



Diskcopy ohne Diskjockey

Diskettenkopien und -vergleiche schnell und effizient

Volker Bührmann

Fast jedes Software-Paket begrüßt Sie im Handbuch mit der Aufforderung, sofort mittels DISKCOPY Arbeitskopien von den Lieferdisketten anzufertigen und die Originale sicher in einen Schrank einzuschließen. Das Kopieren ist so lange nicht weiter tragisch, wie es sich um 360-KB-Scheiben handelt, die sich auch mit nur *einem Laufwerk meist in einem Rutsch kopieren lassen. Bei 720ern wird das aber schon mühsam, weil Ziel- und Quell-Diskette mindestens zweimal eingelegt werden müssen.*

[Unterthema: COMPARE.ASM](#)

[Unterthema: Media-Byte-Zuordnung von DCC](#)

[Unterthema: Allgemeine Kommando-Syntax:](#)

[Unterthema: DMA Boundary Crossing](#)

[Unterthema: DCC.PAS](#)

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

Zur Strapaze werden 1:1-Kopien von 1,44-MB-Scheiben. Fatal, wenn man durch eine Ablenkung einmal Ziel- und Quelldisk verwechselt und die Quelldisk nicht schreibgeschützt war - mit ein bißchen Pech hat man dann eine 'leere Stelle' in die Master-Diskette geschrieben. Etliches, was für DISKCOPY gilt, trifft in fast gleicher Weise auch für DISKCOMP zum Vergleichen zweier Disketten zu.

Unser hier vorgestelltes Programm DCC ersetzt DISKCOPY und DISKCOMP von DOS und bietet noch etliches mehr an Verbesserungen als bisher angeführt.

Wie's DOS macht

DISKCOPY liest nacheinander alle Sektoren einzeln von der Quelldiskette und speichert sie im DOS-RAM. Anschließend werden sie exakt in der gleichen Anordnung auf die Zielscheibe geschrieben. Dazu ist es notwendig, daß Ziel- und Quelldiskette identisches Format haben. Das leuchtet ein, wenn man weiß, daß jedes DOS-Format einen anderen logischen Aufbau hat. Zwar 'paßt' eine 720-KB-Disk ohne weiteres auf eine 1,2-MB-Scheibe, aber würde man Boot-Sektor, FAT und Directory einer 720-KB- auf eine 1,2-MB-Diskette schreiben (was so einfach nicht ist), dann würde DOS später die Disk auch nur als 720-KB-Disk interpretieren.

'Logische Umformatierungen' sind auch deswegen einigermaßen schwierig (aber nicht unmöglich), weil dabei zum Beispiel verschiedene Allokierungsmechanismen berücksichtigt werden müssen: DOS belegt den Speicherplatz bei 1,2-MB-Scheiben in Portionen von 512 Bytes (1 Sektor), bei 720-KB-Disketten hingegen in 1024-Byte-Blöcken. Das heißt, daß jeder FAT-Eintrag in einem Fall also zur Verwaltung von 512-Byte-, im anderen von 1024-Byte-Blöcken dient. Und außerdem müßten die absolute Lage und Größe von FAT, Directory und Datenbereich jeweils dem Zielformat angepaßt werden.

Bei einigen MSDOS-Varianten (bei PCDOS unseres Wissens nicht) prüft DISKCOPY nach, ob Quell- und Zieldiskette in einem gültigen DOS-Format angelegt sind. Das kann sehr hinderlich sein, wenn man

etwa Sicherungskopien von CP/M-86- oder Minix-Disketten mittels DOS ziehen muß. An sich ist diese Prüfung unnötig (wie PCDOS beweist): solange der Controller-Chip die physikalischen Sektoren einzeln von der Quelldiskette lesen kann, solange kann man sie auch bedenkenlos in genau gleicher Weise auf die Zieldiskette schreiben. Wenn's nicht paßt, kann zu gegebener Zeit immer noch eine Fehlermeldung erfolgen.

Um einen vollwertigen (und verbesserten Ersatz) für DISKCOPY zu erstellen, fehlt noch ein letzter Punkt: DISKCOPY kann auch automatisch formatieren. Wenn sich die Zieldisk nicht fehlerfrei im Format der Quelldisk beschreiben läßt, schreitet DISKCOPY automatisch (ohne Eingriffsmöglichkeit, versteht sich) zum Formatieren der Zielscheibe.

Besser machen

Grundidee unseres neuen DISKCOPY-Programms DCC (DCC für Diskcopy und Diskcompare) ist die Speicherung der gelesenen Sektordaten der Quelldiskette als Datei auf einer Festplatte oder RAM-Disk. Es kann eigentlich jeder beliebige Massenspeicher verwendet werden - er muß nur eine freie Speicherkapazität von mindestens der Größe der eingelesenen Diskette haben. Wenn nichts dergleichen vorhanden ist, dann nützt also unser Programm herzlich wenig - nicht nur das, es läuft dann auch nicht.

Die Sektordaten werden als Datei auf dem jeweiligen Datenträger (Festplatte, RAM-Disk) angelegt, wobei man den Namen frei wählen kann. So kann man etliche Diskettenkopien auf der Platte halten und bei Bedarf ohne Neulesen der Master-Disketten vervielfältigen.

Da DCC ebenfalls Disketten vergleichen kann, lassen sich auch spätere Vergleiche von fragwürdigen Disketten anhand solcher Dateien, also ohne Zuhilfenahme der Ursprungsdiskette, durchführen. Weiterhin ist das Programm auch für den Batch-Betrieb ausgelegt: sofern kein ernsthafter Fehler auftritt, erwartet das Programm ein Minimum an Benutzereingaben.

Externes

Das hier vorgestellte Diskettenkopierprogramm ist in Turbo-Pascal 5.0 geschrieben und setzt sich aus folgenden Teilen zusammen:

- DCC: Hauptprogramm
- FORMUNIT.PAS: Unit zum Hauptprogramm
- COMPARE.ASM: Hilfsroutine für Diskettenvergleich

FORMUNIT.PAS wurde in c't 2/90 veröffentlicht [1] und enthält die Basis-Routinen zum sektorweisen Lesen, Schreiben, Verifizieren und Formatieren von Disketten. Nebenbei werden darüber auch die Floppy-Laufwerke auf Existenz und die mit ihnen möglichen Formate untersucht.

COMPARE.ASM ist eine externe Assembler-Routine, die mit einem Assembler (TASM oder MASM) in ein Objekt-File übersetzt werden muß (Endung .OBJ). Erst wenn FORMUNIT.PAS und COMPARE.OBJ verfügbar sind, kann man das Hauptprogramm mit dem Compiler übersetzen.

Ersatzweise kann man die knapp 200 Bytes der Objekt-Datei auch mittels DEBUG hexadezimal eintippen. Der zusätzlich zum Assemblerlisting abgedruckte Hexdump stellt eine komplette DEBUG-Steuerdatei dar. Sie können dieses Listing mit einem Editor als Textdatei (die Sie zum Beispiel DUMP.TXT nennen) eingeben und anschließend mit folgendem DEBUG-Aufruf automatisch in die benötigte Form bringen:

```
DEBUG < DUMP.TXT
```

Achten Sie dabei aber peinlich genau darauf, alles buchstabengenau und mit exakt derselben Anzahl

Leerzeilen einzugeben (auch auf das `q` am Ende muß noch eine Leerzeile folgen). Schauen Sie sich die automatisch erzeugte Datei COMPARE.OBJ mit Hilfe von DEBUG anschließend noch mal genau an. Die Datei bekommt zwar in jedem Fall die richtige Länge von 184 Bytes, bei Fehlern in der Textdatei könnte aber ein Teil der Daten aus zufälligen Mustern bestehen, die gerade im RAM standen.

Möglichkeiten

Wird DCC ohne oder mit falschen Parametern (für Batch-Betrieb) aufgerufen, meldet es sich mit einem einfachen Menü. Man hat hier als erstes die Möglichkeit, Quell- und Ziellaufwerk anzugeben, wobei nur A: und B: zulässig sind. Floppies, die per Treiber (etwa DRIVER.SYS) über andere Laufwerksbuchstaben angesprochen werden müssen, lassen sich nicht erreichen. Unser Programm geht direkt ans BIOS, und das kennt gar keine logischen Laufwerke A: und B:, sondern nur die physikalischen Einheiten 0 und 1. Mit logischen Laufwerksbuchstaben arbeitet nur das DOS; wir setzen A: und B: gleich Laufwerk 0 und 1, damit die Bedienung vertraut bleibt.

Die Option (F)ormat ist beim Programmstart abgeschaltet (OFF). Man sollte das auch nicht ändern, denn bei Format ON wird die Zieldisk in jedem Fall formatiert, auch wenn sie im korrekten Format vorliegt - was das Kopieren unnötig verzögert. Wenn der Boot-Sektor der Zieldisk (zur Formatermittlung) bei `Format OFF` nicht gelesen werden kann oder ein unpassendes Format erkannt wird, fragt das Programm sowieso nach, ob es die Zieldiskette nicht doch formatieren soll. Das bietet Ihnen unter anderm auch `eine letzte Chance`, noch mal zu überlegen, ob überhaupt die richtige Zieldisk im Laufwerk steckt.

Mit einem Druck auf die ESC-Taste können Sie dann abbrechen. ESC bewirkt in allen Lagen (außer während der manuellen Parametereingabe) einen sofortigen Abbruch, auch wird das Programm mit ESC beendet.

