In die Geheimnisse der Floppy eingetaucht (Teil 4)

In dieser Folge beschäftigen wir uns das erste Mal mit dem DOS der Floppystation. Wir wollen uns die Technik der Diskettenaufzeichnung ansehen und die Funktionsweise des DOS genauer unter die Lupe nehmen.

Zuerst wollen wir uns mit dem Aufzeichnungsformat der Diskette beschäftigen: Für einen einwandfreien Betrieb der Floppystation ist es unumgänglich, daß sich Markierungen auf der Diskette befinden. Diese Markierung braucht das Laufwerk, um bestimmte Daten schnell finden zu können. Hierfür gibt es prinzipiell zwei Möglichkeiten: die Hardsektorierung und die Softsektorierung.

Hardsektorierte Disketten erkennt man daran, daß diese eine ganze Anzahl von Indexlöchern besitzen. Damit sind die kleinen Löcher nahe am Innenrand der Magnetscheibe gemeint. Mit einer Fotozelle können nun diese Löcher abgetastet werden, um die jeweilige Position der Diskette festzustellen. Dieses Verfahren hat den Vorteil, daß die Diskettenkapazität voll ausgenutzt werden kann. Es können so bis zu 5 MBytes Daten auf eine 5¼-Zoll-Diskette geschrieben werden. Allerdings erfordert diese Methode einen enormen Hardwareaufwand, der den Preis in die Höhe schnellen läßt. Für preiswerte Laufwerke (wie die 1541) geht man daher einen anderen Weg: die Softsektorierung. Hier besitzt die Diskette nur ein Indexloch zur Drehzahlüberwachung. Bei der 1541 ist sogar noch nicht einmal dieses erforderlich. Die notwendigen Markierungen werden beim Formatierungsvorgang softwaremäßig auf die Diskette aufgebracht, wobei natürlich wertvoller Speicherplatz verloren geht. Softsektorierte Disketten im 5¼-Zoll-Format verfügen daher über zur Zeit maximal 1 MByte Speicherkapazität.

Uns soll also im weiteren die Softsektorierung beschäftigen, wobei in Bild 1 eine Diskette schematisch dagestellt ist, nachdem sie auf der 1541 formatiert wurde. Sie ist in 35 konzentrische Spuren, nachfolgend Tracks genannt, aufgeteilt. Jeder dieser Tracks enthält wiederum eine bestimmte Anzahl von Sektoren, die von außen nach innen abnimmt. Diese Tatsachen sind Ihnen aber schon aus der ersten Folge bekannt. Nun wollen wir genauer auf den Aufbau der Sektoren einer Diskette eingehen.

Jeder Sektor besteht aus einem Blockheader und dem dazugehörigen Datenblock; eine schematische Darstellung zeigt Bild 2. Angeführt werden die Sektoren einer Diskette von den schon erwähnten Markierungen, die der Orientierung dienen. Diese Marken bezeichnet man als Synchron (SYNC)-Markierungen, sie bestehen aus mehreren $FF auf der Diskette. Erkennt der Schreib-/Lesekopf der Floppy also eine solche Marke, dann »weiß« die Floppystation, daß entweder ein Blockheader oder ein Datenblock nachfolgt. Nun müssen wir nur noch diese beiden voneinander unterscheiden können.

Hierzu dient das nächste Kennzeichen auf Diskette. Es folgt direkt nach der SYNC-Markierung und meldet dem Diskcontroller (DC) ob ein Blockheader oder ein Datenblock vorliegt. Hat das Kennzeichen den Wert $08, so handelt es sich um einen Blockheader; findet der Kopf hingegen den Wert $07, so handelt es sich um den Beginn eines Datenblocks.

Wir nehmen jetzt einmal an, der DC hätte das Kennzeichen $08 entdeckt; es handelt sich also um den Header eines Datenblocks. Dann folgt als nächstes Byte die Prüfsumme über den Header, die zur Kontrolle auf Lesefehler dient. Die Reihenfolge der Headerbytes, wie sie im Commodore-Handbuch angegeben ist, stimmt nicht mit der Aufzeichnung auf Diskette überein.

Die nächsten 2 Bytes stellen Sektor- und Tracknummer dieses Sektors dar. Anhand dieser Werte kann der DC bei Trackwechsel sehr schnell die Position des Schreib-/Lesekopfes ausfindig machen.

Das 5. und 6. Byte des Blockheaders geben jeweils einen Teil der ID der Diskette an, und zwar folgen zuerst das zweite und dann das erste Zeichen der ID, die beim Formatieren festgelegt wurden. Mit diesen Angaben ist die Behandlung des Headers bereits abgeschlossen. Es folgen jetzt noch ein paar Bytes, die eine Lücke darstellen.

Mit der nächsten SYNC-Markierung wird der Beginn des eigentlichen Datenblocks eingeleitet. Nach der SYNC-Marke folgt das Datenblockkennzeichen $07. Die nächsten zwei Bytes sind uns bestens bekannt. Sie können mit jedem Diskmonitor angesehen werden und geben Track- und Sektornummer des nächsten Blocks im File an. Man bezeichnet sie deshalb als Linker oder Linkadressen (engl.: to link = verbinden).

