Der Joyce von Schneider ist für die reine Textverarbeitung, für die er eigentlich verkauft wird, viel zu schade. Zahlreiche Joyce-Anwender möchten darum selbst programmieren.

Martin Kotulla hat diese Situation voll erkannt. Am Anfang des Buches stellt er zunächst einige nützliche Programme aus seiner Feder vor. Über eine leichtverständliche Einführung in das CP/M-Betriebssystem führt der Weg hin zu den BIOS- und BDOS-Aufrufen. Eine detaillierte Behandlung der Mallard-Basic-Befehle rundet das interessante Angebot ab. Dabei geht es besonders um die Unterschiede zum Basic der Schneider-CPCs. Es wird dadurch dem Joyce-Eigner ermöglicht, aus dem umfangreichen Softwarereservoir der CPCs zu schöpfen.

Der Leser wird nach dem Durcharbeiten des Buches seinen Joyce nicht mehr als Buch mit sieben Siegeln betrachten. Der eigenen Kreativität steht nichts mehr im Wege.

9 783772 388613 03800
ISB N 3-7723-8861-2 DM +038.00

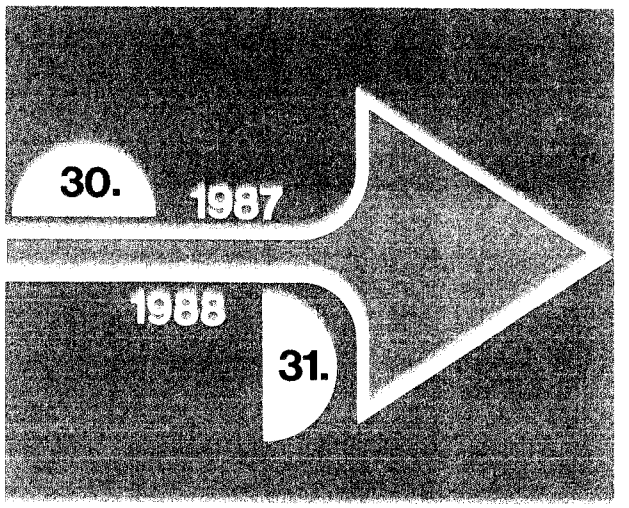
Franzis'

Franzis' Kotulla Den Joyce programmieren

Kotulla

Den Joyce programmieren

Raffinierte Programmier-Tricks



Franzis'

Martin Kotulla

Den Joyce programmieren

Raffinierte Programmier-Tricks

Franzis'

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Kotulla, Martin:
Den Joyce programmieren: raffinierte Programmier-Tricks/Martin Kotulla. - München: Franzis, 1987.
ISBN 3-7723-8861-2

Druck: Wiener Verlag, A-2325 Himberg
Printed in Austria : Imprimé en Autriche

ISB N 3-7723-8861-2

Vorwort

Der Joyce 8256/8512 ist ein Textsystem. Richtig? Falsch! Schneider verkauft diesen Computer zusammen mit einem Matrixdrucker und dem Textverarbeitungsprogramm Locoscript eigentlich nur als intelligente Schreibmaschine.

So kaufen sich denn auch viele Interessenten den Joyce vorrangig, um kleinere oder größere Schreibaufgaben mit ihm zu erledigen.

Nach einiger Zeit erinnert man sich aber einiger der sonstigen Fähigkeiten von Joyce. So ist er "ganz nebenbei" ein richtiger Bürocomputer, mit dem sich Daten verwalten, Buchhaltung betreiben oder Finanzberechnungen durchführen lassen.

Aber kaum jemand kauft sich einen Computer nur für die sogenannten "ernsthaften" Büroanwendungen. Wenn man schon einen Computer auf seinem Schreibtisch stehen hat, ist die Versuchung groß, sich auch einmal im Programmieren zu versuchen. Leider sind Informationen über Joyce rar, besonders wenn sie über das Niveau des Benutzerhandbuchs hinausgehen. Dieses Informationsdefizit soll das vorliegende Buch wenigstens etwas verkleinern. Denn in Joyce stecken Ressourcen, die kaum ein Programmierer je erkannt oder gar genutzt hat. So ist für viele, die in Basic programmieren können, schon die Verbindung von Maschinencode und Basic-Programmen ein Buch mit sieben Siegeln. Geht es dann gar um Dinge wie den Aufruf von systemspezifischen Routinen, weiß kaum noch jemand Bescheid - dank fehlender Informationen. So finden Sie in diesem Buch erstmalig für deutsche Leser eine genaue Darstellung der XBIOS-Systemroutine, die es nur beim Joyce gibt und die Dinge wie Tastaturumbelegung und Bildschirmansteuerung möglich machen.

Martin Kotulla, Nürnberg

Inhalt

1 Nützliche Programe in Mallard-Basic 9
1.1 Rund um den Kalender 11
1.2 Rechnen in allen Zahlensystemen. 28
1.2 Drucken Sie sich Geldscheine! 39

2 Das Betriebssystem CP/M 42
2.1 Was ist ein Betriebssystem? 42
2.2 Warum ein Standard-Betriebssystem? 43
2.3 Historie 43
2.4 So starten Sie CP/M Plus 46
2.5 Gehversuche in CP/M Plus 46
2.6 Die anderen residenten Kommandos 51
2.6.1 ERASE 52
2.6.2 RENAME 52
2.6.3 TYPE 53
2.6.4 DIRSYS 54
2.6.5 USER 55
2.7 Nachbrenner für residente Befehle 56
2.7.1 DIR.COM für übersichtliche Verzeichnisse 56
2.7.2 ERASE.COM schafft Sicherheit 61
2.7.3 RENAME.COM - Mehrdeutigkeit gestattet 62
2.7.4 TYPE.COM wartet auf Sie 63
2.8 Die CP/M-Dienstprogramme 63
2.8.1 PIP - der Alleskopierer 63
2.8.2 SHOW zeigt Systemparameter 66
2.8.3 SET - Systemfiles usw. 67

3 Die Systemroutinen von CP/M 69
3.1 Das Basic Disk Operating System (BDOS) 69
3.2 Das Basic Input/Output System (BIOS) 73
3.3 XBIOS - die Geheimnisse von Joyce 75
3.3.1 Diskettenorientierte XBIOS-Routinen 76
3.3.2 Programmierung der seriellen Schnittstelle 77
3.3.3 Steuerung der Bildschirmausgabe 77

3.3.4 Abfrage und Umbelegung der Tastatur 78
3.3.5 Sonstige Hilfsroutinen 78
3.3.6 Zugriff auf XBIOS-Routinen 79

4 Mallard-Basic - Unterschiede zu den CPCs 108
4.1 Bildschirmdarstellung 108
4.2 Graphikdarstellung 109
4.3 Ton- und Musikwiedergabe 109
4.4 Maschinensprache 109
4.5 Interrupts 110
4.6 Diskettenformat 110
4.7 Vergleichende Befehlsliste 111
4.8 Ersatzroutinen 111
4.9 Alphabetische Befehlsliste 112

5 Literatur 155

6 Disketten-Bezugsquelle 156

7 Sachverzeichnis 157

1 Nützliche Programme in Mallard-Basic

Für den Joyce sind von Haus aus zwei Programmiersprachen verfügbar. Da ist einerseits das oft als "Kindersprache" verschrieene Logo und andererseits das geliebt-gehaßte Basic. Mit Logo beschäftigen wir uns hier nicht weiter. Es ist sicher keine Kindersprache, aber jedenfalls auf Joyce viel zu langsam für sinnvolle Anwendungszwecke.

Wenden wir uns der zweiten beim Joyce verfügbaren Programmiersprache zu, Mallard-Basic. Basic hat seit seiner Erfindung Mitte der sechziger Jahre einen grandiosen Siegeszug in der Computerwelt angetreten. Gründe dafür sind sowohl seine leichte Bedienbarkeit als auch die Tatsache, daß Basic-Interpreter mit relativ wenig Speicherplatz auskommen. So hatten die ersten Basic-Versionen einen Umfang von fünf bis acht KByte und besaßen oft nur einen recht mageren Befehlsumfang.

Gegenüber diesen Programmiersprachen, die oft den bezeichnenden Namen "Tiny-Basic" (kleines, schmalbrüstiges Basic) trugen, ist Mallard-Basic eine gewaltige Weiterentwicklung.

Mallard-Basic basiert auf dem berühmten Microsoft-Basic. Dieser Interpreter war eine der ersten vernünftig für kommerzielle Anwendungen einsetzbaren Basic-Versionen. Microsoft-Basic wurde von Bill Gates geschrieben, der damit die sagenhafte Karriere vom jugendlichen Hobbyprogrammierer bis hin zum Chef des Software-Giganten Microsoft schaffte.

Doch auch Microsoft-Basic, kurz MBasic genannt, ist heute nicht mehr der letzte Stand der Software-Technologie. Im Vergleich dazu ist Mallard-Basic wohl die "Krönung" der unter CP/M lauffähigen Basic-Interpreter. Es bietet einen viel größeren Befehlsumfang, arbeitet Programme wesentlich schneller ab und ist gegenüber Fehlern des Benutzers erheblich nachsichtiger geworden.

So ist es etwa in MBasic ohne weiteres möglich, Diskettendateien anzulegen, deren Dateinamen Kleinbuchstaben enthalten. Das ist bei CP/M nicht vorgesehen und macht diese Dateien für andere Programme unlesbar. Mallard-Basic hingegen wandelt Kleinbuchstaben automatisch in Großbuchstaben um.

Ein weiterer Vorzug von Mallard-Basic sind die Befehle zur Arbeit mit Diskettendateien in verschiedenen Organisationsmethoden. Neben der üblichen Methode der sequentiellen Abspeicherung, bei der die Datensätze

einfach hintereinander auf der Diskette abgelegt werden, gibt es noch die relative Dateiverwaltung, bei der Basic-Programme anhand einer Satznummer direkt einzelne Datensätze schreiben und lesen können. Das Glanzstück ist aber die Jetsam-Dateiverwaltung. Sie ist eine Mischung der sequentiellen und der relativen Speicherungsmethode. Hier werden die einzelnen Datensätze nicht über ihre Satznummer angesprochen, sondern über einen "Schlüssel". Das ist eine Zeichenkette, die in einer anderen Datei abgelegt wird und direkt mit den Datensätzen korrespondiert.

Eine weitere angenehme Eigenschaft von Mallard-Basic betrifft die Verkettung von Basic-Programmen. Im Hauptspeicher läßt Mallard-Basic nur noch etwa 30 KByte für das eigentliche Basic-Programm frei. Damit bleiben viele KByte an Speicher ungenutzt. Sie sind beim Joyce als RAM-Diskette organisiert. Darunter versteht man einen Speicherbereich, der sich so verhält, als wäre er ein echtes Diskettenlaufwerk. Die RAM-Disk ist beim Joyce auf das Laufwerk M: eingestellt. Eine solche RAM-Diskette kann natürlich viel schneller arbeiten als ihre mechanischen Brüder.

So ist es durchaus vorstellbar, ein Mallard-Basic-Programm in verschiedene Einzelsegmente zu zerlegen, die sich gegenseitig mit dem CHAIN-Befehl aufrufen:

```
CHAIN "M: FIBU.BAS"
CHAIN "M: HAUPT.BAS"
CHAIN "M: TEXT.BAS"
```

Dadurch, daß die Programme in der RAM-Disk stehen, kommt es kaum zu Zeitverlusten gegenüber einer Basic-Version, die alle Programmteile gleichzeitig im Hauptspeicher halten könnte.

Doch ganz so perfekt, wie das vielleicht bisher aussah, ist Mallard-Basic doch nicht. Wie wäre es sonst zu erklären, daß keinerlei Befehle vorgesehen wurden, die hochauflösende Grafik darstellen können? Von der Hardware her ist der Joyce ja dazu in der Lage, wie der Logo-Interpreter DR LOGO oder auch Grafikprogramme wie DR DRAW und DR GRAPH sowie der Basic-Compiler CBASIC (CB80) eindeutig beweisen.

Im folgenden finden Sie drei Basic-Programme. Diese sind sicher nicht der Weisheit letzter Schluß, zeigen aber, daß man auch in Mallard-Basic sauber programmieren kann, und bieten sogar noch einen gewissen Nutzeffekt. Die Programme zeigen verschiedene Programmiertechniken, die Ihnen als Anregung dienen sollen. Sie können Teile von ihnen auch als Grundstock für den Aufbau Ihrer eigenen Bibliothek von Programmodulen verwenden. Exemplarisch werden hier Bildschirmansteuerung, Menüauswahl,

Datumsberechnung, Umleitung der Bildschirmausgaben auf den Drucker und einige andere Dinge gezeigt.

1.1 Rund um den Kalender

Haben Sie schon einmal nach einem Kalender Ihres Geburtsjahres gesucht, weil Sie wissen wollten, an welchem Wochentag Sie geboren wurden? Planen Sie bereits Ihren Urlaub und benötigen Sie dazu die Wochentage zu einigen Daten? Das Programm KALENDER berechnet für Sie einzelne Wochentage und druckt auch den Kalender eines ganzen Jahres aus.

Sobald Sie das Programm abgetippt oder von der Diskette geladen haben, können Sie es mit RUN starten. Der Bildschirm wird gelöscht und es erscheint ein Menü, daß aus drei Auswahlpunkten besteht:

```
WOCHENTAG BERECHNEN
KALENDER AUSDRUCKEN
PROGRAMM BEENDEN
```

Mit den Tasten Plus (+) und Minus (-) können Sie einen hellen Balken über die einzelnen Menüpunkte hinwegbewegen. Sobald sich der Balken über dem gewünschten Unterprogramm befindet, drücken Sie kurz auf die Leertaste. Das Programm wird sofort aufgerufen.

```
WOCHENTAG BERECHNEN
```

In diesem Programmteil können Sie ein bestimmtes Datum angeben und den Computer den zugehörigen Wochentag berechnen lassen. Sofern Sie nicht gerade ein Jahr vor der Kalenderreform von Papst Gregor XIII. (1584) einzugeben versuchen, meldet KALENDER.BAS unmittelbar darauf den Wochentag.

Versuchen Sie es doch einmal selbst! Nehmen wir als Beispiel das Weihnachtsfest 1986 her, den 25. 12. 1986. Das Programm fragt die Daten im Dialog ab:

```
Tag im Monat? 25
Welcher Monat? 12
Welches Jahr? 1986


Der 25. 12. 1986 ist ein Donnerstag.
Bitte drücken Sie eine Taste!
```

Sie können statt der Monatsnummer 12 auch den Monatsnamen "Dezember" angeben. Dabei unterscheidet das Programm nicht zwischen Kleinbuchstaben und Großschreibung. Außerdem sind nur die ersten drei Buchstaben signifikant, also von Bedeutung. Deshalb stellen allen folgenden Eingaben gültige Monatsnamen dar:

```
Dezember
DEZEMBER
dezember
dEZEMBER
DEZ
dez
dez.....
```

Auch dürfen Sie die vierstellige Jahreszahl "1986" auf die Kurzform "86" bringen. Das macht natürlich nur Sinn, wenn Sie sich im 20. Jahrhundert bewegen.

Wenn Sie partout den 29. Februar 1986 oder den 1. April 4010 oder ähnliche unsinnige Daten berechnen wollen, ruft das Programm Sie schnell zur Ordnung:

```
Tag im Monat? 29
Welcher Monat? 2
Welches Jahr? 1986


Fehler bei der Eingabe!
Bitte drücken Sie eine Taste!
```

Damit ist der Menüpunkt "Wochentag berechnen" hoffentlich ausreichend erklärt. Wir können damit zum nächsten Unterprogramm übergehen...

KALENDER AUSDRUCKEN

Dieser Teil des Programms erstellt den vollständigen Kalender eines ganzen Jahres. Das kann für Unternehmer recht nützlich sein, wenn die Buchführung ansteht. So ein Kalender kann aber auch dazu dienen, eine Strichliste der noch bis zum Beginn des ersehnten Urlaubs oder der Schulferien verbleibenden Arbeitstage zu erstellen...

Das Programm benötigt eine Jahresangabe und stellt Ihnen dann noch die Frage, ob der Kalender auf dem Bildschirm oder auf dem Drucker aufgelistet werden soll:

```
Für welches Jahr? 1987
Druckerausgabe? N
```

Auch hier kann die Jahreszahl auf zwei Stellen verkürzt werden. Sie ersparen sich ein bißchen Schreibarbeit, wenn Sie statt "1987" nur "87" eintippen.

Eine typische Kalenderliste sieht etwa so aus (hier als Beispiel 1987):

```
Januar:
So-Mo-Di-Mi-Do-Fr-Sa

                01 02 03
04 05 06 07 08 09 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31

Februar:
So-Mo-Di-Mi-Do-Fr-Sa

01 02 03 04 05 06 07
08 09 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
.
.
.
```

Zum letzten Punkt des Hauptmenüs ...

PROGRAMM BEENDEN

... gibt es eigentlich nicht allzuviel zu sagen. Nach einer Sicherheitsabfrage, bei der Sie Ihre Entscheidung zurücknehmen können, bricht der Computer die Abarbeitung des Programms ab.

Hier das Listing von KALENDER.BAS:

```
100 '  ******************************
110 ' ********************************
120 ' *                              *
130 ' *     RUND UM DEN KALENDER     *
140 ' *                              *
150 ' ********************************
160 '  ******************************
170 '
180 ' DEFINITIONEN ZUR BILDSCHIRMSTEUERUNG ***********
190 WIDTH 80
200 cls$=CHR$(27)+"E"+CHR$(27)+"H"
210 german$=CHR$(27)+"2"+CHR$(2)
220 invon$=CHR$(27)+"p"
230 invoff$=CHR$(27)+"q"
240 curon$=CHR$(27)+"e"
250 curoff$=CHR$(27)+"f"
260 block$=invon$+" "+invoff$
270 cr$=CHR$(13)
280 csr$=CHR$(27)+"Y"
290 DEF FNlocate$(s,z)=csr$+CHR$(z+31)+CHR$(s+31)
300 ' DEFINITIONEN ZUM KALENDERPROGRAMM **************
310 DIM month$(12),days.in.month(12)
320 DIM total.days(12),weekday$(7)
330 DEF FNm(y)=year MOD y
340 year1=365
350 ' MONATE UND ZUGEHOERIGE TAGE EINLESEN ***********
360 FOR i=1 TO 12:READ month$(i),days.in.month(i)
370 total.days(i)=total.days(i-1)+days.in.month(i-1)
380 total.days(i-1)=total.days(i-1)+1:NEXT i
390 total.days(12)=total.days(12)+1
400 FOR i=0 TO 6:READ weekday$(i):NEXT i
410 ' TITELBILD UND MENÜ *****************************
420 PRINT cls$;german$;cr$;
430 PRINT FNlocate$(25,1);
440 PRINT CHR$(150);STRING$(40,154);CHR$(156)
450 PRINT FNlocate$(25,2);
460 PRINT CHR$(149);SPACE$(40);CHR$(149)
470 PRINT FNlocate$(25,3);
480 PRINT CHR$(149);SPACE$(10);"RUND UM DEN KALENDER";
490 PRINT SPACE$(10);CHR$(149)
500 PRINT FNlocate$(25,4);
510 PRINT CHR$(149);SPACE$(40);CHR$(149)
520 PRINT FNlocate$(25,5);
530 PRINT CHR$(147);STRING$(40,154);CHR$(153)
540 ' MENÜ AUSGEBEN **********************************
550 num=3:row=10:col=10:choice=1
560 ch$(1)="  WOCHENTAG BERECHNEN"+SPACE$(20)
570 ch$(2)="  KALENDER AUSDRUCKEN"+SPACE$(20)
580 ch$(3)="  PROGRAMM BEENDEN"+SPACE$(23)
590 PRINT FNlocate$(1,row);curoff$;CHR$(13);
600 FOR i=1 TO num:PRINT TAB(25);ch$(i):PRINT:NEXT i
610 PRINT FNlocate$(20,23);"Bitte wählen Sie mit ";
620 PRINT CHR$(208);"+";CHR$(212);", ";CHR$(208);
630 PRINT "-";CHR$(212);
640 PRINT " und der Leertaste!"
```



```
650 PRINT FNlocate$(25,choice*2+row-3);
660 PRINT STRING$(41,95)
670 PRINT TAB(25);invon$;ch$(choice);invoff$
680 a$=INKEY$
690 IF a$=CHR$(32) THEN 790 ' Leertaste gedrückt
700 IF a$="" OR (a$<>"+" AND a$<>"-") THEN 680
710 PRINT FNlocate$(25,choice*2+row-3);SPACE$(41)
720 PRINT TAB(25);ch$(choice)
730 IF a$="-" THEN choice=choice-1
740 IF a$="+" THEN choice=choice+1
750 IF choice<1 THEN choice=num
760 IF choice>num THEN choice=1
770 PRINT FNlocate$(25,choice*2+row-3);STRING$(41,95)
780 PRINT TAB(25);invon$;ch$(choice);invoff$:GOTO 680
790 PRINT curon$;
800 ON choice GOTO 820,1210,1610 ' Subprogramm aufrufen
810 ' 1- Wochentag berechnen *************************
820 PRINT cls$;STRING$(79,95)
830 PRINT invon$;ch$(1);SPACE$(38);invoff$:PRINT
840 PRINT " Hier können Sie ein Datum eingeben ";
850 PRINT "und den Computer den zugehörigen"
860 PRINT:PRINT " Wochentag berechnen lassen."
870 PRINT:PRINT STRING$(79,95):PRINT
880 PRINT:LINE INPUT " Tag im Monat?   ",day$
890 day=VAL(day$)
900 PRINT:LINE INPUT " Welcher Monat?  ",month$
910 month=VAL(month$)
920 PRINT:LINE INPUT " Welches Jahr?   ",year$
930 year=VAL(year$)
940 PRINT:PRINT
950 ' Prüfen, ob Monatseingabe "1" oder "Januar"
960 IF month<>0 THEN 1050
970 FOR i=1 TO 12
980     m1$=LEFT$(UPPER$(month$),3)
990     m2$=LEFT$(UPPER$(month$(i)),3)
1000    IF m1$=m2$ THEN 1040
1010 NEXT i
1020 ' Nicht gefunden
1030 PRINT beep$;:GOTO 1100
1040 month$=MID$(STR$(i),2):month=i
1050 GOSUB 1800 ' Tag berechnen
1060 year$=MID$(STR$(year),2)
1070 month$=MID$(STR$(month),2)
1080 day$=MID$(STR$(day),2)
1090 IF errf<>1 THEN 1130
1100    PRINT beep$;invon$;" Fehler bei der Eingabe! ";
1110    PRINT invoff$;
1120    GOTO 1150
1130 PRINT " Der "+day$+"."+month$+".";
1140 PRINT year$+" ist ein "+dayname$+"."
1150 WHILE INKEY$<>"":WEND ' Tastaturpuffer löschen
1160 PRINT FNlocate$(1,24);beep$;invon$;" Bitte ";
1170 PRINT "drücken Sie eine Taste! ";invoff$;
1180 WHILE INKEY$="":WEND
```

```
1190 PRINT cls$;:GOTO 420
1200 ' 2- Kalender ausdrucken *************************
1210 PRINT cls$;STRING$(79,95)
1220 PRINT invon$;ch$(2);SPACE$(38);invoff$:PRINT
1230 PRINT:PRINT " Hiermit können Sie wahlweise auf ";
1240 PRINT "dem Bildschirm oder Drucker den"
1250 PRINT:PRINT " Kalender eines Jahres ausgeben."
1260 PRINT:PRINT STRING$(79,95):PRINT
1270 PRINT " Für welches Jahr? ";
1280 LINE INPUT year$:year=VAL(year$)
1290 IF year<100 THEN year=year+1900
1300 PRINT:PRINT " Druckerausgabe? ";block$;
1310 in$=UPPER$(INKEY$):IF in$="" THEN 1310
1320 IF in$="J" THEN dv=8:GOSUB 2050:GOTO 1360
1330 IF in$="N" THEN dv=0:GOTO 1350
1340 PRINT beep$;:GOTO 1310
1350 PRINT cls$;
1360 WIDTH 22
1370 FOR month=1 TO 12
1380  PRINT:PRINT month$(month);":":PRINT
1390  PRINT "So-Mo-Di-Mi-Do-Fr-Sa":PRINT
1400  day=1:GOSUB 1800
1410  IF weekday=0 THEN PRINT "01 ";
1420  IF weekday<>0 THEN PRINT TAB(weekday*3+1);"01 ";
1430  IF month=2 AND sjahr=1 THEN lastday=29:GOTO 1450
1440  lastday=days.in.month(month)
1450  FOR i=2 TO lastday
1460   dy$=MID$(STR$(i),2)+" "
1470   IF LEN(dy$)=2 THEN dy$="0"+dy$
1480   PRINT dy$;
1490  NEXT i
1500  PRINT
1510 NEXT month
1520 IF dv=8 THEN WIDTH 80:GOSUB 2120:GOTO 420
1530 WHILE INKEY$<>"":WEND ' Tastaturpuffer löschen
1540 WIDTH 80
1550 PRINT STRING$(41,95)
1560 PRINT invon$;" Bitte drücken Sie ";
1570 PRINT "eine Taste!"+SPACE$(10)+invoff$;
1580 WHILE INKEY$="":WEND ' Auf Tastendruck warten
1590 PRINT cls$;:GOTO 420
1600 ' 3- Programm beenden ****************************
1610 PRINT cls$;STRING$(79,95)
1620 PRINT invon$;ch$(3);SPACE$(38);invoff$:PRINT
1630 PRINT:PRINT " Wollen Sie die Arbeit mit diesem";
1640 PRINT " Programm wirklich beenden? ";
1650 PRINT block$;
1660 in$=UPPER$(INKEY$):IF in$="" THEN 1660
1670 IF in$="J" THEN PRINT cls$;:END
1680 IF in$<>"N" THEN PRINT beep$;:GOTO 1660
1690 PRINT cls$;:GOTO 420
```



```
1700 ' ************************************************
1710 ' *                                              *
1720 ' *         Unterprogramm: Tag ausrechnen        *
1730 ' *         ------------------------------       *
1740 ' *                                              *
1750 ' *        Eingangsvariable: day,month,year      *
1760 ' *          Ausgangsvariable: dayname$          *
1770 ' *                                              *
1780 ' ************************************************
1790 ' Überprüfung der Daten -------------------------
1800 IF year<100 THEN year=year+1900
1810 IF year<1584 THEN errf=1:RETURN ELSE errf=0
1820 IF year>4000 THEN errf=1:RETURN
1830 IF month<1 OR month>12 THEN errf=1:RETURN
1840 IF day<1 THEN errf=1:RETURN
1850 IF day>days.in.month(month) THEN errf=1:RETURN
1860 ' Schaltjahr berücksichtigen --------------------
1870 IF FNm(4)=0 THEN sjahr=1 ELSE sjahr=0
1880 IF FNm(100)=0 AND FNm(400)<>0 THEN sjahr=0
1890 IF month<>2 THEN 1930
1900 IF sjahr=0 AND day>27 THEN errf=1:RETURN
1910 IF sjahr=1 AND day>28 THEN errf=1:RETURN
1920 ' Tag berechnen ---------------------------------
1930 passed=year1*year+year\4-year\100+year\400-year1
1940 passed=passed+total.days(month)+day-1
1950 IF month<=2 AND sjahr=1 THEN passed=passed-1
1960 weekday=passed-7*INT(passed/7)
1970 dayname$=weekday$(weekday):RETURN
1980 ' Daten zur Wochentagsberechnung ****************
1990 DATA Januar,31,Februar,28,März,31,April,30,Mai
2000 DATA 31,Juni,30,Juli,31,August,31,September,30
2010 DATA Oktober,31,November,30,Dezember,31
2020 DATA Sonntag,Montag,Dienstag,Mittwoch
2030 DATA Donnerstag,Freitag,Samstag
2040 ' BILDSCHIRMAUSGABE AUF DEN DRUCKER BRINGEN *****
2050 DATA &H0E,&H31,&H11,&H89,&H00,&HCD,&H05
2060 DATA &H00,&HC9,&H24,&HFE,&H00,&H40
2070 RESTORE 2050
2080 FOR i=&H80 TO &H8C:READ a:POKE i,a:NEXT i
2090 adr=&H80:CALL adr
2100 RETURN
2110 ' BILDSCHIRMAUSGABE WIEDER NORMALISIEREN ********
2120 DATA &H0E,&H31,&H11,&H89,&H00,&HCD,&H05
2130 DATA &H00,&HC9,&H24,&HFE,&H00,&H80
2140 RESTORE 2120
2150 FOR i=&H80 TO &H8C:READ a:POKE i,a:NEXT i
2160 adr=&H80:CALL adr
2170 RETURN
2180 END ' -------------------------------------------
```

Für Programmierer:

Am Programmcode sind mehrere Lösungen für Einzelprobleme sehr interessant. Dazu gehören die standardisierte Menüsteuerung, die Routine zur Berechnung der Wochentage und das Unterprogramm, das die Bildschirmausgabe auf den Drucker umlegt.

Alles der Reihe nach:

1. Die Menüsteuerung

Es gibt verschiedene Methoden der Benutzerführung. Neben den heute etwas aus der Mode gekommenen Funktionstasten und Control-Tasten und den hochaktuellen, allerdings auf 8-Bit-Rechnern wie dem Joyce nur schwer realisierbaren, grafischen Benutzeroberflächen à la GEM oder Microsoft-Windows verwenden unzählige Programme die Menüsteuerung.

Hier hat der Anwender alle Auswahlmöglichkeiten auf einen Blick. Die Anwahl eines Menüpunkts kann auf verschiedene Arten geschehen. Die wohl komfortabelste ist es, mit den Cursortasten oder den Plus- und Minus-Tasten einen invers dargestellten Balken über die Menüpunkte hinwegzubewegen. Auf dieses Verfahren greift auch KALENDER.BAS zurück.

Damit Sie aber auch etwas für eigene Programme haben, ist die Steuerroutine programmunabhängig geschrieben. Indem Sie nur einige wenige Zeilen abändern, können Sie die Menüsteuerung auch in Ihre eigenen Programme integrieren.

In den Zeilen 550 bis 800 ist die komplette Routine enthalten. Dabei befinden sich in der Programmzeile 550 entscheidende Variablendefinitionen:

- "num" enthält die Zahl der anwählbaren Menüpunkte. Da das Menü von KALENDER.BAS drei Auswahlzeilen zeigt, gilt num=3.

- "row" und "col" werden mit der Cursorposition der ersten Auswahlzeile geladen. In "row" steht die Cursorzeile, in "col" ("Column") die Cursorspalte.

- "choice" ist auf diejenige Menüzeile festgelegt, die beim Programmstart als erste invers dargestellt wird. Das ist hier "Wochentag berechnen". Ändern Sie die Zeile in "choice=2" ab, so befindet sich der helle Balken beim Starten von KALENDER.BAS über der Zeile "KALENDER ausdrucken".

In den Zeilen 560 bis 580 werden die Texte definiert, die KALENDER.BAS als Menüpunkte auf dem Bildschirm anzeigt. Dabei gilt es

zu beachten, daß jede Zeichenkette mit Leerzeichen auf eine Länge von 41 Zeichen aufgefüllt werden sollte.

In der Zeile 590 wird vom Programm der Cursor positioniert. Die Formulierung

```
PRINT FNlocate$(1,row);curoff$;
```

würde hier nicht zum Erfolg führen. Dann würde nämlich die Positionierung des Cursors in der folgenden Zeile nicht korrekt durchgeführt. Denn Mallard-Basic zählt bei der TAB-Funktion auch Bildschirm-Steuerzeichen als normale Symbole mit, was natürlich die Bildschirmgestaltung vollkommen durcheinander bringt - bis Computer und Programme eine gewisse Eigenintelligenz entwickeln, vergeht halt doch noch einige Zeit ...

Abhilfe schafft es da, einen Wagenrücklauf an den Bildschirm zu senden. Das ist für Mallard-Basic nämlich der Auslöser, den internen Spaltenzähler auf Null zurückzusetzen.

Die Zeile 600 weist den Computer an, das Menü auf dem Bildschirm auszurollen. Die Programmzeilen 600 bis 640 sorgen für eine Hilfestellung für ungeübte Benutzer. Denn sie zeigen an, wie man den Auswahlbalken mit Plus, Minus und der Leertaste steuern kann.

In der Zeile 680 besorgt KALENDER.BAS die Tastaturabfrage über die Basic-Funktion INKEY$. Diese wartet nicht solange, bis eine Taste gedrückt wird, sondern gibt gegebenenfalls auch einen Leerstring zurück. Deshalb muß dieser Fall in der Zeile 700 ausgeschlossen werden. Dort wird auch gleich festgestellt, ob eine andere Taste als die zulässigen "+" und "-" gedrückt wurde. Ob der Benutzer die Leertaste betätigt hat, wird hingegen schon in 690 geprüft.

Je nachdem, ob der Benutzer den Balken nach oben oder unten bewegt, wird der Zähler in der Variablen "choice" um den Wert 1 vermindert oder erhöht. Der angewählte Menüpunkt wird dann mit Hilfe der Basic-Zeilen 770 und 780 hell dargestellt.

Ja, und die Zeile 800 besorgt den Sprung in das vom Benutzer gewünschte Unterprogramm.

Wenn Sie die Menüsteuerung für ein Programm abändern wollen, beschränken sich diese Änderungen auf die Zeilen 550 bis 580 sowie 800.

2. Die Datumsberechnung

Jetzt wird es ein kleines bißchen mathematisch. Aber keine Sorge, Sie brauchen kein Uni-Absolvent sein, um meinen Ausführungen folgen zu können.

Das erste Problem stellt sich bei der Frage, ob eine angegebene Jahreszahl ein Schaltjahr repräsentiert. Dabei gelten in der "Kalendermathematik" folgende Regeln:

a) Alle vier Jahre ist ein Schaltjahr.

b) Von dieser Regel sind Jahrhundertanfänge ausgenommen, etwa 1700, 1800 oder 1900. Diese Jahre sind keine Schaltjahre!

c) Von der Regel b. sind aber wiederum diejenigen Jahrhundertanfänge ausgenommen, die ganzzahlig durch 400 teilbar sind. Damit handelt es sich bei 1600 und 2000 doch um Schaltjahre.

