18     form12 : array[1..4] of byte
19         = ($f9,1,$e0,7);
20     form14 : array[1..4] of byte
21         = ($f0,1,$e0,9);
22     versuche = 5;      {Zahl der Versuche bei Fehlern}
23
24 type
25     tabelle = array[1..4] of byte;
26
27 var
28     laufwerka, laufwerkb : byte;    {Art der Laufwerke}
29     tabalt, tabneu       : pointer; {Zeiger auf Laufwerkstabelle}
30     formtab             : ^tabelle; {Zeiger auf Diskformate}
31     einzelschritt       : boolean;  {Einzelstep}
32
33 procedure diskreset;                {Reset bei Fehlern}
34 function config(drive : byte) : byte; {Laufwerkskonfiguration}
35 procedure einzelstep;              {720 KB in 5.25 MF-Laufwerk}
36 procedure schreibrate(art, kap, drive : byte);
37     {Schreibrate wählen}
38     {Procedure Schreibrate immer vor Laufwerkstabneu aufrufen !}
39 procedure laufwerkstabneu;         {neuer DPB}
40 procedure laufwerkstabalt;
41 function readwriteverify(was, spur, seite, sektor,
42     anzahl, drive : byte;
43     var buffer) : byte;
44 function spurformat(spur, seite, sektor,
45     anzahl, drive : byte) : byte;
46
47 implementation
48
49 uses dos;
50
51 {*****}
52 procedure diskreset;
53 var
54     cpu      : registers;
55     zaehler  : byte;
56

```

```

57 begin
58   zaehler:=versuche;
59   repeat
60     cpu.ah:=0; {Diskreset}
61     cpu.dl:=0; {Reset für alle Laufwerke}
62     intr($13, cpu);
63     dec(zaehler,1);
64     until (zaehler=0) or (cpu.flags and fcarry=0) or (cpu.ah=0);
65     if einzelschritt then einzelstep; {Wichtig}
66     {Nach jedem Reset umschalten auf Einzelstep !}
67   end;
68   {*****}
69   function config(drive : byte) : byte;
70   var
71     cpu      : registers;
72     zaehler  : byte;
73     { 0 : Kein Laufwerk}
74     { 1 : 360 KB}
75     { 2 : 1.2 MB}
76     { 3 : 720 KB}
77     { 4 : 1.44 MB}
78   begin
79     zaehler:=versuche;
80     repeat
81       cpu.ah:=8; {Feststellen des Laufwerkstyps}
82       cpu.dl:=drive; {Nummer des Laufwerks}
83       intr($13, cpu);
84       dec(zaehler,1);
85       until (zaehler=0) or (cpu.flags and fcarry=0) or (cpu.ah=0);
86       if zaehler=0 then
87         begin {Funktion nicht vorhanden, dann XT oder ähnlich}
88           case drive of
89             0: begin
90               intr($11, cpu); {Feststellung der Konfiguration}
91               config:=(cpu.al and $01) ; {1 = Drive A da}
92             end; {Equipmentword IPL-Bit, überhaupt Diskette da?}
93             1: begin
94               intr($11, cpu); {Feststellung der Konfiguration}
95               zaehler:=(cpu.al and $C0) shr 6;
96               if zaehler=0 then {kein Laufwerk B}
97                 config:=0
98               else
99                 config:=1 {Laufwerk B da}
100             end; {Equipmentword Drive-Zähler, 00=1 Drive, 01=2}
101             else
102               config:=0; {alle anderen Anforderungen abweisen}
103             end;
104           end;
105           config:=cpu.bl; {Art des Laufwerks}
106         end;
107       {*****}
108     procedure einzelstep; {720 KB in 5.25 MF-Laufwerk}
109     var
110       inhalt : byte;
111     begin
112       inhalt:=mem[$0040:$0090]; {Controllerbyte}
113       inhalt:=inhalt and $DF;
114       mem[$0040:$0090]:=inhalt; {Einzelstep für Laufwerk A}
115       inhalt:=mem[$0040:$0091];
116       inhalt:=inhalt and $DF;
117       mem[$0040:$0091]:=inhalt; {Einzelstep für Laufwerk B}
118     end;
119     {*****}
120     procedure schreibrate(art, kap, drive : byte);
121     var
122       cpu      : registers;
123       form, zaehler : byte;
124       ax       : word;
125     begin

```



