

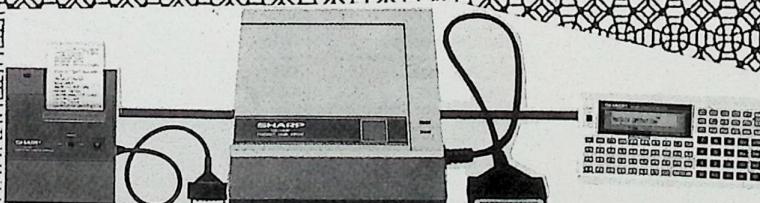
FÜR

SHARP

PC-1403

POCKET-COMPUTER

SYSTEMHANDBUCH



CE-126P
24-digit
thermal printer

CE-140F
2.5" pocket
disk drive

PC-1403
Pocket computer

FISCHEL BERLIN 1987

INGO LAUE

ISBN | 3-924327-56-4

SYSTEMHANDBUCH

für SHARP PC-1403

Pocket-Computer

Autor: Ingo Laue

Herausgeber: Fischel GmbH

Berlin 1987.

ISBN 3-924327-56-4

Copyright © 1987 by Fischel GmbH, Berlin

Alle Rechte vorbehalten.

Ohne ausdrückliche Genehmigung des Herausgebers ist es
nicht gestattet, das Buch oder Teile daraus auf
fotomechanischem (Fotokopie, Mikroskopie) oder sonstigem
Wege zu vervielfältigen.

Für etwaige Schäden durch Anwendung der Anleitungen oder
Programme dieses Buches übernehmen wir keine Haftung.

Druck: Offsetdruckerei Gerhard Weinert GmbH
Saalburgstraße 3, 1000 Berlin 42

Vorwort

Zum Jahreswechsel 1986/87 hat die Firma SHARP ein neues Mitglied ihrer Pocket-Computer-Familie vorgestellt: den PC-1403.

Als bewährte Kombination aus einem BASIC-Taschencomputer und einem wissenschaftlichen Taschenrechner mit festprogrammierten Funktionen reiht er sich nahtlos in die bestehende PC-14xx-Serie ein und wartet mit einer von vielen gewünschten Besonderheit auf: Er verfügt über ein integriertes, umfangreiches Programmepaket zum Rechnen mit Matrizen.

Das vorliegende Systemhandbuch für diesen Taschencomputer ist in fünf Abschnitte gegliedert: Nach einer einführenden Erläuterung einiger Grundbegriffe der Computertechnik folgt ein genauer Blick hinter die Kulissen des Rechners. Adressierung und Formatierung von Programmen und Daten werden untersucht. Es folgt ein kurzes Kapitel über Matrizenrechnung; daran schließen sich einige Tips und Tricks an. Last but not least bildet eine umfangreiche Programmsammlung das Schlußlicht.

Man lernt seinen Computer am besten kennen, indem man ihn benutzt. Deshalb werden immer wieder kleine Beispiele und Demonstrationsprogramme in den Text eingeflochten.

Zusammengefasst bietet dieses Buch allen PC-1403-Benutzern, die nicht nur fertige Programme laufen lassen möchten, sondern sich auch dafür interessieren, was sich genauer im Rechner abspielt und wie er intern organisiert ist, eine Menge wissenswerter Informationen, die ein neues, viel breiteres Anwendungsfeld erschließen.

