## W E N D

**WEND** beendet die zum letzten **WHILE** gehörige **WHILE-WEND**-Schleife (s. **WHILE**). Wenn BASIC ein **WHILE**-Kommando findet, prüft es, ob ein zugehöriges **WEND** vorhanden ist. Ist dies nicht der Fall, wird das Programm abgebrochen. Wenn BASIC fündig wird, nimmt es grundsächlich das nächstbeste passende **WEND**. Es nimmt also keine Rücksicht darauf, ob sich das vom Programmierer gewünschte **WEND** vielleicht in einer Unterroutine befindet.

Die Schachtelung einer **WHILE-WEND**-Schleife kann nur folgendermaßen geschehen:

```
WHILE bedingung 1
   WHILE bedingung 2
      WHILE bedingung 3
            |
            |
      WEND
   WEND
WEND
```

**Beispiel:**

```
10 WHILE a$<>"Ende"     | wiederhole die Schleife solange, bis a$="Ende"
20   INPUT "Bitte Wort eingeben ( Ende beendet ): ",a$
30 WEND
40 PRINT "Programm beendet."
50 END
```

**RUN:**

```
Bitte Wort eingeben ( Ende beendet ): Test
Bitte Wort eingeben ( Ende beendet ): Ende
Programm beendet.
Ok
■
```

## W H I L E

**WHILE bedingung** markiert den Anfang einer **WHILE-WEND**-Schleife. Die **WHILE-WEND**-Schleife ist im Prinzip nichts anderes als eine Endlosschleife, aus der mit einem **IF**-Kommando herausgesprungen werden kann:

```
10 IF bedingung THEN 40        10 WHILE bedingung
20 'Kommandoteil               20 'Kommandoteil
30 GOTO 10                     30 WEND
40 'weiter                     40 'weiter
```



Die Schleife wird solange ausgeführt, bis **bedingung** nicht mehr erfüllt ist. **bedingung** ist in den meisten Fällen eine Vergleichsoperation, z.B. **WHILE a>b**. Die Schleife wird also solange wiederholt, bis **a** nicht mehr größer als **b** ist, denn das Kommando sagt: wiederhole, solange **a größer als b** ist.

Beendet wird die Schleife durch das **WEND**-Kommando (s. dort).

**Beispiel:**

```
10 WHILE a$<>"Ende"
20   INPUT "Bitte Wort eingeben ( Ende beendet ): ",a$
30 WEND
40 PRINT "Programm beendet."
50 END
```

**RUN:**

```
Bitte Wort eingeben ( Ende beendet ): Test
Bitte Wort eingeben ( Ende beendet ): Ende
Programm beendet.
Ok
■
```

## W I D T H

Mit **WIDTH zeilenbreite** kann man die Zeilenbreite der Konsole (Bildschirm) verändern.

**zeilenbreite** gibt an, nach wieviel Zeichen BASIC eine neue Zeile beginnen soll. **zeilenbreite** muß zwischen 1 und 255 liegen.

**Beispiel:**

```
WIDTH 40            | Bildschirm 40 Zeichen breit

WIDTH 80            | Bildschirm 80 Zeichen breit

WIDTH 255           | Bildschirm 255 Zeichen breit (nach 90 Zeich. neue Zeile)

WIDTH 255,255       | Bei Bildschirmausgaben durch CHR$(27)+"Y" wird nicht unkontrol-
                    | liert ein Carriage Return durchgeführt
```

## W I D T H  L P R I N T

Mit **WIDTH LPRINT zeilenbreite** kann man die Zeilenbreite des Druckers verändern.

**zeilenbreite** gibt an, nach wieviel Zeichen der Drucker eine neue Zeile beginnen soll. **zeilenbreite** muß zwischen 1 und 255 liegen.

Für den Ausdruck von BASIC-Programmen (Listings) hat dieses Kommando eine große Bedeutung. Der voreingestellte Wert für die Zeilenbreite (132) ist größer, als der Drucker Zeichen in einer Zeile unterbringen kann. Daher wird beim Ausdruck des Listings teilweise an einer ungewünschten Stelle ein Zeilenvorschub eingefügt, was die Struktur des Listings erheblich durcheinander bringt. Da der Drucker ein sog. 80-Zeichen Drucker ist (also 80 Zeichen pro Zeile drucken kann), muß man, um einen problemlosen Ausdruck zu gewährleisten, mit **WIDTH LPRINT** die **zeilenbreite** auf 80 Zeichen einstellen (**WIDTH LPRINT 80**).

**Beispiel:**

```
WIDTH LPRINT 40        | Druckerzeilenbreite 40 Zeichen

WIDTH LPRINT 80        | Druckerzeilenbreite 80 Zeichen

WIDHTH LPRINT 255      | wichtig für Ausdruck von Grafiken
```

## W R I T E

Der **WRITE**-Befehl arbeitet ähnlich wie der **PRINT**-Befehl. Die Gemeinsamkeit liegt darin, daß man mit beiden Befehlen irgendetwas (Variablen, Texte) auf die Konsole (Bildschirm) bringen kann. Der Unterschied liegt jedoch in der Form. Während bei **PRINT** alles ohne irgendwelchen Zusatz auf die Konsole gebracht wird, trennt der **WRITE**-Befehl bei mehreren Variablen grundsätzlich die einzelnen durch ein Komma. Ist die Variable eine Stringvariable, kleidet er sie auf der Konsole in Anführungsstriche ein. Mehrere Variablen können durch ein Komma oder ein Semikolen im **WRITE**-Befehl getrennt werden. Sie haben nur trennende Bedeutung (der einzelnen Elemente) und haben keinen Einfluß (wie bei **PRINT**) auf den Ausdruck der Variablen.

**Beispiel:**

```
10 a=10:b=20:c=30
20 text$="Dies ist ein Test."
30 WRITE a,b,c,text$
40 WRITE a,"abcdefg",b
50 END
```



**RUN:**

```
10,20,30,"Dies ist ein Test."
10,"abcdefg",20
OK
■
```

## W R I T E  #

Der **WRITE #**-Befehl arbeitet ähnlich wie der **PRINT #**-Befehl. Die Gemeinsamkeit liegt darin, daß man mit beiden Befehlen irgendetwas (Variablen, Texte) in eine Datei schreiben kann. Der Unterschied liegt jedoch in der Form. Während bei **PRINT #** alles ohne irgendwelchen Zusatz in die Datei geschrieben wird, trennt der **WRITE #**-Befehl bei mehreren Variablen grundsätzlich die einzelnen durch ein Komma. Ist die Variable eine Stringvariable, kleidet er sie in der Datei in Anführungsstriche ein. Mehrere Variablen können durch ein Komma oder ein Semikolen im **WRITE #**-Befehl getrennt werden. Sie haben nur trennende Bedeutung (der einzelnen Elemente) und keinen Einfluß (wie bei **PRINT #**) auf den Ausdruck der Variablen.

**Beispiel:**

```
10 OPEN "O",#1,"M:TEST"          | Datei #1 eröffnen
20 a=10:b=20:c=30                | Variablen definieren
30 text$="Dies ist ein Test."    |
40 WRITE #1,a,b,c,text$          | und in Datei schreiben
50 WRITE #1,a,"abcdefg",b        | auch "gemischtes" in Datei schreiben
60 CLOSE                         | Datei schließen
70 DISPLAY "M:TEST"              | und Inhalt kontrollieren
80 END                           |
```

**RUN:**

```
10,20,30,"Dies ist ein Test."
10,"abcdefg",20
OK
■
```

# Dateiverarbeitung mit Jetsam

Die Verarbeitung von Jetsam-Dateien unterscheidet sich grundlegend von der Verarbeitung von ASCII-Dateien, die nur sequentiell, d.h. Satz für Satz verarbeitet werden können, und auch von der Verarbeitung von Dateien mit wahlfreiem Zugriff, bei denen über die Satznummer die Datensätze der Datei in beliebiger Reihenfolge verarbeitet werden können. Auf die Daten in einer Jetsam-Datei kann nicht direkt, sondern nur indirekt über Schlüssel zugegriffen werden, die in einer Schlüsseldatei abgelegt sind. Dies hat den Vorteil, daß man auf die Daten aufgrund von eingegebenen Suchmerkmalen, den Schlüsseln, zugreifen kann, was besonders bei größeren Datenbeständen eine schnellere Verarbeitung ermöglicht, da nur auf die Daten zugegriffen wird, die tatsächlich verarbeitet werden sollen.

Eine Jetsamdatei besteht aus 1.: einer Datendatei, in der die eigentlichen Datensätze mit den Daten abgelegt sind, und 2: aus einer Schlüsseldatei, in der die Schlüssel sind, über die auf die Datensätze in der Datendatei zugegriffen werden kann. In letzterer werden auch die Satznummern der Datensätze in Schlüsselreihen abgelegt. Zur Verarbeitung unter Jetsam müssen beide Teildateien, die zusammen als eine einheitliche Jetsam-Datei zu sehen sind, geöffnet sein, damit sie dem Programm zur Verfügung stehen. Mit den Jetsam-Funktionen zum Suchen nach Daten wird intern die Satznummer bereitgehalten, über die durch ein nachfolgendes **GET** dann die Daten aus der Datendatei eingelesen und dem Programm zur Verfügung gestellt werden.

Die Datendatei entspricht ihrem Aufbau nach einer Datei für wahlfreien Zugriff, nur daß die ersten 128 Bytes für die spezielle Jetsamverarbeitung reserviert sind.

Ein direkter Zugriff auf die Datendatei (ohne über die Schlüsseldatei zu gehen) empfiehlt sich nicht, da durch eine



solche Arbeitsweise nicht sichergestellt ist, daß nur gültige Datensätze verarbeitet werden. Sind durch Löschen von Datensätzen Daten in der Datendatei ungültig geworden, sind die als ungültig gekennzeichneten Sätze nur über die Schlüsseldatei erkennbar. Wird nämlich z.B. ein Datensatz gelöscht, so wird die Datendatei nicht verändert! Es werden lediglich sämtliche Schlüssel zu dem Datensatz in der Schlüsseldatei gelöscht, so daß auf den Satz nicht mehr über einen Schlüssel zugegriffen werden kann!

Daten- und Schlüsseldatei müssen stets stimmig (konsistent) zueinander sein. Dies wird durch den Befehl **CONSOLIDATE** und bei Beendigung der Bearbeitung durch **CLOSE Liste von: Dateisymbol** sichergestellt. Sollten durch einen Fehler die beiden Teildateien unstimmig zueinander werden, sind die Daten verloren und können, falls überhaupt, nur sehr mühsam gerettet werden. Es empfiehlt sich daher, diese Dateien möglichst vor jeder Verarbeitung zu sichern.

Um Datensätze verarbeiten zu können, muß sich das Programm, das mit einer Jetsam-Datei arbeitet, auf den gewünschten Datensatz positionieren. Eine solche Positionierung geschieht z.B. durch einen Suchvorgang (**SEEK**-Funktionen). Die Position innerhalb der Jetsam-Datei, die das Programm eingenommen hat, wird auch aktuelle Position genannt. Über diese aktuelle Position weiß Jetsam, mit welchem Datensatz gearbeitet werden soll, so daß der zum Schlüssel gehörende Datensatz über ein **GET** eingelesen werden kann.

Fast alle Jetsam-Funktionen und -Befehle (Unterschied zwischen Funktion und Befehl vgl.: Ende des Kapitels) arbeiten über die aktuelle Position in der Jetsam-Datei, die durch die Schlüsselreihe, den Schlüsselwert und die Satznummer des Datensatzes in der Datendatei bestimmt wird.

Bei der Verarbeitung von Jetsam-Dateien sind die Schlüssel, die zum Wiederauffinden der Daten dienen, sehr wichtig. Es ist möglich, in Jetsam bis zu acht Schlüsselreihen zu

benutzen (Reihe 0 bis 7). Eine Schlüsselreihe kann man sich als Nachschlagelexikon vorstellen. Jeder Eintrag in diesem Lexikon, ein sogenannter Schlüsseleintrag, der den Schlüsselwert und, für den Benutzer verborgen, die Satznummer des zugehörigen Datensatzes enthält, wird von Jetsam selbständig in der richtigen Sortierfolge (aufsteigende Sortierung) in der Schlüsseldatei vorgehalten. Der Benutzer oder das Programm braucht sich um die Sortierung der Schlüssel nicht zu kümmern. Durch die Satznummer wird vom Schlüsseleintrag auf den Beginn des Datensatzes in der Datendatei verwiesen. Jede Schlüsselreihe ist von den anderen Schlüsselreihen unabhängig.

Vom BASIC-Programm aus muß dann umsortiert werden, wenn eine andere Sortierfolge gewünscht wird, z.B. eine absteigende Sortierung der Daten.

Ist es zulässig, daß in einer Schlüsselreihe mehrere Schlüsseleinträge mit identischen Schlüsselwerten stehen und wird über diese Schlüsselreihe auf die Daten sequentiell zugegriffen, so werden die Sätze innerhalb des Schlüsselsatzes (siehe unten) in der Reihenfolge der Satznummern bereitgestellt. Möchte man diese Datensätze nach einem anderen Sortierbegriff als dem Jetsam-Schlüssel sortiert haben, muß diese Sortierung im Programm selbst vorgenommen werden.

**Beispiel:**

In einer Adreßdatei gibt es als Schlüssel den Namen und die Postleitzahl. Jeder Name darf in der Datei nur einmal vorkommen (durch **RANKSPEC** für die Schlüsselreihe angegeben). Die Postleitzahl darf als Schlüssel mehrfach vorkommen.
Liest man sämtliche Adressen aus der Datei über den Namen, bekommt man die Namen alphabetisch aufsteigend sortiert - Hubert, Maier, Meier, Krause, Schulze.
Liest man einige Adressen gezielt nach einer bestimmten Postleitzahl aus der Datei, erhält man die Daten in der Reihenfolge ihrer Satznummern (in Klammern angegeben): Krause (1), Hubert (2), Schulze (3), Meier (4), Maier (5).
Möchte man alle Adressen mit einer bestimmten Postleitzahl nach Namen alphabetisch sortiert bekommen, müssen die Adressen durch eine Sortierroutine entsprechend nach den Namen sortiert werden, oder man liest sämtliche Datensätze sequentiell nach dem Namen und gibt nur die Adressen aus, die die gewünschte Postleitzahl enthalten.
Welchen Weg man wählt, hängt im wesentlichen von der Größe der Datei ab - bei einer großen Datei ist es besser, gezielt über die Postleitzahl zu lesen und die Adressen umzusortieren; bei einer kleinen Datei kann es vertretbar sein, alle Namen zu lesen, die Datensätze über ein **GET** zu lesen und die Postleitzahl abzufragen.



Ein Schlüsselsatz wird dann in einer Schlüsselreihe gebildet, wenn mehrere Schlüsseleinträge denselben Schlüsselwert haben.

Je Schlüsselreihe sollte nur eine Art von Schlüssel vorgesehen werden. Z.B. Reihe 0: Name und Vorname als kombinierter Schlüssel, Reihe 1: Postleitzahl, Reihe 2: Ort, Reihe 3: Telefonvorwahl. Man könnte sämtliche Schlüssel auch in Reihe 0 einstellen, was jedoch die Handhabung der Datei sehr erschweren würde, denn es müssen nicht zu jedem Datensatz sämtliche Schlüssel ausgefüllt sein (nicht alle Adressaten haben Telefon) und beim Löschen eines Satzes müßte sehr sorgfältig geprüft werden, ob man auch alle Schlüssel gelöscht hat, die in einer Schlüsselreihe eingetragen sind.- Siehe auch die Ausführungen zu **ADDKEY** und **DELKEY** -.

Enthalten die Datensätze sehr viele verschiedene Datenelemente, so sind die Datenelemente, über die als Schlüssel nach den Datensätzen zugegriffen werden soll, sorgfältig auszuwählen. Kann man nicht alle Schlüssel in den acht Schlüsselreihen unterbringen, so sollte man prüfen, ob man aus der einen Jetsam-Datei nicht zwei Dateien machen könnte.

Ggf. wäre zu überlegen, ob die Daten nicht anders gegliedert werden können.

Die Schlüssel sollten auf jeden Fall so gewählt werden, daß die meisten Zugriffe auf die Daten abgedeckt werden. Es gilt, die Frage zu beantworten: "Wer braucht welche Daten mit welchen Suchmerkmalen mit welcher Häufigkeit (laufend, täglich, wöchentlich, monatlich etc.)?" In einem Lager eines Fertigungsbetriebes z.B. werden zu jedem Artikel die Artikelnummer, Bezeichnung des Artikels, Kurzbezeichnung des Artikels, Lieferant, Bestand, Mindestbestellmenge, Preis, MWSt.-Satz, Einkaufs- oder Verkaufsartikel, Einkaufsdatum und Verkaufsdatum festgehalten. Je nach Vorgehensweise des Betriebes wird sehr viel über die Artikelnummer oder auch über die Artikelkurzbezeichnung auf die Artikeldaten zugegriffen. Für die Rechnungsabteilung wird es interessant sein, über den Namen des Lieferanten Artikeldaten auszuwählen, um z.B. Rechnungen zu prüfen.
Um möglichst allen Erfordernissen des Betriebes gerecht zu werden, muß sorgfältig darüber nachgedacht werden, welche Datenelemente als Schlüssel dienen sollen, damit die Artikeldaten in den verschiedenen Bereichen des Betriebes schnell und gezielt zur Verfügung stehen; hier z.B. können die Artikelnummer, die Artikelkurzbezeichnung und der Name des Lieferanten als Schlüssel auf die Artikeldaten dienen.

In den Jetsam-Befehlen und -Funktionen sind Sperren anzugeben. Es gibt folgende Sperren, die über die entsprechende Kennziffer anzugeben sind:

0 **keine Sperre** einrichten
1 **Lesesperre** einrichten, d.h. ein anderer Benutzer darf lesen, aber nicht schreiben
2 **Schreibsperre** einrichten, d.h. ein anderer Benutzer darf weder lesen noch schreiben.

Die Sperren können für die gesamte Datei (bei **CREATE** und **OPEN**) oder für einzelne Sätze (bei den übrigen Jetsam-Befehlen und -Funktionen) gesetzt werden.



Die Sperren haben eigentlich nur bei Mehrbenutzer-Systemen Bedeutung, wenn also mehrere Benutzer gleichzeitig auf eine Datei zugreifen können. Der JOYCE ist in der Regel jedoch ein Einbenutzer-System, so daß die Sperren eigentlich ohne Belang sind. Für eine ordnungsgemäße Verarbeitung sollten die Sperren aber doch sinnvoll angegeben werden.

Bei den **SEEK**-Funktionen genügen Lesesperren. Bei Funktionen, die die Datei verändern, sollten Schreibsperren auf den Datensatz gesetzt werden. Sollte Ihr Programm dann einmal auf einem Mehrbenutzer-System eingesetzt werden, ist dadurch sofort eine fehlerfreie Verarbeitung sichergestellt.

Wie bei jeder Verarbeitung von Dateien, muß auch für Jetsam die Jetsam-Datei angelegt und eröffnet, d.h. für das Programm zugänglich gemacht werden. Es müssen Datensätze geschrieben, verändert, gelesen und gelöscht werden können. Für all diese Aufgaben gibt es auf die Belange der Jetsam-Dateien zugeschnittene BASIC-Befehle, die im folgenden ausführlich erläutert werden.
Hier die BASIC-Befehle, die nur für die Verarbeitung von Jetsam-Dateien vorgesehen sind, in alphabetischer Reihenfolge und gegliedert nach Jetsam-Funktion und Jetsam-Kommando:

Jetsam-Funktion
- ADDKEY
- ADDREC
- CONSOLIDATE
- DELKEY
- FETCHKEY$
- FETCHRANK
- FETCHREC
- LOCK
- RANKSPEC
- SEEKKEY
- SEEKNEXT
- SEEKPREV

SEEKRANK
SEEKREC
SEEKSET

Jetsam-Kommando
BUFFERS
CLOSE
CREATE
FIELD
GET
OPEN
OPTION FIELD
PUT

Außerdem sind noch diese Befehle von Bedeutung:
CLEAR CVD CVI CVIK CVS CVUK FIELD LSET MEMORY MKD$ MKI$ MKIK$ MKS$ MKUK$

Die Befehle sollen an einer Beispieldatei verständlich gemacht werden;
Es ist eine kleine Adreßdatei mit Name, Vorname, Postleitzahl, Ort, und einer Kennung, ob es sich um einen Lieferanten, Kunden oder Geschäftsfreund handelt.
Als Schlüssel wird benutzt:
In Reihe 0 jeweils die ersten 5 Buchstaben des Names und Vornamens als kombinierter Schlüssel; es dürfen in dieser Schlüsselreihe keine Schlüsseleinträge mit identischem Schlüsselwert vorkommen.
In Reihe 1 steht die Postleitzahl.
In Reihe 2 steht die Kennung, ob "Lieferant" (L), "Kunde" (K) oder "Geschäftsfreund" (G) zutrifft.

Sämtliche Schlüsselreihen enthalten 5 Schlüsseleinträge.



| Reihe 0 | ElektMaier | InstaKraus | LudwiElekt | Mayr Großh | XaverUnsin |
|---|---|---|---|---|---|
| Satznr. | 4 | 2 | 6 | 3 | 5 |
| Reihe 1 | 2390 | 2390 | 8000 | 8000 | 8000 |
| Satznr. | 2 | 4 | 3 | 5 | 6 |
| Reihe 2 | G | K | K | L | L |
| Satznr. | 5 | 2 | 4 | 3 | 6 |

Die aktuelle Position in der Datei wird durch ↓ angezeigt.

Auf die Darstellung der Datendatei wird der Einfachheit halber verzichtet.

In Reihe 1 haben wir zwei Schlüsselsätze, einmal mit "2390" und "8000", in Reihe 2 ebenfalls 2 Schlüsselsätze mit "K" und "L". Die nur einmal vorkommenden Schlüsselwerte könnte man ebenfalls als Schlüsselsätze betrachten; diese Schlüsselsätze umfassen statt mehrerer nur einen Schlüsseleintrag.

Die Befehle werden an Hand eines kleinen Beispielprogramms erläutert, das Sie auf der zum Buch erhältlichen Diskette unter dem Namen JETSAM.BAS finden. Die BASIC-Zeilen sind diesem Programm entnommen. Das vollständige Listing ist am Ende dieses Kapitels abgedruckt. Wenn Sie das Programm unter BASIC ablaufen lassen, werden alle aufgeführten Beispiele zu den Funktionen durchgeführt und jeweils die aktuelle Position und der Antwortcode angezeigt.

Um das Programm ablaufen zu lassen, starten Sie CP/M, laden BASIC,

```
A>BASIC
```

legen die zum Buch zu beziehende Diskette mit Seite B in Laufwerk A: - oder Sie geben das Listing ein, das Sie am Ende des Kapitels finden, speichern es mit **SAVE "jetsam** - und starten es mit

```
OK
. run "jetsam".
```

Hier nun die Befehle im einzelnen:

Zu Beginn der Jetsam-Verarbeitung sind die Systemvariablen einzustellen.

```
BUFFERS Zahl der Puffer für die Schlüsseldatei[,Zahl der Satzsperren]
```

oder

```
BUFFERS ,Zahl der Satzsperren
```

**Beispiel:**

```
30 BUFFERS 20
```

Durch diesen Befehl wird die Größe des Dateipuffers für die Schlüsseldatei festgelegt und ein entsprechend großer Bereich im Speicher reserviert. Bei der Verarbeitung von Jetsam-Dateien geschieht der Zugriff auf die Daten über die Schlüsseldatei. Für eine zügige Verarbeitung der Schlüsseldatei ist es günstig, wenn ein möglichst großer Pufferbereich zur Verfügung steht, um möglichst wenig zeitraubende Zugriffe auf Diskette und möglichst viele schnelle Zugriffe auf den Puffer zu haben.
Dieser Befehl ist Voraussetzung, um mit der Schlüsseldatei arbeiten zu können.

Beim **OPEN**-Befehl der Jetsam-Datei versucht Jetsam möglichst viele Einträge der Schlüsseldatei in den durch **BUFFERS** reservierten Speicherbereich einzustellen, um möglichst viele Schüsseleinträge sofort bereit zu halten, damit so wenig wie möglich aus der Schlüsseldatei nachgelesen werden muß, was die Programmlaufzeit verlängern würde.

Da ein Puffer 128 Bytes umfaßt, wird durch das obige Beispiel ein Speicherbereich von 20 * 128 = 2560 Bytes als Puffer für die Schlüsseldatei reserviert. Dieser Speicherbereich steht damit dem BASIC-Programm nicht zur Verfügung.

Im **BUFFERS**-Befehl können auch Pufferbereiche für die Speicherung von Satzsperren reserviert werden. Dies ist jedoch nur für Mehrbenutzer-Systeme bedeutsam. Ein solcher Speicherbereich umfaßt jeweils 7 Bytes.



**BUFFERS** muß vor einem **OPEN** oder **CREATE** der Jetsam-Datei stehen.

**Einige allgemeine Hinweise zum Dateipuffer:**
Jeglicher Zugriff zum Lesen oder Schreiben einer Datei, auch bei sequentiellen Dateien oder Dateien mit wahlfreiem Zugriff, wird nie direkt auf die Datei, die auf der Diskette gespeichert ist, durchgeführt, sondern immer über einen vom Betriebssystem oder auch vom BASIC eingerichteten Dateipuffer. Dieser Puffer steht im Arbeitsspeicher an einer bestimmten Stelle, in die vom Betriebssystem die einzulesenden Daten aus der Datei hineingestellt werden. Dann erst werden die Daten dem Programm zur Verfügung gestellt.
Sollen Informationen auf Diskette gespeichert werden, schreibt das Programm die Daten in den Dateipuffer. Ist der Puffer vollständig gefüllt bzw. ist man am Ende des Progamms angekommen, liest das Betriebssystem die Daten aus dem Pufferbereich und schreibt sie auf Diskette.
Sinn dieser Vorgehensweise ist, daß der tatsächliche physikalische, mechanische Zugriff stets nur von den Routinen des Betriebssystems durchgeführt wird, und daß sich nicht jedes Programm um die Handhabung der Datei und des Diskettenlaufwerks selbst kümmern muß.

Soll ein Datensatz aus einer Datei gelesen werden, prüft das Betriebssystem zunächst, ob der Datensatz schon im Pufferbereich der Datei zu finden ist. Ggf. wird nur ein Zeiger, der auf den Beginn des angeforderten Datensatzes zeigt, umgesetzt und die Daten dadurch für das Anwendungsprogramm zugänglich.
Erst wenn festgestellt wird, daß die Daten nicht im Pufferbereich zu finden sind, werden ein Zugriff auf die Datei auf Diskette durchgeführt und so viele Daten eingelesen, bis dieser Bereich neu gefüllt ist. Der Zeiger wird auf den Beginn des angeforderten Satzes gesetzt, in diesem Fall meist der Beginn des Pufferbereichs.

Soll zum ersten Mal mit einer Jetsam-Datei gearbeitet werden, müssen die Daten- und die Schlüsseldatei durch den Befehl **CREATE** angelegt werden. Bei ASCII-Dateien und Dateien mit wahlfreiem Zugriff genügt der **OPEN**-Befehl zum Anlegen der Datei. Aufgrund der besonderen Erfordernisse bei der Verarbeitung mit Jetsam ist jedoch eine besondere Vorbereitung der Dateien zur Verarbeitung erforderlich.

In der Datendatei sind die ersten 128 Bytes der Datei, bzw. so viele Bytes wie in der Satzlänge beim **CREATE** angegeben wurde, für die Jetsam-Routinen reserviert, d.h., für den Anwender nicht zugänglich. Sollen Datensätze verarbeitet werden, die länger als 128 Bytes sind, ist der Maximalwert von 128 durch einen **MEMORY**- (**MEMORY ,,,maximale Satzlänge**) oder **CLEAR**-Befehl (**CLEAR,,,,maximale Satzlänge**) entsprechend heraufzusetzen. Außerdem sind in jedem Datensatz die ersten zwei Bytes, die die Satznummer des Datensatzes enthalten, für Jetsam reserviert und damit ebenfalls nicht direkt dem Anwender zugänglich, da für ihn die Satznummer in der Regel nicht interessant ist. (Vgl. aber **FETCHREC**.)

Außerdem wird die Schlüsseldatei eingerichtet, die besonders auf die Verarbeitung der Jetsam-Befehle und -Funktionen eingerichtet ist. Die Datei wird zur Aufnahme der Schlüsselwerte und Satznummern in den Schlüsselreihen vorbereitet.

Nach Durchführung des Befehls stehen die Schlüssel- und die Datendatei zur Verarbeitung zur Verfügung, so daß der **OPEN**-Befehl nach dem **CREATE** nicht mehr erforderlich ist.

```
CREATE Datei-Symbol, Name der Datendatei, Name der Schlüsseldatei, Sperre der
Jetsam-Datei, [Länge der Datensätze in der Datendatei (einschließlich 2 Byte
für die Satznummer)], [Benutzer-String]
```

Wurde die Jetsam-Datei in einem ersten Programmlauf durch **CREATE** eingerichtet, müssen bei einem erneuten Programmlauf die Daten- und Schlüsseldatei für die Verarbeitung eröffnet werden.



```
OPEN "K",Datei-Symbol, Name der Datendatei, Name der Schlüsseldatei, Sperre
der Jetsam-Datei, [Länge der Datensätze in der Datendatei (einschließlich 2
Byte für die Satznummer)], [Benutzer-String]
```

Voraussetzung für den **OPEN**-Befehl ist, daß die Datendatei und die Schlüsseldatei ( = Indexdatei) in einem vorangegangenen Programmlauf durch ein **CREATE** angelegt und zur Jetsam-Verarbeitung vorbereitet wurden.

Durch das "K" in der **OPEN**-Anweisung wird kenntlich gemacht, daß eine Jetsam-Datei zu eröffnen ist (K für **K**ey - engl. Schlüssel)

Durch Angabe des Benutzer-Strings kann in die Schlüsseldatei z.B. eine Kurzbezeichnung der Jetsam-Datei geschrieben werden.

**Beispiel:**

```
50 IF FIND$("TEST.DAT")<>"" AND FIND$("TEST.IND")<>""
THEN OPEN "K",#1,"TEST.DAT","TEST.IND",2
ELSE CREATE #1,"TEST.DAT","TEST.IND",2
```

Hier wird zunächst über **FIND$** geprüft, ob die Datendatei **TEST.DAT** und die Schlüsseldatei **TEST.IND** vorhanden sind. Ist dies der Fall, werden die Dateien durch **OPEN** eröffnet, sonst durch **CREATE** eingerichtet. Es wird jeweils eine Schreibsperre auf die Jetsam-Datei gelegt. Eine Satzlänge und ein Benutzer-String wird nicht angegeben.

Bei Programmende ist die Jetsam-Datei zu schließen.

```
CLOSE Liste von: Dateisymbol
```

Um die Verarbeitung der Jetsam-Datei zu beenden, ist es wichtig, daß das Datei-Symbol mit angegeben wird; für Dateien mit wahlfreiem Zugriff und einfach sequentielle Dateien ist die Angabe des Datei-Symbols nicht erforderlich.

Nur so ist sichergestellt, daß die Schlüssel- und Datendatei stimmig (konsistent) bleiben und für eine Weiterverarbeitung in weiteren Programmläufen oder durch andere Programme zur Verfügung stehen.

**Beispiel:**

```
260 CLOSE #1
```

Die unter #1 eingerichtete bzw. eröffnete Jetsam-Datei wird geschlossen. Alle noch nicht auf Diskette geschriebenen Pufferbereiche werden jetzt auf die Diskette gebracht.

Auch während des Programmlaufs ist sicherzustellen, daß die Schlüssel- und Datendatei zueinander stimmig bleiben.

```
CONSOLIDATE (Datei-Symbol)
```

Jede Veränderung wie

Einfügen eines Satzes in die Datendatei
Einfügen von Schlüsseln in die Schlüsseldatei
Überschreiben von Daten in der Datendatei
Löschen von Schlüsseln in der Schlüsseldatei

führt dazu, daß die Daten- und Schlüsseldatei als nicht stimmig markiert werden. Dies bedeutet, daß aufgrund der Veränderung nicht gewährleistet ist, daß alle (Schlüssel-)Informationen, die in den durch den **BUFFERS**-Befehl eingerichteten Pufferbereich hineingestellt wurden, auch auf Diskette geschrieben und damit gesichert sind.

Um zu gewährleisten, daß Daten- und Schlüsseldatei stets zueinander stimmig sind, wird dringend empfohlen, nach jedem Befehl, der die Jetsam-Datei verändert (**ADDKEY, ADDREC, DELKEY, PUT, RANKSPEC**), ein **CONSOLIDATE** durchzuführen. Der Befehl bewirkt einen physikalischen Zugriff auf das Diskettenlaufwerk zum Wegschreiben der Pufferbereiche in die Dateien. Zur Beschleunigung des Programms sollten die Daten- und Schlüsseldatei zur Verarbeitung im Laufwerk M: zur



Verfügung stehen, so daß sich die **CONSOLIDATE**-Funktion kaum auf die Laufzeit des Programms auswirkt.

**Beispiel:**

```
280 rc=CONSOLIDATE(#1)
```

Auf die unter #1 eingerichtete bzw. eröffnete Datei wird ein **CONSOLIDATE** durchgeführt. Da dies eine Funktion ist, muß das Funktionsergebnis einer Variablen - hier **rc** -, zugewiesen werden. Bei Einzelbenutzer-Systemen, wie es der Benutzung des JOYCE (leider) meistens entspricht, ist das Funktionsergebnis stets 0. Bei Mehrbenutzer-Systemen wird angegeben, durch wieviele Benutzer die Datei als inkonsistent markiert wurde, d.h., wie viele Benutzer seit dem **OPEN/CREATE** bzw. seit dem letzten **CONSOLIDATE** die Datei verändert haben.
Um Daten in die Jetsam-Datei schreiben und auch wieder auslesen zu können, muß der Datensatz definiert sein. Dies geschieht durch das Kommando **FIELD**. Damit werden die Datenfelder und ihre jeweilige Länge in Bytes (Feldgröße) festgelegt. Die für Jetsam reservierten ersten beiden Bytes eines Datensatzes können nicht im **FIELD**-Kommando angesprochen werden. Die Satzlänge des Datensatzes errechnet sich somit als Summe der Feldgrößen der Datenfelder zuzüglich der 2 für Jetsam reservierten Bytes.

```
FIELD Datei-Symbol,Länge von Feld AS Name von Feld
```

Der unterstrichene Teil des Befehles kann beliebig oft wiederholt werden.
Um Werte in die Datenfelder zu schreiben, dürfen nur die Befehle **LSET** für linksbündige Wertzuweisung, **RSET** für rechtsbündige Wertzuweisung oder **MID$** benutzt werden:

```
LSET Feld aus der FIELD-Anweisung=String-Ausdruck
RSET Feld aus der FIELD-Anweisung=String-Ausdruck
MID$ (Feld aus der FIELD-Anweisung, Startposition ab der der String-Ausdruck
stehen soll[,Länge in der der String-Ausdruck übernommen werden soll])=String-
Ausdruck
```

**Beispiel:** - nicht in JETSAM.BAS enthalten -

Durch **FIELD** wurde ein Datenfeld mit **10 AS name$** bestimmt.

```
10 LSET name$="Hugo"
```
setzt **name$** auf "Hugo ".

```
10 RSET name"="Hugo"
```
setzt **name$** auf " Hugo" .

```
10 MID$(name$,3,2)="Hugo"
```
setzt **name$** auf " Hu ".

**MID$** wird hierbei als Kommando und nicht als Funktion verwendet.

**Beispiel:**

```
60 FIELD #1,20 AS xname$,20 AS xvorname$,2 AS xplz$,20 AS xort$,1 AS xkunde$
870 LSETxname$="Installation":LSETxvorname$="Krause":xplz=2390:LSET xort$=
"Flensburg":LSET xkunde$="K"
1000 LSET xplz$=MKI$(xplz)
```

In Zeile 60 wird zunächst der Datensatz beschrieben. Der Datensatz ist 20 + 20 + 2 + 20 + 1 zuzüglich 2 Bytes für die Satznummer, also 65 Bytes lang. In Zeile 870 und 1000 werden jeweils über **LSET** den Datenfeldern Werte zugewiesen.

Mit der Funktion **RANKSPEC** wird für eine Schlüsselreihe festgelegt, ob in der Schlüsselreihe identische Schlüsselwerte vorkommen dürfen oder nicht. Diese Kennung der Schlüsselreihe macht sich bei den Befehlen **ADDREC** bzw. **ADDKEY** ggf. durch den Fehlercode 116 bemerkbar.

Die Funktion wirkt sich stets nur für nachfolgende Veränderungen der Jetsam-Datei aus. Wenn man es möchte, könnte man eine Schlüsselreihe, die so gekennzeichnet ist, daß keine mehrfachen Schlüssel vorkommen dürfen, vorübergehend so kennzeichnen, daß mehrfache Schlüssel zulässig sind, dann beliebig viele Datensätze mit gleichen Schlüsselwerten in die Datei einstellen und danach die Schlüsselreihe wieder so kennzeichnen, daß keine mehrfachen Schlüssel zulässig sind.

Nach Ausführung sollte die Jetsam-Datei durch den **CONSOLIDATE**-Befehl wieder stimmig markiert werden.



RANKSPEC(Datei-Symbol,Schlüsselreihe,Information identische Schlüsselwerte in der Schlüsselreihe zulässig (Wert 0) oder nicht (Wert1))

**Beispiel:**

```
70 ergebnis=RANKSPEC(#1,0,1)
```

Da standardmäßig angenommen wird, daß mehrfache Schlüssel zulässig sind, braucht im Programm nur angegeben werden, in welchen Schlüsselreihen mehrfache Schlüssel nicht zulässig sein sollen. Hier wird für Schlüsselreihe 0 festgelegt, daß nur einmalige Schlüssel zulässig sind und bei Einfügen eines Schlüssels in Reihe 0 ggf. als Antwortcode 116 auszugeben ist.

Durch die Funktion **ADDREC** werden zwei Aufgaben zugleich erledigt:

1. Der Datensatz wird in die Datendatei eingestellt.
2. Es wird eine Schlüsselinformation für den Datensatz in die Schlüsseldatei eingestellt - Schlüsselreihe und Schlüsselwert - und Jetsam vergibt eine Satznummer für den Datensatz, die in der Schlüsseldatei zu dem Schlüssel abgelegt wird. Die niedrigste Satznummer, die von Jetsam vergeben wird, ist 2. Der Schlüsselwert wird in aufsteigender Sortierfolge in die Schlüsseldatei eingefügt.

Nur durch diesen Befehl können Datensätze in die Datendatei geschrieben werden. Wird ein Fehlercode 116, 130 oder 131 zurückgemeldet, wurde der Datensatz nicht geschrieben und auch keine Schlüsselinformation in die Schlüsseldatei eingestellt.
Der Fehlercode 116 ist sehr hilfreich für Schlüsselreihen, die durch **RANKSPEC** so gekennzeichnet wurden, daß in der Schlüsselreihe nur eindeutige Schlüssel zulässig sind, d.h., ein Schlüssel darf nur einmal in der Schlüsselreihe vorkommen. Jetsam macht den Benutzer dadurch darauf aufmerksam, daß ein Datensatz mit einem solchen Schlüssel bereits in der Jetsam-Datei vorhanden ist und dieser

Datensatz aufgrund der Kennung der Schlüsselreihe durch **RANKSPEC** nicht in die Jetsamdatei eingefügt werden darf.

```
ADDREC(Datei-Symbol,Sperre für den neuen Satz,Schlüsselreihe in die der
Schlüsselwert eingetragen werden soll,Schlüsselwert)
```

**Beispiel:**

```
980 key$=LEFT$(xname$,5):IF LEN(xname$)<5 THEN key$=key$+SPACE$(5-LEN(xname$))
990 key$=key$+LEFT$(xvorname$,5):IF LEN(xvorname$)<5 THEN key$=key$+SPACE$(5-
LEN(xvorname$))
1010 ergebnis=ADDREC(#1,2,0,key$)
```

In den Zeilen 980 und 990 wird zunächst der Schlüsselwert aus den ersten 5 Buchstaben des Namens und Vornamens - ggf. aufgefüllt mit Leerstellen - gebildet, der dann in Zeile 1010 beim **ADDREC** als Schlüsselwert für die Schlüsselreihe 0 mitgegeben wird.