Nun erst folgen die eigentlichen Daten auf Diskette, die in

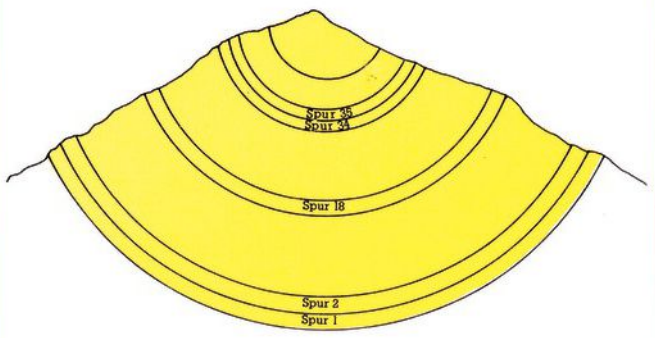

Bild 1. Aufbau einer formatierten Diskette

jedem Block 254 Byte ausmachen.

Hinter diesen Datenbytes steht die Prüfsumme des Datenblocks, die wiederum zum Erkennen von eventuellen Lesefehlern dient. Werden solche Fehler festgestellt, so versucht die Floppystation noch mehrere Male, den Block doch zu lesen. Erst wenn viele Versuche kein befriedigendes Ergebnis bringen, steigt sie mit einer Fehlermeldung aus.

Nach der Prüfsumme des Datenblocks folgt wieder eine Lücke auf der Diskette, bevor die SYNC-Markierung des nächsten Blockheaders kommt. Wenn wir uns diesen Aufbau eines Sektors betrachten, wird klar, warum die Speicherkapazität bei softsektorierten Disketten gegenüber hardsektorierten Disketten deutlich abnimmt.

Jetzt werden sie vielleicht auch die Beschreibung der Fehlermeldungen im Floppyhandbuch verstehen, die wir hier nicht mehr aufführen, da sie dort sehr genau und richtig erläutert werden.

Das Verständnis des Diskettenaufbaus ist für die weitere Behandlung des Floppy-DOS unerläßlich, da wir nur so die Funktionsweise begreifen lernen.

Jetzt wollen wir uns aber einmal mit der grundlegenden Arbeitsweise des Floppybetriebssystems (DOS) befassen, die um einiges komplizierter ist, als die im Computer.

Wenn wir die Floppy einschalten, passiert zunächst das gleiche, wie im Computer. Die RESET-Leitung geht auf Low und der Mikroprozessor, hier ein 6502, holt sich seine Systemstartadresse. Danach läuft das RESET-Programm an, wobei die Floppy einen Selbsttest durchführt. Erkennen können Sie dies daran, daß für kurze Zeit der Motor anläuft und die rote LED leuchtet. Wurde kein Defekt registriert, so erlischt die Leuchtdiode wieder, und der Motor geht aus. Jetzt wird der RAM-Bereich der Floppy initialisiert und alle wichtigen Zeiger werden hergestellt. Danach ist die 1541 betriebsbereit.

Von jetzt an laufen quasi drei Programme gleichzeitig ab:
— das Hauptprogramm läuft in einer Schleife, die nur bei der Ausführung von Befehlen verlassen wird;
— das Diskcontrollerprogramm wird über den IRQ gesteuert und durch den Timer des DC alle 10 ms aufgerufen;
— die Routinen des Buscontrollers (BC) schließlich, werden nur im Bedarfsfall aufgerufen, nämlich, wenn die ATN-Leitung des seriellen Bus auf Low geht.

Wir wollen uns die Funktion dieser Routinen nun einmal etwas genauer betrachten.

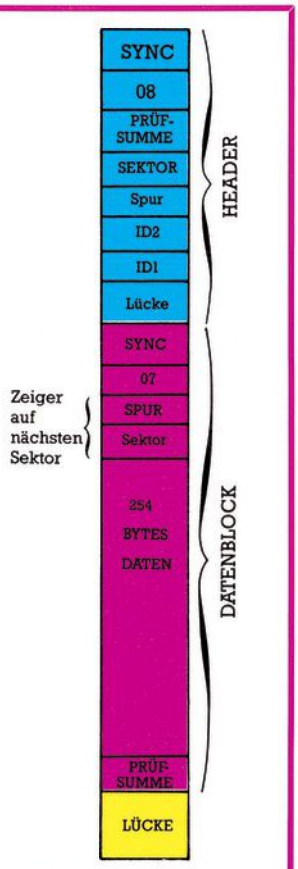

Bild 2. Aufbau eines Sektors

Das Hauptprogramm

Das Hauptprogramm hängt, wie schon gesagt, in einer Warteschleife, bis ein Befehl vom Computer kommt. Dieser aktiviert zuerst die Busroutinen, die die gesendeten Bytes dann entgegennehmen und abspeichern. Jetzt bekommt das Hauptprogramm, das übrigens den Zustand der beiden IRQ-Routinen (DC und BC) ständig überwacht, die Meldung, daß ein Befehl anliegt. Es verzweigt nun zur Befehlsauswertung, ähnlich dem Basic-Interpreter, und führt gegebenenfalls einen Befehl aus, sofern ein Syntaxfehler entdeckt wurde. In diesem Fall würde sonst eine Fehlermeldung generiert, die dann vom Computer ausgelesen werden kann.

Ist ein Befehl korrekt ausgeführt worden, so werden die Befehlsparameter wieder gelöscht, und das Hauptprogramm kehrt in die Warteschleife zurück.

Das Diskcontrollerprogramm

Der Diskcontroller enthält den Baustein VIA 6522, durch den er mit dem Mikroprozessor in Kontakt steht. Dieser Baustein enthält auch Timer, die in einem eingestellten Rhythmus einen IRQ auslösen können. Einer dieser Timer ist in der 1541 so eingestellt, daß er ungefähr alle 10 ms einen IRQ auslöst, der dann seinerseits das Diskcontrollerprogramm aufruft.