im März 1987

Ingo Laue

Inhaltsverzeichnis

| | |
|---------------------------------------------------------------|----|
| Einleitung | 4 |
| 1. Grundlagen | 5 |
| 1.1 Zubehör | 5 |
| 1.2 Zahlensysteme | 5 |
| 1.3 Bits, Bytes, Adressen | 7 |
| 1.4 Speichertypen | 8 |
| 1.5 POKE und PEEK | 9 |
| 1.6 OR und AND | 11 |
| 2. Das System des PC-1403 | 13 |
| 2.1 Speicherplan | 13 |
| 2.2 Funktion der einzelnen Speicher | 14 |
| 2.3 Periodizitäten von Adressen | 14 |
| 2.4 Grundaufbau des User-RAMs | 15 |
| 2.5 Speicherformat von BASIC-Programmen | 16 |
| 2.6 Rettung gelöschter Programme | 18 |
| 2.7 RENEW als Maschinenprogramm | 19 |
| 2.8 Die Standardvariablen | 21 |
| 2.8.1 Format von numerischen Variablen | 21 |
| 2.8.2 Format von Stringvariablen | 23 |
| 2.9 Der interne Code | 23 |
| 2.10 Das Passwort | 24 |
| 2.11 Direktansteuerung der Flüssigkristallanzeige | 25 |
| 2.12 Speicherung der Rechnermodus-Anzeige | 29 |
| 2.13 Der Piezo-Summer | 30 |
| 3. Matrizenrechnung | 37 |
| 3.1 Was sind Matrizen ? | 37 |
| 3.2 Matrizenmultiplikation | 40 |
| 3.3 Matrizeninversion | 42 |
| 3.4 Lösung linearer Gleichungssysteme | 44 |
| 3.5 Eigenwerte und Eigenvektoren | 46 |
| 4. Programme und Programmiertricks | 49 |
| 4.1 Verzicht auf den IF-Befehl | 49 |
| 4.2 Rechenun genauigkeit | 51 |
| 4.3 Die RND-Funktion | 52 |
| 4.3.1 Bestimmung von π nach der Monte-Carlo-Methode | 52 |

| | |
|---------------------------------------------------------------------|-----|
| 4.3.2 Spiel 17+4 | 54 |
| 4.4 Anzeige von Indizes beim INPUT-Befehl | 55 |
| 4.4.1 Berechnung der Inversen einer Matrix | 57 |
| 4.5 Sortierverfahren | 58 |
| 4.5.1 Bubble Sort | 59 |
| 4.5.2 Sortierprogramm mit Min-Max-Suche | 60 |
| 5. Programme | 61 |
| 5.1 Monatskalender | 62 |
| 5.2 Berechnung beweglicher Feiertage | 63 |
| 5.3 Berechnung von Sonnenauf- und -untergang | 65 |
| 5.4 Berechnung der Mondphase | 69 |
| 5.5 Berechnung komplexer Wurzeln | 71 |
| 5.6 Umwandlung arabischer Zahlen in das römische Zahlensystem | 73 |
| 5.7 Integration einer Funktion | 75 |
| 5.8 Primzahlberechnung nach Eratosthenes | 76 |
| 5.9 Tic Tac Toe | 78 |
| 5.10 Erzeugung eines magischen Quadrats | 80 |
| 5.11 Buchstabenkombinationen eines Wortes | 82 |
| 5.12 Simulation von Byte-Operationen | 84 |
| 5.13 Ausdruck des Verlaufs einer Funktion | 86 |
| 5.14 Digitaluhr | 88 |
| 5.15 Strategiespiel AWELE | 93 |
| 5.16 Glücksspielprogramm | 96 |
| 5.17 Reaktionstestprogramm/Stoppuhr | 98 |
| 5.18 Gedächtnistest | 101 |
| A. Anhang | 102 |
| A.1 Literaturverzeichnis | 102 |
| A.2 Der interne Code des PC-1403 | 104 |
| A.3 Die Tastatormatrix | 105 |
| A.4 Hardware-Reset | 106 |
| A.5 Abspeichern von Maschinenprogrammen auf Cassette | 107 |
| A.6 Technische Daten des PC-1403 | 108 |
| A.7 Das Diskettenlaufwerk CE-140F | 109 |

Einleitung

Dem vorliegenden Systemhandbuch liegt das PC-12xx-Anwendungshandbuch /1/ zugrunde, dessen bewährtes Konzept im wesentlichen übernommen worden ist.

Im ersten Teil des Buches werden zunächst Grundbegriffe erklärt, um auch Anfängern den Einstieg zu ermöglichen.

