In die Geheimnisse der Floppy eingetaucht (Teil 3)

In den letzten beiden Folgen sind die Möglichkeiten, die Basic bietet, ausgeschöpft worden. Heute soll deswegen die Bedienung der Floppy in Maschinensprache im Vordergrund stehen.

Wenn wir in dieser Folge von Routinen sprechen, die im Betriebssystem stehen, so werden wir die in Tabelle 1 dargestellten Kürzel verwenden, die Sie übrigens auch in Editorprogrammen gut benutzen können.

FILPAR und FILNAM

Bei OPEN, LOAD und ähnlichen Befehlen müssen Sie entsprechenden Routinen mitteilen, welches File Sie wo öffnen wollen. Um Ihnen eine »Herumwurstelei« in der Zeropage zu ersparen, wo Sie die einzelnen Angaben von Hand setzen müßten, hat das Betriebssystem zwei entsprechende Routinen implementiert. FILPAR setzt für Sie die einzelnen Fileparameter. Diese müssen der Routine in den Prozessorregistern übergeben werden:

— Filenummer (Akku)
— Geräteadresse (X-Register)
— Sekundäradresse (Y-Register)

Ein Beispiel:
Sie wollen für ein File mit der Nummer 1, der Geräteadresse 8 und der Sekundäradresse 15 (Kommandokanal der Floppy) die entsprechenden Fileparameter setzen:

```
LDA   #$01      ; Filenummer 1
LDX   #$08      ; Geräteadresse 8
LDY   #$6F      ; Sekundäradresse + $60
JSR   #FILPAR   ; Fileparameter setzen
```

Wie Sie sehen, muß zu der betreffenden Sekundäradresse ein Wert von $60 addiert werden.

Aber in vielen Fällen müssen Sie ja auch einen Filenamen angeben. Dazu dient die FILNAM-Routine. Hier erfolgt die Parameterübergabe:

— Länge des Filenamens (Akku)
— Adresse LO des Namens (X-Register)
— Adresse HI des Namens (Y-Register)

Und wieder ein Beispiel. Um das Directory-File mit dem Namen »$« anzusprechen, geben Sie bitte folgende Befehle ein:

```
LDA   #$24      ; Code für '$' in Akku
STA   $FF       ; und abspeichern
LDA   #$01      ; Länge des Filenamens
LDX   #FF       ; Adresse LO
LDY   #$00      ; Adresse HI
JSR   FILNAM    ; übergeben
```

Sie müssen also wissen, wo der Filename im Speicher steht und wie lang er ist. Dies ist aber im allgemeinen kein Problem. Auf die gleiche Weise können Sie der Floppy über den Kommandokanal auch Befehle senden, wie Sie in der letzten Folge vorgestellt wurden. Das entspräche der Basic-Sequenz:
OPEN x, 8, 15, "befehl"

Natürlich können Sie auch alle Parameter von Hand setzen, beziehungsweise noch einmal lesen. Wo sich die einzelnen Parameter in der Zero-Page nach Ausführung dieser und der anderen Routinen befinden, ist in Tabelle 2 angegeben.

OPEN und CLOSE

Nachdem wir alle Fileparameter und den Filenamen übergeben haben, können wir die OPEN-Routine mit JSR OPEN aufrufen. Schon ist das entsprechende File geöffnet. Zu beachten wäre folgendes: Es können im Computer niemals mehr als 10 Files gleichzeitig geöffnet sein!

Die CLOSE-Routine arbeitet analog zu OPEN, mit der Ausnahme, daß nur die Filenummer übergeben werden muß. Geräteadresse und Sekundäradresse sucht sich der C 64 aus einer Tabelle heraus, auf die wir später noch zu sprechen kommen:

```
LDA   #$01      ; Filenummer
JSR   CLOSE     in Akku
```

Der Filename wird beim Schließen überhaupt nicht mehr benötigt.

LISTEN und UNLIST, TALK und UNTALK

Nach dem Öffnen eines Files kann die Datenübertragung noch nicht beginnen. Sie müssen dem entsprechenden Gerät zuerst mitteilen, ob es senden oder empfangen soll.

