COMPY-SHOP MAGAZIN INSIDE BIBO-DOS von P.Bee und E.Reuß

Innerhalb sehr kurzer Zeit hat sich das BIBO-DOS zu einem Standard bei den ATARI Usern entwickelt. Aufgrund seiner leichten Bedienbarkeit, der Benutzerfreundlichkeit und nicht zuletzt weil jeder es nach seinem Bedarf konfigurieren kann.

In einer Serie von Artikeln, die sich speziell mit dem BIBO-DOS befassen, werden wir Ihnen die Möglichkeiten und verschiedene Anwendungen für das BIBO-DOS aufzeigen.

Die ersten Teile unserer Serie wenden sich speziell an die Anfänger unter unseren Lesern. Zuerst werden wir einmal die verschiedenen Formate behandeln (SINGLE-, MEDIUM- und DOUBLE-DENSITY).
Sie werden erfahren wie eine Directory aufgebaut ist, was eine VTOC ist und wie das BIBO-DOS diese benutzt. Was passiert beim abspeichern einer Datei? Und was beim laden?

Auf alle diese Fragen werden Sie in diesem Text eine Antwort bekommen. Und schließlich werden wir auch auf diejenigen von Ihnen eingehen, die das BIBO-DOS gerne ändern möchten. Sei es, daß Sie eine eigene Ramdisk entwickelt haben, oder daß Sie ein eigenes DUP.SYS schreiben möchten.

Hier also eine Generelle Einführung in das Betriebssystem. (DOS = Disk Operating System - zu deutsch = Disketten Betriebssystem)

DIE DISKETTE

Um mit einer Diskette arbeiten zu können, müssen Sie zuerst einmal ein Format auf diese Diskette aufbringen, mit dem das DOS arbeiten kann. Um nun die Zusammenhänge richtig zu verstehen, betrachten wir uns zuerst einmal den Aufbau einer Diskette. Unser ATARI 1050 Laufwerk kann Mini-Disketten im 5 1/4 Zoll Format verarbeiten. Wie diese 5 1/4 Zoll Diskette augebaut ist, können Sie auf der nachfolgenden Zeichnung deutlich erkennen.

Eine Diskette hat eine äußere Schutzhülle aus Pappe oder Plastik. Die Schutzhülle umgibt eine flexible Plastikscheibe, die eigentliche Diskette. Sie ist bei einer MD-1D Diskette einseitig mit einer Magnetischen Schicht überzogen. Bei Doppelseitig nutzbaren Disketten (MD-2D) sind beide Seiten mit dieser Schicht überzogen.

Anmerkung: Normalerweise sind sowohl MD-1D als auch MD-2D Disketten beidseitig mit einer magnetischen Schicht überzogen. Aber nur bei den MD-2D Disketten wird auch die Rückseite auf Datensicherheit geprüft.

Zwischen der Schutzhülle und der Plastikscheibe befindet sich noch ein weiches Flies. Das dient dem Schutz der Diskette. Denn wird die empfindliche magnetische Oberfläche der Diskette beschädigt, sind alle Daten verloren und Sie können die Diskette wegwerfen.

Zwei Sachen sind noch sehr wichtig an einer Diskette. Das erste ist der Ausschnitt für den Schreib/Lesekopf, das zweite die Schreibschutzkerbe. Der Ausschnitt für den Schreib/Lesekopf befindet sich zwar immer auf beiden Seiten der Diskette, bei der 1050 wird aber nur die Unterseite der Diskette benutzt. Dieses hier also als Warnung für alle die Ihre Diskette einfach auf den Schreibtisch legen. Die wichtige Seite einer Diskette ist die Unterseite!

Die Schreibschutzkerbe dient der Datensicherheit. Kleben Sie diese Kerbe zu, kann die Diskette nicht mehr beschrieben werden. Die Abfrage dieser Kerbe geschieht im Laufwerk durch eine Lichtschranke.

Als letztes gibt es dann noch das Indexloch, ziemlich in der Mitte der Diskette. Manche Laufwerke fragen dieses Indexloch durch eine Lichtschranke ab um den Trackanfang festzustellen. Nicht so die ATARI Laufwerke.

Das hat auch Vorteile. So können Sie ohne Probleme die Rückseiten der Disketten nutzen und brauchen nicht die teureren "Two Eye" Disketten mit zwei Indexlöchern zu kaufen.

DIE FORMATE

Aber bevor Sie mit einer Diskette arbeiten können, müssen Sie sie erst einmal mit einem für Ihr DOS lesbaren Format versehen. Dazu verfügt jedes DOS über den FORMAT Befehl.

Nun gibt es die verschiedensten DOS-Arten, und die unterschiedlichsten Formate. Wir werden Ihnen hier die beiden Standard Formate beschreiben. Diese Formate werden als das DOS 2.0 und das DOS 2.5 Format bezeichnet. Auch das BIBO-DOS arbeitet mit diesen beiden
Formaten.

DAS DOS 2.0 FORMAT

Das DOS 2.0 Format, oder Single Density, ist das Standard Format für die ATARI Laufwerke. Das DOS 2.0 wurde bereits 1980 von Atari eingeführt. Geschrieben wurde dieses DOS von der Firma OSS, die auch das BASIC für Atari geschrieben hat. Gedacht war dieses DOS für das Atari Laufwerk 810. Dieses Laufwerk konnte eine Diskette mit maximal 88K-Byte Kapazität formatieren. Wie wird aber nun dieses Format auf die Diskette aufgebracht, und wodurch ergibt sich die Speicherkapazität?

Wenn Sie eine Diskette in das ATARI 1050 Laufwerk einlegen und lassen das DOS diese Diskette formatieren, können Sie hören, das der Steppermotor des Laufwerkes den Schreib/Lesekopf 80 mal bewegt. Anschließend werden vom DOS wichtige Daten auf die frisch formatierte Diskette geschrieben. Was passiert aber nun im einzelnen?

Sicher kennen Sie die Scherzfrage: Wieviele Spuren sind auf einer Schallplattenseite? und die Antwort: Eine!

Bei einer Diskette ist es fast genauso. Nur mit dem Unterschied das die Diskette nicht eine sondern 40 Spuren hat. Bei einer Schallplatte wird die Spur spiralförmig aufgebracht, bei einer Diskette konzentrisch. Das bedeutet, daß 40 Kreise auf die Diskette aufgebracht werden, und daß die Kreise zur Mitte der Diskette hin immer enger werden.

Dabei werden diese Kreise (Spuren, Tracks) in 18 Teilkreise (Sektoren) unterteilt. Jetzt werden Sie aber sicherlich fragen, warum 80 Schritte bei 40 Spuren?

Die Antwort liegt in der Datensicherheit. Das Laufwerk geht zweimal über jede Spur. Beim erstenmal wird das Format aufgebracht, beim zweitenmal ein Verify ausgeführt. Das bedeutet, daß das eben aufgebrachte Format Überprüft wird. Sollten sich bei dieser Überprüfung Fehler feststellen lassen, so wird noch einmal formatiert.

Wie bereits gesagt, werden die 40 Spuren in 18 Sektoren unterteilt. Jeder dieser Sektoren hat nun eine Speicherkapazität von 128 Bytes. Das ergibt also auf einer Diskettenseite einen Speicherplatz von (40*18*128=) 92.160 Bytes, oder 92K-Bytes. Errechnen wir die Anzahl der Sektoren auf einer Diskettenseite, so erhalten wir (40*18=) 720 Sektoren. Wie wir gleich sehen werden, müssen wir aber von diesen Zahlen einiges abziehen, da das DOS einigen Speicherplatz für die Datenverwaltung benötigt. Und es gibt einen Sektor, den das DOS 2.0 nicht bearbeiten kann. Das ist der letzte Sektor, Nummer 720.

Nach dem Formatieren bringt das DOS noch einige sehr wichtige Daten auf die Diskette. Zuerst wird in die ersten drei Sektoren eine Laderoutine für das DOS geschrieben. Diese benötigen Sie, wenn Sie später das DOS von dieser Diskette booten wollen. Booten Sie diese Diskette ohne das sich das DOS darauf befindet, erhalten Sie den berühmten "BOOT ERROR".

Als zweites richtet das DOS die VTOC ein. Die VTOC ist eine Speicherbelegungstabelle. Anhand dieser Tabelle kann das DOS jederzeit feststellen, wieviel freier Speicherplatz auf der Diskette noch verfügbar ist, und wo auf der Diskette der noch freie Speicherplatz zu finden ist. Diese VTOC liegt immer im Sektor $0168, Dezimal 360.

Als drittes wird nun noch die Directory eingerichtet. Die Directory ist das Inhaltsverzeichnis der Diskette. Anhand der hier befindlichen Angaben kann das DOS jede Datei auf der Diskette wiederfinden. Das DOS 2.0 legt diese Directory direkt hinter die VTOC, also beginnend bei dem Sektor $0169, Dezimal 361. Die Directory hat eine Länge von 8 Sektoren.