Im zweiten Teil wird das Innenleben des Computers untersucht. Es wird ein Überblick über die verschiedenen Speicher, insbesondere den User-RAM, gegeben. Das Format, in welchem Programme und Daten gespeichert werden, wird eingehend beschrieben. Des Weiteren werden die Anzeige und der Piezo-Summer genauer unter die Lupe genommen und die Anweisungen zum Knacken von Passwörtern gegeben. Alle Abschnitte werden ausführlich mit Demonstrationsprogrammen illustriert.

Der dritte Teil nimmt auf die Besonderheit des PC-1403 Rücksicht und stellt einen kleinen Exkurs in die Matrizenrechnung dar. Es werden einige theoretische Aspekte aufgezeigt und die Durchführung von Matrizenoperationen am Computer beschrieben.

Nach diesem vielleicht etwas trockenen Abschnitt werden im vierten Teil einige Tricks und Programmiertechniken beschrieben. Unter anderem wird gezeigt, wie man ohne IF-Befehl eleganter programmieren kann, wie man mit den willkürlich scheinenden RND-Zahlen recht genaue Näherungsrechnungen durchführen kann und wie man sortiert.

Im fünften und letzten Teil folgt eine Sammlung verschiedenster BASIC- und Maschinenprogramme für jedermann. Die Palette reicht vom Spiel bis zu wissenschaftlichen Berechnungen.

Ein Literaturverzeichnis und eine Tabelle des internen Codes beschließen das Buch.

1. Grundlagen

1.1 Zubehör

- RD-720H: Recorder für CompactCassetten zum Speichern von Programmen. Mit Bandzählwerk, manueller Schnellauf-funktion und abschaltbarem Lautsprecher.
- CE-126P: ThermoDrucker/Cassetten-Recorder-Interface. Das Interface ist für eine Verbindung zwischen Recorder und Computer notwendig.
Drucker: 24 Zeichen pro Zeile,
Geschwindigkeit: 0.8 Zeilen/sec.
- CE-124: Cassetten-Recorder-Interface
- CE-129P: ThermoDrucker/Cassetten-Recorder-Interface
Selbe Funktionen wie CE-126P, jedoch ohne "Kabelsalat" anschließbar dank kompakter Steckverbindung.

1.2 Zahlensysteme

Bevor es richtig losgeht, müssen wir uns zunächst ein wenig mit Mathematik beschäftigen. Im Alltag benutzt man zur Darstellung von Zahlen das Dezimalsystem (Zahlensystem mit der Basis 10). Die Ziffernkombination 158 zum Beispiel repräsentiert die Zahl $1 \cdot 10^2 + 5 \cdot 10^1 + 8 \cdot 10^0$. Jedoch ist die Zahl 10 als Basis willkürlich gewählt. Jede ganze Zahl ≥ 2 ist hierfür geeignet. 158 lässt sich zum Beispiel auch darstellen durch $1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$. Die Ziffernkombination 10011110 wäre somit die Darstellung der Zahl 158 im Zahlensystem mit 2 als Basis (Binärsystem oder

Dualsystem).

Das Binärsystem ist für die Computertechnik von ernormer Wichtigkeit. In ihm kann nämlich jede Zahl als Kombination von Einsen und Nullen dargestellt werden. Während ein Speicher, der eine dezimale Ziffer darstellen soll, zehn verschiedene, eindeutig voneinander unterscheidbare Zustände haben muß, benötigt ein Speicher für die Darstellung einer binären Ziffer nur zwei Zustände, was wesentlich einfacher zu realisieren ist.

Zur Kurzschreibweise von Binärzahlen wird das Zahlensystem mit der Basis 16 (Hexadezimalsystem oder Sedenzimalsystem) verwendet. Das System benötigt die Zahlen 0 bis 15 als Ziffern. Für die Zahlen 10 bis 15 kennen wir aber keine Ziffern, sondern setzen diese Zahlen im gewohnten Dezimalsystem bereits aus jeweils zwei Ziffern zusammen. Man könnte dafür neue Symbole erfinden, hat sich aber darauf geeinigt, sie durch die Buchstaben A bis F zu ersetzen (A=10, B=11, ... , F=15).