Die Beispieldatei baut sich in der Schlüsselreihe 0 Stück für Stück durch **ADDREC** wie folgt auf:

| | | | | | |
|---|---|---|---|---|---|
| Reihe 0 | InstaKraus | | | | |
| Satznr. | 2 | | | | |
| Reihe 0 | InstaKraus | Mayr Großh | | | |
| Satznr. | 2 | 3 | | | |
| Reihe 0 | ElektMaier | InstaKraus | Mayr Großh | | |
| Satznr. | 4 | 2 | 3 | | |
| Reihe 0 | ElektMaier | InstaKraus | Mayr Großh | XaverUnsin | |
| Satznr. | 4 | 2 | 3 | 5 | |
| Reihe 0 | ElektMaier | InstaKraus | LudwiElekt | Mayr Großh | XaverUnsin |
| Satznr. | 4 | 2 | 6 | 3 | 5 |



Mit der Funktion **ADDKEY** können weitere Schlüsselinformationen zu einem Datensatz in der Schlüsseldatei abgelegt werden. Über diese Schlüssel kann später wieder auf den Datensatz zugegriffen werden.

Durch die Funktion **ADDREC** wird bereits eine Schlüsselinformation zu einem Datensatz in die Schlüsseldatei geschrieben. Man möchte jedoch meistens über weitere Schlüssel auf den Datensatz zugreifen können. Diese zusätzlichen Schlüssel werden durch **ADDKEY** in die Schlüsseldatei eingefügt.
Es können weitere Schlüssel zu dem Datensatz in anderen Schlüsselreihen geschrieben werden. Dabei sollte nicht die Schlüsselreihe, in der durch **ADDREC** bereits ein Schlüssel eingestellt wurde, genommen werden.

Soll jedoch ein bestehender Schlüssel zu einem Datensatz in einer Schlüsselreihe geändert werden, so ist durch **ADDKEY** zunächst der neue Schlüssel in die Schlüsseldatei einzustellen, so daß vorübergehend in einer Schlüsselreihe zu einem Datensatz mehrere verschiedene Schlüssel vorhanden sind. Danach wird durch **DELKEY** der alte Schlüssel gelöscht. Diese Vorgehensweise ist erforderlich, damit nicht versehentlich ein Datensatz gelöscht wird. Hat ein Datensatz nur einen Schlüssel und würde zuerst der alte Schlüssel über **DELKEY** gelöscht, so wäre der Datensatz gelöscht und stünde nicht mehr zur Verfügung, selbst wenn unmittelbar danach durch ein **ADDKEY** wieder ein Schlüssel zu dem Datensatz eingefügt werden würde, was jedoch dann schon nicht mehr möglich ist.

Wenn man möchte, kann man sich jedoch die Eigenschaft zu Nutze machen, daß in einer Schlüsselreihe zu einem Datensatz mehrere verschiedene Schlüssel abgelegt werden können. Jedoch wird dadurch ggf. die Handhabung der Jetsam-Datei erschwert und fehleranfällig.

```
ADDKEY(Datei-Symbol,Sperre auf den Datensatz,Schlüsselreihe in der der
Schlüsselwert eingefügt werden soll,der einzufügende Schlüsselwert,Satznummer
des Datensatzes)
```

**Beispiel:**

```
1030 key$=MKIK$(xplz)
1040 satznr=FETCHREC(#1):ergebnis=ADDKEY(#1,2,1,key$,satznr)
1060 key$=xkunde$
1070 satznr=FETCHREC(#1):ergebnis=ADDKEY(#1,2,2,key$,satznr)
```

In den Zeilen 1030 und 1060 wird der neue Schlüsselwert gebildet; zum einen die Postleitzahl, die hier in verdichteter Form als Jetsam-Schlüssel benutzt wird (**MKIK$** siehe unten), und zum anderen die Kennung auf Kunde, Lieferant bzw. Geschäftsfreund.

In Zeile 1040 bzw. 1070 wird zunächst die aktuelle Satznummer des unmittelbar vorher durch **ADDREC** eingefügten Satzes ermittelt und dann wird die Schlüsselinformation in Reihe 1 bzw. Reihe 2 mit der ermittelen Satznummer in die Schlüsseldatei geschrieben. Als Sperre wurde hier eine Schreibsperre angegeben. Die Schlüsselreihen 1 und 2 bauen sich entsprechend wie Schlüsselreihe 0 Stück für Stück auf.

Die Positionierung des Programms, die aktuelle Position in einer Jetsam-Datei, wird von den drei Komponenten **Reihe, Schlüsselwert** und **Satznummer** bestimmt.

Die aktuelle Position in einer Jetsam-Datei wird durch die Befehle **FETCHKEY$, FETCHRANK** und **FETCHREC** ermittelt:

```
FETCHKEY$(Datei-Symbol)
FETCHRANK(Datei-Symbol)
FETCHREC(Datei-Symbol)
```

Diese drei Funktionen stellen jeweils eine Komponente der aktuellen Position zur Verfügung. Die Schlüsselreihe wird durch **FETCHRANK**, der Schlüsselwert durch **FETCHKEY$** und die Satznummer durch **FETCHREC** zur Verfügung gestellt.

Durch die Verarbeitung im Programm sind meistens nicht alle Komponenten der Positionierung bekannt. Oft kennt man nur



den Schlüsselwert, weil man in der Regel über einen Schlüsselwert auf die Datensätze zugreift. Seltener arbeitet man mit der Satznummer, die z.B. in der Suchfunktion **SEEKREC** benötigt wird. Die Schlüsselreihe ist meist bei einer sequentiellen Verarbeitung der Jetsam-Datei von Bedeutung (**SEEKRANK**).

Bei einigen Funktionen ist es sinnvoll und wichtig, die fehlenden Komponenten der aktuellen Position in der Jetsam-Datei zu kennen, wie z.B. für die Funktion **DELKEY** dann, wenn in der Schlüsselreihe zu mehreren Einträgen identische Schlüsselwerte zugelassen sind (kein **RANKSPEC** für die Schlüsselreihe wirksam). Es besteht sonst die Gefahr, daß die Funktion **DELKEY** fehlerhaft arbeitet und Schlüsselwerte unkontrolliert aus der Schlüsseldatei entfernt werden und damit die Jetsam-Datei unbrauchbar wird.

**Beispiel:**

| Reihe 0 | ElektMaier | InstaKraus | LudwiElekt | Mayr Großh | XaverUnsin |
|---|---|---|---|---|---|
| Satznr. | 4 | 2 | 6 | 3 | 5 |
| | | | ↓ | | |
| Reihe 1 | 2390 | 2390 | 8000 | 8000 | 8000 |
| Satznr. | 2 | 4 | 3 | 5 | 6 |
| Reihe 2 | G | K | K | L | L |
| Satznr. | 5 | 2 | 4 | 3 | 6 |

```
310 reihe=FETCHRANK(#1)
320 key$=FETCHKEY$(#1)
330 IF reihe=1 THEN key$=STR$(CVIK(FETCHKEY$(#1)))
340 PRINT "Die aktuelle Position ist auf Reihe"reihe",Schlüssel "key$" und
Satnummer"FETCHREC(#1)
```

**FETCHKEY$** stellt als Schlüssel 8000, **FETCHRANK** als Reihe 1, und **FETCHREC** als Satznummer der aktuellen Position 3 bereit. Da die Postleitzahl über die Funktion **MKIK$** verdichtet

gespeichert wurde (siehe Beispiel zu **ADDKEY**), muß sie hier über **CVIK** wieder aufbereitet werden.
- **MKIK$** und **CVIK** siehe unten -

Die Suchfunktionen richten eine aktuelle Position in der Jetsam-Datei ein.

```
SEEKKEY(Datei-Symbol,Sperre auf den Datensatz, zu durchsuchende
Schlüsselreihe,zu suchender Schlüsselwert)
```

Mit dieser Funktion werden Datensätze aufgrund eines bestimmten Schlüsselwertes in der Jetsam-Datei gesucht. Neben dem eigentlichen Schlüsselwert ist die Schlüsselreihe anzugeben, in der nach Schlüsseleinträgen mit dem angegebenen Schlüsselwert gesucht werden soll.

Die aktuelle Position wird auf den gefundenen Schlüssel in der Jetsam-Datei eingerichtet, so daß der Datensatz über ein **GET** eingelesen werden kann.

**Beispiel:**

| | | | | | |
|---|---|---|---|---|---|
| Reihe 0 | ElektMaier | InstaKraus | LudwiElekt | Mayr Groß h | XaverUnsin |
| Satznr. | 4 | 2 | 6 | 3 | 5 |
| Reihe 1 | 2390 | 2390 | 8000 | 8000 | 8000 |
| Satznr. | 2 | 4 | 3 | 5 | 6 |
| | | | | ↓ | |
| Reihe 2 | G | K | K | L | L |
| Satznr. | 5 | 2 | 4 | 3 | 6 |

```
420 ergebnis=SEEKKEY(#1,0,1,MKIK$(2300))
430 ergebnis=SEEKKEY(#1,0,2,"M")
440 ergebnis=SEEKKEY(#1,0,2,"L")
```

Das erste **SEEKKEY** ergibt den Antwortcode 105 - der Schlüsselwert wurde nicht gefunden, da in Reihe 1 kein Schlüsseleintrag mit dem Wert 2300 vorhanden ist. Die aktuelle



Position wird auf Reihe 1, Schlüssel 2390, Satznummer 2 gesetzt, weil der erste Schlüsseleintrag mit dem Wert 2390 einem Schlüsseleintrag mit dem Wert 2300 folgen würde.

Das **SEEKKEY** in Zeile 430 ergibt den Wert 103 - Schlüsselwert wurde nicht in dieser Reihe gefunden, es wurde das Ende der Datei erreicht. Es gibt keine höheren Schlüsselwerte in dieser oder einer folgenden Reihe, d.h., es konnte keine aktuelle Position eingerichtet werden.

Das **SEEKKEY** in Zeile 440 ergibt den Antwortcode 0 und setzt die aktuelle Position wie angezeigt auf Reihe 2, Schlüssel L, Satznummer 3, weil dies der erste Schlüsseleintrag mit dem gewünschten Schlüsselwert ist.

Mit **SEEKNEXT** wird auf den nächstfolgenden Schlüsseleintrag zugegriffen. Abhängig vom Antwortcode läßt sich ermitteln, ob der neue Schlüsseleintrag denselben Schlüsselwert hat wie der vorhergehende Schlüsseleintrag (Antwortcode = 0). In dieser Schlüsselreihe sind identische Schlüsselwerte zu verschiedenen Schlüsseleinträgen zulässig (siehe **RANKSPEC**) und es wurde ein weiterer Schlüsseleintrag mit demselben Schlüsselwert gefunden. Man befindet sich noch im selben Schlüsselsatz.
Durch den Antwortcode 101 wird angezeigt, daß der Schlüsselwert des Schlüsseleintrags vom vorherigen Schlüsselwert abweicht und in derselben Schlüsselreihe ist.
Wechselte die aktuelle Position in eine andere Schlüsselreihe, so ist der Antwortcode 102.

Gibt man bei dieser Funktion Werte für die Index-Position mit, wird ab dem damit angegebenen Schlüsseleintrag auf den nächsten Schlüsseleintrag zugegriffen (Bsp.: **ergebnis = SEEKNEXT(#1,0,1,"HUGO",43)**: Suche nach dem nächsten Schlüsseleintrag, der auf den Schlüsseleintrag in Schlüsselreihe 1 mit dem Schlüsselwert "HUGO" und der Satznummer 43 folgt).

Antwortcode 103 wird ausgegeben, wenn man sämtliche Schlüsseleinträge der Schlüsseldatei abgearbeitet hat und damit das Dateiende erreicht ist.

SEEKNEXT(Datei-Symbol,Sperre auf den Datensatz [,Schlüsselreihe, Schlüsselwert, Satznummer der aktuellen Position von der bei der Suche ausgegangen werden soll])

**Beispiel:**

| | | | | | |
|---|---|---|---|---|---|
| Reihe 0 | ElektMaier | InstaKraus | LudwiElekt | Mayr Großh | XaverUnsin |
| Satznr. | 4 | 2 | 6 | 3 | 5 |
| Reihe 1 | 2390 | 2390 | 8000 | 8000 | 8000 |
| Satznr. | 2 | 4 | 3 | 5 | 6 |
| | | ↓ | | | |
| Reihe 2 | G | K | K | L | L |
| Satznr. | 5 | 2 | 4 | 3 | 6 |

```
470 ergebnis=SEEKKEY(#1,0,2,"K")
480 ergebnis=SEEKNEXT(#1,0)
490 ergebnis=SEEKNEXT(#1,0)
500 ergebnis=SEEKNEXT(#1,0)
510 ergebnis=SEEKNEXT(#1,0)
520 ergebnis=SEEKNEXT(#1,0)
530 ergebnis=SEEKKEY(#1,0,0,0,"XaverUnsin"):ergebnis=SEEKNEXT(#1,0)
540 ergbenis=SEEKNEXT(#1,0,0,"WalteElekt",6)
550 ergebnis=SEEKNEXT(#1,0,2,"H",5)
```

Durch **SEEKKEY** wurde eine aktuelle Position in der angezeigten Datei eingerichtet.

Das erste **SEEKNEXT** ergibt den Antwortcode 0 und setzt die aktuelle Position auf Reihe 2, Schlüssel K und Satznummer 4.

Ein weiteres **SEEKNEXT** ergibt den Antwortcode 101, denn die aktuelle Position geht auf Reihe 2, Schlüssel L, Satznummer



3. Da der Schlüsselwert sich verändert hat, ist der bisherige Schlüsselsatz verlassen worden.

Mit einem weiteren **SEEKNEXT** erhalten wir wieder den Antwortcode 0, die aktuelle Position wechselt auf Reihe 2, Schlüssel L und Satznummer 6.

Ein nochmaliges **SEEKNEXT** (Zeile 510) ergibt den Antwortcode 103, weil das Dateiende erreicht wurde und keine aktuelle Position mehr eingerichtet werden kann. Wird jetzt wieder ein **SEEKNEXT** ohne Angabe einer Index-Position durchgeführt, käme als Antwortcode 115, weil von einer unbestimmten aktuellen Position aus kein **SEEKNEXT** durchgeführt werden kann.

Befinden wir uns mit der aktuellen Position am Ende einer Schlüsselreihe (siehe Zeile 530 - hier wird durch das **SEEKKEY** die aktuelle Position am Ende von Reihe 0 eingerichtet) und führen dann ein **SEEKNEXT** durch, erhalten wir als Antwortcode 102, weil die Schlüsselreihe gewechselt hat, und die aktuelle Position steht dann auf Reihe 1, Schlüssel 2390, Satznummer 2.

Bei dem **SEEKNEXT** in Zeile 540 erhalten wir den Antwortcode 105, weil die angegebene Indexposition nicht vorhanden ist. In diesem Fall bleibt die aktuelle Position unbestimmt.

In Zeile 550 erhalten wir ebenfalls Antwortcode 105, jedoch würde eine aktuelle Position auf Reihe 2, Schlüssel K Satznummer 2 eingerichtet werden, weil der Schlüsselwert K dem gewünschten Schlüsselwert H folgt - siehe S. 326 im BASIC-Benutzer-Handbuch.

Die Funktion **SEEKPREV** arbeitet genauso wie **SEEKNEXT**, jedoch wird nicht weiter zum Dateiende hin gesucht, sondern rückwärts zum Dateianfang. Der Antwortcode 103 deutet darauf hin, daß der Dateianfang erreicht wurde: vgl. dazu Zeile 650 im folgenden Beispiel - Die Position wird auf den ersten Schlüsseleintrag in der Datei gesetzt. Mit dem folgenden

**SEEKPREV** wird versucht, über den Dateianfang hinaus rückwärts zu suchen, was den Antwortcode 103 ergibt.

SEEKPREV(Datei-Symbol,Sperre auf den Datensatz [,Schlüsselreihe, Schlüsselwert, Satznummer der aktuellen Position von der bei der Suche ausgegangen werden soll])

**Beispiel:**

| | | | | | |
|---|---|---|---|---|---|
| Reihe 0 | ElektMaier | InstaKraus | LudwiElekt | Mayr Großh | XaverUnsin |
| Satznr. | 4 | 2 | 6 | 3 | 5 |
| Reihe 1 | 2390 | 2390 | 8000 | 8000 | 8000 |
| Satznr. | 2 | 4 | 3 | 5 | 6 |
| | | | | ↓ | |
| Reihe 2 | G | K | K | L | L |
| Satznr. | 5 | 2 | 4 | 3 | 6 |

```
580 ergebnis=SEEKKEY(#1,0,2,"L")
590 ergebnis=SEEKPREV(#1,0)
600 ergebnis=SEEKPREV(#1,0)
610 ergebnis=SEEKPREV(#1,0)
620 ergebnis=SEEKPREV(#1,0)
630 ergebnis=SEEKPREV(#1,0,0,"WalteElekt",6)
640 ergebnis=SEEKPREV(#1,0,2,"H",5)
650 ergebnis=SEEKKEY(#1,0,0,"ElektMaier"):ergebnis=SEEKPREV(#1,0)
```

Durch **SEEKKEY** wurde eine aktuelle Position in der angezeigten Datei eingerichtet.

Mit dem ersten **SEEKPREV** in Zeile 590 wird die aktuelle Position auf Reihe 2, Schlüssel K und Satznummer 4 gesetzt und ergibt den Antwortcode 101, weil der Schlüsselwert gewechselt hat.

Ein weiteres **SEEKPREV** ergibt den Antwortcode 0 und die aktuelle Position geht auf Reihe 2, Schlüssel K, Satznummer 2. Weder Schlüsselwert noch Schlüsselreihe haben gewechselt.



Mit dem **SEEKPREV** in Zeile 610 wechselt die aktuelle Position auf Reihe 2, Schlüssel G, Satznummer 5 mit Antwortcode 101, da der Schlüsselwert sich geändert hat.

Ein nochmaliges **SEEKPREV** setzt die aktuelle Position auf Reihe 1, Schlüssel 8000 und Satznummer 6. Antwortcode ist hier 102, weil die Schlüsselreihe gewechselt hat.

Zu den anderen Antwortcodes siehe die Ausführungen zu **SEEKNEXT**. In Zeile 630 wird Antwortcode 105 ausgegeben, die aktuelle Position ist unbestimmt. In Zeile 640 wird Antwortcode 105 ausgegeben, jedoch ist die aktuelle Position auf Reihe 2, Schlüssel K, Satznummer 2.

Die Funktion **SEEKRANK** hat nichts mit Seekrankheit zu tun, sondern ist als SEEK-RANK zu lesen: "Suche eine Schlüsselreihe!"

Durch diese Funktion wird die aktuelle Position des Programms in der Datei auf den Anfang einer Schlüsselreihe gesetzt. Dies ist sinnvoll, wenn man die Daten der Jetsam-Datei sequentiell, abhängig von einer bestimmten Schlüsselreihe abarbeiten möchte. Mit **SEEKRANK** geht man auf den ersten Schlüsseleintrag der Schlüsselreihe und mit **SEEKNEXT** von einem Schlüsseleintrag zum nächsten in der Schlüsselreihe, bis die Funktion **SEEKNEXT** den Antwortcode 102 oder 103 zurückmeldet - d.h., es sind sämtliche Schlüsseleinträge dieser Schlüsselreihe abgearbeitet worden bzw. das Ende der Datei wurde erreicht.

SEEKRANK(Datei-Symbol, Sperre auf den ersten Satz in der Schlüsselreihe, Schüsselreihe)

**Beispiel:**

| Reihe 0 | ElektMaier | InstaKraus | LudwiElekt | Mayr Großh | XaverUnsin |
|---|---|---|---|---|---|
| Satznr. | 4 | 2 | 6 | 3 | 5 |
| | ↓ | | | | |
| Reihe 1 | 2390 | 2390 | 8000 | 8000 | 8000 |
| Satznr. | 2 | 4 | 3 | 5 | 6 |
| Reihe 2 | G | K | K | L | L |
| Satznr. | 5 | 2 | 4 | 3 | 6 |

```
680 ergebnis=SEEKRANK(#1,0,1)
690 ergebnis=SEEKRANK(#1,0,4)
```

Das **SEEKRANK** in Zeile 680 ergibt den Antwortcode 0 und setzt wie angezeigt die aktuelle Position auf den Beginn der Schlüsselreihe 1, d.h., auf Reihe 1, Schlüssel 2390 und Satznummer 2.

In Zeile 690 ergibt **SEEKRANK** den Antwortcode 103, weil zu der angegebenen Schlüsselreihe keine Schlüsseleinträge gefunden werden können.

Gäbe es in unserer Beispieldatei die Schlüsselreihen 0 und 1 nicht, ergäbe **ergebnis=SEEKRANK(#1,0,0)** den Antwortcode 102 und würde die aktuelle Position auf Reihe 2, Schlüssel G und Satznummer 5 einrichten. Die gewünschte Schlüsselreihe konnte nicht gefunden werden, jedoch konnte in einer folgenden Schlüsselreihe eine aktuelle Position eingerichtet werden.

Hat man sich die Index-Position, d.h. Reihe, Schlüsselwert und Satznummer, einer Position in der Datei zwischengespeichert, kann die durch diese drei Werte bestimmte Position in der Datei als aktuelle Position durch

```
SEEKREC(Datei-Symbol,Sperre auf den Datensatz [,Schlüsselreihe, Schlüsselwert,
Satznummer der Position die als aktuelle Position eingerichtet werden soll])
```



wieder hergestellt werden, um z.B. eine sequentielle Verarbeitung der Datei über **SEEKNEXT** von der zwischengespeicherten Position ab fortzusetzen.

**Beispiel**

| Reihe 0 | ElektMaier | InstaKraus | LudwiElekt | Mayr Großh | XaverUnsin |
|---|---|---|---|---|---|
| | | | | ↓ | |
| Satznr. | 4 | 2 | 6 | 3 | 5 |
| Reihe 1 | 2390 | 2390 | 8000 | 8000 | 8000 |
| Satznr. | 2 | 4 | 3 | 5 | 6 |
| Reihe 2 | G | K | K | L | L |
| Satznr. | 5 | 2 | 4 | 3 | 6 |

```
720 reihe=0:schluessel$="Mayr Großh":satznummer=3
730 ergebnis=SEEKREC(#1,0,reihe,schluessel$,satznummer)
740 ergebnis=SEEKREC(#1,0,0,"XaverFranz",6)
750 ergebnis=SEEKREC(#1,0,2,"M",6)
```

In Zeile 720 werden in Variablen die Werte für eine Index-Position angegeben, die dann in Zeile 730 im **SEEKREC** zur Einrichtung einer aktuellen Position benutzt werden. Antwortcode ist 0, die aktuelle Position wird auf Reihe 0, Schlüssel "Mayr Großh" mit Satznummer 3 - wie angezeigt - eingerichtet.

In Zeile 740 wird Antwortcode 105 ausgegeben, die Indexposition konnte nicht gefunden werden, weil in Reihe 0 kein entsprechender Schlüsseleintrag vorhanden ist.

In Zeile 750 wird Antwortcode 103 ausgegeben, die Indexposition konnte nicht gefunden werden - es wurde das Ende der Datei erreicht.

Jetsam faßt gleiche Schlüsselwerte in einer Reihe zu sogenannten Schlüsselsätzen zusammen. **SEEKSET** setzt die aktuelle Position auf den Beginn des nächsten, der aktuellen

Position folgenden Schlüsselsatzes. Ggf. besteht ein Schlüsselsatz nur aus einem Schlüsseleintrag. Die aktuelle Position wird stets auf den nächsten, vom momentanen Schlüsselwert abweichenden Schlüsseleintrag gesetzt.

**SEEKSET** ist ähnlich wie **SEEKNEXT**, nur daß **SEEKNEXT** stets zum nächsten Schlüsseleintrag geht, egal wie der neue Schlüsseleintrag aussieht, während **SEEKSET** die Schlüsseleinträge überspringt, die denselben Schlüsselwert haben wie der aktuelle Schlüsselwert.

Aufgrund der Antwortcodes läßt sich erkennen, ob die aktuelle Position noch in derselben Schlüsselreihe ist (Code 101) oder ob die aktuelle Position in eine andere Schlüsselreihe gewechselt hat (Code 102).

Antwortcode 103 deutet darauf hin, daß das Ende der Datei erreicht wurde.

Antwortcode 0 wird bei dieser Funktion nicht ausgegeben.

SEEKSET(Datei-Symbol, Sperre auf den Datensatz [Schlüsselreihe, Schlüsselwert, Satznummer der aktuellen Position von der bei der Suche ausgegangen werden soll])

**Beispiel:**

| | | | | | |
|---|---|---|---|---|---|
| Reihe 0 | ElektMaier | InstaKraus | LudwiElekt | Mayr GroBh | XaverUnsin |
| Satznr. | 4 | 2 | 6 | 3 | 5 |
| | ↓ | | | | |
| Reihe 1 | 2390 | 2390 | 8000 | 8000 | 8000 |
| Satznr. | 2 | 4 | 3 | 5 | 6 |
| Reihe 2 | G | K | K | L | L |
| Satznr. | 5 | 2 | 4 | 3 | 6 |

```
780 ergebnis=SEEKKEY(#1,0,1,MKI$(2390))
790 ergebnis=SEEKSET(#1,0)
800 ergebnis=SEEKSET(#1,0)
810 ergebnis=SEEKSET(#1,0)
820 ergebnis=SEEKSET(#1,0)
830 ergebnis=SEEKSET(#1,0)
840 ergebnis=SEEKSET(#1,0)
```



Von der aktuellen Position ausgehend ergibt **SEEKSET** in Zeile 790 den Antwortcode 101, d.h., die neue aktuelle Position auf Reihe 1, Schlüssel 8000, Satznummer 3 ist in der Schlüsselreihe geblieben.

Ein nochmaliges **SEEKSET** in Zeile 800 setzt die aktuelle Position auf Reihe 2, Schlüssel G und Satznummer 5 und ergibt den Antwortcode 102, da die Schlüsselreihe gewechselt hat.

Mit zwei weiteren **SEEKSET** (Zeilen 810 und 820) springt man vor auf Reihe 2, Schlüssel L und Satznummer 3, jeweils mit Antwortcode 101.

Ein weiteres **SEEKSET** (Zeile 830) ergibt den Antwortcode 103, da das Ende der Jetsam-Datei erreicht wurde, die aktuelle Position ist unbestimmt.

Von einer unbestimmten aktuellen Position aus kann kein **SEEKSET** ausgeführt werden - Antwortcode 115 (Zeile 840). Zu Antwortcode 115 siehe die Erläuterungen bei **SEEKNEXT**. Statt **SEEKNEXT** ist dort **SEEKSET** einzusetzen.

Wurde durch eine der **SEEK**-Funktionen die aktuelle Position eingerichtet, kann durch den **GET**-Befehl der zugehörige Datensatz eingelesen werden, denn bisher wurde nur der Schlüssel aus der Schlüsseldatei gelesen, aber noch nicht der Datensatz aus der Datendatei.

**GET** stellt den Datensatz aus der Datendatei, der zum aktuellen Schlüsseleintrag (Reihe, Schlüsselwert und Satznummer) gehört, in einen Satzpuffer. Dieser steht dann dem Programm

zur Verfügung. Über die im **FIELD**-Kommando festgelegten Feldnamen der Datenfelder des Datensatzes im Satzpuffer kann auf den Inhalt des Datensatzes zugegriffen werden.
Der Satzpuffer ist ähnlich wie der Pufferbereich, der durch **BUFFERS** für die Schlüsseldatei bestimmt wird, ein besonderer Speicherbereich, über den Datensätze eingelesen und geschrieben werden.

Durch Angabe einer Satznummer kann jeder beliebige Datensatz eingelesen werden. Die Satznummer sollte jedoch durch die Funktion **FETCHREC** ermittelt sein, damit nicht ungültige Datensätze verarbeitet werden.

```
GET Dateisymbol [, Satznummer des zu lesenden Satzes]
```

oder

```
GET Datei-Symbol, [Satznummer des zu lesenden Satzes], Sperre auf den
Datensatz
```

**Beispiel:**

```
350 IF ergebnis<103 THEN GET #1
```

Hier wird durch das vorangestellte **IF** auf den Ergebniswert der letzten Jetsam-Funktion sichergestellt, daß nur dann ein **GET** durchgeführt wird, wenn eine aktuelle Position eingerichtet werden konnte. Andernfalls würde die Fehlermeldung **record number error** ausgegeben werden.

Durch **PUT** werden bestehende Datensätze einer Jetsam-Datei überschrieben, d.h., der Dateninhalt des Datensatzes wird verändert.
Mit diesem Befehl können keine neuen zusätzlichen Datensätze in die Jetsam-Datei eingestellt werden. Dies ist nur durch den Befehl **ADDREC** mögich.

```
PUT Datei-Symbol [, Satznummer des zu verändernden Satzes]
```

oder

```
PUT Datei-Symbol, [Satznummer des zu verändernden Satzes], Sperre auf den
Datensatz
```



Wird keine Satznummer mitgegeben, wird der Datensatz, der zur aktuellen Position in der Jetsam-Datei gehört, überschrieben. Durch Angabe einer Satznummer kann jeder beliebige Datensatz überschrieben werden, jedoch sollte die Satznummer durch **FETCHREC** bestimmt werden, um keine ungültigen Datensätze zu verarbeiten.

**Beispiel:**

```
1110 PRINT"Lesen der Daten zum Geschäftsfreund": ergebnis=SEEKKEY(#1,2,2,"G"):
GET #1
1120 PRINT"Verändern der Ortsangabe von "xort$" auf ";:LSET xort$="München
340":PRINT xort$
1130 PRINT"Zurückschreiben des Satzes":PUT #1
1140 PRINT"Lesen der Daten zu Xaver Unsinn": ergebnis=SEEKKEY(#1,1,0,"Xaver
Unsin"):GET #1
1150 PRINT xvorname$" "xname$" "CVI(xplz$)" "xort$" "xkunde$
```

Wie an den erläuternden Texten der **PRINT**-Befehle zu sehen ist, wird hier ein Datensatz zunächst über **SEEKKEY** und **GET** eingelesen - Zeile 1110, und dann durch **LSET** und **PUT** ein Feldinhalt verändert sowie der Datensatz in die Datendatei zurückgeschrieben - Zeilen 1120 und 1130. In Zeile 1140 wird der Datensatz über einen anderen Schlüssel erneut eingelesen und in Zeile 1150 werden alle Datenfelder ausgegeben, um die Veränderung der Ortsangabe zu prüfen.

In den Datensätzen sind in der Regel sämtliche Schlüssel enthalten, durch die man diesen Datensatz über die Schlüsseldatei ermitteln kann. Wird durch ein **PUT** ein Datenfeld, das als Schlüssel benutzt wird, verändert, muß zusätzlich über **ADDKEY/DELKEY** (siehe dort) die Schlüsselinformation in der Schlüsseldatei angepaßt werden, denn Jetsam paßt die Schlüsselinformation nicht von selbst an.
Da dies sehr leicht vergessen wird, sollte ein Datensatz nicht über **PUT** geändert werden, sondern:

1. durch Lesen des zu ändernden Datensatzes (**SEEK**-Funktion und **GET**),
2. Löschen sämtlicher Schlüssel zum alten Datensatz

**(DELKEY)**,

3. Verändern des Inhalts des Datensatzes im Satzpuffer **(LSET, RSET, MID$)** und
4. als neuen Datensatz über **ADDREC/ADDKEY** wieder in die Jetsam-Datei

zurückgeschrieben werden. Diese Vorgehensweise ist letztlich sicherer, als die Arbeitsweise mit **PUT** und Anpassen der Schlüsselinformationen in der Schlüsseldatei.

Zum Löschen von Datensätzen wird die Funktion **DELKEY** benutzt.

Die Datensätze selbst werden in der Jetsam-Datei nicht gelöscht. Es werden lediglich sämtliche Schlüssel zu dem Datensatz in der Schlüsseldatei durch **DELKEY** entfernt. Damit steht der dazugehörige Datensatz nicht mehr zur Verfügung, weil Jetsam die Satznummer als wieder belegbar gekennzeichnet hat. Solange die Satznummer nicht wieder belegt wird, bleibt dieser Datensatz als ungültiger Datensatz in der Datendatei stehen.

**Entgegen der Darstellung im BASIC-Benutzer-Handbuch** wird durch das Löschen sämtlicher Schlüssel einer Jetsam-Datei die Jetsam-Datei nicht gelöscht. Es sind dann nur sämtliche Datensätze als gelöscht gekennzeichnet und die Jetsam-Datei steht wieder so zur Verfügung, wie nach der Durchführung eines **CREATE**. Ggf. ist die Jetsam-Datei zu löschen, indem die Daten- und die Schlüsseldatei über **KILL** gelöscht werden.

Die aktuelle Position in der Jetsam-Datei wird durch **DELKEY** verändert. Die Antwortcodes geben darüber näher Auskunft. Siehe auch die Ausführungen im Beispiel zu **SEEKNEXT**.

```
DELKEY(Datei-Symbol,Sperre auf den Datensatz der zu der auf den zu löschenden
Schlüssel nachfolgenden Indexposition gehört[,Schlüsselreihe in der der
Schlüssel gelöscht werden soll,zu löschender Schlüsselwert,Satznummer des
Datensatzes zu dem der Schlüssel gelöscht werden soll])
```



**Beispiel**

```
1190 PRINT"Lesen des ersten Lieferanten":ergebnis=SEEKKEY(#1,2,2,"L")
1200 PRINT"Löschen des Schlüssels - hier der Kennung Lieferant":satznr=
FETCHREC(#1):ergebnis=DELKEY(#1,2,2,"L",satznr)
1210 PRINT"Lesen der Postleitzahl zu diesem Datensatz als Schlüssel":ergebnis
=SEEKKEY(#1,2,1,MKIK$(8000))
1220 PRINT"Löschen des Schlüssels - hier der Postleitzahl":ergebnis=DELKEY
(#1,2,1,MKIK$(8000),satznr)
1230 PRINT"Lesen des kombinierten Schlüssels Name und Vorname":ergebnis=
DELKEY(#1,2,0,"Mayr Großh",satznr)
1240 PRINT"Löschen des Schlüssels - hier Name und Vorname":ergebnis=DELKEY
(#1,2,0,"Mayr Großh",satznr)
1250 PRINT"Erst jetzt sind alle Informationen zu Großhandel Mayr gelöscht!!"
1260 ergebnis=SEKKEY(#1,2,0,"Mayr Großh")
1270 ergebnis=SEEKKEY(#1,2,1,MKIK$(8000))
1280 ergebnis=SEEKKEY(#1,2,2,"L")
```

Die erläuternden **PRINT**-Texte beschreiben, was in der Zeile gerade geschieht. Je Schlüsselreihe wird zunächst der Schlüssel gelesen, die Satznummer des Datensatzes ermittelt (nur in Zeile 1200) und dann wird über **DELKEY** der Schlüssel aus der Schlüsseldatei gelöscht.
Sind in allen Schlüsselreihen die Schlüssel zu Mayr Großhandel gelöscht, so ist der Datensatz in der Jetsam-Datei als ungültig gekennzeichnet, da wir keinen Schlüssel mehr haben, über den auf den Datensatz zugegriffen werden kann. Dies wird auch durch **SEEKKEY** in den Zeilen 1260 bis 1280 angezeigt. In Zeile 1260 erhalten wir Antwortcode 105, in Zeile 1270 und 1280 bekommen wir Antwortcode 0, jedoch nur, weil noch andere Schlüsseleinträge mit dem selben Schlüsselwert, wie ihn der gelöschte Datensatz hatte, vorhanden sind, d.h., es wird auf andere Datensätze hingewiesen.

Der Befehl **OPTION FIELD** sollte normalerweise nicht mehr benutzt werden, damit das Programm nicht unnötig fehleranfällig wird. Er ist nur aus Verträglichkeitsgründen zu älteren BASIC-Versionen aufgenommen worden, in denen es möglich war, die zwei für Jetsam in den Datensätzen

reservierten Bytes, in denen die Satznummer notiert ist, dem Benutzer zugänglich zu machen. Auf eine weitere Beschreibung des Befehls wird daher verzichtet. Ggf. kann die Satznummer jetzt besser über **FETCHREC** ermittelt werden.

Der Befehl **LOCK** hat nur in einem Mehrbenutzer-System Bedeutung, um Satzsperren gezielt einzurichten oder aufzuheben.

Die Satzsperren spielen in fast allen Jetsam-Befehlen und -Funktionen eine Rolle. Diese Sperren sind in der Regel jedoch nur vorübergehend eingerichtet. Durch eine der **SEEK**-Funktionen wird eine Lesesperre über einen Satz errichtet, die erst wieder aufgehoben wird, wenn auf einen anderen Satz durch eine **SEEK**-Funktion zugegriffen wird.

Durch die **LOCK**-Funktion kann ein Benutzer versuchen, eine Sperre auf einen Datensatz zu ändern oder zu löschen, um den Satz wieder zur allgemeinen Benutzung durch andere Benutzer freizugeben. Die Satznummer, die mitgegeben wird, sollte durch die Funktion **FETCHREC** ermittelt sein.

Es kann hierdurch nur eine Sperre für einen einzelnen Datensatz errichtet werden, nicht auf die gesamte Jetsam-Datei. Eine Sperre auf die gesamte Datei wird durch **CREATE** oder durch **OPEN** eingerichtet.

```
LOCK(Datei-Symbol,Art der Sperre,Satznummer des Datensatzes zu dem die Sperre
eingerichtet werden soll)
```

Wird als Sperre der Wert 0 angegeben, wird eine vorher eingerichtete Sperre gelöscht. Mit dem Wert 1 wird eine bestehende Sperre auf den Datensatz zur Lesesperre geändert, d.h. ein anderer Benutzer darf den Datensatz lesen, aber nicht verändern. Mit dem Wert 2 wird eine bestehende Sperre auf den Datensatz zu einer Schreibsperre geändert, d.h. ein anderer Benutzer darf den Datensatz weder lesen noch verändern.



Die von den Jetsam-Funktionen zurückgemeldeten Antwortcodes hinsichtlich des Setzens von Sperren (Antwortcodes 130 bis 133) sind bei dem zum JOYCE mitgelieferten BASIC unwichtig, da das BASIC ein Einzelbenutzersystem ist (siehe den Befehl **VERSION** mit dem Parameter 2: **VERSION(2)** meldet eine 1 zurück - siehe S. 351 im BASIC-Benutzer-Handbuch). Wer jedoch auf dem JOYCE BASIC-Programme mit Jetsam entwickelt, die später in eine Mehrbenutzerumgebung übernommen werden, sollte diese Antwortcodes berücksichtigen und entsprechende Routinen im Programm vorsehen, damit auf diese Bedingungen vom Programm oder vom Benutzer her eingegangen werden kann.

In den Jetsam-Funktionen **ADDKEY, DELKEY, LOCK** und **SEEK** sind in der Regel Angaben zum Setzen von Sperren zu machen. Damit die Funktionen ordnungsgemäß arbeiten können, sind die Sperren auch in der Einzelbenutzerumgebung wichtig und sollten der Funktion entsprechend gesetzt werden (z. B. Schreibsperre bei **ADDREC** und nicht nur Lesesperre).

**Platz sparen ist die Devise**

Um die Datensätze nicht unnötig groß definieren zu müssen, gibt es zur Speicherung von Zahlen einige Möglichkeiten, diese in verdichteter Form im Speicher abzulegen:

Doppelt genaue Zahlen

**MKD$** 8 Byte langer String Rückumwandlung über **CVD**

```
10 dopp$=MKD$(11/3)
20 erg#=CVD(dopp$)
```

Einfach genaue Zahlen

**MKS$** 4 Byte langer String Rückumwandlung über **CVS**

```
10 einf$=MKS$(11/3)
20 erg!=CVS(einf$)
```

Integer-Werte

**MKI$** 2 Byte langer String Rückumwandlung über **CVI**

```
10 int$=MKI$(32500)
```

```
20 erg%=CVI(int$)
```

Möglicher Zahlenbereich -32768 bis +32767

Interne Darstellung:

Integer-Zahlen werden im Hexadezimalen Zahlensystem durch Werte von 0000H bis 0FFFFH dargestellt. (Siehe BASIC-Funktion **HEX$**). Die interne Darstellung umfaßt daher stets nur zwei Bytes. Durch **MKI$** wird diese interne Darstellung in einen String-Ausdruck überführt. Dabei werden die beiden Bytes miteinander vertauscht.