Wenn ein `unverhofftes` Formatieren der Zieldiskette stattfindet, also das Programm dies bei `Format OFF` vorschlägt und Sie dem zustimmen, dann springt die Anzeige auf `Format ON` um, ebenso wird auch automatisch Verify auf ON gesetzt und angezeigt. Die eigenständige (V)erify-Option im Menü greift nur dann, wenn Sie bereits vor dem Start des Kopiervorganges Format explizit auf ON setzen, aber Verify auf OFF lassen. Formatieren ohne Verify geht etwas schneller und ist beim Diskcopy nicht so riskant wie beim puren Formatieren, denn schlecht formatierte Sektoren werden (in aller Regel) sofort beim anschließenden Beschreiben mit Daten vom BIOS gemeldet.

Im Menüpunkt `Datei-(N)ame` läßt sich der Name der Image-Datei als vollständiger Pfad angeben. Beim Programmstart wird eine Datei im aktuellen Directory mit Namen DCC.BAC voreingestellt und angezeigt.

Des weiteren läßt sich die eigentliche Funktion - Kopieren oder Vergleichen - mit `(D)iskcopy` oder `(C)ompare Disketten` auswählen. DCC zeigt die jeweils aktive Auswahl an, indem die entsprechende Menüzeile heller (highlighted) dargestellt wird.

Auf die letzte Option `(M)anuelle Parametereingabe` gehen wir etwas später gesondert ein. Jede der Optionen wird durch den geklammerten Buchstaben aktiviert.

Umgang

Ist nach der Wahl von `Diskcopy` oder `Compare` noch kein Diskcopy-File mit dem gleichen Namen wie im Hauptmenü eingestellt vorhanden, fordert das Programm automatisch die Quelldiskette an. Bei vorhandenem File werden die darin abgespeicherten physikalischen Disketteninformationen (Spur-, Seiten- und Sektorzahlen) angezeigt, und der Benutzer kann wählen, ob die Datei mit dem Inhalt einer Quelldiskette überschrieben werden soll oder ob der Inhalt der Datei direkt auf die Zieldiskette zu kopieren ist.

Auch nach dem Einlesen einer Quelldiskette wird deren Format anhand der Boot-Sektor-Daten angezeigt.

Ist eine Datei vorhanden (spätestens nach Einlesen der Quelldiskette), erfolgt die Aufforderung, die Zieldiskette einzulegen. Bei 'Format OFF' prüft DCC zunächst das Format der Zieldisk auf Übereinstimmung mit dem Quellformat. Sind die Formate ungleich oder treten Lesefehler auf, so wird das Zieldiskettenformat (soweit möglich) zur Kontrolle ausgegeben. Der Benutzer muß dann entscheiden, ob formatiert werden soll oder nicht.

Tritt allerdings zu einem späteren Zeitpunkt beim Schreiben ein Fehler auf, so wird ohne Rückfrage die Zieldiskette formatiert und verifiziert (die Anzeige springt dann automatisch um). Nach einem Kopiervorgang existiert in jedem Fall eine Image-Datei der Quelldiskette, und DCC bietet an, eine weitere Kopie von dieser zu erstellen. Der Vergleichsvorgang (Compare) verläuft analog, Verify- und Format-Option spielen dann natürlich keine Rolle.

Handarbeit

Läßt sich das Format einer Quelldiskette nicht ermitteln, fordert DCC zur manuellen Parametereingabe auf (das kann natürlich auch passieren, wenn die Quell-Disk versehentlich unformatiert ist). Gedacht ist diese Möglichkeit für Disketten, deren Boot-Sektor keine irgendwie DOS-konformen Formatinformationen enthält. Dabei beschränkt sich DCC nicht nur auf die Auswertung des Boot-Sektors, sondern versucht bei einem Fehlschlag anhand des Media-Bytes zu Beginn jeder FAT herauszufinden, ob nicht doch ein gültiges DOS-Format erkennbar ist.

Die Tabelle zeigt die Format-Annahmen, die DCC aufgrund des Media-Bytes trifft. Es sei nachdrücklich darauf hingewiesen, daß das Media-Byte F9h für alle möglichen 80-Spur-Formate der unterschiedlichsten Hersteller steht und selbst unter DOS doppelt verwendet wird: zum einen für 720-KB-, zum andern für 1,2-MB-Scheiben. DCC entscheidet sich für das 1,2-MB-Format, denn bei Disketten im wesentlich neueren 720-KB-Format (ab DOS 3.2, bei Laptops auch 3.1) kann man immer davon ausgehen, daß eine vollständige Format-Beschreibung im Boot-Sektor steht.

Die Option 'Manuelle Parametereingabe' können Sie aber auch bereits vor dem Start von Kopier- oder Vergleichsoperationen benutzen. Sie ist immer dann angebracht, wenn man eine Diskette in einem 'Fremdformat', dessen Aufbau man allerdings kennen muß, bearbeiten möchte. Variieren lassen sich dabei Anzahl der Sektoren pro Spur, Anzahl der Seiten und Spuren, Double/Single Stepping und Schreibdichte. Minix-User können damit zum Beispiel ihre Master-Disketten unter DOS duplizieren, wenn DISKCOPY nicht mitspielt.

Ganz wichtig: CP/M-Disketten etwa, deren Format Sektorgrößen ungleich 512 Byte aufweist, kann das Programm ohne größere Änderungen (auch in FORMUNIT.PAS) nicht verarbeiten. Dazu müßte man zum Beispiel in der Funktion 'Spurformat' einen variablen Parameter zur Festlegung der Sektorgröße übergeben, ebenso benötigt man angepaßte DDPTs (Genaueres dazu in [1]). Kleiner Tip am Rande: unter CP/M ist es nicht ungewöhnlich, daß der Startsektor in der Spur größer als 1 ist (etwa bei den Schneider CPCs) oder mit Interleave (heißt bei Disketten auch Skew) ungleich 1:1 gearbeitet wird.

Die CP/M-Formate für die PC-Welt (CP/M-86) basieren physikalisch auf DOS-kompatiblen Formaten (8 Sektoren zu 512 Byte, ein- oder zweiseitig) und lassen sich daher ohne Änderungen von DCC kopieren. Wenn Sie die Parameter per Hand einstellen, findet ganz bewußt keine Plausibilitätsprüfung der eingegebenen Daten statt. Auch so unglaubliche Formatangaben wie 120 Spuren und fünf Seiten werden akzeptiert.

Automatisieren

DCC läßt sich auch von der Kommandozeile beziehungsweise innerhalb von Batch-Dateien starten. Beim Batching sind so wenig Rückfragen wie möglich erbeten, damit man überhaupt zu einer sinnvollen Abkürzung durch automatische Verarbeitung gelangen kann. Deshalb verändern sich die 'Spielregeln' gegenüber dem menügeführten Betrieb.

Zunächst entfällt die manuelle Parametereingabe im Batch-Betrieb. Des weiteren wird die Zieldiskette grundsätzlich und ohne Rückfrage formatiert (immer mit Verify), wenn ihr Format nicht zur Quelldisk oder der angegebenen Image-Datei paßt.

Außerdem ist die Funktion von DCC quasi 'halbiert' worden: Mit einem Aufruf von DCC kann man entweder einen Disketteninhalt lesen und auf Platte schreiben *oder* eine bereits vorhandene Image-Datei auf eine Zieldiskette schreiben beziehungsweise die Datei mit einer Zieldiskette vergleichen. Eine vollständige Kopie (ebenso ein Vergleich) erfordert also zwei Kommandozeilenaufrufe. Das mag umständlich scheinen, ermöglicht aber einen sehr differenzierten und damit vielseitigeren Gebrauch beim Batching.

Eine vollständige Kommandozeile zum Aufruf von DCC sieht folgendermaßen aus:

```
DCC r/w q/z dat p/n [d/c]
```

Fehlen Parameter oder sind sie fehlerhaft eingegeben, verzweigt das Programm ins Menü und zeigt zusätzlich die Kommandozeilen-Syntax an.

Parameter r/w bestimmt, ob eine Diskette gelesen (R wie Read) oder geschrieben (W wie Write) werden soll. Demgemäß wird eine Laufwerksangabe für Parameter q/z (A: oder B:) entweder als Quell- oder als Ziel-Drive interpretiert. Für dat ist der Name der Image-Datei (kompletter Pfad erlaubt) anzuführen.

Mit p/n (P für Prompting, N für No Prompting) entscheiden Sie, ob DCC das Einlegen von Disketten explizit anfordert und auf einen bestätigenden Tastendruck wartet oder sofort loslegt. Wenn Sie N angeben und noch keine Disk eingelegt haben, setzt es natürlich eine Fehlermeldung,

Wenn mit dem ersten Parameter R nur Lesen der Quelldisk gefordert ist, kann die Angabe von d/c entfallen. Wenn hingegen Schreiben (W) beziehungsweise Vergleichen einer schon bestehenden Datei mit einer Zieldisk verlangt wird, muß man mit D (Diskcopy) oder C (Compare) explizit die gewünschte Funktion wählen.

Eine kleine Beispiel-Batch-Datei möge das verdeutlichen. Es werden zunächst zwei Disketten von Laufwerk A: gelesen (R) und die Image-Dateien auf Harddisk C: im Verzeichnis IMAGE als DISK1 und DISK2 abgelegt. P weist das Programm an, das Einlegen der Quelldisk explizit anzufordern und erst nach Tastendruck loszulegen.