Bestes Beispiel ist wieder der Kommandokanal. Über diesen kann die Floppy sowohl Befehle empfangen, als auch Fehlermeldungen senden. Um ein Gerät zum Empfangen zu veranlassen, verwenden wir die Routine LISTEN. Das hat nichts mit dem Basic-Befehl LIST zu tun, sondern kommt vom englischen Wort für »Hören«. Beim Aufruf vom LISTEN ist das angesprochene Gerät auf Empfang und der Computer auf Senden eingestellt.

Wichtig ist, daß der Akku beim Aufruf die Geräteadresse enthält. Dies gilt für alle vier hier beschriebenen Routinen. Wenn Sie mit dem Senden der Daten fertig sind, sollten Sie ein UNLISTEN zum entsprechenden Gerät schicken, um dieses wieder freizugeben. Dies geschieht mit Hilfe der UNLIST-Routine. Analog verhält es sich mit den Routinen TALK und UNTALK. Sie veranlassen das angesprochene Gerät, Daten zu senden, beziehungsweise mit dem Senden aufzuhören und wieder in den Wartezustand zurückzukehren.

SECTLK und SECLST

Die beiden Routinen SECTLK und SECLST sind ebenfalls sehr wichtig für die Datenübertra-

Auflistung aller verwendeten ROM-Routinen

Kürzel	Adresse	SECTLK	$FF96
		SECLST	$FF93
FILPAR	$FFBA	IECOUT	$FFA8
FILNAM	$FFBD	IECIN	$FFA5
OPEN	$FFC0	FILTAB	$F30F
CLOSE	$FFC3	FILSET	$F31F
LISTEN	$FFB1	LOAD	$FFD5
UNLIST	$FFAE	SAVE	$FFD8
TALK	$FFB4	BASOUT	$FFD2
UNTALK	$FFAB	CLALL	$FFE7

Tabelle 1. Die im Artikel erwähnten Betriebssystemroutinen

Wichtige Zeropageadressen	
Adresse	Bedeutung
$90	Status-Flag
$93	Flag für LOAD/VERIFY
$98	Anzahl der offenen Files
$99	Eingabegerät für BASIN
$9A	Ausgabegerät für BASOUT
$B7	Länge Filename
$B8	aktive Filenummer
$B9	Sekundäradresse
$BA	Geräteadresse
$BB/BC	Zeiger auf Filenamen

Tabelle 2. Dies sind die Zeropageadressen unter denen die aktuellen Fileparameter gespeichert werden

```
(In $FA/$FB muß die Adresse, in $FC
 die Länge des Befehlsstrings
 stehen.)

   LDA #$01    ;  Filenummer
   LDX #$08    ;  Gerätenummer
   LDY #$6F    ;  Sekundäradresse
   JSR FILPAR  ;  setzen.
   LDA #$00    ;  Länge Filename =0
   JSR FILNAM  ;  da kein Filename.
   JSR OPEN    ;  File öffnen
   LDA #$08    ;  Geräteadresse
   JSR LISTEN  ;  auf Empfangen
   LDA #$6F    ;  Sekundäradresse
   JSR SECLST  ;  senden.
   LDY #$00    ;  Zähler auf Null
 1 LDA (FA),Y  ;  Befehlsbyte laden
   JSR IECOUT  ;  und übertragen
   INY         ;  Zähler erhöhen
   CMP  $FC    ;  Befehlslänge
   BNE 1       ;  noch ein Byte ?
   LDA #$08    ;  Geräteadresse
   JSR UNLIST  ;  Sendung beenden
   LDA #$01    ;  Filenummer
   JSR CLOSE   ;  Schliessen
   RTS
```