**Beispiel:** Dezimalzahl 32500 ist hexadezimal 7EF4H; Speicherung durch **MKI$** als 0F47EH. Die Funktion **CVI** wandelt den String 0F47EH wieder zurück in die Zahl 32500.

**MKIK$** 2 Byte langer String Rückumwandlung über **CVIK**

```
10 intk$=MKIK$(11/3)
20 erg%=CVIK(intk$)
```

Der hiermit erzeugte String kann als Jetsam-Schlüssel benutzt werden, d.h., dieser String kann über **ADDREC/ADDKEY** in die Schlüsseldatei eingefügt werden.

Möglicher Zahlenbereich -32768 bis +32767

Interne Darstellung:

Die erste Hälfte des ersten Bytes wird zur Darstellung des Vorzeichens benutzt (8 bis F für positive Zahlen und 0 bis 7 für negative Zahlen). In den verbleibenden 1 1/2 Byte wird die Zahl als normale hexadezimale Zahl dargestellt.

**Beispiel:** Dezimalzahl +32500 wird zu 0FEF4H; Dezimalzahl -32500 (hexadezimal 810CH) wird zu 010CH; +10 (000AH) wird zu 800AH und -10 (0FFF6H) wird zu 7FF6H.

**MKUK$** 2 Byte langer String Rückumwandlung über **CVUK**

```
10 intk2$=MKUK$(11/3)
20 erg%=CVUK(intk2$)
```



Diese Funktion arbeitet wie **MKIK$**, nur daß der Zahlenbereich aufgrund des fehlenden Vorzeichens 0 bis 65535 beträgt. Der erzeugte String kann ebenfalls als Jetsam-Schlüssel verwendet werden.

Interne Darstellung:

Die interne Darstellung ist mit der hexadezimalen Zahl identisch.

Die durch die **MKX$**-Funktionen erzeugten Strings sollten nur in der Weise verwendet werden, daß sie anderen Strings zugewiesen werden.

z.B.: **xzahl$=MKI$(integerzahl)**

Die hierdurch erreichte Platzersparnis ist recht beachtlich, denn in Normaldarstellung werden z.B. 2 Byte Länge mit Zahlen ab 100 überschritten, weil diese Zahlen schon mindestens drei Stellen umfassen.

Hier ein kleiner Überblick, an welcher Stelle in einem Programm die Jetsam-Befehle anzuwenden sind:

Anfangsroutinen

**BUFFERS**

**OPEN** oder **CREATE**, abhängig davon, ob die Schlüsseldatei und Datendatei bereits vorhanden sind oder nicht (feststellen über **FIND$**, jeweils für die Schlüssel- und Datendatei getrennt)

**FIELD**-Befehle für die Datensätze der Jetsam-Datei

**RANKSPEC** zur Angabe, ob für bestimmte Schlüsselreihen nur eindeutige oder mehrfache Schlüssel zulässig sind.

Verarbeitungsroutinen

Einlesen von Daten

Über eine **SEEK**-Funktion (**SEEKKEY, SEEKNEXT, SEEKPREV, SEEKRANK, SEEKREC** und/oder **SEEKSET**) die aktuelle Position in der Jetsam-Datei auf den gewünschten Satz einrichten und durch ein **GET** den Datensatz einlesen.

Verändern von Daten

Einfügen

Eingaberoutine zur Entgegennahme der neuen Daten, z.B. **INPUT**

Einfügen der eingegebenen Daten in eine Jetsam-Datei:

**ADDREC** zum Einfügen eines Schlüssels in die Schlüsseldatei und des Datensatzes in die Datendatei

**ADDKEY** zum Einfügen der weiteren Schlüssel in die Schlüsseldatei

Überschreiben

Ermitteln der richtigen Satznummer über **FETCHREC**

Neuschreiben des Datensatzes durch **PUT #1, Satznummer**

Soll ein Schlüsselfeld überschrieben werden, dann zusätzlich Einfügen des neuen Schlüssels über **ADDKEY** und erst dann Löschen des alten Schlüssels über **DELKEY**



Löschen

Löschen sämtlicher Schlüssel zum Datensatz durch entsprechend häufiges Ausführen von **DELKEY** für jeden Schlüssel des zu löschenden Datensatzes. Stets die durch **FETCHKEY$, FETCHRANK** und **FETCHREC** ermittelte Index-Position angeben, um ein korrektes Löschen der Schlüssel sicherzustellen.

Die Routinen zum Einfügen, Überschreiben und Löschen sollten stets mit **CONSOLIDATE** abgeschlossen werden.

Enderoutinen

Das Schließen einer Jetsam-Datei muß ausdrücklich durch **CLOSE** mit Angabe der Dateinummer geschehen, da sonst die Schlüssel- und Datendatei als nicht stimmig (nicht konsistent zueinander) gekennzeichnet sind.

## Unterschied von Funktion und Kommando im BASIC

Vielen ist sicherlich aufgefallen, daß im BASIC-Benutzer-Handbuch bei der ausführlichen Beschreibung der einzelnen Befehle rechts oben in der Ecke steht, ob es sich um eine Funktion oder ein Kommando handelt.

Bei einer Funktion wird nach Ausführung des Befehls in einer Variablen ein Ergebnis an das Programm bzw. an den Anwender zurückgegeben. Bei den Jetsam-Funktionen werden die Antwortcodes der Funktion bereitgestellt. Bei anderen Funktionen wird in der Variable ein Ergebnis bereitgestellt, das dann weiter verarbeitet werden kann.
Der Befehlsaufbau sieht grundsätzlich wie folgt aus:

```
funktionsergebnis=BASIC-FUNKTION(funktionsausgangswert)
```

Eine Funktion kann also nie für sich allein ausgeführt werden, sondern nur in Verbindung mit einer Wertzuweisung des Funktionsergebnisses an eine Variable - wie hier durch das Gleichheitszeichen an die Variable **funktionsergebnis**. In

Klammern sind die Ausgangswerte der Funktion anzugeben, aus denen das Ergebnis erzeugt werden soll.

**Beispiel:**

Berechnen des Cosinuswertes eines Winkels im Bogenmaß

```
10 cosinus=COS(2)
```

Die numerische Variable **cosinus** nimmt das Funktionsergebnis auf. Mit **COS** wird die BASIC-Routine zur Berechnung eines Cosinuswertes aufgerufen. In Klammern ist der Wert angegeben, zu dem der Cosinus berechnet werden soll, hier die Zahl 2. Durch das Gleichheitszeichen wird der Cosinuswert der Variablen **cosinus** zugewiesen und steht somit dem Anwender als Rechenergebnis zur Verfügung.

Eine Besonderheit sind die Jetsam-Funktionen. Sie sind Funktionen, weil von ihnen Funktionsergebnisse erzeugt werden, die abgefragt werden können - die Antwortcodes. Neben der Erzeugung des Antwortcodes bewirken sie einen Zugriff auf die Jetsam-Datei, suchen nach Daten, speichern Daten, löschen Daten etc. Abhängig von den Antwortcodes kann das Programm dann mit den Daten weiter arbeiten.

Die Antwortcodes der Jetsam-Funktionen lassen sich in folgende Gruppen einteilen:

Funktion wurde erfolgreich durchgeführt - Antwortcode ist 0

Funktion wurde erfolgreich durchgeführt - zusätzlich wird darauf hingewiesen, daß der Schlüsselwert (Antwortcode 101) oder die Schlüsselreihe (Antwortcode 102) gewechselt hat.

Funktion konnte nicht erfolgreich durchgeführt werden:
Antwortcode 103 zeigt an, daß das Ende der Datei erreicht wurde
Antwortcode 105 sagt aus, daß der Schlüssel nicht gefunden wurde
Antwortcode 115 zeigt an, daß keine aktuelle Position



eingerichtet ist
Antwortcodes ab 130 sind nur in Mehrbenutzer-Systemen von Bedeutung, weil sie Auskunft darüber geben, ob die Sperren wie gewünscht eingerichtet werden konnten oder nicht.

Ein Kommando steht immer für sich allein. Es wird kein Ergebniswert erzeugt, der einer Variablen zugewiesen werden müßte. Es wird stets direkt etwas bewirkt.

**Beispiel:**

Schließen von Dateien

```
CLOSE
```

Sämtliche Dateien sind geschlossen. Soll nur eine bestimmte Datei geschlossen werden kann zur genaueren Bestimmung der Wirkungsweise des Kommandos die Dateinummer mitgegeben werden (**CLOSE #1**). Eine solche Möglichkeit zur Steuerung der Wirkungsweise der Kommandos findet sich bei vielen BASIC-Kommandos.

Hier das vollständige Listing des Jetsam-Beispielprogramms JETSAM.BAS:

```
10 e$=CHR$(27):cl$=e$+"H"+e$+"J"
20 OPTION FILES "M:"
30 BUFFERS 20
40 PRINT"Beispielprogramm für die Jetsam-Funktionen und Befehle":GOSUB 380
50 IF FIND$("test.dat")<>"" AND FIND$("test.ind")<>"" THEN OPEN
"K",#1,"test.dat","test.ind",2 ELSE CREATE #1,"test.dat","test.ind",2
60 FIELD #1,20 AS xname$,20 AS xvorname$,2 AS xplz$,20 AS xort$,1 AS xkunde$
70 ergebnis=RANKSPEC(#1,0,1)
80 PRINT cl$
90 GOSUB 860'Addrec und Addkey
100 PRINT cl$
110 GOSUB 410'Seekkey
120 PRINT cl$
130 GOSUB 460'Seeknext
140 PRINT cl$
150 GOSUB 570'Seekprev
160 PRINT cl$
170 GOSUB 670'Seekrank
180 PRINT cl$
190 GOSUB 710'Seekrec
200 PRINT cl$
210 GOSUB 770'Seekset
220 PRINT cl$
230 GOSUB 1100'Put
240 PRINT cl$
250 GOSUB 1180'Delkey
260 CLOSE #1
270 OPTION FILES "A:":END
280 rc=CONSOLIDATE(#1)
290 RETURN
300 'Anzeige der aktuellen Position
310 reihe=FETCHRANK(#1)
320 key$=FETCHKEY$(#1)
330 IF reihe=1 THEN key$=STR$(CVIK(FETCHKEY$(#1)))
340 PRINT "Die aktuelle Position ist auf Reihe"reihe", Schlüssel "key$" und
Satznummer"FETCHREC(#1)"."
350 IF ergebnis<103 THEN GET #1:PRINT xvorname$" "xname$" "CVI(xplz$)" "xort$"
"xkunde$:satznr=FETCHREC(#1):GET #1,satznr:PRINT xvorname$" "xname$" "CVI(xplz$)" "xort$"
"xkunde$" "satznr
360 'Ausgabe des Antwortcodes der Jetsamfunktion
370 PRINT"Antwortcode ist"ergebnis;
380 PRINT"Weiter mit beliebiger Taste"
390 t$="":WHILE t$="":t$=INKEY$:WEND
400 RETURN
410 PRINT"Beispiele zu SEEKKEY
420 ergebnis=SEEKKEY(#1,0,1,MKIK$(2300)):GOSUB 310
430 ergebnis=SEEKKEY(#1,0,2,"M"):GOSUB 310
440 ergebnis=SEEKKEY(#1,0,2,"L"):GOSUB 310
450 RETURN
460 PRINT"Beispiele zu SEEKNEXT
470 ergebnis=SEEKKEY(#1,0,2,"K"):GOSUB 310
480 ergebnis=SEEKNEXT(#1,0):GOSUB 310
490 ergebnis=SEEKNEXT(#1,0):GOSUB 310
500 ergebnis=SEEKNEXT(#1,0):GOSUB 310
510 ergebnis=SEEKNEXT(#1,0):GOSUB 310
520 ergebnis=SEEKNEXT(#1,0):GOSUB 310
530 ergebnis=SEEKKEY(#1,0,0,"XaverUnsin"):GOSUB 310:ergebnis=SEEKNEXT(#1,0):GOSUB 310
540 ergebnis=SEEKNEXT(#1,0,0,"WalteElekt",6):GOSUB 310
550 ergebnis=SEEKNEXT(#1,0,2,"M",5):GOSUB 310
560 RETURN
570 PRINT"Beispiele zu SEEKPREV
580 ergebnis=SEEKKEY(#1,0,2,"L"):GOSUB 310
590 ergebnis=SEEKPREV(#1,0):GOSUB 310
600 ergebnis=SEEKPREV(#1,0):GOSUB 310
610 ergebnis=SEEKPREV(#1,0):GOSUB 310
620 ergebnis=SEEKPREV(#1,0):GOSUB 310
```



```
630 ergebnis=SEEKPREV(#1,0,0,"WalteElekt",6):GOSUB 310
640 ergebnis=SEEKPREV(#1,0,2,"M",5):GOSUB 310
650 ergebnis=SEEKKEY(#1,0,0,"ElektMaier"):GOSUB 310:ergebnis=SEEKPREV(#1,0):GOSUB 310
660 RETURN
670 PRINT"Beispiele zu SEEKRANK
680 ergebnis=SEEKRANK(#1,0,1):GOSUB 310
690 ergebnis=SEEKRANK(#1,0,4):GOSUB 310
700 RETURN
710 PRINT"Beispiele zu SEEKREC
720 reihe=0:schluessel$="Mayr Großh":satznummer=3
730 ergebnis=SEEKREC(#1,0,reihe,schluessel$,satznummer):GOSUB 310
740 ergebnis=SEEKREC(#1,0,0,"XaverFranz",6):GOSUB 310
750 ergebnis=SEEKREC(#1,0,2,"M",6):GOSUB 310
760 RETURN
770 PRINT"Beispiele zu SEEKSET
780 ergebnis=SEEKKEY(#1,0,1,MKIK$(2390)):GOSUB 310
790 ergebnis=SEEKSET(#1,0):GOSUB 310
800 ergebnis=SEEKSET(#1,0):GOSUB 310
810 ergebnis=SEEKSET(#1,0):GOSUB 310
820 ergebnis=SEEKSET(#1,0):GOSUB 310
830 ergebnis=SEEKSET(#1,0):GOSUB 310
840 ergebnis=SEEKSET(#1,0):GOSUB 310
850 RETURN
860 PRINT"Beispiele für ADDREC und ADDKEY
870 LSET xname$="Installation":LSET xvorname$="Krause":xplz=2390:LSET
xort$="Flensburg":LSET xkunde$="K"
880 GOSUB 980
890 LSET xname$="Mayr":LSET xvorname$="Großhandel":xplz=8000:LSET xort$="München 50":LSET
xkunde$="L"
900 GOSUB 980
910 LSET xname$="Elektro":LSET xvorname$="Maier":xplz=2390:LSET xort$="Flensburg":LSET
xkunde$="K"
920 GOSUB 980
930 LSET xname$="Xaver":LSET xvorname$="Unsinn":xplz=8000:LSET xort$="München 200":LSET
xkunde$="G"
940 GOSUB 980
950 LSET xname$="Ludwig":LSET xvorname$="Elektronik":xplz=8000:LSET xort$="München
150":LSET xkunde$="L"
960 GOSUB 980
970 RETURN
980 key$=LEFT$(xname$,5):IF LEN(xname$)<5 THEN key$=key$+SPACE$(5-LEN(xname$))
990 key$=key$+LEFT$(xvorname$,5):IF LEN(xvorname$)<5 THEN
key$=key$+SPACE$(5-LEN(xvorname$))
1000 LSET xplz$=MKI$(xplz)
1010 ergebnis=ADDREC(#1,2,0,key$):GOSUB 310
1020 GOSUB 280:IF ergebnis=116 THEN 1090
1030 key$=MKIK$(xplz)
1040 satznr=FETCHREC(#1):ergebnis=ADDKEY(#1,2,1,key$,satznr)
1050 GOSUB 310:GOSUB 280
1060 key$=xkunde$
1070 satznr=FETCHREC(#1):ergebnis=ADDKEY(#1,2,2,key$,satznr)
1080 GOSUB 310:GOSUB 280
1090 RETURN
1100 PRINT"Beispiel für PUT
1110 PRINT"Lesen der Daten zum Geschäftsfreund":ergebnis=SEEKKEY(#1,2,2,"G"):GOSUB 310:GET
#1
1120 PRINT"Verändern der Ortsangabe von "xort$" auf ";:LSET xort$="München 340":PRINT
xort$
1130 PRINT"Zurückschreiben des Satzes":PUT #1
1140 PRINT"Lesen der Daten zu Xaver Unsinn":ergebnis=SEEKKEY(#1,1,0,"XaverUnsin"):GET #1
1150 PRINT xvorname$" "xname$" "CVI(xplz$)" "xort$" "xkunde$
1160 PRINT"Weiter mit beliebiger Taste":t$="":WHILE t$="":t$=INKEY$:WEND
1170 RETURN
1180 PRINT"Beispiel zu DELKEY
1190 PRINT"Lesen des ersten Lieferanten":ergebnis=SEEKKEY(#1,2,2,"L"):GOSUB 310
1200 PRINT"Löschen des Schlüssels - hier der Kennung
Lieferant":satznr=FETCHREC(#1):ergebnis=DELKEY(#1,2,2,"L",satznr):GOSUB 280:GOSUB 310
1210 PRINT"Lesen der Postleitzahl zu diesem Datensatz als
Schlüssel":ergebnis=SEEKKEY(#1,2,1,MKIK$(8000)):GOSUB 310
1220 PRINT"Löschen des Schlüssels - hier der
Postleizahl":ergebnis=DELKEY(#1,2,1,MKIK$(8000),satznr):GOSUB 280:GOSUB 310
1230 PRINT"Lesen des kombinierten Schlüssels Name und
```

```
Vorname":ergebnis=SEEKKEY(#1,2,0,"Mayr Großh"):GOSUB 310
1240 PRINT"Löschen des Schlüssels - hier Name und Vorname":ergebnis=DELKEY(#1,2,0,"Mayr
Großh",satznr):GOSUB 280:GOSUB 310
1250 PRINT"Erst jetzt sind alle Informationen zu Großhandel Mayr gelöscht!!"
1260 ergebnis=SEEKKEY(#1,2,0,"Mayr Großh"):GOSUB 310
1270 ergebnis=SEEKKEY(#1,2,1,MKIK$(8000)):GOSUB 310
1280 ergebnis=SEEKKEY(#1,2,2,"L"):GOSUB 310
1290 RETURN
```



# Verarbeitung von Dateien mit wahlfreiem Zugriff

Bedeutsame Befehle:

- CLOSE
- FIELD
- GET
- INPUT #
- INPUT$
- LOC
- PRINT #
- PUT
- OPEN
- WRITE #

Das Programm RANDOM.BAS am Ende dieses Kapitels ist ein kleines Beispielprogramm, das die Wirkung der Befehle zeigt.

Wahlfreier Zugriff bedeutet, daß die Datei gleichzeitig gelesen und beschrieben werden kann.

**OPEN "R",#1,datei$,168** eröffnet eine Datei für wahlfreien Zugriff mit einer Satzlänge von 168 Stellen. Da hier die Standardeinstellung von 128 Stellen Satzlänge überschritten wird, muß die maximale Satzlänge vorher durch ein **CLEAR,,,,168** oder **MEMORY,,,168** heraufgesetzt werden.

Ist die Datei noch nicht vorhanden, wird sie automatisch angelegt.

```
OPEN "R",Datei-Symbol,Name der Random-Datei[,maximale Länge des Datensatzes]
```

**Beispiel**

```
10 OPEN "R",#1,"test.rdm"
```

Hier wird unter dem Datei-Symbol **#1** die Datei TEST.RDM eingerichtet und für die Verarbeitung im Programm eröffnet.

**GET #1** liest einen Satz der für wahlfreien Zugriff geöffneten Datei in den Satzpuffer.

Wurde der Satzpuffer durch ein **FIELD**-Kommando in verschiedene Datenfelder unterteilt, kann direkt auf die Datenfelder zugegriffen werden.

Wurde kein **FIELD**-Kommando benutzt, kann der Satzpuffer durch **INPUT #** oder **INPUT$** gelesen und mit **PRINT #** oder **WRITE #** geschrieben werden.

```
GET Datei-Symbol der Random-Datei[,Satznummer des zu lesenden Satzes]
```

**Beispiel**

```
65 GET #1,2
```

Hier wird aus der Random-Datei der Satz mit der Satznummer 2 in den Satzpuffer gelesen.

**FIELD** ist im Kapitel über Jetsam im einzelnen beschrieben. **INPUT #, INPUT$, PRINT #** und **WRITE #** sind im BASIC-Teil dieses Buches beschrieben.

**PUT #1** schreibt den aktuellen Inhalt des Satzpuffers in die Random-Datei. Der Inhalt des Satzpuffers wird mit der aktuellen Satznummer in der Random-Datei abgelegt.

**Put #1,35** schreibt den Inhalt des Satzpuffers in Satz Nr. 35 der für wahlfreien Zugriff geöffneten Datei.

Der Satzpuffer kann nur durch die Kommandos **LSET, RSET, MID$, PRINT #** und **WRITE #** gefüllt werden. **LSET, RSET** und **MID$** beziehen sich dabei auf die im **FIELD**-Kommando beschriebenen Datenfelder des Satzpuffers. Ist kein **FIELD**-Kommando vorgesehen, kann der Satzpuffer über **PRINT #** und **WRITE #** beschrieben werden.



**WRITE #** füllt jeweils den Puffer bis zum Ende des Puffers mit Leerstellen und Wagenrücklauf am Ende auf, so daß nach jedem **WRITE #** ein **PUT** kommen muß; sonst wird die Fehlermeldung **record overflow** ausgegeben.

**PRINT** füllt den Puffer nur soweit, wie im **PRINT**-Befehl angegeben: d.h., daß mehrere **PRINT**-Befehle den Satzpuffer auffüllen können, bis er voll ist. Dann erst braucht der Pufferinhalt mit **PUT** auf die Datei geschrieben werden. Nach jedem **PRINT** wird automatisch Wagenrücklauf und Zeilenvorschub eingefügt, es sei denn, der **PRINT**-Befehl wird mit Komma oder Semikolon abgeschlossen. Das Schreiben des Puffers mit dem **PRINT**-Befehl ist genauso zu betrachten, als ob man eine kleine einfache sequentielle Datei füllt, die in ihrer Größe begrenzt ist (hier die Größe des Satzpuffers = Satzlänge)

Zu den Befehlen **LSET, RSET** und **MID$** siehe im Kapitel über Jetsam.

**Beispiel**

```
20 FIELD #1,100 AS text$
25 FOR x=1 to 30:LSET text$=" ":PUT #1:NEXT
30 LSET text$="Dies ist Satz Nummer 1":PUT #1,1
40 LSET text$="Dies ist Satz Nummer 2":PUT #1
50 LSET text$="Dies ist Satz Nummer 3":PUT #1
60 LSET text$="Dies ist Satz Nummer 4":PUT #1
```

Hier wird zunächst in Zeile 20 der Satzpuffer durch das **FIELD** definiert. In Zeile 25 wird die Random-Datei vorbereitet, so daß sie bis zu 30 Sätze aufnehmen kann. Diese Vorbereitung ist wichtig, um sicherzustellen, daß auf allen Sätzen der richtige Inhalt geschrieben wird - wird die Random-Datei gezielt auf den Sätzen Nummer 1, 5, 15, und 20 beschrieben, steht in den anderen Sätzen zufälliger Inhalt, wenn die Datei nicht entsprechend vorbereitet wurde. Lassen Sie das Programm RANDOM.BAS einmal ohne Zeile 25 laufen und sehen Sie sich den Unterschied zu den Sätzen Nr. 5 und 6 an.

In Zeile 30 wird im **PUT** gezielt auf Satz Nummer 1 geschrieben. Ohne Angabe der Satznummer würde Satz Nummer 31 beschrieben werden, denn durch die Schleife in Zeile 25 wird die aktuelle Satznummer automatisch jeweils um 1 auf 30 hochgezählt.

In den Zeilen 40 - 60 kann dann wieder die Satznummer weggelassen werden, denn in Zeile 30 wird durch Angabe der Satznummer die 1 zur aktuellen Satznummer gemacht.

**satznr=LOC(1)** gibt im Feld satznr die Satznummer des zuletzt mit **GET** gelesenen oder mit **PUT** geschriebenen Satzes an. Dies kann hilfreich sein, wenn man z.B. die Nummer des aktuellen Satzes zwischenspeichern möchte, weitere Sätze aus der Datei einliest und später diesen Satz, verändert durch ein **PUT #1,satznr**, zurückschreiben möchte.

```
LOC(Dateinummern-Ausdruck)
```

**Beispiel**

```
74 PRINT"Der Satz mit der Satznummer"LOC(1)" wurde verarbeitet
83 GET #1:PRINT"Lesen ohne Satznummer - aktuelle Nummer ist"LOC(1)
90 PRINT "Der Satz mit der Satznummer"LOC(1)" ist der zuletzt verarbeitete
Satz"
```

**LOC** wird hier jeweils benutzt, um die aktuelle Satznummer des gerade verarbeiteten Datensatzes auszugeben.

**LOC** kann auch bei Dateien benutzt werden, die für sequentielle Ein- (**OPEN "I"**) oder Ausgabe (**OPEN "O"**) eröffnet sind. **saetze=LOC(2)** ergibt dann die Anzahl der in die Datei #2 geschriebenen oder bereits gelesenen 128 Bytes langen Datenblöcke.

CP/M unterteilt jede Datei in Datenblöcke zu jeweils 128 Bytes, egal wie lang die Datensätze oder die Datenelemente in der Datei sind. Ein String von 255 Bytes Länge belegt in einer Datei 2 solcher Dateiblöcke. Das 256. Byte ist ein



systeminternes Längenfeld des Strings, damit CP/M bzw. BASIC erkennen kann, wie lang der String ist. Dieses systeminterne Feld wird durch die Funktion **laenge=LEN(string$)** abgefragt und im Feld **laenge** zur Verfügung gestellt.

Hier das vollständig Listing des Beispielprogramms RANDOM.BAS

```
10 OPTION FILES "M:":OPEN "R",#1,"test.rdm"
20 FIELD #1,100 AS text$
25 FOR x=1 TO 30:LSET text$=" ":PUT #1:NEXT
30 LSET text$="Dies ist Satz Nummer 1":PUT #1,1
40 LSET text$="Dies ist Satz Nummer 2":PUT #1
50 LSET text$="Dies ist Satz Nummer 3":PUT #1
60 LSET text$="Dies ist Satz Nummer 4":PUT #1
65 GET #1,2:PRINT text$
70 LSET text$="Dies ist Satz Nummer 5":PUT #1
74 PRINT "Der Satz mit der Satznummer"LOC(1)" wurde verarbeitet"
75 LSET text$="Dies ist Satz Nummer 6":PUT #1
80 LSET text$="Dies ist Satz Nummer 20":PUT #1,20
83 GET #1:PRINT"Lesen ohne Satznummer - aktuelle Nummer ist"LOC(1):PRINT text$
84 GET #1,3:PRINT text$
90 PRINT "Der Satz mit der Satznummer"LOC(1)" ist der zuletzt verarbeitete Satz"
100 CLOSE #1
110 TYPE test.rdm
120 OPTION FILES "A:":END
```

In Zeile 65 wird Satz Nummer 2 gelesen. Aktuelle Satznummer ist somit 2. In Zeile 70 wird im **PUT** keine Satznummer mitgegeben, so daß der Text "Dies ist Satz Nummer 5" auf Satz Nummer 3 geschrieben wird. Bei einem **PUT** wird zunächst die Satznummer erhöht und dann der Satz mit der erhöhten Satznummer in die Datei geschrieben, wenn keine Satznummer mitgegeben wurde.
In Zeile 75 wird dann der Text "Dies ist Satz Nummer 6" auf die Satznummer 4 geschrieben. Die Sätze 3 und 4 wurden bereits in den Zeilen 50 und 60 beschrieben und werden in den Zeilen 70 und 75 neu eingetragen.

Mit dem **TYPE** in Zeile 110 wird der Inhalt der Datei TEST.RDM angezeigt.

# G e n e r a t o r  f ü r
# J E T S A M - V e r a r b e i t u n g

Um die Dateiverarbeitung mit Jetsam zu erleichtern, wurde der Programmgenerator GENERJET.BAS geschaffen, den Sie auf der zum Buch erhältlichen Diskette auf Seite B: finden. Mit diesem Generator wird ein lauffähiges BASIC-Programm erstellt, mit dem eine Jetsam-Datei verarbeitet wird, und die Daten auf Bildschirm und Drucker ausgegeben werden können.

**Erläuterungen zum Generator**

Der Generator erklärt sich weitgehend von selbst.

Der Generator wird von BASIC gestartet:

Laden Sie zunächst BASIC von CP/M, durch Eingabe von:

```
A>BASIC[ENTER]
```

[ENTER] bedeutet, daß Sie die [RETURN]- bzw. [ENTER]-Taste betätigen.

Schieben Sie jetzt die dem Buch beiligende Diskette mit Seite B in das Laufwerk A: und starten Sie GENREJET durch Eingabe von:

```
OK
run "generjet[ENTER]
```

Sie werden von Generjet begrüßt und nach Betätigen einer beliebigen Taste werden Sie aufgefordert, die notwendigen Eingaben zur Generierung Ihres Jetsam-Programmes zu machen. Im folgenden finden Sie die Erläuterungen zu den Eingaben. Die **fett** angegebenen Texte erscheinen bei Ihnen auf dem Bildschirm.



Es sind folgende Eingaben vorzunehmen - die Eingaben sind in der Regel mit [ENTER] abzuschließen:

<u>Allgemeine Angaben:</u>

**Name des Programms:**
Der Name darf bis zu 8 Stellen lang sein, längere Bezeichnungen würden zu Fehlern führen, da der Programmname als Name für die zu generierende Programmdatei und die Submitdatei zum Start des generierten Programms benutzt wird.
Ggf. werden Sie nach einem Pieps erneut aufgefordert, einen Namen einzugeben.

**Ihr Name:**
Ihr Name wird als Kommentarzeile zu Beginn des generierten Programms aufgenommen und für das Begrüßungsbild des generierten Programms ausgegeben.

**Das heutige Datum:**
Um im Programm genau festzuhalten, wann das generierte Programm erstellt wurde, geben Sie hier bitte das Tagesdatum ein. Die Stellenzahl ist nicht begrenzt. Sie sind daher frei, in welcher Form Sie das Datum eingeben (z.B. 24.12.1988 oder 24. Dezember 1988 etc.)

**Kurzbeschreibung/Aufgabe des Programms:**
In einer Länge von 244 Buchstaben (einschließlich Leerstellen) kann hier etwas ausführlicher die Aufgabe des Programms beschrieben werden. Dieser Text wird als Kommentarzeile zu Beginn des generierten Programms eingefügt und auf dem Begrüßungsbild beim generierten Programm mit ausgegeben.

**Name der Jetsamdatei:**
Der Name darf bis zu 8 Stellen lang sein (vgl. Anmerkungen zu **Name des Programms**). Dieser Name wird

ergänzt um die Erweiterungen ".DAT" für die Datendatei und ".IND" für die Schlüsseldatei.

**Speicherung der Jetsam-Datei und des Programms auf Laufwerk A: oder B:**

Es kann das Laufwerk eingegeben werden, auf dem die Jetsamdatei (Daten- und Schlüsseldatei) und die Programmdatei Ihres generierten Programms gespeichert werden sollen. Es wird nicht geprüft, ob ein Laufwerk "B:" vorhanden ist, falls "B" eingegeben wird. Ggf. tritt ein Fehler bei der Speicherung des generierten Programms auf.
Schieben Sie eine formatierte Diskette in das entsprechende Laufwerk und geben Sie "A" oder "B" ein. Es wird die Datei **Programmname.SUB** angelegt und auf Diskette gesichert.

<u>Angaben zu den Daten, die mit dem zu generierenden Programm verarbeitet werden sollen:</u>

**Geben Sie die Feldnamen ein - die Schlüsselfelder bitte als erstes eingeben:**

Es werden jetzt der Reihe nach die Feldnamen und weitere Angaben zu den Datenfeldern abgefragt. Danach wird die Endeverarbeitung zur Generierung eingeleitet (siehe unten).
Die Schlüsselfelder sind als erstes einzugeben. Beim ersten eingegebenen Feldnamen wird unterstellt, daß es ein Schlüsselfeld ist, bei dem nur eindeutige Schlüsselwerte zugelassen sind. Sind alle Schüsselfelder eingegeben worden, können die übrigen Datenfelder erstellt werden.

**x. Feldname**

Wird nur [ENTER] eingegeben, wird die Abfrage zu den Feldnamen beendet.
**x.** ist eine laufende Nummer der eingegebenen Felder.
Der erste eingegebene Feldname sollte ein



Schlüsselfeld mit eindeutigem Schlüssel sein, da dieser Schlüssel zum Einfügen des neuen Satzes in die Datendatei genommen wird. Damit wird erreicht, daß in der ersten Schlüsselreihe ein Schlüsselwert gespeichert wird, über den man jeden einzelnen Satz eindeutig identifizieren kann.
Für den ersten eingegebenen Feldnamen wird vom Generator daher unterstellt, das es ein Schlüsselfeld ist, bei dem gleiche Schlüsselwerte nicht zugelassen sind (eindeutige Schlüsselwerte).

**Zeile und Spalte auf dem Bildschirm für feldname - zeile,spalte**

Hier kann angegeben werden, an welcher Bildschirmposition zur Eingabe von Werten aufgefordert werden soll. Es wird damit eine Eingabemaske definiert.
Es ist die Bildschirmzeile anzugeben: Zeile 6 bis 27
Es ist die Spalte der Bildschirmzeile anzugeben: Spalte 0 bis 88
Beispiel: Eingabe: 6,4 - Es wird auf Zeile 6 ab Spalte 4 zur Eingabe aufgefordert werden.

**Ist feldname ein Zahlenfeld ? (J/N)**

Hier wird angegeben, ob das Feld Text oder nur Zahlen aufnehmen soll. Wird "J" eingegeben, wird bei der Generierung als Zahlenfeld fortgefahren, bei Eingabe eines "N" wird das Feld als Stringfeld generiert, das Texte aufnehmen kann.

<u>Generierung als Zahlenfeld:</u>

**Wertebereich der Zahl:**

**1 - 0 bis 255**
**2 - -32768 bis +32767**
**3 - 0 bis 65535**
**4 - einfach genaue Zahl**
**5 - doppelt genaue Zahl**

**Auswahl 1 bis 5 eingeben**

Hierdurch kann der Zahlentyp festgelegt werden, außerdem welche Minimum- und Maximumwerte grundsätzlich vorgesehen (Auswahl 1 bis 3) oder ob einfache oder doppelt genaue Zahlen (Auswahl 4 oder 5) verarbeitet werden sollen.

Abhängig von der Auswahl werden entsprechende Routinen vorgesehen, die die Zahlen in der Jetsam-Datei in komprimierter Form speichern, und somit Platz gespart wird. Ebenso werden entsprechende Routinen zur Aufbereitung der komprimierten Werte zur Anzeige und zum Druck vorgesehen.

Geben Sie eine der Zahlen 1 bis 5 zur Festlegung des Zahlentyps bzw. des Wertebereiches ein.

**Möchten Sie Minimum und Maximum von feldname festlegen? (J/N)**

Sollen keine Grenzwerte festgelegt werden, bleibt es bei der grundsätzlichen Festlegung (Auswahl 1 bis 3 der vorherigen Frage) bzw. es werden keine Grenzwerte vorgesehen (Auswahl 4 bzw. 5 der vorherigen Frage).

Geben Sie "J" oder "N" ein.

**Minimal- und Maximalwert von feldname**

Wurde gesagt, daß Grenzwerte festgelegt werden sollen (Auswahl "J" bei der vorherigen Frage), können diese jetzt eingegeben werden.

Im generierten Programm wird sichergestellt, daß diese Grenzwerte berücksichtigt werden.

Beispiel: Eingabe: 50,75 - Zu diesem Feld können nur Werte von 50 bis 75 eingegeben werden.

Wurde versehentlich der Maximalwert zuerst eingegeben (Eingabe: 75,50) wird dies berücksichtigt und die Zahlen vom Generator richtig eingesetzt, und zwar mit 50 als Minimal- und 75 als Maximalwert.

Werden bei den Zahlentypen 1 bis 3 durch die Minimal- und Maximalwerte die grundsätzlichen Wertebereiche unter- bzw. überschritten, wird zur Eingabe des gewünschten Zahlentyps zurückgesprungen.



Generierung als Stringfeld:

**Maximale Datenlänge von feldname**

Hier wird eingegeben, wie viele Stellen der Text maximal lang sein darf. Dies wird für eine Prüfung der Länge des eingegebenen Textes mit herangezogen. Außerdem wird der Text nur bis zu dieser Länge in der Jetsam-Datei abgespeichert.

Geben Sie die maximale Stellenzahl ein.

Gemeinsame Weiterverarbeitung für String- und Zahlenfelder

**Soll feldname als Schlüsselbegriff verwendet werden ? (J/N)**

Hiermit wird angegeben, ob dieses Feld als Schlüssel für die späteren Datenzugriffe - wie Anzeigen, Löschen, Verändern und Drucken - verwendet werden soll. Beim ersten eingegeben Feldnamen wird vom Generator stets in die Routinen für die Generierung als Schlüsselfeld verzweigt.

Wird diese Frage mit Nein beantwortet, wird für alle weiteren eingegebenen Feldnamen unterstellt, daß sie **nicht** als Schlüssel berücksichtigt werden sollen und diese Frage wird nicht mehr angezeigt.

Geben Sie "J" oder "N" ein.

Generierung des Feldes als Schlüsselbegriff:

**Schlüsselgenerierung**

Hiermit wird angezeigt, daß man sich in dem Teil des Generators befindet, durch den der Feldname als Schlüsselbegriff vorbereitet wird. Beim ersten Feldnamen wird stets in diesen Teil verzweigt.

**Dieser Zahlentyp kann nicht als Schlüssel verwendet werden**

Wurde bei einer einfach oder doppelt genauen Zahl gesagt, daß sie als Schlüssel benutzt werden soll, wird die Schlüsselgenerierung zurückgewiesen. Dieses Feld wird als einfaches Datenfeld berücksichtigt. Der nächste Feldname kann wieder als Schlüssel berücksichtigt werden.

**Es wurden bereits alle 8 Schlüsselreihen benutzt. In welcher Reihe soll dieser Schlüssel zusätzlich aufgenommen werden (Reihe 1 bis 7; nicht aufnehmen = 9)**

Jetsam verarbeitet normalerweise nur maximal 8 verschiedene Schlüssel in sogennannten Schlüsselreihen. Es können in einer Schlüsselreihe jedoch mehr als ein Schlüssel abgelegt werden. Wählen Sie eine beliebige Schlüsselreihe aus. Wird eine 9 eingegeben, wird dieser Feldname überhaupt nicht berücksichtigt und kann ggf. erneut eingegeben werden.
Geben Sie eine der Zahlen 1 bis 7 oder 9 ein.

**Als einstellige Kennung für feldname wird buchstabe vorgeschlagen.**
**Nehmen Sie den Vorschlag an? (J/N)**

Mit dieser Kennung wird für den Teil zum Anzeigen bzw. Drucken der gespeicherten Werte im generierten Programm die Wahl erleichtert und vereinfacht, nach welchem Schlüssel Daten gesucht und angezeigt bzw. gedruckt werden sollen.
Bitte entscheiden Sie, ob Sie den vorgeschlagenen Buchstaben für den Feldnamen als einstellige Kennung möchten oder nicht, indem Sie "J" oder "N" eingeben.
Standardmäßig wird das erste Zeichen des Feldnamens vorgeschlagen. Wird dieser bereits für ein anderes Schlüsselfeld als Kennung benutzt, wird ein anderes Zeichen aus dem Feldnamen als Kennung vorgeschlagen.



**Bitte eine einstellige Kennung für feldname eingeben**

Geben Sie eine Zahl oder einen Buchstaben ein. Groß- oder Kleinschreibung wird bei Buchstaben nicht berücksichtigt.
Mit dieser Kennung wird für den Teil zum Anzeigen bzw. Drucken der gespeicherten Werte im generierten Programm die Wahl erleichtert und vereinfacht, nach welchem Schlüssel Daten gesucht und angezeigt bzw. gedruckt werden sollen.
Diese Frage erscheint nur, wenn automatisch keine einstellige Kennung ermittelt werden konnte oder Sie bei der vorherigen Frage angegeben haben, daß Sie eine andere als die vorgeschlagene Kennung haben möchten.