Ziehen wir nun alle die Sektoren, die das DOS belegt (12 + 1 den das DOS nicht bearbeiten kann), von der Anzahl der maximal verfügbaren Sektoren (720) ab, so erhalten wir 707 frei verfügbare Sektoren. Wir sagten bereits, daß jeder Sektor 128 Bytes enthalten kann. Aber auch hier macht das DOS abstriche. Drei Bytes werden vom DOS für die Dateiverwaltung benötigt. Das sind die sogenannten SECTOR-LINKS. Verbleiben also für den Anwender 707 Sektoren * 125 Bytes = 88.500 Bytes oder 88K-Bytes. Damit haben Sie das DOS 2.0 Format.

DAS DOS 2.5 FORMAT

Da beide DOS Versionen praktisch identisch sind, brauchen wir hier weniger Platz für die Eklärung.

Mit Einführung des Atari 1050 Laufwerkes gab es dann auch eine neue Speicherkapazität. Das ist das sogenannte DOS 2.5 Format, oder Medium Density. Die Unterschiede zum DOS 2.0 Format liegen in der Speicherkapazität.

Das DOS 2.5 Format verfügt über 26 Sektoren pro Spur, also 8 Sektoren mehr als das DOS 2.0 Format. Das ergibt einen zusätzlichen Speicherplatz von (8*40*125/1024=39.062) 39K-Byte.

Um nun diese zusätzlichen 39K-Byte zu verwalten, benutzt das DOS 2.5 eine zweite VTOC. Das macht das DOS 2.5 teilweise inkompatibel zum DOS 2.0. Denn die Dateien, die in der zweiten VTOC geführt werden, können mit dem DOS 2.0 nicht gelesen werden. Da das BIBO-DOS aber zu beiden DOS Versionen kompatibel ist, haben Sie da keine Schwierigkeiten.

Die zweite VTOC des DOS 2.5 liegt im Sektor $0400, Dezimal 1024. Daß dieser Sektor für die zweite VTOC benutzt wird, liegt daran, daß das DOS 2.0 und das DOS 2.5 bei der Sectorlink Verwaltung als höchste Sektorenzahl $03FF, Dezimal 1023, verwalten kann. Theoretisch bleiben bei diesem Format die Sektoren $0401 bis $0410, Dezimal 1025 bis 1040, ungenutzt. Dateien, die in dieser zweiten VTOC geführt werden, werden durch ein spezielles Statusbyte in der Directory gekennzeichnet. Dazu später mehr.

DAS BIBO-DOS FORMAT

Das BIBO-DOS Format (Double Density) ist mit dem DOS 2.0 Format (Single Density) praktisch identisch. Die Unterschiede:

In einem Sektor werden nicht 128 Bytes sondern 256 Bytes abgespeichert. Sie können also doppelt soviel Informationen auf einer Diskettenseite unterbringen. Auch hier belegt das DOS die letzten drei Bytes im Sektor für die Dateiverwaltung. Es werden auch exakt die gleichen Sektoren für die VTOC und die Directory belegt. Hier ergibt sich also eine für den Anwender nutzbare Speicherkapazität von 708 Sektoren * 253 Bytes = 179.124 Bytes oder 180K-Byte.

Soviel zu den drei Formaten. Es ist bereits viel von der Directory und der VTOC gesprochen worden. In unserem nächsten Artikel werden wir uns damit befassen.

Im folgenden geht es um die Directory, die VTOC und die Sektorlinks.

DIE DIRECTORY

Die Directory ist das Inhaltsverzeichnis der Diskette. Anhand der Directory findet das DOS jede Datei auf der Diskette wieder. Wie sieht eine Directory aus?

Bei einer Diskette, die wir mit dem BIBO-DOS formatiert haben, liegt die Directory in den Sektoren $0169 bis $0170, Dezimal 361 bis 368. Jeder Eintrag in diese Directory belegt 16 Bytes. Daraus ergibt sich auch die Maximal mögliche Anzahl der Directory Einträge: 8 Sektoren * 128 Bytes / 16 Bytes per Eintrag = 64 mögliche Einträge. Mit Hilfe eines Diskettenmonitors wie dem BIBOMON können wir uns einzelne Sektoren auf der Diskette ansehen. Lesen wir also mal den ersten Directory-Sektor ein. Er hat die Nummer $0169, Dezimal 361. Da wir den Sektor in den Arbeitsspeicher des Computers eingelesen haben, erscheint vor dem eigentlichen Sektorinhalt der Speicherbereich des Computers. Getrennt werden die Angaben durch einen Doppelpunkt. Auf der linken Seite der Tabelle sehen Sie die hexadezimalen Angaben, rechts die entsprechenden ASCII Zeichen.

```
1000:42 21 00 04 00 54 45 53  B!   TES
1008:54 20 20 20 20 43 4F 4D  T    BAS
1010:00 00 00 00 00 00 00 00
1018:00 00 00 00 00 00 00 00
1020:00 00 00 00 00 00 00 00
1028:00 00 00 00 00 00 00 00
1030:00 00 00 00 00 00 00 00
1038:00 00 00 00 00 00 00 00
1040:00 00 00 00 00 00 00 00
1048:00 00 00 00 00 00 00 00
1050:00 00 00 00 00 00 00 00
1058:00 00 00 00 00 00 00 00
1060:00 00 00 00 00 00 00 00
1068:00 00 00 00 00 00 00 00
1070:00 00 00 00 00 00 00 00
1078:00 00 00 00 00 00 00 00
```

Was wir jetzt hier im einzelnen sehen, dazu später mehr. Zuerst einmal, hätten wir uns die Diskette mit dem BIBO-DOS Befehl A. DISK INHALT angesehen, so hätte das DOS mit den folgenden Angaben geantwortet:

```
  TEST    BAS 033
674 FREE SECTORS
```

Wie kommt nun das DOS zu diesen Angaben? Sehen wir uns die ersten drei Zeilen der Directory einmal genauer an.

```
1000:42 21 00 04 00 54 45 53  B!   TES
1008:54 20 20 20 20 43 4F 4D  T    BAS
1010:00 00 00 00 00 00 00 00
```

Da nur eine Datei auf der Diskette gespeichert ist, werden nur die ersten 16 Bytes belegt. 11 dieser 16 Bytes werden für den Dateinamen reserviert. 8 Bytes für den Namen, 3 für den Extender. Verbleiben noch 5 Bytes mit sehr wichtigen

Informationen. Aber gehen wir der Reihe nach vor. Gleich das erste Byte enthält eine sehr wichtige Information. Es ist das Statusbyte. Anhand dieses Bytes kann das DOS feststellen, ob die Datei gesichert ist, geöffnet oder gelöscht.
Hier die Daten:

$42 - Normalzustand einer Datei. Wird vom DOS bei der Directory Abfrage nicht extra angezeigt.
$62 - Datei ist gegen Überschreiben gesichert. Das DOS kann diese Datei nicht löschen. Dieser Status wird durch ein Sternchen vor dem Dateinamen bei der Directory Abfrage angezeigt.
$80 - Datei gelöscht. Diesen Statuswert finden wir bei gelöschten Dateien vor. Weder die Datei noch der Statuswert werden vom DOS bei der Directory Abfrage angezeigt. Erst bei der speziellen Directory Abfrage des BIBO-DOS werden diese Dateien mit einem Gleichheitszeichen (=) vor dem Namen angezeigt. Diese Dateien können mit der BIBO-DOS Funktion J wieder aktiviert werden.
$43 - Geöffnete Datei. Diesen Status finden wir, wenn ein Programm eine Datendatei zum bearbeiten geöffnet hat, es dann aber aus irgend einem Grund nicht wieder geschlossen hat. Diese Dateien sind nicht mehr zu reparieren, da sehr wahrscheinlich auch die VTOC geändert wurde, diese Änderungen aber nicht eingetragen wurden. Hier kopiert man am besten alle noch lesbaren Dateien auf eine andere Diskette und formatiert diese defekte Diskette. Geöffnete Dateien werden bei der normalen Directory Abfrage gar nicht dargestellt. Bei der speziellen Abfrage des BIBO-DOS werden geöffnete Dateien mit einem Fragezeichen vor dem Dateinamen dargestellt.
$00 - Ende der Directory. Das DOS liest solange die Directory Sektoren ein, bis es an das Ende der Directory angekommen ist, oder bis der Statuswert auf 0 steht. Alle Informationen vor der Ende Kennung werden angezeigt.

Statuswerte, die nur im DOS 2.5 Format benutzt werden:

$03 - Normaler Status - DOS 2.5 Format. Entspricht dem Statuswert $42. Wird benutzt, wenn mindestens 1 Sektor der Datei in der erweiterten VTOC geführt wird. Diese Dateien werden bei der Directory Abfrage durch die größer als/kleiner als Zeichen (<>) gekennzeichnet.
$23 - Datei gegen Überschreiben gesichert - DOS 2.5 Format. Entspricht dem Statuswert $62. Wird benutzt, wenn mindestens 1 Sektor der Datei in der erweiterten VTOC geführt wird. Diese Dateien werden bei der Directory Abfrage durch die größer als/kleiner als Zeichen (<>) gekennzeichnet.