158 läßt sich beispielsweise schreiben als $9*16^1 + 14*16^0$. Die hexadezimale Schreibweise lautet also 9E.

Da $16 = 2^4$ ist, stellt eine hexadezimale Ziffer genau vier binäre Ziffern dar:

| hex. | binär | hex. | binär | hex. | binär | hex. | binär |
|------|-------|------|-------|------|-------|------|-------|
| 0 | 0000 | 4 | 0100 | 8 | 1000 | C | 1100 |
| 1 | 0001 | 5 | 0101 | 9 | 1001 | D | 1101 |
| 2 | 0010 | 6 | 0110 | A | 1010 | E | 1110 |
| 3 | 0011 | 7 | 0111 | B | 1011 | F | 1111 |

Die Hexadezimalzahl 9E entspricht nach dieser Tabelle also der Binärzahl 10011110 (vergleiche mit oben), die Hexadezimalzahl F2 der Binärzahl 11110010 etc.

Man kann beim PC-1403 Zahlen auch in ihrer hexadezimalen Schreib-

weise benutzen, wenn man ein Kaufmanns-Und voranstellt: Gibt man zum Beispiel im RUN-Mode "&9E" direkt ein, so antwortet der Rechner nach Druck auf 'ENTER' mit "158.".

Drückt man im CAL-Mode auf '→HEX', so kann man vollständig und ausschließlich mit Hexadezimalzahlen arbeiten. Das Rückschalten auf dezimale Arithmetik erfolgt mit 'SHIFT' '→HEX'.

Es folgt ein Programm, welches Zahlen von einem beliebigen Zahlensystem (Basis 2 bis 36) in ein beliebiges anderes umrechnet. Je nach gewählter Basis treten die Buchstaben A bis Z auf, um die Zahlen 10 bis 35 als Ziffern darzustellen. Deshalb wurde im Programm das 36er-System gerade als oberste Grenze gewählt.

```
1:"A"
3:"Umwandlung einer Za
hl eines beliebigen
Systems in ein ander
es System
5:CLEAR
7:DIM Z$(0)*60
10:INPUT "EINGABESYSTEM
":;"B
20:IF (B<2) OR (B>36)
    THEN PRINT "NUR 2..3
    6":GOTO 10
25:INPUT "ZAHL:";Z$(0)
30:GOSUB "DEC"
35:D=D+((D- INT D)=.5)
40:INPUT "AUSGABESYSTEM
":;"B
50:IF (B<2) OR (B>36)
    THEN PRINT "NUR 2..3
    6":GOTO 40
60:GOSUB "COD"
70:PRINT Z$(0)
75:IF LEN Z$(0)>48 THEN
    PRINT "UNVOLLSTAENDI
    G !"
77:A$="
80:INPUT "NOCHMAL ? ";A
    "
90:IF A$="J" GOTO 10
100:IF A$<>"N" GOTO 80
110:EHD
120:"DEC":D=0
130:FOR S=LEN Z$(0) TO 1
    STEP -1
140:P$=RIGHT$ (Z$(0),S)
150:IF ASC P$<=ASC "9"
    THEN LET Q=ASC P$-
        ASC "0"
160:IF ASC P$>ASC "A"
    THEN LET Q=ASC P$-
        ASC "A"+10
170:D=D+Q*B^(S-1)
180:NEXT S
190:RETURN

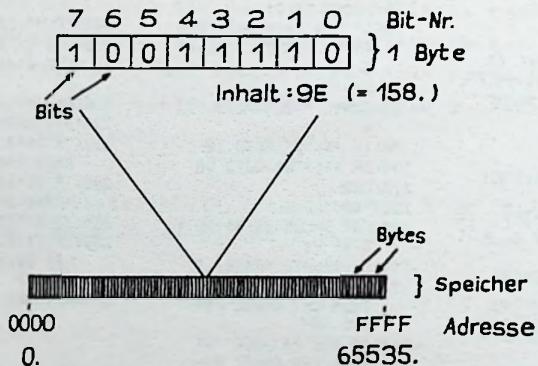
200:"COD":Z$(0)=""
210:FOR S= INT (LOG D)/
    LOG B)+1 TO 0 STEP -
    1
220:Q= INT (D/B^S)
225:D=D-Q*B^S
230:IF Q<=9 THEN LET P$=
    CHR$ (ASC "0"+Q)
240:IF Q>=10 THEN LET P$=
    CHR$ (ASC "A"+Q-10)
250:Z$(0)=Z$(0)+P$
255:IF Z$(0)="0" THEN
    LET Z$(0)=""
260:NEXT S
270:RETURN
```