**buchstabe wird als Kennung bereits benutzt**

Die eingegebene einstellige Kennung wird bereits benutzt. Bitte geben Sie nach Aufforderung eine andere einstellige Kennung ein.

**Länge des Schlüsselbegriffs von feldname**

Ist das Feld ein Stringfeld, kann hier festgelegt werden, wie lang der Schlüssel aus dem eingegebenen Text sein soll, beginnend mit der ersten Stelle des Textes.
Beispiel:

Eingabe zum Stringfeld - 30 Stellen lang

Eingabe zur Länge des Schlüssels - 9 Stellen lang

Damit werden nur die ersten 9 Stellen des Stringfeldes als Schlüsselwert genommen.
Die Länge des Schlüsselbegriffs darf die Länge des Stringfeldes nicht überschreiten. Ggf. wird durch ein Pieps auf eine Fehleintragung aufmerksam gemacht mit der Möglichkeit zur Korrektur der Länge des Schlüsselbegriffs.

**Kann der Schlüssel für mehrere Datensätze mit demselben Wert vorkommen? (J/N)**

Hier wird gefragt, ob zu diesem Schlüssel nur eindeutige Schlüsselwerte vorkommen dürfen (Auswahl "N"), oder ob ein Schlüsselwert zu verschiedenen Datensätzen denselben Schlüsselwert haben darf (mehrdeutige Schlüssel) (Auswahl "J").
Beim ersten Feldnamen wird unterstellt, daß nur eindeutige Schlüssel zugelassen sind. Diese Frage wird beim ersten Feldnamen daher nicht gestellt.
Geben Sie "J" oder "N" ein.

Endeverarbeitung zur Generierung

**Möchten Sie das generierte Programm sofort starten ? (J/N)**

Hier wird angegeben, ob Sie sofort beginnen möchten, mit Ihrem generierten Programm zu arbeiten, oder ob das Programm nur erstellt und gespeichert werden soll. Auch bei einem Sofortstart wird das generierte Programm zunächst abgespeichert, dann jedoch auch gleich gestartet.
Geben Sie "J" oder "N" ein.

**Einen Moment bitte, das generierte Programm wird gesichert!**

Diese Meldung teilt Ihnen mit, daß die Generierung abgeschlossen ist, und das Programm jetzt auf Diskette gespeichert wird. Außerdem wird die Jetsam-Datei mit der Schlüsseldatei **Dateiname.IND** und der Datendatei **Dateiname.DAT** auf Diskette angelegt. Bereits bestehende Dateinamen mit gleichem Namen werden vor dem Anlegen gelöscht!!

**Das Programm steht auf Laufwerk laufwerk mit dem Namen programmname zur Verfügung!**

Der Generator hat seine Arbeit getan. Das Programm steht zum einen im BASIC-Speicher bereit, so daß es mit **RUN** gestartet werden kann. Außerdem wurde das generierte Programm unter dem Namen **Programmname.BAS**



auf Diskette gespeichert.
Wird das generierte Programm erstmals von Diskette gestartet, müssen die Zeilen 1 bis 3 noch gelöscht werden, weil darin noch Anweisungen vom Generator enthalten sind:
Von BASIC her sind einzugeben:

```
OK
LOAD "laufwerk:programmname[ENTER]

OK
delete 1 - 3[ENTER]
SAVE "laufwerk:programmname[ENTER]

OK
```

Damit sind die Zeilen 1 bis 3 dauerhaft aus dem generierten Programm entfernt. Sonst würde jedesmal zu Beginn des Programms ein SAVE ausgeführt werden, der sehr viel Zeit kostet.

Soll der Generator neu gestartet werden, so muß RUN GENERJET unter BASIC eingegeben werden, damit BASIC den Generator neu von Diskette starten kann, denn mit Beendigung der Generierung ist der Generator aus dem Programmspeicher von BASIC entfernt worden.

**Erläuterungen zum generierten Programm**

Start des generierten Programms:

Auf der Arbeitsdiskette müssen folgende Dateien vorhanden sein:
PIP.COM - Dateikopierprogramm
BASIC.COM - BASIC
SUBMIT.COM - Programm zur Abarbeitung von *.SUB-Dateien
Programmname.SUB - Submitdatei zum Start des generierten Programms unter BASIC
Programmname.BAS - Das generierte Jetsamprogramm
Dateiname.DAT - Die verwendete Datendatei
Dateiname.IND - Die verwendete Schlüsseldatei

Die Dateien **Programmname.BAS**, **Programmname.SUB**, **Dateiname.DAT** und **Dateiname.IND** werden vom Generator angelegt. Kopieren Sie die drei COM-Dateien von Ihrer CP/M-Systemdiskette auf die Arbeitsdiskette, auf der die generierten Dateien stehen.

Das Programm wird von CP/M her gestartet:

A>programmname[ENTER]

Die Daten- und Schlüsseldatei werden zur schnelleren Bearbeitung in das Laufwerk M: kopiert. Dann wird BASIC mit dem generierten Jetsam-Programm gestartet. Nach Beendigung des Jetsamprogramms werden die Dateien zurück auf die Diskette kopiert und somit die Änderungen in den Dateien auf Diskette gesichert.

Die Konfiguration der Jetsamdateien und Kommandodateien kann nach Bedarf geändert werden. Es sind die Zeilen 340 - 360 von GENERJET anzupassen. Es wird von GENERJET folgende Konfiguration unterstellt:

*.COM-Dateien auf Laufwerk A:
programmname.* und Dateiname.* auf dem bei der Generierung eingegebenen Laufwerk A: oder B:.



Nach einer Generierung kann die Datei **Programmname.SUB** von jedem Texteditor, z.B. RPED, geändert werden.

Nach einer kurzen Begrüßung wird das Arbeitsmenü des Programms aufgebaut, von dem aus die Auswahlpunkte

E - Eingabe
V - Verändern
L - Löschen
A - Anzeigen
D - Drucken
X - Programmende

durch Eingabe des groß geschriebenen Buchstabens angewählt werden können.

Das "X" steht auch bei sämtlichen anderen Programmteilen zur Verfügung, um den jeweils angewählten Menüpunkt zu beenden und in das Arbeitsmenü zurückzukommen, um einen anderen Menüpunkt auszuwählen oder das Programm durch nochmalige Eingabe eines "X" ganz zu beenden.

Es wird jeweils rechts oben angezeigt, welcher Programmteil aktiv ist.

Zu den einzelnen Programmteilen:

<u>Eingabe</u>

Nacheinander geben Sie zu jedem Feld, daß Sie bei der Generierung angegeben haben, nach Aufforderung einen Wert ein. Jede Eingabe wird mit [ENTER] abgeschlossen.

**Weitere Eingaben?**
Weitere Daten können eingegeben werden, wenn ein "W", "E" oder Leertaste eingegeben wird. Die Eingabe eines "X" beendet den Eingabeteil.

**Wert zum ersten Schlüssel bereits gespeichert**
oder
**Satz mit diesem Schlüssel bereits vorhanden**

Sie haben für einen Schlüssel bei der Generierung angegeben, daß nur eindeutige Schlüsselwerte zugelassen sind. Es wurde versucht, die eingegebenen Daten in die Jetsamdatei zu schreiben, jedoch ist der Schlüssel bereits vergeben.
Es wurden keine Daten eingefügt. Durch Eingabe von "W", "E" oder Leertaste können die Daten mit veränderten Werten erneut eingegeben werden. Die Eingabe eines "X" beendet den Eingabeteil.

Verändern

**Bitte Schlüsselwert eingeben**

Zunächst wird nach dem Schlüsselwert gefragt, um den Datensatz einzulesen, der verändert werden soll.
Soll der Datensatz verändert werden, der über den Schlüssel NAME mit dem Wert "Hugo" gefunden werden kann, wird hier aufgefordert, den Namen einzugeben. Geben Sie dann bitte "Hugo" ein.

**Eingabe der neuen Werte**

Ähnlich wie im Eingabeteil werden Sie jetzt Feld für Feld aufgefordert, die **neuen** Werte einzugeben. Dabei wird jeweils der alte Feldinhalt angezeigt.
Soll der alte Feldinhalt übernommen, d.h., nicht verändert werden, braucht der alte Inhalt nicht erneut eingegeben werden; es genügt, [ENTER] zu betätigen. Die alten Daten werden dann in den neuen Datensatz übernommen.
Soll ein Wert verändert werden, so ist der neue Wert vollständig einzugeben. Es genügt nicht, nur einzelne Zeichen des alten Wertes abzuändern. Das würde zu einer fehlerhaften Speicherung der Daten führen.
Der alte Datensatz bleibt nicht bestehen, er wird



gelöscht. Der veränderte Datensatz wird neu in die Jetsam-Datei eingefügt.

**Neue Werte erneut eingeben? (J/N)**

Beim Verändern der Werte wurde ein Schlüsselwert verändert. Bei der Schlüsselreihe des Schlüssels sind nur eindeutige Schlüssel zugelassen, d.h., es dürfen nicht zwei gleiche Schlüsselwerte zu verschiedenen Datensätzen vorkommen. Der veränderte Datensatz konnte nicht in die Jetsamdatei geschrieben werden, weil der neue Schlüsselwert bereits in der Jetsamdatei vorhanden ist.
Bei Eingabe eines "J" können die Werte erneut überarbeitet werden, um die Daten richtig zu speichern. Soll der Datensatz auf die alten Werte zurück geändert werden, ist ebenfalls "J" einzugeben, um die Veränderungen rückgängig zu machen, denn der **alte Datensatz** wurde programmintern **bereits gelöscht!**
Bei "N" wird die Veränderung nicht beachtet, der alte bisherige Datensatz wurde jedoch gelöscht und steht nicht mehr zur Verfügung!

**Weitere Veränderungen?**

Weitere Daten können verändert werden, wenn ein "W", "V" oder Leertaste eingegeben wird. Die Eingabe eines "X" beendet den Programmteil zum Verändern von Daten.

Zu den Fehlermeldungen vgl. die Erläuterungen zu Eingabe.

Löschen

**Bitte Schlüsselwert eingeben**

Zunächst wird nach dem Schlüsselwert gefragt, um den Datensatz zu suchen und einzulesen, der gelöscht werden soll.
Soll der Datensatz gelöscht werden, der über den Schlüssel NAME mit dem Wert "Hugo" gefunden werden

kann, wird hier aufgefordert, den Namen einzugeben. Geben Sie dann bitte "Hugo" ein.

**Löschung mit "L" bestätigen - andernfalls ENTER drücken**

Es wird hier nochmals nachgefragt, ob die Daten tatsächlich gelöscht werden sollen. Bei Eingabe eines "L" werden die Daten aus der Jetsamdatei entfernt. Wird die Taste [RETURN] oder [ENTER] betätigt, werden die Daten nicht gelöscht.

**Weitere Löschungen?**

Weitere Daten können gelöscht werden, wenn ein "W", "L" oder Leertaste eingegeben wird. Die Eingabe eines "X" beendet den Programmteil zum Löschen von Daten.

Anzeigen

**Anzeige von Schlüssel1 Schlüssel2 ...**
**Auswahl über 12 ...**

Hier wird gefragt, nach welchem Schlüssel nach Daten gesucht werden soll. Durch Eingabe der bei der Generierung angegebenen einstelligen Kennung des Schlüssels wird der entsprechende Schlüssel ausgewählt. Unten auf dem Bildschirm werden sämtliche einstelligen Schlüsselkennungen angezeigt.
Durch Eingabe eines "X" wird der Programmteil zum Anzeigen von Daten beendet.

**Alle oder Einzeln anzeigen?**

Sollen alle Daten zu dem Schüssel angezeigt werden, ist ein "A" einzugeben. Nacheinander werden dann sämtliche Daten aufgelistet.

**Bitte eine Taste drücken**

Ist die letzte Zeile des Bildschirms erreicht, wird aufgefordert, eine beliebige Taste zu drücken, damit die nächste Bildschirmseite zur Anzeige weiterer Daten genutzt werden kann (Seitenwechsel).



Sämtliche Daten zu einem Datensatz werden stets zusammenhängend angezeigt, d.h.: keine Trennung der Daten durch Seitenwechsel.

Sollen nur bestimmte Daten ausgewählt werden, ist ein "E" einzugeben.

**Bitte Schlüsselwert eingeben**

Es wird nach dem Schlüsselwert gefragt, um den Datensatz zu suchen und einzulesen, der angezeigt werden soll.
Soll der Datensatz angezeigt werden, der über den Schlüssel NAME mit dem Wert "Hugo" gefunden werden kann, wird hier aufgefordert, den Namen einzugeben. Geben Sie dann bitte "Hugo" ein.
Es werden sämtliche Datensätze aufgelistet, die den eingegebenen Schlüsselwert bei diesem Schlüssel haben (alle Datensätze des Schlüsselsatzes). Ggf. wird ein Seitenwechsel durchgeführt wie bei 'Alle Anzeigen'.

**Weitere Daten Anzeigen?**
oder
**Weitere Daten Anzeigen? In rückwärtiger Reihenfolge (Druck=T)?**

Weitere Daten können auf dem Bildschirm angezeigt werden, wenn ein "W", "D", "A" oder Leertaste eingegeben wird. Die Eingabe eines "X" beendet den Programmteil zum Anzeigen von Daten.
mit dem Zusatz **"In rückwärtiger Reihenfolge (Druck=T)?":**
Bei Eingabe eines "R" werden die Daten in umgekehrter Reihenfolge - rückwärts - erneut angezeigt. Bei Eingabe eines "T" werden die Daten in umgekehrter Reihenfolge auf dem Drucker ausgegeben - legen Sie dafür bitte Papier in Ihren Drucker ein.

Drucken

**Bitte Papier einlegen und bestätigen**

Legen Sie jetzt bitte das Papier in den Drucker ein. Durch Eingabe von "B" oder "P" wird dem Programm mitgeteilt, daß Papier eingelegt wurde. Die Eingabe eines "X" beendet den Programmteil zum Drucken von Daten.

Ansonsten läuft dieser Programmteil genauso ab wie der Programmteil Anzeigen, jedoch wird nichts auf dem Bildschirm angezeigt, sondern alles auf dem Drucker ausgegeben. Blattwechsel werden automatisch berücksichtigt. Die Daten zu einem Datensatz werden stets zusammenhängend ausgegeben, d.h. sie werden nicht durch Seitenwechsel getrennt. Auf ein Blatt werden maximal 66 Zeilen gedruckt.

**Weitere Daten Drucken?**
oder
**Weitere Daten Drucken? In rückwärtiger Reihenfolge?**

Weitere Daten können gedruckt werden, wenn ein "W", "D" oder Leertaste eingegeben wird. Die Eingabe eines "X" beendet den Programmteil zum Drucken von Daten.
Mit dem Zusatz **"In rückwärtiger Reihenfolge?"**:
Bei Eingabe eines "R" werden die Daten in umgekehrter Reihenfolge - rückwärts - erneut gedruckt.

Die Einstellung des Druckers ist vom Programm wie folgt vorgesehen:

Elite - 12 Zeichen je Zoll
Einzelblatt
Seitenlänge 76 Zeilen
Lücke am Seitenende (nicht bedruckbare Zeilen am Ende der Seite) 6 Zeilen
Linker Rand 10 Stellen
Zeilenlänge 86 Stellen

Möchten Sie eine andere Standardeinstellung, sind die Zeilen 19300 und 19310 im Programm zu ändern. Dabei helfen die vorbereiteten Steuercodes in den Zeilen 18730 bis 19280. Es



sind sämtliche möglichen Steuercodes zur Steuerung des Druckers vorhanden. Die Steuercodes BS, LF und CR sind bereits bei den Steuercodes für die Bildschirmsteuerung berücksichtigt und können auch für die Steuerung des Druckers benutzt werden.

Auf den folgenden Seiten wird der Aufbau des Generators und des generierten Programms näher erläutert.

Es wird die Einteilung des Bildschirms gezeigt und sämtliche Jetsam-Meldungen aufgeführt, die vom generierten Programm ausgegeben werden.

Außerdem wird erklärt, wie das Programmlisting von GENERJET ausgedruckt werden kann.

Dann folgt eine Liste der Variablen, die im generierten Programm und im Generator benutzt werden, und es werden die Zeilen genannt, in denen die einzelnen Programmteile zu finden sind.

Mögen die Programmroutinen in GENERJET Sie zu eigenen Jetsam-Programmen anregen und ermutigen!

Die Bildschirmzeilen werden wie folgt verwendet:

| Zeile | Verwendung |
|---|---|
| 0 | Titelzeile |
| 1 | Leerzeile |
| 2 | Programmauswahlpunkte mit Anzeige des aktiven Programmteils |
| 3- 5 | Abfrage von steuernden Informationen bei Verändern, Anzeigen und Drucken |
| 6-27 | Datenteil für Anzeigen und Eingabe |
| 28 | Leerzeile |
| 29 | Abfrage auf "weiter arbeiten" im Programmmodus oder Ende des Programmmodus (Stets Auswahl "X") |
| 30 | Fehlermeldungen |
| 31 | Leerzeile |

Auf Zeile 30 des Bildschirms werden zu sämtlichen Jetsam-Kommandos und -Funktionen aufgrund des Antwortcodes entsprechende Meldungen ausgegeben:

| Antwort-code | Text |
|---|---|
| 101 | Neuer Schlüsselwert ist ... |
| 102 | Neue Schlüsselreihe ..., neuer Schlüsselwert ... |
| 102 | bei SEEKRANK:<br>Keine Einträge in der angegebenen Schlüsselreihe. Position ist auf Reihe ... |
| 103 | Keine weiteren Werte gespeichert |
| 105 | Die aktuelle Position wurde nicht gefunden. Die aktuelle Position ist unbestimmt.<br>oder<br>Die aktuelle Position wurde nicht gefunden. Die Position wurde eingerichtet auf Reihe ... Satznummer ... mit dem Schlüssel ... |
| 105 | bei SEEKKEY:<br>... nicht gefunden. Position - Reihe ... Satznummer ... Schlüssel ... |
| 115 | Keine aktuelle Position. Eine Positionierung konnte nicht eingerichtet werden. |
| 116 | ... ist bereits vorhanden. Mehrfachschlüssel sind nicht zulässig. |
| 130 | Die Datei bekam von einem anderen Benutzer eine Lesesperre |
| 131 | Der Satz, der zum Schlüssel gehört, konnte nicht schreibgesperrt werden |
| 132 | Eine Lesesperre konnte nicht eingerichtet werden |
| 133 | Eine Schreibsperre konnte nicht eingerichtet werden |

Die Datei wurde von ... Benutzern verändert.



**Programmaufbau**

Um ein Listing von GENERJET.BAS zu erhalten, starten Sie bitte CP/M, indem Sie Ihren JOYCE einschalten und Ihre CP/M-Startdiskette in Laufwerk A: legen.

Tippen Sie dann:

```
A>BASIC[ENTER]
```

**[ENTER]** bedeutet, daß Sie die [RETURN]- bzw. [ENTER]-Taste betätigen.

Es wird BASIC in den Arbeitsspeicher geladen und meldet sich mit **OK**. Laden Sie dann GENERJET von der zum Buch erhältlichen Diskette:

```
OK
load "generjet[ENTER]

OK
```

Damit ist GENERJET geladen. Spannen Sie jetzt einen Bogen Papier in den Drucker und halten Sie 9 weitere Blätter Papier bereit. Tippen Sie dann:

```
llist[ENTER]
```

Das Programmlisting wird jetzt auf dem Drucker ausgegeben.

Es werden folgende Variablen benutzt:

**im generierten Programm**

| | |
|---|---|
| abbruch | Flag, daß beim Einfügen eines Satzes bzw. Schlüssels ein schwerwiegender rc auftauchte |
| aendern | Flag für Programmteil Verändern |
| druck | Flag für Programmteil Drucken; Wert 2: Druck bei rückwärts lesen |
| dsn$ | Dateibezeichnung der Jetsamdatei ohne Extension |
| ein$ | Mögliche Auswahloptionen bei einer Frage an den Benutzer für INKEY$ |
| ein | Weist auf die gewählte Auswahloption hin |
| einzeln | Flag für 'Anzeigen Einzelne' |
| einzig | Flag für RANKSPEC - eindeutige oder mehrdeutige Schlüssel möglich |
| fehler$ | Hinweis auf den zulässigen Wertebereich bei Zahlen |
| frei | Freie Speicherstellen im BASIC-Speicher - für Garbage Collection |
| grenz1$,grenz2$ | Hinweis auf maximal mögliche Stellenzahl bei Texteingaben |
| init | Flag für Begrüßungsteil |
| key$ | Aktueller Schlüsselwert |
| modus$ | Name des aktiven Programmteils |
| modus | Index zu modus$ |
| pgm$ | Programmname |
| rc1 | Zwischenspeicher für rc |
| rc | Antwortcode der Jetsamfunktionen |
| reihe | Aktuelle Schlüsselreihe |
| satzlaenge | Länge des Jetsam-Datensatzes |
| satznr | Aktuelle Satznummer des Datensatzes |
| t$ | Über INKEY$ von der Tastatur eingelesenes Zeichen |
| text$ | Textkonstante für den Begrüßungsteil mit der Kurzbeschreibung des Programms |
| x$ | Enthält CHR$(34), um " auf dem Bildschirm darstellen zu können |
| xrec | Zwischenspeicher für die Satznummer |



| | |
|---|---|
| xrank | Zwischenspeicher für die Reihe |
| xkey$ | Zwischenspeicher für den Schlüssel |
| yrueck | Flag für rückwärts lesen |
| zahl$ | Nimmt Zahlenwerte auf zur Prüfung auf numerischen Inhalt der Zahl, denn auch Zahlen werden in Stringvariablen eingegeben. |
| zz | Zeilenzähler für Bildschirm- und Druckerausgabe |

**im Generator**

| | |
|---|---|
| anzeige$ | Schlüsselbezeichnungen |
| datum$ | Eingegebenes Tagesdatum |
| dein$ | Kürzel der Schlüsselbezeichnungen |
| dfeld$ | Feldname in Großbuchstaben |
| dmax,dmin | Zwischenspeicher für Maximalwert und Minimalwert zu einem Zahlenfeld |
| dreihe$ | Generiert eine Zeile zu Beginn des Eingabeteils, um bei anderen Programmmodi als 'Eingabe' zum Schlüsselfeld zu verzweigen, nach dem in der Jetsamdatei ein Datensatz gesucht werden soll |
| feldlaenge | Länge eines Datenfeldes in Bytes |
| feldname$ | Feldname des Datenfeldes |
| gf | Index für gfield$ |
| gfield$ | Anweisungen für FIELD |
| imax,imin | Eingegebener Maximal- und Minimalwert bei Zahlenfeldern |
| keylen | Länge des Schlüsselbegriffs bei Textfeldern |
| kfeld$ | Name des Schlüsselfeldes |
| lauf$ | Laufwerksbezeichnung |
| ldfeld$ | Feldname in Kleinbuchstaben |
| name$ | Name des Benutzers des Generators |
| rueck$ | Rückumwandlung von komprimiert gespeicherten Zahlen |
| schluessel | Flag für Schlüsselgenerierung |
| spalte | Spalte der Bildschirmzeile, wo der Feldname für die Eingaberoutine angezeigt werden soll |
| string | Flag für Generierung eines Stringfeldes |

| | |
|---|---|
| umw$ | Umwandlung von Zahlen für die komprimierte Speicherung |
| x | Laufvariable für Schleifen |
| y$ | Zwischenspeicher für t$ bei Vorschlag der einstelligen Kennung |
| zahl | Zahlentyp - CHR$, Integer, vorzeichenlose Integerzahl, einfach oder doppelt genaue Zahl |
| zaddkey | Zeilenzähler für ADDKEY-Befehle |
| zdelkey | Zeilenzähler für DELKEY-Befehle |
| zeile | Zeile, wo der Feldname für die Eingaberoutine angezeigt werden soll und für Generierung der RETURN-Zeilen |
| zfeld | Zähler für die Anzahl der generierten Felder |
| zfield | Zeilenzähler für FIELD-Befehle |
| zinput | Zeilenzähler für INPUT-Befehle |
| zleft | Zeilenzähler für LEFT$-Befehle |
| zlprint | Zeilenzähler für LPRINT-Befehle |
| zlset | Zeilenzähler für LSET-Befehle |
| zmax,zmin | Minimum und Maximum des jeweiligen Zahlentyps |
| zprint | Zeilenzähler für PRINT-Befehle |
| zrankspec | Zeilenzähler für RANKSPEC-Befehle |
| zreihe | Zähler der aktuell generierten Schlüsselreihe |
| zseekkey | Zeilenzähler für SEEKKEY-Befehle |

Werden Variablen sowohl im generierten Programm als auch im Generator verwendet, sind sie beim generierten Programm beschrieben.

Alle übrigen Variablen sind Steuercodes für den Bildschirm oder Drucker - siehe die Kommentare im Programmlisting - Zeilen 18290 bis 19310.

Die einzelnen Programmroutinen sind zu finden in den Zeilen:

| Zeilen | Programmteil |
|---|---|
| 10 - 90 | Hauptsteuerleiste<br>Ansteuerung der Programmteile zur Initialisierung des Bildschirms und des Druckers, Aufbau des Begrüßungsbildes, Abfragen der Tastatur, Ansteuern des Generators und der Enderoutinen.<br>In Zeile 80 wird das generierte Programm geladen und der Programmteil des Generators gelöscht. |
| 100 - 1310 | Generator mit folgenden Programmteilen: |
| 100 - 370 | Einleitung und Initialisierung für den Generator |
| 380 - 950 | Felder generieren |
| 960 - 1200 | Schlüssel generieren |
| 1210 - 1270 | Zahlenfelder verdichten und wieder aufbereiten |
| 1280 - 1310 | RETURN-Zeile berücksichtigen |
| 10000 - 10580 | Jetsam-Hauptprogramm mit folgenden Programmteilen: |
| 10000 - 10090 | Ansteurung der Programmteile für Eingabe, Verändern, Löschen, Anzeigen und Drucken mit Angabe des Programmmodus auf dem Bildschirm |
| 10100 - 10170 | Programmteil für die Eingabe von Daten von der Tastatur und Schreiben der Datendatei und Schlüsseldatei |
| 10180 - 10230 | Programmteil für das Verändern von Daten; zunächst werden die Daten eingelesen und sämtliche Schlüssel des alten Datensatzes werden gelöscht; dann werden die einzelnen Daten angezeigt mit der Möglichkeit, neue Werte einzugeben; nach Eingabe wird der überarbeitete Datensatz neu in die Jetsam-Datei eingefügt |
| 10240 - 10310 | Programmteil zum Löschen von Datensätzen; zunächst werden die zu löschenden Daten eingelesen, dann sämtliche Schlüssel zu dem Datensatz gelöscht. |
| 10320 - 10510 | Programmteil zum Anzeigen von Datensätzen auf |

| | |
|---|---|
| | dem Bildschirm |
| 10520 - 10580 | Programmteil zum Ausdrucken von Datensätzen auf dem Drucker |
| | |
| 10590 - 11690 | Hilfs- und Unterroutinen für die Jetsam-Verarbeitung |
| 10590 - 10630 | Einlesen eines bestimmten Zeichens von der Tastatur |
| 10640 - 10670 | Jetsam-Kommando GET |
| 10680 - 10730 | Jetsam-Kommando PUT |
| 10740 - 10800 | Jetsam-Funktion SEEKNEXT |
| 10810 - 10870 | Jetsam-Funktion SEEKPREV |
| 10880 - 10920 | Jetsam-Funktion SEEKSET |
| 10930 - 11030 | Jetsam-Fehlerroutinen |
| 11040 - 11090 | Jetsam-Fehlerroutinen für Mehrbenutzer-Systeme |
| 11100 - 11150 | Jetsam-Funktion RANKSPEC |
| 11160 - 11190 | Jetsam-Funktion CONSOLIDATE |
| 11200 - 11250 | Jetsam-Funktion SEEKRANK |
| 11260 - 11310 | Jetsam-Funktion SEEKKEY |
| 11320 - 11370 | Jetsam-Funktion DELKEY |
| 11380 - 11440 | Jetsam-Funktion ADDKEY |
| 11450 - 11510 | Jetsam-Funktion ADDREC |
| 11520 - 11570 | Ausgabe von Erläuterungen zur Bedienung des Programms bei Programmstart |
| 11580 - 11600 | Jetsam-Funktionen FETCHRANK, FETCHREC und FETCHKEY$ |
| 11610 - 11620 | Bildschirm bis auf die Kopfzeilen löschen |
| 11630 - 11690 | Prüfung auf numerischen Inhalt bei eingegebenen Zahlenwerten - Exponentialdarstellung wird berücksichtigt |
| | |
| 18000 - 18050 | Programmende |
| 18060 - 18180 | Begrüßung und Initialisierung |
| 18190 - 18270 | Einlesen eines beliebigen Zeichens von der Tastatur |
| 18280 - 18710 | Sämtliche Steuercodes zur Bildschirmsteuerung |
| 18720 - 19320 | Sämtliche Steuercodes zur Druckersteuerung |



| Ab Zeile | wird generiert |
|---|---|
| 20000 | Satzbeschreibung - **FIELD** |
| 21000 | Eingabe - **INPUT** |
| 21200 | Datensatz füllen - **LSET** |
| 21400 | Schlüssel setzen - ggf. mit **MKI$, MKU$** |
| 21600 | Schlüssel einfügen - Aufruf **ADDKEY** |
| 22000 | Ausgabe auf Bildschirm - **PRINT**<br>Die hier generierten Befehle können nach Bedarf geändert werden, falls z.B. mehrere Datenfelder in einer Bildschirmzeile angezeigt werden sollen etc. Wichtig ist, den Zeilenzähler **zz** richtig zu setzen. |
| 23000 | Ausgabe auf Drucker - **LPRINT**<br>Die hier generierten Befehle können nach Bedarf geändert werden, falls z.B. mehrere Datenfelder in einer Druckzeile ausgegeben werden sollen etc. Wichtig ist, den Zeilenzähler **zz** richtig zu setzen. |
| 25000 | Eindeutige Schlüssel je Schlüsselreihe - **RANKSPEC** |
| 26000 | Schlüssel suchen - Aufruf **SEEKKEY** |
| 27000 | Schlüssel löschen - Aufurf **DELKEY** |

## Turbo-Pascal

Mittlerweile gibt es eine ganze Reihe von Joyce-Anwendern, die mit Turbo-Pascal arbeiten. Diesen bieten sich einige Vorteile, die Pascal gegenüber anderen Programmiersprachen auszeichnet.
Der Hauptvorteil von Turbo-Pascal ist, daß der eingegebene Programmtext automatisch in schnellen Z80-Code compiliert wird. Das Gegenstück zum Compiler ist der Interpreter, der sich Befehl für Befehl vom Programmtext holt und ihn dann auszuführen versucht (so beim mitgelieferten Mallard-80 BASIC oder bei LOGO). Turbo-Pascal hingegen übersetzt das Programm und sucht schon während des Übersetzens nach Fehlern und meldet sie. Dadurch wird das Programm nicht nur schnell, es kommt auch während der Ausführung seltener zu Abbrüchen.
Aber selbst ungewollte Programmabbrüche z.B. durch falsche Eingaben lassen sich durch entsprechende Programmierung abfangen.
Der größte Vorteil der in Pascal erstellten Programme besteht in ihrer Struktur. Als gegenteiliges Beispiel sei hier auf längere in BASIC geschriebene Programme hingewiesen. Bei ihnen wird der Text trotz guter Vorsätze zum Schluß meist durch häufige Zeilensprünge und Sprünge in Unterprogramme unübersichtlich, zumindest für nicht in das Programm eingearbeitete Personen. Bei Turbo-Pascal hingegen wird man von Anfang an gezwungen, strukturiert zu programmieren, da man keine Zeilennummern verwendet. Vor dem eigentlichen Programm wird festlegt, welche Variablen, Konstanten, etc. verwendet werden sollen. Auch kann man Unterprogramme definieren, die während des Hauptprogramms durch einfache Angabe ihres Namens aufgerufen werden. Der Vorteil dabei liegt darin, daß die Prozeduren vor dem Hauptprogramm definiert werden und der Anwender somit schnell über die Möglichkeiten des Programms informiert wird und nicht mit einer unidentifizierbaren Zeilennummer, sondern mit dem Namen konfrontiert wird. Zum Vergleich führe ich jetzt das schon von LOGO bekannte Programm zur



Dreiecksberechnung in der Turbo-Pascal Version an. Zwar werden an diesem Programm die tatsächlichen Vorzüge Turbo-Pascals nicht deutlich, doch wird hier ein ungefährer Einblick in die Art des Programmierens mit Pascal gewährt:

```
program Dreieck;

var seitea, seiteb, seitec: real;

begin
 clrscr;
 write('Seite a ?'); readln(Seitea);
 write('Seite b ?'); readln(Seiteb);
 seitec := sqrt (sqr (Seitea) + sqr (Seiteb));
 writeln('Seite c = ',Seitec:10:4);
end.
```

Zum Programm: Zunächst werden die Variablen definiert. Dann wird nach Ankündigung des Hauptprogramms mit **begin** der Bildschirm durch **clrscr** gelöscht. Danach wird eine Eingabe für die Seitenlänge von **a** und **b** verlangt. Seite **c** wird ausgehend von Pythagoras (a↑2+b↑2=c↑2 bzw. c=wurzel (a↑2+b↑2)) berechnet. Beim Beispiel ist zu beachten, daß die Wurzelfunktion durch **sqrt** dargestellt wird. Das bekannte **sqr** stellt unter Turbo-Pascal eine Quadratfunktion dar. Zum Schluß wird das Ergebnis auf dem Bildschirm ausgegeben, wobei ein Format von 10 Vorkomma- und 4 Nachkommastellen verwendet wird.
Zur weiteren Information über die Arbeit mit Pascal:

Turbo-Pascal systematisch
B.Ciric/D.Thies
Tewi München 1987

## MI-C / C-Compiler

C ist eine der jüngsten Sprachen. Sie entstand als "Abfallprodukt" aus der Entwicklung eines Betriebssystems für Unixrechner. BCPL, das keine Strukturierung von Daten zuließ, wurde zu diesem Zweck weiterentwickelt, wobei gerade auf mögliche Vielfalt von Datentypen Rücksicht genommen

wurde. Im Endeffekt stellte C dann ein mächtiges Werkzeug zur Programmierung dar. Wie alle jüngeren Programmiersprachen arbeitet C mit einem Compiler, der einen maschinennahen Code erzeugt. Mittlerweile steht dem Joyce-User auch dieser Renner unter den Programmiersprachen zur Verfügung, der bisher in erster Linie MS-Dos oder UNIX Rechnern vorbehalten war.
Zwar gab es schon seit einiger Zeit einige C-Versionen, die sich als lauffähig auf dem Joyce ausgaben, doch entweder waren sie nicht kompatibel zu anderen CP/M Systemen (Arnor) oder verfügten nicht über den Standard nach Kernighan & Ritchie. Diesem Mißstand trat eine deutsche(!) Entwicklung entgegen, die sich auf ihrem Sektor absolut durchgesetzt hat und auch in den verschiedensten Fachzeitschriften schon vor Anpassung an den Joyce (CHIP 7/84; c t 2/87; CPC 5/86, 2/87) lobenswerte Erwähnung fand. Für Entwicklung und Vertrieb dieses MI-C Compilers zeichnet die Fa.:

H.Rose EDV
Buersche Str.43
4390 Gladbeck

Hier soll auch gleich einer der ersten Vorteile dieses Compilers Erwähnung finden: Entwicklung und Vertrieb in Deutschland gewährleisten ein verständliches (und zudem umfangreiches) Handbuch in deutscher Sprache und eventuell auftretende Schwierigkeiten können durch ein kurzes Telefonat kompetent und rasch beseitigt werden. Der Preis (ca. 450,- DM; je nach Ausstattung) ist dem Lieferumfang angemessen. Der Anwender erwirbt hier ein Werkzeug des professionellen Bereichs, daß als einziges u.a. den vollständigen Standard-C Sprachumfang und Bibliotheksfunktionen nach Kernighan & Ritchie sowie Fließkommadatentypen und entsprechende Arithmetikfunktionen bietet. Erwähnenswert ist hier auch das Fehlerprotokoll, das als Datei angelegt oder über Monitor ausgegeben werden kann. Selbstverständlich sind die Meldungen in deutscher Sprache gehalten (wahlw. engl.) und der Fehler wird genau bezeichnet.
Der MI-C kann auch in gewissem Maße Fehler selbst



berichtigen, was sonst nur von Compilern für Großcomputer geboten wird. Eine Tracefunktion, die die meisten Sprachen vermissen lassen (vergl. allerdings trace unter Logo) zeigt nach Aktivierung alle Funktionsaufrufe und Werte der ablaufsteuernden Variablen. Der Anwender kann so den Programmablauf lückenlos nachvollziehen.
Die Grenzen des Compilers sind für den "normalen" Anwender kaum zu erreichen. So können auf jeder Programmzeile 159 Zeichen verwendet werden, in jedem Programmteil können mindestens 300 verschiedene externe, im kompletten Programm unbegrenzt viele Symbole Verwendung finden und bei einem Funktionsaufruf verkraftet der Compiler 40 Parameter.
Die Geschwindigkeit des Compilers ist enorm. Kleinere Programme werden in Sekundenschnelle bearbeitet. Da Assembler-Quellcode erzeugt wird, können kundige Anwender zusätzlich noch Maschinencode-Routinen in ein C-Programm einbinden.
Mit Erwerb des C-Compilers eröffnen sich den Joyce-Usern zusätzliche Möglichkeiten, da dieser "Romfähig" ist, was heißt, daß er 'tools' zur Eprom-Erzeugung enthält. Der Joycer kann jetzt Programme in C schreiben, die via Compiler an einen "Eprombrenner" (etwa den mc-Allprog) geschickt werden, der sie eingesteckten PALs, EPROMs oder PROMs zur Steuerung eines Z80 oder 8051 mit auf den Weg gibt.
(Wer sich näher über den mc-Allprog informieren möchte, sei auf die "mc" 6 & 7 1987 verwiesen. Dort finden sich auf 20 Seiten Bauanleitung und Beschreibung der Hard- und Software). Der Vollständigkeit halber nun nochmal das Dreiecksprogramm in C. Die Vorteile dieser Sprache gegenüber BASIC sind ähnlich wie bei Pascal und können an diesem kleinen Programm kaum demonstriert werden.
(Das Handbuch zum MI-C Compiler gibt übrigens Hilfen zu C)

```
double a, b;
main ()
{printf ("Geben Sie Seite a ein: "); (scanf "% lf", & a);
printf ("\n Geben Sie Seite b ein: "); (scanf "% lf" , & b);
printf ("\n die Seite c = % 8.3f", sqrt(a*a+b*b));}
```

Mit **double** wird zunächst die Gleitkommaverarbeitung signalisiert, danach die Variablen definiert. Die nächsten zwei

Zeilen fordern über Monitor die Eingabe des Wertes an und ordnen ihn der Variablen zu. Der Backslash **\n** bringt die Ausgabe auf die nächste Zeile. Das **l** von **lf** läßt mit doppelter Genauigkeit arbeiten und **f** signalisiert das Gleitkomma. (e wäre exponential) 8.3 Gibt die Länge und Anzahl der Nachkommastellen an.

Wer sich weiter über C informieren möchte, sei auf folgendes Buch verwiesen:

"Programmieren in C"
Kernighan Ritchie
München/Wien 1983 (Hanser Verlag)



# Programmierhilfen und Interna des Joyce

Auf den folgenden Seiten werden sämtliche Codes zur Steuerung des Bildschirms und des Druckers ausführlich erklärt.
Dem Anwender sollen damit entscheidende Hilfen bei der Programmierung an die Hand gegeben werden, um die Bedienung der Peripherie zu vereinfachen.
Dem etwas versierteren Programmierer werden außerdem kleine Routinen gezeigt, um die Möglichkeiten des XBIOS-Systems (e**X**tended **B**asic **I**nput/**O**utput **S**ystem) auszuschöpfen, sowie gezielte Einstellungen des Systems über den SCB (System-Control-Block) vorzunehmen.
Weiterhin wird der Aufbau des Directory erklärt und individuelle Manipulationsmöglichkeiten des Inhaltsverzeichnisses der Diskette über einen Diskettenmonitor (z.B. DU.COM) aufgezeigt.