Diese Forderungen sind in den Zeilen 1870 und 1880 in Basic-Code gefaßt. Ohne die Funktion FNm würden diese Zeilen wie folgt aussehen:

```
1870 IF year MOD 4=0 THEN sjahr=1 ELSE sjahr=0
1880 IF year MOD 100=0 AND year MOD 400<>0 THEN sjahr=0
```

In einer "Pseudo-Programmiersprache" ließe sich das so darstellen:

```
WENN jahr DURCH 4 TEILBAR
DANN schaltjahr=WAHR
SONST schaltjahr=FALSCH

WENN jahr DURCH 100 TEILBAR
UND jahr NICHT DURCH 400 TEILBAR
DANN schaltjahr=FALSCH
SONST schaltjahr=WAHR
```

Ein anderes Problem stellt die Form des Datums selbst dar. Denn mit Jahren, Monaten und Tagen läßt sich nun einmal sehr schwer rechnen. Beispielsweise ist es nahezu unmöglich, die Differenz zwischen zwei Datumsangaben zu errechnen.

Einen Ausweg aus dem Dilemma weist die Umwandlung in einen Zahlenwert auf. Hier gilt das Datum von Christi Geburt, dem Beginn unserer Zeitrechnung, als Tag 0. Darauf folgen Tag 1, Tag 2, Tag 3 ... Tag 5000 usw. Hier ist die Tagesdifferenz zwischen dem zwanzigsten und dem fünzigsten Tag eine einfache Subtraktion.

Um von einem gegebenen Datum auf die Zahl der Tage zu kommen, muß man die Zahl der Tage in einem Jahr mit den vergangenen Jahren multiplizieren:

```
TAGE=365*JAHRE
```

Nicht vergessen darf man den einen Tag, den jedes Schaltjahr mit dem 29. Februar zusätzlich bringt:

```
TAGE=TAGE+(JAHRE/4)
```

Im Programm wird die ganzzahlige Division angewendet, damit kein unnötiger Nachkomma-Wert entsteht.

Laut obigen Regeln müssen aber die besagten Jahrhundertanfänge, die nicht durch 400 teilbar sind, wieder abgezogen werden. Hier zieht man einfach alle Jahrhundertanfänge ab:

```
TAGE=TAGE-(JAHRE/100)
```

Und man fügt die Jahrhundertanfänge, deren Jahreszahl durch 400 teilbar ist, wieder dazu:

```
TAGE=TAGE+(JAHRE/400)
```

Jetzt müssen nur noch die Tage der im aktuellen Jahr vorangehenden Monate addiert werden. Diese hält das Programm der Einfachheit halber in einem Datenfeld, das es sich in den Zeilen 360 bis 390 aus den Monatslängen in den DATA-Zeilen errechnet hat. Und natürlich darf man nicht den Tag vergessen, der in der Variablen "day" übergeben wurde:

```
TAGE=TAGE+IM_JAHR_VERGANGENE_TAGE
TAGE=TAGE+IM_MONAT_VERGANGENE_TAGE
```

Alle diese Berechnungen faßt KALENDER.BAS in den Zeilen 1930 und 1940 zusammen. Außerdem zieht das Programm bei Bedarf in Schaltjahren noch einen Tag ab.

Entscheidend ist jetzt die Formel in der Zeile 1960. Hier errechnet der Computer aus jedem beliebigen Tageswert eine Zahl zwischen 0 und 6. Diese korrespondiert direkt mit den Wochentagen Sonntag, Montag, Dienstag usw. bis zum Samstag.

Der letzte Schritt ist dann die Umwandlung des Zahlenwerts in einen Textstring, der den Namen des Wochentags enthält. Das führt Ihr Joyce in der Programmzeile 1970 durch.

Auch die Routine zur Datumsberechnung ist modular geschrieben. Sie können Sie deshalb in Ihre Programme übernehmen. Die Zeilen 300 bis 400 sollten Sie Ihrem Hauptprogramm voranstellen und die Basic-Zeilen 1700 bis 2030 als Unterprogramm an das Programm anhängen.

Als Eingangswerte erwartet die Subroutine bei ihrem Aufruf in den Variablen "day", "month" und "year" das zu bestimmende Datum. Sie gibt zwei Variablen zurück. In "dayname$" befindet sich die gesuchte Zeichenkette mit dem Wochentag. Sie sollten aber auch das Fehler-Flag in "errf" überprüfen. Haben Sie ein fehlerhaftes Datum angegeben, für das kein Wochentag berechnet werden kann, ist "errf" auf den Wert 1 gesetzt.

Dieser Fall tritt zum Beispiel bei Jahresangaben vor 1584 oder nach 4000 sowie bei unsinnigen Monats- und Tagesangaben ein.

3. Umlenkung der Bildschirmausgabe

Ganz ohne Maschinensprache kommen wir hier nicht weiter. Die Aufgabenstellung besteht darin, den Basic-Interpreter beziehungsweise das Betriebssystem zu veranlassen, daß bei PRINT alle Bildschirmausgaben an den Drucker geschickt werden.

Benötigt wird die Umlenkung der Ausgabe vom Programmteil, der den Kalender ausdruckt. Denn dies soll ja wahlweise auf dem Bildschirm oder auf dem Matrixdrucker von Joyce geschehen.

Natürlich könnte man alle in Frage kommenden Programmzeilen so umschreiben:

```
IF drucker=0 THEN PRINT "HALLO"
             ELSE LPRINT "HALLO"
```

Doch mit der Zeit wird es ganz schön entnervend, solche Konstruktionen eintippen zu müssen. Ganz abgesehen davon, daß diese Methode nicht gerade ökonomisch mit dem vorhandenen Speicherplatz wirtschaftet.

Worin die Lösung besteht, kann man mit etwas Nachdenken durchaus herausfinden. Sie kennen sicher die Möglichkeit von CP/M, mit Control-P

den Drucker zur Bildschirmausgabe dazuzuschalten. Der Versuch, den Steuercode Control-P an den Bildschirm zu senden, schlägt aber fehl.

Eine andere - weit weniger bekannte - Methode der Ausgabeumlenkung bietet das CP/M-Dienstprogramm DEVICE.COM an. Dieses erlaubt es dem Benutzer, die Zuordnung der logischen und physikalischen Geräte neu festzulegen.

Was ist das nun schon wieder? CP/M Plus kennt verschiedene zeichenorientierte Peripheriegeräte. Dazu zählen der Bildschirm, die Tastatur, der Drucker und eventuell eine serielle Schnittstelle. Diese Geräte erhalten verschiedene Kurzbezeichnungen:

CON: Bildschirm und Tastatur ("Konsole")
LST: Drucker
AUX: Hilfsgeräte ("Auxiliary Device")

Nun ist es aber keineswegs immer vorgeschrieben, daß die Tastatureingaben wirklich von der Tastatur geholt werden. Stellen Sie sich zum Beispiel einen CP/M-Rechner vor, der von einem Terminal über die serielle Schnittstelle angesprochen wird. Dann holt sich CP/M die Tastatureinaben von dieser Schnittstelle und übergibt sämtliche Bildschirmausgaben wieder dem Zielgerät über die Schnittstelle.

Um diese Flexibilität zu erreichen, sind CON:, LST: und AUX: nicht auf bestimmte Hardware-Einheiten festgelegt. Stattdessen gibt es eine Anzahl von physikalischen oder tatsächlichen Geräten:

CRT Bildschirm ("Cathode Ray Tube")
LPT Drucker ("Line Printer")
NUL Leergerät

Weitere Namen für physikalische Geräte können vom Computerhersteller festgelegt werden. Denkbar sind etwa RS232 (Schnittstelle), JOYST (Joystick oder Maus) und LASER (Laserdrucker).

Normalerweise gelten Standardzuordnungen der logischen zu den physikalischen Geräten. So ist CON: mit CRT verbunden und LST: mit LPT. DEVICE ändert das ab:

```
A>DEVICE LST:=CRT
```

Ab diesem Zeitpunkt werden alle Druckerausgaben nicht mehr an den Drucker, sondern an den Bildschirm geschickt. Vielleicht ahnen Sie schon,

was passiert, wenn Sie mit Control-P den Bildschirm und Drucker parallel schalten ...

Die Fähigkeit der Umlenkung der Ein- und Ausgabe ist tief im CP/M-Betriebssystem verankert. Das Betriebssystem arbeitet mit einem 100 Bytes langen Speicherbereich, in dem wichtige Systeminformationen bereitgehalten werden. So finden sich dort die Adresse des Diskettenpuffers (DMA-Adresse), die ausgewählte Benutzernummer (User-Bereich), die Nummer der angemeldeten Diskettenstation und eben auch Informationen über die Zuordnung der Peripheriegeräte. In den Bytes hex 24 und hex 25 des "System Control Blocks" (SCB) - so heißt dieser Systemspeicher - befindet sich ein Flag, das die Umleitung der Bildschirmausgabe markiert.

Normalerweise enthält es den Wert 8000H (32768). Dann erfolgt die Bildschirmausgabe wirklich auf dem Gerät CRT. Setzt man aber den Hexwert 4000H (16384) ein, wird der Bildschirm abgeschaltet und stattdessen der Drucker angesprochen.

Das Einfachste wären zwei POKE-Befehle, etwa dieser Art:

```
POKE SCB+&H24,&00:POKE SCB+&H25,&40
```

beziehungsweise

```
POKE SCB+&H24,&00:POKE SCB+&H25,&80
```

Leider kann man aber nicht so einfach von Basic aus den benötigten Wert in den System Control Block schreiben. Denn CP/M verrät dem Anwenderprogramm nicht, an welcher Speicheradresse der SCB beginnt. Außerdem wäre ja gar nicht gewährleistet, daß der SCB auch wirklich in der Programmbank des RAM-Speichers liegt. Ebenso könnte er in der Systembank (Bank 0) zu finden sein.

Abhilfe schafft da eine CP/M-Funktion, die die Aufgabe übernimmt, Werte aus dem SCB zu lesen oder in ihn einzutragen. Entsprechend dem CP/M-Standard ist das eine BDOS-Funktion.

Sie besitzt die Funktionsnummer 31H (dezimal 49). Dieser Wert muß ins C-Register des Z80-Prozessors eingetragen werden:

```
LD C,49
```

Als nächstes definiert man einen vier Bytes langen Speicherbereich, der die Informationen für die BDOS-Funktion bereithält:


```
DB OFFSET  ; Nummer des zu ändernden Bytes
DB AKTION  ; Welche Aktion? 0FFH Byte schreiben
           ;                0FEH Wort schreiben
           ;                000H Wort lesen
DW WERT    ; Zu schreibendes Byte oder Wort
```

In unserem Fall sieht das so aus:

```
DATA: DB 024H     ; Byte 24H des SCB zu bearbeiten
      DB 0FEH     ; Zwei Bytes (Wort) schreiben
      DW 4000H    ; Wort 4000H in den SCB schreiben
```

Das DE-Register zeigt dann auf diesen Speicherbereich:

```
LD, DE,DATA
```

Und ab geht die Post, sprich der Aufruf des BDOS:

```
CALL BDOS
RET
```

Das komplette Maschinenprogramm sieht so aus:

```
START: LD   C,31H
       LD   DE,DATA
       CALL 5
       RET

DATA:  DB   024H
       DB   0FEH
       DW   4000H
```

Das Assemblerprogramm, das die Druckerausgabe wieder zurücksetzt, sieht genauso aus. Lediglich die letzte Zeile muß abgeändert werden:

```
DW 8000H
```

Damit wäre die Codierung des Programms klar. Stellt sich nur die Frage, wo man das Maschinenprogramm im Speicher unterbringen soll. Da bieten sich mehrere Möglichkeiten an.

Schauen wir uns kurz die Speicherverteilung von CP/M an:

0000H	Für das Betriebssystem reserviert
0100H	Beginn des Programmspeichers von CP/M
F606H	Start des BDOS
FC00H	Beginn des BIOS

Dabei handelt es sich beim BDOS um den Betriebssystemkern, der alle CP/M-Funktionen durchführt. Er ist vom Computertyp unabhängig. BDOS steht für "Basic Disk Operating System".

Die systemspezifischen Routinen sind hingegen im BIOS untergebracht. BIOS heißt "Basic Input/Output System".

Nach dem Laden von Mallard-Basic ändert sich diese Speicherverteilung wie folgt:

0000H	Für das Betriebssystem reserviert
0100H	Basic-Interpreter Mallard-Basic
7A94H	Anfang des Basic-Programms
F606H	Beginn des BDOS
FC00H	Beginn des BIOS

Von den Autoren des Mallard-Basic wurden der Befehl MEMORY und die Funktion HIMEM vorgesehen, die die Verbindung von Basic-Programmen und Maschinencode-Routinen regeln.

Die Funktion HIMEM gibt die höchste Adresse an, die vom Basic-Programm noch belegt werden darf. Wenn Sie direkt nach dem Laden von Mallard-Basic den Befehl PRINT HEX$ (HIMEM) eingeben, erhalten Sie den Wert &HF605. Dies ist genau die Startadresse des BDOS minus 1.

Mit dem Befehl MEMORY kann man die höchste für Basic-Programme verfügbare Speicheradresse verändern. So reserviert MEMORY HIMEM-100 genau 100 Bytes am oberen Speicherende.

Doch Programme, die auf diese Art an ein Basic-Programm angekoppelt werden sollen, sollten im Speicher frei verschiebbar sein. Sie dürfen damit keine absoluten internen Sprünge oder Ladebefehle enthalten. Auf solche Befehle kann man aber nur recht schwer verzichten.

Deshalb ist es einfacher, einen Trick anzuwenden. CP/M belegt den Speicher von der Adresse 0 bis 255 mit wichtigen Systemdaten. Dabei ist der Bereich von 128 bis 255 als Diskettenpuffer vorgesehen. Alle Lese- und Schreiboperation von/zur Diskette gehen über diesen Pufferspeicher. Allerdings kann der Speicher, wenn er gerade nicht benötigt wird, mit Maschinenroutinen belegt werden. Das nutzt das Programm aus. Und der Diskettenpuffer kann dennoch von CP/M weiterhin genutzt werden.

Diese Methode setzt nur voraus, daß unmittelbar vor dem Aufruf der Maschinenroutine diese mit einer POKE-Schleife in den Speicherbereich eingelesen wird. Somit wird sichergestellt, daß trotz Diskettenoperationen die Maschinenroutine aufgerufen werden kann.

Hier noch einmal die beiden Routinen als alleinstehendes Programm, das Sie mit MERGE in Ihre eigenen Programme aufnehmen können:

```
100 ' UMLEITUNG DER BILDSCHIRMAUSGABE
110 ' AUF DEN DRUCKER - CP/M-BDOS-AUFRUF &H31
120 ' ---------------------------------------
130 '
140 ' Aufruf: GOSUB 10000
150 '         Druckerausgabe ein
160 '         GOSUB 20000
170 '         Druckerausgabe aus
180 ' ---------------------------------------
190 '
10000 DATA &H0E,&H31,&H11,&H89,&H00,&HCD,&H05
10010 DATA &H00,&HC9,&H24,&HFE,&H00,&H40
10020 RESTORE 10000
10030 FOR i=&H80 TO &H8C:READ a:POKE i,a:NEXT i
10040 adr=&H80:CALL adr
10050 RETURN
10050 ' ---------------------------------------
20000 DATA &H0E,&H31,&H11,&H89,&H00,&HCD,&H05
20010 DATA &H00,&HC9,&H24,&HFE,&H00,&H80
20020 RESTORE 20000
20030 FOR i=&H80 TO &H8C:READ a:POKE i,a:NEXT i
20040 adr=&H80:CALL adr
20050 RETURN
```

Mit GOSUB 10000 wird der Computer veranlaßt, die Bildschirmausgabe auf den Drucker umzulenken. GOSUB 20000 macht das wieder rückgängig.

1.2 Rechnen in allen Zahlensystemen

Welchen Wert hat die Zahl 23? Sie werden sich wahrscheinlich fragen, was diese Frage soll. 23 ist 23 und sonst gar nichts. Aber wer hat denn gesagt, daß die Zahl im Zehnersystem angegeben ist? Vielleicht ahnen Sie schon, worauf ich hinaus will. Die Mathematiker rechnen mit einer Vielzahl verschiedener Zahlensysteme. Auch auf dem Computer sind einige dieser Zahlensysteme von Bedeutung, nämlich die Dualzahlen (Binärsystem), Oktalzahlen (8er-System) und hexadezimalen Zahlen (Sedezimalsystem).

Sehen wir uns doch einmal an, wieso eine Zahl wie 1269 gerade so heißt und nicht anders. Jede Stelle in der Zahl besitzt eine um den Faktor 10 höhere Wertigkeit als der rechts neben ihr stehende Nachfolger. Mathematisch kann man also den Wert 1269 auch so berechnen:

```
  1 * 1000 =  1 * 10^3 = 1000
+ 2 *  100 =  2 * 10^2 =  200
+ 6 *   10 =  6 * 10^1 =   60
+ 9 *    1 =  9 * 10^0 =    9
-------------------------------
                          1269
```

Auf diese Weise lassen sich aber auch Werte aus allen anderen Zahlensystemen in das Zehnersystem umrechnen. Nehmen wir zum Beispiel die Zahl 11101010, die im Binärsystem dargestellt ist. Statt Potenzen zur Basis 10 müssen wir natürlich die Zweierpotenzen verwenden:

```
  1 * 2^7 = 1 * 128 = 128
+ 1 * 2^6 = 1 *  64 =  64
+ 1 * 2^5 = 1 *  32 =  32
+ 0 * 2^4 = 0 *  16 =   0
+ 1 * 2^3 = 1 *   8 =   8
+ 0 * 2^2 = 0 *   4 =   0
+ 1 * 2^1 = 1 *   2 =   2
+ 0 * 2^0 = 0 *   1 =   0
-------------------------
                    234
```

Bei Zahlensystemen jenseits des Dezimalsystems tritt aber ein Problem auf: Es gibt nicht genügend verschiedene Ziffern für diese Systeme. Im

Oktalsystem lauten die Ziffern 0, 1, 2, 3, 4, 5, 6 und 7, im Zehnersystem bekanntlich 0, 1, 2, 3, 4, 5, 6, 7, 8 und 9. Was macht man aber im Elfer- oder Zwölfersystem? Da wenden die Mathematiker einen Kunstgriff an. Wenn die Ziffern ausgegangen sind, müssen die Buchstaben zur Kennzeichnung herhalten. Deshalb kennt das hexadezimale Zahlensystem die Ziffern 0 bis 9 und A, B, C, D, E und F. Die Buchstaben werden der Reihe nach vergeben:

.
.
.
9 hex = 9 dezimal
A hex = 10 dezimal
B hex = 11 dezimal
C hex = 12 dezimal
D hex = 13 dezimal
E hex = 14 dezimal
F hex = 15 dezimal

Deshalb sollte es auch keine Probleme geben, eine Zahl wie 3FAC hex in das Dezimalsystem umzurechnen. Als ersten Schritt schreibt man dazu statt der Buchstaben den Wert der Ziffern zwischen 10 und 15 auf:

```
Original:       3   F   A   C
Zwischenschritt: 3  15  10  12
```

Und damit kann man wieder das bekannte Umrechnungsschema anwenden:

```
+  3 * 16^3 =  3 * 4096 = 12288
+ 15 * 16^2 = 15 *  256 =  3840
+ 10 * 16^1 = 10 *   16 =   160
+ 12 * 16^0 = 12 *    1 =    12
-------------------------------
                        16300
```

Natürlich sind solche Umrechnungen nicht gerade schnell durchführbar. Sie kosten unnötig Zeit. Deshalb hat es nicht sehr lange gedauert, bis findige Basic-Programmierer auf die Iddee kamen, Unterprogramme zu schreiben, die die Umwandlung von einem Zahlensystem in ein anderes bewerkstelligen.

Locomotive-Software - das ist die Fima, die Mallard-Basic im Auftrag von Amstrad entwickelt hat - ging einen Schritt weiter. In das Mallard-Basic

wurden bereits einige Funktionen integriert, die die Umrechnungen vereinfachen. Sie unterstützen Berechnungen mit dem Oktal- und dem Hexadezimalsystem. Die beiden Funktionen, die eine Zahl aus ihrer dezimalen Darstellung in eine Zeichenkette mit Ziffern des jeweiligen Zahlensystems umwandeln, heißen HEX$ und OCT$:

```
PRINT HEX$(16300)
 3FAC
PRINT OCT$(16300)
 37654
```

Aber auch der umgekehrte Weg ist möglich. Hier wird den Zahlen ein kaufmännisches Und (&) sowie ein Kennbuchstabe vorangestellt. Bei oktalen Werten heißt es also "&O". Dabei ist der Kennbuchstabe ein "O", also das erste Zeichen des Wortes "Oktal" und keine Null! Für hexadezimale Zahlen wird als Kenner "&H" verwendet. Überall wo dezimale Konstanten stehen, kann man so auch die beiden anderen Zahlensysteme verwenden:

```
PRINT &H3FAC
 16300
PRINT PEEK(&o37654)
 224
```

Allerdings gibt es bei beiden Funktionen gewisse Stolperstellen. Was erwarten Sie wohl von folgender Befehlszeile?

```
PRINT &H9000
 -28672
```

Bestimmt alles, nur nicht die Ausgabe der Zahl -28672! Doch der Computer hat sich keineswegs verrechnet. Er betrachtet nur die Zahl etwas anders, als wir es gewohnt sind. Die höchste mit 16 binären Stellen darstellbare Zahl ist 2^16-1, also 65535. Der Wertebereich für vierstellige hexadezimale Zahlen liegt also zwischen 0 und 65535. Oft will man aber auch mit negativen Zahlen rechnen. Deshalb hat man einen Kunstgriff benutzt und einfach das am weitesten links stehende Bit von hexadezimalen und binären Zahlen zum Vorzeichenbit erklärt. Ist es gelöscht, handelt es sich um eine positive Zahl, sonst ist sie negativ. Da dieses Bit aber nun nicht mehr zur Speicherung des Zahlenwertes dienen kann, verkleinert sich der Wertebereich im positiven Feld. Es sind nur noch die Werte 0 bis 32767 (2^15-1) zu verarbeiten. Als Ausgleich dafür kommen die negativen Zahlen dazu. Sie können zwischen -1 und -32768 liegen. Fassen wir also noch einmal zusammen:

16-Bit-Darstellung ohne Vorzeichen: 0..65535
15-Bit-Darstellung mit Vorzeichen: -32768..32767

Eine Zahl in Vorzeichen-Darstellung können Sie in eine vorzeichenlose ganz einfach umrechnen:

```
PRINT &9000+65536
PRINT -28672+65536
```

Umgekehrt geht das genauso:

```
PRINT 40960-65536
PRINT 36864-65536
```

Noch einfacher geht der erste Schritt, die Umwandlung in die vorzeichenbehaftete Darstellung mit der Basic-Funktion UNT(x). Sie prüft, ob die Zahl größer als 32767 ist und wandelt sie bei Bedarf in ihr negatives Äquivalent um:

```
PRINT UNT(32767)
 32767
PRINT UNT(40960)
 -24576
```

Bedauerlicherweise wurde von den Programmierern des Mallard-Basic der zweite Fall, die Umwandlung einer negativen Zahl in eine vorzeichenlose Darstellung, nicht mit einer entsprechenden Funktion versorgt. Das ist aber an sich kein Problem. Mit DEF FN läßt sich Abhilfe schaffen:

```
DEF FNplus(x)=x-65536!*(x<0)
```

Hier in einem kleinen Beispielprogramm:

```
10 DEF FNplus(x)=x-65536!*(x<0)
20 PRINT FNplus(30000)
30 PRINT FNplus(-24576)
```

Versuchen Sie doch einmal selbst, die Arbeitsweise dieser kleinen Funktionsdefinition zu ergründen! Ein Tip: Der Dreh- und Angelpunkt ist der logische Vergleich x<0.

Einen handfesten Fehler hat sich Locomotive-Software aber bei der Berechnung von Oktalzahlen geleistet. Solange Ziffern verwendet werden, die

im Oktalsystem wirklich zugelassen sind, geht noch alles gut:

```
PRINT &o77674
 32700
PRINT &o77777
 32767
```

Aber dann kommt es ganz dick:

```
PRINT &o77778
 4095 8
PRINT &o38
 3 8
```

Statt eine Fehlermeldung abzugeben, liest der Basic-Interpreter einfach nur die Ziffern ein, die er verarbeiten kann, und behandelt den Rest als normale Dezimalzahl.

Ganz allgemein sind aber die Fähigkeiten des Interpreters Mallard-Basic beim Rechnen in anderen Zahlensystemen beschränkt. So ist der Wertebereich von hexadezimalen Zahlen nach oben bei 65535 zu Ende, Oktalzahlen machen sogar schon bei 32767 Schluß. Außerdem sind ja diese beiden Zahlensysteme nicht gerade alles, was es gibt. Warum beispielsweise Oktalzahlen implementiert wurden, nicht aber die viel wichtigeren binären Zahlen, wird wohl immer ein Geheimnis von Amstrad bleiben. Bei den Schneider-CPCs hingegen ist das vernünftiger gelöst. Dort gibt es BIN$ für Dualzahlen und HEX$ für die sedezimalen Werte, nicht aber OCT$.

Wer mit den anderen Zahlensystemen arbeiten möchte, kann das Programm ZAHLENWANDLER eintippen.

Dieses Programm rechnet in allen Systemen zwischen dem 2er- und dem 16er-System. Dabei werden die drei grundlegenden Aufgaben, die an ein solches Programm gestellt werden, erfüllt:

- Zahlenumwandlungen
- Grundrechenarten
- Zahlenliste auf dem Drucker

Entsprechend ist auch das Hauptmenü aufgebaut, das lediglich noch um den Menüpunkt "Programm beenden" ergänzt wurde. Die Steuerung des Menüs kennen Sie sicher schon von KALENDER.BAS ("Rund um den Kalender"):

Mit der Plus- und der Minus-Taste bewegen Sie einen invers dargestellten Balken über die Menüzeilen hinweg und bestätigen Ihre Auswahl mit der Leertaste.

Zahlenumwandlungen

Dieser Programmteil wandelt eine Zahl von einem gegebenen Zahlensystem in ein anderes um. Dazu zeigt der Zahlenwandler folgenden Dialog auf dem Bildschirm an:

```
Basis 1:  ->  2
Basis 2:  ->  16
Zahl 1:   ->  10101111010110111011101
```

Als Ergebnis berechnet der Zahlenwandler die hexadezimale Zahl 2BD6DD.

Aber auch Umwandlungen ins Zehnersystem sind hier ohne Schwierigkeiten möglich:

```
Basis 1:  ->  16
Basis 2:  ->  10
Zahl 1:   ->  3FCD35A
```

Das Ergebnis lautet hier 66900830. Versuchen Sie doch einmal, dasselbe in Basic zu erreichen! Sie sollten aber aufpassen, daß Sie die Buchstaben unbedingt in Großschrift eingeben, da sonst falsche oder gar keine Werte berechnet werden.

Grundrechenarten

Dieser Teil des Programms gestattet es Ihnen, Berechnungen unter Einschluß der vier Grundrechenarten in allen Zahlensystemen durchzuführen. Zum Beispiel soll zu der Oktalzahl 3432451 der oktale Wert 45555 addiert werden:

```
Zahlensystem:     ->  8
Zahl 1:           ->  3432451
Rechenoperator:   ->  +
Zahl 2:           ->  45555
```

Der Zahlenwandler berechnet hier in kurzer Zeit als Resultat die Oktalzahl 3500226.

Zahlenliste auf dem Drucker

Hiermit können Sie den Zahlenwandler auf dem Drucker eine Liste erstellen lassen, die dezimale Zahlen mit den Zahlen eines anderen Zahlensystems vergleicht.

Ein Beispiel: Sie wollen die binären Äquivalente der Zahlen 8 bis 14 schwarz auf weiß besitzen. Der Bildschirmdialog spielt sich etwa so ab:

```
Zahlensystem:    ->  2
Von Zahl:        ->  8
Bis Zahl:        ->  14
```

Der Drucker gibt dann folgendes Listing aus:

Zahlenliste:

Dezimal	2er-System
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110

Programm beenden

Die Funktion dieses Programmsegment bedarf wohl keiner Erläuterung. Nach einer Sicherheitsabfrage wird das Programm ZAHLENWANDLER gestoppt.