Es soll an dieser Stelle der Unterschied zwischen Diskcontroller und Diskcontrollerprogramm erläutert werden: Als Diskcontroller (DC) bezeichnet man die Hardware in der Floppy, die für den Laufwerksbetrieb zuständig ist.

Unter dem Diskcontrollerprogramm versteht man den Programmteil im DOS, der, durch IRQ geregelt, die Ansteuerung des DC übernimmt.

Eine vollständige Trennung dieser beiden Begriffe ist jedoch weder notwendig noch zweckmäßig, so daß wir mit dem Ausdruck »DC« immer die Gesamtheit von Hard- und Software beschreiben wollen. Nun aber wieder zu den Aufgaben des DC.

Auch dieses Programm hat eine Art Wartezustand, solange kein Befehl vom Computer anliegt. Wird nämlich das Hauptprogramm über den Bus aktiviert, so wertet dieses die Befehle aus und gibt sie an den DC weiter, der dann seinerseits dafür sorgt, daß das Laufwerk aktiviert wird. Er steuert den Laufwerk- und den Stepper (Schreib-/Lesekopf)-Motor und bedient die Daten, die vom und zum Tonkopf gehen. Die gesamten Vorgänge am Laufwerk werden also interruptgesteuert vorgenommen.

Die Busroutinen

Die Routinen des Buscontrollers (BC) werden ebenfalls über die IRQ-Leitung gesteuert. Auch der BC enthält einen VIA 6522-Baustein. Hier wird der Aufruf der Routinen allerdings nicht über den Timer organisiert, sondern, wie schon erwähnt, über die ATN-Leitung des seriellen Busses. Zieht der Computer also diese Leitung auf Low, so wird in der Floppy (und in allen anderen Peripheriegeräten ebenso) ein IRQ ausgelöst. Dann erfolgt die Abfrage, ob dieser IRQ vom Timer des DC kam. Ist dies nicht der Fall, so wird die BC-Routine aufgerufen, die dann den weiteren Busbetrieb übernimmt. Sollte die Floppy gerade einen Befehl bearbeiten, während schon ein neuer vom Computer gesendet wird, so wartet der BC solang mit der Annahme, bis die Floppy wieder in den Bereitschaftszustand zurückgekehrt ist.

Wie Sie sehen, stellt das DOS eine ziemlich komplizierte Einheit dar, deren Schema in Bild 3 zu sehen ist.

Wie Sie vielleicht bemerkt haben, ist uns in Folge 2 ein Fehler unterlaufen. Das abgedruckte Listing 6 wäre eigentlich Listing 5 gewesen. Als tatsächliches Listing 6 liefern wir Ihnen heute das Directory-Sortierprogramm nach, das Sie in Listing 1 abgedruckt finden.

```
100 REM DIRECTORY-SORTER                          <180>
101 REM SORTIERT DIRECTORY ALPHABETISCH           <141>
102 REM BEI VIELEN EINTRAEGEN BITTE               <226>
103 REM ETWAS GEDULD (MAX. 5.MIN)                 <219>
104 REM SORTIERT AUCH GESCRATCHTE FILES           <052>
105 REM MIT, STELLT SIE ABER NICHT                <087>
106 REM WIEDER HER ! SORTIERALGORITHMUS           <048>
107 REM KANN SICH IN EINEM SOLCHEN FALL           <121>
108 REM IN EINER ENDLOSSCHLEIFE VER-              <039>
109 REM HEDDERN. ABHILFE: NACH 3-4 MIN.           <009>
110 REM STOP-TASTE DRUECKEN, DANN                 <143>
111 REM GOTO 210 EINGEBEN. SIND EINTR.            <019>
112 REM DANN NOCH NICHT VOLLKOMMEN SOR-           <227>
113 REM TIERT, NOCHMALS FUER EINIGE               <236>
114 REM MINUTEN LAUFEN LASSEN.                    <208>
115 REM ACHTUNG !!! NUR ZUSAMMEN MIT              <190>
116 REM DEN UNTERPROGRAMMEN 1 & 2                 <233>
117 REM ABLAUFFAEHIG !!!                          <182>
118 :                                             <176>
119 :                                             <177>
120 DIM DD$(144)                                  <148>
130 MM=MM+1:GOSUB 1000                            <203>
140 IF DD$=NN$THEN MM=MM-1:GOTO 160               <248>
150 DD$(MM)=DD$:DD$="":GOTO 130                   <190>
160 FOR GG=1 TO MM-1                              <172>
170 IF MID$(DD$(GG),4,16)<MID$(DD$(GG+1),4,
    16)THEN 190                                   <054>
180 HH$=DD$(GG):DD$(GG)=DD$(GG+1):DD$(GG+1)=HH$
    :FF=1                                         <049>
190 NEXT GG                                       <206>
200 IF FF THEN FF=0:GOTO 160                      <078>
210 II=MM                                         <176>
220 FOR MM=1 TO II:DD$=DD$(MM):GOSUB 2000
    :NEXT MM                                      <030>
230 END                                           <102>
```

Listing 1. Dieses Listing fehlte in Ausgabe 11/84

Wollen wir also in dieses System einsteigen, um dort eigene Programme ausführen zu lassen, so ist es natürlich unerläßlich, daß wir die »Spielregeln« dieses Prozessorsystems genau kennen, da es sonst leicht zu kleinen Katastrophen kommen kann.