Listing 1. So können Befehlsstrings an die Floppy gesendet werden

gung. Denn obwohl wir beim OPEN-Befehl eine Sekundäradresse angeben, muß diese bei jeder weiteren Übertragung nochmals an das aktuelle Gerät gesendet werden. Dies hat zwei Gründe: Einerseits können Sie ja mehrere Floppykanäle gleichzeitig geöffnet halten. Damit die Floppy nun weiß, für welchen Kanal der nächste Schwung von Daten bestimmt ist, beziehungsweise, welcher Kanal senden soll, muß nach dem Aufruf von TALK SECTLK, beziehungsweise nach dem Aufruf von LISTEN SECLST durchgeführt werden. Außerdem merkt sich der Computer zwar die angegebene Sekundäradresse, sendet sie aber nicht. Dies hat praktische Gründe, wie wir noch bei den LOAD/SAVE-Routinen sehen werden. SECTLK und SECLST benötigen die jeweilige Sekundäradresse +$60 im Akku. Diese kann, wie in unseren Beispielen, direkt geladen oder aber auch der entsprechenden Zero-Page-Adresse entnommen werden.

IECOUT und IECIN

Nachdem wir nun endlich alle Vorbereitungen getroffen haben, können wir munter Bytes von der Floppy zum Computer und umgekehrt übertragen. Dies ist mit den ROM-Routinen denkbar einfach. IECOUT überträgt das im Akku befindliche Byte an das aktuelle Gerät; IECIN empfängt eines und legt es im Akku ab.

Busfehlerbehandlung

Bei aller Sorgfalt; Fehler können immer auftreten, so auch beim Busbetrieb. Um einen in einer Busroutine aufgetretenen Fehler zu signalisieren, verwendet das Betriebssystem das Carry-Flag. Generell gilt: Ist das Carry-Flag gesetzt, so ist etwas nicht in Ordnung, und wir sollten das Statusbyte überprüfen. Dieses Statusbyte steht in der Speicherstelle $90. Immer wenn es ungleich Null ist, liegt irgendein Sonderfall vor. Jedes Bit des Statusbytes hat eine andere Funktion; Tabelle 3 zeigt diese Belegung. Ist zum Beispiel das Bit 7 gesetzt, so ist das angesprochene Gerät entweder nicht vorhanden oder abgeschaltet. In Basic bekämen wir in einem solchen Fall die Meldung »DEVICE NOT PRESENT ERROR«. Interessant ist für uns noch das Bit 6. Ist es gesetzt, so bedeutet das, daß das letzte Byte der angeforderten Informationen übertragen wurde. Dies können wir uns auch in Basic zunutze machen, um beispielsweise die Fehlermeldung der Floppy auszulesen:

```
10 OPEN1,8,15
20 GET#1, A$: PRINTA$;: IF
ST<>64 THEN 20
30 CLOSE1
```

Wie Sie an diesem Beispiel sehen, ist der Inhalt der Speicherstelle $90 in der Variablen ST ►

```
Prinzip des Lesens des Fehlerkanals
mit Ausgabe auf dem Bildschirm.

   LDA #$00    ;  Zurücksetzen des
   STA  $90    ;  Status-Flags
   LDA #$01    ;  Filenummer
   LDX #$08    ;  Geräteadresse
   LDY #$6F    ;  Sekundäradresse
   JSR FILPAR  ;  setzen.
   LDA #$00    ;  Länge Filename =0
   JSR FILNAM  ;  setzen.
   JSR OPEN    ;  File öffnen
   LDA #$08    ;  Geräteadresse auf
   JSR TALK    ;  Senden schalten
   LDA #$6F    ;  Sekundäradresse
   JSR SECTLK  ;  übertragen
 1 JSR IECIN   ;  Byte empfangen
   JSR BASOUT  ;  und ausgeben
   BIT  $90    ;  Bit 6 Status =0?
   BVC 1       ;  dann noch ein Byte
   LDA #$08    ;  Geräteadresse
   JSR UNTALK  ;  Sendung beenden
   LDA #$01    ;  Filenummer
   JSR CLOSE   ;  und schliessen
   RTS
```

Listing 2. So läßt sich der Fehlerkanal auslesen und anzeigen

Das Status-Flag	
Bit	Bedeutung wenn gesetzt
1	Fehler (Zeitüberschreitung) bei IEC-Eingabe
2	Fehler (Zeitüberschreitung) bei IEC-Ausgabe
3-5	nur für Kassettenbetrieb
6	Übertragung ist beendet
7	Gerät meldet sich nicht