Im folgenden finden Sie das Listing des Zahlenwandlers:

```
100 '  *********************************
110 ' ***********************************
120 ' *                                 *
130 ' *          ZAHLENWANDLER          *
140 ' *                                 *
150 ' ***********************************
160 '  *********************************
170 '
180 WIDTH 80
190 cls$=CHR$(27)+"E"+CHR$(27)+"H"
200 german$=CHR$(27)+"2"+CHR$(2)
```



```
210 invon$=CHR$(27)+"p"
220 invoff$=CHR$(27)+"q"
230 curon$=CHR$(27)+"e"
240 curoff$=CHR$(27)+"f"
250 arrow$="="+CHR$(252)
260 csr$=CHR$(27)+"Y"
270 DEF FNlocate$(s,z)=csr$+CHR$(z+31)+CHR$(s+31)
280 ' TITELBILD UND ANLEITUNG ************************
290 PRINT cls$;german$;CHR$(13);
300 PRINT FNlocate$(25,1);
310 PRINT CHR$(150);STRING$(40,154);CHR$(156)
320 PRINT FNlocate$(25,2);
330 PRINT CHR$(149);SPACE$(40);CHR$(149)
340 PRINT FNlocate$(25,3);
350 PRINT CHR$(149);SPACE$(13);"ZAHLENWANDLER";
360 PRINT SPACE$(14);CHR$(149)
370 PRINT FNlocate$(25,4);
380 PRINT CHR$(149);SPACE$(40);CHR$(149)
390 PRINT FNlocate$(25,5);
400 PRINT CHR$(147);STRING$(40,154);CHR$(153)
410 ON ERROR GOTO 2120
420 num=4:row=10:col=10:choice=1
430 all$="0123456789ABCDEF"
440 ch$(1)="  ZAHLENUMWANDLUNGEN"+SPACE$(21)
450 ch$(2)="  GRUNDRECHENARTEN"+SPACE$(23)
460 ch$(3)="  ZAHLENLISTE AUF DRUCKER"+SPACE$(16)
470 ch$(4)="  PROGRAMM BEENDEN"+SPACE$(23)
480 PRINT FNlocate$(1,row);curoff$;CHR$(13);
490 FOR i=1 TO num:PRINT TAB(25);ch$(i):PRINT:NEXT i
500 PRINT FNlocate$(20,23);"Bitte wählen Sie mit ";
510 PRINT CHR$(208);"+";CHR$(212);", ";CHR$(208);
520 PRINT "-";CHR$(212);
530 PRINT " und der Leertaste!"
540 PRINT FNlocate$(25,choice*2+row-3);
550 PRINT STRING$(41,95)
560 PRINT TAB(25);invon$;ch$(choice);invoff$
570 a$=INKEY$
580 IF a$=CHR$(32) THEN 680 ' Leertaste gedrückt
590 IF a$="" OR (a$<>"+" AND a$<>"-") THEN 570
600 PRINT FNlocate$(25,choice*2+row-3);SPACE$(41)
610 PRINT TAB(25);ch$(choice)
620 IF a$="-" THEN choice=choice-1
630 IF a$="+" THEN choice=choice+1
640 IF choice<1 THEN choice=num
650 IF choice>num THEN choice=1
660 PRINT FNlocate$(25,choice*2+row-3);STRING$(41,95)
670 PRINT TAB(25);invon$;ch$(choice);invoff$:GOTO 570
680 PRINT curon$;
690 ON choice GOTO 710,1040,1460,1750
700 ' ZAHLENUMWANDLUNGEN ******************************
710 PRINT cls$;STRING$(79,95)
720 PRINT invon$;ch$(1);SPACE$(38);invoff$:PRINT
730 PRINT " Basis 1:";TAB(20);arrow$:PRINT
740 PRINT " Basis 2:";TAB(20);arrow$:PRINT:PRINT
750 PRINT " Zahl 1:";TAB(20);arrow$:PRINT
760 PRINT " Zahl 2:";TAB(20);arrow$
```

```
770 PRINT FNlocate$(24,6);:LINE INPUT vbasis$
780 vbasis=FIX(VAL(vbasis$))
790 PRINT FNlocate$(23,6);
800 PRINT STR$(vbasis)+SPACE$(18-LEN(STR$(vbasis)))
810 IF vbasis<2 THEN PRINT beep$;:GOTO 770
820 IF vbasis>16 THEN PRINT beep$;:GOTO 770
830 PRINT FNlocate$(24,8);:LINE INPUT nbasis$
840 nbasis=FIX(VAL(nbasis$))
850 PRINT FNlocate$(23,8);
860 PRINT STR$(nbasis)+SPACE$(18-LEN(STR$(nbasis)))
870 IF nbasis<2 OR nbasis>16 THEN PRINT beep$;:GOTO 830
880 PRINT FNlocate$(24,11);SPACE$(17);FNlocate$(24,11);
890 LINE INPUT vzahl$
900 FOR k=1 TO LEN(vzahl$)
910     md$=MID$(vzahl$,k,1)
920     IF INSTR(MID$(all$,1,vbasis),md$)=0 THEN 880
930 NEXT k
940 basis=vbasis:g.adisch$=vzahl$
950 GOSUB 1860:nzahl$=decimal$
960 PRINT FNlocate$(24,13);
970 basis=nbasis:decimal$=nzahl$
980 GOSUB 2010:PRINT g.adisch$
990 PRINT FNlocate$(1,24);beep$;" Bitte drücken Sie";
1000 PRINT " die Leertaste! ";invon$;" ";invoff$;
1010 WHILE INKEY$<>" ":WEND
1020 PRINT cls$;:GOTO 290
1030 ' GRUNDRECHENARTEN ********************************
1040 PRINT cls$;STRING$(79,95)
1050 PRINT invon$;ch$(1);SPACE$(38);invoff$:PRINT
1060 PRINT " Zahlensystem:";TAB(20);arrow$:PRINT:PRINT
1070 PRINT " Zahl 1:";TAB(20);arrow$:PRINT
1080 PRINT " Rechenoperator:";TAB(20);arrow$:PRINT
1090 PRINT " Zahl 2:";TAB(20);arrow$:PRINT:PRINT
1100 PRINT " Ergebnis:";TAB(20);arrow$
1110 PRINT FNlocate$(24,6);:LINE INPUT basis$
1120 basis=FIX(VAL(basis$))
1130 PRINT FNlocate$(23,6);
1140 PRINT STR$(basis)+SPACE$(18-LEN(STR$(basis)))
1150 IF basis<2 OR basis>16 THEN PRINT beep$;:GOTO 1110
1160 PRINT FNlocate$(24,9);:LINE INPUT num1$
1170 FOR k=1 TO LEN(num1$)
1180     md$=MID$(num1$,k,1)
1190     instring=INSTR(MID$(all$,1,basis),md$)
1200     IF instring=0 THEN PRINT beep$;:GOTO 1160
1210 NEXT k
1220 PRINT FNlocate$(24,11);
1230 a$=INKEY$:IF a$="" THEN 1230
1240 IF INSTR("*+-:/",a$)=0 THEN PRINT beep$;:GOTO 1230
1250 PRINT a$:operator$=a$
1260 PRINT FNlocate$(24,13);:LINE INPUT num2$
1270 FOR k=1 TO LEN(num2$)
1280     md$=MID$(num2$,k,1)
1290     instring=INSTR(MID$(all$,1,basis),md$)
1300     IF instring=0 THEN PRINT beep$;:GOTO 1260
1310 NEXT k
```



```
1320 g.adisch$=num1$:GOSUB 1860:num1=decimal
1330 g.adisch$=num2$:GOSUB 1860:num2=decimal
1340 IF operator$="+" THEN sum=num1+num2
1350 IF operator$="-" THEN sum=num1-num2
1360 IF operator$="*" THEN sum=num1*num2
1370 IF operator$="/" THEN sum=INT(num1/num2)
1380 IF operator$=":" THEN sum=INT(num1/num2)
1390 decimal$=STR$(sum):GOSUB 1970
1400 PRINT FNlocate$(24,16);g.adisch$
1410 PRINT FNlocate$(1,24);beep$;" Bitte drücken Sie";
1420 PRINT " die Leertaste! ";invon$;" ";invoff$;
1430 WHILE INKEY$<>" ":WEND
1440 PRINT cls$;:GOTO 290
1450 ' ZAHLENLISTE AUF DRUCKER ***********************
1460 PRINT cls$;STRING$(79,95)
1470 PRINT invon$;ch$(3);SPACE$(38);invoff$:PRINT
1480 PRINT " Zahlensystem:";TAB(20);arrow$:PRINT:PRINT
1490 PRINT " Von Zahl:";TAB(20);arrow$:PRINT
1500 PRINT " Bis Zahl:";TAB(20);arrow$
1510 PRINT FNlocate$(24,6);:LINE INPUT basis$
1520 basis=FIX(VAL(basis$))
1530 PRINT FNlocate$(23,6);
1540 PRINT STR$(basis)+SPACE$(18-LEN(STR$(basis)))
1550 IF basis<2 OR basis>16 THEN PRINT beep$;:GOTO 1510
1560 PRINT FNlocate$(24,9);:LINE INPUT num1$
1570 num1=ABS(FIX(VAL(num1$)))
1580 PRINT FNlocate$(24,11);:LINE INPUT num2$
1590 num2=ABS(FIX(VAL(num2$)))
1600 LPRINT:LPRINT "ZAHLENLISTE:":LPRINT
1610 LPRINT "Dezimal";TAB(39);STR$(basis);"er-System"
1620 LPRINT
1630 LPRINT STRING$(60,"-"):LPRINT
1640 FOR lp=num1 TO num2
1650     LPRINT lp;
1660     decimal$=STR$(lp)
1670     GOSUB 2010
1680     LPRINT TAB(40);g.adisch$
1690 NEXT lp
1700 PRINT FNlocate$(1,24);beep$;" Bitte drücken Sie";
1710 PRINT " die Leertaste! ";invon$;" ";invoff$;
1720 WHILE INKEY$<>" ":WEND
1730 PRINT cls$;:GOTO 290
1740 ' PROGRAMM BEENDEN ********************************
1750 PRINT cls$;STRING$(79,95)
1760 PRINT invon$;ch$(4);SPACE$(38);invoff$:PRINT
1770 PRINT " Sind Sie sicher? ";
1780 a$=UPPER$(INKEY$)
1790 IF a$<>"J" AND a$<>"N" THEN 1780
1800 PRINT cls$;:IF a$="N" THEN 290 ELSE END
1810 '
1820 ' UNTERPROGRAMM: G-adische Zahl ins Dezimalsystem
1830 '                umwandeln
1840 ' ---------------------------------------------
1850 '
```

```
1860 all$="0123456789ABCDEF"
1870 decimal=0
1880 length=LEN(g.adisch$)
1890 FOR i=1 TO length
1900      t$=MID$(g.adisch$,i,1)
1910      k=INSTR(all$,t$)-1
1920      decimal=decimal+k*basis^(length-i)
1930 NEXT i
1940 decimal$=STR$(decimal)
1950 RETURN
1960 '
1970 ' UNTERPROGRAMM: Dezimalsystem nach G-adisch
1980 '                umwandeln
1990 ' ------------------------------------------
2000 '
2010 g.adisch$=""
2020 decimal=VAL(decimal$)
2030 FOR i=0 TO 255
2040      r=decimal-basis*INT(decimal/basis)
2050      g.adisch$=MID$(all$,r+1,1)+g.adisch$
2060      decimal=INT(decimal/basis)
2070      IF decimal=0 THEN 2090 ' Ende
2080 NEXT i
2090 decimal$=STR$(decimal)
2100 RETURN
2110 ' FEHLERBEHANDLUNG ******************************
2120 IF (ERR<>6 AND ERR<>11 AND ERR<>15) THEN 2180
2130 PRINT FNlocate$(2,20);beep$;invon$;
2140 PRINT " Fehler bei den Zahlenwerten! ";
2150 PRINT invoff$;
2160 WHILE INKEY$="":WEND
2170 PRINT cls$;:RESUME 290
2180 PRINT cls$;beep$;"* Fehler in Zeile";ERL
2190 ERROR ERR
```

Für Programmierer:

Das Programm ist modular gegliedert. Die einzelnen Segmente des Basic-Programms werden durch REM-Zeilen erläutert. So dürfte es kaum Verständnisschwierigkeiten geben.

Die eigentlichen Umwandlungsroutinen für die Zahlensysteme sind, vom Hauptprogramm klar getrennt, in den Zeilen 1810 bis 1950 (Umwandlung von anderen Systemen ins Dezimalsystem) und 1960 bis 2100 (Umrechnungen aus dem Dezimalsystem) zu finden.

Die Algorithmen zur Umrechnung basieren auf der Ihnen bereits bekannten Methode, alle Zahlen als Potenzen zur jeweiligen Basis darzustellen. Sie sind eine Abwandlung des Horner-Schemas. Auch in anderen Programmen können Sie die beiden Unterprogramme benutzen, ja sogar im Direktmodus. Laden Sie dazu den Zahlenwandler und tippen Sie im Direktmodus folgende Befehlszeile ein:


```
g.adisch$="3FAC"
basis=16
GOSUB 1860
PRINT decimal$
```

Das System antwortet mit der Meldung "16300". Auch der umgekehrte Weg ist möglich:

```
decimal$="16300"
basis=16
GOSUB 2010
PRINT g.adisch$
```

Im ersten Fall sind also g.adisch$ und basis die Eingangsvariablen, der berechnete Wert wird in der Variablen decimal$ übergeben. Bei der umgekehrten Umwandlung dienen decimal$ und basis als Variablen beim Aufruf, während die berechnete Zahl in g.adisch$ zurückgegeben wird.

1.3 Drucken Sie sich Geldscheine!

Ganz so stimmt das natürlich nicht, was in der Überschrift gesagt wurde. Aber mit dem Programm DOUBLE.BAS können Sie die Unterhaltskosten für Ihren Drucker senken. Das läuft im Endeffekt auf eine erhebliche Kostenersparnis hinaus.

Die Aufgabe des Programms DOUBLE.BAS besteht darin, die Nutzungsdauer für die recht teuren Drucker-Farbbänder erheblich zu verlängern. DOUBLE.BAS ist eigentlich ein ganz normaler Dateilister - mit einer Besonderheit, die es in sich hat: Jede Zeile wird zweimal auf derselben Stelle auf dem Papier gedruckt. Dadurch erscheint die Schrift wesentlich dunkler und lesbarer. Auch altersschwache Farbbänder können noch für einige Zeit ihren Dienst tun.

Das Programm benutzt keine Druckersteuerzeichen und ist damit auch bei anderen eventuell an Ihren Joyce angeschlossenen Druckern höchstwahrscheinlich ohne Probleme einsetzbar. Natürlich halbiert diese Druckmethode die Geschwindigkeit, mit der die Zeichen zu Papier gebracht werden. Besonders bei der NLQ-Schönschrift sinkt die Geschwindigkeit stark ab. Das ist eben der Preis für das bessere Schriftbild und das gesparte Geld ...

```
100 '  ********************************
110 ' **********************************
120 ' *                                *
130 ' *      FARBBÄNDER SPAREN         *
140 ' *                                *
150 ' **********************************
160 '  ********************************
170 '
180 ' BILDSCHIRM-STEUERZEICHEN ************************
190 WIDTH 80
200 cls$=CHR$(27)+"E"+CHR$(27)+"H"
210 german$=CHR$(27)+"2"+CHR$(2)
220 inverseon$=CHR$(27)+"p"
230 inverseoff$=CHR$(27)+"q"
240 ulineon$=CHR$(27)+"r"
250 ulineoff$=CHR$(27)+"u"
260 beep$=CHR$(7)
270 csr$=CHR$(27)+"Y"
280 DEF FNlocate$(s,z)=csr$+CHR$(z+31)+CHR$(s+31)
290 ' TITELBILD UND ANLEITUNG *************************
300 PRINT cls$;german$;inverseoff$;
310 PRINT FNlocate$(25,1);
320 PRINT CHR$(150);STRING$(40,154);CHR$(156)
330 PRINT FNlocate$(25,2);
340 PRINT CHR$(149);SPACE$(40);CHR$(149)
350 PRINT FNlocate$(25,3);CHR$(149);
360 PRINT SPACE$(17);"DOUBLE";SPACE$(17);CHR$(149)
370 PRINT FNlocate$(25,4);
380 PRINT CHR$(149);SPACE$(40);CHR$(149)
390 PRINT FNlocate$(25,5);
400 PRINT CHR$(147);STRING$(40,154);CHR$(153)
410 PRINT:PRINT:PRINT
420 PRINT FNlocate$(27,10);
430 PRINT "Dieses Programm hilft Ihnen, Drucker-"
440 PRINT FNlocate$(27,12);
450 PRINT "farbbänder zu sparen: Es druckt Texte"
460 PRINT FNlocate$(27,14);
470 PRINT "aus Dateien zweifach übereinander aus."
480 PRINT
490 PRINT:PRINT TAB(27);inverseon$;
500 INPUT " Dateiname?  ",file$:PRINT inverseoff$;
510 PRINT FNlocate$(28,24);ulineon$;
520 PRINT "BITTE WARTEN!";ulineoff$
530 ' AUSDRUCKEN DER DATEI ****************************
540 OPEN "I",1,file$
550 WHILE NOT EOF(1)
560       LINE INPUT #1,a$
570       LPRINT a$;CHR$(13);a$;CHR$(13);CHR$(10);
580 WEND
590 CLOSE 1
600 PRINT cls$;
610 END ' ***********************************************
```


Die Bedienung des Programms ist wirklich äußerst einfach. Nach dem Start von DOUBLE.BAS zeigt das Programm einen kurzen Informationstext an. Es will dann von Ihnen den Namen der auszudruckenden ASCII-Datei wissen. Danach läuft alles automatisch; Sie müssen nur von Zeit zu Zeit ein neues Blatt Papier in den Drucker einlegen. Sobald der Ausdruck beendet ist, erscheint wieder die Bereitschaftsmeldung des Basic-Interpreters.

Für Programmierer gibt es bei DOUBLE.BAS eigentlich kaum etwas zu sagen. Dieses Programm gehört in die Gruppe der Software, die die Amerikaner "quick and dirty" nennen. Darunter versteht man ein schnell für einen Anwendungszweck geschriebenes Programm, bei dem vorher keine ausführlichen Programmplanungen stattgefunden haben und das auch kein Nachdenken über die Funktionsweise erfordert.

2 Das Betriebssystem CP/M

Wer sich mit Mallard-Basic beschäftigen will, muß auch zumindest grundlegende Kenntnisse über CP/M besitzen, das dahinterstehende Betriebssystem. Das Benutzerhandbuch von Schneider ist nicht gerade ein Ausbund an Übersichtlichkeit. Deshalb.holen wir hier einiges nach.

2.1 Was ist ein Betriebssystem?

Ein Computer, der nur aus einem Mikroprozessor und verschiedenen Speicherbausteinen besteht, ist "nackt". Man kann nichts mit ihm anfangen. Wie sollte man mit ihm auch in Kommunikation treten?

Um einen Computer sinnvoll einsetzbar zu machen, muß er ein "Betriebsprogramm" besitzen, das dem Benutzer die Möglichkeit zum Dialog mit dem System über die Tastatur gibt und ihm erlaubt, Programme zu starten. Und ein "Betriebssystem" ist nichts anderes als ein leistungsfähiges Betriebsprogramm.

Bei Homecomputern der Klasse Commodore 64 oder Schneider-CPC merkt der Anwender gar nicht, daß ein Betriebssystem vorhanden ist. Es "versteckt" sich hinter dem Basic-Interpreter. Oft sind Basic und Betriebssystem gar nicht klar voneinander getrennt. Deshalb spricht man in diesem Fall von basicorientierten Betriebssystemen. Das Betriebssystem wird erst dann sichtbar, wenn sich ein Computerbesitzer als Programmierer auf die Maschinensprache-Ebene begibt, um dort Systemfunktionen aufzurufen.

Bei Personalcomputern - zu diesen muß man den Joyce-PCW wohl zählen - ist das ganz anders. Der Basic-Interpreter ist nicht schon beim Einschalten des Computers verfügbar, sondern muß von der Diskette geladen werden. Im ROM, dem dauerhaften Speicher des Computers, ist nur eine kurze Programmroutine vorhanden, die das Betriebssystem von der Diskette in den RAM-Speicher lädt. Der Benutzer kommt auch hier nicht direkt mit dem Betriebssystem in Kontakt, sondern mit dessen Kommandoprozessor. In diesem kann er Befehle zur Verwaltung seiner Disketten eingeben und Programme starten.

Dieser Aufbau von Betriebssystem und Basic-Interpreter hat gegenüber Homecomputern einen entscheidenden Vorteil. Solange der Benutzer nicht in Basic programmieren möchte, nimmt der Basic-Interpreter auch keinen

Speicherplatz weg. Und viele Anwender wollen gar nicht programmieren, sondern ihren PC ausschließlich für Anwendungen wie Textverarbeitung, Tabellenkalkulation und Datenverwaltungsaufgaben einsetzen.

2.2 Warum ein Standard-Betriebssystem?

Stellen Sie sich diese Situation vor: Jeder Computer ist anders aufgebaut. Das Betriebssystem ist speziell an die Hardware angepaßt. Dadurch funktioniert es zwar optimal. Es ist aber schwierig, wenn nicht sogar unmöglich, Programme anderer Computer einzusetzen. Die Softwarehäuser müssen in mühsamer Kleinarbeit die Programme "von Hand" übersetzen oder gänzlich neu codieren. Diesen erhöhten Arbeitsaufwand lassen sich die Software-Hersteller natürlich entsprechend bezahlen. Die Folge sind höhere Preise für Computerprogramme. Und wenn Sie sich einen Computer gekauft haben, der insgesamt keine überwältigenden Absatzzahlen erreicht, sitzen Sie vielleicht ganz auf dem Trockenen. Kaum ein Softwarehaus ist bereit, Programme für "Flop-Computer" anzubieten.

Das Schönste wäre es natürlich, einen Computer zu produzieren, der möglichst viele Programme anderer Computer verarbeiten kann. Das scheitert aber sowohl am Preis als auch an den technischen Möglichkeiten.

Ein anderer Weg ist es, ein Betriebssystem zu entwickeln, das Programmen genau definierte Möglichkeiten bietet und relativ einfach an verschiedene Computertypen anzupassen ist. Dann können die Programmanbieter Software genau für dieses Betriebssystem entwickeln und für eine große Zahl verschiedener Computertypen anbieten.

Exakt das machen Betriebssysteme wie CP/M für Computer wie Apple, Commodore 128, Schneider-CPC, Kaypro und natürlich Schneider-Joyce oder auch MS-DOS für IBM-kompatible Computer. Auch für den Benutzer hat die Standardisierung einen entscheidenden Vorteil. Arbeitet er beispielsweise beruflich mit einer CP/M-Maschine von Kaypro, braucht er sich nach Feierabend nicht auf ein neues Betriebssystem einzustellen, sondern kann genau dieselben Befehle und Programme auch auf seinem Joyce einsetzen.

2.3 Historie

Die Geschichte der Personalcomputer ist eng mit der Geschichte von CP/M verknüpft. Mit CP/M wiederum hängen zwei Namen zusammen: Digital Research und Gary Kildall.

Versetzen Sie sich zurück in das Jahr 1972, sozusagen das Frühmittelalter der Computergeschichte. Da hatte ein Mann namens Gary Kildall gerade seinen Doktortitel auf dem Gebiet der "Computer Science", der Informatik also, abgelegt. Er unterrichtete an einer Militärschule in der kalifornischen Stadt Monterey. Anscheinend füllte diese Position ihn nicht voll aus, denn er trug sich mit dem Gedanken, eine eigene Firma zu gründen und Software zu entwickeln. Das Produkt seiner Überlegungen war eine kleine Firma mit dem Namen "Microcomputer Application Associates", abgekürzt M.A.A.

Kildall sah 1973 erstmals in einem Labor der Elektronikfirma Intel das Labormuster eines Mikroprozessors mit der Bezeichnung "8080". Der heutzutage verkaufte Z80-Prozessor, der sich auch in Ihrem Joyce befindet, ist ein direkter Nachkomme dieses legendären Chips. Kildall war von den Fähigkeiten des Intel-8080 sehr angetan und bot der Firma an, einen Compiler für die Programmiersprache PL/M zu schreiben, die direkt auf diesen Mikroprozessor zugeschnitten sein sollte.

Es gab nur ein einziges Problem: Der 8080-Prozessor wurde noch nicht produziert und war auch noch nicht vollkommen ausgereift. Deshalb überlegte sich Kildall, ob er PL/M nicht auf einem PDP-10-Computer von Digital Equipment in der Programmiersprache FORTRAN entwickeln und austesten sollte. Nachdem Gary Kildall den PL/M-Compiler tatsächlich auf einer PDP-10 fertiggestellt hatte, bot sich ihm die Gelegenheit, einen kleinen Computer auf der Basis des 8080-Prozessors zu kaufen. Nach langwierigen Überredungsversuchen schaffte er es sogar, von der damals noch winzigen Firma Shugart Associates ein gebrauchtes Diskettenlaufwerk geschenkt zu bekommen, allerdings ohne Kontrollprogramm, Kabel und Stromversorgung. Damit konnte er nichts anfangen und arbeitete mit der PDP-10 weiter.

Als Kildall seinen PL/M-Compiler austesten wollte, stand er vor dem Problem, daß er auf der PDP-10 den 8080-Prozessor von Intel simulieren mußte. Für diesen fehlte natürlich noch ein Betriebssystem, das er auch noch schreiben mußte. So entstand die erste Version von CP/M, dem "Control Program for Microcomputers".

Im Herbst 1973 traf Kildall dann John Torode, der eine Ahnung von Computer-Hardware hatte und es fertigbrachte, die alte Shugart-Floppy an den 8080-Einplatinencomputer anzuschließen. Das Überraschendste war, daß das Betriebssystem CP/M - bisher nur auf einem anderen Computer simuliert - nahezu auf Anhieb auf dem 8080-Computersystem funktionierte.

Kildall bot sowohl den PL/M-Compiler als auch das CP/M-Betriebssystem dem Chiphersteller Intel zur Vermarktung an. Über PL/M konnten sich beide

schnell einigen, es wird seitdem von Intel vertrieben. Aber an CP/M hatte die Firma kein Interesse. Denn man baute dort noch Großcomputer, die mehrere Benutzer gleichzeitig bedienen konnten. Wozu sollte man also ein Betriebssystem verwenden, das für jeden Benutzer einen eigenen Computer vorsieht?

Wie jeder gute Unternehmer gab Gary Kildall trotz dieses Rückschlags nicht auf, sondern entschied sich dafür, CP/M unter eigenem Namen zu vermarkten. Einige Zeit später hatte er schon mehrere Lizenznehmer. Das waren Firmen, die kein Interesse daran hatten, für ihre Computer systemspezifische Betriebssysteme zu entwickeln. Ab 1976 verkaufte Kildall das CP/M-System unter einem neuen Firmennamen: Digital Research Incorporated war gegründet.

Per Postwurfsendungen machte er CP/M in seiner Heimat in Kalifornien bekannt. Bei der gerade erwachenden Begeisterung für Computer fiel diese Werbung auf fruchtbaren Boden. CP/M 1.3 für 70 Dollar - das war ein Geschäft für Digital Research und ein Segen für die Computerfreaks, die nun hemmungslos Programme zwischen ihren Computern austauschen konnten.

Gary Kildall entwickelte CP/M ständig weiter. So brachte er 1979 die Version 2.0 heraus, die schon bald durch CP/M 2.2 ersetzt wurde. Diese Variante von CP/M ist auch heute noch die am weitesten verbreitete, denn sie bietet relativ viel Computerleistung, ohne allzu großen Speicherplatz zu belegen.

Weitere Entwicklungen von Digital Research tendierten hin zu Betriebssystemen, die mehrere Programme und mehrere Benutzer auf einmal verwalten konnten. MP/M und MP/M II waren solche Betriebssysteme. CP/M Plus, das Sie heute auf Ihrem Joyce-PCW benutzen, wurde relativ spät im Jahr 1982 veröffentlicht - eigentlich zu spät, wie man es heute beurteilen kann.

Denn die Firma DRI hatte die Entwicklung des Marktes hin zu den 16-Bit-Computern der Klasse des IBM-PC regelrecht verschlafen. Ihr Angebot an Betriebssystemen für diese Computer beschränkte sich auf CP/M-86. Dieses war nur sehr halbherzig entwickelt worden und bot gegenüber den alten 8-Bit-Systemen keinerlei Vorteile.

Als sich die Firma IBM nun zu entscheiden hatte, welches Betriebssystem der IBM-PC bekommen sollte, führte man durchaus Verhandlungen mit Digital Research. Aber gleichzeitig verhandelte man auch mit Microsoft. Diese Firma hatte kurz vorher von Seattle Computers Inc. ein Betriebssystem

für 16-Bit-Computer eingekauft, das den Namen QDOS trug. Microsoft machte daraus "MS-DOS" und wurde mit IBM schnell handelseinig. Da jeder IBM-Personalcomputer standardmäßig mit MS-DOS ausgeliefert wird, CP/M-86 aber extra hinzugekauft werden muß, dürfte klar sein, daß kaum jemand noch Interesse am Digital Research-Programm hatte, das teurer war und weniger leistete. Neue Entwicklungen von DRI wie DOS-Plus und GEM, die beide beim Schneider PC-1512 mitgeliefert werden, lassen die Zukunft von Digital Research aber wieder positiver erscheinen.

Und uns kann es relativ egal sein, welche Marktposition die Firma, die CP/M programmiert hat, heutzutage besitzt. Wir wollen nur CP/M anwenden.

2.4 So starten Sie CP/M Plus

Joyce arbeitet ausschließlich CP/M-orientiert. Deshalb müssen Sie beim Einschalten des Computers eine Systemdiskette in das obere Diskettenlaufwerk einlegen und warten, bis sich der Computer meldet.
Wenn kein Fehler auftritt, erscheint nach kurzer Zeit in der linken oberen Bildschirmecke der folgende Text:

CP/M Plus Amstrad Consumer Electronics plc
v 1.4, 61K TPA, 1 Laufwerk, 112K Laufwerk M:

Es kann durchaus sein, daß diese Anzeige etwas von der Ihrigen abweicht. So zeigt der Joyce PCW-8512 diese Meldung:

CP/M Plus Amstrad Consumer Electronics plc
v 1.4, 61K TPA, 2 Laufwerke, 368K Laufwerk M:

Findet aber der Computer auf der Diskette die Datei J14GCPM3.EMS nicht, gibt er Piepstöne von sich und stellt sich tot. In diesem Fall sollten Sie überprüfen, ob Sie wirklich die Systemdiskette eingelegt haben. Einen neuen Ladeversuch können Sie starten, indem Sie die Leertaste betätigen.

2.5 Gehversuche in CP/M Plus

Sie sehen nach dem Laden von CP/M Plus die Bereitschaftsmeldung des Systems:

A>

CP/M wartet also darauf, daß Sie irgendetwas eingeben. Tippen Sie also einen beliebigen Text ein, zum Beispiel:

A>HALLO JOYCE!

Betätigen Sie dann die Taste, die mit RETURN beschriftet ist. Der Computer antwortet kurz darauf: .

HALLO?

Offensichtlich kann er mit HALLO JOYCE nichts anfangen. Aber wir können daran gehen, die Zeile zu korrigieren. Betätigen Sie dazu die Tastenkombination ALT-W. Derselbe Text erscheint erneut und kann jetzt korrigiert werden. Dazu drücken Sie auf die Cursor-links-Taste und stellen den Cursor auf das "A" von "HALLO". Wenn Sie jetzt ein "E" eintippen, sieht das so auf dem Bildschirm aus:

A>HEALLO JOYCE!

Der Computer hat also den neuen Buchstaben eingefügt und den Rest der Zeile um eine Position nach rechts gerückt. Der Cursor steht jetzt über dem "A". Wenn Sie jetzt die Taste DEL-> drücken, verschwindet dieser Buchstabe, denn Sie haben ihn mit dieser Taste gelöscht. Während DEL-> nach rechts löscht, entfernt <-DEL die Zeichen in der umgekehrten Richtung:

A>HLLO JOYCE!

Sie kennen jetzt folgende Tasten, die bei der Arbeit mit CP/M nützlich sind:

Taste	Gleichwertig mit	Funktion
RETURN	ALT-M	Abschluß der Eingabe
<--	ALT-A	Cursor nach links
-->	ALT-F	Cursor nach rechts
<-DEL	ALT-H	Löschen nach links
DEL->	ALT-G	Löschen nach rechts
ALT-W		Wiederholung der Eingabe

Wenn Sie mit Ihrer Eingabe einmal gar nicht zufrieden sein sollten, können Sie die Befehlszeile mit ALT-X wieder löschen. Der Computer entfernt sie vom Bildschirm und aus dem internen Speicher. Umfaßt die Eingabezeile aber

mehr als eine Bildschirmzeile, entfernt CP/M nur die letzte Zeile vom Bildschirm, auch wenn intern die gesamte Eingabe gelöscht wurde.

Eine alternative Lösung ist es, ALT-U zu drücken. Dann bleibt die Anzeige zwar auf dem Bildschirm stehen, wird aber intern gelöscht. Das ist ganz praktisch, wenn Sie eine komplizierte Zeile eintippen wollen. Sie können nämlich in Ihrer vorhergehenden Zeile "spicken".

Zwei Tastenkombinationen besitzen noch eine besondere Bedeutung: ALT-P schaltet den Drucker parallel zur Bildschirmanzeige. Alles was auf dem Bildschirm erscheint, wird auch vom Drucker mitprotokolliert. Ein erneuter Tastendruck auf ALT-P schaltet diese Protokollfunktion wieder ab. Und die Taste TAB, die identisch ist mit ALT-I, fügt Leerzeichen bis zum nächsten Tabulatorstop ein. Sicher kennen Sie Tabulatoren schon von Locoscript oder auch von elektrischen Schreibmaschinen.

Nun wollen wir aber einmal eine vernünftige Eingabe machen. CP/M kennt verschiedene Befehle, die sofort ausgeführt werden. Sie heißen "resident", weil sie ständig verfügbar sind. Dazu gehört das Kommando zum Anzeigen des Inhaltsverzeichnisses einer Diskette:

```
A>DIR
A: J14GCPM3 .EMS  : BASIC     COM  : DIR      COM : ED       COM
A: ERASE    COM  : LANGUAGE  COM  : PALETTE  COM : PAPER    COM
A: PIP      COM  : RENAME    COM  : SET      COM : SET24X80 COM
A: SETDEF   COM  : SETKEYS   COM  : SETLST   COM : SETSIO   COM
A: SHOW     COM  : SUBMIT    COM  : TYPE     COM : KEYS     WP
A: PROFILE  GER  : RPED      BAS  : RPED     SUB : DISCKIT  COM
```

Sie sehen hier eine Vielzahl von Dateien und Programmen. Programme erkennen Sie an der Namensendung ("Extension") .COM. Diese Programme können Sie durch Eingabe ihres Namens starten. Bei BASIC.COM sieht das so aus:

```
A>BASIC
```

Zu den Funktionen der einzelnen Programme erfahren Sie später noch mehr. Denn zu DIR gibt es noch einiges zu sagen. Sie sehen hier 24 Dateinamen, es können aber auf jeder Diskette bis zu vierundsechzig werden. So eine Liste von Namen wird schnell unübersichtlich. Deshalb wurde in CP/M eine Möglichkeit integriert, mal schnell nachzusehen, ob eine bestimmte Datei vorhanden ist:


```
A>DIR BASIC.COM
A: BASIC    COM
```

oder wenn die Datei fehlt:

```
A>DIR X.Y
No File
```

Hier haben wir den DIR-Befehl um einen Dateinamen ergänzt. Was machen wir aber, wenn wir wissen wollen, welche Programme - nicht Dateien - eine Diskette enthält? Dazu hat man das Prinzip der "Wildcards" eingeführt. Immer wenn Sie statt eines Buchstabens ein Fragezeichen angeben, läßt der Computer bei den Vergleichen mit den tatsächlichen Dateinamen auf der Diskette dieses Zeichen aus. So könnten Sie sich alle .COM-Programme ansehen:

```
A>DIR ????????.COM
A: BASIC    COM : DIR      COM : ED       COM : ERASE  COM
A: LANGUAGE COM : PALETTE  COM : PAPER    COM : PIP    COM
A. RENAME   COM : SET      COM : SET24X80 COM : SETDEF COM
A. SETKEYS  COM : SETLST   COM : SETSIO   COM : SHOW   COM
A. SUBMIT   COM : TYPE     COM : DISCKIT  COM
```

Mit der Zeit werden aber die vielen Fragezeichen lästig. Eine Abkürzung für sie ist der Stern. Er steht als mehrdeutiger Ersatz für alle Zeichen bis zum Ende eines Namensteils. Die folgenden Formulierungen sind damit gleichwertig:

```
NAMEN.*     NAMEN.???
N*.COM      N???????.COM
*.*         ????????.???
BAS*.*      BAS?????.???
```

Erkennen Sie das Prinzip? Diese mehrdeutigen Dateinamen lassen sich bei vielen Befehlen in CP/M anwenden.

Wenn Sie zwei Diskettenlaufwerke besitzen, werden Sie nach einer Möglichkeit suchen, sich auch ein Verzeichnis aller Dateien im zweiten Laufwerk anzusehen. Dazu müssen Sie wissen, daß die Diskettenlaufwerke unter CP/M durchnumeriert werden - ja, nicht ganz. Sie erhalten Buchstaben von A bis P, gefolgt von einem Doppelpunkt, also A:, B: usw. bis P:. Kaum ein Computer nutzt allerdings alle Bezeichnungen für Laufwerke aus. So kennt Joyce genau drei Kenner:

```
A:
B:
M:
```

Hier bezeichnet A: das in den Joyce eingebaute obere Laufwerk, B: die darunterliegende Floppy. Bei Verwendung eines Joyce 8256 simuliert der Computer das zweite Laufwerk. Dazu zeigt er in der untersten Bildschirmzeile abwechselnd zwei Meldungen an:

```
Laufwerk ist A:
Laufwerk ist B:
```

Sobald Sie einen Zugriff auf das B:-Laufwerk durchführen, erscheint eine Meldung in der untersten Textzeile, in der Sie aufgefordert werden, eine andere Diskette in das Laufwerk A: einzulegen.

Dies ist eine recht intelligente Methode, dem Computer und den Programmen das Vorhandensein zweier Laufwerke vorzuspiegeln. Manche Programme arbeiten aber ständig abwechselnd mit beiden Laufwerken, so daß Sie andauernd am Diskettenwechseln sind. Der Begriff "Disc-Jockey" ist sicher nicht ganz unpassend ...

Das Laufwerk M: ist eine ganz besondere Einrichtung. Es handelt sich hierbei um eine RAM-Diskette. Darunter versteht man einen RAM-Speicherbereich im Rechner, der sich genauso verhält, als wäre er ein Diskettenlaufwerk. Diese RAM-Disk ist beim PCW-8256 genau 112 KByte groß, bei seinem großen Bruder immerhin 368 KByte.