Zu Ihrer weiteren Arbeit mit der 1541 noch ein paar Tips: Wenn Sie vorhaben, Programme in der Floppy ablaufen zu lassen, sollten Sie Ihre Floppy öffnen und ohne Deckel betreiben. So können Sie genau beobachten, wie der Kopf positioniert wird und was bei Lesefehlern geschieht. Sie werden unter anderem auch entdecken, daß Disketten nicht etwa auf der Seite beschrieben werden, auf der sich das Etikett befindet, sondern auf der Rückseite. Dies ist um so bemerkenswerter, als man eine Diskette immer nur auf der Vorderseite schonend behandelt, die ja eigentlich nicht benutzt wird. Auch wir mußten die Erfahrung machen, daß wir Disketten lange Zeit mit der wertvollen Seite auf Tische gelegt haben, stets darauf achtend, daß ja kein Staubkorn auf die von uns so gehütete Vorderseite kam.

```
0 GOTO 10                                          <234>
1 , 0300 AD 00 1C LDA $1C00                        <018>
2 , 0303 29 10    AND #$10                         <023>
3 , 0305 4A       LSR                              <093>
4 , 0306 8D 00 1C STA $1C00                        <041>
5 , 0309 4C 00 03 JMP $0300                        <005>
6 :                                                <064>
10 OPEN 1,8,15                                     <208>
20 FOR X=0 TO 11:READ A                            <215>
30 PRINT#1,"M-W"CHR$(X)CHR$(3)CHR$(1)CHR$(A)
   :NEXT                                           <065>
40 PRINT#1,"M-E"CHR$(0)CHR$(3)                     <179>
50 DATA 173,0.28,41,16,74,141,0,28,76,0,3          <005>
```

Listing 2. Unser erstes Floppy-Maschinenprogramm

Disketten werden auf ihrer Rückseite beschrieben!

Das Betreiben des Laufwerks ohne Deckel hat auch den Vorteil besserer Wärmeableitung. Die ICs werden es Ihnen danken.

Nachdem Sie Ihre 1541 also auf »Arbeitsbetrieb« getrimmt haben, wollen wir gleich einmal mit kleinen Programmen beginnen. In Tabelle 1 sehen Sie eine Aufstellung einiger wichtiger Zeropageadressen, die uns im weiteren Verlauf noch beschäftigen werden. Für ein DOS-Listing ist in unserer Serie natürlich kein Platz vorhanden; auch können wir nur mit kleinen Beispielen versuchen, Ihnen die Programmierung der Floppy nahezubringen. Für diejenigen unter Ihnen, die jedoch vorhaben, tiefer in die Floppyprogrammierung einzusteigen, sei an dieser Stelle ein Buch angesprochen, das voraussichtlich im Februar 1985 von Markt & Technik herausgegeben wird. Es behandelt die 1541 bis ins kleinste Detail, ist unter anderem mit einem ausführlichst kommentierten DOS-Listing ausgestattet und geht weit über das in dieser Reihe besprochene hinaus.

Programmieren der Floppy

So, jetzt soll es aber endlich losgehen. Wir wollen unser erstes Programm schreiben und in der Floppy ablaufen lassen.

Es handelt sich um Listing 2. Dieses »Miniprogramm« schreiben wir in den Puffer 0 der Floppy, das heißt ab Adresse $0300. Das Basic-Programm haben wir der Kürze halber gleich an den Assemblercode angehängt. Wenn Sie das Programm starten, wird das Bit abgefragt, das beim DC für den Zustand der Schreibschutzplakette verantwortlich ist. Sie werden vielleicht wissen, daß die Floppy die Schreibschutzkerbe bei den Disketten mit Hilfe einer Lichtschranke abfragt. Ist die Lichtschranke unterbrochen, das heißt es liegt eine Diskette mit Schreibschutzaufkleber im Laufwerk, dann steht das entsprechende Bit auf 0.

Unser Programm schiebt nun einfach das Bit der Lichtschranke an die Stelle des Bits für die rote LED und speichert diesen Wert wieder ab. Starten Sie einmal unser kleines Programm, dann werden Sie feststellen, daß die Leuchtdiode am Laufwerk erlischt, wenn die Lichtschranke unterbrochen wird. Holen Sie die Diskette dagegen aus dem Laufwerk oder legen Sie eine Diskette ohne Schreibschutzplakette ein, so beginnt die rote LED zu leuchten.

Mit diesem Programm können Sie also testen, ob von Ihnen selbst angefertigte Schreibschutzkerben in der Diskettenhülle an der richtigen Stelle liegen, um eine Diskette eventuell doppelseitig benutzen zu können.

Da unser Programm aus einer Endlosschleife besteht, können Sie die Floppy nur durch einen RESET wieder in einen ansprechbaren Zustand versetzen.

Das Programm hat aber einen Schönheitsfehler; es beeinflußt nämlich nicht nur die beiden LED-Bits in Speicherstelle $1C00, sondern löscht bei jedem Durchgang auch alle anderen Bits dieses Registers, deren Belegung Sie Tabelle 2 entnehmen können. Für unsere Testzwecke ist diese »Pfuscherei« jedoch unwesentlich.

Der »&«-Befehl

Nach diesem aufregenden Beispiel wollen wir Sie nun mit einem Befehl bekanntmachen, den Sie sehr wahrscheinlich noch nicht kennen. Er nennt sich »&« und wird unverständlicherweise in noch keinem uns bekannten Buch beschrieben. Der &-Befehl entspricht in gewisser Weise einem BLOCK-EXECUTE-Befehl; auch hier wird ein Programm von Diskette geladen und sofort ausgeführt.