Anschließend wird jedes Image-File zweimal auf Zieldisketten im Laufwerk A: geschrieben (W und D), wobei vor dem Schreiben jeweils die explizite Anforderung (P) zum Einlegen der Zieldisk gewünscht ist. Das anschließende Vergleichen (C) soll aber ohne Rückfrage erfolgen (N), weil die richtige Disk ja eingelegt ist. Auch beim Compare muß W angegeben werden, denn sonst würde das Programm eine neue Image-Datei erstellen und Parameter d/c ignorieren.

Werden während des Vergleichs Fehler erkannt oder gibt es Fehler beim Schreiben/Formatieren, so werden diese zwar angezeigt, aber das Programm läuft weiter. Erst wenn es sein Ende erreicht hat, wird Errorlevel 1 gesetzt, so daß in der Batch-Datei entschieden werden kann, wie es weitergehen soll. Wenn keine Fehler auftreten, ist der Errorlevel wie üblich 0.

Internes

Beim Start über das Menü wird zunächst ein CASE-Block durchlaufen, der die bereits aufgeführten Einstellungen ermöglicht. Laufwerk B: läßt sich nur dann wählen, wenn die durch FORMUNIT besetzte globale Variable 'laufwerkb' einen Wert ungleich null hat (null bedeutet 'kein Laufwerk angeschlossen').

Nach Wahl von 'Diskcopy' oder 'Compare' (das erledigt beides dieselbe Prozedur) werden über 'NEW' zwei Datenpuffer eingerichtet. Der Puffer für die Image-Datei hat mit 64 KByte die Maximalgröße für Blockread/-write, der zweite ist 9 KByte lang und dient als Spurpuffer. Seine Startadresse wird an die

BIOS-Funktionen zum Schreiben und Lesen von Sektoren übergeben.

Eigentlich könnte man auch nur mit dem großen Puffer allein arbeiten, indem man jeweils den aktuellen Zeiger an die BIOS-Routinen weiterreicht. Die Praxis hat aber gezeigt, daß bestimmte Verwicklungen aus ROM-BIOS, DOS-Version und Lage des Programms im Speicher manchmal zu DMA-Fehlern führten (DMA boundary crossing, siehe Kasten). Wenn dieser Fehler auftritt, wird der Spurpuffer einfach erneut angelegt, und zwar 'hinter' dem alten Puffer. Auf Grund seiner geringen Größe ist dann sichergestellt, daß er keine DMA-Grenze mehr kreuzt.

In diesem Zusammenhang sei auch angemerkt, daß dieser Fehler in unserem Formatprogramm aus c't 2/90 nicht auf diese Art behandelt wird. Aber auch dort muß ein Puffer angelegt werden, um Boot-Sektor, FATs und Directory sektorweise auf die Diskette zu schreiben. Sollten Sie also auf gelegentliche, schwer verständliche Fehler stoßen (DMA boundary crossing wird nicht als eigenständige Fehlermeldung ausgegeben), so sollten Sie sich die BIOS-Fehlermeldungen explizit ausgeben lassen. Wenn Nummer 9 auftaucht, dann sollten Sie wie bei DCC den Puffer ein zweites Mal anlegen.

Als nächstes prüft das Programm, ob eine Datei mit gleichem Namen wie im Hauptmenü schon vorhanden ist und ob es sie gegebenenfalls überschreiben oder direkt auf die Zieldiskette schreiben soll. Im folgenden wird der etwas aufwendigere Fall weiterverfolgt, daß die Quelldiskette einzulesen ist.

Nach dem Einlegen der Quelldiskette wird deren Boot-Sektor zur Formatermittlung eingelesen. Bei DOS-Disketten ist das immer der absolut erste Sektor (Spur 0, Seite 0, Sektor 1). Die Formatinformationen - Anzahl der Spuren, Seiten- und Sektorenzahl (pro Spur) - zeigt DCC in einem eigenen Fenster beim Kopieren an. Wenn sich kein Format ermitteln läßt, fordert DCC zur manuellen Parametereingabe auf.

Bei gültigen Formatdaten wird nun die richtige Schreibrate für das Diskettenformat (Normal oder High Density) eingestellt und eine passende Laufwerkstabelle (DDPT, siehe[1]) eingerichtet. Bei manuell eingestellten, nicht DOS-üblichen Formaten werden diese Werte für die BIOS-Aufrufe in 'verträgliche' Werte umgerechnet.

Die Daten von der Quelldiskette werden nun spurweise eingelesen und auf dem Massenspeicher in 64-KB-Blöcken abgelegt. Die Image-Datei enthält daneben noch einen 512 Byte langen Vorspannblock: in den ersten Bytes sind Anzahl der Spuren, Seiten- und Sektoren pro Spur abgelegt, die letzten 18 Bytes stellen einen Identifizierungs-String dar, der eine eindeutige Erkennung als Diskcopy-Datei ermöglicht. Die Datei wird nach dem Einlesen der Quelldaten wieder geschlossen und liegt nun zur weiteren Bearbeitung vor.

Das Schreiben der Image-Datei auf eine Zieldiskette geschieht analog zum Einlesen. Die Formatinformationen im Vorspannblock müssen jedoch zuerst mit denen der Zieldiskette verglichen werden. Stimmen beide nicht überein, so wird das Format der Zieldisk ausgegeben und zur Sicherheit nachgefragt, ob sie nun doch formatiert werden soll.

Beim Vergleichen erfolgt diese Überprüfung auch, da ein Vergleich von Disketten in unterschiedlichem Format keinen Sinn macht. Bei manueller Parameterübergabe werden diese Überprüfungen natürlich außer Kraft gesetzt, der Anwender ist dann allein für Sinn und Unsinn seiner Aktionen verantwortlich.

Manipulieren

Die Image-Dateien lassen sich natürlich - wenn sie erst mal auf Festplatte stehen - noch weiterbearbeiten. Wem Utilities fehlen, mit denen man sich komfortabel (also nicht mit DEBUG) sektorweise durch Disketten schlängeln kann, der kann dies nun mit dateiorientierten Hilfsmitteln tun.

Es gibt auch Fälle, in denen selbst DEBUG die Mitarbeit verweigert. Die Versionen ab DOS 3.2 lassen nämlich nur noch den Zugriff auf Disketten zu, die in einem der DOS-Formate erstellt und mit entsprechenden Angaben versehen sind. Auch wenn man nicht recht weiß, mit was für einer Diskette man es eigentlich zu tun hat, kann man über manuelle Parametereingabe zu einer (wenn auch nur teilweisen)

Image-Datei gelangen, solange die untersuchte Disk mit 512-Byte-Sektoren formatiert ist. Man liegt mit 8 Sektoren, 1 Seite und 1 Spur eigentlich immer erst mal richtig. Jetzt kann man über die Image-Datei einen Blick in den 'Boot-Sektor' oder generell die erste Spur tun und weitere Schlüsse auf das Format ziehen.

Außerdem kann man die Image-Dateien natürlich insofern manipulieren, daß man die Abbilder zweier 360-KB-Scheiben in das einer 720-KB-Scheibe umgestaltet. Das ist wie gesagt nicht gerade unkompliziert, entsprechende Experimente mit Image-Dateien fallen aber sicherlich leichter als mit Disketten.

'Diskettensparer' werden ansonsten zunächst enttäuscht feststellen, daß die Image-Datei einer 360-KB-Scheibe 369 KByte Platz verschlingt: es müssen ja nicht nur der Datenspeicherplatz, sondern auch Boot-Sektor, FATs und Directory gespeichert werden. Zwei solche Dateien passen also leider nicht auf eine 720-KB-Scheibe.

Wer verhindern will, daß seine Diskettenschränke überquellen, aber keine komprimierenden Backup-Programme sein eigen nennt, kann sich auch hier behelfen. Mit den Shareware-Kompressoren PKARC/PKXARC etwa lassen sich Disk-Images von 360-KB-Scheiben fast immer so weit reduzieren, daß mindestens zwei auf eine 720-KB-Scheibe passen. Es ist vielfach auch möglich, vier 360-KB-Scheiben so zusammenzuquetschen, daß sie nur noch eine 1,2-MB-Scheibe belegen. Man kann diese Prozedur natürlich auch einfach durch Zusammenkopieren und Komprimieren der Dateien bewerkstelligen; bei der Benutzung von DISKCOPY-Files bleibt aber die exakte Rekonstruktion der Original-Scheiben möglich.

Formatkreuzungen

Seit die 3,5-Zoll-Laufwerke immer mehr an Bedeutung gewonnen haben, sind die meisten ATs mit jeweils einem 3,5- (720 KB/1,44 MB) und einem 5,25-Zoll-Laufwerk (360 KB/1,2 MB) ausgestattet. Leider kann immer nur eines davon als Boot-Laufwerk A: fungieren, und Änderungen (Stecker tauschen, neues Setup) sind in engen AT-Gehäusen eine Qual. Und prompt ist die dringend benötigte Software nur im 'falschen' Format erhältlich, und das unsäglich dumme Installationsprogramm weigert sich, in Laufwerk B: zu arbeiten.