1.3 Bits, Bytes, Adressen

Mit dem Wissen aus dem vorhergehenden Kapitel können wir nun einige Grundvokabeln aus dem Computer-Chinesisch verstehen und den Grundaufbau des Rechners kennenlernen.

Wie bereits gesagt, kann die kleinste Einheit eines Speichers genau zwei Zustände darstellen und repräsentiert damit eine binäre Ziffer. Auf englisch heißt das "binary digit" und wird mit "Bit" abgekürzt. Acht solcher Bits werden zu einer Speicherzelle, einem Byte, zusammengefügt und von 0 bis 7 durchnummieriert. Man kann somit mit einem Byte die Zahlen 0 bis 255 (= $2^8 - 1$) im Binärsystem darstellen. Diese Zahl nennen wir den Inhalt dieses Bytes.

Die einzelnen Bytes werden durchnummieriert (von 0 bis 65535 (= $2^{16} - 1$)); die entsprechende Nummer wird Adresse genannt. Allerdings muß es nicht unbedingt 65535 verschiedene Speicherzellen geben; einigen Adressen wird kein Byte zugeordnet.



Adressen und Inhalte von Adressen werden im allgemeinen hexadezimal angegeben. Sollte abweichend einmal die dezimale Schreibweise verwendet werden, so wird dieses durch einen angefügten Dezimalenpunkt gekennzeichnet (z.B. ABCD oder 43981.).

1.4 Speichertypen

RAM --- Random Access Memory (Speicher mit frei wählbarem Zugriff)

Die in einem Speicher dieses Typs gespeicherten Informationen können sowohl gelesen als auch verändert werden (Schreib-Lese-Speicher). Deshalb werden BASIC-Programme, Variablen etc. hier gespeichert.

Wird bei einem RAM die Versorgungsspannung abgeschaltet, so geht der Inhalt des Speichers verloren. Deshalb fließt beim PC-1403 nach dem Ausschalten immer noch ein kleiner Strom, um Programme, Daten etc. zu erhalten.

ROM --- Read Only Memory (Nur-Lese-Speicher)

Der Inhalt dieses Speichers kann nur gelesen, aber nicht verändert werden. Er ist gewissermaßen fest "eingebrannt" und bleibt auch im stromlosen Zustand erhalten.

Im ROM sind dementsprechend unveränderliche Informationen gespeichert, wie z.B. Maschinenunterprogramme, das Betriebssystem, die LCD-Muster für die einzelnen Schriftzeichen und das Maschinenprogramm, welches den BASIC-Programmtext im RAM in die dem Computer einzige verständliche Maschinensprache übersetzt. Ein solches Programm wird Interpreter genannt und nimmt im allgemeinen den größten Teil des ROMs in Anspruch.

1.5 POKE und PEEK

Zum Verändern bzw. zum Lesen der Inhalte von Bytes dient in BASIC der POKE-Befehl und die PEEK-Funktion, die leider beide in der Bedienungsanleitung des Rechners unterschlagen wurden.