Der Unterschied besteht nur darin, daß mit dem &-Befehl nicht nur ein Block, sondern ein ganzes File, das im Directory verzeichnet ist, geladen und im Puffer als Programm ausgeführt wird.

Außerdem müssen die Files, die mit dem Befehl »&« gestartet werden sollen, speziell gekenn-

```
100 REM ERZEUGT &-FILE, DAS LISTING 2              <220>
110 REM (LED-TEST) ENTSPRICHT.                     <194>
120 :                                              <178>
130 DATA 0,7,12,173,0,28,41,16,74,141              <093>
140 DATA 0,28,76,0,7,93                            <197>
150 OPEN 1,8,2,"&,U,W"                             <194>
160 FOR X=1 TO 16:READ A                           <105>
170 PRINT#1,CHR$(A);:NEXT X                        <071>
180 CLOSE 1                                        <133>
```

Listing 3. So macht man Listing 2 zu einem »&-File«

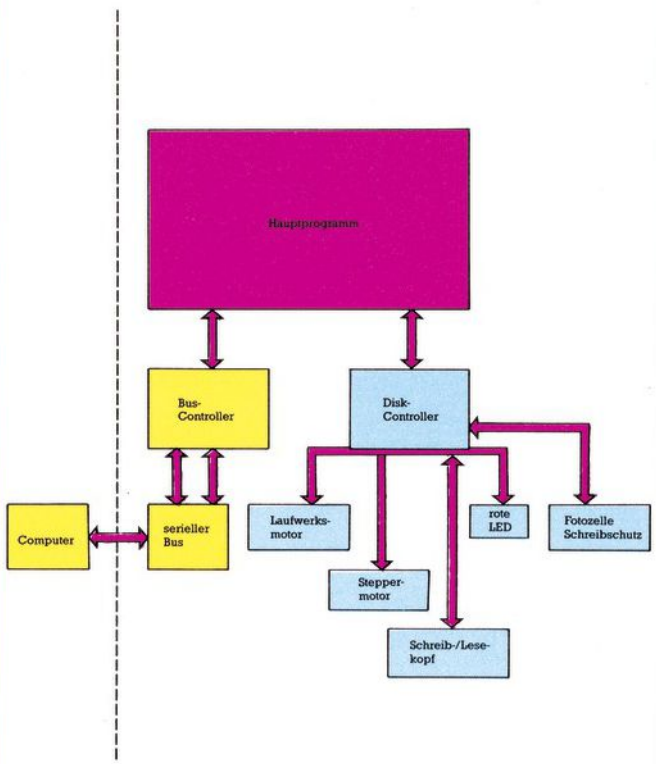

Bild 3. Schematischer Aufbau der 1541

Tabelle 1. Die wichtigsten Zeropageadressen der Floppy

```
$0000     Jobspeicher für Puffer 0
$0001     Jobspeicher für Puffer 1
$0002     Jobspeicher für Puffer 2
$0003     Jobspeicher für Puffer 3
$0004     Jobspeicher für Puffer 4
$0005     Jobspeicher für Puffer 5 (im RAM nicht vorhanden)
$0006/7   Spur- und Sektornummer für Befehl in Puffer 0
$0008/9   Spur- und Sektornummer für Befehl in Puffer 1
$000A/B   Spur- und Sektornummer für Befehl in Puffer 2
$000C/D   Spur- und Sektornummer für Befehl in Puffer 3
$000E/F   Spur- und Sektornummer für Befehl in Puffer 4
$0010/1   Spur- und Sektornummer für Befehl in Puffer 5
$0012/3   ID der Diskette im ASCII-Code; die beiden Zeichen der
          aktuellen ID werden bei jedem Blocksuchbefehl gelesen
          und hier aktualisiert abgespeichert. Auch das Initia-
          lisierkommando benutzt diesen Befehl und bringt die ID
          dadurch auf den neuesten Stand.
$0016-    Hier sind die Bytes für den aktuellen Blockheader ge-
$001A     speichert, und zwar sind dies:
            $0016  erstes Zeichen der ID
            $0017  zweites Zeichen der ID
            $0018  Spurnummer des Blocks
            $0019  Sektornummer des Blocks
            $001A  Prüfsumme über den Blockheader
          Auf der Diskette stehen diese Werte in der umgekehrten
          Reihenfolge!
$001C     Flag für Änderung beim Schreibschutz der Diskette
$002E/F   Zwischenspeicher für aktuelle Zeiger
$0030/1   Zeiger in aktuellen Puffer
$0032/3   Zeiger auf aktuellen Blockheader beim Schreiben
$0038     Kennzeichen ($07) für Beginn eines Datenblocks
$0039     Kennzeichen ($08) für Beginn eines Blockheaders
$003A     Zwischenspeicher für Prüfsummen
$003D     aktuelle Laufwerksnummer; bei der VC 1541 immer 0
$003E     gerade arbeitendes Laufwerk ($FF = kein Laufwerk)
$003F     Puffernummer des eben ausgeführten Befehls (0-5)
$0043     zählt die Anzahl der Sektoren bei der Formatierung
$0044     Zwischenspeicher beim Arbeiten
$0045     Zwischenspeicher für aktuellen Befehlscode
$0047     enthält aktuelles Kennzeichen für Beginn eines Daten-
          blocks. wird nur bei RESET einmal auf $07 gesetzt und
          kann vom Benutzer verändert werden, wobei das Hi-Nybble
          des Wertes immer auf 0 ($0-) stehen sollte, um Lesepro-
          bleme des DC zu vermeiden. Wird versucht, einen Daten-
          block mit einer anderen, als der hier gespeicherten,
          Nummer zu lesen, so erfolgt der Fehlercode $04 des DC
          und die Floppy sendet Fehlermeldung Nummer 22 zum Bus.
$0049     Zwischenspeicher für den Stackpointer
$004A     Zähler für Kopftransport; Zahlen bis 127 bewegen den
          Kopf nach außen; Zahlen von 128 bis 255 bewegen ihn
          nach innen (höhere Spurnummer).
$0051     aktuelle Spurnummer bei der Formatierung; steht auf
          $FF, wenn keine Formatierung erfolgt.
$0065/6   Zeiger auf die NMI-Routine; wird bei einem RESET ge-
          stellt.
$0067     Flag zum Anzeigen eines NMI
$0068     Flag zum Ermöglichen (0) oder Sperren (1) der automa-
          tischen Initialisierung einer Diskette, falls ein