DCC kann Ihnen in einigen Fällen viel Mühe ersparen. Im Prinzip reicht dazu übrigens schon das Formatierprogramm aus c't 2/90 in Verbindung mit dem normalen DISKCOPY aus. Zwei Beispiele:

Laufwerk A: im AT ist ein Multifunktionslaufwerk im 5,25-Zoll-Format für 360 KByte und 1,2 MByte, Laufwerk B: ein 3,5-Zoll-Laufwerk für 720 KB/1,44 MB. Die Software war nur auf 3,5-Zoll-Scheibe mit 720 KB zu bekommen, und das Installationsprogramm verlangt, in Laufwerk A: zu stecken. Zwei Vorgehensweisen sind jetzt möglich.

Sie formatieren mit unserm [Spezial-Formatter](#) aus c't 2/90 eine Standard-Diskette (nicht High Density) auf 720 KB. Außerdem müssen Sie das im selben Artikel vorgestellte kleine Programm INT13h resident laden, damit es das Double-Stepping im 1,2-MB-Laufwerk abstellt. Nur so wird DOS-DISKCOPY in die Lage versetzt, den Inhalt der 3,5-Zoll- auf die 5,25-Zoll-Disk zu kopieren, und nur mit geladenem INT13h können Sie später diese 80-Spur-Diskette fehlerfrei im 1,2-MB-Laufwerk schreiben und lesen.

Alternativ können Sie gleich DCC benutzen, da es die 720-KB-Formatierung, die DISKCOPY auf einem 1,2-MB-Drive verweigern würde, anstandslos durchführt. Später brauchen Sie natürlich auch hier INT13h.

Im zweiten Beispiel sei ein 3,5-Zoll-Drive (720 KB/1,44 MB) als Laufwerk A: eingebaut, das Laufwerk B: ein 5,25-Zoll-Drive (1,2 MB). Zwar hat sich die Lage mittlerweile entspannt, aber es kann immer noch vorkommen, daß man Unix oder Xenix nur auf 5,25-Zoll-Scheiben mit 1,2 MByte bekommt. Auch hier können unser Formatter nebst DISKCOPY oder DCC allein helfen: Damit ist es nämlich möglich, eine 1,44-MB-Scheibe als 1,2-MB-Disk zu 'tarnen', die von DOS auch 'im falschen Laufwerk' beim

Schreiben und Lesen nicht beanstandet wird. DISKCOPY oder FORMAT von DOS würden sich jedoch strikt weigern, solche ungehörigen Kombinationen aus Format und Laufwerk zu formatieren. (gr)

Literatur

[1] Volker Bührmann, Mit eigenem Format, Flexibles Formatierprogramm (nicht nur) für ATs, [c't 2/90, S. 202](#) ff.

Kasten 1

COMPARE.ASM enthält die Routinen zum Vergleichen von Disketten und wird als OBJ-File zu DCC.PAS hinzugelinkt. Wer keinen Assembler hat, kann die Hexdump-Datei via DEBUG zur Erzeugung des OBJ-Files benutzen (siehe Text).

```
1  ; Name      : COMPARE.ASM
2  ; Aufgabe   : Hilfsprogramm für Diskcopy zum Diskettencompare.
3  ;           : Assemblieren mit MASM ab 4.0 o.ä. zu COMPARE.OBJ
4  ;           : Wird dann von Turbo-Pascal mit DCC
5  ;           : zusammengelinkt.
6
7  code        segment
8              assume  cs:code
9
10 ; Parameter (Offset +2 wegen push bp)
11
12 spuffer      equ     dword ptr ss:[bp+12]
13 dpuffer      equ     dword ptr ss:[bp+8]
14 anzsektor    equ     byte  ptr ss:[bp+6]
15
16 vergleich    proc     far
17 public       vergleich
18 mov          dx, ds      ;Datensegment retten
19 push         bp
20 mov          bp, sp      ;Zeiger auf Stack
21 lds          si, spuffer ;Adresse der Sourcedaten
22 les          di, dpuffer ;Adresse der Destinationdaten
23 mov          bh, 0
24 mov          bl, anzsektor ;Anzahl der Comparesektoren
25 mov          cl, 8
26 sal          bx, cl      ;*256 = Anzahl der Wörter
27 mov          cx, bx      ;Als Zähler verwenden
28 cld
29 repe         cmpsw       ;Vergleich der Daten
30 cmp          cx, 0        ;Alle Daten verglichen?
31 je           weiter      ;Ja
32 mov          ax, 1        ;Fehler bei Vergleich
33 jmp          short ende
34 weiter:
35 mov          ax, 0        ;Kein Fehler aufgetreten
36
37 ende:
38 mov          ds, dx      ;DS von Turbo wiederherstellen
39 pop          bp
40 ret          10
41 vergleich    endp
42 code         ends
43 end
```

```
eds:100 80 0D 00 0B 43 4F 4D 50 41 52 45 2E 41 53 4D 52
eds:110 88 1F 00 00 00 54 75 72 62 6F 20 41 73 73 65 6D
eds:120 62 6C 65 72 20 20 56 65 72 73 69 6F 6E 20 31 2E
```



```
eds:130 30 BA 88 13 00 40 E9 8C 85 24 14 0B 43 4F 4D 50
eds:140 41 52 45 2E 41 53 4D D2 88 03 00 40 E9 4C 96 02
eds:150 00 00 68 96 06 00 04 43 4F 44 45 45 98 07 00 60
eds:160 2C 00 02 01 01 D1 90 10 00 00 01 09 56 45 52 47
eds:170 4C 45 49 43 48 00 00 00 BD 88 04 00 40 A2 01 91
eds:180 A0 30 00 01 00 00 8C DA 55 8B EC C5 76 0C C4 7E
eds:190 08 B7 00 8A 5E 06 B1 08 D3 E3 8B CB FC F3 A7 83
eds:1a0 F9 00 74 05 B8 01 00 EB 03 B8 00 00 8E DA 5D CA
eds:1b0 0A 00 7F 8A 02 00 00 74
```

```
ncompare.obj
rcx
b8
w
q
```

Kasten 2

Media-Byte-Zuordnung von DCC

Media-Byte	Kapazität	Spuren	Seiten	Sekt./Spur	Schreibdichte
F0h	1,44 MB	80	2	18	High Density
F9h	1,2 MB	80	2	15	High Density
FCh	180 KB	40	1	9	Double Density
FDh	360 KB	40	2	9	Double Density
FEh	160 KB	40	1	8	Double Density
FFh	320 KB	40	2	8	Double Density

Kasten 3

Im Batch-Betrieb funktioniert DCC etwas anders als per Menü. Hier werden zwei Disketten je zweimal dupliziert und die Kopien mit dem Original verglichen.

```
REM allgemeine Kommando-Syntax:
REM DCC r/w q/z dat p/n [d/c]

REM zwei Disketten auf Platte einlesen:

DCC R A: C:\IMAGE\DISK1 P
DCC R A: C:\IMAGE\DISK2 P

REM von jeder Disk zwei Kopien machen
REM und dabei jeweils zur Sicherheit
REM ein Diskcompare ohne Rückfragen
REM anschließen:

DCC W A: C:\IMAGE\DISK1 P D
DCC W A: C:\IMAGE\DISK1 N C

DCC W A: C:\IMAGE\DISK1 P D
DCC W A: C:\IMAGE\DISK1 N C

DCC W A: C:\IMAGE\DISK2 P D
DCC W A: C:\IMAGE\DISK2 N C

DCC W A: C:\IMAGE\DISK2 P D
DCC W A: C:\IMAGE\DISK2 N C
```

Kasten 4

DMA Boundary Crossing

Die vorsintflutlichen DMA-Chips in PCs und ATs können mit ihren 16-Bit-Adressen nur 64 KByte im Stück transferieren. Die PC-Entwickler haben nun ein zusätzliches Problem dadurch eingebaut, daß diese 64-KB-Blöcke immer auf absoluten 20-Bit-Adressen von 10000h, 20000h und so weiter aufeinanderfolgen. Anders ausgedrückt: Der DMA-Controller kann - im Gegensatz zur CPU - keineswegs auf ein beliebiges Segment eingestellt werden und dann einen Offset-Bereich von 64 KByte bearbeiten.

Ein Beispiel: Mit NEW von Pascal werde ein 64 KB großer Speicherblock reserviert. Turbo-Pascal normalisiert typischerweise die Startadresse so, daß der Zeiger auf diesen Speicherblock einen Offset-Anteil von 0 erhält. Es sei angenommen, daß die Startadresse den Wert 8845:0000h erhalte. Die absolute 20-Bit-Adresse lautet dann 88450h. Arbeitet man mit diesem Puffer in Verbindung mit DMA-Zugriffen, dann muß irgendwann die magische, von der Hardware vorgegebene Grenze bei 90000h gekreuzt werden, und zwar mitten innerhalb eines Sektortransfers von 200h Länge.

Dieses Problem ist so alt wie PC und DOS, und längst kümmert sich das BIOS des DOS selbst darum, nachdem das ROM-BIOS hier anfangs seine Probleme hatte. Das Verfahren, mit dem die Klippen des DMA-Crossing umschifft werden, funktioniert in der Regel recht gut, so daß der Programmierer selten mit der Nase darauf gestupst wird, daß es auch mal schiefgehen kann.