Wie Sie sehen, erhalten Sie also bereits aus dem ersten Byte eines Directory Eintrages sehr wichtige Informationen über den Zustand Ihrer Datei!

Die Bytes zwei und drei geben Ihnen dann die Länge Ihrer Datei an. In unserem Beispiel finden wir dort die Einträge:

```
1000:42 21 00 04 00 54 45 53  B!   TES
1008:54 20 20 20 20 43 4F 4D  T    BAS
1010:00 00 00 00 00 00 00 00
```

Hexadezimal $21 und $00. Da diese Daten im Low-High Format abgelegt sind, müssen wir Sie zuerst einmal ausrechnen. Hexadezimal ist das sehr einfach. $21 und $00 im Low-High Format ergibt $0021 (Dezimal 33). Das ist die Länge unserer Datei in Sektoren.

Die darauf folgenden Bytes 4 und 5 geben dann den Startsektor unserer Datei auf der Diskette an. Auch hier haben wir das Low-High Format. Also, $04 und $00 ergibt $0004 (Dezimal 4). Schon wissen wir, wo der erste Sektor unserer Datei zu finden ist.

Die restlichen 11 Bytes geben den von Ihnen eingegebenen Namen der Datei wieder.

Soviel zu der Directory. Eigentlich müßten wir Ihnen jetzt die Funktion der VTOC erklären. Aber wir haben bereits mehrmals, speziell im Zusammenhang mit der Directory, von den Sektor-Links gesprochen. Also was genau ist ein Sektor-Link und wozu braucht das DOS ihn?

DIE SECTORLINKS

Anhand der Directory kann das DOS eine Datei auf der Diskette finden. Auch der erste Sektor ist kein Problem. Aber woher weiß das DOS, wo die übrigen 30 Sektoren unseres Beispielprogrammes liegen?

Die Antwort liegt in den Sectorlinks. Diese Sectorlinks (Link = Kettenglied) werden vom DOS beim abspeichern der Datei auf der Diskette angelegt. Wir finden die Sectorlinks in den letzten drei Bytes eines Sektors, also in den Bytes 126, 127 und 128. Aus den ersten beiden Bytes 126 und 127 kann das DOS die Nummer des nächsten zu lesenden Sektors für diese Datei errechnen. Dazu werden vom Byte 127 alle 8 Bits benutzt, vom Byte 126 aber nur die untersten 2. Hier ist die Information im High/Low Format abgelegt. Die beiden unteren Bits des Byte 126 reichen aus, da wir ja maximal nur $03FF Sektoren verwalten können. Sind nun im Byte 127 alle 8 Bits gesetzt, also $FF, und im Byte 126 die unteren beiden Bits, also $03, so hat der nächste zu lesende Sektor die Nummer $03FF. Die oberen 6 Bits des 126ten Byte stellen die Nummer der Datei in der Directory dar. Anhand dieser Nummer überprüft das DOS beim laden, ob der einzulesende Sektor mit der einzulesenden Datei übereinstimmt, bzw. ob der Sektor zu der Datei gehört, die das DOS einlesen soll. Ergibt sich nun aus den unteren beiden Bits des 126ten Byte und dem 127ten Byte der Sektor 0 als nächster zu lesender Sektor, so weiß das DOS, daß es das Ende der Datei erreicht hat.

Das letzte, das 128te Byte im Sektor gibt dem DOS die Anzahl der zu lesenden Byte im Sektor an. Gewöhnlich steht dieser Wert auf $7D, Dezimal 125. Es kann jedoch passieren, am Ende einer Datei, oder falls die Datei durch einen APPEND (Anhängen an eine vorhandene Datei beim Kopieren) verlängert wurde, daß dieser Wert kleiner ist als 125. Es kann auch schon einmal vorkommen, daß dieser Wert mitten in der Datei viel kleiner erscheint.

Nach diesem Ausflug zu den Sektorlinks, kommen wir nun zu der VTOC. Die Belegungstabelle (VTOC) ist schwer zu erklären. Wir werden daher ein paar Tabellen zur Unterstüzung benutzen.

DIE VTOC

Die VTOC (VOLUME TABLE OF CONTENTS) ist die Belegungstabelle der Diskette. Um ihre Funktion richtig zu verstehen, lesen wir eine VTOC in den Arbeitsspeicher unseres Computers. Das machen wir wieder mit der Hilfe eines Diskettenmonitors, wie dem BIBOMON. Lesen wir also den VTOC-Sektor ein. Er hat die Nummer $0168, Dezimal 360. Da wir den Sektor in den Arbeitsspeicher des Computers eingelesen haben, erscheint vor dem eigentlichen Sektorinhalt der Speicherbereich des Computers. Getrennt werden die Angaben durch einen Doppelpunkt.

```
1000:02 C3 02 C3 02 00 00 00
1008:00 00 0F FF FF FF FF FF
1010:FF FF FF FF FF FF FF FF
1018:FF FF FF FF FF FF FF FF
1020:FF FF FF FF FF FF FF FF
1028:FF FF FF FF FF FF FF FF
1030:FF FF FF FF FF FF FF 00
1038:7F FF FF FF FF FF FF FF
1040:FF FF FF FF FF FF FF FF
1048:FF FF FF FF FF FF FF FF
1050:FF FF FF FF FF FF FF FF
1058:FF FF FF FF FF FF FF FF
1060:FF FF FF FF FF FF FF FF
1068:FF FF FF FF FF FF FF FF
1070:FF FF FF FF 00 00 00 00
1078:00 00 00 00 00 00 00 00
```

Was Sie hier sehen, ist die VTOC einer frisch formatierten Diskette. Das DOS erkennt anhand dieser VTOC wieviele Sektoren noch frei sind, und wo diese freien Sektoren zu finden sind. Aber gehen wir der Reihe nach vor und beginnen wir am Anfang, beim ersten Byte.

Byte 1, in unserem Falle $02, gibt die DOS Version an, mit der die Diskette formatiert wurde, also DOS 2. Die Bytes 2 und 3 geben die Anzahl der benutzbaren Sektoren auf dieser Diskette an. Abgelegt sind die Daten wieder im Low/ High Format, so daß wir hier also eine maximale Anzahl von $02C3, Dezimal 707, zur Verfügung haben.

Aus den Bytes 4 und 5 liest das DOS die Anzahl der noch freien Sektoren ab. Auch hier sind die Daten wieder im Low/ High Format abgelegt.

Nach den Werten für die Anzahl der freien Sektoren folgen fünf für das DOS unwichtige Bytes, die immer auf $00 stehen. Wichtig sind aber die darauf folgenden Bytes, beginnend mit Byte 11 und endend bei Byte 100. Auch die darauf folgenden Bytes 101 bis 128 sind für das DOS unwichtige Bytes.

Bei der Berechnung für die Belegungstabelle richtet sich das DOS nach der folgenden Formel:

Für 8 Sektoren werden 8 Bits eines Bytes benötigt. Da wir aber in einer Spur 18 Sektoren haben, braucht das DOS für die Darstellung einer Spur 2 Bytes und 2 Bits des darauf folgenden Bytes. Um Ihnen das ganze etwas einfacher darzustellen, sehen wir uns die folgende Tabelle einmal genauer an.

```
BYTE    ---1---- ---2---- ---3----

BIT     87654321 87654321 87654321

GESETZT 00001111 11111111 11111111

SEKTOR- 00000000 00111111 11100000
NUMMER  01234567 89012345 67812345

SPUR    ------- 1 ------- ---
```

Was Sie oben sehen ist die Darstellung der Belegungstabelle der ersten Spur einer frisch formatierten Diskette. Die dargestellten drei Bytes haben die VTOC Positionen 11, 12 und 13. Das bedeutet, Byte 1 dieser Tabelle entspricht Byte 11 der VTOC, Byte 2 der Tabelle entspricht Byte 12 der VTOC und Byte 3 der Tabelle entspricht Byte 13 der VTOC. Da das DOS nach dem Formatieren alle leeren Sektoren durch setzen eines Bits kennzeichnet, sind in den Bytes 12 und 13 alle Bits gesetzt, die Bytes haben also die Wertigkeit $FF.

Beschriebene Sektoren werden durch löschen eines Bits gekennzeichnet. Da das DOS in die ersten drei Sektoren der Diskette die Bootsektoren schreibt, sind diese schon einmal belegt. Also sind im ersten Byte 3 Bits als gelöscht eingetragen. Das Byte erscheint in der VTOC mit dem Wert $0F. Die erste Spur setzt sich also aus den kompletten Bytes 1 und 2 und den ersten 3 Bits des dritten Bytes unserer Tabelle zusammen.

Wenn Sie sich nun die VTOC noch einmal ansehen, werden Sie feststellen, daß mitten in der Tabelle ein Byte mit $00 und das darauf folgende mit $7F eingetragen sind. Diese beiden Bytes geben Ihnen die Lage der VTOC und der Directory auf der Diskette wieder. Wenn Sie wollen, können Sie nun die genaue Sektorzahl für die VTOC (die einen Sektor vor der Directory liegt) ausrechnen. Beachten Sie dabei bitte, daß immer 2 Bytes und 2 Bits zu einer kompletten Spur gehören, also insgesamt 18 Bits!