          ID Type Mismatch Error erkannt wurde
$0069     Abstand der Sektoren bei der Zuteilung; erhält bei
          einem RESET den Wert 10.
$006A     Anzahl der Leseversuche eines Sektors; steht nach
          RESET auf 5.
$006B/C   Zeiger auf Sprungtabelle der USER-Befehle; steht nor-
          malerweise auf $FFF6 nach einem RESET.
$006D/E   Zeiger auf den Beginn der 'Bit Map'; steht auf $0400
          und wird beim Initialisieren gesetzt.
$006F     Zwischenspeicher; steht nach RESET auf $6F
$0070     Zwischenspeicher
$0071     Zwischenspeicher
$0072     Zwischenspeicher; steht nach RESET auf $FF
$0073     Zwischenspeicher
$0074     Zwischenspeicher
$0075/6   Indirekter Zeiger auf $0100; wird bei RESET gestellt
$0077     Gerätenummer + $20 für das LISTEN-Kommando
$0078     Gerätenummer + $40 für das TALK-Kommando
$0079     Flag für LISTEN (1/0)
$007A     Flag für TALK (1/0)
$007B     Flag für Adressierung
$007C     Flag für ATN-Signal vom seriellen Bus
$007D     Flag für Prozessor im ATN-Modus
$007F     Aktuelle Laufwerksnummer; hier immer 0
$0080     Aktuelle Spurnummer; enthält $00 nach Ausführung
$0081     Aktuelle Sektornummer; enthält $00 nach Ausführung
$0082     Aktuelle Kanalnummer
$0083     Aktuelle Sekundäradresse
$0084     übliche Sekundäradresse
$0085     Aktuelles Datenbyte
$0086     Speicher für Zwischenergebnisse
$0087     Speicher für Zwischenergebnisse
$0088     Speicher für Zwischenergebnisse
$0089     Speicher für Zwischenergebnisse
$008A     Speicher für Zwischenergebnisse
$008B-    Speicher für Ergebnisse bei Berechnungen
$008E
$008F-    Akkumulator für Berechnungen
$0093
$0094/5   Zeiger auf Directory-Puffer; enthält $05/02
$0096     Kommando vom IEEE-Bus; hier unbenutzt
$0098     Bitzähler für seriellen Bus
$0099/A   Buffer-Pointer für Puffer 0; steht auf $0300
$009B/C   Buffer-Pointer für Puffer 1; steht auf $0400
$009D/E   Buffer-Pointer für Puffer 2; steht auf $0500
$009F/0   Buffer-Pointer für Puffer 3; steht auf $0600
$00A1/2   Buffer-Pointer für Puffer 4; steht auf $0700
          Alle diese Pointer werden durch den B-P-Befehl ver-
          ändert!
$00A3/4   Zeiger auf nächstes Zeichen im INPUT-BUFFER ($0200)
$00A5/6   Zeiger auf nächstes Zeichen im ERROR-BUFFER ($02D6)
$00A7-    Tabelle; enthält für jeden aktiven Puffer die entspre-
$00AD     chende Kanalnummer. Kanalnummer = $FF, wenn Puffer
          unbenutzt.
$00AE-    Tabelle; enthält für jeden aktiven Puffer die entspre-
$00B4     chende Kanalnummer. Kanalnummer = $FF, wenn Puffer
          unbenutzt.
$00B5-    Tabelle der Lo-Bytes der Recordnummern für jeden Puffer
$00BA
$00BB-    Tabelle der Hi-Bytes der Recordnummern für jeden Puffer
$00C0
$00C1-    Tabelle der nächsten zu bearbeitenden Recordnummern
$00C6     für jeden Puffer
$00C7-    Tabelle der Recordlängen für jeden Puffer
$00CC
$00CD-    Tabelle der Side-Sektoren für jeden Puffer
$00D2
$00E2-    Standardwerte für Laufwerk; hier alle 0
$00E6
$00E7-    Tabelle der Filetypen
$00EB
$00EC-    Kanal Filetyp
$00F1
$00F2-    Kanalstatus
$00F7
$00F8     Zwischenspeicher für EOI
$00F9     Aktuelle Puffernummer für Befehlscode
$0101     Formatkennzeichen von Spur 18 Sektor 0
$0104-    Bereich des Hardware-Stack; nicht benutzbar
$0145
$0200-    INPUT-BUFFER; hier werden alle Befehlsstrings vom Com-
$0229     puter zwischengespeichert und nach Syntaxprüfung aus
          geführt
$022A     Codenummer des auszuführenden Befehls
$022B-    Kanaltabelle; diese Tabelle enthält für jede mögliche
$023E-    Aktuelles Datenbyte für jeden Kanal; Belegung der
$0243     Adressen wie bei der Kanalstatustabelle ($022B)
$0244-    Tabelle der Zeiger auf das letzte aktuelle Zeichen in
$0249     jedem, für den Kanal zuständigen, Pufferspeicher
$024A     Gerade behandelter Filetyp
$024B     Länge des Befehlsstrings
$024C     Zwischenspeicher für Sekundäradresse
$024D     Zwischenspeicher für Befehlscode
$024E     Arbeitsspeicher beim Suchen des nächsten Sektors
$024F/0   Pufferbelegungsspeicher; 1 = Puffer belegt