Unsere Erfahrungen zeigten, daß es mit einigen Rechnern sehr wohl zu der bewußten Fehlermeldung des BIOS (09h, Attempt to DMA across a 64K boundary) kommen kann. Unter anderem zeigte sich DRDOS 3.41 hier anfällig, aber auch ein 80386er mit AMI-BIOS unter PCDOS 4.0 präsentierte den Fehler.

Es ist daher grundsätzlich ratsam, mit kleinen DMA-Puffern zu arbeiten, die möglichst mehrfach in einem Bereich von 64 KByte unterzubringen sind. Damit kann man leicht der Empfehlung des BIOS-Handbuchs folgen, den Puffer bei DMA-Fehler ein zweites Mal anzulegen, am besten am Ende des ersten Puffers.

Kasten 5

DCC.PAS ist in Turbo-Pascal 5.0 geschrieben und ersetzt die Programme DISKCOPY und DISKCOMP von DOS. Es ermöglicht unter anderem das Kopieren 'großer' Disketten auf einem Laufwerk ohne dauernde Disk-Wechsel. DCC basiert dabei auf FORMUNIT.PAS aus c't 2/90.

```

1  {*** Bührmann / c't 3/1990 ***}
2
3  {*** benötigt Formunit aus c't 2/1990 und Compare.obj ***}
4
5  PROGRAM dcc; {$I-,E-,V-,O-,R-}
6  USES Crt, Formunit;
7
8  CONST
9    NAME : STRING[25] = 'Bührmann - c't 1990';           {Dateierkennung}
10
11  TYPE
12    sektor = ARRAY[0..511] OF Byte;                      {Ein Sektor}
13    puffer = ARRAY[0..126] OF sektor;                     {64 KB Datenpuffer}
14    dest = ARRAY[0..19] OF sektor;                        {Vergleich für max. 20 Sektoren}
15    pufferzeig = ^puffer;                                 {Zeiger auf Datenpuffer}
16    destzeig = ^dest;                                     {Zeiger auf Comparepuffer}
17
18  VAR
19    source, target : ShortInt;                             {Quell- und Zieldisk}
20    format, verify : Boolean;                              {Format oder Verify ?}
21    compare : Boolean;                                     {Disketten vergleichen ?}
22    manuell, estep : Boolean; {Manuelle Parametereingabe ?, Einzelstep ?}
23    prompt, low : Boolean; {Einlegen der Disk bestätigen ?, Low Density}
24    sdisk, tdisk, vfehler : Boolean; {Fehler bei Read/Write oder Verify}
25    trk, side, sek : Byte;                                {Manuelle Parameter}