DIE ZWEITE VTOC DES DOS 2.5

Eine Besonderheit des DOS 2.5 wollen wir Ihnen hier noch etwas ausführlicher erklären. Das ist die zweite VTOC, die das DOS 2.5 im Sektor $0400 ablegt. Hier zuerst wieder eine Abbildung einer normalen VTOC des DOS 2.5 und der dazu gehörigen zweiten VTOC. Auch diese Diskette ist frisch formatiert.

NORMALE VTOC (SEKTOR $0168)

```
1000:02 C3 02 C3 02 00 00 00
1008:00 00 0F FF FF FF FF FF
1010:FF FF FF FF FF FF FF FF
1018:FF FF FF FF FF FF FF FF
1020:FF FF FF FF FF FF FF FF
1028:FF FF FF FF FF FF FF FF
1030:FF FF FF FF FF FF FF 00
1038:7F FF FF FF FF FF FF FF
1040:FF FF FF FF FF FF FF FF
1048:FF FF FF FF FF FF FF FF
1050:FF FF FF FF FF FF FF FF
1058:FF FF FF FF FF FF FF FF
1060:FF FF FF FF FF FF FF FF
1068:FF FF FF FF FF FF FF FF
1070:FF FF FF FF 00 00 00 00
1078:00 00 00 00 00 00 00 00
```

ZWEITE VTOC (SEKTOR $0400)

```
2000 FF FF FF FF FF FF FF FF
2008 FF FF FF FF FF FF FF FF
2010 FF FF FF FF FF FF FF FF
2018 FF FF FF FF FF FF FF FF
2020 FF FF FF FF FF FF FF 00
2028 7F FF FF FF FF FF FF FF
2030 FF FF FF FF FF FF FF FF
2038 FF FF FF FF FF FF FF FF
2040 FF FF FF FF FF FF FF FF
2048 FF FF FF FF FF FF FF FF
2050 FF FF FF FF FF FF FF FF
2058 FF FF FF FF FF FF FF FF
2060 FF FF FF FF FF FF FF FF
2068 FF FF FF FF FF FF FF FF
2070 FF FF FF FF FF FF FF FF
2078 FF FF 30 01 00 00 00 00
```

Wie Sie leicht erkennen können, ist ein Teil der zweiten VTOC identisch mit der ersten. Das ausrechnen der Sektorenzahl geschieht genauso wie in der ersten VTOC. Wichtig sind die Bytes 123 und 124. Bei unserer frisch formatierten Diskette stehen diese Bytes auf $01 und $30. Sie geben die zusätzliche Anzahl der freien Sektoren an, die mit Hilfe der zweiten VTOC verwaltet werden können. Da auch diese Daten als High/Low Daten abgelegt sind, können wir die Anzahl leicht ausrechnen. In unserem Fall also $0130 Sektoren, dezimal 304.

Soweit der zweite Teil unserer Serie über das BIBO-DOS. In der nächsten Ausgabe werden wir sehen, was alles passiert, wenn wir vom BASIC aus eine Datei laden oder abspeichern. Bis dahin viel Spaß mit dem BIBO-DOS!

- Da die XF551 nicht eigenständig zwischen Quad und Double unterscheiden kann, ist es notwendig, auf jeder Diskette die jeweilige Dichte zu verzeichnen. Dies wird durch ein Bit in der VTOC bewerkstelligt.

- Durch die erhöhte Sectorenzahl bei Quad-Density reicht die von DOS 2.x vorgesehene Kapazität der Linker-Bits (3) nicht mehr aus. BIBO-DOS benutzt einfach die Bits, die eigentlich für die Zuordnung des Sectors zu einem FIle bestimmt sind, ... Aus diesem Grund reserviert Turbo-DOS bei Quad-Density 4 (statt 3) Bytes eines Sectors für das FMS, damit keine Informationen verloren gehen.

Im folgenden geht es um das Abspeichern, das Löschen und das Laden einer Datei. Wie Sie sehen werden, muß das BIBO-DOS sehr viel Arbeit leisten, wenn Sie eine Datei abspeichern wollen.

DAS ABSPEICHERN EINER DATEI

Beim Abspeichern einer Datei muß das BIBO-DOS nicht nur die Daten der Datei auf die Diskette schreiben. Das BIBO-DOS muß die Directory und die VTOC der Diskette mit bearbeiten. Da wir Ihnen die VTOC ja bereits erklärt haben, wissen Sie, wie kompliziert die Berechnungen sind. Aber beginnen wir am Anfang.

Nehmen wir an, wir haben ein BASIC Programm geschrieben und wollen es nun abspeichern. Da es noch nicht ganz komplett ist, nennen wir es "TEST.BAS". Zum Abspeichern geben Sie vom Basic aus ein:

SAVE"D:TEST.BAS <RETURN>

Was passiert? Als erstes prüft das Betriebssystem den angesprochenen Gerätetreiber auf Gültigkeit. In unserem Fall ist das "D:". Wir hätten auch "D1:" schreiben können. Nötig ist die 1 nach dem "D" nicht, denn wird hinter dem Gerätekennzeichen keine Nummer eingegeben, setzt das Betriebssystem die 1 als gegeben voraus.

Das Zweite, was auf Gültigkeit geprüft wird, ist der Dateiname. Diese Arbei wird vom DOS erledigt. Dabei gilt eines zu Beachten: Das erste Zeichen des Dateinamens muß immer ein Alphanumerisches Zeichen sein. Das heißt, der Dateiname muß immer mit einem Buchstaben zwischen A und Z beginnen, und dieser Buchstabe muß groß geschrieben sein.

Die Länge des Dateinames ist auf 8 Zeichen festgelegt. Es kann noch ein Extender an den Namen angehängt werden, zur besseren Identifizierung der Dateien. Dieser Extender darf 3 Zeichen lang sein und wird vom Dateinamen durch einen Punkt getrennt. Hier eine Liste der gebräuchlichsten Extender und eine Kurzbeschreibung.

.BAS - Abkürzung für BASIC
.COM - Ausführbares Maschinenprogramm
.EXE - Ausführbares Maschinenprogramm
.OBJ - Objektdatei, zumeist Datendatei
.SYS - DOS - Systemdateien
.PAS - PASCAL Sourcecode Dateien
.TXT - Textdateien, zumeist reine ASCII
Dateien
.DOC - Dokumentendateien für Textver-
arbeitungen
.LIB - Library Dateien - Runtime
Library's für Compilersprachen
.ASM - Assembler Source Dateien
.MAC - Macro Dateien für Macro-
Assembler
.INC - Include Dateien, Textver-
arbeitungen, Assembler, Compiler
.DAT - Datendateien

Usw.

In der Wahl des Extenders können Sie völlig frei entscheiden, oder ihn sogar weglassen. Denken Sie nur bitte daran, daß Sie eine Datei leichter identifizieren können, wenn diese einen Extender hat.

Hat Ihre Eingabe diese Prüfungen bestanden, kann das eigentliche abspeichern der Daten beginnen. Dazu wird vom BIBO-DOS erst einmal die Directory in den Computer eingelesen. Als nächstes sucht das BIBO-DOS die Directory Einträge nach einem Eintrag mit dem gleichen Namen ab. Wird ein Eintrag gefunden, muß

die bereits bestehende Datei erst einmal gelöscht werden. Das löschen einer Datei erklären wir etwas später.

Ist der Name noch nicht vorhanden, sucht das BIBO-DOS den nächsten freien Eintrag in der Directory. Das kann entweder ein leerer Eintrag sein, oder ein gelöschter Eintrag. Ist kein Eintrag mehr frei, erhalten Sie eine Fehlermeldung und der Speichervorgang wird abgebrochen.

Hat das BIBO-DOS aber einen freien Eintrag gefunden, wird dort der Dateiname eingetragen. Der Status der Datei wird auf "geöffnet" ($43) gesetzt und die Directory wieder auf die Diskette geschrieben. Dabei merkt sich das BIBO-DOS die Position der neuen Datei in der
Directory.

Und nun beginnen eine Reihe von Arbeiten, die sehr kompliziert sind.

Zuerst wird die VTOC eingeladen. Aus den Daten der VTOC erechnet das BIBO-DOS den ersten freien Sektor, trägt diesen als belegt in der Sektorliste ein und zieht den belegten Sektor von der Anzahl der freien Sektoren ab.

Anschließend werden die ersten 125 Bytes unseres Programmes in einen speziellen Buffer im Computer abgelegt.

Soweit, so gut. Nun muß das BIBO-DOS den nächsten freien Sektor aus der VTOC ausrechnen und als belegt eintragen. Die Anzahl der als belegt eingetragenen Sektoren wird dabei gleich von der Anzahl der verfügbaren Sektoren abgezogen.