Der Vorteil einer solchen RAM-Diskette ist, daß der Zugriff auf sie viel schneller vonstatten geht als auf ein normales mechanisches Laufwerk. Und natürlich werden auch keine mechanischen Bauteile abgenutzt. Auf der negativen Seite muß man aber beachten, daß die Daten in der RAM-Disk nicht sehr sicher sind. Bevor Sie Ihren Computer abschalten, müssen Sie die Dateien von M: auf eine reale Diskette kopieren. Wenn aber Ihr Computer abstürzt oder ein Stromausfall auftritt, sind die Daten unwiederbringlich verloren. Bei Abwägung der Risiken und der Vorteile siegt aber meistens doch die Bequemlichkeit, was bedeutet, daß Sie einer gewissen Gefahr ausgesetzt sind, die Daten zu verlieren.

Bei der Ausgabe des Directory (Inhaltsverzeichnis) eines der beiden anderen Laufwerke gibt es zwei Möglichkeiten. Die eine sieht so aus:

```
A>DIR B:
A>DIR M:
```


Hier geben Sie einfach die Laufwerksbezeichnung nach dem Befehlswort DIR an.

Bei der anderen Möglichkeit spielt die Systemmeldung "A>" eine Rolle. Denn dieser Text hat natürlich eine Bedeutung. Er zeigt an, daß das Laufwerk A: "angemeldet" ist. Das heißt, daß alle Diskettenzugriffe, bei denen nicht ausdrücklich eine Laufwerksbezeichnung angegeben wird, an diese Floppy gerichtet werden. So erhalten Sie bei

```
A>DIR
```

eine Inhaltsauflistung von A:, nicht etwa von B: oder M:. Denn A: ist das angemeldete oder "aktuelle" Laufwerk.

Mit den Befehlen "A:", "B:" und "M:" können Sie ein neues Laufwerk anmelden. Der System-Prompt (so nennt man die Anzeige "A>") ändert sich entsprechend:

```
A>B:
B>M:
M>A:
A>
```

Schalten Sie einmal mit "M:" auf das RAM-Laufwerk um und lassen Sie sich dann das Inhaltsverzeichnis anzeigen:

```
A>M:
M>DIR
No File
```

Hier sind ja noch keine Dateien gespeichert, weil sie stets beim Ausschalten vom Computer gelöscht werden.

2.6 Die anderen residenten Kommandos

CP/M Plus kennt neben DIR und den Befehlen zur Anwahl der Laufwerke noch einige weitere Befehle. Sie sind in der folgenden Liste aufgeführt:

```
ERA  ERASE
REN  RENAME
```

```
TYPE TYP
DIRS DIRSYS
USE  USER
```

Sie können hier sowohl die verständlichere Langform als auch die zeitsparende Abkürzung verwenden.

2.6.1 ERASE

ERASE löscht Dateien von der Diskette. Genauso wie bei DIR können Sie eindeutige oder mehrdeutige Dateinamen angeben. Auch ist eine Laufwerksbezeichnung gestattet.

Wollen Sie die Datei BRIEF.TXT vom angemeldeten Laufwerk löschen, teilen Sie das dem Computer so mit:

```
A>ERA BRIEF.TXT
A>ERASE BRIEF.TXT
```

Bei mehrdeutigen Dateinamen fragt CP/M Plus sicherheitshalber noch einmal nach, ob Sie wirklich sicher sind:

```
A>ERA *.TXT
ERASE *.TXT (Y/N)?
```

Zusammen mit der Benennung eines Floppy-Laufwerks sieht ERASE so aus:

```
A>ERASE M:PIP.COM
A>ERASE B:BRIEF.TXT
A>ERA B:*.*
```

2.6.2 RENAME

Das REN-Kommando benennt Diskettendateien um. Dabei steht zuerst der neue Dateiname, gefolgt von einem Gleichheitszeichen, dann der alte Name der Datei:

```
A>REN NEU.TXT=ALT.TXT
```


Sofern der alte Dateiname gefunden wird, benennt CP/M die Datei nach Ihren Vorstellungen um. Existiert aber bereits eine Datei mit demselben Namen wie dem, den die Datei erhalten soll, fragt CP/M nach:

```
A>RENAME SUBMIT.COM=ED.COM
ERROR: NOT renamed, SUBMIT.COM file already exists, delete (Y/N)?
```

Hier können Sie sich entscheiden, ob die Zieldatei gelöscht werden soll oder CP/M die Neubenennung abbrechen soll.

Es ist natürlich auch zulässig, vom angemeldeten Laufwerk aus Dateien auf anderen Laufwerken umzubenennen:

```
A>RENAME B:NEU=ALT
```

Die Laufwerksbezeichnung kann wahlweise beim ersten oder zweiten Dateinamen stehen. CP/M macht da keinen Unterschied. Allerdings dürfen Sie nicht zwei unterschiedliche Laufwerke aufführen. Denn ein solches "Umbenennen" wäre effektiv ein Kopieren der Datei. Das kann RENAME aber nicht. Dazu benötigen Sie PIP.COM, das ebenfalls auf Ihrer Systemdiskette zu finden ist.

2.6.3 TYPE

Bevor Sie Diskettendateien manipulieren wollen, ist es oft ratsam nachzuschauen, ob Sie wirklich mit der richtigen Datei arbeiten. Das Löschen einer eigentlich noch benötigten Datei ist sehr ärgerlich. Ohne Tricks lassen sich solche Files nicht "wiederbeleben".

Deswegen besitzt CP/M den Befehl TYPE, der Textdateien auf dem Bildschirm auflistet:

```
A>TYPE PROFILE.GER
setdef m:,* [order = (sub,com) temporary = m:]
pip
<m:=basic.com[o]
<m:=dir.com[o]
<m:=erase.com[o]
<m:=paper.com[o]
<m:=pip.com[o]
<m:=rename.com[o]
```

```
<m:=setkeys.com[o]
<m.=show.com[o]
<m.=submit.com[o]
<m.=type.com[o]
<
```

Aber versuchen Sie nicht, Programmfiles oder codierte Dateien anzuschauen. Denn Sie erhalten allenfalls unleserliche Grafiksymbole. Im schlimmsten Fall gerät die ganze Bildschirmgestaltung durcheinander.

TYPE, abgekürzt TYP, darf nur mit eindeutigen Dateinamen aufgerufen werden. Um sich also alle .TXT-Dateien einer Diskette anzusehen, müssen Sie tatsächlich entsprechend viele TYPE-Kommandos eingeben:

```
A>TYPE BRIEF.TXT
A>TYP BUCH.TXT
A>TYPE RECHNUNG.TXT
  .
  .
  .
```

2.6.4 DIRSYS

Manche Dateien sollen, obwohl sie auf einer Diskette vorhanden sind, nicht unbedingt im Inhaltsverzeichnis auftauchen. Denn sie könnten ja unerfahrene Benutzer eher verwirren, als ihnen zu helfen. Dazu gehören System-Dateien, die nur das Betriebssystem für sich benötigt und die nicht für den Benutzer gedacht sind.

System-Dateien werden mit einem speziellen "Datei-Attribut" gespeichert, das sie für DIR unsichtbar macht. Das können Sie mit dem CP/M-Programm SET.COM erreichen, das zum Lieferumfang von Joyce gehört und auf einer der Systemdisketten gespeichert ist.

Ab und zu ist man aber doch neugierig und will unbedingt wissen, welche versteckten Dateien die Diskette enthält. Für diesen Fall hält CP/M Plus den DIRSYS-Befehl bereit:

```
A>DIRSYS
A>DIRS *.COM
```

Auf der Locoscript-Diskette sind verschiedene Dateien im SYS-Format gespeichert.

2.6.5 USER

Ein vollständig gefülltes Inhaltsverzeichnis sieht sehr unübersichtlich aus. Eine Möglichkeit, es lesbarer zu machen, kennen Sie bereits in der Form der Systemdateien. Doch diese Dateien sind mehr oder weniger unsichtbar. Aus diesem Grund kennt CP/M ein zusätzliches Feature, die sogenannten "Benutzerbereiche", auf Englisch "User Areas".

Normalerweise arbeiten Sie, ohne es so richtig zu bemerken, im Benutzerbereich 0. Geben Sie aber bitte einmal diesen Befehl ein:

```
A>USER 1
```

Der Systemprompt ändert sich etwas:

```
1A>
```

Dies zeigt an, daß Sie im Userbereich 1 arbeiten. Wenn Sie jetzt DIR eingeben, zeigt das System die Meldung "No File" an. Der USER-Befehl schaltet also sozusagen zwischen verschiedenen Inhaltsverzeichnissen ein und derselben Diskette um. Allerdings bleibt das Limit der 64 Dateien pro Diskette trotz USER-Befehl erhalten.

Geben Sie bei USER oder USE keinen Wert an, fragt CP/M ausdrücklich nach:

```
A>USER
Enter User #: 3
3A>
```

Eine sehr praktische Kurzform kombiniert die Bereichsnummer mit dem Umschaltbefehl für Laufwerke:

```
A>14M:
14M>3B:
3B>
```

Im allgemeinen arbeitet man bei Disketten niedriger Speicherkapazität wie beim Joyce nicht mit verschiedenen Benutzerbereichen. Sie sind hauptsächlich für Festplatten gedacht.

2.7 Nachbrenner für residente Befehle

Mit den eingebauten Befehlen kann man unter CP/M nur die allerwichtigsten Aufgaben zur Verwaltung der Disketten leidlich gut erfüllen. Andererseits würden komfortablere Routinen mehr Speicherplatz kosten, der dann den Anwenderprogrammen fehlen würde.

Digital Research hat hier in die Trickkiste gegriffen. Solange ein residenter Befehl die geforderte Leistung ausführen kann, ist das kein Problem. Will aber der Benutzer eine komplizierte Aufgabe vom Computer erfüllt haben, lädt CP/M Plus von der Diskette ein Programm in den Speicher, das diese Aufgabe erfüllt.

Ständig im Speicher steht also nur ein Rumpfprogramm, das bei Bedarf das Gesamtprogramm von der Diskette holt. Diese Programme besitzen sinnigerweise denselben Namen wie die residenten Befehle:

DIR	DIR.COM
ERA/ERASE	ERASE.COM
REN/RENAME	RENAME.COM
TYP/TYPE	TYPE.COM

Sie finden diese Programme auf Ihrer CP/M-Systemdiskette, mit der Sie auch das Betriebssystem nach dem Einschalten des Computers starten. Wenn Sie die zusätzlichen Fähigkeiten der residenten Befehle nutzen wollen, müssen diese Programme auf der jeweils angemeldeten Diskette vorhanden sein.

2.7.1 DIR.COM für übersichtliche Verzeichnisse

Eine einfache Auflistung aller Dateien einer Diskette ist nicht sehr aussagekräftig. Zumindest alphabetisch sortiert oder mit Angabe der Dateigröße sollte die Anzeige schon erfolgen.

Dies und noch viel mehr schafft DIR.COM. Sie rufen DIR wie gewohnt wahlweise mit einem ein- oder mehrdeutigen Dateinamen auf. Zusätzlich fügen Sie aber in eckigen Klammern die gewünschte Option an. Die wichtigste Option heißt [FULL]:

```
A>DIR [FULL]
```


Finden Sie auf Ihrer Tastatur die eckigen Klammern nicht? Sie verbergen sich hinter den deutschen Umlauten Ä und Ü. Sofern sie den deutschen Zeichensatz auf dem Bildschirm eingeschaltet haben, geben Sie eben diese Umlaute ein:

```
A>DIR ÄFULLÜ
```

Der Effekt ist genau derselbe. CP/M Plus zeigt ein sehr schön geordnetes und ausführliches Inhaltsverzeichnis an:

```
A>DIR [FULL]

Scanning Directory ...

Sorting Directory ...

Directory For Drive A: User 0
```

Name		Bytes	Recs	Attributes	Name		Bytes	Recs	Attributes
BAS	COM	28k	224	Dir RW	DIR	COM	15k	114	Dir RW
DISCKIT	COM	7k	56	Dir RW	ED	COM	10k	73	Dir RW
ERASE	COM	4k	29	Dir RW	J14GCPM3	EMS	40k	320	Dir RW
KEYS	WP	1k	7	Dir RW	LANGUAGE	COM	1k	8	Dir RW
PALETTE	COM	1k	8	Dir RW	PAPER	COM	2k	16	Dir RW
PIP	COM	9k	68	Dir RW	PROFILE	GER	1k	2	Dir RW
RENAME	COM	3k	23	Dir RW	RPED	BAS	7k	56	Dir RW
RPED	SUB	1k	1	Dir RW	SET	COM	11k	81	Dir RW
SET24X80	COM	1k	8	Dir RW	SETDEF	COM	4k	32	Dir RW
SETKEYS	COM	2k	16	Dir RW	SETLST	COM	2k	16	Dir RW
SETSIO	COM	2k	16	Dir RW	SHOW	COM	9k	66	Dir RW
SUBMIT	COM	6k	42	Dir RW	TYPE	COM	3k	24	Dir RW

Total Bytes = 170k Total Records = 1306 Files Found = 24
Total 1k Blocks = 170 Used/Max Dir Entries For Drive A: 27/ 64

In der ersten Spalte sehen Sie den Dateinamen samt Namenserweiterung. Sie werden gefolgt von der Programmlänge in KByte und Records (128-Byte-Aufzeichnungen) sowie den Dateiattributen.

Als Dateiattribute sind SYS, DIR, RW und RO vorgesehen. SYS zeigt an, daß es sich bei einer Datei um ein Systemfile handelt, DIR hingegen steht für normale Dateien. RW ist die Abkürzung für "Read/Write", RO die Kurzform für "Read Only". Normale Dateien befinden sich im Read/Write-Status. Das heißt, daß sie sowohl gelesen als auch beschrieben und gelöscht werden können. Besonders schützenswerte Files, die man nur lesen will, kann man in den Read Only-Status überführen. Solche Dateien lassen sich nicht mehr mit ERA löschen und auch nicht mehr beschreiben. Read-Only-Dateien sind eine Spezialität, die man mit SET.COM erzeugen kann.

Zusätzlich zu diesen bereits eingeführten Dateiattributen erlaubt es CP/M Plus noch, bis zu vier benutzerdefinierte Attribute einzuführen. Diese tragen die Bezeichnungen F1, F2, F3 und F4. Sie werden aber vom Computer nicht weiter ausgewertet, so daß dies im Aufgabenbereich von speziell dafür geschriebenen Programmen oder des Computerbenutzers liegt.

Die anderen von DIR.COM ausgewerteten Optionen sehen so aus:

ATT — Zeigt die benutzerdefinierten Dateiattribute F1, F2, F3 und F4 an.

Beispiel: DIR [ATT]

DATE — Gibt zu jeder Datei des Datums und die Uhrzeit des Zeitpunkts der Erstellung beziehungsweise des letzten Zugriffs an. Solche Dateien müssen aber mit INITDIR .COM für die Datumsstempelung vorbereitet werden.

Beispiel: DIR B:*.COM [DATE]

DIR — Listet nur Dateien auf, die keine Systemfiles sind.

Beispiel: DIR *.COM [DIR]

DRIVE=ALL — Führt die Dateien aller vorhandenen und bereits angesprochenen Laufwerke auf.

Beispiel: DIR [DRIVE=ALL]

DRIVE=(A,B) — Zeigt das Verzeichnis der Laufwerke A: und B:. Auch andere Kombinationen sind natürlich zulässig.

Beispiel: DIR [DRIVE=(M, B, A)]

DRIVE=n — Gibt eine Liste der Dateien nur für das entsprechende Laufwerk "n:" aus. Diese Option ist identisch mit DIR n: [...]

Beispiel: DIR [DRIVE=B]

EXCLUDE — DIR.COM zeigt alle Dateien, bis auf diejenigen, die ausdrücklich ausgeschlossen werden.

Beispiel: DIR [EXCLUDE] *.TXT

FF — Falls der Drucker mit ALT-P parallel zum Bildschirm geschaltet wurde, sendet DIR.COM einen Blattvorschub (Form-Feed) vor der Ausgabe an den Drucker.

Beispiel: DIR [FF]

FULL — Diese Option ruft das oben bereits gezeigte Inhaltsverzeichnis auf.

Beispiel: DIR [FULL]

LENGTH=n — Legt fest, nach wieviel Zeilen ein Titelkopf auf dem Bildschirm angezeigt beziehungsweise an den Drucker ein Form-Feed gesandt wird.

Beispiel: DIR [LENGTH=20]

MESSAGE — Wenn Sie bei DIR.COM die USER- oder DRIVE-Option verwenden, läßt DIR diejenigen Laufwerke und Benutzerbereiche aus, in denen keine Dateien vorhanden sind. Die MESSAGE-Option führt dazu, daß alle aufgeführt werden, bei Bedarf mit der Meldung "No File".

Beispiel: DIR [DRIVE=ALL,MESSAGE]

NOPAGE — Sobald der Bildschirm vollgeschrieben ist, zeigt der Computer die Meldung "Press RETURN to Continue" und wartet auf einen Tastendruck. Geben Sie aber die NOPAGE-Option an, wird diese Meldung unterdrückt.

Beispiel: DIR [USER=ALL,MESSAGE,NOPAGE]

NOSORT — DIR.COM sortiert normalerweise die Verzeichniseinträge alphabetisch. Die NOSORT-Option verhindert das.

Beispiel: DIR [NOSORT]

RO — Nur schreibgeschützte Dateien (RO-Files) werden aufgeführt.

Beispiel: DIR [RO]

RW — Nur RW-Dateien, die sowohl gelesen als auch verändert werden können, listet DIR.COM auf.

Beispiel: DIR [RW]

SIZE — Zeigt ein verkürztes Inhaltsverzeichnis, das nur Dateinamen und Filegrößen enthält (siehe Bild).

Beispiel:

```
A>DIR [SIZE]

A: BASIC    COM 28 k : DIR        COM 15 k : DISCKIT  COM  7k
A: ED       COM 10 k : ERASE      COM  4 k : J14GCPM3 EMS 40k
A: KEYS     WP   1 k : LANGUAGE   COM  1 k : PALETTE  COM  1k
A: PAPER    COM  2 k : PIP        COM  9 k : PROFILE  GER  1k
A: RENAME   COM  3 k : RPED       BAS  7 k : RPED     SUB  1k
A: SET      COM 11 k : SET24X80   COM  1 k : SETDEF   COM  4k
A: SETKEYS  COM  2 k : SETLST     COM  2 k : SETSIO   COM  2k
A: SHOW     COM  9 k : SUBMIT     COM  6 k : TYPE     COM  3k

Total Bytes     = 170k  Total Records =  1306  Files Found =  24
Total 1k Blocks = 170   Used/Max Dir Entries For Drive A:  27/64
```

SYS — Listet nur Dateien auf, die das SYS-Attribut tragen und somit Systemfiles sind.

Beispiel: DIR [SYS]

USER=ALL — Durchsucht alle Benutzerbereiche nach Dateien.

Beispiel: DIR [USER=ALL]

USER=(1,2) — Durchsucht die angegebenen Userbereiche nach Files.

Beispiel: DIR [USER=(1,2,14,15)]

USER=n — Untersucht nur den einen angegebenen Bereich. Gleichwertig ist die Gn-Option.

Beispiele: DIR [USER=4]
DIR [G4]

Die verschiedenen Optionen können Sie auch kombinieren, soweit das sinnvoll ist. Diese werden dann durch Kommas voneinander getrennt:

A>DIR [FULL,USER=ALL,DRIVE=ALL,NOSORT,NOPAGE,MESSAGE]

Durch Kombination der einzelnen Optionen erhalten Sie eine Auflistung des Diskketteninhalts, die bestimmt Ihren Wünschen entspricht. Und wenn Sie vorher ALT-P gedrückt haben, protokolliert der Joyce-Drucker alles mit.

2.7.2 ERASE.COM schafft Sicherheit

ERA kann sowohl mit eindeutigen Dateinamen als auch solchen, die Wildcards enthalten, aufgerufen werden. Aber gerade bei der Angabe mehrdeutiger Dateinamen "rutscht" schnell mal eine Datei durch, die eigentlich noch nicht gelöscht werden sollte. Da wäre es schön, wenn CP/M alle Dateinamen vor dem Löschen anzeigen und den Benutzer fragen würde, ob alles wie vorgesehen läuft.

Das ist mit ERASE.COM durchaus möglich. Die einzige zulässige Option ist hier CONFIRM:

```
A>ERA *.COM  [CONFIRM]
A: BASIC     .COM (Y/N)? N
A: DIR       .COM (Y/N)? Y
```

```
A: ED          .COM (Y/N)? Y
A: ERASE       .COM (Y/N)? N
A: LANGUAGE    .COM (Y/N)? ^C

*** Aborted by ^C ***
A>
```

2.7.3 RENAME.COM - Mehrdeutigkeit gestattet

RENAME benennt - als residentes Kommando - Dateien nur dann um, wenn ihre Namen eindeutig sind. Manchmal wäre es aber recht praktisch, ganzen Dateigruppen auf einmal neue Namen zu geben. Denkbar ist zum Beispiel der Fall, daß alle *.TXT-Dateien in *.TEX-Dateien umbenannt werden müssen, weil ein Programm diese Namenserweiterung verlangt.

REN weiß da selber nicht mehr weiter und wendet sich deshalb an seinen "großen Bruder" RENAME.COM. Dieser gestattet durchaus die folgende Konstruktion:

```
A>RENAME *.TEX=*.TXT
```

Die Platzhalter * und ? dürfen auch an anderen Positionen stehen, müssen aber in den beiden Dateinamen exakt übereinstimmen. Deshalb sind folgende Befehle zulässig:

```
A>REN F*.COM=E*.COM
A>REN ??XYZ.*=??ABC.*
A>REN *.CIM=*.COM
```

Verboten sind hingegen beispielsweise diese Kommandos:

```
A>REN *.COM=DIR.*
A>REN ??AAAA.TXT=BB?.TXT
A>REN *.*=*.COM
```

CP/M Plus "beschwert" sich bei den letzten drei Befehlen gehörig:

```
ERROR: Not renamed, BASIC.COM Invalid wildcard.
```


2.7.4 TYPE.COM wartet auf Sie

Immer wenn CP/M Plus nach Eingabe des TYPE-Befehls eine Diskettendatei auf dem Bildschirm auflistet, ist es so höflich, auf einen Tastendruck zu warten, wenn der Bildschirm vollgeschrieben ist.

Gefällt Ihnen dieses "Feature" nicht, können Sie es mit einer Option abschalten. Sobald Sie die NOPAGE-Option aufrufen, lädt CP/M das Programm TYPE.COM in den Speicher und läßt die Datei ohne Zwischenstopps über den Bildschirm flitzen:

```
A>TYPE KEYS.WP [NOPAGE]
```

Haben Sie aber mit SETDEF.COM den Seitenmodus der Bildschirmausgabe abgeschaltet, können Sie TYPE auch mit der Angabe PAGE aufrufen. TYPE.COM verhält sich dann, als wäre der Computer in den Seitenmodus ("Console Page Mode") geschaltet:

```
A>TYPE KEYS.WP [PAGE]
```

TYPE.COM wird aber auch aktiv, wenn Sie TYPE ohne Angabe eines Dateinamens aufrufen. In diesem Fall fragt TYPE.COM nach:

```
A>TYPE
Enter file: KEYS.WP
```

2.8 Die CP/M-Dienstprogramme

2.8.1 PIP - der Alleskopierer

Das CP/M-Programm PIP.COM kann Dateien zwischen allen möglichen Diskettenlaufwerken und anderen Peripheriegeräten übertragen. Dazu gibt man den Namen der Ziel- und der Quelldatei an:

```
A>PIP ziel=quell
```

Sie können verschiedene Laufwerksnamen und beliebige Namenserweiterungen angeben:

```
A>PIP B:DATEI=M:FILE.DAT
A>PIP M:TEXT.TXT=T.T
```

Soll die Datei auf dem Ziellaufwerk denselben Namen erhalten wie die Quelldatei, können Sie den Zielnamen weglassen:

```
A>PIP B:=DATEI
A>PIP M:=B:TEXT
```

Auch mehrdeutige Dateinamen dürfen angegeben werden. Allerdings erhalten die Zieldateien dann stets den Namen der Quelldateien:

```
A>PIP B:=*.*
A>PIP A:=M:*.COM
```

Mehrere Dateien - vorrangig Textdateien - lassen sich zu einer Riesendatei verbinden, indem Sie alle Dateien durch Kommas getrennt angeben:

```
A>PIP GESAMT=TEIL1,TEIL2,TEIL3,TEIL4
```

Die anderen Peripheriegeräte sind mit PIP.COM ebenfalls ansprechbar. So trägt der Drucker den Namen LST:, der Bildschirm die Abkürzung CON: (Konsole). AUX: repräsentiert die serielle Schnittstelle.

Auf diese Weise können Sie sehr einfach Dateien auf dem Drucker ausgeben oder über die serielle Schnittstelle absenden:

```
A>PIP LST:=BRIEF
A>PIP AUX:=MELDUNG.TXT
```

Als Eingabegeräte werden lediglich die serielle Schnittstelle und die Tastatur unterstützt. Sie tragen die bereits bekannten Namen AUX: und CON:. So können Sie eine recht primitive Schreibmaschine mit PIP.COM simulieren:

```
A>PIP LST:=CON:
```

Am Zeilenende müssen Sie hier jeweils RETURN und ALT-J drücken, am Schluß des Textes ALT-Z. Korrigieren können Sie Ihre Tippfehler aber nicht auch wenn das vielleicht am Bildschirm so aussieht.

PIP ist aber nicht nur ein Kopierprogramm, es kann Dateien auch "filtern". So legt beispielsweise das Textverarbeitungsprogramm WordStar Dateien in einem verschlüsselten Format an. Denn der letzte Buchstabe jedes Wortes in einem Text besitzt ein gesetztes Highbit. Solche Texte kann man sich unmöglich mit TYPE ansehen. Aber PIP kann sie mit der Z-Option ("Zero") filtern:


```
A>PIP CON:=TEXT.WS [Z]
```

Es gibt eine ganze Reihe weiterer Filter, von denen die wichtigsten hier aufgeführt sind:

C Confirm
Bei jeder Datei wird vorher gefragt, ob der Benutzer sie kopieren möchte.

Dn Delete
Alle Zeichen rechts von Spalte n werden in der Zieldatei gelöscht.

E Echo
Beim Kopieren werden die Dateien auch am Bildschirm angezeigt.

F No Form Feeds
Alle Form-Feeds werden aus der Zieldatei gelöscht.

Gn Get/Goto User
Dateien werden aus dem Benutzerhandbereich n geholt (Quelldatei) oder dorthin geschrieben (Zieldatei).

H Hex Files
Intel-Hexdateien, die MAC.COM erzeugt, werden auf Korrektheit geprüft.

L Lower Case
Alle Großbuchstaben werden in Kleinbuchstaben umgewandelt.

N Line Numbers
In der Zieldatei werden Zeilennummern eingefügt.

N2 Line Numbers (2)
Zeilennummern mit führenden Nullen in der Zieldatei.

Pn New Page
Allen Zeilen wird ein Seitenvorschub eingefügt.

R Read SYS files
Auch Systemdateien werden kopiert.

Tn Tabs
Tabulatoren werden in n Leerzeichen umgewandelt.

U Upper case
Alle Kleinbuchstaben werden durch Großbuchstaben ersetzt.

V Verify
Prüft bei Diskettendateien, ob sie korrekt geschrieben worden sind.

W Write over RO files
Überschreibt Read-Only-Dateien ohne Warnung.

Z Zero high bit
Setzt das siebte Bit aller Zeichen auf Null.

2.8.2 SHOW zeigt Systemparameter

Wichtige Informationen über Ihre Disketten liefert das Dienstprogramm SHOW.COM. Es wird mit folgenden Parametern aufgerufen:

A>SHOW [SPACE]

Zeigt den noch freien Speicherplatz einer Diskette an.

A>SHOW [DRIVE]

Listet Eigenschaften über die Formatierung einer Diskette.

A>SHOW [USER]

Gibt an, welche Benutzerbereiche belegt sind.

A>SHOW [DIR]

Meldet, wieviele Einträge im Inhaltsverzeichnis der Diskette noch frei sind.

A>SHOW [LABEL]

Zeigt den Namen einer Diskette an, der mit SET.COM festgelegt wurde.

2.8.3 SET - Systemfiles usw.

SET.COM ist ein Vielzweckprogramm, das Dateien auf der Diskette bestimmte Attribute verleiht.

A. Disketten mit Namen

Unter CP/M Plus dürfen Disketten auch Namen besitzen. SET legt diese Namen fest:

A>SET [NAME=CPM3.DSK]

Er kann dann mit SHOW [LABEL] wieder abgefragt werden.

B. Read-Only-Dateien

Solche Dateien können nicht gelöscht oder überschrieben werden. Lediglich das Lesen ist erlaubt:

A>SET datei [RO]

Aufgehoben wird dieser Schutz mit [RW]

C. Systemdateien

Systemdateien tauchen im Inhaltsverzeichnis einer Diskette normalerweise nicht auf. Lediglich DIRSYS bringt sie an den Tag.

A>SET datei [SYS]

Wieder sichtbar werden sie mit [DIR]

D. Dateischutz und Zeitstempelung

CP/M Plus erlaubt es auch, Dateien gegen unbefugten Zugriff zu schützen und mit Zeitstempelung zu versehen. Doch die Zeitstempelung macht solche Disketten für CP/M 2.2 unbrauchbar und der Zugriffschutz ist wiederum bei CP/M 2.2 schlicht und einfach unwirksam.

Damit kennen Sie die wichtigsten CP/M-Kommandos und transienten Programme. Natürlich finden Sie auf Ihren Systemdisketten weitere Programme. Aber allein schon vom Platz her kann dies keine vollständige Beschreibung aller Besonderheiten und Spezialitäten von CP/M sein. Die vorangegangenen Seiten sollten Ihnen lediglich das Rüstzeug in die Hand geben, die wichtigsten Dateioperationen ausführen zu können. Denn auch wenn Sie nicht mit CP/M arbeiten, sondern lediglich in Mallard-Basic programmieren wollen, kommen Sie nicht darum herum, ab und zu auf die Systemebene zurückzukehren und einige Befehle einzugeben.

Die anderen sehr systemspezifischen Programme wie SETKEYS, LANGUAGE und DISCKIT sind sehr ausführlich im Benutzerhandbuch Ihres Joyce-Computers erläutert.

3 Die Systemroutinen von CP/M

3.1 Das Basic Disk Operating System (BDOS)

Normale CP/M-Progamme, die nicht speziell für den Schneider-Joyce geschrieben wurden, rufen Funktionen des Betriebssystems über das BDOS auf. Das BDOS - diese Abkürzung steht für "Basic Disk Operating System" - ist der hardwareunabhängige Teil von CP/M. Es ist bei allen CP/M-Computern identisch. Speziell für einen Computer muß lediglich das BIOS (Basic Input/Output System) entwickelt werden, das niedere Funktionen erfüllt und vom BDOS benutzt wird.

Das BDOS von CP/M Plus verarbeitet eine Reihe von Funktionsaufrufen, die die Erstellung von Programmen unter CP/M möglich machen. Im folgenden finden Sie eine Liste der BDOS-Funktionen. Für eine vollständige Aufstellung und weitergehende Informationen zur Programmierung seien Sie auf das "CP/M Plus Operating System Programmer's Guide" von Digital Research verwiesen.

Alle BDOS-Funktionen werden nach demselben Schema aufgrufen. Stets wird die Funktionsnummer in das C-Register des Prozessors geladen. Danach erfolgt ein Sprung in das BDOS über die Adresse 0005H. Soll beispielsweise mit der BDOS-Funktion 13 das Diskettensystem zurückgesetzt werden, sieht das im 8080-Maschinencode so aus:

```
DISK$RESET  MVI   C,13
            CALL  5
```

Einige Funktionen erwarten Parameter, Werte also, mit denen sie arbeiten können. 8-Bit-Werte werden an das BDOS im E-Register übergeben, 16-Bit-Werte im DE-Doppelregister:

```
SET$DMA     MVI   C,26
            LXI   D,3FACH
            CALL  5
SET$DRIVE   MVI   C,14
            MVI   E,1
            CALL  5
```

Die anderen Register sind hingegen für die Parameterübergabe tabu. Falls doch einmal mehrere Werte übermittelt werden müssen, dient das DE-Doppelregister als Zeiger auf einen Speicherbereich, in dem die Werte abgelegt sind.

Schreibt eine BDOS-Funktion Werte an das aufrufende Programm zurück, werden diese im Akkumulator (8 Bit) oder HL-Register (16 Bit) übertragen.

Die BDOS-Routinen dürfen alle Z80-Register, die auch beim 8080-Prozessor vorhanden sind, verändern. Lediglich der alternative Registersatz und die Indexregister IX und IY des Z80-Prozessors sind vor dem BDOS sicher.