```

```
26 batch, fparam : Boolean;      {Flag für Batchbetrieb, Fehler bei Batch}
27 taste, einzel, srate : Char;   {Taste, Einzelstep, Schreibrate}
28 schreibpuffer : pufferzeig;    {64 KB Datenpuffer}
29 diskpuffer, destpuffer : destzeig; {10 KB Disketten-, Comparepuffer}
30 heap : Pointer;                {Zeiger auf aktuellen Heap}
31 diskprompt, diskcomp : STRING[1]; {Prompt, Diskcopy oder Compare}
32 aktion, laufwstr : STRING[1];   {Read oder Write bei Batch, Laufwerk}
33 datdmy, dateiname : STRING[50]; {Massenspeicherdatei}
34 {*****Unterprogramme*****}
35 PROCEDURE ausgang(VAR datei : FILE);
36 BEGIN
37   Close(datei); Release(heap); {alle dynamischen Variablen löschen}
38   laufwerkstabalt; {Datei schließen und alte Laufwerkstabelle}
39 END;
40 {*****}
41 FUNCTION dnotready : Char;      {Keine Diskette im Laufwerk}
42 BEGIN
43   IF NOT batch THEN window(1, 13, 80, 25);
44   WriteLn;
45   WriteLn('Es befindet sich keine Diskette im Laufwerk, ');
46   WriteLn('oder die Laufwerksklappe ist nicht geschlossen. ');
47   WriteLn('Bitte Diskette einlegen, oder Klappe schließen. ');
48   WriteLn('Abbruch mit ESC ');
49   dnotready := readkey; {Taste lesen}
50   IF NOT batch THEN
51     BEGIN
52       clrscr; window(1, 1, 80, 25); {Fenster löschen}
53     END;
54 END;
55 {*****}
56 PROCEDURE diskettenformat(anzspur, anzseite, anzsektor : Byte);
57 VAR x, y : Byte;
58 BEGIN
59   x := wherex; y := wherey; {Alten Cursor merken}
60   IF NOT batch THEN window(60, 2, 80, 25);
61   WriteLn; WriteLn('Diskettenformat');
62   WriteLn(anzspur:4, ' : Spuren');
63   WriteLn(anzseite:4, ' : Seite(n)');
64   WriteLn(anzsektor:4, ' : Sektoren');
65   WriteLn(Round(anzspur*anzsektor*anzseite*0.5):4, ' : KByte');
66   IF NOT batch THEN
67     BEGIN
68       window(1, 1, 80, 25);
69       gotoxy(x, y); {Cursor an alte Position}
70     END;
71 END;
72 {*****}
73 {$F+}
74 FUNCTION vergleich(VAR spuffer, dpuffer;
75   anzsektor : Byte) : Boolean; EXTERNAL; {$L compare.obj}
76 {$F-}
77 {*****}
78 PROCEDURE formchange;
79 BEGIN
80   IF NOT batch THEN
81     BEGIN
82       highvideo; gotoxy(1, 4);
83       WriteLn('(F)ormat : ON '); WriteLn('(V)erify : ON ');
84       lowvideo;
85     END;
86   format := True; verify := True;
87 END;
88 {*****}
89 PROCEDURE parameter;
90 BEGIN
91   window(1, 15, 80, 25); gotoxy(14, 1);
92   Write('Manuelle Parametereingabe, ');
93   WriteLn('ohne Überprüfung der Werte');
94   gotoxy(1, 3); Write('Anzahl der Spuren : '); ReadLn(trk);
```

```
95 Write('Anzahl der Seiten : '); ReadLn(side);
96 Write('Anzahl der Sektoren pro Seite : '); ReadLn(sek);
97 IF (laufwerka = 2) OR (laufwerkb = 2) OR
98 (laufwerka = 4) OR (laufwerkb = 4) THEN
99 BEGIN
100 Write('Schreibdichte (H/L) : '); ReadLn(srate);
101 srate := Ucase(srate); {Gewünschte Schreibrate}
102 IF srate = 'H' THEN low := False ELSE low := True;
103 IF (laufwerka = 2) OR (laufwerkb = 2) THEN
104 BEGIN {Schrittweite}
105 Write('Einzelstep (J/N) : '); ReadLn(einzel);
106 einzel := Ucase(einzel);
107 IF einzel = 'J' THEN estep := True ELSE estep := False;
108 END;
109 END;
110 clrscr;
111 window(1, 1, 80, 25); gotoxy(13, 9);
112 END;
113 {*****}
114 FUNCTION bootsektor(VAR anzspur, anzseite,
115 anzsektor : LongInt; laufwerk : Byte;
116 VAR media : Byte) : Byte;
117 TYPE
118 boot = RECORD
119 nothing1 : ARRAY[0..18] OF Byte;
120 sektoren : Word; {Anzahl Sektoren auf Disk}
121 nothing2 : ARRAY[0..2] OF Byte;
122 sekprotr : Word; {Sektoren pro Spur}
123 anzkopf : Word; {Anzahl der Seiten}
124 nothing3 : ARRAY[0..995] OF Byte;
125 END;
126 VAR
127 bootpuffer : boot;
128 BEGIN
129 bootpuffer.sekprotr := $ff; bootpuffer.anzkopf := $ff;
130 {Physikalische Parameter löschen, Bootsektor + 1. FAT Sektor lesen}
131 bootsektor := readwriteverify(2, 0, 0, 1, 2,
132 laufwerk, bootpuffer);
133 anzsektor := bootpuffer.sekprotr;
134 anzseite := bootpuffer.anzkopf;
135 IF (anzseite <> 0) AND (anzsektor <> 0) THEN
136 anzspur := Round(bootpuffer.sektoren/anzseite/anzsektor)
137 ELSE anzspur := 0; {Ungültiges Format}
138 media := bootpuffer.nothing3[484]; {Mediabyte lesen}
139 IF (anzspur = 0) AND (media >= $F0) THEN anzspur := 1;
140 {Es kann doch ein gültiges Format sein}
141 END;
142 {*****}
143 PROCEDURE diskcopy;
144 VAR
145 anzspur, anzseite, anzsektor : LongInt; {Quellparameter}
146 dspur, dseite, dsektor : LongInt; {Zielparameter}
147 spur, seite : Byte; {Aktuelle Werte}
148 media : Byte; {Mediabyte der Diskette}
149 gesamt, dummy : Word; {Gesamte Diskkapazität}
150 maxsektor, sekread, i : Word;
151 kap, abssektor, anzahl : Byte; {Art der Schreibrate, Sektor}
152 artsources, arttarget : Byte; {Art der Laufwerke}
153 tabelle : Boolean; {Neue DDPT}
154 fileerror : Boolean; {Dateierkennung}
155 booterr : Boolean; {Bootsektorfehler}
156 fehler : Byte; {Fehler bei Diskzugriff aufgetreten}
157 taste : Char;
158 datei : FILE;
159 BEGIN
160 Mark(heap); {aktuellen Heap merken}
161 IF NOT batch THEN
162 BEGIN
163 highvideo;
```

```
164     IF compare THEN
165         BEGIN
166             gotoxy(23, 8); Write('Diskettencompare von ');
167         END
168     ELSE
169         BEGIN
170             gotoxy(15, 9); Write('Diskettendiskcopy von ');
171         END;
172     IF source = 0 THEN Write('A') ELSE Write('B');
173     Write(' nach ');
174     IF target = 0 THEN WriteLn('A') ELSE WriteLn('B');
175     lowvideo;
176     END;
177     sdisk := False;           {Fehler aufgetreten, Disk unbrauchbar}
178     booterr := False;        {Unbrauchbarer Bootsektor}
179     IF source = 0 THEN artsources := laufwerka
180     ELSE artsources := laufwerkb;
181     IF target = 0 THEN arttarget := laufwerka
182     ELSE arttarget := laufwerkb;
183     tabelle := False;        {Noch keine neue DDPT}
184     taste := 'ü';
185     New(schreibpuffer);      {64 KB Datenpuffer}
186     New(diskpuffer);         {10 KB Diskettenpuffer}
187     Assign(datei, dateiname);
188     IF aktion = 'R' THEN Erase(datei);      {Neues Imagefile anlegen}
189     Reset(datei, 512);                     {Datei zum Schreiben öffnen}
190     IF (IoResult = 0) AND NOT batch THEN
191         BEGIN                             {Datei ist schon auf Massenspeicher vorhanden}
192             IF NOT batch THEN gotoxy(1, 11);
193             IF NOT compare THEN
194                 BEGIN
195                     Write('Datei vorhanden : (Ü)berschreiben oder ');
196                     Write('auf (Z)ieldiskette');
197                 END
198             ELSE
199                 BEGIN
200                     Write('Comparedatei vorhanden : (Ü)berschreiben oder ');
201                     Write('mit (Z)ieldiskette vergleichen');
202                 END;
203             REPEAT
204                 taste := Ucase(readkey);
205             UNTIL (taste = #129) OR (taste = #154) OR
206                 (taste = 'Z') OR (taste = #27);
207             delline; Close(datei);
208         END;
209     IF taste = #27 THEN
210         BEGIN
211             Release(heap);
212             IF NOT batch THEN Exit ELSE Halt(1);
213         END;
214     IF batch AND (aktion = 'W') THEN taste := 'Z'; {Direkt auf Zieldisk}
215     IF ((taste = #129) OR (taste = #154)) THEN
216         BEGIN
217             IF NOT batch THEN gotoxy(1, 11);
218             IF prompt THEN
219                 BEGIN
220                     Write('Bitte Quelldiskette in ');
221                     highvideo;
222                     IF source = 0 THEN Write('A') ELSE Write('B');
223                     lowvideo;
224                     WriteLn(' einlegen, weiter mit beliebiger Taste');
225                     taste := readkey;
226                     IF taste = #27 THEN
227                         BEGIN
228                             Release(heap);
229                             IF NOT batch THEN Exit ELSE Halt(1);
230                         END;
231                     delline;
232                 END;
233             END;
```

```
233 REPEAT
234   IF NOT manuell THEN                                     {Keine manuellen Parameter}
235     fehler := bootsektor(anzspur, anzseite, anzsektor,
236                        source, media)
237   ELSE
238     BEGIN                                                 {Manuelle Parameter für Diskcopy}
239       anzspur := trk;
240       anzseite := side;
241       anzsektor := sek;
242     END;
243   IF fehler = $80 THEN taste := dnotready;   {Keine Disk eingelegt}
244   IF taste = #27 THEN
245     BEGIN
246       Release(heap);
247       IF NOT batch THEN Exit ELSE Halt(1);
248     END;
249   UNTIL fehler <> $80;                                     {Abbruch, wenn Disk ready}
250   gesamt := Round(anzspur*anzsektor*anzseite*0.5);
251   IF NOT manuell THEN                                     {keine manuellen Parameter}
252     BEGIN
253       IF (gesamt <> 360) AND (gesamt <> 720) AND
254          (gesamt <> 1200) AND (gesamt <> 1440) THEN
255         {keine Standardformate}
256         IF (media < $F0) THEN anzspur := 0
257         ELSE booterr := True;
258         {Unbekanntes Diskettenformat}
259         IF ((fehler <> 0) OR (anzspur = 0)) THEN
260           BEGIN
261             IF NOT batch THEN gotoxy(1, 11);
262             Write('Bootsektor ist nicht lesbar, ');
263             WriteLn('oder ungültiges Diskettenformat  !');
264             Write('Diskettenparameter bitte Manuell eingeben ');
265             taste := readkey;
266             Release(heap);
267             IF NOT batch THEN Exit ELSE Halt(1);
268           END;
269         IF booterr THEN
270           {Mediabyte auswerten, wenn kein brauchbarer Bootsektor}
271           CASE media OF
272             $F0 : BEGIN anzspur := 80; anzseite := 2; anzsektor := 18; END;
273                   {1440 KB}
274             $F9 : BEGIN anzspur := 80; anzseite := 2; anzsektor := 15; END;
275                   {1220 KB}
276             $FC : BEGIN anzspur := 40; anzseite := 1; anzsektor := 9; END;
277                   {180 KB}
278             $FD : BEGIN anzspur := 40; anzseite := 2; anzsektor := 9; END;
279                   {360 KB}