```
126   if art=1 then
127   begin
128       form:=1; {360 KB Laufwerk}
129       tabneu:=@tab36; {360 KB Laufwerkstabelle}
130       formtab:=@form36; {360 KB Diskettenformat}
131       cpu.ax:=$1828; cpu.cx:=$2709; {Parameter für 18h}
132   end
133   else
134   if art=2 then {1.2 MB MF-Laufwerk}
135       if kap<=2 then
136       begin
137           form:=2;
138           tabneu:=@tab36; {360 KB oder 720 KB in 5,25 MF-Drive}
139           cpu.ax:=$1828; cpu.cx:=$2709;
140           if kap=1 then formtab:=@form36 else formtab:=@form72;
141       end
142       else
143       begin
144           form:=3;
145           tabneu:=@tab12; {1.2 MB in 1.2 MB-Laufwerk}
146           formtab:=@form12;
147           cpu.ax:=$1850; cpu.cx:=$4f0f;
148       end
149       else
150   if art>=3 then {720 KB oder 1.44 MB Laufwerk}
151       if kap<=2 then
152       begin
153           form:=4;
154           tabneu:=@tab36;
155           formtab:=@form72; {720 KB}
156           cpu.ax:=$1850; cpu.cx:=$4f09;
157       end
158       else
159       begin
160           form:=3;
161           tabneu:=@tab14;
162           formtab:=@form14; {1.44 MB}
163           cpu.ax:=$1850; cpu.cx:=$4f12;
164       end;
165   zaehler:=versuche;
166   ax:=cpu.ax; {AX-Register wird durch Funktion verändert}
167   repeat
168       cpu.ax:=ax;
169       cpu.dl:=drive; {Nummer des Laufwerks}
170       intr($13, cpu); {Funktion Nr. 18h aufrufen}
171       if cpu.flags and fcarry=1 then
172       begin
173           diskreset; {Fehler aufgetreten}
174           dec(zaehler);
175       end;
176   until (cpu.flags and fcarry=0) or (zaehler=0) or (cpu.ah=0);
177   if (zaehler=0) then {Funktion 18h nicht vorhanden}
178   begin
179       zaehler:=versuche; {letzte Rettung}
180       repeat {Funktion Nr. 17h, falls 18h nicht vorhanden}
181           cpu.ah:=$17; {Schreibrate setzen}
182           cpu.al:=form;
183           cpu.dl:=drive; {Welches Laufwerk formatieren}
184           intr($13, cpu);
185           if cpu.flags and fcarry=1 then
186           begin
187               diskreset; {Fehler aufgetreten}
188               dec(zaehler);
189           end;
190       until (cpu.flags and fcarry=0) or (zaehler=0) or (cpu.ah=0);
191   end;
192   if (art=2) and (kap=2) then
193   begin {720 KB in 5,25}
194       einzelstep;
```

```
195     einzelschritt:=true; {Merker für Einzelstep eingeschaltet}
196   end
197   else      {anderes Format gewählt}
198     einzelschritt:=false; {Kein Einzelschritt eingeschaltet}
199   end;
200   {*****}
201   procedure laufwerkstabneu;
202
203   begin
204     getintvec($1e,tabalt);
205     setintvec($1e,tabneu);
206   end;
207   {*****}
208   procedure laufwerkstabalt;
209
210   begin
211     setintvec($1e,tabalt);
212   end;
213   {*****}
214   function readwriteverify(was, spur, seite, sektor,
215                             anzahl, drive : byte;
216                             var buffer) : byte;
217   var
218     cpu      : registers;
219     zaehler  : byte;
220
221   begin
222     {was=2 : Sektoren von Diskette lesen}
223     {was=3 : Sektoren auf Diskette schreiben}
224     {was=4 : Sektoren verifizieren}
225     repeat
226       cpu.ah:=was;
227       cpu.dl:=drive;
228       cpu.dh:=seite;
229       cpu.ch:=spur;
230       cpu.cl:=sektor;
231       cpu.al:=anzahl;
232       cpu.es:=seg(buffer);
233       cpu.bx:=ofs(buffer);
234       intr($13, cpu);
235       readwriteverify:=cpu.ah; {Rückgabe des Fehlercodes}
236       if cpu.flags and fcarry=1 then
237         begin
238           readwriteverify:=cpu.ah; {Fehlercode}
239           diskreset;
240         end;
241       dec(zaehler);
242     until (zaehler=0) or (cpu.flags and fcarry=0) or (cpu.ah=0);
243   end;
244   {*****}
245   function spurformat(spur, seite, sektor,
246                       anzahl, drive : byte) : byte;
247   type
248     formrec = record
249       trackdisk, seitedisk, sektordisk, zahlbyte : byte;
250     end;
251   var
252     cpu      : registers;
253     formattab : array[1..18] of formrec; {Max. 18 Sektoren}
254     zaehler   : byte;
255     sekzahl   : byte;
256
257   begin
258     zaehler:=versuche;
259     for sekzahl:=1 to anzahl do
260       begin
261         formattab[sekzahl].trackdisk:=spur;
262         formattab[sekzahl].seitedisk:=seite;
263         formattab[sekzahl].sektordisk:=sektor;
264         formattab[sekzahl].zahlbyte:=2;
265       end;
266     end;
```

```
264     repeat
265         cpu.ah:=5;          {Spur formatieren}
266         cpu.dl:=drive;
267         cpu.dh:=seite;
268         cpu.ch:=spur;
269         cpu.al:=anzahl;
270         cpu.es:=seg(formattab);
271         cpu.bx:=ofs(formattab);
272         intr($13, cpu);
273         spurformat:=cpu.ah; {Rückgabe des Fehlercodes}
274         if cpu.flags and fcarry=1 then
275             begin
276                 spurformat:=cpu.ah;
277                 diskreset;
278             end;
279         dec(zaehler);
280     until (zaehler=0) or (cpu.flags and fcarry=0) or (cpu.ah=0);
281 end;
282 {*****Hauptprogramm der Unit*****}
283 begin
284     laufwerka:=config(0);
285     laufwerkb:=config(1); {Art von Laufwerk A und B}
286 end.
```

Zu diesem Artikel existieren Programmbeispiele

[0290_202.doc](#)

[0290_202.zip](#)