Tabelle 3. Für uns wichtige Bits im Statusflag

```
Prinzip des Ladens von Programmen.
LDX #$08    ;  Geräteadresse
LDY #$00    ;  Sekundäradresse für
               relativ laden
JSR FILPAR ;  und setzen.
LDX # (Filename LO-Byte)
LDY # (Filename HI-Byte)
LDA # (Filename Länge)
JSR FILNAM
LDA #$00    ;  LOAD-Flag
LDX # (Startadresse LO-Byte)
LDY # (Startadresse HI-Byte)
JSR LOAD
RTS
```

Listing 3. Das Laden von Programmen an beliebige Adressen

enthalten. Vor jeder neuen Datenübertragung sollten Sie darauf achten, daß das Statusbyte gelöscht wird, da sonst irrtümlich Fehler festgestellt werden könnten. Zur Verdeutlichung des bisher Gesagten dienen die Listings 1 und 2, die jedoch nur Anhaltspunkte geben sollen. Sie sind weder perfekt noch eintippfertig und sollten auf den jeweiligen Bedarf abgestimmt werden.

Bearbeiten mehrerer Files

Sie werden festgestellt haben, daß wir bisher immer nur mit einem einzigen File gearbeitet haben. Was aber, wenn Sie gleichzeitig zwei Files offen halten müssen, zum Beispiel, um einen Block von Diskette zu lesen. Sie erinnern sich ja, daß wir dazu sowohl den Kommandokanal als auch einen Übertragungskanal benötigen. Wir könnten zwar jeweils, wenn wir den Kanal wechseln wollen, mit CLOSE den alten schließen und mit OPEN den neuen öffnen, aber es geht auch einfacher.

Voraussetzung ist, daß alle benötigten Files schon geöffnet sind. Dann kann mit Hilfe einer, schon erwähnten, Filetabelle zwischen — bis zu 10 — Files beliebig umgeschaltet werden. Diesen Zweck erfüllen die Routinen FILTAB und FILSET.

FILTAB benötigt im Akku die Nummer des Files, auf das Sie umschalten wollen. Die Routine sucht dann in der Filetabelle nach den entsprechenden anderen Parametern. Tritt hier ein Fehler auf, weil das File noch gar nicht geöffnet wurde, so wird das Zero-Flag gelöscht und es kann mit BNE auf einen Fehler überprüft werden.

FILSET schreibt dann die gefundenen Parameter in die entsprechenden Zero-Page-Adressen. Die komplette Routine zum Umschalten auf das File x lautet also: ►

```
LDA   #$ xx     ; Nummer des Files
JSR   FILTAB    ; Durchsuchen der Tabelle
BNE   ERROR     ; Fehler ?
JSR   FILSET    ; Parameter setzen
```

Die ERROR-Routine müssen Sie natürlich noch selbst schreiben. Danach ist das angewählte File zum aktuellen File geworden. Alle LISTEN, TALK und so weiter, beziehen sich jetzt auf dieses neue File.

In den Zero-Page-Adressen aus Tabelle 2 stehen nun die für dieses File aktuellen Parameter, da sie aus der großen Filetabelle automatisch übertragen werden. Eine Ausnahme bildet hier der Filename, da er nur beim Öffnen des Files benötigt wird.

Diese große Filetabelle befindet sich übrigens an den Speicherstellen $0259 bis $0276.

Denken Sie immer daran, vor einem erneuten Umschalten UNLIST oder UNTALK aufzurufen. CLOSE braucht dagegen erst aufgerufen zu werden, wenn die Bearbeitung eines Files völlig abgeschlossen ist.

LOAD und SAVE

Prinzipiell könnten Sie mit dem bisher Erwähnten auch schon Programme laden und speichern, allerdings nur sehr mühselig. Da unser Computer das aber schon von selbst beherrscht, geben wir ihm gern diese Arbeit ab.