Liste der BDOS-Funktionen:

Nr.	Funktion	Übergeben	Erhalten
0	Warmstart des Computers		
1	Tastaturabfrage		A=Taste
2	Bildschirmausgabe	E=Zeichen	
3	RS232 lesen		A=Zeichen
4	RS232 schreiben	E=Zeichen	
5	Druckerausgabe	E=Zeichen	
6	Direkte Ein-/Ausgabe:		
	-Tastatur lesen ohne Warten	E=0FFH	A=Zeichen
	-Tastaturstatus abfragen	E=0FEH	A=Status
	-Auf Taste warten	E=0FDH	A=Zeichen
	-Bildschirmausgabe	E=Zeichen	
7	Status von RS232 (Eingabe)		A=Status
8	Status von RS232 (Ausgabe)		A=Status
9	Stringausgabe	DE=Adresse	
10	String von Tastatur lesen	DE=Adresse	
11	Tastaturstatus ermitteln		A=Status
12	CP/M-Versionsnummer erfragen		HL=Version
13	Diskettensystem zurücksetzen		
14	Laufwerk anwählen	E=Laufwerk	A=Fehlercode
15	Existierende Datei öffnen	DE=FCB	A=Fehler
16	Datei schließen	DE=FCB	A=Fehler
17	Directory-Eintrag suchen	DE=FCB	A=Fehler
18	Nächster Directory-Eintrag		A=Fehler
19	Datei löschen	DE=FCB	A=Fehler
20	Sequentiell aus Datei lesen	DE=FCB	A=Fehler
21	Sequentiell schreiben	DE=FCB	A=Fehler

22	Datei erzeugen	DE=FCB	A=Fehler
23	Datei umbenennen	DE=FCB	A=Fehler
24	Momentane Laufwerke erfragen		HL=Vektor
25	Aktuelles Laufwerk erfragen		A=Nummer
26	DMA-Adresse festlegen	DE=Adresse	
27	Belegungstabelle holen		HL=Adresse
28	Diskette schreibschützen		
29	R/O-Laufwerke ermitteln		HL=Vektor
30	Dateiattribute setzen	DE=FCB	A=Fehler
31	DPB-Adresse ermitteln		HL=Adresse
32	Benutzerbereich		
	-Nummer erfragen	E=0FFH	A=Nummer
	-Bereich festlegen	E=Nummer	
33	Wahlfrei lesen (Random)	DE=FCB	A=Fehler
34	Wahlfrei schreiben	DE=FCB	A=Fehler
35	Dateigröße ermitteln	DE=FCB	A=Fehler
36	Random-Satznummer festlegen	DE=FCB	
37	Laufwerke zurücksetzen	DE=Vektor	
38	Access Drive *		
39	Free Drive *		
40	Wahlfrei Schreiben & Füllen	DE=FCB	A=Fehler
41	Test and Write Record *		
42	Lock Record *		
43	Unlock Record *		

* Diese Funktionen werden nur von dem Mehrbenutzersystem MP/M unterstützt und sind in CP/M nicht implementiert.

Die bisher genannten BDOS-Funktionen sind - wenn auch teilweise mit geringerer Leistungsfähigkeit - auch in CP/M 2.2 vorhanden, dem Vorläufer von CP/M Plus. Diese ältere Version wird zum Beispiel beim Schneider CPC-464 und CPC-664 verwendet. Die folgenden BDOS-Funktionen sind ausschließlich unter CP/M Plus verfügbar.

Nr.	Funktion	Übergeben	Erhalten
43	Zähler für das Lesen mehrerer Sektoren setzen	E=Zahl	A=Fehler
45	Fehlerbehandlung des BDOS bestimmen	E=Modus	

46	Freien Platz auf einer Diskette ermitteln	E=Laufwerk	A=Fehler
47	Programm von der Diskette nachladen und starten	E=Flag	
48	Diskettenpuffer aus dem RAM zurückschreiben	E=Flag	A=Fehler
49	System Control Block lesen oder schreiben	DE=ParmAdr	A/HL=Wert
50	Direkter Aufruf von BIOS-Routinen	DE=ParmAdr	...
59	Overlay-Datei von der Diskette nachladen	DE=FCB	A=Fehler
60	RSX von CP/M Plus aufrufen	DE=ParmAdr	
98	Diskettenpuffer freimachen und Disketten reorganisieren		A=Fehler
99	Datei abschneiden	DE=FCB	A=Fehler
100	Namen für eine Diskette festlegen	DE=FCB	A=Fehler
101	Daten aus dem Disketten-label holen	E=Laufwerk	A=Wert
102	Dateidatum und Paßwort-status ermitteln	DE=FCB	A=DirCode
103	XFCB schreiben	DE=FCB	A=DirCode
104	Datum und Uhrzeit festlegen	DE=DatAdr	
105	Datum und Uhrzeit ermitteln	DE=DatAdr	A=Sekunden, DatAdr
106	Paßwort festlegen	DE=Adresse	
107	CP/M-Seriennummer ermitteln		DE=Adresse
108	Programm-Returncode		
	-holen	DE=0FFFFH	HL=Code
	-festlegen	DE=Code	
109	Konsolenmodus		
	-holen	DE=0FFFFH	HL=Modus
	-festlegen	DE=Modus	
110	Stringbegrenzer für BDOS 9		
	-holen	DE=0FFFFH	A=Zeichen
	-festlegen	E=Zeichen	
111	String auf dem Bildschirm ausgeben	DE=ParmAdr	
112	String auf dem Drucker ausgeben	DE=ParmAdr	
152	String in einen FCB-gerechten Namen wandeln	DE=ParmAdr	HL=Code

3.2 Das Basic Input/Output System (BIOS)

Das BIOS erfüllt niedere Aufgaben wie das direkte Lesen und Schreiben von Diskettensektoren und die Umschaltung der Speicherbänke. Einige der BIOS-Funktionen lassen sich durchaus auch sinnvoll in Anwenderprogrammen verwenden, während andere absolut keinen Sinn in diesem Bereich ergeben und lediglich vom BDOS benötigt werden. Eine vollständige Erläuterung des BIOS-Aufbaus und der BIOS-Funktionen enthält das "CP/M Plus Operating System System Guide" von Digital Research.

Die Konventionen für die Parameterübergabe an BIOS-Routinen unterscheiden sich etwas von denen, die Sie von den BDOS-Funktionen her kennen. Daten werden nämlich im C-Register (8 Bit) oder BC-Doppelregister (16 Bit) übergeben und über den Akkumulator oder das HL-Doppelregister von der BIOS-Routine zurückgegeben.

Unter CP/M 2.2 war es üblich, BIOS-Routinen anhand ihres Eintrags in der BIOS-Sprungtabelle aufzurufen. Dazu holte man sich aus der Adresse 0001H den Warmstart-Einsprung des BIOS, zog von diesem drei Bytes ab und erhielt damit den Start der BIOS-Sprungtabelle. Über die Formel Start + 3 * Funktionsnummer gelangte man zum richtigen Eintrag, den man per CALL aufrief.

In CP/M Plus ist das alles etwas komplizierter geworden. Denn CP/M Plus verwaltet mehrere Speicherbänke. Der größte Teil des BIOS steht nun nicht mehr in der Programmbank, sondern in der Systembank des Speichers. Ruft man also über den gerade beschriebenen Weg eine BIOS-Routine auf, springt man mitten in die Systembank hinein - ohne sie allerdings in den Speicher eingeblendet zu haben! Daß dies zum Systemabsturz führt, dürfte klar sein.

Aus diesem Grund gibt es auch einige Programme von CP/M 2.2 die unter CP/M Plus nicht mehr korrekt arbeiten. Da aber viele CP/M-Programme zumindest ihre Bildschirmausgabe und Tastaturabfrage über das BIOS abwickeln, können diese äußerst wichtigen BIOS-Funktionen auf die für CP/M 2.2 übliche Art aufgerufen werden.

Alle anderen BIOS-Funktionen sollten über die BDOS-Funktion 50 gestartet werden. Diese heißt "Direct BIOS Call" und übernimmt im DE-Register einen Zeiger auf einen Parameterblock. Dieser ist wie folgt aufgebaut:

```
BIOS$PB:    DB    FUNC      ;  Nummer der BIOS-Funktion
            DB    ACCU      ;  Wert des Akkumulators
            DW    BCREG     ;  Wert des BC-Registers
```

```
                DW    DEREG     ;  Wert des DE-Registers
                DW    HLREG     ;  Wert des HL-Registers

BIOS$CALL:      MVI   C,50      ;  BDOS-Funktionsnummer 50
                LXI   DE,BIOS$PB;  Zeiger auf Parameterblock
                CALL  5         ;  Aufruf des BDOS
```

Aus Sicherheitsgründen fängt die BDOS-Funktion 50 Aufrufe der BIOS-Funktion 27, "Select Memory", ab.

Liste der BIOS-Funktionen:

Nr.	Funktion	Übergeben	Erhalten
0 BOOT	Kaltstart des Computers		
1 WBOOT	Warmstart von CP/M		
2 CONST	Tastaturstatus erfragen		A=Status
3 CONIN	Auf Tastendruck warten		A=Zeichen
4 CONOUT	Bildschirmausgabe	C=Zeichen	
5 LIST	Druckerausgabe	C=Zeichen	
6 AUXOUT	Ausgabe auf RS232	C=Zeichen	
7 AUXIN	Eingabe von RS232		A=Zeichen
8 HOME	Diskette auf Spur 0		
9 SELDSK	Laufwerk auswählen	C=Laufwerk	HL=DPH-Adresse
10 SETTRK	Spur anwählen	BC=Spur	
11 SETSEC	Sektor anwählen	BC=Sektor	
12 SETDMA	DMA-Adresse wählen	BC=Adresse	
13 READ	Sektor lesen		
14 WRITE	Sektor schreiben		
15 LISTST	Druckerstatus ermitteln		A=Status
16 SECTRN	Sektorübersetzung	DE=TrnAdr BC=LogSek	HL=PhysSek
17 CONOST	Bildschirmstatus		A=Status
18 AUXIST	Eingabestatus der RS232		A=Status
19 AUXOST	Ausgabestatus der RS232		A=Status
20 DEVTBL	Gerätetabelle ermitteln		HL=Adresse
21 DEVINI	Gerät initialisieren	C=Nummer	
22 DRVTBL	Drive-Tabelle ermitteln		HL=Adresse
23 MULTIO	Sektorzähler festlegen	C=Zahl	

24 FLUSH	Blocking/Deblocking	
25 MOVE	Speicher kopieren	DE=Quelle HL=Ziel BC=Zahl
26 TIME	(nicht implementiert)	
27 SELMEM	(von BDOS-50 abgefangen)	
28 SETBNK	Speicherbank für Diskettenoperationen	A=Bank
29 XMOVE	Speicherbänke für MOVE festlegen	B=Zielbank C=Quellbank
30 USERF	Aufruf von XBIOS-Routinen bei Joyce	

3.3 XBIOS - die Geheimnisse von Joyce

In den vorangehenden Kapiteln haben wir uns in die Grundlagen der Programmierung des CP/M-Betriebssystems eingearbeitet und die Funktionen des BDOS und BIOS kennengelernt.

Das BDOS und BIOS besitzen eine Reihe von Routinen, die sich von CP/M-Programmen aus aufrufen lassen. Sie sind aber sehr allgemein gehalten und erlauben nur sehr beschränkt die Entwicklung von Programmen, die besondere Systemeigenschaften ausnutzen.

Herkömmliche CP/M-Computer belassen es bei diesen BDOS- und BIOS-Einsprüngen und bieten keine weiteren Systemroutinen. Bei diesen Computern werden meistens nur Anwendungsprogramme angeboten, die Standardmöglichkeiten nutzen und nicht die oft viel vorteilhafteren systemspezifischen Eigenschaften.

Aber Amstrad wäre nicht Amstrad, wenn Joyce ein Standardcomputer wäre, der nicht über versteckte Besonderheiten verfügte. Irgendwie muß man doch die Tastatur umbelegen, den Diskettencontroller programmieren oder die Bildschirmausgabe beeinflussen können. Schließlich schaffen das die Dienstprogramme wie SETKEYS.COM und DISCKIT.COM auch. Bloß wie?

Die Lösung ist relativ einfach: Der Joyce-PCW besitzt neben der BIOS-Sprungleiste, die dem CP/M-Standard entspricht, eine weitere Sprungleiste, die man wohl "Extended BIOS Jumpblock" nennen müßte. Im folgenden wird der Einfachheit halber vom "XBIOS" gesprochen. Diese Abkürzung hat sich auch bei anderen Computern eingebürgert.

Da stellt sich natürlich sofort die Frage, wo sich diese Sprungleiste befindet. Also schnell mal den Systemdebugger SID.COM geladen und den CP/M-Speicher abgesucht. Vermuten wird man die XBIOS-Leiste zuerst am oberen Speicherende. Denn der Speicher ist ja beim Joyce-PCW wie folgt aufgebaut:

0000H	Für das Betriebssystem reserviert
0100H	Beginn des Programmspeichers von CP/M
F606H	Start des BDOS
FC00H	Beginn des BIOS

Nachdem alle normalen CP/M-Programme diese Speicherverteilung erwarten, war natürlich der Spielraum der Programmierer bei Amstrad relativ eng. Ein denkbarer Platz wäre oberhalb des BIOS gewesen.

Doch das hätte zur Folge gehabt, daß Amstrad den für Anwenderprogramme nutzbaren Speicherplatz hätte verkleinern müssen. Nachdem diese Lösung folglich nicht sehr attraktiv erschien, hat man sich etwas anderes einfallen lassen: Da das CP/M Plus-System ohnehin mit mehreren Speicherbänken arbeitet, liegt der Sprungblock des XBIOS in der Systembank.

Dieser XBIOS-Sprungblock beginnt an der Adresse 0080H in der Systembank und hat den folgenden Aufbau:

3.3.1 Diskettenorientierte XBIOS-Routinen

0080H DD INIT
Diskettentreiber initialisieren
0083H DD SETUP
Diskettenparameter festlegen
0086H DD READ SECTOR
Diskettensektor lesen
0089H DD WRITE SECTOR
Diskettensektor schreiben
008CH DD CHECK SECTOR
Sektor mit einem Speicherauszug vergleichen
008FH DD FORMAT
Eine einzelne Diskettenspur formatieren
0092H DD LOGIN
Diskettenformat ermitteln

0095H DD SEL FORMAT
Eines der üblichen Formate festlegen
0098H DD DRIVE STATUS
Status des Floppycontrollers ermitteln
009BH DD READ ID
Sektor-ID lesen
009EH DD INIT DPB
Einen Disk Parameter Block initialisieren
00A1H DD INIT XDPB
Einen XDPB initialisieren
00A4H DD ON MOTOR
Laufwerksmotor einschalten
00A7H DD TIMEOFF MOTOR
Laufwerksmotor mit Timeout-Verzögerung ausschalten
00AAH DD OFF MOTOR
Laufwerksmotor ausschalten
00ADH DD CONTROLLER READ
Floppycontroller zum Lesen programmieren
00B0H DD CONTROLLER WRITE
Floppycontroller zum Schreiben programmieren
00B3H DD SEEK TRACK
Eine Diskettenspur ansteuern

3.3.2 Programmierung der seriellen Schnittstelle

00B6H RS INIT
Serielle Schnittstelle (Kanal A) initialisieren
00B9H RS SET BAUD
Baudrate für Kanal A programmieren
00BCH RS GET PARAMS
Parameter des A-Kanals holen

3.3.3 Steuerung der Bildschirmausgabe

00BFH TE GET WINDOW & CURSOR
Holt Fenstergröße und aktuelle Cursorposition
00C2H TE RESET
Setzt den Bildschirmtreiber zurück
00C5H TE STLINE ASK
Ermittelt, ob die Statuszeile eingeschaltet ist

00C8H TE STLINE ON/OFF
Schaltet die Statuszeile ein oder aus.
00CBH TE SET INK
Legt die Stiftfarbe fest
00CEH TE SET BORDER
Legt die Rahmenfarbe fest
00D1H TE SET SPEED
Legt die Blinkgeschwindigkeit für Farben fest

3.3.4 Abfrage und Umbelegung der Tastatur

00D4H KM SET EXPAND
Definiert einen Erweiterungsstring für eine Taste
00D7H KM SET KEY
Definiert eine Taste um
00DAH KM KEY GET
Versucht, einen Tastaturcode zu lesen
00DDH KM KEY PUT
Speichert einen Tastaturcode im Tastaturpuffer
00E0H KM SET SPEED
Legt die Geschwindigkeit fest, mit der die Anschlagswiederholung der Tastatur einsetzt

3.3.5 Sonstige Hilfsroutinen

00E3H MAC GET VERSION
Stellt Maschinentyp und BIOS-Version fest
00E6H MAC GET INFO
Holt Systeminformationen und stellt fest, welche Hardware-Erweiterungen angeschlossen sind
00E9H SCR RUN ROUTINE
Blendet den Bildschirmspeicher ein und startet in diesem Kontext eine Maschinenroutine

Das Interessante daran ist, daß alle Routinen nicht nur beim Joyce-PCW verfügbar sind, sondern auch auf dem CPC-6128 von Schneider! Lediglich bei der Adressierung und dem Aufbau des Bildschirmspeichers unterscheiden sich beide Computer so stark, daß die XBIOS-Funktion SCR RUN ROUTINE beim Schneider CPC-6128 weggelassen wurde und nur auf dem Joyce verfügbar ist.

3.3.6 Zugriff auf XBIOS-Routinen

Die nächste Frage, die Sie stellen werden, dürfte klar sein. Wie um alles in der Welt werden diese doch äußerst nützlichen XBIOS-Funktionen denn nun aufgerufen? Dafür hat sich Amstrad etwas besonderes einfallen lassen. Liest man die Originaldokumentationen zu CP/M Plus von Digital Research (CP/M Plus System Guide) genau, findet man eine Liste der BIOS-Funktionen. Diese besitzen die Nummer 0 bis 29. Außerdem wurden die Funktionsnummern 31 und 32 für zukünftige Verwendung reserviert. Entscheidend ist aber die BIOS-Funktion 30. Sie heißt USERF und kann von den Programmierern, die ein CP/M Plus-System implementieren, beliebig verwendet werden.

Amstrad hat sich dafür entschieden, hier den Einsprung in die XBIOS-Routinen einzubauen. Und das funktioniert folgendermaßen: Man ruft die BIOS-Funktion USERF auf und übergibt die Adresse der gewünschten XBIOS-Routine als Inline-Parameter. Das ist ein 16-Bit-Wert, der direkt auf den Call-Befehl folgt. Ein Aufruf von TE RESET könnte so aussehen:

```
CALL    USERF
DW      00C2H
```

Dabei kann man vorher in die Register beliebige Werte laden. Diese werden an die Zielroutine unverändert weitergegeben. Bei der Rückkehr aus einer XBIOS-Routine sind die Prozessorregister entsprechend gesetzt. Sie werden von USERF nicht verändert. Allerdings ist der alternative Registersatz des Z80-Prozessors für Programmierer beim Aufruf von XBIOS-Routinen tabu. Er kann weder Werte über den XBIOS-Aufruf hinweg behalten noch zur Übergabe von Zahlen benutzt werden.

Es ergibt sich nur ein "winziges" Problem. Wer sagt denn, daß alle Versionen von CP/M Plus, die Amstrad beziehungsweise Schneider in Zukunft vertreiben wird, genau dieselbe Speichergröße haben werden? Und vom Umfang der TPA (Transient Program Area) hängt natürlich auch die Startadresse des BIOS und damit die exakte Adresse der USERF-Funktion unmittelbar ab.

Um solche Unwägbarkeiten zu vermeiden, bietet sich ein praktibler Weg an. Ein Programm, das XBIOS-Routinen nutzen will, ermittelt anhand des Wertes in der Speicheradresse 0001H den Start des BIOS und zählt zu diesem Wert dreimal 29 Bytes hinzu. Durch diese Addition von 87 erhält man genau die Adresse der USERF-Funktion, deren Position innerhalb des BIOS-Sprungblocks ja festgelegt ist.

Im unsäglichen 8080-Maschinencode, auf den wir leider wegen der Assembler MAC.COM und RMAC.COM angewiesen sind, läßt sich dies so programmieren:

```
; USERF-Adresse ermitteln und in VECTOR speichern *****

BOOT    EQU  0

LHLD    BOOT+1     ; Einsprung ins BIOS holen
LXI     D,87       ; Offset 87 zum USERF-Eintrag
DAD     D          ; Addieren
SHLD    VECTOR+1   ; In VECTOR speichern

; Vor VECTOR einen JMP abspeichern ********************

MVI     A,0C3H     ; Maschinencode für JMP
STA     VECTOR     ; Vor der Adresse in VECTOR ablegen

; XBIOS-Funktion USERF aufrufen ***********************

CALL    VECTOR     ; Über den Vektor nach USERF springen
DW      00BFH      ; Adresse der XBIOS-Routine
RET                ; Rückkehr zum aufrufenden Programm

; Vektor für USERF ************************************

VECTOR  DEFS  3
        END
```

In deutscher Pseudosprache liest sich das so:

- Hole den BIOS-Einsprung aus der Adresse 0001H.
- Addiere den Offset von 87 Bytes zum Wert
- Speichere die sich ergebende Adresse in VECTOR+1.
- Setze den Maschinenbefehl JMP (0C3H) davor.
- Rufe über VECTOR die USERF-Funktion auf.
- Übergebe ihr die Adresse der XBIOS-Funktion.
- Gehe zum aufrufenden Programm zurück.

Im folgenden sind die Routinen detailliert mit Übergabe- und Übernahmeparametern aufgelistet. Damit Sie diese XBIOS-Routinen möglichst einfach ausprobieren und mit ihnen experimentieren können, finden Sie hier ein Basic-Programm, das den Aufruf des XBIOS aus Mallard-Basic möglich macht:


```
100 ' ************************************************
110 ' *                                              *
120 ' *  XBIOS - Aufruf von Systemroutinen aus Basic  *
130 ' *                                              *
140 ' ************************************************
150 '
160 ' Maschinenroutine aus DATAs einlesen *************
170 MEMORY &HF000-1
180 FOR i=&HF000 TO &HF04A
190     READ byte
200     POKE i,byte
210 NEXT i
220 ' Variablen zeigen auf Speicherstellen ************
230 xbios=&HF04E
240 reg.bc=&HF050
250 reg.de=&HF052
260 reg.hl=&HF054
270 reg.ix=&HF056
280 reg.iy=&HF058
290 DEF FNmsb(x)=INT(x/256)
300 DEF FNlsb(x)=x-INT(x/256)*256
350 ' Daten fuer XBIOS *********************************
360 DATA 42,1,0,17,87,0,25,34,76,240,62,195,50,75,240
370 DATA 42
380 DATA 78,240,34,48,240,42,80,240,229,241,237,75,82
390 DATA 240,237,91
400 DATA 84,240,42,86,240,221,42,88,240,253,42,90,240
410 DATA 205,75,240
420 DATA 191,0,34,86,240,245,225,34,80,240,237,67,82
430 DATA 240,237,83
440 DATA 84,240,221,34,88,240,253,34,90,240,201
500 '
510 ' Beispiel: TE RESET (Zurücksetzen des Videoteils)
520 '
530 POKE xbios,&HC2:POKE xbios+1,0
540 POKE reg.bc,1:POKE reg.bc+1,1
550 adr=&HF000:CALL adr
560 END ' *********************************************
```

Die Zeilen 500 bis 550 zeigen ein einfaches Beispiel für die Anwendung des Programms. Sie demonstrieren das an der XBIOS-Funktion TE RESET, die den Bildschirmtreiber zurücksetzt. Was sie genau macht, erfahren Sie weiter unten.

Damit die Übergabe von Parametern aus einem Basic-Programm an die Maschinenroutine gelingt, zeigen verschiedene Variablen auf Speicherstellen, die von der Maschinenroutine ausgelesen werden. Über POKE können Sie so der Maschinenroutine Werte übermitteln:

xbios Die Adresse der XBIOS-Routine

reg.af Register AF (Akkumulator und Flagregister)

reg.bc BC-Register

reg.de DE-Register

reg.hl HL-Register

reg. ix Indexregister IX

reg.iy Indexregister IY

Nachdem der POKE-Befehl nur 8-Bit-Werte behandeln kann, verschiedene Routinen des XBIOS aber 16-Bit-Zahlen benötigen, enthält das Programm auch noch benutzerdefinierte Funktionen, die einen Wert in sein High- und Lowbyte aufspalten:

FNlsb (x) liefert das niederwertige Byte eines 16-Bit-Speicherwortes (least-significant byte).

FNmsb (x) liefert das höherwertige Byte eines Speicherwortes mit 16 Bit Länge (most-significant byte).

So läßt sich die Zahl 10 000 (2710H) in ihre 8-Bit-Bestandteile 39 (27H) und 16 (10H) zerlegen:

```
PRINT FNmsb (10000)
 39

PRINT FNlsb (10000)
 6
```

Die Gegenrechnung liefert den Beleg:

```
PRINT 39*256+16
 10000
```

Stellen Sie sich vor, Sie hätten folgende Routine in Maschinensprache programmiert:


```
LXI   B, 3FACH
LXI   D, 16384
CALL  USERF
DW    0086H
```

Dann stellen Sie das mit dem Basic-Programm so dar:

```
10  ' Adresse der XBIOS-Routine
15  POKE xbios, &H86
20  ' Inhalt des BC-Doppelregisters
25  POKE reg.bc, FNlsb (&H3FAC)
30  POKE reg.bc+1, FNmsb (&H3FAC)
35  ' Inhalt des DE-Doppelregisters
40  POKE reg.de, FNlsb (16384)
45  POKE reg.de+1, FNmsb (16384)
50  ' hier Aufruf der Maschinenroutine aus dem
55  ' Basic-Programm, z. B. GOSUB 10000
```

Wenn eine XBIOS-Routine Werte in den Prozessorregistern zurückgibt, können Sie diese mit PEEK aus denselben Adressen auslesen:

```
PRINT PEEK (reg.bc); PEEK (reg.bc+1)
PRINT PEEK (reg.hl)
```

In einigen der Beschreibungen zu den XBIOS-Routinen werden Sie Hinweise auf einen XDPB finden. Darunter versteht man den "Extended Disk Parameter Block". CP/M unterhält einen DPB ("Disk Parameter Block"), in dem die physikalischen Daten jedes Diskettenlaufwerks vermerkt sind. Dazu gehören die Zahl der Einträge im Inhaltsverzeichnis, die Zahl der Sektoren und Blocks und einige andere wichtige Informationen.

Aber das XBIOS kommt mit diesen Angaben nicht aus. Es benötigt weitere Werte. Deshalb wird der DPB zu einem XDPB erweitert, indem an sein Ende weitere Bytes angehängt werden. Die Adresse eines XDPB erfahren Sie über die BDOS-Funktion 31. Sie liefert die Startadresse des DPB eines Laufwerks, die mit der Startadresse des entsprechenden XDPBs zusammenfällt.

Die folgende Tabelle zeigt die Zusammenhänge zwischen DPB und XDPB.

Normaler DPB:

SPT 128 Byte-Sektoren pro Spur
BSH Block-Shift
BLM Block-Maske
EXM Extent-Maske
DSM Zahl der Blocks -1
DRM Zahl der Einträge im Directory -1
AL0 Belegung des Directory (1)
AL1 Belegung des Directory (2)
CKS Größe des Prüfsummenvektors bei Diskettenwechsel
OFF Zahl der Systemspuren
PSH Physikalischer Sektor-Shift
PHM Physikalische Sektor-Maske

Zusätzlich im XDPB:

SID Zahl der Seiten
0 = einseitig
1 = zweiseitig, Spuren abwechselnd auf der Vorder- und Rückseite
2 = zweiseitig; zuerst alle Spuren der Vorderseite, dann alle Spuren der Rückseite

TRK Spuren pro Seite

SEC 512 Byte-Sektoren pro Spur

FSC Nummer des ersten Sektors jeder Spur

SIZ Sektorgröße in Bytes

GRW Länge des Gap beim Lesen und Schreiben

GFO Länge des Gap beim Formatieren

MOD Betriebsart
Bit 7=0: Einzelspur-Operationen
Bit 7=1: Multispur-Operationen
Bit 6=0: Modulation der Aufzeichung FM-Modus
Bit 6=1: Modulation der Aufzeichung MFM-Modus
Bit 5=0: Gelöschte Adreßmarkierung nicht überlesen
Bit 5=1: Gelöschte Adreßmarkierungen überlesen
Bits 4 bis 0: Stets 0

AUT Automatisches Selektieren des Diskettenformats
AUT=0: Automatische Auswahl eingeschaltet
AUT=255: Automatische Auswahl abgeschaltet

Alle Einträge im erweiterten DPB sind 8 Bit breit, bis auf SIZ, das zwei Bytes umfaßt.

Sie dürfen diese Disketteneigenschaften verändern, soweit Sie die folgenden Beschränkungen einhalten:

Belegungsvektor (2 Bit):	maximal 91 Bytes
Prüfsummenvektor:	maximal 32 Bytes
Größe der Hash-Tabelle:	maximal 512 Bytes
Sektorgröße:	maximal 512 Bytes

Gehen Sie bis an diese Grenzen, erhalten Sie eine Diskette mit 160 Spuren, 9 Sektoren pro Spur und 128 Einträgen im Inhaltsverzeichnis.

Die drei Diskettenformate, die der Joyce verarbeiten kann, besitzen die folgenden Standardwerte:

Typ	Joyce-Format	Systemformat	Data-Only-Format
SPT	36	36	36
BSH	3	3	3
BLM	7	7	7
EXM	0	0	0
DSM	174	170	179
DRM	63	63	63
AL0	192	192	192
AL1	0	0	0
CKS	16	16	16
OFF	1	2	0
PSH	2	2	2
PHM	3	3	3
SID	0	0	0
TRK	40	40	40
SEC	9	9	9
FSC	1	65	193
SIZ	512	512	512
GRW	42	42	42
GFO	82	82	82

MOD	96	96	96
AUT	0	0	0

DD INIT 0080H

Diese XBIOS-Routine initialisiert den Diskettentreiber, setzt alle Diskettenparameter auf ihre Standardwerte und schaltet den Laufwerksmotor aus.

Es gelten die folgenden Standardwerte:

Verzögerung beim Einschalten des Motors: 1 s
Verzögerung beim Ausschalten des Motors: 5 s
Zeit zum Aufsetzen des Schreibkopfs: 4 ms
Zeit zum Abheben des Schreibkopfs: 480 ms
Steprate: 12 ms
Kopfberuhigungszeit (Head settle time): 30 ms
DMA-Modus: aus

Diese Daten sind hauptsächlich für Programmierer interessant, die einen sehr systemnah arbeitenden Diskettenmonitor schreiben oder Fremdlaufwerke anschließen wollen.

Einsprungbedingungen:
Keine

Aussprungbedingungen:
Die Register AF, BC, DE und HL sind zerstört.

DD SETUP 0083H

Die Werte, die für Floppyoperationen benutzt und von DD INIT auf ihre Standardwerte gebracht werden, lassen sich mit DD SETUP gezielt verändern.

Einsprungbedingungen:

HL zeigt auf einen Parameterblock im Speicher zwischen 0C000H und 0FFFFH. Dieser Parameterblock besitzt den folgenden Aufbau:

Byte 0: Einschaltverzögerung für den Diskettenmotor in 100 Millisekunden-Schritten

Byte 1: Ausschaltverzögerung für den Diskettenmotor in 100 Millisekunden-Schritten
Byte 2: "Write current off "-Zeit in 10 Millisekunden-Schritten
Byte 3: Kopfberuhigungszeit (Head settle time) in Millisekunden
Byte 4: Steprate des Diskettenmotors in Millisekunden
Byte 5: Abhebeverzögerung des Schreibkopfes in Einheiten zu 32 Millisekunden (Bereich 32 bis 480 ms)
Byte 6: Bits 1-7 enthalten die Aufsetzverzögerung. Bit 0 enthält die Kennung für den Betrieb im Non-DMA-Modus.

Die mit dieser XBIOS-Funktion gesetzten Werte gelten stets für beide Laufwerke.

Aussprungbedingungen:

Die Inhalte von AF, BC, DE und HL sind zerstört.

DD READ SECTOR 0086H

Diese XBIOS-Funktion liest einen Sektor von der Diskette in den RAM-Speicher ein. Diskettensektoren besitzen normalerweise eine Länge von 512 Bytes.

Einsprungbedingungen:

B enthält die Nummer der Speicherbank als Ziel
C gibt die Laufwerksnummer an
D enthält die Spurnummer
E bezeichnet die Nummer des logischen Sektors
HL zeigt auf den Zielpuffer im RAM
IX zeigt auf den XDPB im Common-Memory zwischen 0C000H und 0FFFFH.

Aussprungbedingungen:

Wenn der Sektor gelesen werden konnte, ist der Akkuinhalt zerstört und das Carry-Flag auf wahr gesetzt.

Gab es einen Lesefehler, ist Carry falsch und der Akkumulator enthält den Fehlercode:

A=0 Laufwerk nicht ansprechbar
A=2 Seek fail (Sektor nicht ansteuerbar)
A=3 Datenfehler
A=4 Keine Daten vorhanden
A=5 Adreßmarkierung fehlt auf der Diskette
A=8 Datenträger gewechselt

Der Fehler 8 bedeutet, daß die Sektornummern der Diskette nicht mit den Sektornummern im XDPB übereinstimmen.

Stets sind die Registerinhalte von BC, DE und HL verändert.

DD WRITE SECTOR 0089H

Die XBIOS-Funktion DD WRITE SECTOR schreibt einen Sektor von 512 Bytes aus dem Speicher auf die Diskette.

Einsprungbedingungen:

B enthält die Nummer der Speicherbank des Sektorpuffers
C enthält die Laufwerksnummer
D enthält die Spurnummer
E enthält die Nummer des logischen Sektors
HL zeigt auf den Sektorpuffer im RAM-Speicher
IX zeigt auf den XDPB im Speicher oberhalb 0C000H

Aussprungbedingungen:

Wenn der Sektor fehlerfrei geschrieben werden konnte, ist der Akkuinhalt zerstört und das Carry-Flag wahr.

Bei einem Fehler ist Carry falsch und der Akkumulator enthält einen Fehlercode:

A=0 Laufwerk nicht ansprechbar
A=1 Diskette schreibgeschützt
A=2 Seek fail (Sektor nicht ansteuerbar)
A=3 Datenfehler
A=4 Keine Daten
A=5 Adreßmarkierung fehlt
A=8 Datenträger gewechselt

Der Fehler 8 deutet darauf hin, daß die Sektornummern auf der Diskette nicht mit den Sektornummern übereinstimmen, die im XDPB angegeben wurden.

Stets sind BC, DE und HL verändert.

DD CHECK SECTOR 008CH

Die XBIOS-Funktion DD CHECK SECTOR vergleicht einen Diskettensektor mit einem Speicherauszug im RAM. So kann einfach festgestellt werden, ob die beiden übereinstimmen.