Mit diesen Daten, der Dateinummer in der Directory und dem nächsten zu schreibenden Sektor, kann das DOS den Sektorlink ausrechnen und an die 125 Bytes des Programmes anhängen.

Erst jetzt kann der erste Sektor, oder besser der erste Datenblock, abgespeichert werden.

Anschließend werden die nächsten 125 Bytes des Programmes in den Buffer geschrieben.

Es folgt ein Test, ob das Ende des Programmes erreicht ist. Wenn nicht, wiederholt sich der Vorgang. Also erst den nächsten zu lesenden Sektor aus der VTOC ausrechnen, als belegt eintragen, einen Sektor von der Anzahl der freien Sektoren abziehen, Sektorlink ausrechnen und an die 125 Bytes anhängen, Sektor schreiben, u.s.w.

Hat das DOS aber das Ende des Programmes erreicht, wird der Rest der Daten in den Buffer geschrieben. Beim ausrechnen des Sektorlinks wird als nächster zu schreibender Sektor der Sektor $00 eingetragen. Nun kann der letzte Sektor abgespeichert werden.

Jetzt befinden sich alle Daten unseres Programmes auf der Diskette. Das DOS muß aber noch zwei sehr wichtige Arbeiten ausführen, bevor wir diese Daten wieder laden können.

Erstens muß die geänderte VTOC abgespeichert werden.

Zweitens ist unsere Datei in der Directory als geöffnet eingetragen. Also wird die Directory noch einmal eingelesen. Der Status der abgespeicherten Datei wird auf Normal ($42) gesetzt, der Startsektor und die Länge der Datei in Sektoren werden eingetragen. Anschließend wird die geänderte Directory wieder abgespeichert.

Mit diesem letzten Speichervorgang ist das Abspeichern unserer Datei beendet. Wie Sie gesehen haben, ist das Abspeichern einer Datei sehr kompliziert. Aber da unserer ATARI ja ein schneller Rechner ist, merken Sie von diesen ganzen Arbeiten nichts.

DAS LÖSCHEN EINER DATEI

Auch beim löschen muß das DOS den angesprochenen Gerätetreiber und den Namen der Datei auf Gültigkeit prüfen. Diesen Vorgang haben wir beim Abspeichern einer Datei ja schon erklärt, so daß wir gleich zum nächsten Punkt gehen.

Auch beim Löschen einer Datei muß zuerst die Directory eingelesen werden. Aus dem Directory Eintrag liest das DOS den ersten Sektor der Datei, die wir löschen wollen. Hat das DOS diese Sektornummer, wird die Directory wieder "vergessen".

Als nächstes wird nun die VTOC und der erste Sektor der Datei in den Computer eingelesen. Der Sektor wird in der VTOC gleich als frei eingetragen.

Anschließend wird der nächste zu lesende Sektor aus dem Sektorlink errechnet, eingelesen und als frei in der Sektortabelle eingetragen. Gleichzeitig mit dem Eintrag daß dieser Sektor frei ist, wird selbstverständlich die Anzahl der freien Sektoren um eins erhöht.

Diese Arbeit wird nun solange fortgesetzt, bis das Ende der Datei erreicht wird.

Ist der letzte von der Datei belegte Sektor in der VTOC als frei eingetragen worden, und die Anzahl der freien Sektoren aktualisiert worden, wird die VTOC abgespeichert.

Nun muß nur noch die Directory eingelesen und der Status der Datei als gelöscht ($80) eintragen werden.

Als letzte Arbeit wird die Directory wieder abgespeichert.

Sollte während des einlesens der Sektoren ein Lesefehler auftreten (defekter Sektor oder defekter Sektorlink, falsche Dateinummer), wird der Löschvorgang sofort unterbrochen und Sie erhalten eine Fehlermeldung. Die Datei kann dann nicht mehr gelöscht werden. In diesem Fall muß die Diskette neu formatiert werden.

DAS LADEN EINER DATEI

Gemessen am Abspeichern oder am Löschen einer Datei ist das Laden einer Datei eine einfache Sache, zumal die CIO Routine des Betriebssystemes dem BIBO-DOS Arbeit abnimmt. Wobei es sogar egal ist, ob wir ein BASIC Programm, eine Datendatei oder ein Maschinensprachprogramm laden.

Auch beim laden wird der angesprochene Gerätetreiber und der Dateiname auf Gültigkeit geprüft. Ist diese Hürde erfolgreich genommen worden, wird die Directory in den Computer eingelesen und die Datei in der Directory gesucht.

Findet das BIBO-DOS den Dateinamen nicht, erhalten Sie eine Fehlermeldung. Wird der Name aber gefunden, so merkt sich das BIBO-DOS den ersten Sektor der Datei und liest diesen ein.

Stimmt die Dateinummer des Sektorlinks mit der Dateinummer des Directory Eintrages überein, werden die Daten an die CIO Routine des Betriebssystemes zur weiterbearbeitung weitergegeben. Aus dem Sektorlink errechnet das BIBO-DOS den nächsten zu lesenden Sektor und liest auch diesen ein. Dieser Vorgang wird solange wiederholt, bis das BIBO-DOS das Dateiende erreicht hat und die Kontrolle über das Programm wieder an das BASIC oder an das Programm abgibt.

Somit wäre der Ladevorgang bereits beendet. Kein Wunder also, wenn das Laden schneller geht als das Abspeichern.

DATEI UMBENNENEN

Auch hier wird erst wieder der angesprochene Gerätetreiber und der Dateiname auf Gültigkeit überprüft.

Anschließend wird die Directory eingelesen. Der alte Dateiname wird in der Directory gesucht und, falls vorhanden, durch den neuen Namen ersetzt. Findet das BIBO-DOS den alten Namen nicht in der Directory, wird die Arbeit unterbrochen und Sie erhalten eine Fehlermeldung. Und schon kann die Directory wieder abgespeichert werden.Das wär's auch schon.

DATEI SICHERN

Auch hier wieder: erst den Namen und den Gerätetreiber auf Gültigkeit prüfen, dann erst die Directory einlesen. Ist der Dateiname gefunden, wird der Status von Normal ($42) auf Gesichert ($62) geändert. Und schon kann die Directory wieder abgespeichert werden.

Eine Besonderheit gibt es bei Medium Density (DOS 2.5) formatierten Disketten. Dort kennzeichnet der Wert $03 den Normalzustand einer Datei, gesichert wird durch den Wert $23 angezeigt.

DATEI FREIGEBEN

Praktisch gleich mit der Funktion Datei sichern: erst den Namen und den Gerätetreiber auf Gültigkeit prüfen, dann die Directory einlesen. Ist der Dateiname gefunden, wird der Status von Gesichert ($62) auf Normal ($42) geändert. Und schon kann die Directory wieder abgespeichert werden.

Auch hier wieder die Besonderheit bei Medium Density (DOS 2.5) formatierten Disketten. Hier kennzeichnet der Wert $23 den gesicherten Zustand einer Datei, und der Wert $03 den Normalzustand.

DATEI ZURüCKHOLEN

Datei zurückholen ist eine Spezialfunktion des BIBO-DOS. Sie finden diese Funktion weder im ATARI DOS 2.0 noch im ATARI DOS 2.5. Diese Funktion ist wieder etwas schwerer zu erklären.

Auch bei dieser Funktion wird zuerst der angesprochene Gerätetreiber und der Dateiname auf Gültigkeit geprüft.

Anschließend wird die Directory eingelesen und der Dateiname gesucht. Der status "gelöscht" ($80) wird überprüft. Anhand der Directory-Einträge wird der erste Sektor der Datei ermittelt.

Nun wird die VTOC und der erste Sektor der Datei eingelesen. Der Sektor wird als belegt in die Sektortabelle eingetragen und die Anzahl der freien Sektoren um eins heruntergezählt.

Aus dem Sektorlink errechnet das BIBO-DOS den nächsten zu lesenden Sektor der Datei und liest ihn ein. Auch dieser Sektor wird in der Sektortabelle als belegt eingetragen und die Anzahl der freien Sektoren um eins herunter gezählt. Diese Arbeit wird solange wiederholt, bis das Ende der Datei erreicht ist, oder bis ein Fehler auftritt. Bei einem Fehler wird die Arbeit sofort unterbrochen und Sie erhalten eine Fehlermeldung.

Ist das Ende der Datei erreicht, und ist kein Fehler aufgetreten, wird die VTOC abgespeichert und die Directory ein zweitesmal eingelesen. Der Status der Datei wird von Gelöscht ($80) auf Normal ($42) gesetzt und die Directory wieder abgespeichert.

Das war unsere Beschreibung aller BIBO-DOS Funktionen. Damit wäre dann auch unsere Einführung in die Funktionen des BIBO-DOS beendet. Im nächsten Bereich werden wir dann etwas tiefer in die Materie (in das FMS System des BIBO-DOS) einsteigen. Wir werden damit beginnen, die Einsprünge und die Parametertabelle des BIBO-DOS zu erklären. Waren unsere Artikel bislang für den blutigen Anfänger gedacht, so werden die nachfolgenden wohl mehr den Maschinensprachprogrammierer interessieren. Bis dahin wie immer viel Spaß bei der Arbeit mit dem BIBO-DOS.