Der Befehl "POKE" (engl.: stechen, stoßen, stochern) dient zum Verändern des Inhalts eines Bytes und hat die allgemeine Syntax

POKE Adresse,inhalt

Nach Ausführung dieses Befehls steht der Wert inhalt in der Adresse adresse. Dieses funktioniert allerdings natürlich nur, wenn die Adresse im RAM-Bereich liegt; andernfalls passiert nichts.

Beispiele für die Benutzung sind:

```
POKE 32768,253
POKE A,&E4
POKE C(D),E+1
```

Will man den Inhalt mehrerer, zusammenhängender Adressen ändern, kann man dieses auch mit nur einem Befehl erledigen:

```
POKE 50000,24,53,37
```

zum Beispiel schreibt die Zahl 24. in die Adresse 50000., die Zahl 53. in die Adresse 50001. usw.

Beim Umgang mit dem POKE-Befehl ist übrigens Vorsicht geboten. Verändert man nämlich den Inhalt bestimmter Systemadressen, so kann dieses zum Absturz des Rechners führen, d.h. man kann ihn solange nicht mehr bedienen (und auch nicht abschalten !!), bis die Taste 'ALL RESET' auf der Geräterückseite gedrückt worden ist.

Die Funktion "PEEK" (engl.: nachschauen) dient dem Lesen der Inhalte von Bytes. Der allgemeine Ausdruck

```
PEEK adresse
```

repräsentiert den Inhalt der Adresse adresse. Leider kann man große Teile des ROMs auf diese Weise nicht einlesen; im allgemeinen wird die Funktion aber auch nur im RAM-Bereich angewendet.

Beispiele für die Anwendung der Funktion sind:

```
A=PEEK 12345
IF PEEK B=&5A THEN LET C$=STR$(PEEK C)
FOR I=PEEK F(1) TO PEEK F(2)
```

Will man den Inhalt einer Adresse um 1 erhöhen, muß folgender Befehl ausgeführt werden:

POKE adresse, PEEK adresse + 1

Entsprechend der im Kapitel 1.3 geschilderten Grenzen müssen folgende Bereiche eingehalten werden:

0 ≤ adresse ≤ 65535.

0 ≤ inhalt ≤ 255.

Andernfalls wird der Versuch mit einer "ERROR 3"-Meldung quittiert.

1.6 OR und AND

"OR" und "AND" werden im allgemeinen zum Verknüpfen von logischen Ausdrücken bei "IF"-Befehlen verwendet (z.B. "IF A=0 AND B=1 THEN ...").

Doch kann man damit auch Zahlen (kleiner als 32768) in ihrer binären Darstellung bitweise miteinander verknüpfen.

Jedes Bit von A OR B ist genau dann gleich 1 gesetzt, wenn das entsprechende Bit von A oder das entsprechende Bit von B gleich 1 gesetzt ist (oder auch beide).

Jedes Bit von A AND B ist genau dann gleich 1 gesetzt, wenn das entsprechende Bit von A und das entsprechende Bit von B gleich 1 gesetzt ist.

Beispiel:

Sei A = 240 und B = 204.

| | | |
|----|----------|--------|
| A: | 11110000 | = 240. |
| B: | 11001100 | = 204. |

A OR B: 11111100 = 252.

Also: 240 OR 204 = 252

| | | |
|----|----------|--------|
| A: | 11110000 | = 240. |
| B: | 11001100 | = 204. |

A AND B: 11000000 = 192.

Also: 240 AND 204 = 192

Will man ermitteln, wie in einer Zahl A das n-te bit gesetzt ist, muß man die Verknüpfung "A AND 2^n " durchführen.

Ist das Ergebnis Null, so ist das Bit gleich Null, ist das Ergebnis 2^n , so ist es gleich Eins gesetzt.