## Steuercode-Tabelle für den Bildschirm

**CHR$(7)** läßt den eingebauten Signalgeber einen Ton von sich geben.

**CHR$(27)+"0"** Hiermit wird die Statuszeile ausgeschaltet, so daß sie für andere Zwecke nutzbar wird. Die Statuszeile ist die unterste Zeile des Bildschirms, in der beim JOYCE mit einem Laufwerk das Laufwerk angezeigt wird und die Auskunft über den aktuellen Druckerstatus gibt (aufgerufen durch die [PTR]-Taste). Außerdem werden bei Bedarf CP/M-Fehlermeldungen in der Statuszeile angezeigt (z.B. **B: Laufwerk nicht bereit - Wiederholen, Ignorieren oder Abbrechen**). Wenn nun diese Zeile ausgeschaltet wird, kann sie auch durch normale **PRINT**-Kommandos genutzt werden. Die CP/M-Fehlermeldungen erscheinen dann mit der übrigen Bildschirmausgabe (an der aktuellen Stelle des Cursors), wo sie jedoch nach eventueller Eingabe (**W**, **I** oder **A**) nicht wieder gelöscht werden.

**CHR$(27)+"1"** schaltet die Statuszeile wieder ein und macht alle Einstellungen von **CHR$(27)+"0"** rückgängig. Ein weiterer Effekt ist, daß ein eventuell bestehendes Fenster (s. **CHR$(27)+"X"**) aufgehoben wird.

**CHR$(27)+"2"+CHR$(zeichensatz)** schaltet den eingestellten Zeichensatz auf den durch **zeichensatz** gekennzeichneten um. Die Möglichkeiten sind folgende:

**zeichensatz** =

0 Amerikanisch
1 Französisch
2 Deutsch (Normaleinstellung)
3 Englisch
4 Dänisch
5 Schwedisch
6 Italienisch
7 Spanisch
8 Japanisch

Die Unterschiede liegen (vom deutschen ausgehend) bei folgenden Zeichen: #, $, §, Ä, Ö, Ü, ↑, ` , æ, 0.

**CHR$(27)+"A"** bewegt den Cursor eine Zeile nach oben. Wenn er sich schon in der obersten Zeile befindet, hat diese Escape-Folge keine Bedeutung.

**CHR$(27)+"B"** bewegt den Cursor eine Zeile nach unten. Wenn er sich schon in der untersten Zeile befindet, hat diese Escape-Folge keine Bedeutung.

**CHR$(27)+"C"** bewegt den Cursor eine Spalte nach rechts. Wenn er sich schon in der ersten Spalte befindet, hat diese Escape-Folge keine Bedeutung.

**CHR$(27)+"D"** bewegt den Cursor eine Spalte nach links. Wenn er sich schon in Spalte 90 befindet, bleibt die Escape-Folge ohne Wirkung.

**CHR$(27)+"E"** löscht das aktuelle Fenster. Der Cursor bleibt an seiner aktuellen Position.

**CHR$(27)+"H"** rückt den Cursor in die linke obere Fensterecke.

**CHR$(27)+"I"** setzt den Cursor eine Zeile höher. Ist der Cursor schon in der obersten Zeile, rollt das Bild eine Zeile nach unten.

**CHR$(27)+"J"** löscht den Bildschirm ab der aktuellen Position des Cursors.

**CHR$(27)+"K"** löscht den Rest der aktuellen Zeile ab der Position des Cursors.

**CHR$(27)+"L"** fügt eine Zeile ein. Die aktuelle und alle darunter befindlichen Zeilen rücken eine Zeile nach unten, so daß eine Leerzeile entsteht. Der Cursor bleibt an seiner aktuellen Position.



**CHR$(27)+"M"** löscht die aktuelle Zeile. Alle darunter befindlichen rücken eine Zeile nach oben. Der Cursor bleibt an seiner aktuellen Position.

**CHR$(27)+"N"** löscht das Zeichen, das sich an der Stelle des Cursors befindet. Alle rechtseitigen Zeichen rücken ein Zeichen nach links. Die Wirkung dieser Escape-Folge ähnelt also der [DEL→]-Taste.

**CHR$(27)+"X"+CHR$(32+zeile)+CHR$(32+spalte)+CHR$(31+hoehe)+CHR$(31+breite)** grenzt ein Fenster auf dem Bildschirm ein. Das Fenster ist sozusagen ein zusätzlicher verkleinerter Bildschirm des restlichen Monitors, da um das Fenster herum der Bildschirminhalt erhalten bleibt, solange nicht gescrollt wird. **zeile** gibt die obere Begrenzungszeile des Fensters an. **spalte** gibt die linke Begrenzungsspalte an, **hoehe** wie viele Zeilen das Fenster haben soll und **breite** nennt die Fensterbreite in Zeichen. Der Cursor wird an eine Stelle innerhalb des Fensters gesetzt. Um das Fenster ohne große Probleme wieder auf den ganzen Bildschirm auszuweiten, benutzt man am besten **CHR$(27)+"1"**, was diesen zusätzlichen Effekt hervorruft.

**CHR$(27)+"Y"+CHR$(32+zeile)+CHR$(32+spalte)** rückt den Cursor an eine bestimmte Position auf dem Bildschirm. **zeile** gibt die Zeile an, **spalte** die Spalte der Position, auf die der Cursor gesetzt werden soll. Wenn die bezeichnete Position außerhalb des Bildschirms liegt, wird der Cursor an die nächste innerhalb des Bildschirms liegende Position gesetzt.

**CHR$(27)+"b"+CHR$(farbe)** tauscht die Hintergrund- und Schriftfarbe. Für **farbe** nimmt man am besten entweder 0 oder 9. 0 setzt den Bildschirmhintergrund dunkel und die Schriftfarbe hell, 9 den Bildschirmhintergrund hell und die Schriftfarbe dunkel. Wenn die Schriftfarbe dunkel ist, werden z.B. Hi-Res Grafiken schärfer, da sich zwei Pixel nebeneinander nicht verstärken.

**CHR$(27)+"c"+CHR$(farbe)** wie **CHR$(27)+"b"+CHR$(farbe)**

**CHR$(27)+"d"** löscht das aktuelle Fenster vom Beginn bis zur aktuellen Position des Cursors. Der Cursor bleibt dabei an seiner Position.

**CHR$(27)+"e"** macht den Cursor wieder sichtbar.

**CHR$(27)+"f"** macht den Cursor unsichtbar.

**CHR$(27)+"j"** speichert die aktuelle Cursorposition.

**CHR$(27)+"k"** setzt den Cursor auf die zuletzt durch **CHR$(27)+"j"** gespeicherte Position.

**CHR$(27)+"l"** löscht die aktuelle Zeile. Im Gegensatz zu **CHR$(27)+"M"** rücken die darunter liegenden Zeilen nicht nach oben nach, so daß eine Leerzeile entsteht.

**CHR$(27)+"o"** löscht die aktuelle Zeile bis zur Position des Cursors. Der Cursor bleibt danach an seiner Position

**CHR$(27)+"p"** stellt die inverse Darstellungsart ein. Dabei werden bei einem Zeichen jeweils die Pixelfarben vertauscht, das Zeichen wird also dunkel auf hellem Hintergrund geschrieben.

**CHR$(27)+"q"** stellt die durch **CHR$(27)+"p"** eingestellte inverse Darstellungsart wieder ab.

**CHR$(27)+"r"** stellt den Unterstreichungsmodus ein.

**CHR$(27)+"u"** stellt den Unterstreichungsmodus wieder ab.

**CHR$(27)+"v"** schaltet den Bildumlaufmodus ein, bei dem automatisch eine neue Zeile begonnen wird, wenn der Cursor am Ende einer Zeile angekommen ist. Die Zeichen, die nicht mehr auf die Zeile passen, werden an den Anfang der nächsten Zeile gesetzt.

**CHR$(27)+"w"** schaltet den Bildumlaufmodus aus, so daß der Cursor, wenn er am Ende einer Zeile angekommen ist, nicht in die nächste Zeile springt.

**CHR$(27)+"x"** verkleinert den Bildschirm auf 24 Zeilen und 80 Spalten.

**CHR$(27)+"y"** vergrößert den Bildschirm wieder auf Normalmaß (32 x 90).

Da jeder dieser Codes einen String darstellt, muß er auch entsprechend angewendet werden, z.B. durch `PRINT CHR$(27)+ "f"` (Cursor abstellen). Auch kann man einen String bilden, der aus mehreren Codes zusammengesetzt ist, z.B. `cls$=CHR$(27)+"E"+CHR$(27)+"H":PRINT cls$` (Bildschirm löschen und Cursor nach links oben).



## Steuercode-Tabelle für den Drucker

**CHR$(8)** Dieser Code bewirkt das Rücksetzen des Druckkopfs um ein Zeichen der eingestellen Schriftart. Bei Proportionalschrift rückt der Druckkopf um 1/12 Zoll nach links.

**CHR$(10)** Das Papier wird um eine Zeile mit dem eingestellten Zeilenabstand vorgeschoben.

**CHR$(12)** Der Drucker führt einen Seitenvorschub aus, d.h. das Papier wird bis zur ersten Zeile der folgenden Seite vorgeschoben.

**CHR$(13)** Der Druckkopf wird zum linken Rand geführt (Wagenrücklauf).

**CHR$(24)** Alle im Druckerpuffer befindlichen Daten werden gelöscht (Reset).

**CHR$(27)+CHR$(10)** Der automatische Zeilenvorschub wird eingestellt. Jeder Wagenrücklauf wird als Wagenrücklauf plus Zeilenvorschub ausgeführt.

**CHR$(27)+CHR$(13)** Der automatische Zeilenvorschub wird aufgehoben.

**CHR$(27)+CHR$(15)** 17 Zeichen / Zoll wird eingestellt.

**CHR$(27)+CHR$(18)** Rückkehr von 17 Zeichen / Zoll zu 10 Zeichen / Zoll.

**CHR$(27)+"$"** Als Papier wird Einzelblatt verwendet.

**CHR$(27)+"-"+CHR$(0)** Die Unterstreichung der Schrift wird aufgehoben.

**CHR$(27)+"-"+CHR$(1)** Die gegenwärtige Schrift wird unterstrichen gedruckt.

**CHR$(27)+"4"** Die gegenwärtige Schrift wird kursiv gedruckt.

**CHR$(27)+"5"** Rückkehr von Kursivschrift zu Normalschrift.

**CHR$(27)+"8"** Das Papierende wird ignoriert.

**CHR$(27)+"9"** Das Papierende wird angezeigt.

**CHR$(27)+"§"** Der Drucker wird auf seine Standardeinstellung zurückgesetzt.

**CHR$(27)+"C"+CHR$(n)** Die Seitenlänge wird auf **n** Zeilen des aktuellen Zeilenabstandes eingestellt.

**CHR$(27)+"C"+CHR$(0)+CHR$(n)** Die Seitenlänge wird auf **n** Zoll gesetzt.

**CHR$(27)+"E"** Die gegenwärtige Schrift wird **fett** gedruckt.

**CHR$(27)+"H"** Aufhebung der durch Fettdruck verstärkten Schrift.

**CHR$(27)+"G"** Die gegenwärtige Schrift wird durch Doppelanschlag verstärkt.

**CHR$(27)+"H"** Aufhebung der durch Doppelanschlag verstärkten Schrift.

**CHR$(27)+"J"+CHR$(n)** Das Papier wird um **n**/216 Zoll vorgeschoben (**n**=1..255).

**CHR$(27)+"K"+CHR$(a)+CHR$(b)** Es können Bit-Bilder mit normaler Dichte (niedrige Auflösung) gedruckt werden. a und b berechnen sich aus der Anzahl Bit-Bilder, die in eine Zeile gedruckt werden sollen:

```
a=bit_bild_anzahl MOD 256
b=INT(bit_bild_anzahl / 256)
```

Wenn der Drucker in den Grafik-Modus versetzt worden ist, können die Bit-Bilder jeweils mit **LPRINT CHR$(wert)** zu Papier gebracht werden (**wert**=0..255).
Eine Zeile mit normaler Dichte kann maximal 480 Bilder aufnehmen. Ist die Zeile fertig gedruckt, wird der Graphik-Modus wieder abgeschaltet. Vorher **WIDTH LPRINT 255** einstellen.

**CHR$(27)+"L"+CHR$(a)+CHR$(b)** Es können Bit-Bilder mit doppelter Dichte (hohe Auflösung) gedruckt werden. a und b berechnen sich aus der Anzahl Bit-Bilder, die in eine Zeile gedruckt werden sollen:

```
a=bit_bild_anzahl MOD 256
b=INT(bit_bild_anzahl / 256)
```

Wenn der Drucker in den Grafik-Modus versetzt worden ist, können die Bit-Bilder jeweils mit **LPRINT CHR$(wert)** zu Papier gebracht werden (**wert**=0..255).
Eine Zeile mit doppelter Dichte kann maximal 960 Bilder aufnehmen. Ist die Zeile fertig gedruckt, wird der Graphik-Modus wieder abgeschaltet. Vorher **WIDTH LPRINT 255** einstellen.

**CHR$(27)+"M"** 12 Zeichen / Zoll wird eingestellt.

**CHR$(27)+"N"+CHR$(n)** Am Ende des Papieres werden **n** Zeilen freigelassen (n=1..127).

**CHR$(27)+"O"** Hebt **CHR$(27)+"N"** auf.

**CHR$(27)+"P"** Rückkehr von 12 Zeichen / Zoll zu 10 Zeichen / Zoll.



**CHR$(27)+"R"+CHR$(n)** schaltet den eingestellten Zeichensatz auf den durch **n** gekennzeichneten um. Die Möglichkeiten sind folgende:

| **n** = | |
|---|---|
| | 0 Amerikanisch |
| | 1 Französisch |
| | 2 Deutsch (Normaleinstellung) |
| | 3 Englisch |
| | 4 Dänisch |
| | 5 Schwedisch |
| | 6 Italienisch |
| | 7 Spanisch |
| | 8 Japanisch |

Die Unterschiede liegen (vom deutschen ausgehend) bei folgenden Zeichen: #, $, §, Ä, Ö, Ü, ↑, ` , æ, 0.

**CHR$(27)+"S"+CHR$(0)** Die gegenwärtige Schrift wird hochgestellt gedruckt.

**CHR$(27)+"S"+CHR$(1)** Die gegenwärtige Schrift wird tiefgestellt gedruckt.

**CHR$(27)+"T"** Aufhebung von Hoch- oder Tiefstellung der Schrift.

**CHR$(27)+"W"+CHR$(0)** Rückkehr von doppelter Schriftgröße zur normalen Größe.

**CHR$(27)+"W"+CHR$(1)** Die gegenwärtige Schrift wird doppelt breit gedruckt.

**CHR$(27)+"X"** Die Null wird mit Schrägstrich gedruckt (Ø).

**CHR$(27)+"c"** Als Papier wird Endlospapier verwendet.

**CHR$(27)+"d"** Alle gegenwärtigen Einstellungen für den Drucker werden als Standardeinstellung festgesetzt.

**CHR$(27)+"o"** Die Null wird ohne Schrägstrich gedruckt (0).

**CHR$(27)+"p"+CHR$(0)** Rückkehr von Proportionalschrift zu 10 Zeichen / Zoll.

**CHR$(27)+"p"+CHR$(1)** Proportionalschrift wird eingestellt.

**CHR$(27)+"x"+CHR$(0)** Die Zeichen werden in Entwurfs-Qualität gedruckt.

**CHR$(27)+"x"+CHR$(1)** Die Zeichen werden in Korrespondenz-Qualität gedruckt.

Da jeder dieser Codes einen String darstellt, muß er auch entsprechend angewendet werden, z.B. durch `LPRINT CHR$(27)+"E"` (Text fett drucken). Auch kann man einen String bilden, der aus mehreren Stuercodes zusammengesetzt ist, z.B. `breit$= CHR$(27)+"M"+CHR$(27)+"W"+CHR$(1):LPRINT breit$` (12 Zeichen / Zoll doppelt breit drucken).

# Nützliche XBIOS-Routinen

## Bildschirm zurücksetzen

Die folgende Routine ist die einfachste der XBIOS-Routinen, da keinerlei Parameter übergeben werden. Sie löscht den Bildschirm, setzt den Cursor in die linke obere Ecke des Bildschirms und schaltet den Cursor an.

```
10 MEMORY &HF4FF
20 FOR i=&HF500 TO &HF505
30   READ a$
40   POKE i,VAL("&H"+a$)
50 NEXT
60 adr=&HF500
70 CALL adr
80 END
90 DATA CD,5A,FC          :' CALL XBIOS
100 DATA C2,00            :' DW 00C2H
110 DATA C9               :' RET
```

## Statuszeile ermitteln

Die folgende Routine ermittelt, ob die Statuszeile ein- oder ausgeschaltet ist.

```
10 MEMORY &HF4FF
20 FOR i=&HF500 TO &HF510
30   READ a$
40   POKE i,VAL("&H"+a$)
50 NEXT
60 adr=&HF500
90 CALL adr
91 PRINT "Die Statuszeile ist ";
93 IF PEEK(&HF511)=255 THEN PRINT "aus"; ELSE PRINT "ein";
94 PRINT "geschaltet."
100 DATA 3E,00            :' LD A,00H        ; Speicherplatz
110 DATA 32,11,F5         :' LD (0F511H),A   ; initialisieren
120 DATA CD,5A,FC         :' CALL XBIOS      ; und prüfen...
130 DATA C5,00            :' DW 00C5H
140 DATA C0               :' RET NZ          ; Statuszeile ist an
150 DATA 3E,FF            :' LD A,0FFH       ; Speicherplatz
160 DATA 32,11,F5         :' LD (0F511H),A   ; mit 0FFH belegen
170 DATA C9               :' RET             ; und zurück
```

## Tastatur umbelegen

Die folgende Routine weist einer Taste der Tastur ein neues Zeichen zu.

```
10 OPTION RUN
20 DIM a(6)
30 MEMORY &HF54F
40 adr=&HF550
50 FOR i=&HF550 TO &HF55B
60   READ a$
70   POKE i,VAL("&H"+a$)
```



```
80 NEXT
90 CALL adr
100 DATA 06,FF            :' LD B,0FFH   ; FF = neuen ASCII-Code
110 DATA 0E,07            :' LD C,07H    ; 07 = Taste
120 DATA 16,02            :' LD D,02H    ; 02 = Codierung (siehe unten)
130 DATA CD,5A,FC         :' CALL XBIOS
140 DATA D7,00            :' DW 00D7H
150 DATA C9               :' RET
```

Die Codierung:

| | |
|---|---|
| Bit 0 | Normale Tastenebene |
| Bit 1 | SHIFT |
| Bit 2 | ALT |
| Bit 3 | SHIFT+ALT |
| Bit 4 | EXTRA |

Die Tastennumerierung entnehmen Sie bitte dem CP/M Plus Benutzer-Handbuch zum Joyce Anhang 1 Seite 7.

## Tastatur abfangen

Mit dieser Routine kann man ein Zeichen von der Tastatur abfangen, so daß auch z.B. die [SHIFT]-Taste als solche erkannt werden kann. Das Programm läuft solange, bis die SPACE-Taste gedrückt wird.

```
10 MEMORY &HF54F
20 adr=&HF550
30 b=&HF55E                    ' Speicherstelle der er- : b=Status
40 c=&HF55D                    ' mittelten Parameter    : c=Tastennummer
45 cls$=CHR$(27)+"E"+CHR$(27)+"H":PRINT cls$
50 FOR i=&HF550 TO &HF55C
60   READ a$
70   POKE i,VAL("&H"+a$)
80 NEXT
90 DATA CD,5A,FC        :' loop: CALL XBIOS   ; warten, bis Zeichen
100 DATA DA,00          :' DW 00DAH           ; von Tastatur anliegt
110 DATA D2,50,F5       :' JP NC,loop         ; kein Zeichen, dann loop
120 DATA ED,43,5D,F5    :' LD (0F55DH),BC     ; Tastencode abspeichern
130 DATA C9             :' RET
140 CALL adr
150 PRINT CHR$(27)+"HTaste:";PEEK (c)
160 PRINT "Status: ";
170 z=PEEK(b)
180 IF (z AND 128)=128 THEN PRINT "ALT  ";:'          Hier erfolgt die
190 IF (z AND 64)=64 THEN PRINT "SHIFT LOCK  ";:'     bitweise Dekodierung
200 IF (z AND 32)=32 THEN PRINT "SHIFT  ";:'          des in Register B
210 IF (z AND 16)=16 THEN PRINT "NUM LOCK  ";:'       übergebenen Status
220 IF (z AND 8)=8 THEN PRINT "Tastenwiederholung  ";
230 IF (z AND 4)=4 THEN PRINT "CAPS LOCK  ";
240 IF (z AND 2)=2 THEN PRINT "EXTRA  ";
250 PRINT SPC(50):PRINT
260 IF PEEK(c)=47 THEN END
270 GOTO 140
```

### Tastaturwiederholung einstellen

Mit dieser Routine kann man die Wiederholgeschwindigkeit der Tastatur festlegen. Sie besagt, wann das Wiederholen einer gedrückten Taste einsetzt und wie schnell sie danach wiederholt wird.

```
10 MEMORY &HF4FF
20 FOR i=&HF500 TO &HF509
30   READ a$
40   POKE i,VAL("&H"+a$)
50 NEXT
51 adr=&HF500
55 CALL adr
60 DATA 26,19            :' LD H,19H      ; Verzögerung für Einsetzen
70 DATA 2E,01            :' LD L,01H      ; Verzögerung bei Wiederholung
80 DATA CD,5A,FC         :' CALL XBIOS    ; Werte einstellen
90 DATA E0,00            :' DW 00E0H
100 DATA C9              :' RET           ; und fertig
```

### Computertyp ermitteln

Die nächste Routine ermittelt, ob dieses Programm auf einem JOYCE oder CPC 6128 abläuft. Liefert das A-Register als Ergebnis 1, so läuft das Programm auf einem JOYCE, bei A=0 handelt es sich um einen CPC 6128.

```
10 MEMORY &HF4FF
20 FOR i=&HF500 TO &HF508
30   READ a$
40   POKE i,VAL("&H"+a$)
50 NEXT
60 adr=&HF500
70 CALL adr
80 PRINT "Dieses Programm läuft auf dem SCHNEIDER ";
90 IF PEEK(&HF509)=0 THEN PRINT "CPC-6128" ELSE PRINT "JOYCE"
100 DATA CD,5A,FC      :' CALL XBIOS     ; ermitteln
110 DATA E3,00         :' DW 00E3H
120 DATA 32,09,F5      :' LD (0F509H),A  ; Ergebnis sichern
130 DATA C9            :' RET            ; und fertig
```

### Hardware-Erweiterungen ermitteln

Diese Routine informiert, wieviel Laufwerke angeschlossen sind, wieviele Speicherblöcke (zu je 16 Kbyte) zur Verfügung stehen und ob eine Schnittstelle angeschlossen ist. Das A-Register informiert über die angeschlossenen Laufwerke (A=0 ein Laufwerk, A=255 zwei Laufwerke), das B-Register enthält die Zahl der verfügbaren Speicherblocks zu je 16 KByte, das C-Register belegt, ob eine serielle Schnittstelle angeschlossen ist (C=0 keine Schnittstelle, C=255 Schnittstelle angeschlossen).

```
10 MEMORY &HF4FF
20 FOR i=&HF500 TO &HF50C
30   READ a$
40   POKE i,VAL("&H"+a$)
```



```
50 NEXT
60 adr=&HF500
70 CALL adr
80 PRINT "Angeschlossene Laufwerke: ";
90 IF PEEK(&HF50D)=0 THEN PRINT "eins" ELSE PRINT "zwei"
100 PRINT "Joyce Version: ";
110 PRINT PEEK(&HF50F)*16"K"
120 PRINT "Serielle Schnittstelle: ";
130 IF PEEK(&HF50E)=255 THEN PRINT "angeschlossen" ELSE PRINT "nicht vorhanden"
140 DATA CD,5A,FC      :' CALL XBIOS          ; Hardware-Erweiterungen
150 DATA E6,00         :' DW 00E6H              ; ermitteln
160 DATA 32,0D,F5      :' LD (0F50DH),A       ; und Ergebnisse ab-
170 DATA ED,43,0E,F5   :' LD (0F50EH),BC   ; speichern
180 DATA C9            :' RET                   ; und fertig
```

## S y s t e m - C o n t r o l - B l o c k  ( S C B )

Im SCB stehen manche für den Anwender nützliche Informationen. Sie können durch einen entsprechenden POKE geändert oder mit einem PEEK ausgelesen werden.

| Adr. | Offset | Inhalt Hex | Inhalt Dez | Erklärung | |
|---|---|---|---|---|---|
| FBA1 | 05 | 31 | 49 | BDOS-Versionsnummer | |
| FBA2 | 06 | 00 | 0 | User Flags | |
| FBA3 | 07 | 00 | 0 | | |
| FBA4 | 08 | 00 | 0 | | |
| FBA5 | 09 | 00 | 0 | | |
| FBAC | 10 | 00 | 0 | Programm Error Return Code | |
| FBAD | 11 | 00 | 0 | | |
| FBB6 | 1A | 58 | 88 | Anzahl der Bildschirmspalten | |
| FBB7 | 1B | 00 | 0 | Augenblickliche Bildschirmzeile | |
| FBB8 | 1C | 1E | 30 | Seitenlänge Bildschirm | |
| FBBE | 22 | 00 | 0 | Console Input-Flag | für die folgenden Flags |
| FBBF | 23 | 80 | 128 | | gilt: |
| FBC0 | 24 | 00 | 0 | Console Output-Flag | Bit gesetzt: |
| FBC1 | 25 | 80 | 128 | | 4: Centronics (CEN) |
| FBC2 | 26 | 00 | 0 | Auxiliary Input-Flag | 5: Seriel Input Output (SIO) |
| FBC3 | 27 | 00 | 0 | | 6: Drucker (LPT) |
| FBC4 | 28 | 00 | 0 | Auxiliary Output-Flag | 7: Console (CRT) |
| FBC5 | 29 | 00 | 0 | | |
| FBC6 | 2A | 00 | 0 | Printer Output-Flag | |
| FBC7 | 2B | 40 | 64 | | |
| FBC8 | 2C | 00 | 0 | Page Modus | |

| Adr. | Offset | Inhalt Hex | Inhalt Dez | Erklärung |
|---|---|---|---|---|
| FBCA | 2E | 00 | 0 | Flag für CTRL-H = DEL |
| FBCB | 2F | FF | 255 | Flag für DEL = CTRL-H |
| FBCF | 33 | 00 | 0 | Konsolen-Modus (s. BDOS-Aufruf 109) |
| FBD0 | 34 | 00 | 0 | |
| FBD3 | 37 | 24 | 36 | Ausgabe-Delimiter (s. BDOS-Aufruf 110) |
| FBD4 | 38 | 00 | 0 | Flag für parallele Druckerausgabe (z.B. mit CTRL-P unter CP/M Plus): 0=aus, 1=ein |
| FBD8 | 3C | DE | 222 | Derzeitige DMA-Adresse |
| FBD9 | 3D | 76 | 118 | |
| FBDA | 3E | 01 | 1 | Derzeitiges Bezugslaufwerk |
| FBE0 | 44 | 00 | 0 | Derzeitige Benutzernummer |
| FBE6 | 4A | 01 | 1 | BDOS Multi-Sector Count (s. BDOS-Aufruf 44) |
| FBE7 | 4B | 00 | 0 | BDOS Error Mode (s. BDOS-Aufruf 45) z.B. 255: wenn ein BDOS-Error auftritt, wird das laufende Programm normalerweise abgebrochen, bei 255 wird jedoch zum laufenden Programm zurückgekehrt (unter BASIC bei Diskettenarbeit sehr nützlich) |
| FBE8 | 4C | 00 | 0 | Suchpfad für max. 4 Laufwerke (s. SETDEF) |
| FBE9 | 4D | FF | 255 | (0=A, 1=B, 2=C,..,15=P) |
| FBEA | 4E | FF | 255 | |
| FBEB | 4F | FF | 255 | |
| FBEC | 50 | 00 | 0 | Laufwerk für temporäre Dateien |
| FBED | 51 | 00 | 0 | Laufwerk, an dem zuletzt ein Fehler auftrat |
| FBF3 | 57 | 80 | 128 | Flag für die Ausgabeart der BDOS-Fehlermeldungen 0=kurz, 128 (80H)=ausführlich |
| FBF4 | 58 | 3E | 62 | Tage seit dem 1. Januar 1978 |
| FBF5 | 59 | 0E | 14 | |
| FBF6 | 5A | 15 | 21 | Stunden im BCD-Format ( Hier: 21 Uhr, |
| FBF7 | 5B | 14 | 20 | Minuten im BCD-Format 20 Minuten, |
| FBF8 | 5C | 32 | 50 | Sekunden im BCD-Format und 50 Sekunden) |
| FBF9 | 5D | 00 | 0 | Basisadresse des Common-Speicherbereiches |
| FBFA | 5E | C0 | 192 | Hier: 0C000H |

## O U T ' e r

```
OUT 245,x   'verwandelt Bildschirm in Bitmuster
    245,91  'wandelt Bildschirm wieder zurück...
    246,x   'legt Punkt 0,0 auf Punkt 0,x
```

**Beispiel:**

```
10 PRINT CHR$(27)+"E"+CHR$(27)+"H"+CHR$(27)+"f"
20 PRINT SPC(35)"JOYCE-BILDSCHIRMTEST"
20 FOR I=0 TO 255:OUT 246,I:FOR U=1 TO 20:NEXT
40 NEXT:GOTO 20
```



```
OUT 248,1   'Kaltstart
      ,7    'Bildschirm ein
      ,8    'Bildschirm aus
      ,9    'Diskettenmotor ein
      ,10   'Diskettenmotor aus
      ,11   'Dauerpieps ein
      ,12   'Dauerpieps aus
```

## P O K E ' s ( u n t e r B A S I C )

```
POKE    28000,0     (63)              BASIC-Befehle werden im Direktmodus nicht angenommen
        8793,234    (239)             Bildschirmausgabe wird auf Drucker umgeleitet
        8793,241    (239)             Bildschirmausgabe wird unterdrückt
        &HFBF6,VAL("&H"+"STUNDEN")    Stellt die Stunden der JOYCE-UHR   | Adressen
        &HFBF7,VAL("&H"+"MINUTEN")    Stellt die Minuten der JOYCE-UHR   | im
        &HFBF8,VAL("&H"+"SEKUNDEN")   Stellt die Sekunden der JOYCE-UHR  | SCB
```

Entsprechend kann mit PRINT HEX$(PEEK(adr)) die Uhrzeit ausgelesen werden, ein spezielles Maschinenprogramm entfällt also.

## S o n s t i g e s

Verfahren, um **listgeschützte** BASIC-Programme **listbar** zu machen:

```
LOAD "DATEI              'Datei laden
OPEN "O",#1,"M:PASS.OFF" 'In M: Datei einrichten
MERGE "M:PASS.OFF"       'Datei aus M: dazuladen

LIST                     'Nun ist das Programm ungeschützt
```

Bildschirmfenster können mit **PRINT CHR$(27)+"1"** unter BASIC und mit ↑Ä1 (↑Ä=[EXIT]-Taste) unter CP/M aufgehoben werden.

**PRINT CHR$(27)+"b"+CHR$(9)** tauscht Hintergrund- und Schriftfarbe.

**?** ist gleichwertig mit **PRINT**

```
a=0:CALL a      'gleichwertig mit SYSTEM
a=&H100:CALL a  'Neustart BASIC
```

Da die BASIC-Funktion **FIND$** mit steigender Zahl an Dateien sehr langsam wird, ist es angebracht, sich selbst eine eigene Routine in Assembler zu schreiben. Ein mögliches Beispiel sieht folgendermaßen aus:

```
10 MEMORY &HF54F:RESTORE 360:FOR i=&HF550 TO &HF5DD:READ a$:POKE i,VAL("&H"+a$):NEXT
20 erst%=&HF550:dann%=&HF56C
30 DIM s(260),datei$(260):OPTION NOT TAB
40 DEF FNlocate$(x,y)=CHR$(27)+"Y"+CHR$(32+x)+CHR$(32+y)
50 PRINT CHR$(27)+"E"
60 j=0
70 PRINT CHR$(27)+"f"+FNlocate$(3,67);"Gefundene Dateien: ";
```

```
80 CALL erst%
90 WHILE PEEK(&H57)<>&HFF
100 j=j+1
110 datei$(j)=""
120 PRINT FNlocate$(3,85) USING "####";j
130 FOR i=&H3B TO &H42+3:datei$(j)=datei$(j)+CHR$(PEEK(i)):NEXT
140 CALL dann%
150 WEND
160 PRINT FNlocate$(4,67);"Sortiere Dateien ":PRINT:PRINT
170 p=1:s(1)=0:s(2)=j                                    'QUICKSORT-Verfahren
180 l=s(p):r=s(p+1):p=p-2
190 i=l:j1=r
200 d=(l+r)/2:g$=datei$(d)
210 IF datei$(i)>=g$ THEN 230
220 i=i+1:GOTO 290
230 IF datei$(j1)<=g$ THEN 250
240 j1=j1-1:GOTO 290
250 IF i>j1 THEN 290
260 SWAP datei$(i),datei$(j1)
270 i=i+1
280 j1=j1-1
290 IF i<=j1 THEN 210
300 IF i>=r THEN 320
310 p=p+2:s(p)=i:s(p+1)=r
320 r=j1
330 IF l<r THEN 190
340 IF p<>-1 THEN 180
350 FOR i=1 TO j:PRINT LEFT$(datei$(i),8)+"."+RIGHT$(datei$(i),3);" ";:NEXT:PRINT CHR$(27
)+"e":END
360 DATA 3E,FE,32,E7,FB,0E,1A,11,80,00,CD,05,00,0E,11,11
370 DATA BE,F5,CD,05,00,32,57,00,CD,7E,F5,C9,3A,57,00,FE
380 DATA FF,C8,0E,12,CD,05,00,32,57,00,CD,7E,F5,C9,3A,57
390 DATA 00,FE,00,C2,8C,F5,11,81,00,C3,B3,F5,3A,57,00,FE
400 DATA 01,C2,9A,F5,11,A1,00,C3,B3,F5,3A,57,00,FE,02,C2
410 DATA A8,F5,11,C1,00,C3,B3,F5,3A,57,00,FE,03,C2,BD,F5
420 DATA 11,E1,00,62,6B,11,3B,00,01,1a,00,ED,B0,C9,00,3F
430 DATA 3F,3F,3F,3F,3F,3F,3F,3F,3F,3F,00,00,00,00,00,00
440 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00
```

Dieses Beispielprogramm liest alle auf der Diskette befindlichen Dateinamen ein und gibt sie sortiert auf dem Bildschirm aus. Kernstück des Programm ist dabei die Assembler-Routine, die das heraussuchen der Dateinamen übernimmt. Wenn Sie nur bestimmt Dateien herausgesucht haben möchten, müssen Sie die 11 ?-Wildcards (3F's in Zeile 420/430) gegen Ihre Dateispezifikation austauschen.
Der dokumentierte Assembler-Quellcode befindet sich auf der zu dem Buch erhältlichen Diskette.



## Aufbau des Directories

Jede angefangenen 16 KByte einer Datei haben ihren eigenen Directoryeintrag auf der Diskette. Beim Standard JOYCE-Format befinden sich die Directoryeinträge auf Spur 1 Sektor 0-9. Der Eintrag ist wie folgt aufgebaut:

```
00 44 49 53 43 4B 49 54 20 43 4F 40 00 00 00 38  .DISCKIT COM...8
A5 A6 A7 A8 A9 AA AB 00 00 00 00 00 00 00 00 00  %&'()*+.........
```

Die Bedeutung der einzelnen Bytes:

**0** Kennzeichnung für den Modus der Datei. Für die Werte 00H..0FH kennzeichnet es den entsprechenden USER-Bereich der Datei (0..15), für den Wert 0E5H kennzeichnet es die Datei als gelöscht.

**1..11** Hier ist der Dateiname abgelegt. Die Bytes können beliebige Werte annehmen, es ist jedoch für den normalen Gebrauch ratsam, nur Großbuchstaben zu verwenden, da CP/M bei anderen Werten die Datei nicht laden (finden) kann.

**12** Dieses Byte kennzeichnet die Nummer des Eintrages (Extent). Jede angefangenen 16 KByte benötigen ihren eigenen Directoryeintrag. Um diese Einträge auseinanderzuhalten, wird jeder Eintrag durch das 12. Byte nummeriert.

**13..14** Für interne Zwecke reserviert

**15** Anzahl der abgelegten Records im Extent (Byte 16..31). Byte 15 kann nur Werte zwischen 00H und 7FH annehmen.

**16..31** Hier wird die Stelle auf der Diskette gekennzeichnet, an der die Datei abgelegt ist. Jedes einzelne Byte kennzeichnet einen Block, in dem ein Teil der Datei abgelegt ist.

Tips zur Verwendung: Wenn man eine Datei versehentlich gelöscht hat, muß man nur den entsprechenden Eintrag im Directory suchen und Byte 0 (vor dem Dateinamen) von E5 (gelöscht) auf 00H oder den entsprechenden USER-Wert setzen.

Durch die Verwendung von Escape-Sequenzen im Dateinamen kann man beim Auflisten des Directories zum Beispiel den Bildschirm löschen oder den Cursor an/aus schalten, wie es der Simulator TOMAHAWK tut.

## Aufbau des Directory-Labels

Das Directory-Label, das Informationen über die verwendete Diskette enthält, befindet sich wie die Directory-Einträge auf Spur 1. Pro Diskette kann nur ein Label verwendet werden, das folgendes Format hat:

```
20 42 45 52 4E 48 41 52 44 47 52 41 81 24 00 00   BERNHARDGRA.$..
6C 63 62 61 60 67 66 65 00 00 00 06 12 07 00 30   lcba gfe.......0
```

Die Bedeutung der einzelnen Bytes:

**0** Beim JOYCE ist dieses Byte zur Erkennung des Directory-Labels immer auf 20H gesetzt.

**1..11** Hier ist der Diskettenname eingetragen. Die Bytes können beliebige Werte annehmen. Es ist jedoch nicht ratsam, Bytes zu verwenden, deren Wert kleiner als 32 ist, da dies den Bildschirmaufbau unter CP/M stören könnte.

**12** Dieses Byte behinhaltet weitere Informationen über die Dateien.

Bit gesetzt:
- 0 Label existiert
- 4 Create-Timestamps aktiviert
- 5 Update-Timestamps aktiviert
- 6 Access-Timestamps aktiviert
- 7 Password-Schutz aktiviert

**13..15** Reserviert

**16..23** Diese Bytes beinhalten das Password für das Label. Da das Password verschlüsselt abgelegt ist, läßt es sich nicht einsehen. Im obigen Beispiel lautet das Password: ABCDEFGH.

**24..27** Create- oder Access-Timestamps für das Label.

**28..31** Update-Timestamps für das Label.

Tips zur Verwendung: Wenn man ein durch Password geschütztes Disketten-Label ändern möchte, braucht man nur die entsprechenden Bytes des Eintrages mit den neuen Buchstaben zu vertauschen, wobei man sich um das Password nicht zu kümmern braucht.

Um das Label komplett zu löschen, muß man Byte 0 des Eintrages auf den Wert 0E5H setzen.



## Aufbau des Datei-Labels

Im Datei-Label, das sich ebenfalls im Directory auf Spur 1 der Diskette befindet, sind weitere Informationen über eine Datei abgelegt (vollständiger Schutz gegen Zugriffe, Schutz gegen Schreiben und Schutz gegen Löschen durch Passwordabfrage). Das Datei-Label ist wie folgt aufgebaut:

```
10 42 41 53 49 43 20 20 20 43 4F 4D 80 DA 00 00   .BASIC   COM.Z..
FA FA FA 9F 99 83 95 90 00 00 00 00 00 00 00 00   zzz............
```

Die Bedeutung der einzelnen Bytes:

**0** Beim JOYCE ist dieses Byte zur Erkennung des Datei-Labels immer auf 10H gesetzt.

**1..11** Dateiname der zu diesem Label gehörigen Datei

**12** Kennzeichnung des Password-Schutzes

Bit gesetzt:
- 4 Create-Timestamping aktiviert
- 5 Delete-Modus (Schutz nur gegen Löschen)
- 6 Write-Modus (Schutz nur gegen Schreiben)
- 7 Read-Modus (vollständiger Schutz gegen Zugriffe)

**13..15** Für interne Zwecke reserviert

**16..23** Hier befindet sich das verschlüsselte Password. Im Beispiel wurde als Password **JOYCE** benutzt.

**24..31** Für interne Zwecke reserviert

Tips zur Verwendung: Wenn man den durch Password geschützten Modus einer Datei ohne Wissen des Passwords aufheben möchte, muß man Byte 0 des Eintrages auf den Wert 0E5H setzen.