Betrachten wir zunächst die LOAD-Routine. Auch hier muß wieder eine Vielzahl an Parametern übergeben werden. Mit FILPAR werden Gerätenummer und Sekundäradresse gesetzt. Eine Filenummer braucht nicht gesetzt zu werden. Für die Sekundäradresse gilt folgendes:

Ist sie gleich Null, so wird das Programm an eine, von Ihnen festgelegte, Speicherstelle geladen. Ist sie gleich Eins, so wird das Programm an die Speicherstelle geladen, an der es bei SAVE stand. Der erste Modus wird vom Betriebssystem ausgenutzt, um Programme ab $0800 zu laden, wenn beim LOAD-Befehl keine Sekundäradresse angegeben wird. Prinzipiell kann aber an jede beliebige Adresse geladen werden! Der Filename wird, wie gewohnt, mit FILNAM gesetzt. Vor dem Aufruf der LOAD-Routine treten zwei, uns neue, Parameter hinzu, die wie folgt übergeben werden:
LOAD/VERIFY Flag (Akku)
Ladeadresse LO (X-Register)
Ladeadresse HI (Y-Register)

Steht beim Aufruf der Routine im Akku 0, so wird geladen. Steht dort hingegen eine 1, so wird ein VERIFY durchgeführt.

Die Startadresse in den X/Y-Registern wird nur beachtet, wenn die Sekundäradresse gleich Null ist. Alles übrige erledigt die LOAD-Routine, und Sie brauchen nur noch deren Ende abzuwarten. Zur Sekundäradresse wäre noch folgendes zu bemerken:

Egal, was Sie für eine Adresse angeben, zur Floppy wird immer nur 0 gesendet. Wie Sie schon wissen, ist diese Sekundäradresse floppyintern für den LOAD-Befehl reserviert und darf nicht ohne weiteres bei OPEN-Befehlen verwendet werden. Nach Beendigung der LOAD-Routine wird im X und Y-Register die Endadresse des Programms übergeben.

Die SAVE-Routine hat eine etwas kompliziertere Parameterübergabe. FILPAR braucht nur mit der Gerätenummer im X-Register aufgerufen zu werden, da weder Sekundäradresse noch Filenummer benötigt werden. Das Setzen des Filnamens erfolgt normal über FILNAM.

Übergeben werden müssen nun noch Anfangsadresse und Endadresse+1 des zu speichernden Bereichs. Die Anfangsadressen müssen Sie irgendwo in der Zero-Page in der Reihenfolge LO/HI ablegen. Empfehlenswert wären die Adressen $FB/FC, da diese nicht vom Betriebssystem oder Basic benutzt werden. Im Akku muß dann die Adresse des LO-Byte übergeben werden; wenn Sie die Adresse also unter $FB/FC speichern, muß im Akku $FB stehen.

Die Endadresse übergeben Sie wie folgt:
LO-Byte im X- und HI-Byte im Y-Register. Es muß immer 1 zur Engadresse addiert werden, da sonst das letzte Byte des Programms nicht abgespeichert

```
Prinzip des Speicherns von
Bereichen.

LDX #$08    ;  Geräteadresse
JSR FILPAR ;  setzen
LDX # (Filename LO-Byte)
LDY # (Filename HI-Byte)
LDA # (Filename Länge)
JSR FILNAM ;  setzen
LDX # (Startadresse LO-Byte)
LDY # (Startadresse HI-Byte)
STX  $FB    ;  zwischenspeichern
STY  $FC
LDA #$FB    ;  Pointer auf Startadr.
LDX # (Endadresse +1 LO-Byte)
LDY # (Endadresse +1 HI-Byte)
JSR SAVE
RTS
```

Listing 4. Und so funktioniert das Abspeichern

wird. Danach kann die Routine SAVE aufgerufen werden. Wieder haben wir für Sie zur Verdeutlichung zwei Listings: Listing 3 zeigt, wie man ein Programm an eine beliebige Adresse lädt; Listing 4 wie man einen beliebigen Bereich auf Diskette speichert. Erwähnenswert ist noch die Routine CLALL, die **alle** Files im Computer schließt; die Kanäle in der Floppy bleiben davon jedoch unberührt. Hier müssen Sie also sorgfältig mit CLOSE arbeiten, da Sie sonst Daten verlieren können.