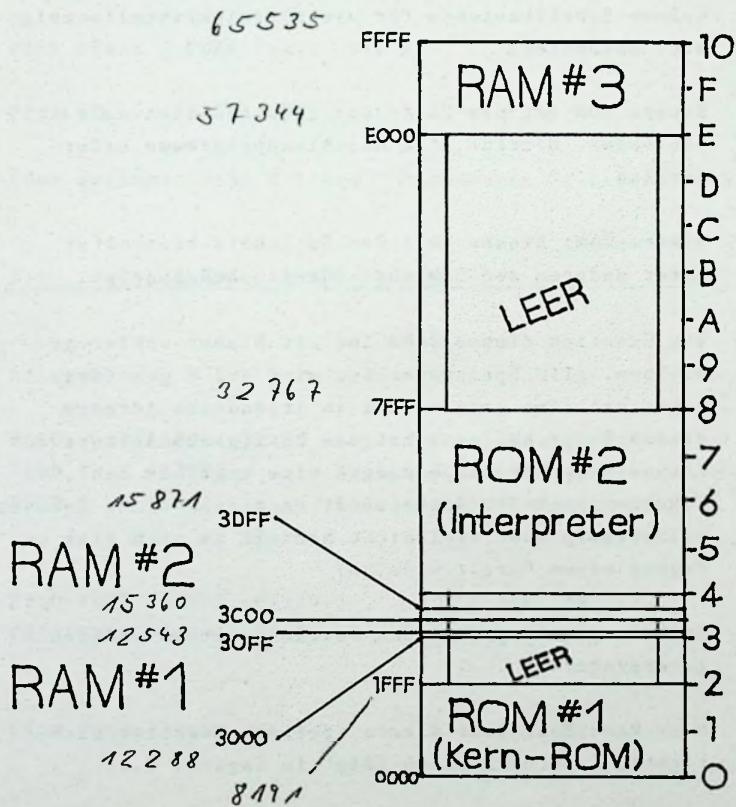
Dieses Verfahren wird im folgenden Programm benutzt, um eine Dezimalzahl binär darzustellen. Aufgrund der Rechenungenauigkeit ist die Ausrechnung der Zweierpotenzen durch " 2^N " ungeeignet. Man muß sie vielmehr durch fortgesetztes Dividieren der höchsten Potenz durch 2 bestimmen.

```

1;"A"
3;"Umwandlung von Dezi-
    malzahlen in Binaerz
    ahlen
5:CLEAR
7:DIM B$(0)*16
10:INPUT "DEZ.-ZAHL: ";
    D
20:IF ABS D>32767 GOTO
    10
25:B$(0)="";P=16384
30:FOR N=14 TO 0 STEP -
    1
40:B$(0)=LEFT$ (B$(0),
    LEN B$(0)-(B$(0)="0")
    ))>CHR$ (48+((D AND
    P)<>0))
45:P=P/2
50:NEXT N
60:PRINT B$(0)
70:INPUT "NOCHMAL ? ";Q
    $
80:IF Q$="J" GOTO 10
90:IF Q$<>"N" GOTO 70
100:END

```

2.1 Speicherplan



2.2 Funktion der einzelnen Speicher

ROM #1 --- Kern-ROM: Hierin sind die initialisierenden Routinen ("Kaltstart", "Warmstart"), die Bitmuster der einzelnen Schriftzeichen für die Flüssigkristallanzeige etc. enthalten.

Dieses ROM ist mit Hilfe der PEEK-Funktion nicht einlesbar. Hierfür sind Maschinenprogramme erforderlich.

RAM #1 --- System-RAM: Dieser Teil des Speichers beinhaltet unter anderem den Speicher für die LCD-Anzeige.

RAM #2 --- Die Funktion dieses RAMs ist mir bisher unklar geblieben. Alle Speicherzellen sind auf 8 gesetzt; poket man eine gerade Zahl in irgendeine Adresse dieses Bereichs, so führt das häufig zum Absturz des Rechners; poket man hingegen eine ungerade Zahl, so schaltet sich der Rechner oft nach rund einer Sekunde selbsttätig aus. Vielleicht handelt es sich hier um irgend einen Port.