Um die genannten Einträge auf Spur 1 zu ändern, benutze man am besten das Public Domain Programm **DU.COM** (Disk Utility). Es beinhaltet einen kompletten Diskettenmonitor, mit dem man bequem die Einträge der Diskette einsehen und manipulieren kann.

# D i e  " h a r t e  J o y c e - W a r e "

Wer effektiv mit seinem Joyce arbeiten will, stößt früher oder später an Grenzen, die sich softwaremäßig kaum umgehen lassen. Den Besitzern des PCW 8256 geht es ein wenig schneller so, weil ihnen ein großzügiger RAM-Speicher fehlt. Dabei gehört gerade dieser Speicher zu den Dingen, die den Joyce bei seinen Anwendern so beliebt gemacht haben. Hätte dem Joyce von Anfang an eine derartig große Palette von Hardwareerweiterungen zur Verfügung gestanden, wie es im Moment der Fall ist, hätten sich die PCW's nicht nur bei Usern beliebt gemacht, die sie in erster Linie als Textsystem benutzen wollten. Mit den folgenden Kapiteln sollen nun die Hardware-Basteleien und -Erweiterungen erklärt werden, die ein effektiveres Arbeiten mit dem Joyce ermöglichen und ein völlig neues Spektrum von Einsatzmöglichkeiten der PCW eröffnen.

## Die Erweiterung des Speichers

Anwender, die den PCW 8256 benutzen, können eigentlich gar nicht besser ca. 200,- DM in Hardware investieren, wenn sie sich die Speichererweiterung für den Joyce zulegen. Sie besteht aus 8 dynamischen NMOS Bausteinen, vorzugsweise mit der Typenbezeichnung 257. Bei Toshiba z.B. lautet die vollständige Typenbezeichnung: 41257C-15. (Der Autor arbeitet mit den billigeren D41256C-15 von Nec, wobei keine Probleme auftreten.)
Die Vorteile eines größeren Speichers liegen auf der Hand, und dies trifft erst Recht beim Joyce zu, denn Dateien, die in seinem Speicher (oder Laufwerk M:) abgelegt wurden, können wie von einer Diskette abgerufen werden. Das Laufwerk M: wird einfach als Standardlaufwerk angesprochen. (Für Neulinge: hinter das A> wird M: geschrieben und [RETURN] gedrückt.) Der erste Erfolg dieser Aktion ist, daß sich die Zugriffszeit beim Aufruf der Programme wesentlich verringert. (Bei 200 Einträgen unter Jetsam von 3.47 auf

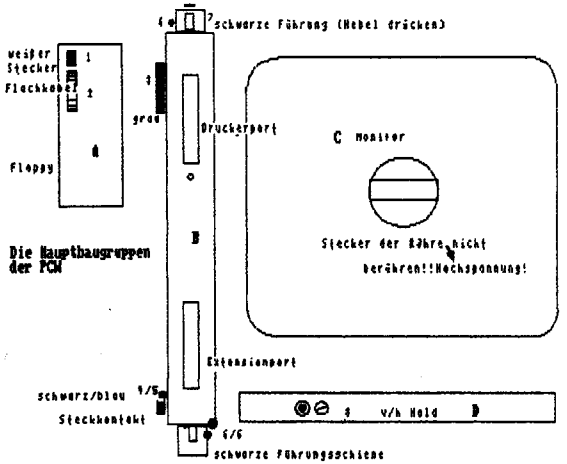


3.00 Minuten) Größere Programme, die in Laufwerk A: einen Diskettenwechsel bei der Abarbeitung notwendig machen, können komplett in M: geholt und aufgerufen werden, ohne den Anwender als "Disk-Jockey" in Anspruch zu nehmen.

Wer keine Angst davor hat, einen Schraubenzieher in die Hand zu nehmen und derartig bewaffnet seinem Joyce entgegenzutreten, kann leicht seinen RAM-Speicher mit den oben erwähnten Bausteinen aufrüsten, wenn er sich an die nachstehende Anleitung hält. Zu diesem Thema wurden zwar schon einige Informationen veröffentlicht, aber die waren teils so widersprüchlich, daß sie eher verwirrten als weiterhalfen. Einige Anwender hatten dadurch Nachteile in Kauf zu nehmen, die sich zwar gegenüber dem erweiterten Speicher gering ausnahmen, aber es war doch ärgerlich, daß ein in den Speicher gelegtes Betriebssystem durch einen Neustart mit [SHIFT]+[EXTRA]+[EXIT] nicht gelöscht werden konnte. Nun aber zur Anleitung:
Netzstecker ziehen (!) und das Tastaturkabel trennen. Nun kann Joyce mit dem "Gesicht" nach unten auf eine weiche Unterlage (Sessel, Couch etc.) gelegt werden, so daß das Standbein dem Betrachter zugekehrt und die schöne Rückfront sichtbar ist. Auf die Rückfront sind sechs Pfeile gestanzt, die auf sechs Schrauben zeigen. Zwei befinden sich am oberen Rand (lange Schrauben), je eine unter dem Printer- und Expansionsport (kleine Schrauben) und zwei am unteren Rand des Monitors (dicke Schrauben). Zum Herausdrehen dieser Schrauben erweist sich ein langer Schraubenzieher als vorteilhaft. Nach erfolgreicher Entfernung läßt sich die rückwärtige Monitorabdeckung leicht nach oben abziehen und der Blick kann ungehindert auf die Hauptbaugruppen des Joyce fallen, (s. Skizze folgende Seite) dem Laufwerk (A), der Hauptplatine und Speicherträger (B), dem Monitor (C) und seiner Steuerplatine (D).
Im folgenden müssen nun die Kabel, die die Blechdose (B) mit den anderen Baugruppen verbindet, abgezogen werden.

Die Stecker 1,2,4,5 sind in eine Plastikführung geschoben, die eine kleine Wölbung an den Steckern in sie einrasten läßt. Es ist ganz hilfreich, wenn die Plastikführung bei Zug

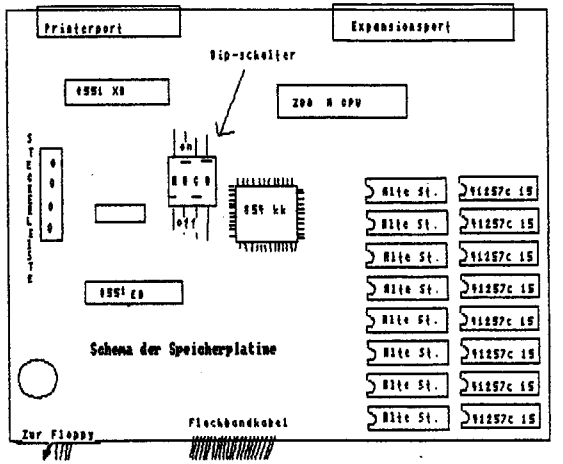
am Stecker mit einem spitzen Gegenstand leicht abgebogen wird. Der graue Stecker (3) sitzt erfahrungsgemäß sehr fest.




Es hilft hier weiter, wenn man immer im Wechsel die rechte und linke Längsseite ein wenig abhebelt. Die mit 6 gekennzeichneten drei schwarzen Massekabel werden am besten an der Mantelung des Blechkastens abgeschraubt. (Merken, in welche Löcher die Schrauben beim Einbau wieder hineinkommen!) Wurden alle Verbindungen gelöst, kann der Platinenträger nach Druck auf den schwarzen Hebel (7) an der Führung nach oben herausgezogen werden. Rund um den Rand der Blechumhüllung sitzen Schrauben, die jetzt herausgedreht werden müssen, um den Deckel abheben zu können. Ein kurzer Blick aufs Innere läßt gleich die acht in Reihe stehenden Speicherbausteine erkennen. (s. Skizze) Rechts daneben sind noch acht Steckplätze frei. Hier werden die neuerworbenen dyn. NMOS Steine eingesteckt. Die Aufschrift der Steine muß nach dem Einbau aus der gleichen Blickrichtung zu lesen sein, wie die der alten Steine. Ihre runde Einkerbung muß in die gleiche Richtung zeigen. (s. Skizze folgende Seite)
Eventuell müssen die Beinchen der neuen Bausteine noch ein



wenig zurechtgebogen werden, bevor sie in ihre Führungen passen. Auch wenn es beim Reindrücken ein wenig knirscht - sind die Beinchen einmal richtig in ihren Führungen, sollten die Bausteine auch fest angedrückt werden. (Nur nicht die Platine durchbrechen). Jetzt muß der Rechner noch so

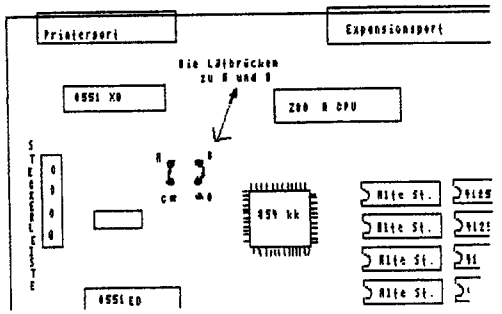



eingestellt werden, daß er auch weiß, daß ihm ein größerer Speicher zur Verfügung steht. Ungefähr in der Mitte der Platine (s.↑ auf der Skizze) befindet sich ein kleiner Schalter mit 4 Knöpfen, bei Rechnern älterer Baureihen eine Lötbrücke. Oben auf dem Schalter ist "on" zu lesen. Wird der Schlitz im Schaltknopf dorthin bewegt, ist die entsprechende Leiterbahn durchgeschaltet. Wurden alle Bausteine eingesetzt, so muß die Schalterstellung A=off B=on C=off D=on eingestellt sein, damit sich der Rechner mit 512 KB meldet.

| A | B | C | D | |
|---|---|---|---|---|
| off | on | off | on | = 512 KB |
| on | off | off | on | = 256 KB |
| off | on | on | off | = 128 KB |
| on | off | on | off | = 128 KB |

Diese Angaben nur zur Information, was bei anderen Schalterstellungen geschehen würde. Also, wenn schon Erweiterung,

dann auf 512 KB! Bei den Lötbrücken läuft es analog. Hier sind 4 Lötpunkte mit den Buchstaben A-D gekennzeichnet. Dazwischen liegen zwei Punkte, die Brücken zu B und C schlagen. Die Verbindung vom Mittelpunkt zu B wird jetzt getrennt (durchschneiden) und dafür nach A gelegt und angelötet (s. Skizze).




Nach erfolgter Operation kann der Umbau in umgekehrter Reihenfolge vonstatten gehen. Beim Zusammenbau der Blechkiste sollte man darauf achten, welche Löcher für die Masseanschlüsse frei bleiben sollten.
Wer jetzt allerdings noch ein wenig weiter an seiner Hardware basteln will, der kann die Rückwand seines Joyce gleich auflassen. Es wäre zum Beispiel nützlich, wenn man über den Helligkeitsregeler den Monitor sowohl ganz hell (gut für Lightpen) oder ganz dunkel (gut für Lebensdauer und Augen) schalten könnte. Zum entsprechenden Einbau eines 10K Helligkeitspotis mit 1,8K Vorwiderstand (auswechseln) sei auf den Artikel des Joyce Sonderheftes Nr. 2 (DMV-Verlag) verwiesen.



**Das Zweitlaufwerk**

Ein flüchtiger Blick auf die "Innereien" des Joyce zeigt, daß der Einbau des Zweitlaufwerks keine Schwierigkeit darstellt. Die Kabel, die zum Standardlaufwerk führen, sind zweifach vorhanden, wobei die zweite Ausführung nirgends angeschlossen ist. Diese "toten" Leitungen werden an ein Zweitlaufwerk gesteckt, wobei die Steckerformung keine falschen Anschlüsse zuläßt. An dieser Stelle gilt es jedoch zu erwägen, für welche Art von Zweitlaufwerk man sich entscheidet. Ästheten unter den Joycern werden wohl dazu tendieren, daß dem Design des Joyce angepaßte Original-Zweitlaufwerk einzubauen. Nach Entfernung der Abdeckklappe unter dem Standardlaufwerk kann es leicht von hinten in den freigewordenen Schacht gesetzt werden.
Praktiker haben die Qual der Wahl. Sie können zwischen einem 5¼- und einem 3½-Zoll Laufwerk wählen. Hier müssen jedoch die Kabel aus dem Joyce nach außen geführt werden, woran die Laufwerke dann im Freien hängend "brummen". Entscheidungen für das ein oder andere Laufwerk sind schwer zu fällen. Hier können die Zahlen für sich sprechen:
Original FD-2 3"-Laufwerk: 2x80 Spuren; 1MB unformatiert, 706 KB formatiert. Preis: ca. 350,-DM.
TEAK 3½" 1MB unformatiert, 706KB formatiert,
Preis: ca. 340,- DM.
TEAK 5¼" 1MB unformatiert, 706KB formatiert,
Preis: ca. 435,- DM.
Wahlweise kann bei diesem Laufwerk zwischen 40 und 80 Tracks umgeschaltet werden. Außerdem gibt es ein Programm dazu, welches beliebige Datenfiles (z.B. ASCII, Turbo Pascal, DBase, Wordstar) **von CP/M- auf MsDos-Rechnerformat** (und umgekehrt) überträgt. Bei der Zusammenstellung dürfen die Diskettenpreise nicht außer Acht gelassen werden! So kosten:

10 St. im 3 Zoll Format mindestens 59,- DM;
10 St. im 3½ " " " 28,- DM;
10 St. im 5¼ " " " 10,- DM (no name).

Anhand dieser knappen Zusammenstellung mag sich jeder Anwender selbst ein Bild machen, ob überhaupt und wenn ja, welche Anschaffung eines Laufwerks sich lohnt.

## Die Reinigung des Druckkopfes

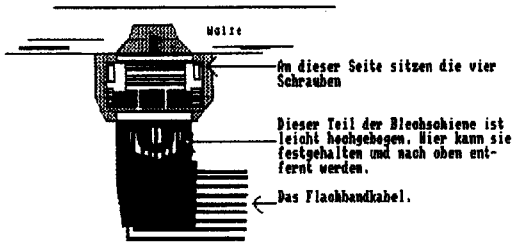
Eines der empfindlichsten Teile des Joyce ist der Drucker. Dies liegt daran, daß er, ebenso wie Tastatur und Laufwerk, mechanischen Beanspruchungen unterworfen ist. Wenn er trotzdem klaglos seine Arbeit verrichtet, liegt das an seiner grundsoliden Konstruktion. Dort, wo es darauf ankommt, wurde nicht am Material gespart. (z.B. Messinghülsen an der Laufschiene des Druckkopfes etc.) Bis auf die Tatsache, daß er ab und an - je nach Beanspruchung - ein neues Farbband braucht, arbeitet er recht wartungsfrei.
Dennoch, nach einer Zeit von etwa hundert Betriebsstunden kann es sein, daß bei Grafikausdrucken mit dunklen Flächen schwarze Streifen auf dem Papier erscheinen, wo sie nichts zu suchen haben. In Extremfällen kann dies auch beim normalen Schreiben geschehen. Dann hilft alles nichts mehr, entweder man gibt den Drucker zur Wartung in eine Spezialwerkstatt, oder man macht sich selbst die Finger schmutzig, denn dann ist der Druckkopf verunreinigt und muß auseinander genommen werden. Dazu folgende Anleitung:

Im Druckkopf sitzen neun kleine Nadeln, die von Elektromagneten ans Farbband gedrückt werden und einen Punkt aufs Papier schlagen. Diese Nadeln können von Farb- und Farbbandresten so verkleben, daß sie nicht mehr schnell genug in ihre Ruheposition zurückkehren können und beim Weiterlauf des Kopfes Schlieren aufs Papier ziehen. Entfernt man Abdeckklappen und Farbband, so erkennt man unter der Stelle, wo das Farbband gesessen hat, ein Flachbandkabel, das in den Druckkopf läuft. Bevor es in diesen hineingelangt, muß es unter einer Blechklammer hindurch. Diese Blechklammer (sie hält den Druckkopf) hat eine kleine, nach oben gebogene Nase. An dieser Nase wird die Klammer mit einer entsprechend kleinen Zange nach oben hin abgezogen. Der Druckkopf kann jetzt ein wenig von der Papierrolle weggezogen und nach oben aus seiner Halterung geholt werden.
Um das Flachbandkabel nicht zu beschädigen - es bleibt am Druckkopf - muß desweiteren entsprechend vorsichtig mit dem



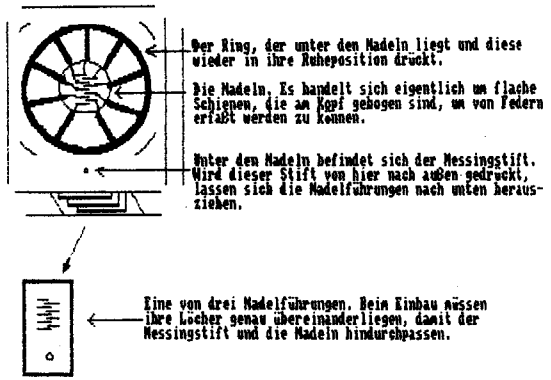
Kopf umgegangen werden. Von der Papierrollenseite (vorne) werden jetzt vier Schrauben am Kopf sichtbar. Diese Schrauben werden (ohne rohe Gewaltanwendung - sonst verdreht sich die Magnetanordnung und muß durch Probieren neu justiert werden) herausgedreht, wobei im Endstadium die Schrauben nach unten zeigen sollten, damit die Nadeln nicht aus dem Druckkopf fallen. Jetzt kann die hintere Platte mit der Aufschrift "HOT" entfernt werden, und das schwarze vordere Plastikteil vom inneren Eisenteil getrennt werden. Im Endeffekt hat man den magnetischen Kern, in den das Flachbandkabel hineinläuft, und den Nadelträger in der Hand. Das Teil am Kabel kann am Drucker bleiben, der Rest muß gesäubert werden.
Man sollte sich jetzt genau die Lage der Zwischenringe merken, sie müssen in der gleichen Reihenfolge wieder hinein (am besten der Reihe nach auf ein Blatt Papier in eine sichere Ecke legen). Einer dieser Ringe kann nicht entfernt werden, ohne die Nadeln aus der Führung zu ziehen. Dieser Ring sorgt dafür, daß die Nadeln wieder in ihre Ruhestellung zurückgedrückt werden. Also müssen zuerst die Nadeln raus.
Mit einer Pinzette kann man die Nadeln ganz leicht von innen aus ihren Führungen herausziehen. Dabei muß man sich unbedingt ihre Lage merken, denn sie müssen genauso wieder hinein. Am besten man bereitet ein Stück Styropor oder Karton mit Zahlen von 1-9 vor, in das die Nadeln ihrer Lage entsprechend hineingesteckt werden. Danach kann auch der Ring, der sie zurückdrückt, abgenommen werden. Im unteren Teil der Plastikhalterung befindet sich ein kleiner Stift aus Messing. Dieser kann von innen nach außen herausgedrückt werden. Dieser Stift hält drei Führungen für die Nadeln. Sie lassen sich danach nach unten herausziehen. (Auch ihre Lage muß gemerkt werden!)
Alle Teile können nun mit Benzin gereinigt werden. Für die drei Nadelführungen eignet sich am besten eine in Benzin getauchte Zahnbürste.
Nach der Reinigung werden die Teile in umgekehrter Reihenfolge wieder zusammengesetzt. Die vier Schrauben des Kopfes sollten zwar fest, jedoch nicht allzu stramm

angezogen werden. Vor Anbringen des Druckkopfes sollte die Walze gründlich mit einem dafür vorgesehenen Reinigungsmittel gesäubert werden (sonst Spiritus). Danach steht einem Probedruck nichts mehr im Wege.




Der Druckkopf von oben gesehen

Schematische Darstellung des Druckkopfes von hinten/innen

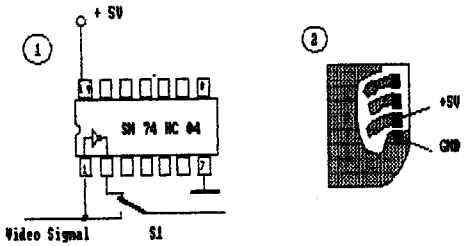





## Ein Bildschirminverter

Nach Einbau entsprechender Teile ist es möglich, den Bildschirm unabhängig vom gerade laufenden Programm durch Umlegen eines Schalters in seinen Darstellungsfarben umzukehren (zu invertieren). Ein Invertieren wäre etwa für das Mica-CAD Programm (Mica ist hervorragend geeignet für wissenschaftliche Zeichnungen) sehr nützlich, da Mica einen softwaremäßig umgeschalteten Bildschirm wieder zurücksetzt. Überhaupt ist der Umschalter bei all den Programmen gut einzusetzen, bei denen es auf Darstellungsschärfe am Bildschirm ankommt. Die Funktionsweise des Inverters ist denkbar einfach:
Ein Bildschirm-(Video-)Signal besteht aus zwei Pegeln: entweder low (0 V) oder high (+5 V). Steht der Pegel auf high, wird ein Punkt auf den Bildschirm gesetzt, andernfalls nicht. Jetzt liegt der Gedanke nahe, die Pegel zu vertauschen. Dies wird durch Verwendung des TTL-Bausteins SN 74 HC 04, der aus sechs Invertern besteht (einer wird nur benutzt), erreicht. Wird ein Umschalter mit eingebaut, (s. Skizze folgende Seite) kann man unabhängig von der gerade laufenden Software zwischen hellem oder dunklem Hintergrund wählen. Der Einbau ist einfach durchzuführen, wenn man nicht gerade mit dem Lötkolben auf Kriegsfuß steht.
Stückliste: 1 SN 74 HC 04 / 1 Sockel 14-polig. / mind. 50 cm Kabel 4-adrig / 1 Schalter 1 * um / Schrumpfschlauch oder Isolierband.
Netzstecker des Gerätes ziehen und das Video-Kabel kappen, um das IC dazwischen zu setzen. Das Viedeo-Kabel verläuft vom Blechkasten B (s. Skizze S. 265) zur Monitorplatine D. Es ist orange und kommt (als 3./vorletztes) Kabel aus der blauen Steckverbindung der Hauptplatine.
Die vom Blechkasten B kommende Video-Leitung wird mit Pin 1 des Sockels verbunden, Pin 2 über den Umschalter mit dem weiterführenden Kabel zur Monitorplatine (D). Der andere Pol des Umschalters wird an Pin 1 des Sockels gelötet. Die Plusleitung der Tastaturplatine (dort wird die Tastatur angeschlossen) (s. Skizze 2) muß an Pin 14, die Minusleitung an

Pin 7 gelötet werden. Wurden Lötstellen und Kontakte ausreichend gegen Kurzschlüsse isoliert, kann das IC in den Sockel gesteckt werden. (Auf richtige Polung achten!)




Zum Schluß kommt es nur noch auf eine entsprechende Anbringung des Umschalters an. Eine besonders dekorative Art ist, ihn in den Standfuß des Monitors zu setzen. Der Aufwand dafür ist jedoch recht hoch, da viel gebohrt werden muß. Einfacher ist es, das Kabel durch irgendeinen Lüftungsschlitz nach außen zu führen. Ist dies geschehen, steht dem Zusammenbau des Gehäuses - oder falls man weiterbasteln will - einem Testlauf nichts mehr im Weg.

Wenn es zu **Fehlfunktionen nach Einbauten** kommt...

Ein besonders häufiger Fehler ist eine sogenannte kalte Lötstelle, bei der das Lötzinn nicht korrekt verlaufen ist. Sämtliche Lötverbindungen sind hierauf zu prüfen (sehen grau, rauh und runzelig aus) und eventuell nachzulöten.

Der Bildschirm bleibt ständig hell:

Überprüfen Sie die Pegel des IC 2001 (SN 74 HC 00) auf der Monitorplatine links hinten und die richtige Anbringung des Umschalters nach dem IC SN 74 HC 04.

Nichts geht mehr:

Überprüfen Sie sämtliche Verbindungen und Lötstellen auf Kurzschluß oder Lötbrücken. Außerdem sind die Spannungen und Sicherungs-IC's P501 und P502 zu überprüfen (+5 V und +12 V). Sollten die Sicherungs-IC's hochohmig sein, reicht eine einfache Überbrückung.



## Der Expansions-Port

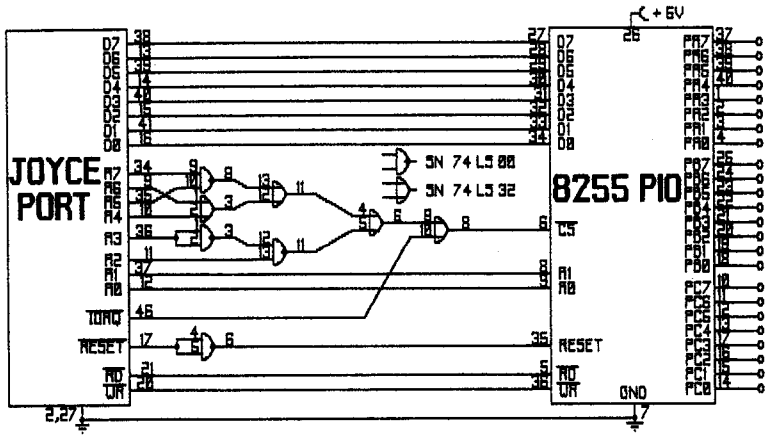
Über den Expansions-Port lassen sich vielfältige Aufgaben bewältigen. Für die meisten Joyce-User ist er allerdings nur als Steckplatz für die CPS 8256 (zur Schnittstelle später mehr) in Erscheinung getreten.

Um den Hardware-Freaks Gelegenheit zu geben, eigene Experimente anzustellen, kann hier die Belegung der einzelnen Ports nachgesehen werden.

| Pin | Belegung | Belegung | Pin |
|---|---|---|---|
| 1 | nc | nc | 26 |
| 2 | GND | GND | 27 |
| 3 | +5 V | +5 V | 28 |
| 4 | nc | +12 V | 29 |
| 5 | A14 | A15 | 30 |
| 6 | A12 | A13 | 31 |
| 7 | A10 | A11 | 32 |
| 8 | A8 | A9 | 33 |
| 9 | A6 | A7 | 34 |
| 10 | A4 | A5 | 35 |
| 11 | A2 | A3 | 36 |
| 12 | A0 | A1 | 37 |
| 13 | D6 | D7 | 38 |
| 14 | D4 | D5 | 39 |
| 15 | D2 | D3 | 40 |
| 16 | D0 | D1 | 41 |
| 17 | /Reset | /M1 | 42 |
| 18 | /BUSRQ | /INT | 43 |
| 19 | /BUSAK | /WAIT | 44 |
| 20 | /WR | /MREQ | 45 |
| 21 | /RD | /IORQ | 46 |
| 22 | nc | NSYNC | 47 |
| 23 | /MIDS | VIDEO | 48 |
| 24 | /32 MHZ | /4MHZ (Z80A Takt) | 49 |
| 25 | GND | GND | 50 |

Wem z.B. die Anschaffung der CPS8256 zu teuer erscheint, kann sich mit Hilfe dieses Belegungsplans und der Anleitung auf den folgenden Seiten seine **Schnittstelle** selbst bauen. Hier wird gezeigt, wie drei parallele Ausgänge durchzuschleifen sind.

Kernstück einer aus drei mal acht Bit bestehenden Schnittstelle ist der Ein / Ausgabebaustein 8255. Er verfügt über die angesprochenen 3 E/A-Ports zu je 8 Bit, die in ihrer Funktion auf verschiedene Weise programmiert werden können. Im folgenden soll nur auf die Betriebsart 0 (Standard-Ein/ Ausgabe) eingegangen werden.




Wer sich obigen Schaltplan nicht selbst verdrahten will, der kann entweder auf die zur Ätzung der fertigen Platinenlayouts (zweiseitig!!) zurückgreifen, die sich im Anhang des Buches befinden, oder sich, wie bereits besprochen, seine geätzten Platinen bei der Firma Joyce-Platinenservice bestellen. Diese Platinen sind bereits fertig durchkontaktiert, gebohrt, verzinnt und mit Bestückungsaufdruck



versehen.
Diese Schnittstelle wird über den Port 0A7H programmiert. Sie erwartet einen aus 8 Bit bestehenden Wert, der nach folgender Tabelle kodiert wird:

D0: Port C, untere 4 Bits:
1 = Eingang
0 = Ausgang

D1: Port B:
1 = Eingang
0 = Ausgang

D2: Betriebsartdefinition für Port C (untere 4 Bits) und Port B:
0 = Betriebsart 0 (Standard-Ein/Ausgabe)
1 = Betriebsart 1 (getaktete Ein/Ausgabe)

D3: Port C, obere 4 Bits:
1 = Eingang
0 = Ausgang

D4: Port A:
1 = Eingang
0 = Ausgang

D5, D6: Betriebsartdefinition für Port C (obere 4 Bits) und Port A:
00 = Betriebsart 0 (Standard-Ein/Ausgabe)
01 = Betriebsart 1 (getaktete Ein/Ausgabe)
1X = Betriebsart 2 (getaktete bidirektionale Bus-Ein/Ausgabe)

D7: Kennbit für Betriebsart definieren"
1 = aktiv

Um eine neue Betriebsart einzustellen, muß mindestens Bit 7 (D7) zusammen mit den gewünschten Bits gesetzt werden. Falls die Schnittstelle 3 8-Bit Ausgabekanäle erhalten soll, muß der Wert 128 (80H) an die Adresse 0A7H übergeben werden.

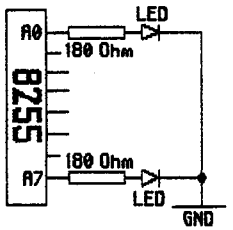
Die Portadressen für die Programmierung sind folgende:

0A4H: Port A
0A5H: Port B
0A6H: Port C
0A7H: Steuerport (zum Definieren)

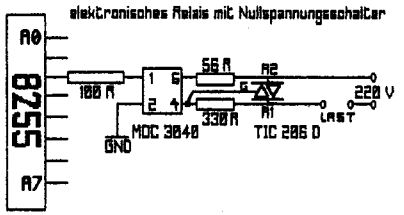
Das Beispiel

```
OUT &HA7,&H80:OUT &HA4,&HFF
```

setzt alle Kanäle auf Ausgabe und gibt an Port A den Wert 255. Um die Datenbits sichtbar zu machen, bieten sich verschiedene Möglichkeiten an:




Beschaltung mit LED




Beschaltung mit Relais

Es ist natürlich auch möglich, über die Schnittstelle Daten einzulesen. Dazu muß jedoch die Schnittstelle als Eingabemedium definiert sein. Ein möglicher Initialisierungscode könnte zum Beispiel lauten:

```
OUT &HA7,&H89
```

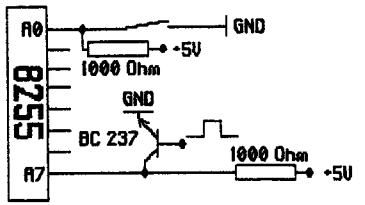
Dieses Beispiel definiert Port A und B als Ausgang, Port C als Eingang.
Mit:

```
PRINT INP(&HA6)
```

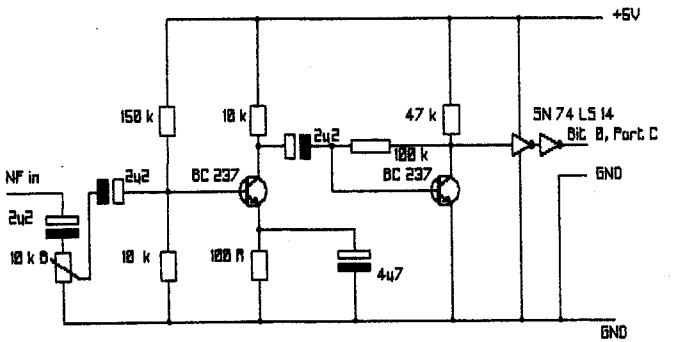


wird der aktuelle Wert an Port C auf dem Bildschirm ausgegeben. Es ist dabei zu beachten, daß die Werte über Pull-Up Widerstände angelegt werden (s. Schaltplan), um Datenfehler zu vermeiden:

Eingabe mit Taster oder Impuls




Ein wirkungsvolles Anwendungsbeispiel für Ein- und Ausgabe über die Schnittstelle ist die Ansteuerung einer musikgesteuerten **8-Kanal Lightshow**. Eine solche Steuerung läßt sich am besten nach folgender Schaltung aufbauen:




Die Ansteuerung der Lichterkette kann wie auf Seite 272 dargestellt erfolgen.

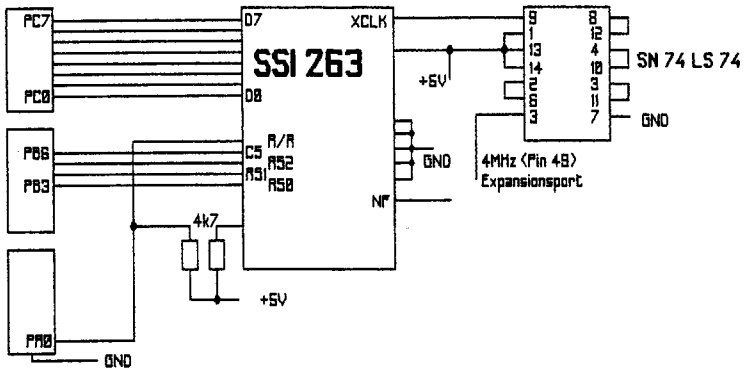
Das Programm basiert darauf, daß die Schaltung die NF der Musik-Anlage in eine positive Spannung umsetzt. Die Tatsache, daß ein Beat (Tief-Ton) eine größere Spannung als hochfrequente NF erzeugt, wird von dem Logikbaustein SN 74 LS 14 so ausgenutzt, daß er einen Beat in einen Impuls umsetzt, der von dem folgenden Programm verwertet werden kann:

```
10 OUT &HA7,&H89                    'Port A, B: Ausgabe;   Port C: Eingabe
20 PRINT CHR$(27)+"E"+CHR$(27)+"f"
30 RESTORE 300
40 READ wert$                       'Lichtmuster auslesen
50 IF wert$="-1" THEN 30            'Lichtmustertabelle zu Ende
60 wert=VAL("&H"+wert$)
70 IF (INP(&HA6) AND 1)<>1 THEN 70 'Solange Zeile wiederholen, bis Bit 1 gesetzt
80 OUT &HA4,wert                    'Programmiertes Lichtmuster  Port A
90 OUT &HA5,INT (RND*256)           'Zufallslichtmuster          Port B
100 FOR i=0 TO 95:NEXT              'Warteschleife
110 GOTO 40
297 '
298 ' Ab Zeile 300 stehen die programmierten Lichtmuster (Bit gesetzt = Lampe an)
299 '
300 DATA 80,C0,E0,F0,F8,FC,FE,FF,FE,FC,F8,F0,E0,C0,80,00,80,C0,E0,F0,F8,FC,FE,FF,FE
310 DATA FC,F8,F0,E0,C0,80,00,80,C0,E0,F0,F8,FC,FE,FF,FE,FC,F8,F0,E0,C0,80,00,80,C0
320 DATA E0,F0,F8,FC,FE,FF,FE,FC,F8,F0,E0,C0,80,00,80,C0,E0,F0,F8,FC,FE,FF,FE,FC,F8
330 DATA F0,E0,C0,80,00,80,C0,E0,F0,F8,FC,FE,FF,FE,FC,F8,F0,E0,C0,80,01,03,07,0F,1F
340 DATA 3F,7F,FF,7F,3F,1F,0F,07,03,01,00,01,03,07,0F,1F,3F,7F,FF,7F,3F,1F,0F,07,03
350 DATA 01,00,01,03,07,0F,1F,3F,7F,FF,7F,3F,1F,0F,07,03,01,03,07,0F,1F,3F,7F,FF,7F
360 DATA 3F,1F,0F,07,03,01,00,01,03,07,0F,1F,3F,7F,FF,7F,3F,1F,0F,07,03,01,00,01,03
370 DATA 07,0F,1F,3F,7F,FF,7F,3F,1F,0F,07,03,06,0C,18,30,60,C0,81,03,06,0C,18,30,60
380 DATA C0,81,03,06,0C,18,30,60,C0,81,03,06,0C,18,30,60,C0,81,03,06,0C,18,30,60,C0
390 DATA 81,03,06,0C,18,30,60,C0,81,03,06,0C,18,30,60,C0,81,03,06,0C,18,30,60,C0,81
400 DATA 82,84,88,90,A0,C0,C1,C2,C4,C8,D0,E0,E1,E2,E4,E8,F0,F1,F2,F4,F8,F9,FA,FC,FD
410 DATA FE,FF,81,82,84,88,90,A0,C0,C1,C2,C4,C8,D0,E0,E1,E2,E4,E8,F0,F1,F2,F4,F8,F9
420 DATA FA,FC,FD,FE,FF,81,82,84,88,90,A0,C0,C1,C2,C4,C8,D0,E0,E1,E2,E4,E8,F0,F1,F2
430 DATA F4,F8,F9,FA,FC,FD,FE,FF,81,42,24,18,FF,00,FF,18,24,42,81,42,24,18,FF,00,FF
440 DATA 18,24,42,81,42,24,18,FF,00,FF,18,24,42,81,42,24,18,FF,00,FF,18,24,42,81,42
450 DATA 24,18,FF,00,FF,18,24,42,81,42,24,18,FF,00,FF,18,24,42,81,42,24,18,FF,00,FF
460 DATA 18,24,42,81,42,24,18,FF,00,FF,18,24,42,81,AA,55,AA,55,AA,55,AA,55,AA,55,AA
470 DATA 55,AA,55,AA,55,AA,55,AA,55,AA,55,AA,55,AA,55,AA,55,AA,55,AA,55,CC,33,CC,33
480 DATA CC,33,CC,33,CC,33,CC,33,CC,33,CC,33,CC,33,CC,33,CC,33,CC,33,CC,33,CC,33,CC
490 DATA 33,CC,33,CC,C3,3C,C3,3C,C3,3C,C3,3C,C3,3C,C3,3C,C3,3C,C3,3C,C3,3C,C3,3C,C3
500 DATA 3C,18,3C,7E,FF,7E,3C,18,00,18,3C,7E,FF,7E,3C,18,00,18,3C,7E,FF,7E,3C,18,-1
```



**Ein Sprachsynthesizer**

Mit dem bisherigen Wissen über die Schnittstelle läßt sich dem Joyce das Reden beibringen. Eine entsprechende Schaltung zu diesem Zweck ist weiter nicht schwierig aufzubauen, wenn man die hauseigene HiFi-Anlage endlich einmal sinnbringend als NF-Verstärker einsetzt. (s. Skizze)




An dieser Stelle wollen wir der Phantasie der Anwender keine Grenzen setzen. Lediglich zur Anregung folgendes kleines Listing. Erste Probeschritte können durch Austausch der Dateneinträge in Zeile 140 in Abstimmung mit dem Phonem-inventar vorgenommen werden. Die Datazeile wird durch die Werte 0 (Pause) und -1 (Ende Liste) abgeschlossen.

```
40 OUT &HA7,&H90                'Port A Eingabe, Port B und C Ausgabe
50 dac=&HA6:dab=&HA5
60 c=128:b=24:GOSUB 160         'Initialisierung des SSI 263
70 c=128:b=0:GOSUB 160          '(siehe CPC-International
80 c=95:b=24:GOSUB 160          ' Sonderheft 7-88/89
90 c=50:b=32:GOSUB 150          ' S. 121 ff)
100 c=105:b=16:GOSUB 150        '
110 c=40:b=8:GOSUB 150          '
120 READ a$:IF a$="-1" THEN 130 ELSE c=VAL("&H"+a$):b=0:GOSUB 150:GOTO 120
130 b=0:GOSUB 150:END
140 DATA 25,7,2a,30,0,7,30,28,0,0,30,27,7,7,28,32,2e,7,2a,0,-1 :'Phonemliste / 0,-1=Ende
150 IF INP(&HA4)<>0 THEN 150    'warten, bis SSI 263 fertig
160 OUT dac,c:OUT dab,b OR 64:OUT dab,b AND 56:RETURN
```

Der Sprachchip SSI 263 besitzt ein Phoneminventar, das nach entsprechender Initialisierung ausgegeben wird. Im folgenden nun die Phonemliste mit den dazugehörigen Codes. Bei der Anwendung sollte beachtet werden, daß die Laute der englischen Sprache angepaßt wurden. Bei der Programmierung von Lautfolgen sollte der Anwender in einer Laut- und nicht in einer Schriftsprache konstruieren.: - hier = hia -

Das Phoneminventar:

| Hex-Code | Symbol | Beispielwort (in Englisch) | Hex-Code | Symbol | Beispielwort (in Englisch) |
|---|---|---|---|---|---|
| 00 | PA | (Pause) | 20 | L | lift |
| 01 | E | meet | 21 | L1 | play |
| 02 | E1 | bent | 22 | LF | fall (End-L) |
| 03 | Y | before | 23 | W | water |
| 04 | Y1 | year | 24 | B | bag |
| 05 | AY | please | 25 | D | paid |
| 06 | IE | any | 26 | KV | tag |
| 07 | I | six | 27 | P | pencil |
| 08 | A | made | 28 | T | tart |
| 09 | AI | care | 29 | K | kit |
| 0A | EH | nest | 2A | HV | (Vokal halten) |
| 0B | EH1 | belt | 2B | HVC | d(h)oubt |
| 0C | AE | dad | 2C | HF | heart |
| 0D | AE1 | after | 2D | HFC | p(h)ound |
| 0E | AH | got | 2E | HN | (Nasal halten) |
| 0F | AH1 | father | 2F | Z | zero |
| 10 | AW | office | 30 | S | same |
| 11 | O | store | 31 | J | pleasure |
| 12 | OU | boat | 32 | SCH | sheep |
| 13 | OO | look | 33 | V | very |
| 14 | IU | you | 34 | F | five |
| 15 | IU1 | could | 35 | THV | there |
| 16 | U | tune | 36 | TH | with |
| 17 | U1 | moon | 37 | M | most |
| 18 | UH | wonder | 38 | N | nine |
| 19 | UH1 | love | 39 | NG | song |
| 1A | UH2 | what | 3A | :A | Mädchen (*) |
| 1B | UH3 | hut | 3B | :OU | Möwe (*) |
| 1C | ER | bird | 3C | :U | Tür (*) |
| 1D | R | roof | 3D | :UH | menu (+) |
| 1E | R1 | rug | 3E | E2 | Bitte (*) |
| 1F | R2 | Kutter.(*) | 3F | LB | il (+) |

(*) bedeutet, daß das Beispielwort aus dem Deutschen kommt; (+), daß es sich um ein französiches Wort handelt.