Nachdem wie Sie nun mit Theorie überschwemmt haben, sollen Sie sogleich in den Genuß Ihrer neuen Kenntnisse kommen. Haben Sie schon einmal etwas

```
.
., 033C 20 79 00 JSR $0079
., 033F F0 43    BEQ $0384
., 0341 20 E7 FF JSR $FFE7
., 0344 A0 00    LDY #$00
., 0346 B9 A5 03 LDA $03A5,Y
., 0349 F0 06    BEQ $0351
., 034B 20 D2 FF JSR $FFD2
., 034E C8       INY
., 034F D0 F5    BNE $0346
., 0351 20 54 E2 JSR $E254
., 0354 20 C1 F5 JSR $F5C1
., 0357 A6 B7    LDX $B7
., 0359 F0 66    BEQ $03C1
., 035B A9 01    LDA #$01
., 035D A2 08    LDX #$08
., 035F A0 02    LDY #$02
., 0361 20 BA FF JSR $FFBA
., 0364 20 C0 FF JSR $FFC0
., 0367 A9 04    LDA #$04
., 0369 20 B1 FF JSR $FFB1
., 036C 20 BE ED JSR $EDBE
., 036F A2 01    LDX #$01
., 0371 20 C6 FF JSR $FFC6
., 0374 20 BE ED JSR $EDBE
., 0377 20 85 EE JSR $EE85
., 037A 20 97 EE JSR $EE97
., 037D A9 00    LDA #$00
., 037F 85 99    STA $99
., 0381 85 98    STA $98
., 0383 60       RTS
., 0384 A9 01    LDA #$01
., 0386 85 98    STA $98
., 0388 20 AE FF JSR $FFAE
., 038B 20 AB FF JSR $FFAB
., 038E A9 01    LDA #$01
., 0390 20 C3 FF JSR $FFC3
., 0393 A0 00    LDY #$00
., 0395 B9 AF 03 LDA $03AF,Y
., 0398 F0 06    BEQ $03A0
., 039A 20 D2 FF JSR $FFD2
., 039D C8       INY
., 039E D0 F5    BNE $0395
., 03A0 A9 00    LDA #$00
., 03A2 4C 74 A4 JMP $A474
.D 03A5
.?
.
.'03A5 53 50 4F 4F 4C 49 4E 47
.'03AD 20 00 45 4E 44 20 4F 46
.'03B5 20 53 50 4F 4F 4C 49 4E
.'03BD 47 8D 00 00 4C 08 AF 00
.I 03C5
.?
.
., 03C1 4C 08 AF JMP $AF08
.D 03C4
.?
.
READY.
```

Listing 5. Mit diesem Programm können Sie ein Floppy-Drucker-Spooling durchführen. Näheres im Text.

von Spooling gehört? Nein? Macht nichts, wir werden uns mit dieser Technik nämlich jetzt auseinandersetzen, und Sie werden dabei die Vorzüge dieser Möglichkeit genießen lernen.

Spooling? Was ist das?

Unter dem Begriff Spooling verbirgt sich eigentlich eine ganz einfache Technik, die jedoch enorme Vorteile besitzt: Es handelt sich um das Drucken direkt von Diskette. Haben Sie nicht auch schon öfters versucht, ein meterlanges Listing auf Papier zu bringen und den Drucker dabei mit wütenden Blicken zu größerer Eile aufgefordert, weil Sie nämlich unter Zeitdruck standen und sich bei der Arbeit keine Verzögerung erlauben konnten? Dann ist Spooling genau das Richtige für Sie. Bei dieser Methode wird ein Listing, das ausgedruckt werden soll, auf Diskette gebracht. Danach starten Sie ein Spooling-Programm und siehe da; der Drucker beginnt Ihr Listing auf Papier zu bringen, und der Computer meldet sich betriebsbereit mit READY.