280             $FE : BEGIN anzspur := 40; anzseite := 1; anzsektor := 8; END;
281                   {160 KB}
282             $FF : BEGIN anzspur := 40; anzseite := 2; anzsektor := 8; END;
283                   {320 KB}
284           END;
285         END;
286       diskettenformat(anzspur, anzseite, anzsektor);
287       {Format der Diskette ausgeben}
288       IF anzsektor < 11 THEN kap := 1 ELSE kap := 3;
289       {Niedrige oder hohe Schreibdichte als Default}
290       IF manuell THEN                                     {Parameter manuell eingegeben}
291         BEGIN
292           IF artsourc = 1 THEN kap := 1;                 {360 KB, 5.25 Laufwerk}
293           IF artsourc = 3 THEN kap := 2;                 {720 KB, 3.5 Laufwerk}
294           IF artsourc = 2 THEN                           {1200 KB, 5.25 Laufwerk}
295             BEGIN
296               IF estep AND low THEN kap := 2
297               ELSE IF estep AND NOT low THEN kap := 3
298               ELSE kap := 1;
299             END;
300           IF artsourc = 4 THEN
301             IF NOT low THEN kap := 4 ELSE kap := 1
```

```
302     END;
303 CASE gesamt OF
304     360 : kap := 1;                {Schreibdichte bei Standardformaten}
305     720 : kap := 2;
306     1200 : kap := 3;
307     1440 : kap := 4;
308 END;
309 schreibrate(artsource, kap, source);
310 laufwerkstabneu;
311 maxsektor := anzspur*anzseite*anzsektor;      {abs. Sektorenzahl}
312 Rewrite(datei, 512);
313 FOR i := 0 TO 511 DO schreibpuffer^[0, i] := 0;
314 {Anfangsblock der Datei löschen}
315 schreibpuffer^[0, 0] := anzspur;
316 schreibpuffer^[0, 1] := anzseite;
317 schreibpuffer^[0, 2] := anzsektor;
318 FOR i := 493 TO 511 DO schreibpuffer^[0, i] := Ord(NAME[i-492]);
319 {Dateikennung in Puffer schreiben}
320 BlockWrite(datei, schreibpuffer^[0, 0], 1);
321 {Physikalische Parameter der Diskette in Datei schreiben}
322 Dec(anzspur); Dec(anzseite);      {Spur und Seite beginnen bei 0}
323 sekread := 0; abssektor := 0;
324 FOR spur := 0 TO anzspur DO
325     BEGIN
326     FOR seite := 0 TO anzseite DO
327     BEGIN
328     IF (abssektor+anzsektor) > 126 THEN
329     BEGIN
330     BlockWrite(datei, schreibpuffer^[0, 0],
331     abssektor, dummy);
332     IF abssektor <> dummy THEN
333     BEGIN
334     IF NOT batch THEN gotoxy(8, 6);
335     Write('Zwischenspeicher ist voll !');
336     taste := readkey;
337     ausgang(datei);
338     Erase(datei);      {Datei schließen und löschen}
339     IF NOT batch THEN Exit ELSE Halt(1);
340     END;
341     abssektor := 0;
342     END;
343 REPEAT
344     fehler := readwriteverify(2, spur, seite, 1, anzsektor,
345     source, diskpuffer^[0, 0]);
346     {Diskettendaten in Diskpuffer schreiben}
347     Move(diskpuffer^[0, 0],
348     schreibpuffer^[abssektor, 0], anzsektor*512);
349     {Daten in Schreibpuffer kopieren}
350     IF fehler = $80 THEN taste := dnotready;
351     {Keine Diskette eingelegt}
352     IF taste = #27 THEN
353     BEGIN
354     ausgang(datei);
355     Erase(datei);
356     IF NOT batch THEN Exit ELSE Halt(1);
357     END;
358     IF fehler = 9 THEN New(diskpuffer);
359     {DMA Übertragung über 64 KB Grenze}
360     UNTIL (fehler <> $80) AND (fehler <> 9);
361     {Abbruch, wenn Disk ready und kein DMA Fehler}
362     IF fehler <> 0 THEN
363     BEGIN
364     IF NOT batch THEN gotoxy(1, 14);
365     sdisk := True;      {Fehler bei Quelldisk}
366     WriteLn(' Fehler Seite : ', seite, ' Spur : ', spur);
367     END;
368     Inc(abssektor, anzsektor);
369     Inc(sekread, anzsektor);
370     IF NOT batch THEN gotoxy(1, 12) ELSE gotoxy(1, wherey);
```

```
371      Write('Lese Seite : ', seite, '   Spur : ', spur);
372      IF keypressed THEN                      {wurde eine Taste betätigt ?}
373      BEGIN
374          taste := readkey;
375          IF taste = #27 THEN                  {Abbruch durch ESC}
376          BEGIN
377              ausgang(datei);
378              Erase(datei);
379              IF NOT batch THEN Exit ELSE Halt(1);
380          END;
381      END;
382  END;
383  END;
384  BlockWrite(datei, schreibpuffer^[0, 0], abssektor);
385  Close(datei); delline; delline; delline;
386  laufwerkstabalt;
387  END;
388  IF aktion <> 'R' THEN
389  BEGIN
390      REPEAT
391          tdisk := False;
392          vfehler := False;
393          Reset(datei, 512);                  {Datei zum Schreiben öffnen}
394          IF batch AND (IoResult <> 0) THEN
395          BEGIN
396              WriteLn('Massenspeicherdatei existiert nicht !');
397              Halt(1);
398          END;
399          BlockRead(datei, schreibpuffer^[0, 0], 1);
400          {Physikalische Parameter der Diskcopydatei lesen}
401          anzscur := schreibpuffer^[0, 0];
402          anzseite := schreibpuffer^[0, 1];
403          anzsektor := schreibpuffer^[0, 2];
404          i := 493;                            {Start der Dateikennung}
405          REPEAT
406              IF Ord(NAME[i-492]) <> schreibpuffer^[0, i] THEN
407                  fileerror := True            {Keine Backupdatei}
408              ELSE fileerror := False;
409              Inc(i, 1);
410          UNTIL fileerror OR (i > 511);
411          IF fileerror THEN
412          BEGIN
413              IF NOT batch THEN gotoxy(1, 11);
414              Write('Datei ist kein Diskcopyfile. ');
415              Write('Weiter mit beliebiger Taste');
416              taste := readkey;
417              Close(datei);
418              Release(heap);
419              IF NOT batch THEN Exit ELSE Halt(1);
420          END;
421          gesamt := Round(anzscur*anzsektor*anzseite*0.5);
422          {Diskettenkapazität}
423          IF anzsektor < 11 THEN kap := 1 ELSE kap := 3;
424          {Niedrige oder hohe Schreibdichte als Default}
425          IF manuell THEN                      {Parameter manuell eingegeben}
426          BEGIN                                {arttarget}
427              IF arttarget = 1 THEN kap := 1;    {360 KB, 5.25 Laufwerk}
428              IF arttarget = 3 THEN kap := 2;    {720 KB, 3.5 Laufwerk}
429              IF arttarget = 2 THEN              {1200 KB, 5.25 Laufwerk}
430              BEGIN
431                  IF estep AND low THEN kap := 2
432                  ELSE IF estep AND NOT low THEN kap := 3
433                  ELSE kap := 1;
434              END;
435              IF arttarget = 4 THEN
436                  IF NOT low THEN kap := 4 ELSE kap := 1
437              END;
438          CASE gesamt OF
439              360 : kap := 1;                  {Schreibdichte bei Standardformaten}
```



```
440       720 : kap := 2;
441       1200 : kap := 3;
442       1440 : kap := 4;
443   END;
444   diskettenformat(anzspur, anzseite, anzsektor);
445   maxsektor := anzspur*anzseite*anzsektor;      {abs. Sektorenzahl}
446   IF NOT batch THEN gotoxy(1, 11);
447   IF prompt THEN
448     BEGIN
449       Write('Bitte Zieldiskette in ');
450       highvideo;
451       IF target = 0 THEN Write('A') ELSE Write('B');
452       lowvideo;
453       WriteLn(' einlegen, weiter mit beliebiger Taste ');
454       taste := readkey;
455       IF taste = #27 THEN
456         BEGIN
457           Close(datei);
458           Release(heap);
459           IF tabelle THEN laufwerkstabalt;
460           IF NOT batch THEN Exit ELSE Halt(1);
461         END;
462       delline;
463     END;
464   Dec(anzspur);
465   Dec(anzseite);      {Seite und Spur beginnen bei 0}
466   IF (NOT format) OR (format AND compare) THEN
467     BEGIN
468       REPEAT
469         fehler := bootsektor(dspur, dseite, dsektor,
470                               target, media);
471         IF fehler = $80 THEN taste := dnotready;
472         {Keine Diskette eingelegt}
473         IF taste = #27 THEN
474           BEGIN
475             Close(datei);
476             Release(heap);
477             IF tabelle THEN laufwerkstabalt;
478             IF NOT batch THEN Exit ELSE Halt(1);
479           END;
480         UNTIL fehler <> $80;      {Abbruch, wenn Disk ready}
481         IF ((dspur <> anzspur+1) OR (dseite <> anzseite+1) OR
482            (dsektor <> anzsektor) OR (fehler <> 0)) AND NOT batch THEN
483           BEGIN
484             Write('Ungleiches Zielformat : ');
485             WriteLn((dspur*dsektor*dseite*512) DIV 1024, ' KByte');
486             IF compare THEN
487               Write('Doch Vergleichen ? (J/N) ');
488             ELSE Write('Doch Formatieren ? (J/N) ');
489             taste := Ucase(readkey);
490             delline;
491             IF (taste = #27) OR (taste = 'N') THEN
492               BEGIN
493                 Close(datei);
494                 Release(heap);
495                 IF tabelle THEN laufwerkstabalt;
496                 IF NOT batch THEN Exit ELSE Halt(1);
497               END;
498             IF NOT compare AND (taste = 'J') THEN formchange;
499             {Diskcopy mit formatieren}
500           END;
501         END;
502       schreibrate(arttarget, kap, target);
503       IF NOT tabelle THEN
504         BEGIN
505           laufwerkstabneu;      {Noch keine neue DDPT}
506           tabelle := True;      {Neue DDPT eingerichtet}
507         END;
508       IF compare THEN New(destpuffer);
```

```
509 {Diskettendaten vergleichen}
510 abssektor := 0; sekread := 0;
511 FOR spur := 0 TO anzspur DO
512 BEGIN
513   FOR seite := 0 TO anzseite DO
514   BEGIN
515     IF abssektor = 0 THEN
516     BEGIN
517       anzahl := (127 DIV anzsektor)*anzsektor;
518       BlockRead(datei, schreibpuffer^[0, 0], anzahl, dummy);
519     END;
520   IF NOT compare THEN
521   BEGIN
522     IF NOT batch THEN gotoxy(1, 12) ELSE gotoxy(1, wherey);
523     Write('Schreibe Seite : ', seite, ' Spur : ', spur);
524     Write(' '); {Andere Meldung überschreiben}
525     REPEAT
526       taste := #0; {Tastaturcode löschen}
527       IF (fehler <> 0) AND (fehler <> $80)
528       AND NOT format THEN formchange;
529       {Fehler beim Schreiben, nun doch formatieren}
530       IF format THEN
531         fehler := spurformat(spur, seite, 1,
532                               anzsektor, target);