Wie versprochen beschäftigen wir uns im Rahmen unserer Serie über das BIBO-DOS auch mit der Parametertabelle des BIBO-DOS. Gerade für Programmierer, die das BIBO-DOS für Ihre Zwecke ändern wollen, wird es jetzt interessant. Das folgende ist also für den fortgeschrittenen Maschinensprachprogrammierer gedacht.

Bereits mit dem ATARI DOS 2.0 wurde eine dem BIBODOS sehr ähnliche Parametertabelle eingeführt. Mit dem ersten Sektor wird beim booten der Diskette in die Page 7 des Ramspeichers diese Tabelle eingeladen.

Die Anzahl der einzustellenden Parameter war beim DOS 2.0 allerdings auf nur wenige Funktionen begrenzt. Dieses waren zum Beispiel die Anzahl der Laufwerke und der Filebuffer.

Im BIBO-DOS wurde die Parametertabelle um mehrere Funktionen erweitert und zusätzlich eine Sprungtabelle eingesetzt, deren Prinzip sich schon in der SPEEDY 1050 bewährt hat. Die Sprungtabelle gewährleistet, daß man direkt auf Routinen des DOS zugreifen kann, ohne die unendlichen Weiten des Disk-File-Managers zu ergründen und Speicherstellen ausfindig machen zu müssen. Ein weiterer Vorteil ist, daß die Sprungtabelle (und natürlich auch die Parametertabelle) immer an der gleichen Stelle im Speicher liegen wird, auch bei einem Update oder einer Änderung des BIBODOS.

DIE PARAMETERTABELLE

Adresse - 1822 - $71E DUPDEN
Aktuelle Density.

Diese Adresse wird vom DUP benötigt um die Kapazität des aktuellen Laufwerkes zu ermitteln. Dieses Byte ist das Statusbyte des zuletzt angesprochenen Laufwerkes.

Bit 7 = 1 Medium Density
Bit 5 = 1 Double Density

Sind Bit 7 und Bit 5 gleich Null, handelt es sich um Single Density. Diese Adresse darf nicht verändert werden.

Adresse - 1823 - $71F SECLEN
Aktuelle Sektorlänge.

Hier steht die Datenlänge eines Sektors des zuletzt angesprochenen Laufwerkes.

Single und Medium Density = $7D 125
Double Density = $FD 253

Auch diese Adresse darf nicht verändert werden.

Adresse - 1824 - $720 DRIVES
Anzahl der angeschlossenen Laufwerke.

Hier kann angegeben werden, für wieviele Laufwerke Dateibuffer reserviert werden sollen. Maximal 8 Laufwerke sind theoretisch möglich. Da sich die Laufwerke allerdings nur bis zu Nummer 4 einstellen lassen, wäre es natürlich unsinnig, hier ein Laufwerk höher als Nummer 4 anzumelden. Eine Ramdisk darf hier nicht angemeldet werden. Die Daten in der Speicherstelle DRIVES sind Bitweise orientiert. BIT 0 steht für Laufwerk 1, BIT 1 für Laufwerk 2 usw. Somit ist es auch möglich Laufwerk 1 und 3 anzumelden. Hierzu schreiben Sie in die Speicherstelle den Wert 5 (Binär 00000101). Wenn der Wert der Speicherstelle DRIVES geändert wird, muß die aktuelle Konfiguration erst mit einem DOS-Init Aufruf aktiviert werden ($709). Vorsicht!!! Hierdurch verändert sich der LOMEM-Zeiger, Programmteile (BASIC) können zerstört werden.

Adresse - 1825 - $721 BUFFERS
Anzahl der Filebuffer.

Hier kann eingetragen werden, wieviele Dateien maximal gleichzeitig geöffnet werden dürfen. Hier sollte mindestens 1 stehen. Zur sicheren Funktion des DUP muß hier allerdings eine 2 eingetragen sein, da das BIBODOS DUP für einige Funktionen (COPY, DELETE) zwei Dateien gleichzeitig öffnet. Die maximale Anzahl ist hier auf 6 begrenzt. Der Kanal 0 ist für den Bildschirm und die Tastatur reserviert, so daß nur Kanal 1 bis 7 für die Arbeit mit dem DOS zur Verfügung steht. Achtung!!! Basic benutzt den Kanal 6 für das Arbeiten mit dem Grafikbildschirm. Auch hier kann eine Änderung erst durch aufruf des Dos-Init aktiviert werden. Der LOMEM-Zeiger wird ebenfalls verändert.

Adresse - 1826 - $722 RDRIVE
Nummer der Ramdisk.

Hier steht, unter welcher Laufwerksnummer die Ramdisk angesprochen werden kann. Die Laufwerksnummer darf nicht gleich mit der eines aktiven Laufwerkes werden. Sind Laufwerke 1 und 2 angemeldet, muß die Ramdisk-Nummer größer als 2 sein (3-8). Ein negativer Wert (128/$80) in dieser Speicherstelle deaktiviert die Ramdisk. Auch hier muß nach Veränderung des Wertes in dieser Speicherstelle die Funktion Dos-Init ($709) ausgeführt werden. Eine Veränderung der Ramdisk-Nummer geschieht ohne Veränderung des LOMEM-Zeigers. Lediglich ein aus- oder einschalten der Ramdisk kann diesen verändern.

Adresse - 1827 - $723 RESID
Flag für Ramdisk resident.

Dieses Flag gibt an, ob der Inhalt der Ramdisk bei einem Kaltstart des Rechners erhalten bleiben soll. Schreibt man in diese Speicherstelle 255/$FF wird die Ramdisk bei jedem Kaltstart gelöscht. Ein Wert von 0 zeigt an, daß die Ramdisk resident ist. Der Inhalt dieser Speicherstelle ist nur beim booten des DOS wichtig und ist natürlich auch nur von Bedeutung, wenn eine Ramdisk angemeldet und auch vorhanden ist.

Adresse - 1828 - $724 MAXBANK
Maximale Anzahl der Ram-Banks.

Für diese Adresse sind drei verschiedene Werte möglich:

$04 bei 64k Ramdisk (130XE mit 128KB)
$08 bei 128k Ramdisk (XL/XE mit 192KB)
$10 bei 256k Ramdisk (XL/XE mit 320KB)

Beachten Sie!!! Der Ramdisktreiber im BIBODOS testet nicht die tatsächliche Größe der im Rechner befindlichen Ramdisk. Wird eine zu große Ramdisk angemeldet, können bei Überschreitung der Ram-Kapazität wichtige Daten zerstört werden.

Adressen - 1829-1844 - $725-$734 RDTABL
Ramdisk-Bank Bittabelle.

Diese 16 Speicherstellen geben an, in welcher Reihenfolge die Banks der Ramdisk eingeschaltet werden sollen. Hierbei wird der Wert angegeben, der in die Speicherstelle $D301 (Port B) geschrieben werden soll. Folgende Tabelle bezieht sich auf alle COMPY-SHOP Ramdisks und der 130 XE Ramdisk. Für Ramdisks von Fremdherstellern muß die Tabelle eventuell angepaßt werden (Fragen Sie beim Hersteller der Ramdisk nach!).

Wichtig ist nur, daß auch diese Ramdisk-Banks den Speicherbereich $4000- $7FFF benutzen und die einzelnen Banks mit der Speicherstelle $D301 (PIO Port B) eingeschaltet werden.

Adresse D301	HIGH (1)	LOW (0)
Bit 0	HIGH (1)	OS-Rom aktiv
	LOW (0)	OS-Rom abgeschaltet
Bit 1	HIGH (1)	Basic abgeschaltet
	LOW (0)	Basic aktiv
Bit 2	/ Wahl der Banknummer bei 130	
Bit 3	/ XE und COMPY-SHOP-Ramdisk	
	/ 2 Bits = 4Banks	
Bit 4	HIGH (1)	kein CPU-Zugriff
	LOW (0)	CPU-Zugriff auf Ramdisk erlaubt
Bit 5	HIGH (1)	kein ANTIC-Zugriff
	LOW (0)	ANTIC-Zugriff auf Ramdisk erlaubt
Bit 6	Wahl der Banknummer bei 128k und 256k Ramdisk	
Bit 7	HIGH (1)	Selbsttest aus
	LOW (0)	Selbsttest ein

Wahl der Banknummer bei 256k Ramdisk, wenn CPU- oder ANTIC-Zugriff auf die Ramdisk möglich ist. Aufbau der Ramdisktable im BIBODOS:

```
 $725  EC E8 E4 E0     Block 1
 $729  6C 68 64 60     Block 3
 $72D  AC A8 A4 A0     Block 2
 $731  2C 28 24 20     Block 4
```

Bei einer 64k Ramdisk wird Block 1 benötigt, bei einer 128k Ramdisk Block 1 und 2, bei einer 256k Ramdisk werden alle 4 Blöcke benutzt. Die Blöcke 2 und 3 sind vertauscht, da hierdurch ein geringerer Programmieraufwand für den Ramdiskhändler erreicht wurde. Beachten Sie, daß die Tabelle immer in dieser vertauschten Reihenfolge vom Ramdisktreiber ausgelesen wird. Die Werte in der Ramdisk-Tabelle setzen sich aus den 4 wichtigen Bits für die Anwahl der Ram-Bank und den Bits für den Ramdisk Zugriff zusammen. Die Bits 0 und 1 werden im

Ramdisktreiber ausmaskiert und sind in der Tabelle ohne Bedeutung. Die Abkürzung B. steht in der Tabelle für BANK.