Dies ist kein Wunder, sondern die Eigenschaft des seriellen Bus Ihres Computers. Sie haben vorhin gelernt, wie man den Bus des Computers in Maschinensprache bedient. Dabei fielen auch Worte wie TALK, LISTEN, SENDEN und EMPFANGEN. Der Trick des Spooling ist nun der: Mit Hilfe des CMD-Befehls in Basic können Sie ein Listing auf Diskette »umleiten«, und zwar geschieht dies ähnlich wie beim Drucker: Sie eröffnen ein File und schicken mit dem CMD-Kommando alle weiteren Bildschirmausgaben auf den Bus. Nur ist jetzt nicht der Drucker der Adressat sondern die Floppy. Hier ein Beispiel:
Sie haben ein Listing im Speicher und wollen dieses auf Diskette ablegen, sein Name soll »TEST« sein:

```
OPEN 1,8,2,"TEST,U,W"
CMD1
LIST
```

Drucken ohne Umwege

Nach dieser Befehlsfolge wird Ihr Listing als USR-File auf Diskette geschrieben. Wie wäre es nun, wenn die Floppy ein TALK-Kommando erhalten würde, das sie veranlaßt, das eben geschriebene File auf den Bus zu bringen? Der »Hörer« ist aber jetzt nicht, wie üblich, der Computer sondern der Drucker, den wir zuvor mit einem LISTEN dazu aufgefordert haben. Die Folge wäre das, was Sie sich jetzt schon denken können:

Die Floppy schickt das gesamte Listing über den Bus, und der Drucker, der ja auf Empfang programmiert ist, bekommt dieses Listing und druckt es aus. Der Computer hat mit der ganzen Sache nichts zu tun, da er sich nach Senden der Kommandos »zurückgezogen« hat und bleibt demzufolge frei für weitere Arbeit.

Der Zugriff auf den Bus ist dem Computer natürlich für die Zeit der Übertragung verwehrt, aber Sie können währenddessen intern weiterarbeiten. Ist die Übertragung beendet, so sind beide Peripheriegeräte noch auf Sendung und müssen erst »zur Ruhe gebracht« werden, bevor sie wieder ansprechbar sind. Aber auch das erledigt ein kleines Programm für uns. Sehen Sie sich jetzt einmal Listing 5 an. Es enthält ein Spooling-Programm, das mit

```
SYS828,"filename"
```

aufgerufen wird. Danach meldet sich der Computer mit

```
SPOOLING filename
READY
```

und der Drucker beginnt zu arbeiten. Ist der Druckvorgang beendet, so tippen Sie noch einmal

```
SYS828
```

ohne Filenamen, und die Leuchtdiode an der Floppy erlischt. Es erscheint die Meldung

```
END OF SPOOLING
READY
```

Dieses Programm ist, im Gegensatz zu unseren anderen Listings, zum sofortigen Eintippen gedacht.

Wie Sie aus diesem Beispiel sehen, kann es von großem Nutzen sein, wenn Sie das Prinzip des seriellen Bus verstehen und dessen »Verkehrsregeln« kennen, da viele Programme nur deshalb mit geringem Aufwand große Effekte und Nutzen erzielen. Ein weiteres Beispiel in dieser Reihe dürfte wohl HYPRA-LOAD sein, das Sie in Ausgabe 10 des 64'er-Magazins fanden. Dieses Programm nutzt aber noch einige weitere Tricks der Maschinenspracheprogrammierung, die wir in den nächsten Ausgaben besprechen wollen.

Was kommt demnächst

In Teil 4 unseres Kurses wollen wir nämlich in die direkte Programmierung der Floppy einsteigen, das heißt, das Abspeichern von Maschinenprogrammen in ihren Pufferspeicher und das Ausführen derselben. Als Beispiel werden wir unser HYPRA-LOAD ein wenig »zerlegen«, um Ihnen die Möglichkeiten dieser Programmiertechnik nahezubringen.

Bis zum nächsten Mal also noch viel Spaß in der Busprogrammierung und in der Anwendung des Druckerspooling.

(B. Schneider/K. Schramm/gk)