Wer sich noch ein wenig mehr über den Sprachsynthesizer informieren möchte, der sei auf den Artikel des CPC-International Sonderheftes 7-88/89 S. 121 ff verwiesen.
Der dort vorgeführte "Laberkasten" wurde an unsere Schnittstelle angepaßt.



# Erweiterungen zur Joyce-Hardware

Zur Hardware läßt sich nicht nur das zählen, was die elektrischen Eigenschaften des Joyce unterstützt. So kommt es, daß an dieser Stelle auch drei Dinge Erwähnung finden, die - ohne Produktwerbung betreiben zu wollen - sich als ausgezeichnete Hilfsmittel bei der Arbeit mit dem Joyce erwiesen haben.
Wer lange am Bildschirm arbeiten muß, wird es über kurz oder lang als äußerst lästig empfinden, daß sich helle Gegenstände, Lampen oder Fenster im Bildschirm spiegeln. Speziell für den Joyce wurde ein **Bildschirmfilter** entwickelt, der sich formvollendet dem Design des Gehäuses anpaßt und mit Klettverschlüssen, die unsichtbar unter dem Rahmen des Bildschirmfilters angebracht sind, am Monitor befestigt wird. Dieser Filter - er besteht im wesentlichen aus einer schwarzen Gaze - fängt unerwünschte Reflexionen auf, läßt einen eventuell leicht flackernden Monitor ruhiger erscheinen und die matte, grünliche Aura, die die Zeichen am Monitor begleitet, leuchtet nicht mehr so schnell durch. Bei diesem Bildschirmfilter handelt es sich um ein nützliches Utensil, daß jedem Anwender, der Augen und Nerven schonen möchte, nur empfohlen werden kann.
Bei der zweiten nützlichen Erweiterung handelt es sich um einen **Einzelblatteinzug** für den Drucker, den der DMV-Verlag in Eschwege zum Preis von DM 29.95 anbietet. Der Vorteil gegenüber anderen Einzügen besteht darin, daß stufenlos auf verschiedene Blätter von 45 mm bis 260 mm eingestellt werden kann (bei Rastereinstellungen haken oft die Blätter und fallen nicht von allein auf die Walze), und daß er vom Design her dem Joyce-Drucker angepaßt ist. Der Einzug wird einfach auf den Drucker gesteckt. Die für den Drucker mitgelieferten zwei schwarzen Papierhalter können an den Einzelblatteinzug angeclippt werden. Unter Verwendung einer Einzugsführung wird selbst bei einer größeren Anzahl von Druckseiten die Seitenzahl immer an genau der gleichen Stelle erscheinen und auf Grund der rasterlosen Einstell-

möglichkeit (fast jedes Blatt ist bei geforderter Genauigkeit verschieden) gelingen Doppeldrucke eines Blattes (dazu später mehr) auf den Pixel genau.
Eine weitere wichtige Neuerung auf dem Hardwaremarkt sind **Farbbänder** für den Joyce, wobei "Farb"- jetzt wörtlich zu verstehen ist. Mittlerweile gibt es eine Vielzahl von Firmen, die sich von der Angebotspalette her auf den Joyce konzentrieren und auch immer ein wachsames Auge auf den Englischen Markt werfen, um interessante Neuerungen gleich parat zu haben. Unter anderem bieten sie Farbbänder für den Joyce-Drucker in z.B. rot, blau, grün oder braun an. Der Preis für diese Farbbänder, (die auch leicht selbst nachgefärbt werden können) liegt bei ca. 24,- DM. Textverarbeitungssysteme, die von den Druckbefehlen her eine Druckunterbrechung zulassen (Prowort z.B. könnte während des Druckvorgangs den Drucker stoppen und die Meldung "Farbband wechseln!" auf den Monitor bringen) könnten Texte jetzt sogar farbig markieren. Jedoch lassen sich auch Grafiken produzieren, deren Hauptaussagewert sich durch ihre Farbigkeit ergibt! (s. Hardwareerweiterungen zur Grafik)
Im Folgenden möchte ich kurz auf Erweiterungen eingehen, die die elektrischen Eigenschaften des Joyce in effektivem Maße verbessern. Hier muß zuerst

**Die Schnittstelle CPS 8256**

Erwähnung finden, denn am Anfang jeder Hardwareerweiterung steht eine passende Schnittstelle, über die die Vermittlung zwischen Joyce und seiner Außenwelt vonstatten geht. Wer mit Fremddruckern, Akustikkopplern, Modems, anderen Terminals oder dergleichen arbeiten will, ist mit der CPS8256 Schnittstelle von Schneider gut beraten. Sie stellt einen seriellen RS232C und einen parallelen (Centronics) Bus zur Verfügung. Im Handel ist diese Schnittstelle für ca. 180,- DM erhältlich. Der erste Vorteil dieser Schnittstelle besteht darin, daß sie sich optimal dem Design des Joyce anpaßt. Sie wird mit zwei Schrauben solide am Computergehäuse befestigt und führt Anschlüsse nach rechts zur Seite weg. Zum Lieferumfang dieser Schnittstelle gehört ein kleines Handbuch, das allerdings entsprechend der komplexen Technik ein wenig



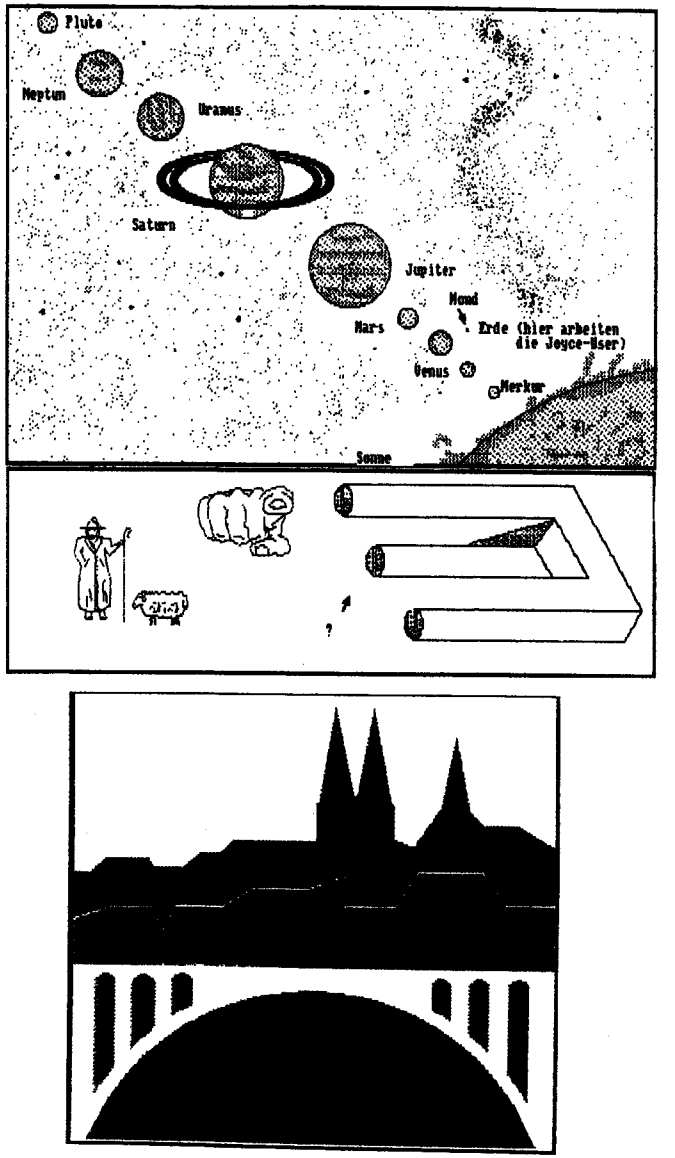
umfangreicher hätte ausfallen dürfen. Die Schnittstelle kann unter CP/M u.a. mit **device, setsio** oder **pip**, welches ganze Dateien an die Schnittstelle sendet, angesprochen werden.
Auf Seite eins der Systemdisketten befindet sich unter dem Modus "versteckt" (LocoScript) das Programm MAIL232, das unter diesem Namen von CP/M aus gestartet werden kann. Dieses Programm arbeitet klaglos mit der CPS8256 zusammen, und gestattet auch, die Daten-Fernübertragung (DFÜ) via Akustikkoppler oder Modem mit dem Joyce vorzunehmen (s. unten). Über diese Schnittstelle können auch mehrere Rechner im Zusammenhang mit Festplatten und/oder leistungsfähigen Druckern zu einem mächtigen Verbund zusammengeschlossen werden, der selbst Geschäftsabwicklungen größerer Firmen über mehrere Bildschirmarbeitsplätze hinweg koordinieren und durchführen kann. Aber auch der Anwender, der mit seinem PCW nicht so hoch hinaus möchte und ihn im kleineren Rahmen für Grafik und Text benutzen möchte, ist mit der CPS8256 gut beraten. Bis auf die der englischen Steckernorm entsprechende Hardware laufen sämtliche Hardwareerweiterungen über diese Schnittstelle. So auch die:

**Gerdes-mouse.**

Dieses Eingabegerät wird mit einer ausführlichen Anleitung und einem Softwarepaket ausgeliefert, daß keine Wünsche offen läßt. Soft- und Hardware werden im sogenannten Joyce-MousePack für ca. 180,- DM von der Fa. Reis-Ware angeboten; ein Preis, der in Anbetracht der Fähigkeiten des Systems nicht zu hoch gegriffen ist. Die zur Mouse gelieferte Software erlaubt es, sie in LocoScript und CP/M zum Einsatz zu bringen.
Unter Basic wird zum normalen Basic-Befehlssatz eine vielzahl von mächtigen Befehlen zur Grafikprogrammierung hinzugefügt. Der Anwender braucht sich allerdings um die Programmierung nicht mehr zu kümmern, wenn er unter dem erweiterten Basic das Programm "Centaur" aufruft. Via Menue können mit der Mouse Funktionen angeklickt werden, die alle bekannten Grafikbefehle von horizontal/vertikal spiegeln, drehen, zoomen, vergrößern, kopieren und und und beinhalten.

Pixelweise können unter dem Lupenmenue Punkte auf dem Monitor gesetzt und gelöscht werden. Über das Beschriftungsmenue stehen 24 Schriftarten zur Verfügung, die jeweils in über 20 Größen und 4 Breiten zur Ausführung kommen können.
Neue Schriftarten lassen sich über ein extra Programm ebenso wie die Füllmuster leicht selbst kreieren. Schrift kann auch unter dem "Untagmodus" in vorgegebenen Zeilen- und Spaltenbreiten der Grafik angemessen ausgegeben werden, wobei dann auch schrittweise mit den Cursortasten angesteuert werden kann. Das "Centaur-Programm" simuliert damit einen Desk-Top-Publisher. Faszinierend ist bei alledem die Präzision, mit der der Cursor Punkte ansteuert und festhält.
Was den Programmierern dieses Packs noch zusätzlich Bewunderung abringt, ist die enorme Geschwindigkeit, mit der die Aktionen abgewickelt werden.
Die Option "Ausschnitt löschen" gestattet jetzt auch, Grafik bunt auszudrucken. Nach Erstellung wird ein Bildschirminhalt abgespeichert, danach Teile, die in anderen Farben gedruckt werden sollen, gelöscht. Der noch stehende Bildschirmteil wird unter Zuhilfenahme des Einzelblatteinzugs, in gewünschter Farbe gedruckt. (Der Einzug garantiert, daß die Blätter bei einem erneuten Druckdurchlauf wieder genauso zu liegen kommen, wie beim ersten. Dazu müssen die Blätter auch durch Eigengewicht durch den Einzug auf die Walze fallen können, um gleiche Druckhöhe zu erreichen.)
Jetzt kann die Grafik wieder neu von Diskette geladen, entsprechende Teile gelöscht und der Druck mit einer anderen Farbe nach Wechsel des Farbbandes vorgenommen werden. Dies wird so oft wiederholt, wie Farben in einer Grafik vorhanden sein sollen.
Die Bilder der folgenden Seite entstanden auf dem Joyce-Drucker, und sollen nicht für sich selbst, sondern ebenso wie die technischen Zeichnungen in diesem Buch, die ebenfalls alle mit dem Mousepack erstellt wurden, für dieses Eingabemedium und seine Möglichkeiten sprechen (im Druckermenue hätten auch andere Größen für den Ausdruck - von A4 bis Posterformat - gewählt werden können).






Grafikdemos zum Joyce-MousePack

Neben dem Gerdes MousePack von Reis-Ware gibt es noch andere Eingabemedien, für die aber keine derart gute Grafiksoftware entwickelt wurde. Diese "Mäuse" haben jedoch auch ihre Vorteile. Aufgrund ihrer englischen Herkunft (Kempston, AMX, Electric Studio) besitzen sie entsprechende Schnittstellen, die sie sowohl Hard- als auch Softwaremäßig kompatibel zueinander und zu einer ganzen Reihe von Programmen machen. Gemeinsames Merkmal fast aller Produkte ist die Möglichkeit, das DeskTopPublishing Programm **News Desk** zu steuern, welches sich auch allgemein bei den Joyce-Usern gegen den konkurrierenden Fleet Street Editor durchgesetzt zu haben scheint.

News Desk gestattet dem Anwender über das eingebaute Grafikprogramm Bilder zu erstellen und diese dann mit Text auf einer Din A4 Seite zu arrangieren. Der Text kann über einen Fremdeditor eingegeben werden, und dann nach gutdünken in diverse Spaltenbreiten gebracht werden. Gearbeitet wird bei News Desk immer an Ausschnitten einer A4 Seite. Wurde alles richtig plaziert, kann man sich über ein spezielles Menue die vollständige Seite auf den Monitor holen.

Dem Anwender stehen danach aber immer noch alle Möglichkeiten offen, Text und Grafik zu manipulieren, so daß er sich nach kürzester Zeit als sein eigener Verleger fühlt. Zusätzlich sind zu News Desk allerhand Disketten im Handel, auf denen schon die verschiedensten Schriftarten- und Grafikfonts zur Verfügung stehen. Der Anwender braucht sie nur noch seinen Zwecken entsprechend einzusetzen.

Ohne geeignete Eingabemedien gestaltet sich die Arbeit mit dem Grafikprogramm von News Desk allerdings ein wenig schwierig. Schon der Versuch, eine gebogene Linie mit Hilfe der Cursortasten auf den Monitor zu bringen, scheitert kläglich und hinterläst ein kammartiges Gebilde auf dem Schirm.

Hier könnte nun auch der **Light Pen** von Electric Studio weiterhelfen. (Wird News Desk (ND) gestartet und der Light Pen ist an den PCW angeschlossen, fragt ND automatisch ab, ob mit Cursortasten oder Light Pen gearbeitet werden soll. Dies geschieht auch mit Mäusen des englischen Standards)



Der Light Pen wird gehandhabt wie ein kleiner Zeigestock. In seiner Spitze befindet sich ein Photosensor, der über ein flexibles Kabel die Meldung an den Monitor weitergibt, auf welchen Punkt er gerade zeigt. So können einzelne Punkte eines Menues angewählt werden, die, nachdem der Light Pen auf sie gezeigt hat, invers dargestellt und durch Druck auf die Leertaste angeklickt werden. Befindet man sich im Zeichnenmodus, erscheint ein kleines Kreuz direkt unter dem Light Pen am Monitor, das, je nach gewählter Option, nun Linien hinter sich herzieht, Kreise oder Vielecke zeichnet oder auch dreidimensionale einfache geometrische Gebilde herstellt, von denen jeweils nur die Eckpunkte angegeben werden müssen.

Hat man schon einiges durch Führung des Light Pen über den Monitor gezeichnet und vielleicht schon einige Flächen dunkler gefüllt, wird es schwierig, den Kreuzcursor exakt zu führen. Er vollzieht dann einige Sprünge und ist nicht so leicht zu bewegen, eine gerade Linie über ein dunkleres Feld hinweg zu ziehen. Sollte sich der Cursor dennoch exakt unter der Spitze des Light Pen befinden (wo er ja hingehört) wird es schwierig einen bestimmten Zielpunkt Pixelgenau zu erwischen - man sieht ihn ja nicht, der Lichtgriffel hängt davor. Hat man jetzt eine Technik entwickelt, an der Seite des Griffels vorbeizuschielen, so hat man zumindest nach den ersten Sitzungen einen steifen Arm davongetragen (ist wohl Trainingssache), erst recht dann, wenn der Monitor auf Augenhöhe steht. (Ich habe tatsächlich schon von Anwendern gehört, die sich ihren Monitor auf den Schoß gelegt haben!) Für kleinere Freihandmalereien zwischendurch mag der Light Pen (Preis ca. 280,-DM) seine Berechtigung haben, komplexere Zeichnungen oder Schaltpläne etc. geraten damit allerdings zu einer Qual.

Die Schnittstelle des Light Pen wird über ein Flachbandkabel an den Joyce Bus angeschlossen und hängt dann recht locker an der Rückseite des Joyce. Bei der mitgelieferten Software handelt es sich um ein ähnliches Malprogramm wie bei ND, nur daß einige Optionen nicht zur Verfügung stehen.

Ob nun irgendwelche Zeichnungen gelingen oder nicht ist

nicht immer eine Frage des Eingabemediums. Oft genug steckt mangelnde Kreativität dahinter. Dieses Manko läßt sich mit dem **Scanner** nebst Software beheben, die als "Master Pack" von der Fa. Database Software im Handel sind (ca.380,- DM). Die Idee bei diesem Scanner ist, daß eine Photozelle, neben der eine kleine Lichtquelle angebracht ist, dicht über ein Blatt geführt wird. Sobald das Licht der Birne nicht im gleichen Maß vom weißen Papier zurückstrahlt, wenn es also auf eine Linie trifft, gibt die Photozelle einen Impuls weiter. Werden Photozelle und Birne auf einen Druckerkopf gesetzt, kann man über den Drucker, der Zeile für Zeile langsam abfährt, genau die Stellung eines Punktes auf einem Papier festhalten. Wie der Light Pen wird der Scanner über ein Flachbandkabel an den Expansionsport des Joyce gesteckt. An diesem Kabel baumelt dann ein kleiner Schaltkasten, der im wesentlichen einen Operationsverstärker (CA 3140), je einen 74133 / 74125 und einen Poti enthält. Über diesen nach außen gelegten Poti läßt sich die Empfindlichkeit des Lesekopfes regulieren. Aus diesem Kästchen führt dann ein flexibles Kabel zu der in Plastik geschweißten Photozelle und der Birne.

Das Plastikteil wurde ordentlich dem Joyce-Druckkopf angemessen und läuft millimetergenau zwischen Farbbandabdeckung und Papierandruckrolle. Will man allerdings via Papiereinzugshebel eine Vorlage in den Drucker einziehen, muß man vorher den Lesekopf vom Druckkopf abziehen. Das Farbband sollte aus dem Drucker entfernt werden.

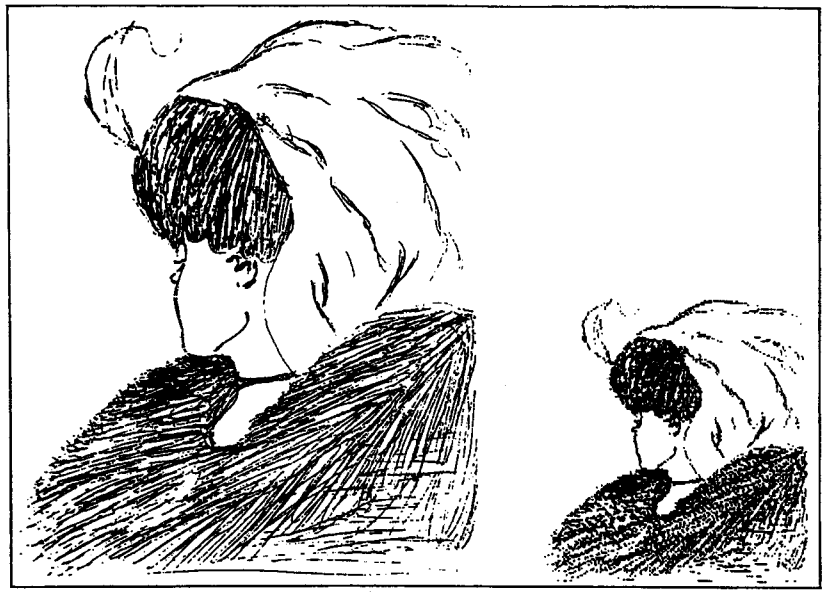
Dem Scanner liegt ein Softwarepaket bei, das sich in zwei Hauptprogramme gliedert - dem zum Scannen und dem zum Bearbeiten der gescannter Bilder -. Wurde das Scanprogramm eingelegt und eine Vorlage in den Drucker gespannt, kann man über ein Menue die verschiedensten Punkte anwählen. So etwa den zu scannenden Ausschnitt oder die Vergrößerung beim Scannen. Dann kann der Scanvorgang starten. Je nach Vergrößerung fällt nun auch die Schrittweite beim Zeilenvorschub des Druckers aus, wenn der Druckkopf Zeile für Zeile der Vorlage abfährt und dabei "beleuchtet". Ebenso wie der Lesekopf das Blatt abfährt, erscheint nun Zeile für Zeile



das Bild auf dem Monitor. Ist man mit dem Ergebnis zufrieden, kann man das Bild abspeichern, wobei man noch entscheiden kann, für welche weitere Software das Ergebnis abgelegt werden soll. Hier kann man sich für News Desk-, Fleet Street Editor- oder Master Paint-Format entscheiden!

Bei größeren dunklen Flächen, Zeitungsausschnitten oder Fotografien etc. wird kein gutes Ergebnis erzielt. Knappe schwarz-weiß Darstellungen gelingen recht gut. Im Beispiel auf der nächsten Seite (Kippbild junges Mädchen oder alte Frau???) war das größere Bild die Vorlage, das kleinere die Scanner-Reproduktion auf dem Joyce-Drucker, die allerdings noch ein wenig größer hätte ausfallen können, ohne Qualitätseinbußen hinzunehmen. Das gescannte Bild wurde im Nachhinein nicht mit dem Malprogramm überarbeitet.

Wer sich von der vorherigen guten Kritik über das Joyce-Mouse-Pack hat überzeugen lassen, jetzt aber doch nicht auf einen Scanner verzichten möchte, sei getröstet. Wurde ein Bild gescannt und für das Master Paint-Programm mit der Extension .pic abgespeichert, kann es auch mit dem Reis-Ware Paket bearbeitet werden! Es entsteht lediglich eine minimale Rechtsverschiebung, die leicht korrigiert werden kann. (Anm.: Bis auf die Grafik in diesem Kapitel, sind alle anderen ohne Scanner entstanden!) Das dem Scanner beiliegende recht ordentliche Grafikprogramm weicht von der Steuerung her von der ND-Ausführung ab. Am linken und oberen Rand des Bildschirms erscheinen Symbole, die mit einem kleinen Stift angeklickt werden müssen. Die Symbole sind leicht verständlich (z.B. Radiergummi) und schnell zu merken. Umständlich erscheint nur, daß der Stift keine Menuefenster aufrufen kann, die an seiner Position erscheinen, sondern immer wieder an den Rand geführt werden muß. Insgesamt stehen 37 Befehle zur Verfügung, allerdings gilt hier dasselbe wie für ND: Ohne Eingabemedium (englische Mouse) wird es schwer, nur mit den Tasten ein ordentliches Bild zustande zu bringen. Mir persönlich erscheint es sinnvoll, lediglich Bilder unter den besprochenen Umständen zu scannen und diese dann mit der Gerdes-Mouse zu bearbeiten. Auf das Scanner.-Grafikprogramm könnte man dann verzichten.



Original gescannt

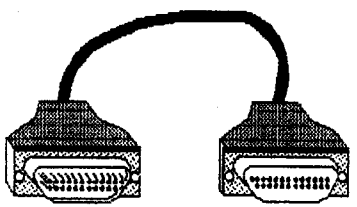
(Die Vorlage und die gescannte Joyce-Printerausgabe mußten aus drucktechnischen Gründen photomechanisch reproduziert werden. Beide sind in natura ein wenig farbkräftiger.)



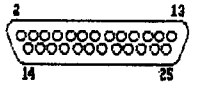
## Die Datenfernübertragung

In einem kurzen Abriß über sinnvolle Joyce-Hardwareerweiterungen darf ein kleines Kapitel über die Datenfernübertragung (DFÜ) nicht fehlen. Zwar nehmen die Disketten des 3"-Formats nicht viel Platz in Anspruch, so daß Dateien mit diesem Medium leicht via Postversand ausgetauscht werden können, doch gibt es auch Situationen, bei denen es auf ständigen und schnellen oder auf wahlfreien Zugriff aus einer Fülle von Informationen ankommt. Es wäre nicht sinnvoll und auch kaum möglich, den Inhalt ganzer Datenbanken (z.B. Juris) auszutauschen, wenn nur einige bestimmte Auskünfte gewünscht werden.
Die einfachste Möglichkeit, mit dem Joyce in ein fremdes Betriebssystem einzusteigen und mit dessen Daten zu arbeiten, bietet sich mit einem **Akustikkoppler.** Knapp beschrieben wandelt ein Akustikkoppler Signale des Computers in hörbare Signale um, die über ein Telefon an einen anderen Koppler (oder Modem) gelangen und wieder in Signale verwandelt werden, mit denen ein angeschlossener Computer etwas anfangen kann.
Getestet wurde dieser Vorgang am Joyce mit einem Akustikkoppler von Dataphon und zwar mit dem s21/23d (ca. 300,-DM). Für den Joyce werden keine kompletten Pakete mit Kabel und Software ausgeliefert, so daß man sich selbst ein Verbindungskabel zwischen CPS8256 und dem Dataphon beschaffen muß. Bei dem zu verwendenden Kabel handelt es sich um ein "MC" (Modem Kabel); das heißt, daß die Verbindungen gerade durchgeschleift sind also Pin2 an Pin2, Pin3 an Pin3 etc. (Sollen Computer ohne Modem direkt miteinander verbunden werden, so geschieht dies über ein "NMC", dem "Null Modem Kabel". Beim NMC werden die Pins 2 und 3 miteinander vertauscht. Tests mit Atari ST und Joyce Mail232 waren erfolgreich!)
Bei den Steckern wird ein Männchen für das Dataphon und ein Weibchen für die Schnittstelle gebraucht (s. Skizze der folgenden Seite). Am preiswertesten ist es, wenn man sich die Verbindung selbst lötet. Die zwei Stecker und ein halber

Meter achtadriges Datenkabel kosten nur ein paar Mark. Die Pins der Stecker sind durchnummeriert und die Adern des Kabels haben verschiedene Farben, so daß es ein Leichtes ist, die Pins 2-8 der Stecker über das Kabel miteinander zu verbinden.



(Null) Modem Kabel (M)MC




Pin 2 Data out (TXD)
Pin 3 Data in (RXD)
Pin 4 RTS out
Pin 5 CTS in
Pin 7 GND
Pin 8 DCD in
Pin 20 DTR
Pin 22 Ring indicator
Die anderen Pins NC

Serieller RS232C Stecker der CPC464

Ein fehlendes besseres DFÜ-Programm als Mail232 läßt sich jetzt bei einem Probelauf leicht beschaffen (Ist aber nicht unbedingt nötig).
Dem Dataphon liegt ein gutes Handbuch bei, so daß ich auf die einzelnen Funktionen nur so weit eingehen möchte, wie es für dieses Beispiel dienlich ist.
Über die RS232 wird das Dataphon mit der Schnittstelle verbunden und für eine ausreichende Stromversorgung (9V Block, Akku oder Netzteil) des Datenföns gesorgt. Nach dem Booten von CP/M wird die Seite 1 der Systemdisketten (Loco-Script) eingelegt und hinter das A> wird MAIL232 eingetippt und so aufgerufen. Die Systemdiskette kann aus dem Laufwerk entnommen und eine formatierte Diskette eingelegt werden. Am oberen Rand des Schirms sind die Menues von Mail232 zu sehen. Durch Druck von [F1] kann man jetzt die Parameter z.B. einer Mailbox seiner Wahl einstellen.
Mailboxen fungieren als Sammelstellen von Nachrichten. Jeder Anwender, der sich dort eingetragen hat (oft kostenlos!), kann unter einem Passwort seiner Wahl Nachrichten für sich hinterlegen lassen oder Nachrichten für andere (auch alle



Mailboxbenutzer - schwarzes Brett -) hinterlegen. In diesen Mailboxen findet man auch Programme der Public Domain, die man sich dort via Akustikkoppler abholen kann.
Für unser Beispiel kann die Mailbox RBBS (0431/336038) in Kiel dienen (viele CP/M Programme). Für diese Box müßten die Parameter von [F1] auf 300 Baud senden, 300 Baud empfangen, 8 Datenbits, keine Parität und 1 Stopbit eingestellt werden. Hardware handshaking kann unterbleiben. Durch Druck auf [EXIT] verschwindet das Menue und die Parameter sind eingestellt. Am Dataphon wird der Bereich Auto gewählt. Danach wird der Telefonhörer in die Gummimuffen des Datenföns gepreßt und das Ganze auf die Seite gelegt (Bei langem Betrieb mit Hörer nach unten können Kohlemembranen älterer Telefone zusammenbacken und so den Schallpegel senken). Jetzt kann die Telefonnummer der Mailbox angewählt werden. Kommt die Verbindung zustande, kann der zweite Schalter des Dataphons von off auf 300 plaziert werden. (Geschieht das vor Zustandekommen der Verbindung, erscheinen oft noch Störzeichen auf den Monitor).
Nach kurzer Zeit meldet sich die Mailbox auf dem Bildschirm und gibt dem Anwender genaue Hinweise, z.B. was er beachten muß, wie er sich einzutragen hat und wie Programme gelesen werden können.
All das, was über den Monitor läuft, kann Joyce speichern. Durch Druck auf [F3] erscheint ein Fenster, in das man einen Dateinamen für das zu Empfangende (oder zu Sendende) einträgt. Bei Druck auf Exit geschieht nichts weiter, als daß Joyce sich diesen Namen merkt. Läuft jetzt ein interessantes Programm über den Monitor, braucht der Anwender nur nochmals [F3] zu drücken und durch [ENTER] zu bestätigen. Jetzt wird eine Datei mit diesem Namen auf Diskette angelegt und alles, was über den Monitor läuft, wandert in den Speicher von Joyce. Ist dieser voll, wird durch Abspeichern in die Diskettendatei neuer Platz geschaffen. Hat der Anwender alle Informationen erhalten, wird durch gleichzeitigen Druck auf [ALT] und [STOP] der restliche Speicherinhalt auf Diskette geschrieben und die Datei geschlossen. Sie steht jetzt dem Anwender zur Verfügung.

Auf diese Weise kann man sich nun über eine Mailbox komplexere Programme zur DFÜ beschaffen (z.B. Kermit) und der Welt der Datenfernübertragung steht nichts mehr im Weg.
An dieser Stelle noch ein wichtiger Tip:
Wird Mail232 von Laufwerk A aus gestartet, muß Joyce seinen Zwischenspeicher immer wieder dahin entleeren. Bei dieser brummigen Arbeit gehen ihm immer einige Zeichen verloren, die der Sender in der Zwischenzeit munter weitergibt. Um dies zu verhindern, ist es zweckmäßig, Mail232 in den das Laufwerk M zu holen, und M als Standardlaufwerk anzusprechen. ("M:" schreiben und RETURN drücken). So werden alle Signale gleich in der Ramdisk abgelegt, wobei kein Zeichen verloren geht. Bei der Einstellung 300 Baud kann unser Joyce dann mehr als eine Stunde seinen erweiterten Speicher vollschreiben.
Zur Erreichung dieses Ziels muß allerdings eine Hürde genommen werden! Mail232 ist als "System-Datei" auf Diskette zu finden. Wenn man sie mit der Dateiverwaltung von Locoscript ansehen will, geht das nur, wenn der Modus "Versteckte Dateien sichtbar" eingeschaltet ist. Solche Systemdateien können nicht mit **Pip** in den M-Speicher geholt werden. Hier hilft uns das CP/M-Programm **Set** weiter. Mit der Dateiverwaltung von Locoscript kopiert man sowohl **Set** als auch **Mail232** auf eine Diskette. Nach dem Neubooten von CP/M wird diese Diskette ins Laufwerk geschoben und eingetippt: **set mail232.com dir** . **Set** wird dann aus der Systemdatei **Mail232** eine normale Datei herstellen, die wie jede andere auch mit **Pip** nach "M" kopiert werden kann.
Zum Experimentieren hier weitere Mailboxen und ihre dazugehörigen Parameter:

```
Data Becker 0211/340071 24h online 300/8/N/1
IBB 030/6818679 24h online 300/7/N/1
MCS 040/2512371  & 2512373 24h online 300/7/E/1
c.l.i.n.c.h 040/6323517 24h online 300/8/N/1
OIS München 089/4606021 24h online 300/8/N/1
```

In den Mailboxen sind Listen über Rufnummern anderer Boxen erhältlich. (In fast jeder größeren Stadt existiert mindestens eine!)