$0253     Flag für Directory-Eintrag gefunden
$0254     Flag für $-Befehl zum Listen des Directory
$0255     Flag für Befehlsausführung (<>$00, wenn Befehl anliegt)
$0257     Nummer des letzten benutzten Puffers
$0258     Recordlänge
$0259     Side-Sector Spur
$025A     Side-Sector Sektor
$025B-    Tabelle; enthält den letzten Befehlscode der Puffer
$025F
$0260-    Sektornummern der Directoryeinträge in den Puffern
$0265
$0266-    Zeiger auf die Directoryeinträge in den Puffern
$026B
$026D     Flag für LED Blinken bei Fehler
$026E     Nummer des letzten aktiven Laufwerks
$026F     Nummer des letzten bearbeitenden Sektors
$0270     aktueller Schreibkanal
$0271     aktueller Lesekanal
$0274     Länge des Befehlsstrings im INPUT-BUFFER
$027A-    Tabelle der Zeiger auf die Filenamen
$027F
$0280-    Spurnummern der Files für den aktuellen Puffer
$0284
$0285-    Sektornummern der Files für den aktuellen Puffer
$0289
$028A     Joker (*) Flag
$028E     Standardwert für die Nummer des Laufwerks
$028F     Flag für Fileeintrag im Directory gefunden
$0290     Sektornummer des aktuellen Directory Sektors
$0291     Sektornummer des ersten Directoryeintrags
$0292     Zeiger auf ersten gültigen Directoryeintrag
$0293     Zeigt letzten Block an; enthält dann 0
$0294     Aktueller Pufferzeiger
$0295     Zähler für Fileeinträge
$0297     Betriebsart des aktuellen Files (Lesen/Schreiben)
$029D/E   Spurnummer der BAM
$02A1-    Zwischenspeicher für BAM Eintragungen
$02B0
$02B1-    Puffer für Directory
$02D4
$02D5-    ERROR-BUFFER; enthält auszugebende Fehlermeldung
$02F8
$02FA     Lo-Byte der Anzahl der freien Blocks auf Diskette
$02FC     Hi-Byte der Anzahl der freien Blocks auf Diskette
$0300-    Puffer 0
$03FF
$0400-    Puffer 1
$04FF
$0500-    Puffer 2
$05FF
$0600-    Puffer 3
$06FF
$0700-    Puffer 4 (enthält normalerweise die BAM)
$07FF
$0800-    Nicht mit RAM belegt
$FFFF
```

```
100 REM AUTO-'&'-MAKER                          <106>
110 REM --------------                          <115>
120 REM                                         <007>
130 REM  03.11.84.  BORIS SCHNEIDER             <224>
140 :                                           <198>
150 :                                           <208>
160 REM INITIALISIERUNG                         <159>
170 INPUT"STARTADRESSE DES &-FILES";SA          <121>
180 INPUT"NAME DES &-FILES";NA$                 <046>
190 IF LEN(NA$)>15 THEN 180NA$                  <026>
200 OPEN 1,8,2,"&"+NA$+",U,W"                   <063>
210 DIM X(256)                                  <158>
220 PRINT"BITTE GEBEN SIE JETZT IHRE DATEN EIN"
                        <116>
230 PRINT"ABSCHLUSS MIT -1!"                    <212>
240 :                                           <042>
250 REM DATENEINGABE UND TEST AUF               <227>
260 REM UEBERLAUF                               <047>
270 Y=1                                         <075>
280 INPUT X(Y)                                  <160>
290 IF X(Y)<0 THEN Y=Y-1:GOTO 350               <213>
300 PR=PR+X(Y):IF PR>255 THEN PR=PR-255         <103>
305 Y=Y+1:IF Y>254 THEN 350                     <026>
310 GOTO 280                                    <090>
320 :                                           <123>
330 REM ABSPEICHERN DER VORHANDENEN             <017>
340 REM DATEN IN DAS USR-FILE                   <006>
350 SH=INT(SA/256)                              <144>
360 SL=SA-256*SH                                <221>
370 PR=PR+SH+SL+Y                               <250>
380 PRINT#1,CHR$(SL);CHR$(SH);                  <082>
390 PRINT#1,CHR$(Y);                            <040>
400 FOR I=1 TO Y                                <059>
410 PRINT#1,CHR$(X(I));                         <213>
420 NEXT                                        <039>
430 PR=PR-(255*INT(PR/256))                     <219>
440 PRINT#1,CHR$(PR);                           <163>
450 IF X(Y+1)<0 THEN GOTO 470                   <217>
460 SA=SA+Y:PR=0:GOTO 270                       <196>
470 CLOSE 1                                     <168>
```