Einsprungbedingungen:

B enthält die Speicherbank des RAM-Puffers
C enthält die Laufwerksnummer
D enthält die Nummer der Diskettenspur
E enthält die Nummer des logischen Sektors
HL zeigt auf den Speicherbereich im RAM
IX zeigt auf den XDPB oberhalb der Adresse 0C000H

Aussprungbedingungen:

Konnte der Sektor gelesen werden, dann ist das Carry-Flag auf wahr gesetzt. Stimmen zudem noch die beiden Sektoren überein, ist auch das Zero-Flag wahr, sonst ist es falsch. In beiden Fällen ist der Akkumulatorinhalt gelöscht.

Gab es einen Lesefehler, ist das Carry-Flag des Z80-Prozessors falsch und der Akkumulator enthält den Fehlercode:

A=0 Laufwerk nicht betriebsbereit
A=2 Seek fail (Sektor nicht ansteuerbar)
A=3 Datenfehler
A=4 Keine Daten vorhanden
A=5 Adreßmarkierung fehlt im Sektor
A=8 Datenträger gewechselt

Der Fehler 8 läßt darauf schließen, daß die Sektornummern im XDPB-Block nicht mit denen der Diskette übereinstimmen.

DD FORMAT 008FH

Die XBIOS-Funktion DD FORMAT formatiert eine einzelne Spur auf der Diskette.

Einsprungbedingungen:

B enthält die Speicherbank mit dem Formatpuffer
C spezifiziert das Laufwerk
D enthält die Nummer der zu formatierenden Spur
E ist das Byte, mit dem die Sektoren gefüllt werden (normalerweise E5H)
HL zeigt auf den Formatpuffer
IX zeigt auf den XDPB im Common-Memory oberhalb 0C000H

Der Formatpuffer besitzt den folgenden Aufbau:

Erster Sektor
Byte 0: Spurnummer
Byte 1: Kopfnummer
Byte 2: Sektornummer
Byte 3: Sektorgröße entsprechend log2 (Größe)-7

Zweiter Sektor
Byte 0: Spurnummer
Byte 1: Kopfnummer
Byte 2: Sektornummer
Byte 3: Sektorgröße entsprechend log2 (Größe)-7
.
.
.

Aussprungbedingungen:

Konnte die Spur formatiert werden, ist das Carry-Flag wahr und der Akkumulatorinhalt zerstört.

Gab es einen Fehler, erkennt man das am Zustand des Carry-Flags. Es ist dann falsch. Der Akkumulator zeigt an, was schiefgegangen ist:

A=0 Laufwerk nicht bereit
A=1 Diskette schreibgeschützt
A=2 Spur konnte nicht angesteuert werden
A=3 Datenfehler
A=4 Keine Daten

A=5 Fehlende Adreßmarkierung
A=8 Sektornummern im XDPB sind falsch

Stets gilt, daß das BC-, DE- und HL-Register veränderte Inhalte haben.

DD LOGIN 0092H

Die XBIOS-Funktion DD LOGIN versucht, das Format einer Diskette zu bestimmen, die in einem der Laufwerke eingelegt ist. Sie legt dann einen passenden XDPB (Extended Disk Paramter Block) an.

Einsprungbedingungen:

Das C-Register enthält die Laufwerksnummer. Das IX-Register zeigt auf den zu initialisierenden XDPB-Block im Common-Memory oberhalb 0C000H.

Aussprungbedingungen:

Konnte das Format ermittelt werden, ist Carry auf wahr gesetzt. DD LOGIN initialisiert den XDPB und übergibt dann die folgenden Registerwerte:

A- Diskettentyp
0 - Format des Joyce-PCW
1 - Systemformat des Schneider-CPC
2 - Data-Only-Format des Schneider-CPC

DE - Größe des 2-Bit-Belegungsvektors
HL - Größe der Hash-Tabelle

Konnte DD LOGIN das Format nicht identifizieren, ist das Carry-Flag auf falsch gesetzt, und der Akkumulator enthält einen Fehlercode:

A=0 Laufwerk nicht bereit
A=2 Konnte nicht angesteuert werden
A=3 Datenfehler
A=4 Keine Daten
A=5 Fehlende Adreßmarkierung
A=6 Falsches Format

DD SEL FORMAT 0095H

Die XBIOS-Routine DD SEL FORMAT initialisiert einen XDPB-Block auf ein bestimmtes Diskettenformat. Dies geschieht unabhängig vom Format der ins Laufwerk eingelegten Disketten und wird beispielsweise von DISCKIT beim Formatieren von Disketten benötigt.

Einsprungbedingungen:

A enthält das gewünschte Format:
- 0 - das normale Format von Joyce
- 1 - das Systemformat des Schneider-CPC
- 2 - das Data-Only-Format des Schneider-CPC

IX zeigt auf den zu initialisierenden XDPB im Common-Speicher zwischen 0C000H und 0FFFFH.

Aussprungbedingungen:

Wurde DD SEL FORMAT korrekt abgearbeitet, ist das Carry-Flag auf wahr gesetzt und der Z80-Akkumulator enthält das entsprechende Diskettenformat (0, 1 oder 2). Außerdem steht im DE-Doppelregister die Größe des 2-Bit-Belegungsvektors einer Diskette und in HL die Größe der Hash-Tabelle.

Haben Sie ein falsches Format angewählt, zeigt das XBIOS dieses dadurch an, daß das Carry-Flag des Z80-Prozessors auf falsch gesetzt ist und der Akkumulator den Wert 6 enthält.

DD DRIVE STATUS 0098H

Die XBIOS-Funktion DD DRIVE STATUS ermittelt den Zustand eines Diskettenlaufwerks. Sie stellt fest, ob es betriebsbereit ist, schreibgeschützt usw.

Einsprungbedingungen:

Die Bits 0 und 1 des C-Registers enthalten die Laufwerksnummer, das Bit 2 den gewünschten Laufwerkskopf.

Aussprungbedingungen:

Im Akkumulator wird ein bitweise aufgebautes Statusbyte übergeben:

Bit 1 und 0:	Laufwerksnummer
Bit 2:	Kopfnummer
Bit 3:	nicht definiert
Bit 4:	Spur 0 eingestellt
Bit 5:	Laufwerk bereit
Bit 6:	Diskette schreibgeschützt
Bit 7:	nicht definiert

Das HL-Register ist verändert, alle anderen Registerinhalte sind hingegen geschützt.

DD READ ID 009BH

Die XBIOS-Funktion DD READ ID liest die Sektoridentifikation des ersten gefundenen Sektors.

Einsprungbedingungen:

Im C-Register ist die Laufwerksnummer enthalten
D beinhaltet die Nummer der logischen Spur
In IX übergibt man die Adresse des XDPB im Common-Speicher zwischen 0C000H und 0FFFFH.

Aussprungbedingungen:

Konnte die Sektor-ID ermittelt werden, wird sie im Akkumulator des Prozessors an das aufrufende Programm übergeben. Außerdem ist in diesem Fall das Carry-Flag auf den Zustand "wahr" gesetzt.

Ist das Carry-Flag des Z80 aber falsch, trat ein Fehler auf, der anhand des Fehlercodes im Akkumulator entschlüsselt werden kann:

A=0 Laufwerk nicht bereit
A=2 Nicht ansteuerbar
A=3 Datenfehler
A=4 Keine Daten
A=5 Fehlende Adreßmarkierung

In beiden Fällen zeigt das HL-Register auf den Resultatspuffer, der vom Floppy-Controller angelegt wurde. Er befindet sich im Common-Memory und ist folgendermaßen aufgebaut:

Byte 0: Zahl der Bytes im Puffer
Byte 1: Erstes Resultatbyte
Byte 2: Zweites Resultatbyte
.
.
.

DD INIT DPB 009EH

Die XBIOS-Routine DD INIT DPB initialisiert einen Disk-Parameter-Block (DPB), der den Konventionen von CP/M Plus entspricht.

Einsprungbedingungen:

HL zeigt auf eine Tabelle mit Spezifikationen (oberhalb 0C000H im Common-Memory)
IX zeigt auf den zu erzeugenden DPB (oberhalb 0C000H)

Die Spezifikationstabelle sollte folgendermaßen aufgebaut sein:

Byte 0 Diskettentyp
Byte 1 Zahl der Seiten
Byte 2 Zahl der Spuren pro Seite
Byte 3 Zahl der Sektoren pro Spur
Byte 4 Logarithmus der Sektorgröße
Byte 5 Zahl der Systemspuren
Byte 6 Logarithmus der Blockgröße
Byte 7 Zahl der Blocks im Inhaltsverzeichnis
Byte 8 Länge der Gaps beim Lesen und Schreiben
Byte 9 Länge der Gaps beim Formatieren

Aussprungbedingungen:

Das Carry-Flag zeigt den korrekten Ablauf an. Die Register BC, DE und HL sind verändert.

DD INIT XDPB 00A1H

Im Gegensatz zu DD INIT DPB wird hier ein kompletter Extended DPB (XDPB) initialisiert. Ansonsten gelten aber alle Angaben von DD INIT DPB.

DD ON MOTOR 00A4H

Die XBIOS-Funktion DD ON MOTOR kann dazu benutzt werden, den Motor des Diskettenlaufwerks einzuschalten. Sie wartet dann noch einen Augenblick und kehrt zum aufrufenden Programm zurück.

Einsprungbedingungen:

Keine

Aussprungbedingungen:

Alle Register und Flags sind geschützt.

DD TIMEOFF MOTOR 00A7H

Die XBIOS-Funktion DD TIMEOFF MOTOR startet eine Interruptroutine, die nach kurzer Zeitverzögerung den Motor des Diskettenlaufwerks ausschaltet. DD OFF MOTOR wartet aber selbst diesen Augenblick nicht ab.

Einsprungbedingungen:

Keine

Aussprungbedingungen:

Alle Register und Flags sind geschützt.

DD OFF MOTOR 00AAH

Im Gegensatz zu DD TIMEOFF MOTOR schaltet die XBIOS-Routine DD OFF MOTOR den Motor des Laufwerks sofort ab und wartet nicht auf eine Timeout-Routine.

Einsprungbedingungen:

Keine

Aussprungbedingungen:

AF, BC, DE und HL sind verändert.

DD CONTROLLER READ 00ADH
DD CONTROLLER WRITE 00B0H

Diese beiden Routinen erlauben die direkte Programmierung des Floppy-Controllers 765A. DD CONTROLLER READ übernimmt die Lesefunktionen, DD CONTROLLER WRITE die Schreibfunktionen. Diese Funktionen werden hier nicht weiter erläutert, da sie genaue Kenntnisse über die Funktionsweise des Floppy-Controllers voraussetzen und auch kaum jemals benötigt werden.

DD SEEK TRACK 00B3H

Diese Routine kann dazu benutzt werden, eine Spur auf der Diskette anzusteuern. Sie ist aber kaum von Nutzen, da dies die Funktionen wie DD READ SECTOR und DD WRITE SECTOR selbst tun.

RS INIT 00B6H

Initialisiert den Kanal A der seriellen Schnittstelle auf vorgebbare Werte.

Einsprungbedingungen:

Der Akku zeigt an, ob Handshaking gewünscht ist.
D gibt die Zahl der Stopbits an (0=1, 1=1,5, 2=2)
E bezeichnet die Parität (0 = keine, 1 = ungerade, 2 = gerade)
H ist die Zahl der zu empfangenden Bits (5, 6, 7, 8)
L ist die Zahl der zu sendenden Bits (5, 6, 7, 8)

Aussprungbedingungen:

AF, BC, DE, und HL sind verändert.

RS SET BAUD 00B9H

RS SET BAUD legt die Baudrate fest, mit der über den Kanal A der seriellen Schnittstelle Daten übertragen werden.

Einsprungbedingungen:

H enthält die codierte Baudrate für den Empfang
L enthält die codierte Baudrate zum Senden

Die Codierung erfolgt nach dem folgenden Schema:

50, 75, 110, 134.5, 150, 300, 600, 1200, 1800, 2400, 3600, 4800, 7200, 9600, 19200

Diese Baudraten werden der Reihe nach mit den Werten 1 bis 15 bezeichnet. So steht 1 für 50 Baud, 13 für 7200 Baud.

Aussprungbedingungen:

AF, BC, DE und HL sind verändert.

RS GET PARAMS 00BCH

Die XBIOS-Funktion RS GET PARAMS macht die gerade aktuellen Parameter der seriellen Schnittstelle für Anwenderprogramme verfügbar.

Einsprungbedingungen:

Keine

Aussprungbedingungen:

A enthält den Handshake-Modus (0=aus, 255=an)
B enthält die Baudrate für den Empfang
C enthält die Baudrate beim Senden
D enthält die codierte Zahl der Stopbits
E enthält die Parität
H enthält die Zahl der Datenbits beim Empfang
L enthält die Zahl der Datenbits beim Senden

TE GET WINDOW & CURSOR 00BFH

Diese XBIOS-Funktion ermittelt die aktuelle Cursorposition auf dem Bildschirm und stellt fest, wie groß das gerade verwendete Bildschirmfenster ist.

Einsprungbedingungen:

Keine

Aussprungbedingungen:

B enthält die obere Grenze des Windows
C enthält die linke Spalte des aktuellen Fensters
D enthält die Höhe des Fensters -1
E enthält die Breite des Fensters -1
H enthält die Cursorzeile
L enthält die Cursorspalte

Die Zählung beginnt bei den Koordinaten in der linken oberen Bildschirmecke mit (0,0).

TE RESET 00C2H

Will man den Bildschirmtreiber zurücksetzen, kann man TE RESET aufrufen. Diese XBIOS-Funktion löscht den Bildschirm und positioniert den Cursor in der linken oberen Ecke des Screens. Außerdem wird die Cursordarstellung eingeschaltet.

Einsprungbedingungen:

Keine

Aussprungbedingungen:

AF, BC, DE und HL sind verändert.

TE STLINE ASK 00C5H

Ob die Statuszeile auf dem Bildschirm eingeschaltet ist, kann man mit TE STLINE ASK ermitteln. Über die Statuszeile gibt das Betriebssystem Meldungen wie "Laufwerk ist A": oder "Laufwerk ist B:" an den Benutzer weiter.

Einsprungbedingungen:

Keine

Aussprungbedingungen:

Das Zero-Flag des Z80-Prozessors ist auf den Zustand "falsch" gesetzt, wenn die Statuszeile dargestellt wird. Sonst ist das Zero-Flag wahr.

In beiden Fällen ist der Inhalt des Akkumulators verändert, die anderen Registerwerte sind geschützt.

TE STLINE ON/OFF 00C8H

Wollen Sie die Statuszeile ein- oder ausschalten, können Sie die XBIOS-Funktion TE STLINE ON/OFF hierzu heranziehen.

Einsprungbedingungen:

A=0 Statuszeile wird ausgeschaltet
A=255 Statuszeile wird eingeschaltet

Aussprungbedingungen:

Die Flags sowie BC, DE und HL sind verändert.

Ein einfacherer Weg als über diese XBIOS-Funktion ist es, die Bildschirmsteuerzeichen ESC 0 (dezimal 27 48) beziehungsweise ESC 1 (dezimal 27 49) über die BDOS- oder BIOS-Funktionen auszugeben.

TE SET INK 00CBH

Mit TE SET INK können Sie bei farbtüchtigen Schneider-Computern die Stiftfarbe festlegen. Da der Joyce lediglich über einen Grünmonitor verfügt und auch keine verschieden hellen Grüntöne darstellen kann, ist diese Funktion eigentlich sinnlos. Sie kann lediglich eine inverse Darstellung erreichen.

TE SET BORDER 00CEH

Diese XBIOS-Funktion legt die Rahmenfarbe des Bildschirms fest. Die Werte für blinkende Farben werden in den Registern B und C übergeben. Da Joyce eine monochrome Ausgabe hat, ist diese Funktion sinnlos.

TE SET SPEED 00D1H

Die XBIOS-Routine TE SET SPEED bestimmt, mit welcher Geschwindigkeit sich blinkende Farben abwechseln. Das Register H enthält die Zeitperiode für die erste Farbe, das L-Register die Periode für die zweite. Der Wertebereich liegt zwischen 0 und 255, wobei 0 als 256 interpretiert wird. Eine Einheit entspricht dabei einer 1/50 Sekunde.

KM SET EXPAND 00D4H

Mit KM SET EXPAND können Sie Ihre Tastatur umbelegen. Auf die Funktionstasten lassen sich nämlich Zeichenketten legen. Die Funktionstasten besitzen die Codes 128 bis 158.

Einsprungbedingungen:

B enthält die Nummer der Funktionstaste
C enthält die Stringlänge
HL zeigt auf den String im Common-Memory oberhalb 0C000H

Aussprungbedingungen:

Das Carry-Flag wird abhängig vom positiven Ausgang der Tastenumbelegung gesetzt. Der Akku sowie BC, DE und HL sind verändert.

KM SET KEY 00D7H

Die Funktion KM SET KEY weist einer beliebigen Taste auf der Tastatur ein neues Zeichen zu. Dies kann auch ein Funktionstasten-Code sein.

Einsprungbedingungen:
B enthält die neue Tastenübersetzung
C enthält die Nummer der Taste (siehe Joyce-Handbuch)
D ist bitweise codiert und zeigt an, welche Tastaturebenen belegt werden sollen:

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

Aussprungbedingungen:

Der Akku, die Flags und das HL-Register sind verändert.

KM KEY GET 00DAH

Hiermit weist man den Computer an, die Tastatur abzufragen und festzustellen, ob eine Taste gedrückt wurde. Entgegen der normalen Tastaturabfrage werden die Übersetzungstabellen nicht beachtet. Es werden also genau einzelne Tastenkappen untersucht.

Einsprungbedingungen:

Keine

Aussprungbedingungen:

Wurde ein Tastendruck entdeckt, ist das Carry-Flag des Z80-Prozessors wahr und das C-Register enthält die Tastennummer.

Stets wird im B-Register der Zustand der Umschalttasten angegeben:

Bit 0 Nicht definiert
Bit 1 EXTRA
Bit 2 CAPS LOCK
Bit 3 Tastenwiederholung
Bit 4 NUM LOCK
Bit 5 SHIFT
Bit 6 SHIFT LOCK
Bit 7 ALT

KM KEY PUT 00DDH

Die XBIOS-Funktion KM KEY PUT simuliert einen Tastendruck. Ein beliebiges Zeichen wird in den Tastaturpuffer (Größe 10 Zeichen) eingefügt. Beim nächsten KM KEY GET wird es dann zurückgemeldet. So kann ein Programm unabhängig von den Aktionen des Benutzers beispielsweise die NUM LOCK- oder CAPS LOCK-Tasten setzen - SHIFT LOCK jedoch nicht, da es hardwaregesteuert ist.

Einsprungbedingungen:

C enthält den Tastencode
B enthält den Zustand der Umschalttasten (siehe KM KEY GET)

Aussprungbedingungen:

Alle Register sind geschützt.

KM SET SPEED 00E0H

Diese Funktion legt die Geschwindigkeit fest, mit der die Tastenwiederholung einsetzt und mit der sie fortgesetzt wird.

Einsprungbedingungen:

H enthält die Verzögerung für das Einsetzen
L enthält die Verzögerung bei der Wiederholung

Aussprungbedingungen:

Der Akkumulator und die Flags sind verändert.

MAC GET VERSION 00E3H

Hiermit können Sie erfahren, auf welchem Computer Ihr Programm läuft, welche BIOS-Version verwendet wird, und welches ROM aktiv ist.

Einsprungbedingungen:

Keine

Aussprungbedingungen:

A=0 CPC-6128
A=1 Joyce
B Versionsnummer des BIOS (größere Änderung)
C Versionsnummer des BIOS (kleinere Änderung)
HL bei Joyce: undefiniert
beim CPC-6128: ROM-Nummer

MAC GET INFO 00E6H

Holt Systeminformation und stellt fest, welche Hardware-Erweiterungen angeschlossen sind.

Einsprungbedingungen:

Keine

Aussprungbedingungen:

A enthält die Zahl der Diskettenlaufwerke:

A=0: ein Laufwerk
A=255: zwei Laufwerke

B enthält die Zahl der verfügbaren Speicherblocks

C legt fest, ob die serielle Schnittstelle angeschlossen ist:

C=0: nicht vorhanden
C=255: angeschlossen

HL zeigt auf die Zeigertabelle der Diskettenpuffer

Diese Zeigertabelle ist wie folgt aufgebaut:

Zeiger auf Puffer 0
Zeiger auf Puffer 1
.
.
.

Diese Puffer wiederum besitzen das folgende Format:

Byte 0: Speicherbank
Bytes 1 und 2: Startadresse
Bytes 3 und 4: Größe in Bytes

SCR RUN ROUTINE 00E9H

Der Bildschirmspeicher des Schneider-Joyce ist unzugänglich in die Systembank und eine andere Speicherbank ausgelagert worden. Damit man

aber dennoch auf den Bildschirmspeicher zugreifen kann, besitzt Joyce die XBIOS-Routine SCR RUN ROUTINE. Sie blendet alle Speicherblocks des Bildschirms in den Hauptspeicher ein und startet in dieser Umgebung eine Maschinenroutine, die im Common-Memory oberhalb 0C000H liegen muß.

Einsprungbedingungen:

BC zeigt auf die Routine, die aufgerufen werden soll.

Aussprungbedingungen:

BC ist verändert, die anderen Registerwerte geschützt.

Der Bildschirmspeicher besteht aus drei Bereichen: der Matrixtabelle für die Bildschirmzeichen, dem eigentlichen Bildschirmspeicher und einer Zeigertabelle.

Die Matrixtabelle besitzt eine Länge von zwei KByte und startet bei 0B800H. Für jedes Zeichen sind acht Bytes reserviert, die ein Zeichen Pixelzeile für Pixelzeile definieren. Werden Werte in diesen Speicher geschrieben, erhalten die Bildschirmzeichen ein neues Aussehen.

Der Bildschirmspeicher besitzt keine feste Adresse. Er erstreckt sich über einen weiten Bereich in der Systembank. Das ist zwar eine enorme Speicherplatzverschwendung, hat aber den Vorteil, daß das Scrolling stark beschleunigt werden kann. Denn der Bildschirmspeicher ist vollständig über Pointer gesteuert. Das heißt, daß eine Zeigertabelle existiert und diese für jede Pixelzeile einen Eintrag enthält. Dieser Eintrag stellt einen Zeiger auf die tatsächliche Speicheradresse dar. Beim Scrolling muß nun nicht mehr der gesamte Bildschirmspeicher umkopiert werden. Vielmehr reicht es aus, lediglich die Einträge in der Zeigertabelle zu verändern.

Diese Zeigertabelle kann man auch für interessante Effekte "mißbrauchen". So lassen sich mit minimalem Programmieraufwand doppelt- oder dreifachhohe Zeichen darstellen. Man muß nur mehrere Einträge in der Zeigertabelle auf denselben Speicherbereich zeigen lassen. Auch das Umschalten zwischen verschiedenen Bildschirmdarstellungen ist blitzschnell möglich.

Allerdings bedeutet die Zeigersteuerung auch zusätzlichen Programmieraufwand. Denn man kann nun nicht mehr von einer fixen Adresse für eine Rasterzeile ausgehen, sondern jede Adresse muß erst bei der Laufzeit des Programms bestimmt werden.

Die Zeigertabelle beginnt an der Adresse 0B600H und ist 512 Bytes groß. Sie besitzt folgenden Aufbau:

Bytes 0 und 1: Zeiger auf Pixelzeile 0
Bytes 1 und 2: Zeiger auf Pixelzeile 1
.
.
.

Der Aufruf von SCR RUN ROUTINE ist nicht ganz einfach. Sie benötigen unbedingt Maschinensprache dafür. Eine exemplarische Lösung zeigt das folgende Assemblerprogramm:

```
*****************************************************************
*                                                               *
*              Auslesen des Bildschirmspeichers                 *
*                                                               *
*****************************************************************

; Übernimmt Adresse in 28
; Übergibt Wert in 30

        ORG   0C400H    ; Beliebig, hier 0C400H

BOOT    EQU   0000H     ; Warmstartvektor

HOLDSP. EQU   26                ; Speicher für den Stack
VALUE   EQU   28                ; Speicher für gelesenen Wert
ADDR    EQU   30                ; Vektor für Sprung
USERF   EQU   0D100H    ; Vektor für USERF
DESTIN  EQU   0D200H    ; Zielroutine im Screen-Environment
STACK   EQU   0D000H    ; Neuer Stack im Common-Memory

GETVDU  LXI   H,0       ; Stackpointer holen
        DAD   SP
        SHLD  HOLDSP    ; und sichern
        LXI   SP,STACK  ; Neuen Stack laden

        LHLD  BOOT+1    ; BIOS-Einsprung holen
        LXI   D,87      ; 87 hinzuzählen
        DAD   D         ; Start von USERF ermitteln
        SHLD  USERF+1   ; Und für später speichern

        MVI   A,0C3H    ; Z80-Opcode für JMP
        STA   USERF     ; Vor dem Vektor ablegen

        MVI   A,7EH     ; Z80/8080-Code "MOV A,M"
        STA   DESTIN    ; In der Zielroutine speichern
        MVI   A,0C9H    ; Z80/8080-Code "RET"
        STA   DESTIN+1  ; In der Zielroutine speichern
```

```
        LHLD ADDR      ; Zu lesende Adresse aus Basic holen
        LXI  B,DESTIN  ; Adresse der Zielroutine
        CALL USERF     ; Sprung ins XBIOS
        DW   00E9H     ; Mit dem Code für die SCR-Funktion
        STA  VALUE     ; Gelesenen Wert für Basic speichern

        LHLD HOLDSP    ; Alten Stackpointer zurückholen
        SPHL           ; Wieder laden
        RET            ; Rücksprung zum Hauptprogramm

        END

; ****************************************************
```

Diese Routine ist so aufgebaut, daß sie in der Speicheradresse 28 die Adresse übernimmt, die aus dem Bildschirmspeicher ausgelassen werden soll, und in 30 den gelesenen Wert zurückgibt. Über PEEK und POKE kann man daher mit dem Maschinenprogramm von Mallard-Basic aus kommunizieren.

Nur wie bekommt man den Assembler-Quellcode in DATA-Zeilen? Das Programm HEX2DATA zeigt den Weg: Man assembliert mit MAC.COM den Quellcode:

A>MAC SCREEN

Der Assembler erzeugt eine .HEX-Datei. Diese wird mit HEX2DATA behandelt. Das Programm ist in Basic geschrieben und wird so gestartet:

A>BASIC HEX2DATA

Das Programmlisting von HEX2DATA.BAS:

```
100  ' * * * * * * * * * * * * * * * * * *
110 ' *                     *
120 ' *     HEX2DATA        *
130 ' *                     *
140 ' *******************
150 '
160 DIM b$(50),z(50),z$(50)
170 zeile=10000:zeile$=STR$(zeile)+" ":f=0
180 cls$=CHR$(27)+"E"+CHR$(27)+"H":PRINT cls$
190 INPUT "Name der Eingabedatei (ohne .HEX) ";infile$
200 infile$=infile$+".HEX"
210 PRINT:INPUT "Name der Ausgabedatei (ohne .BAS) ";outfile$
220 outfile$=outfile$+".BAS"
230 OPEN "I",#1,infile$
```



```
240 OPEN "O",#2,outfile$
250 LINE INPUT #1,a$
260 IF MID$(a$,2,2)="00" THEN CLOSE:PRINT "Endadresse:    ";addr$:PRINT:END
270 addr$=MID$(A$,4,4):IF f=0 THEN PRINT:PRINT "Startadresse: ";addr$:f=1
280 a$=RIGHT$(a$,LEN(a$)-9)
290 a$=LEFT$(a$,LEN(a$)-2)
300 FOR i=1 TO (LEN(a$)/2)
310     b$(i)=MID$(a$,2*i-1,2)
320     z(i)=VAL("&H"+b$(i))
330     z$(i)=STR$(z(i))
340 NEXT i
350 PRINT #2,zeile$;"DATA ";RIGHT$(z$(1),LEN(z$(1))-1);
360 FOR i=2 TO (LEN(a$)/2)
370     PRINT #2,",";RIGHT$(z$(i),LEN(z$(i))-1);
380 NEXT i
390 PRINT #2
400 zeile=zeile+10
410 zeile$=STR$(zeile)+" "
420 GOTO 250
```

HEX2DATA funktioniert einwandfrei. Man muß nur aufpassen, daß unbedingt alle Zeilen mit DS (Define Space) am Ende des Quellcodes stehen. Statt

```
START:  JMP  CONT
SPACE:  DS   30
STACK:  DS   20
CONT:   LXI  H,3FACH
        .
        .
        .
```

schreiben Sie also:

```
START:  JMP  CONT
CONT:   LXI  H,3FACH
        .
        .
        .
SPACE:  DS   20
STACK:  DS   20
        END
```

Nebenbei wird Ihr Programm so auch noch viel übersichtlicher.

4 Mallard-Basic - Unterschiede zu den CPCs

Mallard-Basic ist wirklich eine professionelle Implementation der Programmiersprache Basic. Doch was nützt die beste Programmiersprache, wenn es keine Programme dafür gibt?

Das vorliegende Buch soll auch Programmierer anregen, die Software-Lücke für Mallard-Basic zu schließen oder zumindest - nachdem das nur schwer möglich ist - zu verkleinern.

Im Vergleich zu der Vielzahl von Programmen, die es beispielsweise für die CPC-Reihe der Firma Schneider in Zeitschriften und Büchern gibt, ist das Angebot an Basic-Programmen für Mallard-Basic mehr als mager. Deshalb erfahren Sie hier, wie Sie viele Programme aus der Basic-Version des Schneider-CPC ins Mallard-Basic übertragen können. Da die beiden Basic-Interpreter von derselben Firma, nämlich Locomotive-Software, entwickelt wurden, ist es oft gar nicht so schwer, Programme zu übersetzen. In anderen Fällen muß man relativ komplizierte Ersatzroutinen in Basic oder gar Maschinensprache schreiben. Manche Befehle lassen sich gar nicht übertragen. Deshalb wollen wir uns zuerst einige Problemfälle ansehen:

4.1 Bildschirmdarstellung

Der Bildschirm des Joyce kann mehr Zeilen und Spalten auf einmal darstellen als der Monitor des Schneider-CPC. Das ist zwar eine angenehme Eigenschaft, stört aber bei der Übernahme von Programmen, die auf die exakten Dimensionen des Bildschirms angewiesen sind. Aber Joyce läßt sich in einen Modus schalten, der 24 Zeilen und 80 Zeichen pro Zeile bietet. Das schafft das CP/M-Programm SET24X80.COM, das sich auf einer Ihrer Systemdisketten befindet. Sie sollten es vor dem Laden von Mallard-Basic starten:

```
A>SET24X80 ON
A>BASIC
```


Alternativ können Sie auch in die erste Befehlszeile des Programms eine Bildschirm-Steuersequenz einfügen:

```
10 PRINT CHR$(27);"X";CHR$(32);CHR$(32);
20 PRINT CHR$ (56);CHR$(111);
```

4.2 Grafikdarstellung

Mallard-Basic besitzt keinerlei Befehle zur Ansteuerung der - technisch bei Joyce möglichen - hochauflösenden Grafik. Deshalb sind Programm des Schneider-CPC nicht direkt übertragbar, wenn sie Befehle wie MOVE, DRAW und PLOT enthalten.

Aber Joyce wird mit dem Grafikpaket GSX geliefert. Dieses stellt Programmen, sobald installiert, den BDOS-Aufruf 115 zur Verfügung, der Grafikausgaben möglich macht. Leider ist GSX nicht gerade schnell. Wenn Sie gut programmieren können, sollten Sie sich eigene Maschinenroutinen für die Grafikausgabe schreiben.

4.3 Ton- und Musikwiedergabe

Der Schneider-CPC besitzt einen ziemlich leistungsfähigen Tongenerator. Dieser kann bis zu drei Stimmen gleichzeitig über den Lautsprecher ausgeben. Schließt man eine Stereoanlage an den Computer an, kann Software bei geschickter Programmierung sogar Stereo-Effekte simulieren. Im Gegensatz dazu ist der einzige Ton, den man dem Joyce-PCW entlocken kann, ein simpler Piepser, der sich nicht einmal in der Tonhöhe variieren läßt. Deshalb müssen alle Soundeffekte bis auf PRINT CHR$ (7) aus CPC-Programmen entfernt werden.

4.4 Maschinensprache

Beide Computer basieren auf dem Z80A-Prozessor des amerikanischen Chip-Herstellers Zilog. Aber abgesehen von dieser Gemeinsamkeit unterscheiden sie sich auf Maschinensprache-Ebene, wie es zwei Computer nur tun können. Deshalb sind Maschinenroutinen, die auf Programmteile des Betriebssystems zugreifen - und das sind fast alle -, nicht übertragbar.

4.5 Interrupts

Das Locomotive-Basic des Schneider-CPC ist interruptfähig. Das heißt, daß ein Anwender ein Unterprogramm in Basic erstellen kann, das vom Basic-Interpreter in regelmäßigen Zeitintervallen oder auch ein einziges Mal nach Ablauf einer bestimmten Zeit aufgerufen wird. Die beiden entsprechenden Befehle heißen EVERY und AFTER:

```
10 EVERY 50 GOSUB 50
20 PRINT "HALLO"
30 GOTO 20
40 ' Basic-Interrupt:
50 PRINT CHR$(7)
60 RETURN
```

In diesem kleinen Programm wird die Subroutine ab der Zeile 50 bei jedem fünfzigsten Interruptaufruf ausgeführt. Da der Interrupt alle 0,02 Sekunden auftritt, ist das genau einmal in der Sekunde. Das Hauptprogramm druckt andauernd den Text "HALLO" auf den Bildschirm, während die Interruptroutine Piepstöne von sich gibt.

Mit AFTER läßt sich zum Beispiel ein Wecker programmieren:

```
10 AFTER 20000 GOSUB 50
20 PRINT "HALLO"
30 GOTO 20
40 'AFTER-Routine:
50 PRINT:PRINT
60 PRINT "Du mußt noch einkaufen! Deshalb ist für"
70 PRINT "heute Schluß mit der Computerei."
80 END
```

Weder EVERY noch AFTER können auf irgendeine Weise im Mallard-Basic nachgeahmt werden.