Nr	Byte	B.	B.	ANTIC	CPU	B.	B.	Bit1	Bit0
00	EC	1	1	1	0	1	1	-	-
01	E8	1	1	1	0	1	0	-	-
02	E4	1	1	1	0	0	1	-	-
03	E0	1	1	1	0	0	0	-	-
04	AC	1	0	1	0	1	1	-	-
05	A8	1	0	1	0	1	0	-	-
06	A4	1	0	1	0	0	1	-	-
07	A0	1	0	1	0	0	0	-	-
08	6C	0	1	1	0	1	1	-	-
09	68	0	1	1	0	1	0	-	-
0A	64	0	1	1	0	0	1	-	-
0B	60	0	1	1	0	0	0	-	-
0C	2C	0	0	1	0	1	1	-	-
0D	28	0	0	1	0	1	0	-	-
0E	24	0	0	1	0	0	1	-	-
0F	20	0	0	1	0	0	0	-	-

Wie Sie sehen, kann die Ramdisk-Größe hierbei nur durch ändern der Bank-Anzahl geändert werden, die RamdiskTablelle stimmt automatisch.

Adresse - 1845 - $735 RUNFLG
Flag ob geladenes File automatisch gestartet werden soll.

Ein mit der Funktion FLOADR ($70F) eingeladenes File wird abhängig von diesem Byte gestartet. Steht hier der Wert 00 wird das File auf jeden Fall initialisiert und gestartet. Bei einem negativen Wert (255/$FF) wird das Programm nur eingeladen ohne gestartet zu werden. Dieses ist manchmal notwendig, wenn es sich bei der Datei um ein Datenfile handelt, das kein funktionsfähiges Programm enthält.

Adresse - 1846 - $736 DUPFLG
Flag ob sich das BDUP unter dem Betriebssystem befindet.

Steht in dieser Speicherstelle der Wert $FF, befindet sich das BIBODOS-DUP unter dem Betriebssystem. Bei einem Wert 00 steht das DUP ab Speicherstelle $2400 im Ram. Diese Speicherstelle darf nur vom DUP geändert werden.

Adresse - 1847 - $737 DEXIST
Flag ob das BDUP bereits geladen wurde.

Hier wird angegeben, ob das BDUP.SYS bereits erfolgreich eingeladen wurde. Steht hier 00 konnte das DUP noch nicht geladen werden. Bei einem negativen Wert (255/$FF) wurde das DUP eingeladen und kann ab Adresse $2400 gestartet werden. Diese Speicherstelle darf nicht verändert werden.

Adresse - 1848 - $738 SETDRV
Absolute Anzahl der vom DOS verwalteten Laufwerke.

Hier ist eingetragen, welche Laufwerke tatsächlich angemeldet sind. Ist keine Ramdisk vorhanden, entspricht der Wert in dieser Speicherstelle genau dem der Speicherstelle $720 DRIVES. Ist jedoch eine Ramdisk vorhanden, ist diese hier zusätzlich eingetragen. Diese Speicherstelle wird bei der Dos-Init Routine ausgelesen um die Laufwerk-Buffer zu erstellen. Ein ändern der Speicherstelle hat keinen Einfluß auf die Funktion des DOS.

Adresse - 1849 - $739 KBFLG
Tastaturbuffer-Flag.

Hier kann bestimmt werden, ob der Tastaturbuffer aktiviert sein soll. Bei einem Wert 00 ist der Buffer aktiv, bei einem negativen Wert (255/$FF) ist der Tastaturbuffer abgeschaltet. Bitte beachten Sie: Der Tastaturbuffer sperrt den Keyboard-Interrupt ($208/209). Der VBI-Vektor ($222/223) wird verändert. Dieses kann bei manchen Programmen Probleme geben. Sollte ein Programm einmal nicht laufen, schalten Sie den Tastaturbuffer ab und versuchen es noch einmal.

Soweit die Beschreibung der Parametertabelle des BIBO-DOS. Wir hoffen, daß interessierte Programmierer viele Anregungen für eigene Programme bekommen haben. Vieleicht können wir ja irgendwann einmal ein Listing oder ein Programm veröffentlichen, das auf diesen Artikel beruht.

Jetzt beschäftigen wir uns im Rahmen unserer Serie über das BIBO-DOS mit der Sprungtabelle des BIBO-DOS. Gerade für Programmierer, die ein Programm speziell für das BIBO-DOS schreiben wollen, wird es jetzt interessant. Das folgende ist also für den fortgeschrittenen Maschinensprachprogrammierer gedacht. Alle Listings im Artikel wurden, wie immer, mit dem BIBO-Assembler geschrieben.

Im BIBODOS wurde eine Sprungtabelle eingesetzt, deren Prinzip sich schon in der SPEEDY 1050 bewährt hat. Die Sprungtabelle gewährleistet, daß man direkt auf Routinen des DOS zugreifen kann, ohne die unendlichen Weiten des Disk-File-Managers zu ergründen und Speicherstellen ausfindig machen zu müssen. Ein weiterer Vorteil ist, daß die Sprungtabelle immer an der gleichen Stelle im Speicher liegen wird, auch bei einem Update oder einer Änderung des BIBODOS.

DIE SPRUNGTABELLE:

Adresse - 1801 - $709 JMP RESINI
Dos-Reset Initialisierung.

Soll die Handler-Tabelle für das DOS erneuert, die DOS-Buffer oder die HighSpeed-Routine neu initialisiert werden, kann zu dieser Adresse gesprungen werden.

Adresse - 1804 - $70C JMP TSTRDS
Sprung zur High-Speed-Routine der Fast-Version des BIBODOS oder Sektoren der Ramdisk lesen/schreiben.

Der Sprung zeigt bei der Normalversion des BIBODOS auf den SIO-Vektor des Betriebssystems ($E459). Soll die High-Speed der SPEEDY 1050 voll genutzt werden, können Programme direkt die High-Speed-Routine des BIBODOS benutzen. Die Parameter werden genauso wie bei dem Einsprung in das Betriebssystem übergeben. Die Übertragungsrate der High-Speed Routine paßt sich automatisch an die SPEEDY 1050 an. Sollte ein Übertragungsfehler auftreten oder die SPEEDY sich im Slow-Modus befinden, wird die Übertragungsrate automatisch auf normal eingestellt.

Soll die hohe Geschwindigkeit wieder aktiviert werden, muß DOS-Init ($709) aufgerufen werden. Über den gleichen Vektor können Sektoren der Ramdisk gelesen oder geschrieben werden, wenn die Laufwerksnummer in $301 mit dem Wert in RDRIVE ($722) übereinstimmt (Ab BIBODOS Version 5.2). Als Bufferadresse in $304/305 darf keine Adresse zwischen $4000-$7FFF vorkommen.

Adresse - 1807 - $70F JMP FLOADR
                       File-Laderoutine des BIBODOS aufrufen.

Eine COM- oder EXE-Datei wird in den Computer eingeladen und abhängig vom RUNFLG ($735) gestartet. Als Übergabeparameter wird nur die Adresse des Dateinamens benötigt:

```
          LDA #FILENAME   (Low Byte)
          LDY /FILENAME   (High Byte)
          JMP $70F
FILENAME  .DA  "DATEI.BAS",#$9B
```

Der Dateiname muß mit $9B abgeschlossen sein. Bevor die Datei eingeladen wird, wird das BDUP.SYS unter das Betriebssystem geschoben, damit der vollständige Ramspeicher zur Verfügung steht. Sollte das geladene Programm beendet werden, wird das BDUP.SYS zurückgeholt, falls es vor dem Aufruf unter das Betriebssystem geschoben wurde.

Adresse - 1810 - $712 JMP SAVDAT
                       Universelle SIO-Routine.

Diese Routine kann zum abspeichern von Programmen benutzt werden. Hierbei wird das BDUP.SYS vor dem Aufruf der CIO-Routine unter das Betriebssystem geschoben und nach beenden der CIO-Routine wieder zurückgeholt. Der CIO-Kanal 1 muß vor dem Aufruf bereits geöffnet sein und muß nachher wieder geschlossen werden.