Listing 4. Komfortable »&-Files« erzeugen

zeichnet sein. Sie enthalten als erstes Zeichen im Filenamen das Zeichen »&«. Soll also zum Beispiel ein File mit dem Namen »Test« als Autostartprogramm in der Floppy ausgeführt werden, so geben Sie diesem File den Namen »&Test« und starten Sie es danach mit:
OPEN1,8,15,"&TEST"

Haben Sie nur ein einziges Autostartfile auf Diskette, so können Sie es auch nur mit »&« abspeichern und ebenso mit
OPEN1,8,15,"&" starten.

Leider erwartet die Floppy von Autostartfiles eine spezielle Syntax, die in Tabelle 3 zu sehen ist.

Als Listing 3 haben wir noch einmal unser LED-Testprogramm; nur wird diese Routine durch das Basic-Programm als &-File auf Diskette geschrieben und kann danach durch den schon erwähnten Befehl direkt von Diskette in den Pufferspeicher geschrieben und dort gestartet werden.

Zu Tabelle 3 noch einige Anmerkungen:

Zuerst muß die Startadresse des Programms im Pufferspeicher der Floppy in das File geschrieben werden. Danach folgt die Anzahl der Bytes im Programm. Jetzt werden die Programmbytes abgespeichert, und schließlich folgt noch eine Prüfsumme, die sich wie folgt errechnet:

Es werden alle Bytes des Programms addiert und zum Ergebnis noch die zwei Bytes der Startadresse und die Anzahl der Bytes im Programm hinzugezählt. Dieses Ergebnis ist als Integerzahl zu verstehen und besteht also aus einem niederwertigen (LO) und einem höherwertigen (HI) Byte. Das niederwertige Byte ist die Prüfsumme, zu der noch der Übertrag im höherwertigen Byte addiert werden muß. Diese Berechnung klingt kompliziert; ist es aber nicht. In Listing 4 wird Ihnen diese Rechnerei abgenommen. Die allgemeine Formel hier noch einmal:
HB = INT(SUMME/256)
LB = SUMME—HB*256
dabei bedeuten:
HB — das höherwertige Byte
LB — das niederwertige Byte
SUMME — die Gesamtsumme der Programmbytes

Achtung: Die Übertragsberechnung muß nach jedem neu dazugezählten Wert erfolgen, da das Endergebnis kleiner als 256 sein muß! Wie Sie sehen, ist das Anlegen eines &-Files nicht ganz einfach. Bisher wurde diese Fileart fast nur von Profis zum Programmschutz angewandt, da sie, wie schon erwähnt, nahezu unbekannt war.

Zu erwähnen wären noch zwei seltsame Fehlermeldungen der Floppy:

»OVERFLOW IN RECORD« erscheint, wenn die Anzahl der tatsächlichen Bytes mit der Angabe nicht übereinstimmt.

»RECORD NOT PRESENT« erscheint, wenn die Prüfsumme nicht stimmt.

Da wir stets darum bemüht sind, Ihnen die Arbeit mit der Floppy so angenehm wie möglich zu machen, haben wir unserem Artikel noch Listing 4 beigefügt. Es handelt sich hier um ein Programm, das es Ihnen gestattet, auf einfachste Weise &-Files zu erstellen. Diese können sogar länger als 256 Byte sein, da das Programm dann automatisch eine Prüfsumme und die Anschlußadresse einfügt. Ununterbrochene &-Files, die länger als 256 Zeichen sind, kann es ja nicht geben, da die Anzahl der Programmbytes im File nur in einem Byte abgespeichert wird.

Mit dieser neuen Fileart wollen wir Sie für dieses Mal entlassen. Ruhen Sie sich für die nächste Folge aus. Wir werden dann auf die Technik der Jobschleifenprogrammierung eingehen, die Ihnen eine Fülle von Anwendungen eröffnen wird.

(K. Schramm/B. Schneider/gk)

DISKCONTROLLER (DC)

VIA 6522, $ 1800, PORT B

Bit #	Bedeutung
0	DATA IN
1	DATA OUT
2	CLOCK IN
3	CLOCK OUT
4	ATN OUT
5 }	GERÄTENUMMER
6 }	
7	ATN IN (CB 2)

BUSCONTROLLER (BC)

VIA 6522, $1C00, PORT B

Bit #	Bedeutung
0	Steppermotor für Laufwerk 1 (n.v.)
1	Steppermotor für Laufwerk 0
2	Laufwerksmotor
3	LED am Laufwerk (rot)
4	Schreibschutzkennung
5	Bitsynchronisation für DC bei den vier
6	Spurbereichen
7	SYNC-Signal

Tabelle 2. Belegung der beiden Controll-Ports der 1541

Byte	Bedeutung
1-2	Startadresse in der 1541 im HI/LO-Format
3	Anzahl der folgenden Programmbytes
4-N	Programm
N+1	Prüfsumme
N+2	Hier kann bei längeren Programmen ein weiterer Teil eingefügt werden. Format: wieder bei Byte 1 beginnend.

Tabelle 3. Aufbau eines &-Files. In dieser Tabelle sind die Linker- beziehungsweise Endekennzeichen, die in den ersten beiden Bytes eines Datenblocks stehen, nicht enthalten, da sie beim Öffnen und Beschreiben eines &-Files automatisch gesetzt werden.