4.6 Diskettenformat

Es ist schon etwas seltsam: Legt man in ein Diskettenlaufwerk beim Schneider-CPC eine Diskette ein, die von Joyce formatiert wurde, scheint der CPC sie lesen zu können. Zumindest das Inhaltsverzeichnis kann man mit DIR ansehen. Doch versucht man, zum Beispiel eine Textdatei mit TYPE

aufzulisten, kommt nur Unsinn heraus. Kurz gesagt: das Format kann der CPC nicht verarbeiten, nur merken tut er es nicht!

Andersherum gibt es aber keinerlei Probleme. Der Schneider-Joyce kann CPC-Disketten verarbeiten.

Mallard-Basic kann allerdings keine Programme im codierten Format des Locomotive-Basic einlesen. Deshalb sollten Sie solche Programme im Schneider-Basic als ASCII-Textdateien speichern:

```
SAVE "PROG.BAS",A
```

Es ist nicht anzunehmen, daß das Programm sofort ohne Änderungen verwendbar ist. Deshalb sollten Sie es noch nicht von Basic aus laden, sondern mit einem Texteditor erst einmal in den Sprachdialekt von Mallard-Basic übersetzen. Als Texteditor eignet sich RPED von Ihrer CP/M-Systemdiskette, sofern das Programm nicht zu lang ist. Ansonsten können Sie diese Anpassung mit jedem beliebigen Texteditor durchführen, der normale ASCII-Textdateien erstellt. Locoscript ist dafür also nicht geeignet, wohl aber beispielsweise WordStar im Non-Document-Mode.

4.7 Vergleichende Befehlsliste

Abgesehen von diesen Gebieten, auf denen die Umsetzung von Fremdprogrammen Schwierigkeiten bereitet, ist es sonst oft nicht besonders kompliziert, ein für den Schneider-CPC konzipiertes Programm auf dem Joyce zum Laufen zu bringen. Was fehlt, sind Informationen zu den einzelnen Befehlen des Locomotive-Basic.

Der Einfachheit halber sind im folgenden die Anleitungen zum Umschreiben der Befehle alphabetisch geordnet. Es sind auch diejenigen Befehle aufgeführt, die der Joyce überhaupt nicht simulieren kann, damit Sie wenigstens wissen, welche Aufgaben sie erfüllen.

4.8 Ersatzroutinen

Manche Befehle sind in Mallard-Basic nicht vorhanden, lassen sich aber über Maschinenroutinen simulieren. Denn Joyce besitzt eine XBIOS-Sprungleiste, die bereits Thema des vorangehenden Kapitels war. Wenn Sie

die Simulationen für diese Befehle benutzen wollen, müssen Sie Ihrem Programm die folgenden Basic-Zeilen voranstellen:

```
100 ' ************************************************
110 ' *                                              *
120 ' *  XBIOS - Aufruf von Systemroutinen aus Basic  *
130 ' *                                              *
140 ' ************************************************
150 '
160 ' Maschinenroutine aus DATAs einlesen *************
170 MEMORY &HF000-1
180 FOR i=&HF000 TO &HF04A
190     READ byte
200     POKE i,byte
210 NEXT i
220 ' Variablen zeigen auf Speicherstellen ************
230 xbios=&HF04E
240 reg.af=&HF050
250 reg.bc=&HF052
260 reg.de=&HF054
270 reg.hl=&HF056
280 reg.ix=&HF058
290 reg.iy=&HF05A
300 DEF FNmsb(x)=INT(x/256)
310 DEF FNlsb(x)=x-INT(x/256)*256
320 ' Daten fuer XBIOS *********************************
330 DATA 42,1,0,17,87,0,25,34,76,240,62,195,50,75,240
340 DATA 42,78,240,34,48,240,42,80,240,229,241,237,75
350 DATA 82,240,237,91,84,240,42,86,240,221,42,88,240
360 DATA 253,42,90,240,205,75,240,191,0,34,86,240,245
370 DATA 225,34,80,240,237,67,82,240,237,83,84,240
380 DATA 221,34,88,240,253,34,90,240,201
390 ' ************************************************
```

Sie dürfen danach das Speicherende mit MEMORY nicht wieder höher setzen. Außerdem sollten Sie die definierten Variablen- und Funktionsnamen nicht anderweitig verwenden.

4.9 Alphabetische Befehlsliste

ABS

Keine Unterschiede

AFTER

Syntax: AFTER <Zeitperiode>[,<Timer>] GOSUB <Zeile>

Nach der angegebenen Zeitperiode (Intervall 0,02 Sekunden) wird die Routine mit der Zeilennummer <Zeile> aufgerufen. Die Nummer des Timers steht in <Timer> und darf einen Wert zwischen 0 und 3 annehmen. Wird sie weggelassen, verwendet das Basic den Timer 0.

Dieser Befehl läßt sich in Mallard-Basic nicht simulieren.

AND

Keine Unterschiede

ASC

Keine Unterschiede

ATN

Syntax: ATN(<Wert>)

Die ATN-Funktion berechnet den Arcus-Tangens (arc tan) des angegebenen numerischen Wertes. Mit den Befehlen DEG und RAD läßt sich festlegen, ob das Ergebnis in Winkelgraden oder im Bogenmaß angezeigt werden soll.

Die Funktion ATN wird vom Mallard-Basic ebenfalls verstanden. Allerdings fehlen die Kommandos DEG und RAD. Der Joyce befindet sich ständig im RAD-Modus. Zur Umrechnung eines RAD-Ergebnisses in den DEG-Modus kann man dieses durch (PI/180)=1.745329E-02 teilen:

```
pi=3.141593
deg=pi/180
PRINT ATN(6)/deg
```

AUTO

Keine Unterschiede

BIN$

Syntax: BIN$(<Zahl>,[<Ausgabelänge>])

BIN$ berechnet aus der Zahl ihr binäres Äquivalent als Zeichenkette. In der Form BIN$(x) nimmt der Binärstring die niedrigstmögliche Länge der Zeichenkette an:

```
PRINT BIN$(5)        101
```

Zusätzlich kann man einen Wert für die Ausgabelänge angeben. Dies bewirkt, daß der Ausgabestring mit führenden Nullen erweitert wird, bis die vorgegebene Länge erreicht ist:

```
PRINT BIN$(5,8)      00000101
```

Aus unerklärlichen Gründen fehlt diese wichtige Funktion im Mallard-Basic. Sie läßt sich aber mit etwas Basic-Akrobatik sogar als benutzerdefinierte Funktion nachbilden:

```
100 ' Ersatz fuer die BIN$-Funktion
110 ' ----------------------------
120 bi$="0000.0001.0010.0011.0100.0101.0110.0111.1000
    .1001.1010.1011.1100.1101.1110.1111."
130 DEF FNplus(a)=a-65536!*(a<0)
140 DEF FNbin1$(a)=MID$(bi$,a*5+1,4)
150 DEF FNbin2$(a)=FNbin1$(INT(a/16))+FNbin1$(a-16*IN
    T(a/16))
160 DEF FNbin3$(a)=FNbin2$(INT(a/256))+FNbin2$(a-INT(
    a/256)*256)
170 DEF FNbin$(a)=FNbin3$(FNplus(a))
```

Die Funktion FNbin$(x) berechnet einen Binärstring für Zahlen im Wertebereich -32768...65535. Sie entspricht damit auch in dieser Hinsicht der Funktion BIN$ auf dem Schneider-CPC.

FNbin$(x) liefert stets einen sechzehnstelligen Binärstring. Wollen Sie ihn von führenden und daher unnötigen Nullen befreien, geht das mit einer FOR-NEXT-Schleife ohne Schwierigkeiten:

```
100 INPUT x
110 x$=FNbin$(x)
120 FOR i=16 TO 2 STEP -1
130 IF LEFT$(x$,1)="0" THEN x$=RIGHT$(x$,LEN(x$)-1):NEXT i
140 PRINT x$
```

BORDER

Syntax: BORDER <Rahmenfarbe> [,Rahmenfarbe 2]

Dieser Befehl legt auf dem Schneider-CPC die Rahmenfarbe des Bildschirms fest. Werden zwei Farben angegeben, blinkt der Rahmen abwechselnd in diesen Farben.

Der Joyce ist bekanntlich nur mit einem Grünmonitor ausgestattet. Er kann keine Farben oder Farbschattierungen darstellen. Deswegen läßt sich der Befehl BORDER in der üblichen Form nicht simulieren.

Allerdings kann Joyce den Bildschirmrand hell oder dunkel darstellen. Dazu dient die XBIOS-Routine TE SET BORDER, die in der Systembank an der Adresse 00CEH einen Sprungvektor besitzt. Sie benötigen das Programm XBIOS.BAS, das am Anfang dieses Kapitels abgedruckt ist. An diesen fügen Sie die folgenden Zeilen an:

```
500 POKE xbios,&HCE:POKE xbios+1,0
510 POKE reg.bc,<erste Farbe>
520 POKE reg.bc+1, <zweite Farbe>
530 adr=&HF000:CALL adr
```

Für <erste> und <zweite Farbe> setzen Sie bitcodierte Werte ein:

Bit 7: Invertierte Darstellung mit schwarzen Zeichen auf hellem Untergrund.

Bit 6: Randfarbe bedeckt den gesamten Bildschirm.

Sind die Werte für das B- und das C-Register unterschiedlich, blinkt der Bildschirmrand. Sonst bleibt er in einer Farbe ruhig stehen.

CALL

Syntax: CALL Adresse [,Liste von Parametern]

Der CALL-Befehl hat die Aufgabe, eine Maschinenroutine an der gewünschten Adresse aufzurufen. Dabei können eine Reihe von Zahlen als Parameter übergeben werden.

Der Aufruf von Maschinenroutinen ist in Mallard-Basic ebenso mit CALL möglich; allerdings darf man die Adresse nicht direkt als Zahlenwert angeben. Stattdessen muß sie in eine Variable geladen werden. Anstelle von CALL 40000 heißt es also:

```
adr=40000
CALL adr
```

Es ist aber nahezu unmöglich, Maschinenroutinen vom Schneider-CPC ohne größere Änderungen zu übernehmen. Obwohl beide Computer denselben Mikroprozessor, einen Z80A von Zilog, besitzen, sind sie vom Betriebssystem her so unterschiedlich aufgebaut, daß ein Austausch von Maschinenprogrammen ohne weitere Anpassung nahezu ausgeschlossen erscheint.

CAT

Syntax: CAT

CAT listet auf dem Bildschirm das Inhaltsverzeichnis der gerade angemeldeten Diskette auf. Die Dateinamen sind alphabetisch sortiert.

In Mallard-Basic gibt es zur Ausgabe des Directory zwei gleichwertige Befehle. Sie heißen DIR und FILES. Im Gegensatz zu CAT werden die Namen der Dateien aber nur unsortiert aufgeführt.

CHAIN

Syntax: CHAIN <Dateiname> [,<Zeilennummer>]

Dieser Befehl ruft von einem Basic-Programm ein anderes Programm auf, das auf der Diskette gespeichert ist. Im Gegensatz zu RUN <Dateiname> bleiben die im alten Programm definierten Variablenwerte erhalten. Wird eine Zeilennummer aufgeführt, beginnt die Abarbeitung des Programms bei dieser Zeile.

Eine Ersatzroutine ist für Mallard-Basic nicht nötig. Denn hier befindet sich CHAIN ebenfalls im Befehlsumfang des Basis-Interpreters. Allerdings löscht CHAIN in seiner Grundform beim Starten des neuen Programms die bereits definierten Variablen. Dem können Sie abhelfen, indem Sie an den CHAIN-Befehl das Schlüsselwort "ALL" mit zwei vorangehenden Kommas anhängen:

```
CHAIN "PROG.BAS",,ALL
```

Auch diejenige Zeile des nachzuladenden Programms, bei der der Basic-Interpreter mit der Abarbeitung beginnen soll, kann angegeben werden:

```
CHAIN "PROG.BAS",5010,ALL
```


CHAIN MERGE

Syntax: CHAIN MERGE <Dateiname> [,Zeilennummer]
[,DELETE <Zeilenbereich>]

Dieser Befehl ähnelt CHAIN. Im Gegensatz zu diesem wird das alte Programm aber nicht gelöscht. Die neuen Programmzeilen werden stattdessen hinzugeladen. Definierte Variablen bleiben erhalten.

Bei Angabe von DELETE löscht Basic den gewünschten Zeilenbereich aus dem entstehenden Gesamtprogramm.

Hier gilt für Mallard-Basic wieder, daß die Variablen nur erhalten werden, wenn Sie das Schlüsselwort "ALL" angeben:

```
CHAIN MERGE "PROG.BAS",,ALL
```

Passen Sie aber auf, daß Ihr Programm nicht unversehens in eine Endlosschleife läuft. Ein kleines Beispiel, wie das unversehens passieren kann:

Programm PROG1.BAS:
```
30 PRINT a
```

Programm PROG2.BAS:
```
10 a=3.14
20 CHAIN MERGE "PROG1.BAS",,ALL
```

Startet man hier PROG2, lädt PROG2 den Programmteil PROG1 nach und startet das ganze Programm erneut. Dann lädt es also wieder PROG1 ... Abhilfe ist nicht besonders schwierig. Man muß im Programm PROG2.BAS nur die CHAIN-MERGE-Anweisung durch Angabe einer Zeilennummer ergänzen:

```
10 a=3.14
20 CHAIN MERGE "PROG1.BAS",30,ALL
```

CHR$

Keine Unterschiede

CINT

Keine Unterschiede

CLEAR

Keine Unterschiede

CLEAR INPUT

Syntax: CLEAR INPUT

Die Schneider-CPCs besitzen einen Tastaturpuffer, in dem die Tasten vermerkt werden, die der Benutzer während des Programmlaufs drückt und die der Computer noch nicht verarbeiten konnte. CLEAR INPUT löscht diesen Tastaturpuffer wieder und ist vor Tastaturabfragen anzuwenden, um "Müll" aus dem Puffer zu entfernen. Ein Beispiel:

```
10 CLEAR INPUT
20 PRINT "Bitte drücken Sie eine Taste!"
30 a$=INKEY$
40 IF a$="" THEN 30
```

Der Tastaturpuffer des Joyce kann maximal ein Zeichen aufnehmen. Als Ersatzlösung für CLEAR INPUT bietet sich eine WHILE-WEND-Konstruktion an:

```
10 WHILE INKEY$<>"":WEND
20 PRINT "Bitte drücken Sie eine Taste!"
30 A$=INKEY$
40 IF A$="" THEN 30
```

CLG

Syntax: CLG [<Farbstift>]

Hiermit löscht der Schneider-CPC den Grafikbildschirm und füllt ihn wahlweise mit der angegebenen Schriftfarbe auf.

Da das Mallard-Basic keine Befehle für die Darstellung von hochauflösender Grafik besitzt, kann CLG nicht durch Ersatzbefehle simuliert werden.

CLOSEIN

Syntax: CLOSEIN

Dieser Befehl weist den Schneider-CPC an, die vorher mit OPENIN zum Lesen geöffnete Diskettendatei zu schließen.

Der CPC kann immer nur je eine Datei zum Lesen und eine zum Schreiben gleichzeitig offenhalten. Deshalb muß dort keine Dateinummer genannt werden. Sie ist ohnehin auf den Datenkanal #9 festgelegt. Das Mallard-Basic ist in dieser Hinsicht leistungsfähiger. Es können mehr als zwei Dateien auf einmal geöffnet sein.

Sie sollten deshalb am besten strikt festlegen, daß die Dateinummer #1 ausschließlich für das Lesen von Dateien reserviert ist, #2 für Schreibvorgänge in Dateien. Dann entspricht dem CLOSEIN-Befehl im Mallard-Basic CLOSE 1 oder CLOSE #1.

CLOSEOUT

Syntax: CLOSEOUT

Nach Eingabe von CLOSEOUT schließt der Computer die geöffnete Ausgabedatei.

Analog zu dem bei der Erläuterung von CLOSEIN Gesagten können Sie CLOSEOUT auf Joyce durch CLOSE #2 ersetzen.

CLS

Syntax: CLS [#<Fenster>]

CLS löscht den Bildschirm. Wird eine Fensternummer angeführt, löscht der Computer ausschließlich einen Ausschnitt des Bildschirms, der durch dieses Fenster festgelegt wird.

Als typisches CP/M-Basic kennt der Mallard-Interpreter keinerlei Befehle zur direkten Bildschirmsteuerung. Sie lassen sich aber mit Escape-Steuersequenzen relativ einfach nachbilden. Statt CLS schreiben Sie also:

```
10 cls$=CHR$(27)+"E"+CHR$(27)+"H"
20 PRINT cls$;
```

CONT

Keine Unterschiede

COPYCHR$

Syntax: COPYCHR$ (#<Fensternummer>)

Die Funktion COPYCHR$ liest beim Schneider-CPC das Zeichen vom Bildschirm, das an der aktuellen Cursorposition steht. Dabei ist die Angabe des zu untersuchenden Bildschirmfensters unerläßlich.

Eine Simulation für COPYCHR$ ist relativ schwierig. Der einzige gangbare Weg ist es, über SCR RUN ROUTINE den Bildschirmspeicher auszulesen und einzelne Bytes mit der Matrixtabelle der Zeichendefinitionen zu vergleichen. Da das in Basic sehr langsam ist, wäre Maschinensprache vorzuziehen.

COS

Syntax: COS(<Zahl>)

Die Funktion COS berechnet den Cosinus des in Klammern folgenden Arguments.

Selbstverständlich verfügt auch Mallard-Basic über die Cosinus-Funktion. Aber es berechnet die Werte stets wie der Schneider-CPC, wenn dieser sich im RAD-Modus befindet. Um den DEG-Modus zu simulieren, sollten Sie das Ergebnis durch (PI/180)=1.745329E-02 teilen:

```
pi=3.141593
deg=pi/180
PRINT COS (0.5)/deg
```

CREAL

Keine Unterschiede

CURSOR

Syntax: CURSOR [Schalter 1] [,Schalter 2]

Der Befehl CURSOR dient zum Ein- und Ausschalten des Cursorsymbols auf dem Bildschirm. Es kann Fälle geben, in denen der Cursor beispielsweise bei INPUT nicht sichtbar sein soll.

CURSOR 0,0 schaltet die Darstellung des Cursors ab, der Befehl CURSOR 1,1 blendet ihn wieder ein.

Der CURSOR-Befehl läßt sich in Mallard-Basic mit Bildschirm-Steuerzeichen nachbilden:

```
10 ' Cursor ausschalten
20 cursoroff$=CHR$(27)+"f"
30 ' Cursor einschalten
40 cursoron$=CHR$(27)+"e"
```

Der Aufruf erfolgt nach der Definition dieser zwei Variablen aus Basic-Programmen über PRINT:

```
PRINT cursoron$;
PRINT cursoroff$;
```

DATA

Keine Unterschiede

DEC$

Keine Unterschiede

DEF FN

Keine Unterschiede

DEFINT
DEFREAL
DEFSTR

Keine Unterschiede

DEG

Syntax: DEG

DEG schaltet bei der Bearbeitung trigonometrischer Funktionen auf dem Schneider-CPC vom Bogenmaß auf das Winkelgrad-Maß um.

Es gibt in Mallard-Basic kein direktes Äquivalent zu DEG. Der Joyce arbeitet ständig im Bogenmaß-Modus. Ergebnisse im Bogenmaß lassen sich aber mit Hilfe einer Division durch (PI/180) ins Winkelgrad-Maß umwandeln:

```
10 pi=3.1415926
20 deg=pi/180
30 PRINT TAN (0.5)/deg
```

DELETE

Keine Unterschiede

DERR

Syntax: DERR

DERR liefert in CPC-Programmen den numerischen Code des letzten aufgetretenen Diskettenfehlers.

Diese Funktion kann man nicht direkt ersetzen. Denn beim Auftreten eines Diskettenfehlers springt der Joyce meistens auf die Systemebene von CP/M zurück und gibt dort eine Fehlermeldung aus. Eventuell ist aber auch eine Abfrage der Systemvariablen ERR sinnvoll.

DI

Syntax: DI

Dieser Befehl sperrt die Abarbeitung von Interruptroutinen im Schneider-Basic. Ein Ersatz ist hier für Joyce nicht möglich, weil das Mallard-Basic gar nicht fähig zur Verarbeitung von Interrupts ist.

DIM

Keine Unterschiede

DRAW
DRAWR

Syntax: DRAW <x-Koordinate>,<y-Koordinate>
DRAW <x>,<y>,<Farbstift>
DRAWR <x-Koordinate>,<y-Koordinate>
DRAWR <x>,<y>,<Farbstift>

DRAW zeichnet eine Linie in der hochauflösenden Grafik des Schneider-CPC mit absoluten Koordinaten. Im Gegensatz dazu arbeitet DRAWR relativ zur alten Cursorposition. Mallard-Basic ist normalerweise nicht grafikfähig.

EDIT

Syntax: EDIT <Zeilennummer>

EDIT ruft eine Programmzeile zur Bearbeitung in den Zeileneditor. Diesen Befehl gibt es auch im Mallard-Basic. Allerdings sind die bei diesem Zeileneditor zulässigen Editiertasten anders als beim CPC-Basic.

EI

Syntax: EI

EI läßt die Abarbeitung der vorher mit DI gesperrten Interruptroutinen im Schneider-Basic wieder zu. Mallard-Basic kennt keine Interruptsteuerung. Deshalb ist dieser Befehl hier sinnlos.

END

Keine Unterschiede

ENT
ENV

Syntax: ENT <Hüllkurven-Nummer> [,<Liste von Hüllkurven>]
ENV <Hüllkurven-Nummer> [,<Liste von Hüllkurven>]

ENT und ENV dienen zur Definition von Hüllkurven für die Ausgabe von Tönen und Musikstücken auf dem Schneider-CPC.

Der einzige Ton, den Joyce sich entlocken läßt, ist ein simpler Piepser, der mit PRINT CHR$(7) aufgerufen wird. Deshalb sind hier Ton- und Lautstärke-Hüllkurven natürlich ohne Bedeutung.

EOF

Syntax: EOF

Will man erfahren, ob das Ende einer zum Lesen geöffneten Datei ("End of File") erreicht ist, kann man im CPC-Basic die Systemvariable EOF abfragen. Sie gibt dann den Wert -1 aus. Sonst ist EOF=0.

Mallard-Basic kann mehrere Dateien gleichzeitig offenhalten. Deshalb muß die Dateinummer stets explizit angegeben sein. Entsprechend unserer bei CLOSEIN getroffenen Vereinbarung ist die Lesedatei auf #1 festgelegt. Es heißt also statt PRINT EOF jetzt PRINT EOF(1).

ERASE

Keine Unterschiede

ERL

Syntax: ERL

ERL ist eine Systemvariable, die die Zeilennummer des letzten aufgetretenen Fehlers enthält.

Die Funktion ist auch in Mallard-Basic vorhanden. Allerdings gibt es einen Unterschied: Während der Schneider-CPC den Wert 0 ausgibt, wenn kein Programmfehler aufgetreten ist, meldet das Mallard-Basic den Wert 65535. Dies liegt daran, daß bei Mallard-Basic auch die Zeilennummer 0 als Programmzeile zulässig ist.

ERR

Syntax: ERR

Die Systemvariable ERR speichert den Code des letzten aufgetretenen Fehlers.

ERR kennt Mallard-Basic ebenfalls. Die meisten der Fehlernummern sind bei beiden Interpretern identisch. Es gibt folgende Unterschiede:

Code	Schneider-CPC	Joyce
21	Direct command found	O/S dependent error
24	EOF met	Unknown error
25	File type error	Unknown error
27	File already open	Unknown error
28	Unknown command	Unknown error
31	File not open	Unknown error
32	Broken in	Unknown error

ERROR

Syntax: ERROR <Fehlercode>

Der Befehl ERROR leitet bewußt eine Fehlerbehandlung ein. So gibt ERROR 2 einen Syntax-Error auf dem Bildschirm aus.

Beim Joyce gibt es ERROR ebenfalls. Die Fehlercodes differieren aber etwas, wie die bei der Funktion ERR aufgeführte Tabelle zeigt.

EVERY

Syntax: EVERY <Zeitperiode> [,<Timer>] GOSUB <Zeile>

EVERY dient im CPC-Basic zur Programmierung von Interruptroutinen. Das angegebene Unterprogramm wird zu der <Zeitperiode> regelmäßig aufgerufen.

Ein Ersatz im Mallard-Basic ist nicht möglich, da dieses keine Events (Ereignisse) und Basic-Interrupts zuläßt.

EXP

Keine Unterschiede

FILL

Syntax: FILL [<Stiftfarbe>]

FILL ist ein Grafikbefehl des Schneider-CPC, der unregelmäßig geformte Körper auf dem Bildschirm mit Farbe ausfüllt. Da Mallard-Basic von Hause aus nicht grafikfähig ist, kann kein Ersatzbefehl angegeben werden.

FIX

Keine Unterschiede

FOR - TO - STEP

Keine Unterschiede

FRAME

Syntax: FRAME

Nach Eingabe des Befehls FRAME wartet der Schneider-CPC, bis der Elektronenstrahl auf dem Monitor zurückgefahren ist. Dies führt zu flüssigeren Bewegungsabläufen bei bewegter Blockgrafik. FRAME ist beim Joyce nicht nötig und kann deshalb ersatzlos entfallen.

FRE

Keine Unterschiede

GOSUB

Keine Unterschiede

GOTO

Keine Unterschiede

GRAPHICS PAPER
GRAPHICS PEN

Syntax: GRAPHICS PAPER <Farbstift>
GRAPHICS PEN <Farbstift> [,<Videomodus>]

Diese beiden Befehle legen die Hintergrund- und Stiftfarbe des Grafikfensters beim Schneider-CPC fest. Sie sind nicht auf den Joyce übertragbar, da dieser nur eine monochrome Bildschirmausgabe besitzt.

HEX$

Syntax: HEX$(<Integerzahl>[,Stringlänge])

Die Funktion HEX$ wandelt eine dezimale Zahl in eine hexadezimale Zeichenkette um. Sie kennt auch das Mallard-Basic. Allerdings ist die Umkehrfunktion beim Joyce etwas sonderbar programmiert. Während beim

Schneider-CPC die Formulierungen PRINT &H3FAC und PRINT &3FAC identisch sind, akzeptiert Joyce nur die erste Schreibweise.

HIMEM

Syntax: HIMEM

Diese Systemvariable liefert die höchste von Basic-Programmen belegbare Speicheradresse. Auch Programme in Mallard-Basic können HIMEM verwenden. Sie müssen aber beachten, daß HIMEM hier einen völlig anderen Wert ausgibt. Das liegt im total unterschiedlichen Speicheraufbau der beiden Computer begründet.

IF - THEN - ELSE

Keine Unterschiede

INK

Syntax: INK <Farbstift>,<Farbe> [,Farbe]

Mit dem Basic-Befehl INK kann beim Schneider-CPC eine Farbe für einen der Farbstifte definiert werden. Bei Angabe von zwei Farben blinken diese abwechselnd im Takt.

Der Joyce ist nicht auf Farbwiedergabe ausgelegt. Deshalb erscheint INK hier unsinnig. In manchen Fällen dient aber INK nur dazu, die Schrift- und Hintergrundfarbe zu vertauschen. Dafür kann das folgende Programm benutzt werden, das den Hintergrund hell- und die Schriftzeichen dunkelschaltet:

```
100 ' hier XBIOS.BAS einbinden
110 ' Hintergrundfarbe einstellen **********
120 POKE reg.af+1,0
130 POKE reg.bc,2
140 POKE reg.bc+1,2
150 POKE xbios,&HCB
160 POKE xbios+1,0
```



```
170 adr=&HF000:CALL adr
180 ' Schriftfarbe einstellen **********
190 POKE reg.af+1,1
200 POKE reg.bc,1
210 POKE reg.bc+1,1
220 adr=&HF000:CALL adr
```

Geben Sie unterschiedliche Werte für das B- und das C-Register ein, erreichen Sie den Effekt, daß die entsprechenden Farben abwechselnd blinken.

INKEY

Syntax: INKEY(<Tastennummer>)

Dieser Befehl fragt ab, ob einzelne Tasten allein oder in Kombination mit SHIFT oder CONTROL (bei Joyce ALT) gedrückt wurden. Im Gegensatz zu INKEY$ wird der Tastaturpuffer nicht ausgelesen. Denn INKEY arbeitet direkt mit der Tastatur-Hardware.

INKEY erwartet als Argument die Nummer der Taste, die überprüft werden soll. Es liefert einen Wert, der den Zustand der Taste sowie der SHIFT- und CONTROL-Taste wiedergibt:

Wert	SHIFT gedrückt?	CONTROL/ALT gedrückt?	Taste gedrückt?
-1	unbedeutend	unbedeutend	nein
0	nein	nein	ja
32	ja	nein	ja
128	nein	ja	ja
160	ja	ja	ja

INKEY ist nur schwer zu simulieren. Das Einfachste ist es hier, die Tastennummer in den tatsächlichen Tastaturcode umzusetzen und die Tastatur dann mit INKEY$ zu untersuchen.

Eine Tabelle der Tastennummern und der ASCII-Wert der Tasten des Joyce finden Sie im Anhang Ihres Benutzerhandbuches. Sie kennen die Tastennummern sicherlich schon vom CP/M-Programm SETKEYS.COM, das die Tastaturbelegung umdefiniert.

Ein Beispiel: Das zu übersetzende CPC-Programm fragt ab, ob der Benutzer SHIFT-V gedrückt hat. Aus der Tabelle im CPC-Handbuch kann man erfahren, daß die Tastennummer von "V" den Wert 55 hat.

Das Originalprogramm sieht beispielsweise so aus:

```
10 IF INKEY(55) < >32 THEN 10
20 PRINT "SHIFT-V gedrückt!"
30 END
```

Der Ergebniswert 32 sagt aus:

- Die Taste "V" wurde gedrückt
- Die SHIFT-Umschalttaste wurde nicht betätigt
- Die CONTROL-Taste (ALT-Taste) wurde nicht gedrückt

SHIFT-V ist aber einfach ein großgeschriebenes "V". Also ersetzen wir die Zeilen durch eine INKEY$-Formulierung:

```
10 IF INKEY$< >"V" THEN 10
20 PRINT "SHIFT-V gedrückt!"
30 END
```

INKEY$

Keine Unterschiede

INP

Syntax: INP (<Kanalnummer>)

Die Funktion INP liest einen Wert über den angegebenen Kanal (Port) des Z80-Prozessors ein. Sie ist auch beim Joyce vorhanden. Allerdings arbeitet der Joyce nur mit 8 Bit breiten Portadressen, beim CPC sind die Ports 16 Bit breit. Programme, die mit INP arbeiten, sind deshalb nicht ohne tiefgreifende Änderungen übertragbar.

INPUT

Syntax: INPUT [#<Kanal>,][;][<Meldung><Trennzeichen>]
<Liste von Variablen>

Der INPUT-Befehl liest Daten vom angegebenen Kanal ein. Dies ist entweder die Tastatur (Kanal #0 bis #7) oder die Diskettenstation (#9). Daß auch die Konstruktion INPUT #8 (Druckerkanal) erlaubt ist, ist sicher nur ein Programmierfehler von Amstrad ...

Solange die Variablenwerte nur von der Tastatur angefordert werden, läßt sich der INPUT-Befehl auf Mallard-Basic auf Anhieb übertragen. Daten von Windows einzulesen, ist schon schwieriger. Dazu müssen Sie zuerst das Bildschirmfenster festlegen, wofür Sie eine Lösung bei der Beschreibung des Befehls WINDOW finden. Und die Diskettenstation sprechen Sie mit INPUT #1 an. Erinnern Sie sich noch? Diese Übereinkunft haben wir bei CLOSEIN getroffen.

INSTR

Keine Unterschiede

INT

Keine Unterschiede

JOY

Syntax: JOY(<Zahl>)

Mit der JOY-Funktion läßt sich ein angeschlossener Joystick abfragen. Dabei gibt die <Zahl> an, ob der Joystick 0 oder 1 untersucht wird. Der von der JOY-Funktion gemeldete Wert ist bitsignifikant:

Bitnummer	Wertigkeit	Bedeutung
0	1	aufwärts
1	2	abwärts
2	4	links
3	8	rechts
4	16	Feuertaste 2
5	32	Feuertaste 1

Durch Addition der Wertigkeiten können gleichzeitig gedrückte Joystick-Kombinationen festgestellt werden. So gibt "Joystick-abwärts" und "Joystick-links" und "Feuer 1 gedrückt" den Wert 2+4+32=38 aus.

An den Joyce kann - zumindest ohne Hardware-Änderungen - kein Joystick angeschlossen werden. Deshalb ist es wohl das Sinnvollste, den Joystick durch eine Tastaturabfrage zu simulieren.

Dazu nehmen wir für die vier Hauptbewegungsrichtungen die Tasten E, S, D und X her. Die Diagonalen werden durch W, R, Z und C repräsentiert. Die Taste "N" dient als Ersatz für die Feuertaste 1, die "M"-Taste als Feuertaste 2:

```
100 GOSUB 120:PRINT joy;:GOTO 100
110 ' Ersatz fuer die JOY-Funktion *********
120 a$=UPPER$(INKEY$)
130 IF a$="" THEN joy=0:GOTO 220
140 IF a$="E" THEN joy=1:GOTO 220
150 IF a$="S" THEN joy=4:GOTO 220
160 IF a$="D" THEN joy=8:GOTO 220
170 IF a$="X" THEN joy=2:GOTO 220
180 IF a$="W" THEN joy=4+1:GOTO 220
190 IF a$="R" THEN joy=8+1:GOTO 220
200 IF a$="Z" THEN joy=4+2:GOTO 220
210 IF a$="C" THEN joy=8+2
220 IF UPPER$(INKEY$)="M" THEN joy=joy OR 16
230 IF UPPER$ (INKEY$)="N" THEN joy=joy OR 32
240 RETURN
```

Wenn Sie jetzt in einem Listing die Zeile "A=JOY(0)" finden, schreiben Sie stattdessen "GOSUB 120:A=JOY".