533       Move(schreibpuffer^[abssektor, 0],
534            diskpuffer^[0, 0], anzsektor*512);
535       {Daten in Diskpuffer kopieren}
536       fehler := readwriteverify(3, spur, seite, 1,
537                                 anzsektor, target,
538                                 diskpuffer^[0, 0]);
539       {Diskpuffer auf Diskette schreiben}
540
541       IF verify THEN
542         fehler := readwriteverify(4, spur, seite, 1,
543                                   anzsektor, target,
544                                   diskpuffer^[0, 0]);
545       IF fehler = $80 THEN taste := dnotready;
546       IF fehler = 3 THEN {Diskette schreibgeschützt}
547       BEGIN
548         IF NOT batch THEN gotoxy(1, 14);
549         WriteLn('Diskette ist schreibgeschützt !');
550         Write('Bitte Schreibschutz entfernen, ');
551         WriteLn('weiter mit beliebiger Taste ');
552         taste := readkey;
553         IF taste = #27 THEN {Abbruch durch ESC}
554         BEGIN
555           ausgang(datei);
556           IF compare THEN Release(heap);
557           IF NOT batch THEN Exit ELSE Halt(1);
558         END;
559         IF NOT batch THEN {Fehlermeldung löschen}
560         BEGIN
561           window(1, 14, 80, 25);
562           clrscr;
563           window(1, 1, 80, 25);
564         END;
565         fehler := $80; {Schreiben noch einmal versuchen}
566       END;
567       IF fehler = 9 THEN New(diskpuffer);
568       {DMA-Übertragung über Segmentgrenze}
569       UNTIL ((fehler = 0) OR format) AND
570       (fehler <> $80) AND (fehler <> 9);
571     END
572   ELSE
573   BEGIN
574     REPEAT
575       vfehler := False;
576       {Vergleichsfehler aufgetreten ?}
577
```

```
578         IF NOT batch THEN gotoxy(1, 12)
579         ELSE gotoxy(1, wherey);
580         Write('Vergleiche Seite : ', seite, ' Spur : ', spur);
581         Write(' '); {Andere Meldung überschreiben}
582         fehler := readwriteverify(2, spur, seite, 1,
583                                   anzsektor, target,
584                                   destpuffer^[0, 0]);
585         IF fehler = $80 THEN taste := dnotready;
586         IF fehler = 9 THEN New(destpuffer);
587         {DMA-Übertragung über 64 KB Grenze hinaus}
588         UNTIL (fehler <> $80) AND (fehler <> 9);
589         {Abbruch, wenn Laufwerk bereit}
590         vfehler := vergleich(schreibpuffer^[abssektor, 0],
591                             destpuffer^[0, 0], anzsektor);
592         {Daten der Massenspeicher vergleichen}
593         END;
594         IF (fehler <> 0) OR vfehler THEN
595             BEGIN
596                 IF NOT batch THEN gotoxy(1, 14);
597                 tdisk := True; {Fehler bei Zieldisk}
598                 WriteLn(' Fehler Seite : ', seite, ' Spur : ', spur);
599             END;
600         Inc(abssektor, anzsektor); Inc(sekread, anzsektor);
601         IF abssektor = anzahl THEN abssektor := 0;
602         IF keypressed THEN {wurde eine Taste betätigt ?}
603             BEGIN
604                 taste := readkey;
605                 IF taste = #27 THEN {Abbruch durch ESC}
606                     BEGIN
607                         ausgang(datei);
608                         IF compare THEN Release(heap);
609                         IF NOT batch THEN Exit ELSE Halt(1);
610                     END;
611                 END;
612             END;
613         END;
614         IF NOT batch THEN
615             BEGIN
616                 gotoxy(1, 12);
617                 IF sdisk OR tdisk OR vfehler THEN
618                     WriteLn('Kopie der Diskette evt. unbrauchbar !')
619                 ELSE
620                     BEGIN
621                         IF NOT compare THEN Write('Diskcopy konnte ')
622                         ELSE Write('Vergleich konnte ');
623                         WriteLn('fehlerfrei durchgeführt werden');
624                     END;
625                 delline;
626                 Write('Weitere Kopie oder weiterer Vergleich ');
627                 Write('(J/N) ');
628                 taste := Ucase(readkey);
629                 window(1, 12, 80, 25);
630                 clrscr;
631                 window(1, 1, 80, 25);
632             END;
633         UNTIL (taste <> 'J') OR batch;
634         END;
635         ausgang(datei);
636         IF compare THEN Release(heap); WriteLn;
637     END;
638     {*****Hauptprogramm*****}
639 BEGIN
640     lowvideo;
641     checkbreak := False; {Kein Abbruch durch CTRL-Break}
642     IF ParamCount < 1 THEN batch := False ELSE batch := True;
643     {Programmaufruf mit oder ohne Kommandozeile}
644     source := 0; target := 0; {Standarddiskcopy von A nach A}
645     dateiname := 'dcc.bac'; {Name der Massenspeicherdatei}
646     format := False; verify := False; {Kein Format, kein Verify}
```

```
647 compare := False; {Keine Disketten vergleichen}
648 manuell := False; {Manuelle Parameter}
649 prompt := True; aktion := ''; {Diskettenwechsel bestätigen}
650 fparam := False; {Fehler in Kommandozeile}
651 IF batch THEN
652 BEGIN
653 aktion := ParamStr(1); {Read oder Write eines Imagesfiles}
654 aktion := Ucase(aktion[1]);
655 laufwstr := ParamStr(2); {Ausgewähltes Laufwerk}
656 IF aktion = 'R' THEN
657 BEGIN
658 source := Ord(Ucase(laufwstr[1]))-65;
659 taste := 'D'; {Bei Read Diskcopy ausführen}
660 END
661 ELSE IF aktion = 'W' THEN target := Ord(Ucase(laufwstr[1]))-65
662 ELSE fparam := True; {Fehler bei Parameter}
663 IF (source > 1) OR (target > 1) THEN fparam := True; {Fehler}
664 IF ((source = 0) OR (target = 0)) AND (laufwerka = 0)
665 THEN fparam := True; {Fehler, kein Laufwerk}
666 IF ((source = 1) OR (target = 1)) AND (laufwerkb = 0)
667 THEN fparam := True; {Fehler, kein Laufwerk}
668 dateiname := ParamStr(3);
669 IF dateiname = '' THEN
670 BEGIN
671 fparam := True; {Fehler, kein Dateiname}
672 dateiname := 'dcc.bac';
673 {Name der Massenspeicherdatei}
674 END;
675 diskprompt := ParamStr(4);
676 diskprompt := Ucase(diskprompt[1]);
677 IF diskprompt = 'N' THEN prompt := False ELSE prompt := True;
678 diskcomp := ParamStr(5);
679 diskcomp := Ucase(diskcomp[1]);
680 IF NOT(aktion = 'R') THEN {optionaler Parameter bei Read}
681 BEGIN
682 IF diskcomp = 'D' THEN compare := False {Diskcopy}
683 ELSE IF diskcomp = 'C' THEN compare := True
684 ELSE fparam := True; {Falscher Parameter}
685 IF compare THEN taste := 'C' ELSE taste := 'D';
686 END; {Wahl von Diskcopy oder Compare}
687 IF fparam THEN batch := False; {Fehler bei Parametern}
688 END;
689 REPEAT
690 IF NOT batch THEN
691 BEGIN
692 clrscr;
693 IF fparam THEN
694 BEGIN
695 gotoxy(1, 16);
696 Write('Fehler in den Kommandozeilenparametern, ');
697 WriteLn('oder falsches Laufwerk');
698 Write('Korrektur lautet : ');
699 WriteLn('DCC R/W Q/Z dat P/N [D/C]');
700 WriteLn('R/W : Read bzw. Write eines Imagesfiles');
701 WriteLn('Q/Z : Quell- bzw. Ziel-Laufwerk');
702 WriteLn('dat : Name der Massenspeicherdatei');
703 WriteLn('P/N : Prompting / No Prompting des Diskwechsels');
704 WriteLn('D/C : Anzugeben bei Write. Diskcopy oder Compare');
705 END;
706 gotoxy(35, 1); Write('Hauptmenü : ');
707 gotoxy(1, 2); Write('(Q)uell-Laufwerk : ');
708 IF source = 0 THEN WriteLn('A') ELSE WriteLn('B');
709 Write('(Z)iel-Laufwerk : ');
710 IF target = 0 THEN WriteLn('A') ELSE WriteLn('B');
711 Write('(F)ormat : ');
712 IF format THEN WriteLn('ON') ELSE WriteLn('OFF');
713 Write('(V)erify : ');
714 IF verify THEN WriteLn('ON') ELSE WriteLn('OFF');
715 Write('Datei-(N)ame [Pfad] : '); WriteLn(dateiname);
```

```
716     IF manuell THEN
717         BEGIN
718             highvideo;                                {High wenn Manuellbetrieb}
719             diskettenformat(trk, side, sek);           {Manuellformat ausgeben}
720             WriteLn(' (M)anuelle Parametereingabe : ON ');
721             lowvideo;
722         END
723     ELSE WriteLn(' (M)anuelle Parametereingabe : OFF ');
724     WriteLn(' (C)ompare Disketten ');
725     Write(' (D)iskcopy ');
726     taste := Uppcase(readkey);
727     IF fparam AND NOT(taste = #27) THEN
728         BEGIN
729             window(1, 16, 80, 25); clrscr; window(1, 1, 80, 25);
730             fparam := False;                           {Kein Batching mehr}
731         END;
732     END;
733     CASE taste OF
734         'Q' : IF (source = 0) AND (laufwerkb <> 0) THEN
735             source := 1 ELSE source := 0;
736         'Z' : IF (target = 0) AND (laufwerkb <> 0) THEN
737             target := 1 ELSE target := 0;
738         'F' : IF format THEN format := False ELSE format := True;
739         'V' : IF verify THEN verify := False ELSE verify := True;
740         'N' : BEGIN
741             gotoxy(15, 12); Write('Name der Datei : ');
742             ReadLn(datdmy);
743             IF datdmy <> '' THEN dateiname := datdmy;
744             {Abfangen eines Leerstrings}
745         END;
746         'M' : BEGIN
747             IF manuell THEN manuell := False
748             ELSE
749                 BEGIN
750                     manuell := True;                    {Manuelle Parametereingabe}
751                     parameter;                          {Parametermenü}
752                     IF taste = #27 THEN
753                         BEGIN
754                             taste := #0;
755                             manuell := False;           {Abbruch mit ESC}
756                         END;
757                     END;
758             END;
759         'C' : BEGIN
760             compare := True;                            {Disketten vergleichen}
761             diskcopy;                                   {Start des Vergleiches}
762             compare := False;
763         END;
764         'D' : diskcopy;                                {Start des Diskcopyprogramms}
765     END;
766     UNTIL (taste = #27) OR batch;
767     IF batch AND (sdisk OR tdisk OR vfehler) THEN Halt(1); {Batchfehler}
768     IF NOT batch AND NOT fparam THEN clrscr;
769     IF fparam THEN gotoxy(1, 23);
770 END.
```

Zu diesem Artikel existieren Programmbeispiele

[0490_406.doc](#)

[0490_406.zip](#)