# Der Joyce nach Feierabend

Nach der Fülle von trockenem Stoff und zum Abschluß nun zu einem leichteren Gebiet, auf dem der Joyce eigentlich nichts zu suchen hat.
In den Anwenderkreisen, in denen der Joyce zu Hause ist, wurde beim Kauf sicher nicht in Erwägung gezogen, daß zur Entspannung auch Spiele auf ihm gestartet werden können.
Gegenüber einigen Soundmaschinen mit Buntmonitor, Joystick, Trackball und dergleichen mehr, die den Eindruck machen, als seien sie einer Spielhöhle entstiegen, um Kinder von ihrem pädagogisch wertvollen Spielzeug wegzulocken, gibt sich der Joyce mit seinem Grünmonitor und dem kleinen Piepser zugegebenermaßen sehr bieder. Dementsprechend spät wurde auch erst die Software für ihn angepaßt, die in erster Linie der Entspannung (oder auch nicht) dienen sollte. Die Softwareentwickler und -händler dieses Genres ließen sich Zeit, und als sie dann ihre Waren auf den Markt brachten, waren diese auf das Käuferpotential der PCW's abgestimmt.
"Weltraumballerkillspiele" wurden erst gar nicht ins Repertoire aufgenommen. Dafür gab es gleich zu Anfang drei verschiedene Schachspiele für den Joyce, wobei das "Cyrus-Chess" bei den mir bekannten Usern zum Favoriten avancierte.
Bald danach erschienen die ersten Textadventures wie "Silikon Dreams" (mit ein wenig Grafik) oder "Lord of the Rings", bei denen man sich durch einen Roman lesen muß, und durch Anweisungen wie "go, put, take" oder "read" den Held der Geschichte durch ein Gestrüpp von Widernissen lenken muß. Daß diese Adventures in Englisch gehalten waren, diente so manchem User als Ausrede für seinen Spieltrieb, konnte er doch die Aufbesserung seiner nachlassenden Englischkenntnisse vorschieben.
Geschicklichkeitsspiele wie "Bounder", bei dem ein Ball hüpfender Weise über ein Feld voll böser Hindernisse gelenkt werden mußte (für Eingeweihte: cheat-mode: gleichzeitig die Tasten "c"h"e"a"t drücken) oder Flugsimulatoren wie "Tomahawk" (Hubschrauber) zogen nach.
Die absoluten Spielehits für den Joyce kamen aber immer nur

von einer Firma: OCEAN.
Der größte Erfolg dieses Softwarehauses war ein Spiel namens "Batman", wobei diesmal der Spruch vom nomen, welches omen sein soll, nicht zutraf. Zwar war das Titelbild noch arg vom stupiden Vorbild der Comicfigur beeinflußt, doch das Spiel selbst schlug alles Bisherige. Grafik und Sound des Joyce waren voll ausgereizt und das Spiel an sich verlangte dem Joycer soviel Witz und Verstand, Erinnerungsvermögen und spielerisches Geschick ab, daß der Reiz dieses Spieles so manchen eine Masse Zeit gekostet hat. Eigentlich ist der Spieler nur damit beschäftigt, eine kleine quirlige Figur durch ein Labyrinth von 150! Räumen zu schicken und sie dabei 8 Teile für ein Flugzeug sammeln zu lassen, mit dem es dann fliehen kann. Eigentlich ist aber die ganze Aktion eine einzige Flucht vor Fallen, bösen Maschinen oder finsteren Gestalten. Geballert wird nicht, nur weggelaufen, wobei "nur" eine schlimme Untertreibung ist. Wer das Spiel nicht kennt, sollte es ausprobieren. Ich habe keinen Joycer kennengelernt, der nicht begeistert war!
Von diesem Erfolg angespornt lieferte OCEAN dann gleich einen weiteren Hit nach: "Head over Heals". Hier galt es fast doppelt soviele Räume mit zwei Figuren, die man zwecks Erweiterung ihrer Fähigkeiten übereinanderstellen konnte, zu erkunden. Wurde Joyce mit einem Sound Digitizer und einem Joystickkontroller englischer Herkunft ausgestattet, hatte man neben dem bequemen Knüppeleingabemedium sogar noch ständig eine Rockmelodie im Ohr.
Als weiteres Produkt für den Joyce erschien dann "Match-Day II" von OCEAN. Hier wird sehr realistisch ein Fußballspiel simuliert, wobei der User in der Lage ist, zwischen einer Menge von Möglichkeiten wie: Austragung eines Pokals, Computer gegen Computer oder zwei Spieler gegen Computer spielen zu lassen etc. wählen kann. Auch mit diesem Spiel setzte OCEAN die Tradition, hochwertige Spiele für den Joyce zu liefern, fort.
Das jüngste der qualitativ hochwertigen Spiele, war ein Geschicklichkeitsspiel, das schon auf anderen Computern zum Spiel des Jahres ernannt worden war: "Tetris". Hier fallen



geometrische Figuren vom Computerhimmel, und der gequälte Anwender muß versuchen, sie durch geschickte Drehung und Placierung so aufeinander fallen zu lassen, daß sie möglichst wenig Raum einnehmen und immer den Boden bedecken. Nur so können folgende Schichten nachsacken und es verhindern, daß die Türme in den Himmel wachsen, was den Abbruch des Spiels bedeuten würde. Ein simples Spiel mit ungeheurem Spielreiz!
Wer nun bei diesem knappen Review, das vieles aus dem Spielesektor nicht erwähnt hat, auf den Geschmack gekommen ist und bisher nicht daran gedacht hat, mal mit oder gegen seinen Joyce zu spielen, der kann, ohne Geld im nächsten Softwareladen auszugeben, das folgende kleine BASIC-Listing abtippen und mal sehen, wie sich Joyce denn so als Spielpartner macht. Bei dem winzigen Bonbon zum Abschluß handelt es sich um "Kniffel", ein Würfelspiel, daß wohl den meisten vertraut sein dürfte (von einer Karte mit Ergebnisvorlagen muß jeweils nach dreimaligem Würfeln pro Durchgang eine Spalte ausgefüllt werden, wobei man zumindest zu Anfang in der Wahl der Spalten frei ist). Die Qualität des Spiels entspricht zwar nicht gerade dem Standart von Ocean, dafür wurde es aber selbst entwickelt. Wer sich das Abtippen ersparen möchte, findet dieses Programm ready to run auf der Diskette zum Buch. Wir wünschen:

Viel Spaß dabei!

```
10 'Spielprogramm: Kniffel
20 '
30 '
40 stu$=HEX$(PEEK(&HFBF6))
50 min$=HEX$(PEEK(&HFBF7))
60 sec$=HEX$(PEEK(&HFBF8))
70 date=PEEK(&HFBF4)+PEEK(&HFBF5)*256
80 random=date-VAL(stu$)-VAL(min$)-VAL(sec$)
90 RANDOMIZE random
100 '
110 'Copyright 1988 by Bernhard Graßhoff, Heikendorf
120 '
130 'Initialisierung
140 DEFINT a-z
150 DIM z(6,13)
160 GOSUB 3040'Maschinenprogramm INPUT
```

```
170 CALL disable% 'Ctrl-C und Ctrl-S außer Betrieb
180 esc$=CHR$(27):co$=esc$+"e":cf$=esc$+"f":csave$=esc$+"j":cload$=esc$+"k":invon$=esc$+"p
":invoff$=esc$+"q":uon$=esc$+"r":uoff$=esc$+"u":umon$=esc$+"v":umoff$=esc$+"w":home$=esc$+
"H":cls$=esc$+"E"+home$:links$=esc$+"D"+esc$+"D"+" "
190 bell$=CHR$(7):deutsch$=esc$+"2"+CHR$(2)
200 normal$=esc$+"X"+CHR$(33)+CHR$(33)+CHR$(60)+CHR$(119)'Fenster auf ganzen Bildschirm au
sweiten
210 DEF FNfenster$(oz,ls,h,b)=esc$+"X"+CHR$(32+oz)+CHR$(32+ls)+CHR$(31+h)+CHR$(31+b) 'Fens
ter einrichten
220 '
230 PRINT FNfenster$(0,0,31,90) 'Fenster auf ganzen Bildschirm ausweiten
240 PRINT cls$
250 a=3:b=20:c=13:d=50:GOSUB 2870 'Rahmen zeichnen
260 PRINT FNfenster$(4,21,10,49) 'Fenster für 1. Spielmeldung einrichten
270 PRINT "          "CHR$(164)" 1988 by Bernhard Graßhoff":PRINT
280 PRINT "            ";invon$+"  K  N  I  F  F  E  L  "+invoff$
290 PRINT:PRINT:PRINT "    Wv. Spieler (1-6) nehmen am Spiel teil ?":PRINT
300 PRINT "      Bitte Zahl eingeben !   ";
310 OPTION RUN
320 a$=INPUT$(1)
330 spieler=VAL (a$):IF spieler<1 OR spieler>6 THEN 320
340 OPTION STOP
350 PRINT FNfenster$(0,0,31,90) 'Fenster auf ganzen Bildschirm ausweiten
360 PRINT cls$
370 a=0:b=0:c=31:d=88:PRINT cf$;umoff$;:GOSUB 2870 'Rahmen zeichnen
380 '
390 FOR runde=1 TO 13
400 FOR p=1 TO spieler
410 '
420 GOSUB 440:GOTO 480
430 '
440 FOR i=1 TO 5
450 t(i)=INT (RND (1)*6)+1'Würfeln Wert zuweisen
460 NEXT i:RETURN
470 '
480 GOSUB 520:GOTO 610
490 '
500 'Eingefügtes Unterprogramm 'Sortieren der Würfel-Werte'
510 '
520 f=0
530 FOR i=1 TO 4
540 IF t(i+1)>=t(i) THEN 570
550 SWAP t(i),t(i+1)
560 f=1
570 NEXT i
580 IF f=1 THEN 520
590 RETURN
600 '
610 PRINT FNfenster$(1,2,28,87) 'Fenster für Spielfeld einrichten
620 '
630 PRINT cls$
640 PRINT " Spieler";p
650 PRINT:PRINT
660 PRINT " Sie haben folgende Zahlen gewürfelt:":PRINT
670 FOR f=1 TO 5
680 PRINT "          Würfel";f;"=";t(f)
690 NEXT
700 '
```



```
710 PRINT FNfenster$(12,1,21,90) 'Fenster für Spielkartenausgabe einrichten
720 '
730 PRINT "  Hier ist Ihre Kniffelkarte":PRINT
740 PRINT " 1er  =1            ";ei(p);:IF ei(p)=0 AND z(p,1)=0 THEN PRINT links$:ELSE PRI
NT
750 PRINT " 2er  =2            ";zw(p);:IF zw(p)=0 AND z(p,2)=0 THEN PRINT links$:ELSE PRI
NT
760 PRINT " 3er  =3            ";dr(p);:IF dr(p)=0 AND z(p,3)=0 THEN PRINT links$:ELSE PRI
NT
770 PRINT " 4er  =4            ";vi(p);:IF vi(p)=0 AND z(p,4)=0 THEN PRINT links$:ELSE PRI
NT
780 PRINT " 5er  =5            ";fu(p);:IF fu(p)=0 AND z(p,5)=0 THEN PRINT links$:ELSE PRI
NT
790 PRINT " 6er  =6            ";se(p);:IF se(p)=0 AND z(p,6)=0 THEN PRINT links$:ELSE PRI
NT
800 PRINT " Dreierpasch =7     ";dp(p);:IF dp(p)=0 AND z(p,7)=0 THEN PRINT links$:ELSE PRI
NT
810 PRINT " Viererpasch =8     ";vp(p);:IF vp(p)=0 AND z(p,8)=0 THEN PRINT links$:ELSE PRI
NT
820 PRINT " Full-House  =9     ";fh(p);:IF fh(p)=0 AND z(p,9)=0 THEN PRINT links$:ELSE PRI
NT
830 PRINT " Kleine Str. =10    ";ks(p);:IF ks(p)=0 AND z(p,10)=0 THEN PRINT links$:ELSE PR
INT
840 PRINT " Große  Str. =11    ";gr(p);:IF gr(p)=0 AND z(p,11)=0 THEN PRINT links$:ELSE PR
INT
850 PRINT " Kniffel     =12    ";kn(p);:IF kn(p)=0 AND z(p,12)=0 THEN PRINT links$:ELSE PR
INT
860 PRINT " Chance      =13    ";ch(p);:IF ch(p)=0 AND z(p,13)=0 THEN PRINT links$:ELSE PR
INT
870 FOR wn=1 TO 2
880 '
890 'Beginn des Dialoges
900 '
910 PRINT FNfenster$(13,44,5,43):PRINT cls$;"Wv. Würfel sollen neu gesetzt werden ?";csave
$;
920 PRINT co$;cload$;:maxlen=1:GOSUB 2990:PRINT cf$;
930 IF x$="1" THEN n=1:GOTO 1000
940 IF x$="2" THEN n=2:GOTO 1000
950 IF x$="3" THEN n=3:GOTO 1000
960 IF x$="4" THEN n=4:GOTO 1000
970 IF x$="5" THEN n=5:GOTO 1000
980 IF x$="0" THEN 1380
990 PRINT bell$;cload$;"  ";:GOTO 920
1000 PRINT:IF n=5 THEN GOSUB 440:GOTO 1210
1010 PRINT FNfenster$(13,44,10,44):PRINT cls$;
1020 PRINT co$;
1030 '
1040 'Neue Würfel-Wert Zuweisung
1050 '
1060 FOR nn=1 TO n
1070 PRINT "Würfelnummer=";csave$;
1080 maxlen=1:GOSUB 2990:nu(nn)=VAL(x$):IF nu(nn)<1 OR nu(nn)>5 THEN PRINT bell$;cload$;:G
OTO 1080 ELSE PRINT
1090 NEXT nn
1100 PRINT cf$
1110 f=0 'Sortierung der neu zu würfelnden Werte, um Fehlfunktion auszuschließen
1120 FOR nn=1 TO n-1
1130 IF nu(nn+1)>=nu(nn) THEN 1160
```

```
1140 SWAP nu(nn),nu(nn+1)
1150 f=1
1160 NEXT
1170 IF f=1 THEN 1110
1180 FOR nn=1 TO n
1190 t(nu(nn))=INT(RND(1)*6)+1
1200 NEXT nn
1210 GOSUB 520
1220 PRINT
1230 PRINT home$;
1240 '
1250 'Ausgabe der Werte in entspr. Fenstern
1260 '
1270 IF wn=2 THEN PRINT FNfenster$(6,65,7,15):GOTO 1290
1280 PRINT FNfenster$(6,39,7,15)
1290 FOR i=1 TO 5
1300 PRINT "Würfel"i"="t(i)
1310 NEXT i
1320 '
1330 PRINT normal$
1340 NEXT wn
1350 PRINT:PRINT normal$:GOTO 1380
1360 PRINT bell$;
1370 '
1380 PRINT FNfenster$(13,43,5,42):PRINT cls$;"Wo wollen Sie die Augenzahl eintragen?";csav
e$
1390 PRINT co$;cload$;:maxlen=2:GOSUB 2990:we=VAL(x$):PRINT cf$;
1400 '
1410 'Prüfen der Eingaben auf Richtigkeit und Setzen einzelner Flags
1420 '
1430 IF we<1 OR we>13 THEN 1450
1440 ON we GOTO 1460,1510,1560,1610,1660,1710,1760,1800,1840,1880,1910,1950,1980
1450 PRINT bell$;:GOTO 1390
1460 n=1:IF z(p,1)=1 THEN 1360 ELSE z(p,1)=1
1470 GOSUB 2030:GOSUB 2550:IF y=4 THEN gy(p)=gy(p)+100
1480 ei(p)=e
1490 e=0
1500 GOTO 2090
1510 n=2:IF z(p,2)=1 THEN 1360:ELSE z(p,2)=1
1520 GOSUB 2030:GOSUB 2550:IF y=4 THEN gy(p)=gy(p)+100
1530 zw(p)=e
1540 e=0
1550 GOTO 2090
1560 n=3:IF z(p,3)=1 THEN 1360:ELSE z(p,3)=1
1570 GOSUB 2030:GOSUB 2550:IF y=4 THEN gy(p)=gy(p)+100
1580 dr(p)=e
1590 e=0
1600 GOTO 2090
1610 n=4:IF z(p,4)=1 THEN 1360:ELSE z(p,4)=1
1620 GOSUB 2030:GOSUB 2550:IF y=4 THEN gy(p)=gy(p)+100
1630 vi(p)=e
1640 e=0
1650 GOTO 2090
1660 n=5:IF z(p,5)=1 THEN 1360:ELSE z(p,5)=1
1670 GOSUB 2030:GOSUB 2550:IF y=4 THEN gy(p)=gy(p)+100
1680 fu(p)=e
1690 e=0
1700 GOTO 2090
```



```
1710 n=6:IF z(p,6)=1 THEN 1360:ELSE z(p,6)=1
1720 GOSUB 2030:GOSUB 2550:IF y=4 THEN gy(p)=gy(p)+100
1730 se(p)=e
1740 e=0
1750 GOTO 2090
1760 zp=3:n=5:IF z(p,7)=1 THEN 1360:ELSE z(p,7)=1:GOSUB 2370
1770 GOSUB 2550:IF y=4 THEN gy(p)=gy(p)+100
1780 dp(p)=g
1790 GOTO 2090
1800 zp=4:n=5:IF z(p,8)=1 THEN 1360:ELSE z(p,8)=1:GOSUB 2370
1810 GOSUB 2550:IF y=4 THEN gy(p)=gy(p)+100
1820 vp(p)=g
1830 GOTO 2090
1840 GOSUB 2450:IF z(p,9)=1 THEN 1360:ELSE z(p,9)=1
1850 GOSUB 2550:IF y=4 THEN gy(p)=gy(p)+100
1860 fh(p)=g
1870 GOTO 2090
1880 GOSUB 2500:IF z(p,10)=1 THEN 1360:ELSE z(p,10)=1
1890 IF y>=3 THEN ks(p)=30:ELSE ks(p)=0
1900 GOTO 2090
1910 GOSUB 2500:IF z(p,11)=1 THEN 1360:ELSE z(p,11)=1
1920 IF y=4 THEN gr(p)=40:ELSE gr(p)=0
1930 GOSUB 2550:IF y=4 THEN gy(p)=gy(p)+100
1940 GOTO 2090
1950 IF z(p,12)=1 THEN 1360:ELSE z(p,12)=1:GOSUB 2550
1960 IF y=4 THEN kn(p)=50:ELSE kn(p)=0
1970 GOTO 2090
1980 IF z(p,13)=1 THEN 1360:ELSE z(p,13)=1
1990 FOR i=1 TO 5
2000 ch(p)=ch(p)+t(i)
2010 NEXT i:GOSUB 2550:IF y=4 THEN gy(p)=gy(p)+100
2020 GOTO 2090
2030 IF t(1)=n THEN e=e+n
2040 IF t(2)=n THEN e=e+n
2050 IF t(3)=n THEN e=e+n
2060 IF t(4)=n THEN e=e+n
2070 IF t(5)=n THEN e=e+n
2080 RETURN
2090 PRINT:NEXT p 'Nächster Spieler
2100 PRINT:NEXT runde
2110 FOR i=1 TO 30:t$=INKEY$:NEXT 'Leeren des Tastaturpuffers
2120 PRINT normal$:PRINT cls$:PRINT SPC(34);uon$;" A B R E C H N U N G ";uoff$
2130 '
2140 spielermax=0
2150 a=5:b=28:c=17:d=34:GOSUB 2890
2160 PRINT FNfenster$(6,29,14,33)
2170 FOR i=1 TO spieler
2180 PRINT home$;cls$;"Spieler"i"hat:":PRINT
2190 q(i)=ei(i)+zw(i)+dr(i)+vi(i)+fu(i)+se(i)
2200 PRINT "Oberer Teil      : ";q(i)
2210 IF q(i)>=63 THEN q(i)=q(i)+35:bonus=35
2220 PRINT "Bonus            : ";bonus:bonus=0
2230 unten=dp(i)+vp(i)+fh(i)+ks(i)+gr(i)+kn(i)+ch(i)
2240 PRINT "Unterer Teil     : ";unten
2250 IF kn(i)=50 THEN q(i)=q(i)+gy(i) ELSE gy(i)=0
2260 PRINT "Extrakniffel     : ";gy(i)
2270 PRINT "_____________________________"
2280 q(i)=q(i)+unten
```

```
2290 IF q(i)>q(spielermax) THEN spielermax=i
2300 PRINT:PRINT "Gesamt           : "q(i)"Punkte"
2310 t$=INPUT$(1)
2320 NEXT i
2330 PRINT co$:PRINT "Gewonnen hat Spieler "spielermax
2340 PRINT:PRINT "Nochmal ÄJ/NÖ";:maxlen=1:GOSUB 2990:IF UPPER$(x$)="J" THEN PRINT:RUN
2350 PRINT FNfenster$(0,0,31,90)
2360 PRINT cls$:END
2370 GOSUB 520
2380 y=0:FOR i=1 TO 4
2390 IF t(i)=t(i+1) THEN y=y+1
2400 NEXT:ok=0
2410 IF zp=3 THEN GOSUB 2590
2420 IF zp=4 THEN GOSUB 2670
2430 IF ok=1 THEN g=t(1)+t(2)+t(3)+t(4)+t(5)+t(6):ELSE g=0
2440 RETURN
2450 GOSUB 520
2460 GOSUB 2780
2470 IF ok=1 THEN g=25
2480 IF ok=0 THEN g=0
2490 RETURN
2500 GOSUB 520
2510 y=0:FOR i=1 TO 4
2520 IF t(i)=t(i+1)-1 THEN y=y+1
2530 NEXT
2540 RETURN
2550 y=0:FOR i=1 TO 4
2560 IF t(i)=t(i+1) THEN y=y+1
2570 NEXT
2580 RETURN
2590 FOR i=1 TO 3
2600 IF t(i)=t(i+1) THEN bh=1:ELSE bh=0
2610 IF bh=1 THEN 2640
2620 NEXT
2630 ok=0:RETURN
2640 IF t(i)=t(i+2) THEN ok=1:RETURN
2650 GOTO 2620
2660 ok=0:RETURN
2670 FOR i=1 TO 4
2680 IF t(i)=t(i+1) THEN 2710
2690 GOTO 2750
2700 ok=0:RETURN
2710 IF t(i)=t(i+2) THEN 2740
2720 GOTO 2750
2730 ok=0:RETURN
2740 IF t(i)=t(i+3) THEN 2770
2750 NEXT i
2760 ok=0:RETURN
2770 ok=1:RETURN
2780 IF t(1)=t(2) THEN 2800
2790 ok=0:RETURN
2800 IF t(4)=t(5) THEN 2820
2810 ok=0:RETURN
2820 IF t(3)=t(2) THEN 2850
2830 IF t(3)=t(4) THEN 2850
2840 ok=0:RETURN
2850 ok=1:RETURN
2860 '
```



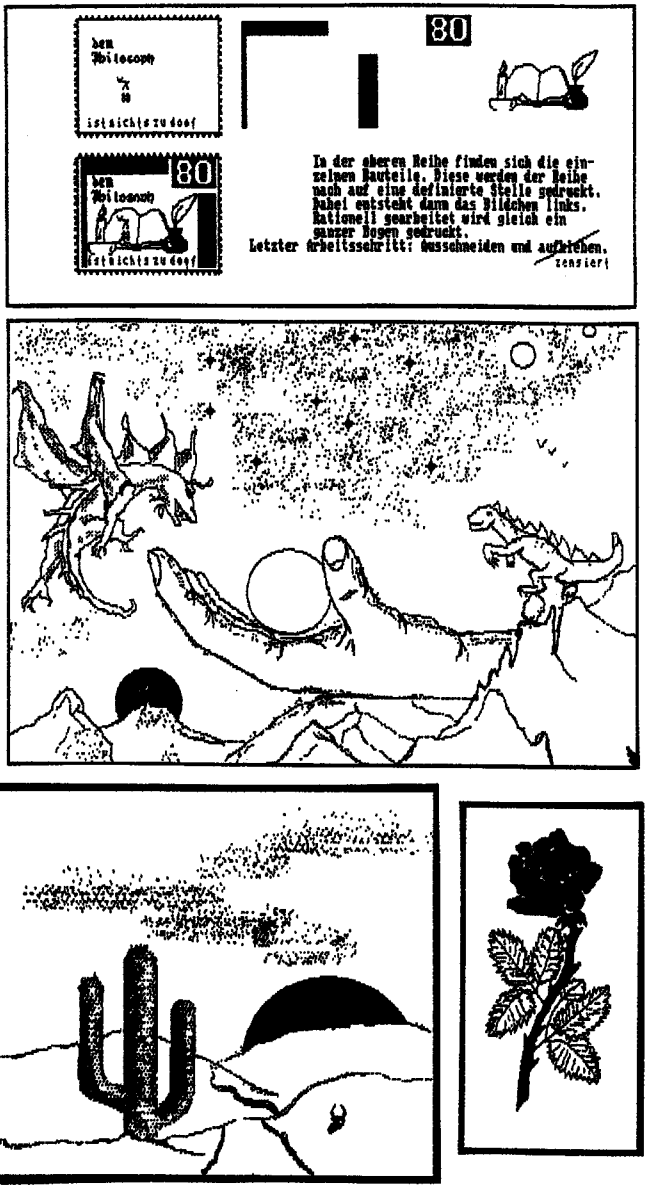
```
2870 'Rahmen zeichnen
2880 '
2890 PRINT FNfenster$(a-1,b-1,c+3,d+3)
2900 PRINT home$;CHR$(134);'Ecke links oben
2910 PRINT STRING$(d,CHR$(138));'Balken waagerecht
2920 PRINT CHR$(140);:PRINT umon$'Ecke rechts oben
2930 FOR i=1 TO c-2:PRINT CHR$(133);STRING$(d," ");CHR$(133):NEXT
2940 PRINT CHR$(131);
2950 PRINT STRING$(d,CHR$(138));
2960 PRINT CHR$(137);
2970 PRINT home$;
2980 PRINT FNfenster$(a,b+2,c+2,d+2):RETURN
2990 x$=""
3000 POKE &H80,maxlen:POKE &H82,0
3010 CALL dskres%:CALL inp%
3020 FOR yyy=&H82 TO &H81+PEEK(&H81):x$=x$+CHR$(PEEK(yyy)):NEXT
3030 RETURN
3040 REM --- DATA ---
3050 DATA 5c,0c,00,00,0e,68,11,a2,f5,cd,05,00,c9
3060 DATA 00,00,00,00,0e,69,11,af,f5,cd,05,00,c9
3070 DATA 59,0e,04,cd,05,00,c9,00
3080 DATA 11,00,00,0e,0a,cd,05,00,c9
3090 DATA 0e,0d,cd,05,00,c9
3100 DATA 11,0a,00,0e,6d,cd,05,00,c9
3110 MEMORY &HF5A1:RESTORE 3050:FOR i=&HF5A2 TO &HF5D8:READ a$:POKE i,VAL("&H"+a$):NEXT
3120 dset%=&HF5A6:dget%=&HF583:dmp%=&HF5BC:inp%=&HF5C4:dskres%=&HF5CD
3130 disable%=&HF5D3
3140 RETURN
```

# S t i c h w o r t v e r z e i c h n i s

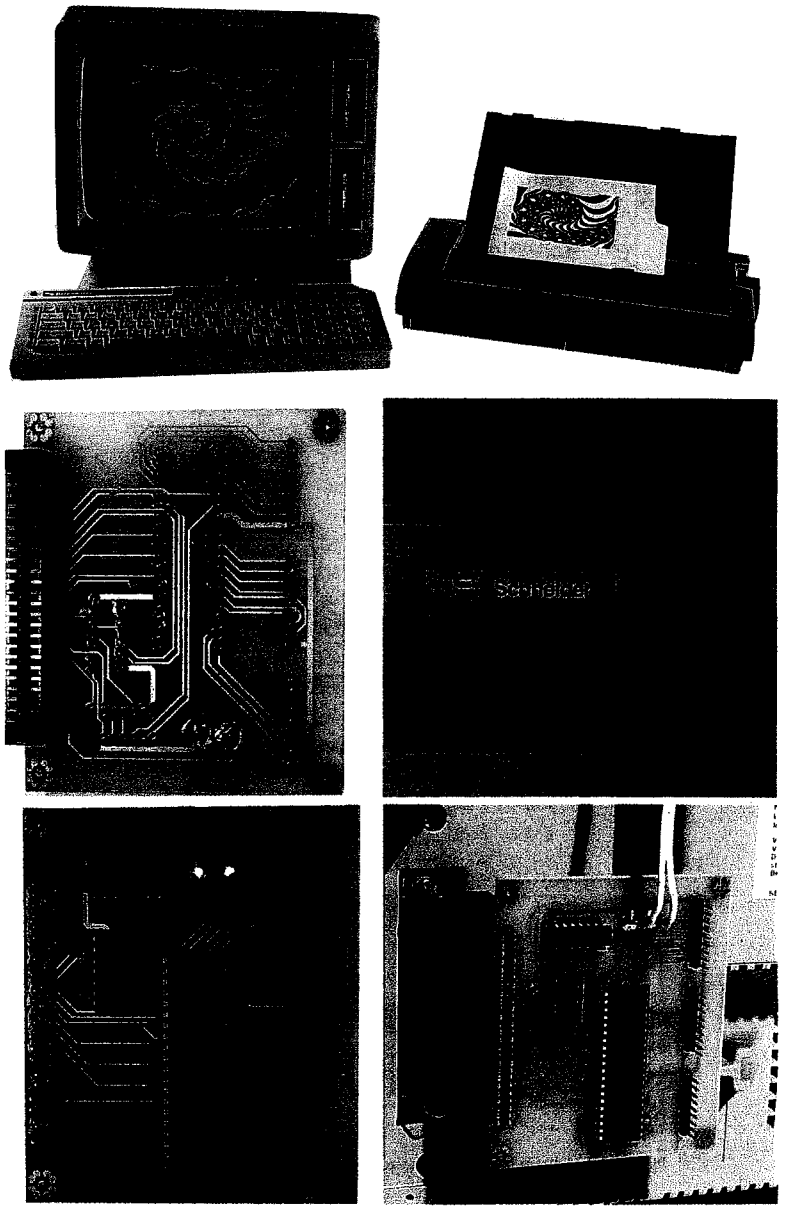
# Vollständige ASCII-Code-Tabelle

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0= | 1= ☺ | 2= ☻ | 3= ♥ | 4= ♦ | 5= ♣ | 6= ♠ | 7= • | 8= ◘ |
| 9= ○ | 10= ◙ | 11= ♂ | 12= ♀ | 13= ♪ | 14= ♫ | 15= ☼ | 16= ► | 17= ◄ |
| 18= ↕ | 19= ‼ | 20= ¶ | 21= § | 22= ▬ | 23= ↨ | 24= ↑ | 25= ↓ | 26= → |
| 27= ← | 28= ∟ | 29= ↔ | 30= ▲ | 31= ▼ | 32= | 33= ! | 34= " | 35= # |
| 36= $ | 37= % | 38= & | 39= ' | 40= ( | 41= ) | 42= * | 43= + | 44= , |
| 45= - | 46= . | 47= / | 48= 0 | 49= 1 | 50= 2 | 51= 3 | 52= 4 | 53= 5 |
| 54= 6 | 55= 7 | 56= 8 | 57= 9 | 58= : | 59= ; | 60= < | 61= = | 62= > |
| 63= ? | 64= @ | 65= A | 66= B | 67= C | 68= D | 69= E | 70= F | 71= G |
| 72= H | 73= I | 74= J | 75= K | 76= L | 77= M | 78= N | 79= O | 80= P |
| 81= Q | 82= R | 83= S | 84= T | 85= U | 86= V | 87= W | 88= X | 89= Y |
| 90= Z | 91= [ | 92= \ | 93= ] | 94= ↑ | 95= _ | 96= ` | 97= a | 98= b |
| 99= c | 100= d | 101= e | 102= f | 103= g | 104= h | 105= i | 106= j | 107= k |
| 108= l | 109= m | 110= n | 111= o | 112= p | 113= q | 114= r | 115= s | 116= t |
| 117= u | 118= v | 119= w | 120= x | 121= y | 122= z | 123= { | 124= \| | 125= } |
| 126= ~ | 127= ⌂ | 128= Ç | 129= ü | 130= é | 131= â | 132= ä | 133= à | 134= å |
| 135= ç | 136= ê | 137= ë | 138= è | 139= ï | 140= î | 141= ì | 142= Ä | 143= Å |
| 144= É | 145= æ | 146= Æ | 147= ô | 148= ö | 149= ò | 150= û | 151= ù | 152= ÿ |
| 153= Ö | 154= Ü | 155= ø | 156= £ | 157= Ø | 158= × | 159= ƒ | 160= á | 161= í |
| 162= ó | 163= ú | 164= ñ | 165= Ñ | 166= ª | 167= º | 168= ¿ | 169= ® | 170= ¬ |
| 171= ½ | 172= ¼ | 173= ¡ | 174= « | 175= » | 176= ▒ | 177= ▓ | 178= │ | 179= ┤ |
| 180= ╡ | 181= ╢ | 182= ╖ | 183= ╕ | 184= ╣ | 185= ║ | 186= ╗ | 187= ╝ | 188= ╜ |
| 189= ╛ | 190= ┐ | 191= └ | 192= ┴ | 193= ┬ | 194= ├ | 195= ─ | 196= ┼ | 197= ╞ |
| 198= ╟ | 199= ╚ | 200= ╔ | 201= ╩ | 202= ╦ | 203= ╠ | 204= ═ | 205= ╬ | 206= ╧ |
| 207= ╨ | 208= ╤ | 209= ╥ | 210= ╙ | 211= ╘ | 212= ╒ | 213= ╓ | 214= ╫ | 215= ╪ |
| 216= ┘ | 217= ┌ | 218= █ | 219= ▄ | 220= ▌ | 221= ▐ | 222= ▀ | 223= α | 224= ß |
| 225= Γ | 226= π | 227= Σ | 228= σ | 229= µ | 230= τ | 231= Φ | 232= Θ | 233= Ω |
| 234= δ | 235= ∞ | 236= φ | 237= ε | 238= ∩ | 239= ≡ | 240= ± | 241= ≥ | 242= ≤ |
| 243= ⌠ | 244= ⌡ | 245= ÷ | 246= ≈ | 247= ° | 248= ∙ | 249= · | 250= √ | 251= ⁿ |
| 252= ² | 253= ■ | 254= ▪ | 255= | | | | | |



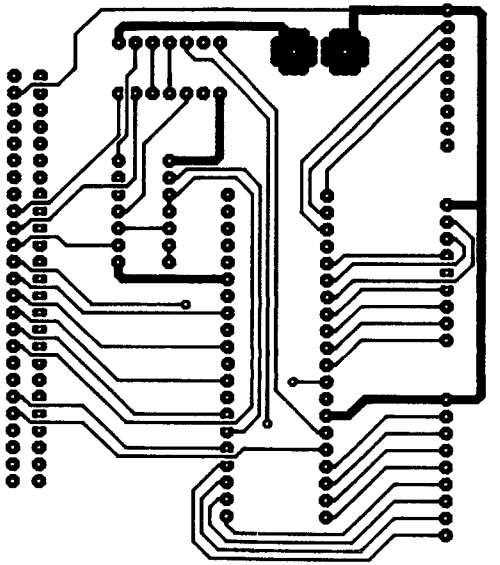



Buntgrafik erstellt mit dem Joycedrucker. (Vgl. S. 280f)

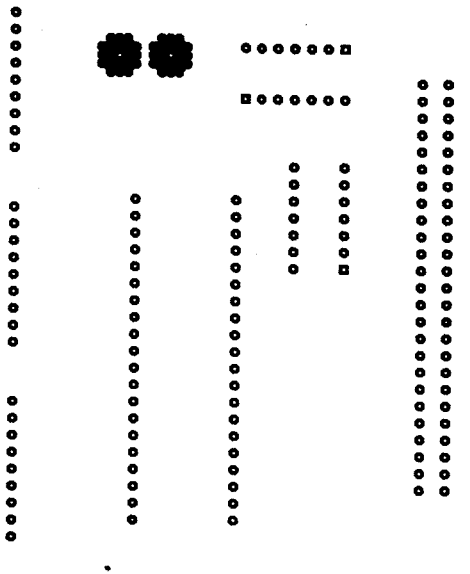


Detailfotos zum Hardware-Kapitel (vgl. Seiten 258 - 276)





Layout unten
(Vorlagen mit Tuscheseite)
(auf Kupferfolie legen ! )

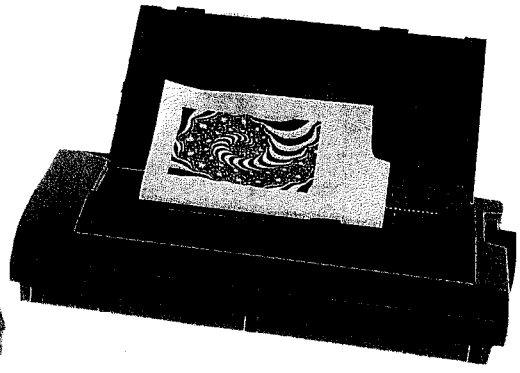


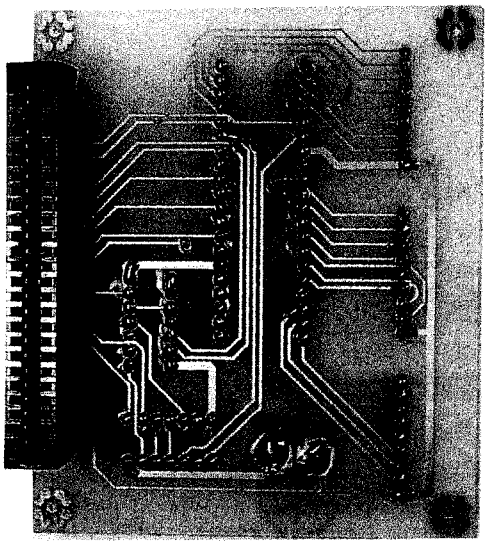
Loetstopmaske

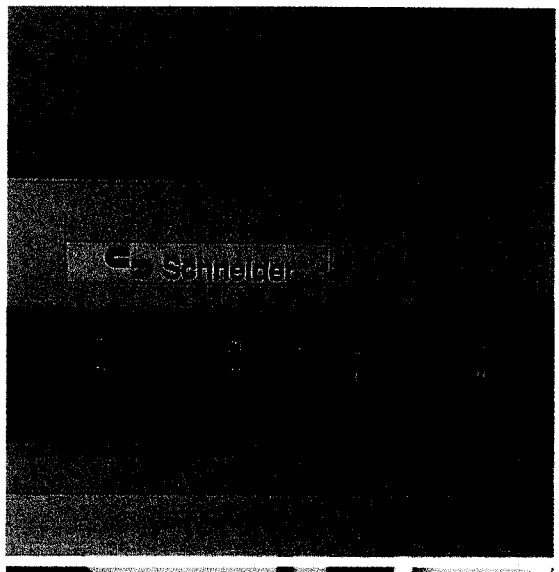
Layout oben
(gespiegelt)

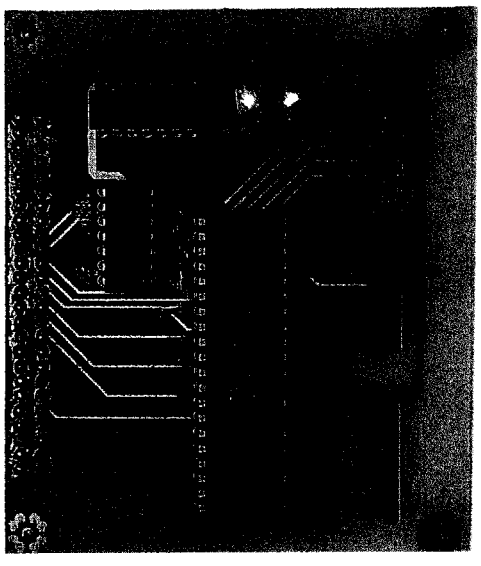


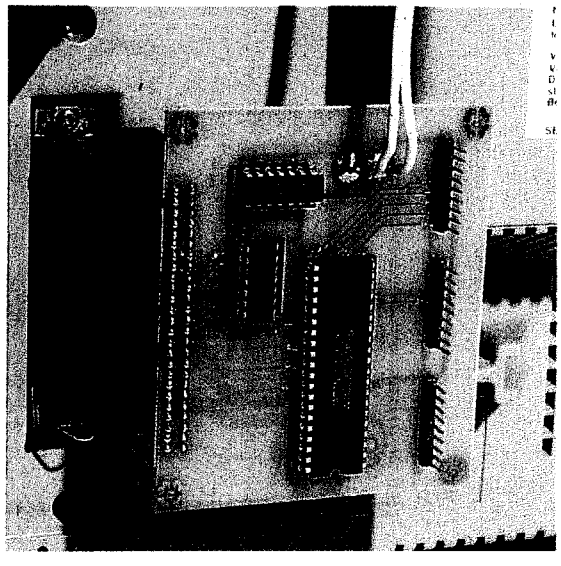

Bohrplan






C5 Schneider





**Norbert Kröger**
(Jg.1958)
schlug nach seinem Abitur die Laufbahn des Organisationsprogrammierers in der Finanzverwaltung ein. Heute schreibt er Programme, die die Finanzämter bei der Bewältigung anfallender Datenmengen unterstützen. Privat ein PCW-User der ersten Stunde, führte er auf seinem JOYCE u.a. das weiter, was er beruflich an Großrechenanlagen entwickelte. So galt sein Interesse hauptsächlich Jetsam, dem Instrument zur Datenverwaltung und Stiefkind der PCW-Handbücher.

**Winfried Kersting**
(Jg.1956)
Nutzte den JOYCE während seines Studiums der Philosophie, Germanistik und Literaturwissenschaft zunächst für die Textverarbeitung und die ED-Verwaltung von Buchbeständen. Daraus entwickelte sich sein generelles Interesse am PCW-System, speziell an der Erweiterung der Hardware und an verschiedenen Programmiersprachen. Die Erkenntnis, daß geeignete "Read-Ware" zum JOYCE fehlte, motivierte ihn, Gelerntes anzuwenden und vorliegendes Buch unter Berücksichtigung didaktischer Aspekte zu konzipieren.

**Bernd Graßhoff**
(Jg.1971)
ebenfalls JOYCE-User der ersten Stunde, baute schon in den unteren Klassen des Gymnasiums als versierter Elektroniker komplizierte Versuchsaufbauten und Anlagen für den Unterricht und Schulfeiern, die über den PCW gesteuert wurden. Seine Kenntnisse im EDV- und Elektronikbereich konnte er so weit ausbauen, daß sie mittlerweile im kommerziellen EDV-Geräteservice gefragt sind und sich diverse seiner Softwareprodukte auf dem Markt etablieren konnten.

## Zum Inhalt dieses Buches

Es ist das erklärte Ziel des vorliegenden Buches, den JOYCE/PCW als das zu präsentieren, was er de facto ist:
Ein vollwertiger Computer.

Das Konzept zur Erreichung dieses Ziels ist im Rahmen der handelsüblichen "Read-Ware" einmalig. Dem Anwender soll die Arbeit mit der zum System gelieferten Software nähergebracht und erleichtert werden, wobei sowohl der fortgeschrittene User als auch der absolute Laie an diesem Werk eine wertvolle Hilfe finden werden. Der erfahrene Programmierer erhält hier effektive Unterstützung durch die Befehlsübersichten und Befehlszusammenstellungen, die alphabetische Ordnung der BASIC-Kommandos, das ausführliche Stichwortverzeichnis und die vielen in die Tiefe gehenden Hinweise rund um die Programmierbeispiele. Anfänger erhalten durch kurze Beispielprogramme Einsicht in die Wirkungsweise der Befehle. Neben den zahlreichen Tips und Tricks in den einzelnen Programmen gestaltet sich das Buch für den erfahrenen User zu einem unentbehrlichen Nachschlagewerk, wobei die Hinweise zur Erweiterung und zum Ausbau der Peripherie noch keine Erwähnung gefunden haben. Mit einem Kapitel über LOGO, das auch dem erfahrenen Programmierer noch einiges Neue bieten kann, wird dem Anfänger ein leichter und verständlicher Einstieg in die Welt der Computersprachen inklusive Grafikausgabe geboten. Alle programmierbaren Vorgänge wurden zur Erleichterung für die Anwender auf eine Diskette gebracht, die zum Buch über den Verlag zu beziehen ist. Neben der Arbeit mit den Sprachen des JOYCE wird auch der Ausbau und die Verbesserung der Hardware gezeigt. Stichworte wie Bildschirminverter, Datenfernübertragung, Druckkopfreinigung, Buntdruck oder Sprachsynthesizer werden auch für den JOYCE-User zu vertrauten Begriffen. Hierauf aufbauend stellt eine Anleitung zum Selbstbau einer leistungsfähigen Schnittstelle eine attraktive Erweiterungsmöglichkeit dar.
Fazit: Das Konzept des informativen Nachschlagewerks plus leicht verständlicher Anleitung macht dieses Buch zu einem unentbehrlichen Helfer für jeden JOYCE-Besitzer.

Zum Buch erhältlich:
- Diskette 3" mit den abgedruckten Programmen
- Schnittstellenplatine zum problemlosen Aufbau

Bestellkarten im Buch


3440 Eschwege
ISBN 3-926177-02-0
Preis: 69,- DM