KEY

Syntax: KEY <Funktionstasten-Nummer>,<Zeichenkette>

Der Key-Befehl definiert die Belegung einer der Funktionstasten mit einem String. Auf dem Joyce haben Sie zwei Möglichkeiten: Sie können entweder auf der CP/M-Systemebene das Programm SETKEYS.COM aufrufen oder mit einer Maschinensprache-Routine den KEY-Befehl simulieren. Letztgenannte Lösung finden Sie hier beschrieben. Sie binden wieder das am Anfang des Kapitels aufgelistete Basic-Programm ein, das den Zugriff auf das XBIOS ermöglicht:


```
100 ' hier XBIOS.BAS einbinden!
110 key$="Beispielstring"
120 ftaste=128
130 GOSUB 200
140 END
190 ' Ersatzroutine für KEY *********************
200 POKE xbios,&HD4:POKE xbios+1,0
210 FOR i=1 TO LEN(key$)
220   POKE &HF0FF+i,ASC(MID$(key$,i,1))
230 NEXT i
240 POKE reg.bc,LEN(key$)
250 POKE reg.bc+1,ftaste
260 POKE reg.hl,0
270 POKE reg.hl+1,&HF1
280 adr=&HF000:CALL adr
290 RETURN
300 END ' ***************************************
```

KEY DEF

Syntax: KEY DEF <Tastennummer>,<Repeat> [,Normal]
[,<Shift>] [,<Control>]

KEY DEF weist beim Schneider-CPC einer Taste einen neuen ASCII-Wert zu. Dabei gilt für die Tastennummer die Tabelle, die Sie bei diesem Buch im Anhang finden.

Zu jeder Taste kann festgelegt werden, ob eine Wiederholfunktion erwünscht ist und welche Codes die Taste auf normaler, SHIFT- und CONTROL-(ALT-)Ebene liefern soll.

Beim Joyce kann KEY DEF durch CP/M-Programm SETKEYS ersetzt werden, das allerdings nur auf der Kommandoebene von CP/M arbeitet. Eine Maschinensprache-Lösung für Mallard-Basic ist im folgenden abgedruckt:

```
100 ' hier XBIOS.BAS einbinden!
110 keynumber=66 ' Tastencode 66 (Escape)
120 char=33 ' Zeichen 33 ("!")
130 flag=1 ' Übersetzung nur für normale Ebene
140 GOSUB 200
150 END
190 ' Ersatzroutine für KEY DEF *********************
200 POKE xbios,&HD7
210 POKE xbios+1,0
220 POKE reg.bc,keynumber
230 POKE reg.bc+1,char
240 POKE reg.de+1,flag
250 adr=&HF000:CALL adr
260 RETURN
270 END ' ***************************************************
```

Dem Unterprogramm ab Zeile 200 werden verschiedene Variablenwerte übergeben:

• KEYNUMBER enthält die Nummer der Taste, für die eine Zeichenübersetzung durchgeführt werden soll.

• CHAR ist das Zeichen, das beim Drücken der Taste erscheinen soll.

• FLAG ist ein Kennerbyte, das bitweise codiert ist und angibt, für welche Tastenebene die Übersetzung wirksam sein soll:

Bit 0: Normale Tastenebene
Bit 1: SHIFT-Ebene
Bit 2: ALT-Ebene
Bit 3: SHIFT-ALT-Ebene
Bit 4: EXTRA-Ebene

Die übrigen Bits werden von der XBIOS-Systemroutine KM SET KEY ignoriert.

Die Nummern der Tastencodes können Sie dem CPC-Benutzerhandbuch entnehmen. Zusätzlich verarbeitet die Routine KM SET KEY noch die Werte 80H bis 9EH als Funktionstasten und 9FH als Kennzeichen dafür, daß eine Taste ingoriert werden soll.

LEFT$

Keine Unterschiede

LEN

Keine Unterschiede

LET

Keine Unterschiede

LINE INPUT

Syntax: LINE INPUT [#<Kanal>,] [;] [<Meldung>
<Trennzeichen>] <Stringvariable>

Dieser Befehl liest eine Zeichenkette von der Tastatur oder von der Diskettenstation ein. Dabei sind auch Kommas und Anführungszeichen zugelassen. Solange die Eingabe von der Tastatur erfolgt, kann LINE INPUT in Mallard-Basic übernommen werden. Soll dabei ein Bildschirmfenster benutzt werden, muß dieses vorher definiert worden sein (siehe WINDOW). Disketten-Leseoperationen ändern Sie am besten in LINE INPUT #1 ab.

LIST

Syntax: LIST [<Zeilenbereich>] [,#<Kanal>]

Mit LIST kann man sich die Zeilen eines Basic-Programms auflisten lassen. Bei LIST #8 erfolgt die Ausgabe auf dem Drucker. Dieser Befehl muß beim Joyce durch LLIST ersetzt werden. Der Schneider-CPC kennt außerdem die Konstruktion OPENOUT "NAME", LIST #9 und CLOSEOUT. Dies kann man mit SAVE "NAME",A simulieren.

LOAD

Syntax: LOAD <Dateiname> [,Adresse]

Der Basic-Befehl LOAD lädt Programme von der Diskette in den Speicher. Auf dem Schneider-CPC können auch Binärdateien so eingelesen werden. Letzteres ist beim Joyce nicht möglich.

LOCATE

Syntax: LOCATE [#<Fenster>,] <Spalte>,<Zeile>

Mit LOCATE positioniert der Schneider-CPC den Cursor. Diese Art der Cursoransteuerung ist bei Joyce nicht möglich. Definieren wir uns also eine Funktion, die eine Escape-Sequenz verwendet:

```
10 DEF FNlocate$ (spalte,zeile)=CHR$(27)+"Y"+CHR$
   (zeile+31)+CHR$(spalte+31)
```

Finden Sie also in einem CPC-Programm die Zeile

```
10 LOCATE 23,v+2
```

... dann ersetzen Sie sie durch:

```
10  PRINT FNlocate$(23,v+2);
```

LOG
LOG10

Syntax: LOG(<Zahl>)
LOG10(<Zahl>)

Diese beiden Funktionen berechnen den natürlichen Logarithmus und den Logarithmus zur Basis 10. Sie sind unverändert im Mallard-Basic einsetzbar.

Falls Sie übrigens jemals in die Lage geraten sollten, Logarithmen zu anderen Basen berechnen zu müssen, können Sie sich mit der folgenden Definition behelfen:

```
10 DEF FNlogarithm(basis,wert)=LOG(wert)/LOG(basis)
20 ' Beispiele ----------------------------------------
30 PRINT LOG(9999);FNlogarithm(2.718282,9999)
40 PRINT LOG10(9999);FNlogarithm(10,9999)
50 PRINT "log(5) 3 = ";FNlogarithm(5,3)
```

LOWER$

Syntax: LOWER$(<Zeichenkette>)

Die Funktion LOWER$ wandelt alle Großbuchstaben einer Zeichenkette in Kleinbuchstaben um. Sie ist sowohl auf dem Schneider-CPC als auch JOYCE-PCW vorhanden.

Allerdings scheint die LOWER$-Funktion nicht in allen Versionen des Mallard-Basic richtig zu funktionieren. Bei Eingabe des Befehls PRINT LOWER$("A") erscheint beispielsweise nicht ein kleines "a", sondern das Grafikzeichen mit dem ASCII 128. Dieser Fehler ist aber bis jetzt nur dann aufgetreten, wenn der umzuwandelnde String direkt angegeben war. Offensichtlich gibt es Probleme mit der Stringverwaltung.

Finden Sie in einem Listing die Zeile

```
10 PRINT LOWER$("HALLO")
```


dann schreiben Sie stattdessen:

```
10 a$="HALLO":PRINT LOWER$(a$)
```

MASK

Syntax: MASK <Zahl>
MASK <Zahl>,<Zahl 2>
MASK ,<Zahl 2>

Der Mask-Befehl gehört zu den Kommandos, die beim Schneider-CPC grafikorientiert sind. Er legt das Linienmuster von DRAW- und DRAWR-Aufrufen fest. Da der Joyce mit Mallard-Basic keine hochauflösenden Grafiken erzeugen kann, ist ein Ersatz nicht möglich.

MAX

Keine Unterschiede

MEMORY

Syntax: MEMORY <Adresse>

Basic-Programme nutzen normalerweise den gesamten verfügbaren Speicherplatz aus. Wenn man Maschinenprogramme verwenden will, setzt man im allgemeinen die obere Speichergrenze herunter und reserviert sich so den benötigten Speicherbereich. MEMORY sorgt dafür, daß der Basic-Interpreter diesen Speicher keinesfalls mehr verändert. Auch Joyce kennt den MEMORY-Befehl. Programme können aber dennoch nicht ohne Änderungen übernommen werden, da die Speicherorganisation der Computer zu unterschiedlich ist.

MERGE

Keine Unterschiede

MID$

Syntax: MID$(<Zeichenkette>,<Start>,[,<Länge>])

Mit der Funktion MID$ lassen sich Zeichenketten zerteilen. Sie entnimmt die angegebene Anzahl von Zeichen aus dem String, beginnend mit der

<Start>-Position. Wenn die Längenangabe weggelassen wird, nimmt Basic die Gesamtlänge des Strings an.

Eine spezielle Form dieser Funktion erlaubt es, MID$ links des Zuweisungsoperators zu schreiben:

```
a$="Joyce-Computersystem"
MID$(s$,7)="Schreib-S"
WRITE a$
```

Der Computer zeigt an: "Joyce-Schreib-System"

Beide Formen der MID$-Funktion werden vom Mallard-Basic verstanden.

MIN

Keine Unterschiede

MOD

Keine Unterschiede

MODE

Syntax: MODE <Zahl>

Beim Schneider-CPC übernimmt MODE die Aufgabe, zwischen den verschiedenen Bildschirm-Modi umzuschalten. Denn dieser Computer kann 20, 40 und 80 Zeichen pro Zeile darstellen. Je höher die Auflösung ist, desto weniger Farben können auf einmal auf den Bildschirm gebracht werden:

MODE	Zeichen pro Zeile	Auflösung	Farben
0	20	160 * 200	16
1	40	320 * 200	4
2	80	640 * 200	2

Falls Sie sich wundern, warum die Auflösung in Y-Richtung nur

zweihundert Punkte beträgt, die Grafikbefehle aber mit einer 400-Punkt-Auflösung arbeiten, finden Sie hier des Rätsels Lösung: Der CPC kann wirklich nur 200 Punkte in der Bildschirmhöhe zeigen. Würde der Computer aber mit den Koordinaten 640 mal 200 Punkte arbeiten, sähen Kreise wie Ellipsen aus. Durch die interne Korrektur, die vom Betriebssystem automatisch durchgeführt wird, ist ein Kreis wirklich ein Kreis.

Den Mode-Befehl auf dem Joyce nachzubilden, ist nicht möglich. Sie können aber ein Fenster in der entsprechenden Größe setzen.

MOVE
MOVER

Syntax: MOVE <X-Koordinate>,<Y-Koordinate>
MOVER <X-Abstand>,<Y-Abstand>

Dieses Basic-Kommando legt die Position des Grafikcursors neu fest.

Auch MOVER positioniert den Cursor für die Grafikausgabe. Allerdings berechnet er die Koordinaten in Abhängigkeit von seiner alten Position. So verschiebt MOVER -2,4 den Cursor zwei Pixel nach links und vier nach oben.

Nur in Zusammenarbeit mit GSX oder über Maschinenroutinen kann Mallard-Basic Grafiken darstellen.

NEW

Keine Unterschiede

NEXT

Keine Unterschiede

NOT

Keine Unterschiede

ON BREAK CONT

Syntax: ON BREAK CONT

Dieser Befehl verhindert, daß der Benutzer ein Programm durch Drücken der ESCAPE-Taste abbricht. Bei Joyce hingegen muß der Befehl OPTION RUN verwendet werden.

ON BREAK GOSUB

Syntax: ON BREAK GOSUB <Zeilennummer>

Sobald der Benutzer ein Programm durch Drücken von ESCAPE stoppen will, ruft der Interpreter das Unterprogramm auf, das bei der <Zeilennummer> beginnt. ON BREAK GOSUB läßt sich auf dem Joyce nicht simulieren. Hier muß man sich entscheiden, ob die Programmunterbrechung erlaubt (OPTION STOP) oder untersagt ist (OPTION RUN).

ON BREAK STOP

Syntax: ON BREAK STOP

Dieser Befehl gestattet es dem Benutzer eines Basic-Programms, dieses per Tastendruck wieder zu unterbrechen. Bei Joyce heißt das Pendant OPTION STOP.

ON ERROR GOTO

Keine Unterschiede

ON - GOSUB

Keine Unterschiede

ON - GOTO

Keine Unterschiede

ON SQ GOSUB

Syntax: ON SQ(<Tonkanal>) GOSUB <Zeilennummer>

Der Schneider-CPC kann gleichzeitig ein Basic-Programm abarbeiten und Musik ertönen lassen. Dazu werden die Töne in eine Warteschlange eingereiht. Der Befehl ON SQ(x) GOSUB dient zum Aufruf eines Unterprogramms, das Töne nachlädt, wenn die Warteschlange geleert ist. Da der Joyce außer einem simplen Piepser über keine weitere Tonausgabe verfügt, kann kein Ersatzbefehl angegeben werden.

OPENIN

Syntax: OPENIN <Dateiname>

Hiermit kann man auf dem Schneider-CPC eine Disketten- oder Cassettendatei zum Lesen öffnen. Bei Joyce muß der OPEN-Befehl dagegen so ausgedrückt werden:

```
OPEN "I",1,"Dateiname"
```

Dabei gehen wir wieder von unserer Vereinbarung aus, zum Lesen geöffnete Dateien mit der Filenummer #1 zu kennzeichnen.

OPENOUT

Syntax: OPENOUT <Dateiname>

Analog zu OPENIN öffnet der Basic-Befehl OPENOUT eine Datei, in die Datensätze geschrieben werden sollen. Auf Joyce formuliert man das so:

```
OPEN "O",2,"Dateiname"
```

Alle Schreiboperationen werden ja über den Kanal #2 abgewickelt.

OR

Keine Unterschiede

ORIGIN

Syntax: ORIGIN <x-Koordinate>,<y-Koordinate> [,<links>,
<rechts>,<oben>,<unten>]

Der ORIGIN-Befehl legt den Koordinaten-Ursprung von Grafikausgaben fest. Außerdem können die Dimensionen des Grafikfensters bestimmt werden. Das Mallard-Basic ist nicht grafikfähig.

OUT

Syntax: OUT <Kanalnummer>,<Zahl>

Über einen der Ports (Kanäle) des Z80-Prozessors wird ein Wert ausgegeben. Der Schneider-CPC arbeitet mit 16-Bit-Portadressen, die Ports von Joyce sind nur acht Bit breit. Deswegen kann man einen OUT-Befehl nicht unbesehen übernehmen.

PAPER

Syntax: PAPER [#<Fensternummer>,] <Stiftnummer>

PAPER legt fest, daß die Hintergrundfarbe durch die angegebene Stiftfarbe repräsentiert werden soll. Farbbefehle wie dieser sind natürlich auf dem monochromen Monitor von Joyce unsinnig.

PEEK

Syntax: PEEK(<Adresse>)

Die PEEK-Funktion liefert den Wert, der in der benannten Speicherstelle steht. Sie ist auch beim Joyce vorhanden. Allerdings stimmen die Systemvariablen von Joyce und CPC nicht überein, so daß PEEK nicht unverändert übernommen werden kann.

PEN

Syntax: PEN [#<Fensternummer>,] <Farbstift>

Dieser Basic-Befehl bestimmt, welche INK-Farbe zur Ausgabe von Texten benutzt werden soll. Auf Joyce sind Farben nicht darstellbar.

PI

Syntax: PI

Diese Funktion liefert den Wert der Kreiszahl PI. Sie ist beim Joyce nicht vordefiniert, läßt sich aber als Variable einsetzen. Schreiben Sie deshalb an den Anfang Ihres Programms:

```
10 PI=3.1415926
```

PLOT
PLOTR

Syntax: PLOT <X-Koordinate>, <Y-Koordinate>[,Farbstift]
PLOTR <X-Koordinate>,<Y-Koordinate>[,Farbstift]

Der Befehl PLOT setzt einen Grafikpunkt (Pixel) an die durch die X- und Y-Koordinate bestimmte Position auf dem Bildschirm. PLOTR hingegen setzt den Punkt abhängig von den vorherigen Koordinaten des Grafikcursors. Beim Joyce läßt sich PLOT nur mit GSX realisieren.

PRINT

Syntax: PRINT [#<Ausgabekanal>, <Ausgabeliste>

PRINT gibt Daten auf einem Peripheriegerät aus. PRINT ohne Kanalangabe und PRINT #0 sind identisch. Sie sprechen den Bildschirm an. PRINT #8 arbeitet mit dem Drucker und ist auf dem Joyce durch LPRINT zu ersetzen. PRINT #9 schreibt Daten auf die Diskette. Sie sollten es mit PRINT #2 simulieren.

Die Sonderformen PRINT TAB, PRINT SPC und PRINT USING sind beim Joyce gleichermaßen vorhanden.

RAD

Syntax: RAD

Die trigonometrischen Funktionen wie Sinus und Cosinus können auf dem Schneider-CPC im Bogenmaß (Befehl RAD) und Winkelgrad-Maß (Basic-Befehl DEG) berechnet werden. Das Mallard-Basic auf Joyce befindet sich stets im RAD-Modus. Deshalb können Sie den Befehl RAD einfach weglassen.

RANDOMIZE

Keine Unterschiede

READ

Keine Unterschiede

RELEASE

Syntax: RELEASE <Tonkanal>

Das ist ein CPC-spezifisches Kommando. Es hat die Aufgabe, Töne, die sich im Wartezustand befinden, freizugeben. Aufgrund fehlender Existenz eines Soundgenerators ist bei Joyce RELEASE nicht zu simulieren.

REM

Syntax: REM <Kommentar>

Die REM-Anweisung fügt Kommentare ein. Sie kann - wie auch bei Joyce - durch einen Apostroph abgekürzt werden.

REMAIN

Syntax: REMAIN(<Timer>)

Die REMAIN-Funktion liefert die Restzeit des Timers bei der Verarbeitung von Interrupts. Sie läßt sich in Mallard-Basic nicht nachbilden.

RENUM

Keine Unterschiede

RESTORE

Keine Unterschiede

RESUME
RESUME NEXT

Keine Unterschiede

RETURN

Keine Unterschiede

RIGHT$

Keine Unterschiede

RND

Keine Unterschiede

ROUND

Keine Unterschiede

RUN

Keine Unterschiede

SAVE

Syntax: SAVE <Dateiname>
SAVE <Dateiname>,P
SAVE <Dateiname>,A
SAVE <Dateiname>,B,<Startadresse>,<Länge>

Der SAVE-Befehl sichert ein Basic- oder Maschinenprogramm als Datei auf der Diskette. Abgesehen von Binärfiles - Option "B" - sind die anderen Spielarten von SAVE auch auf dem Joyce vorhanden.

SGN

Keine Unterschiede

SIN

Syntax: SIN(<Zahl>)

Mit SIN können Sie den Computer den Sinus-Wert einer Zahl errechnen lassen.

Natürlich verfügt auch Mallard-Basic über die Funktion. Aber es berechnet die Werte stets wie der Schneider-CPC, wenn dieser sich im RAD-Modus befindet. Um den DEG-Modus zu simulieren, sollten Sie das Ergebnis durch (PI/180)=1.745329E-02 teilen:

```
pi=3.141593
deg=pi/180
PRINT SIN(0.9)/deg
```

SOUND

Syntax: SOUND <Kanal>,<Tonperiode> [,<Länge>]
[,<Lautstärke>]
[,<ENV-Hüllkurve>]
[,<ENT-Hüllkurve>]
[,<Geräusch>]

Die vielen Optionen, die der SOUND-Befehl anbietet, machen die Situation auch nicht besser: SOUND läßt sich nicht so herrichten, daß Mallard-Basic diesen Befehl versteht.

SPACE$

Keine Unterschiede

SPEED INK

Syntax: SPEED INK <Zeitperiode 1>,<Zeitperiode 2>

Der Schneider-CPC kann zwei Farben abwechselnd blinken lassen. Mit SPEED INK legt man das Zeitverhältnis fest. Bei Joyce ist diese Fähigkeit nicht vorhanden. Hier ist es lediglich möglich, den ganzen Bildschirm blinken zu lassen. Dazu muß man die Routine TE SET SPEED im System-BIOS aufrufen. Wie das geht, haben Sie im Kapitel über den XBIOS-Sprungblock erfahren.

SPEED KEY

Syntax: SPEED KEY <Startverzögerung>,
<Wiederholungsverzögerung>

Die Wiederholungs-Geschwindigkeit der Tastatur kann der Programmierer auf dem Schneider-CPC mit SPEED KEY festlegen. Beim Joyce ist das nur auf Maschinensprache-Ebene möglich. Hierzu seien Sie auf das Kapitel über den XBIOS-Sprungblock verwiesen.

SPEED WRITE

Syntax: SPEED WRITE 0
SPEED WRITE 1

SPEED WRITE legt die Geschwindigkeit fest, mit der der Cassettenrecorder beim Schneider-CPC Daten aufzeichnet. An Joyce kann ohnehin kein Recorder angeschlossen werden.

SQ

Syntax: SQ (<Tonkanal>)

Gibt den Status eines der Tonkanäle des Soundgenerators im Schneider-CPC an. Programme müssen beim Vorkommen dieser Funktion ziemlich stark abgeändert werden.

SQR

Keine Unterschiede

STOP

Syntax: STOP

Keine Unterschiede

STR$

Keine Unterschiede

STRING$

Syntax: STRING$ (<Zeichenzahl>, <Zeichen>)
STRING$ (<Zeichenzahl>, <Zeichencode>)

Die STRING$-Funktion liefert die entsprechende Anzahl von Zeichen. Das CPC-Handbuch verschweigt aber, daß der Computer zwei verschiedene syntaktische Formen akzeptiert:

```
PRINT STRING$ (20, "*")
PRINT STRING$ (20, 42)
```

42 ist bekanntlich der ASCII-Wert des Zeichens "*". Auch das Mallard-Basic verarbeitet die beiden Formen ohne Einschränkung.

SYMBOL

Syntax: SYMBOL <Zeichennummer>, <Liste von Zahlen>

Der SYMBOL-Befehl definiert Grafikzeichen auf dem Schneider-CPC. Dies setzt voraus, daß das entsprechende Zeichen vorher mit SYMBOL AFTER in den RAM-Speicher kopiert wurde. Auch der Joyce besitzt einen frei

definierbaren Zeichensatz. Allerdings ist diese Fähigkeit sehr tief im Computer verborgen und wird nicht vom Betriebssystem unterstützt.

Das ist aber für uns kein Hindernis: Durch Schreiben von Bytes in die Matrixtabelle (Adresse 0B800H in der Systembank) kann man auch bei Joyce Zeichen definieren.

SYMBOL AFTER

Syntax: SYMBOL AFTER <Zahl>

Der SYMBOL AFTER-Befehl kopiert den Zeichensatz vom angegebenen Zeichen bis zum Zeichen 255 in den RAM-Speicher. Dort kann er mit SYMBOL manipuliert werden. Der Joyce-Zeichensatz steht ohnehin komplett im RAM, allerdings versteckt in der Systembank.

TAG
TAGOFF

Syntax: TAG
TAGOFF

Texte werden vom Schneider-CPC normalerweise an der Position des Textcursors ausgegeben. Nach TAG erscheint der Text an Grafikpositionen. TAGOFF schaltet wider in den Normalzustand zurück. Nur in Verbindung mit GSX oder Maschinensprache wird Mallard-Basic in die Lage versetzt, hochauflösende Grafik darzustellen.

TAN

Syntax: TAN (<Zahl>)

Berechnet den Tangens des angegebenen Ausdrucks.

Mallard-Basic kennt ebenfalls die TAN-Funktion. Es berechnet aber die Werte stets wie der RAD-Modus des Schneider-CPC. Um den DEG-Modus zu simulieren, sollten Sie das Ergebnis durch (PI/180) = 1.745329E-02 teilen:

```
pi=3.141593
deg=pi/180
PRINT TAN (0.5)/deg
```

TEST
TESTR

Syntax: TEST (<X-Koordinate>, <Y-Koordinate>)
TESTR (<X-Koordinate>, <Y-Koordinate>)

TEST prüft, welche Farbe ein bestimmter Punkt auf dem Bildschirm besitzt. Die Farbstiftnummer wird von der Funktion übergeben. TEST arbeitet mit absoluten, TESTR mit relativen Koordinaten. Mallard-Basic ist bekanntlich nicht grafikfähig.

TIME

Syntax: TIME

Die Funktion TIME liefert die Zeit, die seit dem Einschalten des Computers vergangen ist. Dabei gilt die Zeiteinheit 1/300 Sekunden.

Joyce besitzt zwar eine interne Uhr. Diese läßt sich aber in Basic nicht so einfach auslesen. Das folgende Programm simuliert die TIME-Funktion durch Aufruf der BDOS-Funktion 105, "Get Date And Time":

```
100 ' Simulation der TIME-Funktion ********************
110 DATA &H0E,&H69,&H11,&H90,&H00,&HCD,&H05,&H00
120 DATA &H32,&H94,&H00,&HC9
110 FOR i=&H80 TO &H8B:READ a:POKE i,a:NEXT i
120 adr=&H80:CALL adr
130 hour=PEEK(&H92)
140 hour$=CHR$(48+(hour AND 240)/16)+CHR$(48+(hour AND
    15))
150 hour=VAL(hour$)
160 minute=PEEK(&H93)
170 minute$=CHR$(48+(minute AND 240)/16)+CHR$(48+(minu
    te AND 15))
180 minute=VAL(minute$)
190 second=PEEK(&H94)
200 second$=CHR$(48+(second AND 240)/16)+CHR$(48+(seco
    nd AND 15))
210 second=VAL(second$)
220 ' Hier eventuell: PRINT hour,minute,second
230 time=300*(hour*3600+minute*60+second)
240 PRINT time
250 GOTO 120
```


Die Auflösung dieses Timers ist nicht so gut, weil die CP/M-Uhr nur in Sekundenschritten zählt, nicht dreihundertmal pro Sekunde, wie bei der Basic-Funktion TIME. Im allgemeinen reicht allerdings diese Auflösung vollkommen aus.

TRON
TROFF

Keine Unterschiede

UNT

Keine Unterschiede

UPPER$

Syntax: UPPER$ (<Zeichenkette>)

Diese Funktion wandelt alle Kleinbuchstaben in Zeichenketten in Großbuchstaben um. Sie funktioniert bei Joyce aber nur dann richtig, wenn keine direkte Zeichenkette angegeben wird. Statt PRINT UPPER$ ("joyce") wählen Sie besser den Umweg über eine Variablenzuweisung:

```
a$="joyce"
PRINT UPPER$(a$)
```

VAL

Keine Unterschiede

VPOS

Syntax: VPOS (#<Fensternummer>)

Die Funktion VPOS meldet die Nummer der Zeile, in der sich der Cursor befindet. Auf dem Joyce muß sie durch einen Aufruf des XBIOS-

Sprungblocks simuliert werden. Statt x=VPOS (#0) schreiben Sie GOSUB 130:x=VPOS. Sie fügen außerdem an Ihr Programm die folgenden Zeilen an:

```
100 ' VPOS-Simulation für Joyce **********************
110 GOSUB 190:PRINT vpos:END
120 ' Unterprogramm: VPOS-Simulation ----------------
130 MEMORY &HF000-1
140 FOR i=&HF000 TO &HF025
150     READ byte
160     POKE i,byte
170 NEXT i
180 adr=&HF000:CALL adr
190 vpos=PEEK(&HF02E)
200 RETURN
210 DATA 33,41,240,34,52,0,42,1,0,17,87,0,25,34,39,240
220 DATA 62,195,50,38,240,205,38,240,191,0,237,67,41
230 DATA 240,237,83,43,240,34,45,240,201
240 END
```

WAIT

Syntax: WAIT <Portnummer>, <Maske> [,<Inversion]

Der WAIT-Befehl weist den Computer an, solange zu warten, bis ein bestimmter Wert von einem Port des Z80-Prozessors eingelesen werden kann. Auch wenn Joyce den WAIT-Befehl versteht, kann er nicht übernommen werden, da er direkt auf die Hardware zugreift. Diese ist aber bei den beiden Computern sehr unterschiedlich aufgebaut.

WHILE - WEND

Keine Unterschiede

WIDTH

Syntax: WIDTH <Zahl>

WIDTH hat beim Schneider-CPC die Aufgabe, die Länge von Zeilen auf dem Drucker festzulegen. Sobald diese Länge erreicht ist, fügt der Computer automatisch einen Wagenrücklauf in den übertragenen Text ein.

WIDTH steuert beim Joyce hingegen die Breite des Bildschirms. Zur Druckerausgabe müssen Sie WIDTH LPRINT <Zahl> aufrufen.

WINDOW

Syntax: WINDOW [#<Fensternummer>,] <links>, <rechts>,
<oben>, <unten>

Der Befehl WINDOW definiert eines der Bildschirmfenster von #0 bis #7, die der Schneider-CPC verwalten kann. Der VT52-Terminalemulator von Joyce kennt nur ein einziges Fenster, das mit dem Steuerzeichen ESC X definiert wird.

WINDOW SWAP

Syntax: WINDOW SWAP <Fenster1>, <Fenster2>

Vertauscht die Definitionen zweier Fenster. Da der Joyce ohnehin nur ein einziges Fenster verwalten kann, ist WINDOW SWAP unsinnig.

WRITE

Syntax: WRITE [#<Gerät>,] [Ausgabeliste]

Ist auch in Mallard-Basic vorhanden. Lediglich die Angabe des Peripheriegerätes ist problematisch. Denn #0 bis #7 stehen für Bildschirmfenster, #8 für den Drucker und #9 für Diskettendateien.

XPOS
YPOS

Syntax: XPOS
YPOS

Diese beiden Funktionen geben beim Schneider-CPC die Position des Grafikcursors an. Mallard-Basic ist ja nicht grafikfähig. Deshalb kann kein Ersatzbefehl genannt werden.

ZONE

Keine Unterschiede

Mit diesen Kenntnissen ist es nun (hoffentlich) für Sie ein Leichtes, Basic-Programme des Schneider-CPC auch für Mallard-Basic auf Joyce verfügbar zu machen. Viel Erfolg dabei!

5 Literatur

CP/M Plus Operating System Programmer´s Guide, Digital Research, Pacific Grove (CA) 1983

CP/M Plus Operating System System Guide, Digital Research, Pacific Grove (CA) 1983

CP/M Plus Operating System User´s Guide, Digital Research, Pacific Grove (CA) 1983

CP/M 3 on the PCW8256, Locomotive Software, 1985

6 Disketten-Bezugsquelle

Alle in diesem Buch abgedruckten Programme sind direkt beim Autor auf Diskette erhältlich. Die 3-Zoll-Diskette kostet DM 30,- inklusive Versand und Verpackung.

Bestellanschrift:
Martin Kotulla
Grabbestraße 9
8500 Nürnberg 90

7 Sachverzeichnis

A
A: 51
A> 46
ALT-I 48
ALT-P 48
Alphabetische Befehlsliste 112

B
B: 51
BASIC 48
BDOS 69
BDOS-Funktionsliste 70-72
BIOS 73
BIOS-Funktionsliste 74/75
Betriebssystem 42
Bildschirmansteuerung 10

C
CP/M 42
CP/M-86 45
CP/M-Dienstprogramme 63
Commodore-64 42
Cursorpositionierung 19

D
DEF FN 31
DIR 48, 49, 56-61
DIRS (DIRSYS) 52, 54
DOUBLE.BAS 39, 40
DPB 83-86
Datumsberechnung 11, 20-22
Digital Research 43, 45
Direct BIOS Call 73
Diskettenformat 110

E
ERA (ERASE) 51, 52, 56, 61/62
Editiertasten unter CP/M 47

F
Farbbänder 39, 40
Funktionstasten 18

G
Grafik 10
Grafik 109
Grundrechenarten 33

H
HEX2DATA.BAS 106/107
Horner-Schema 38

I
IBM-PC 45
Intel 44
Interrupts 110

J
J14GCPM3.EMS 46

K
KALENDER.BAS 11, 14
Kildall 43

L
Laufwerksimulator 50
Literatur 155

M
M: 51
MP/M 45
MS-DOS 46
Mallard-Basic 108
Menüsteuerung 10, 18, 32/33
Microcomputer Application Ass. 44
Microsoft 45

N
Nachbrenner 56

O
Oktalzahlen 31/32

P
PIP 63-66
PL/M 44

Q
QDOS 45

R
RAM-Diskette 50
REN (RENAME) 51, 52, 53, 56, 62

S
SCR RUN ROUTINE 103-106
SET 67/68
SET24X80 108
SHOW 66
Schneider-CPC 42
Shugart 44
Standard-Betriebssystem 43
Systemroutinen 69

T
TAB 48
TYP (TYPE) 52, 53, 54, 56, 63
Torode 44

U
UNT (x) 31
USE (USER) 52, 55
Umlenkung der Bildschirmausgabe 22-27

W
Wildcards 49

X
XBIOS 75, 112
XBIOS-Aufruf 79-83
XBIOS-Funktionsliste 86-103
XBIOS: Bildschirmausgabe 77/78
XBIOS: Diskettenroutinen 76/77
XBIOS: Diverse Routinen 78
XBIOS: Serielle Schnittstelle 77
XBIOS: Tastatur 78
XDPB 83-86

Z
Z80, Z80A 44, 109
Zahlensysteme 28
Zahlenumwandlungen 33
Zahlenwandler 34-38