Programmbeispiel:

```
LDX #$10       Kanal 1
LDA #3         Befehl für OPEN
STA $342,X
LDA #NAME      Adresse des Dateinamens
STA $344,X
LDA /NAME
STA $345,X
LDA #8         Öffnen zum schreiben
STA $34A,X
JSR $E456      CIO-Aufruf
LDA #$B        Befehl für PUT Character
STA $342,X
LDA #ADRESSE   zu speichernder
               Adressbereich
STA $344,X
LDA /ADRESSE
STA $345,X
LDA #LAENGE    Anzahl der zu speichern-
               den Daten
STA $348,X
LDA /LAENGE
STA $349,X
JSR SAVDAT     (nach $0712)
LDX #$10       X-Register wird von
               SAVDAT verändert
LDA #$C        Befehl CLOSE
STA $342,X
JSR $E456      CIO-Aufruf
...
...
u.s.w.
```

Adresse - 1813 - $715 JMP GOCART
                       Zur Cartridge oder zum Basic springen.

Das BDUP.SYS wird, falls notwendig unter das Betriebssystem geschoben. Dieser Einsprung wird vom DUP des BIBODOS benutzt.

Adresse - 1816 - $718 JMP RUNADR
                       Zur angegebenen Adresse springen.

Auch dieser Einsprung wird vom BIBODOS DUP benutzt. Das BDUP.SYS wird unter das Betriebssystem geschoben. Die Adresse, zu der gesprungen werden soll, wird in den Registern X=Low und Y=High übergeben. Beispiel:

```
LDX #START     Adresse low
LDY /START     Adresse high
JSR RUNADR     (nach $0718)
...
...
u.s.w.
```

Die Sprungadressen $70F (FLOADR), $712 (SAVDAT), $715 (GOCART) und $718 (RUNADR) sollten nur von einem DUP.SYS aufgerufen werden, da dieses hierdurch kurzzeitig unter das Betriebssystem verschoben wird und den Speicher dadurch freigibt. Sollte ein neues DUP geschrieben werden, darf dieses nur im Speicherbereich $2400 bis $3FFF liegen, da nur dieser Bereich unter das Betriebssystem geschoben wird (mehr paßt auch nicht darunter). Die Startadresse muß bei $2400 liegen. Alle Einsprünge in das BDOS.SYS dürfen nur über die Sprungtabelle oder über den CIO-Handler ($E456) durchgeführt werden.

So weit zur Sprungtabelle. Sollten sich im laufe der Zeit Änderungen oder Erweiterungen dieser Tabelle ergeben, werden wir Sie selbstverständlich darüber Informieren. Jetzt aber noch zu den Unterschieden der BIBO-DOS Versionen 5.x und 6.0 (bzw. 6.1).

Das BIBO-DOS 6.x

Das ATARI-Laufwerk XF 551 kann eine Diskette beidseitig formatieren. Die Kapazität einer Diskette läßt sich so auf 360 kByte steigern. Leider war bis zum erscheinen des BIBO-DOS 6.0 kein anderes DOS in der Lage, diese hohe Kapazität auf der XF 551 fehlerfrei auszunutzen. Warum das DOS speziell auf dieses Laufwerk angepaßt werden mußte und welches die großen Unterschiede zu den BIBO-DOS Versionen 5.x sind, werde ich in diesem Artikel erklären.

Die XF 551 erkennt keinen Diskettenwechsel selbsständig, das heißt, wenn zum Beispiel eine Datei von einer Single-Density formatierten Diskette auf eine Double-Density Diskette kopiert werden soll, stellt sich das Laufwerk nicht von selber auf die geänderte Diskettendichte ein. Die logische Folge wäre ein Disk-Error und das Kopierprogramm bricht den Vorgang ab. Hier ist auch schon der Grund, warum die XF 551 nicht mit jedem anderen DOS vernünftig arbeitet. Das DOS muß nämlich den Befehl an das Laufwerk geben, die Diskettendichte zu testen und sich somit auf diese neue Dichte einzustellen.

Dieser Vorgang geschieht bei der XF 551 durch lesen eines Sektors auf Track 0. Warum das Umstellen der Dichte nicht mit den Sektoren 1 bis 3 klappt, ist mir bis jetzt immer noch ein Rätsel. Sollten Sie jetzt denken, die Sektoren 1 bis 3 sind ohnehin nur in Single-Density, haben Sie nur teilweise Recht. Wenn einer dieser drei Sektoren angesprochen wird, sendet das Laufwerk immer nur die ersten 128 Bytes dieses Sektors zum Laufwerk. Auf der Diskette kann der Sektor aber bei Double-Density volle 256 Bytes lang sein. Der Rest des Sektors kann nicht gelesen und beschrieben werden.

Im BIBO-DOS wird nun zum Density-Test bei jedem OPEN-Befehl der Sektor 4 gelesen. Da man von vornherein nicht weiß, wieviele Daten nun vom Laufwerk gesendet werden (128 oder 256) wird als Länge einfach 0 angegeben. Die SIO-Routine meldet hierbei zwar einen Checksum-Error, da die Anzahl der empfangenen Daten nicht mit der geforderten Anzahl übereinstimmt. Diese Fehlermeldung wird allerdings ignoriert, wichtig ist nur, daß das Laufwerk diesen Sektor liest.

Erst nach diesem Vorgang kann der Status des Laufwerks gelesen werden und die Dichte der Diskette dem DOS mitgeteilt werden. Zwischen dem lesen des Sektors 4 und dem Statusbefehl befindet sich eine kleine Verzögerungsschleife. Diese wird benötigt um dem Laufwerk Zeit zu geben die restlichen Daten des gelesenen Sektors 4 zu senden. Diese Verzögerung und das Lesen des Sektors 4 bei jedem OPEN-Befehl bedeutet leider, daß hier einiges an Zeit benötigt wird, was das arbeiten mit der XF 551 noch zeitraubender macht als es ohnehin schon ist. Wenn ATARI die Technik des 1050 Laufwerk beibehalten hätte, wäre dieser Aufwand and Software nicht notwendig.

Im BIBO-DOS wurde nun eine Zeitschleife eingebaut, die diesen Density-Test unterdrückt, wenn er zu schnell hintereinander erfolgt. Wenn Sie im BIBO-DOS schnell hintereinander die Taste 1 für Directory Laufwerk 1 drücken, werden sie feststellen, daß dieser Vorgang das erste mal länger dauert als bei den nachfolgenden. Drücken Sie die Taste im Abstand von etwa 3-4 Sekunden, werden Sie feststellen, daß der Vorgang immer gleich lange braucht. Hier wird vor jedem Directory-Aufruf erst der Density-Test ausgeführt.

Auf einer Diskette in QUAD-Density (360 kByte) befinden sich 1440 Double-Density Sektoren. Wenn man die 3 Bootsektoren, den VTOC-Sektor, die 8 Directory-Sektoren und den nicht benutzten Sektor 1440 abzieht, bleiben 1427 Sektoren frei für Programme und Daten. Der VTOC-Sektor wird auf volle 256 Bytes genutzt. Hierdurch lassen sich alle Sektoren mit einem einzigen VTOC-Sektor verwalten.

64 Einträge in die Directory wären bei dieser Datenmenge jedoch viel zu wenig. Eine Verdopplung der Einträge hat sich als Notwendig erwiesen. Auch die Directory-Sektoren werden voll benutzt, das heißt, es passen auf jeden Sektor 16 Einträge, was bei 8 Sektoren genau 128 Einträge ergibt.

Das letzte Problem, das sich bei dieser hohen Sektoranzahl ergeben hat, waren die Sektorlinks und die Dateinummer im Sektorlink. 1440 Sektoren, Hex 5A0, benötigen im High-Byte der Sektorlinks schon 1 Bit mehr als beim normalen DOS 5.2 Format. Hinzu kommt, daß für die Dateinummer auch noch ein Bit mehr benötigt würde ($80 statt $40). Aus diesem Grund wurde auf die Dateinummer im Sektorlink vollständig verzichtet, wenn eine Diskette auf 360 kByte formatiert wird. Der Sektor-Link Error #164 wird hierdurch nicht mehr auftreten, wenn etwas auf der Diskette nicht stimmt. Hier ist deshalb Vorsicht geboten, wenn sich defekte Dateien auf der Diskette befinden.

Die Funktion 'J' Datei zurückholen ist ebenfalls mit Vorsicht zu genießen und nur anzuwenden, wenn man absolut sicher ist, nach dem löschen der Datei keine neue Datei auf die Diskette geschrieben zu haben. Die Datei wird auf jeden Fall zurückgeholt, auch wenn die Daten gar nicht mehr zu der alten Datei gehören.

Dieses sind die wichtigsten Unterschiede der BIBO-DOS Version 6.0 zu den Versionen 5.x. Leider haben sich in der Version 6.0 zwei kleine Fehler eingeschlichen, die mit der neuen Version 6.1 beseitigt sind. Besitzer des BIBO-DOS 6.0 bekommen die neue Version durch Einsendung ihrer Original-Diskette und beigelegtem Rückporto.

Damit beenden wir nun die Artikel Serie über das BIBO-DOS. Wir hoffen, das sie Ihnen Spaß gemacht hat und das Sie etwas gelernt haben. Sollten bei Ihnen durch die Serie Fragen zum BIBO-DOS entstanden sein, stehen wir Ihnen natürlich gerne mit Rat und Tat zur verfügung.