ROM #2 --- Enthält den Hauptteil des Betriebssystems und den Interpreter.

RAM #3 --- User-RAM: Mehr über diesen, für den Benutzer wichtigsten Speicherbereich folgt im Kapitel 2.4.

2.3 Periodizitäten von Adressen

Oft findet man, daß sich ganze Speicherbereiche periodisch wiederholen, d.h. verschiedene Adressen mit gleichem Abstand weisen auf ein und dieselbe Speicherzelle. Dieses ist hardwarebedingt und liegt an einer ungenügenden Adreßdekodierung (die aber nicht stört, da die "überflüssigen" Adressen sonst auf leere Speicher-

bereiche weisen würden).

Beim PC-1403 finden sich folgende Periodizitäten:

PEEK &30xx ≡ PEEK &31xx

PEEK &Exxx ≡ PEEK &nxxx mit n = 8, A, C

PEEK &Fxxx ≡ PEEK &nxxx mit n = 9, B, D

(Das mathematische Zeichen " \equiv " bedeutet "ist identisch mit".)

2.4 Grundaufbau des User-RAMs

Adresse

| | | |
|------|----------|-----------------------------------------|
| E000 | (57344.) | Beginn des RAMs mit Hilfspointern |
| E030 | (57392.) | Beginn des BASIC-Speichers (6878 Bytes) |
| FBOF | (64271.) | Ende des BASIC-Speichers |

Variablen:

| | | |
|-----------|-----------------|------------|
| FB10-FB17 | (64272.-64279.) | Z bzw. Z\$ |
| FB18-FB1F | (64280.-64287.) | Y bzw. Y\$ |
| ... | ... | ... |
| ... | ... | ... |
| FBD8-FBDF | (64472.-64479.) | A bzw. A\$ |

| | | |
|------|----------|----------------------------------------------------------------------------|
| FBE0 | (64480.) | Beginn des zweiten Teils des System-RAMs mit Pointern und Passwort etc. |
| FFFF | (65535.) | Ende des RAMs |

Formel zur Berechnung der Anfangsadresse der Variable
buchstabe bzw. buchstabe\$:

64992-8*ASC"buchstabe"

Beispiel:

F bzw. F\$ beginnt bei der Adresse 64992-8*ASC"F" = 64432 (dezimal).

2.5 Speicherformat von BASIC-Programmen

Als nächstes wollen wir uns mit dem Format beschäftigen, in dem ein Programm im RAM gespeichert wird. Dazu wird folgendes Programm eingegeben, welches sich gewissermaßen selbst untersucht:

```
10:A=&E030
20:PRINT HEX(A),HEX(PEEK A)
30:A=A+1
40:GOTO 20
```

Das Programm listet den Inhalt der Adressen ab E030, also gerade den Adreßbereich, den das Programm selbst belegt.

| Adresse | Inhalt | Bedeutung |
|---------|--------|--------------------------------------------------------|
| E030 | FF | Beginn des Programms |
| E031 | 00 | Zeilennummer 10 (1. Byte) |
| E032 | 0A | Zeilennummer 10 (2. Byte) |
| E033 | 08 | Zeilenlänge in Bytes einschließlich Carriage Return |
| E034 | 41 | A |
| E035 | 3D | = |
| E036 | 26 | & |
| E037 | 45 | E |
| E038 | 30 | 0 |
| E039 | 33 | 3 |
| E03A | 30 | 0 |
| E03B | 0D | Carriage Return (Zeilenende) |
| E03C | 00 | Zeilennummer 20 (1. Byte) |
| E03D | 14 | Zeilennummer 20 (2. Byte) |
| ... | ... | ... |
| ... | ... | ... |
| E054 | 00 | Zeilennummer 40 (1. Byte) |
| E055 | 28 | Zeilennummer 40 (2. Byte) |

| Adresse | Inhalt | Bedeutung |
|---------|--------|--------------------------------------|
| E056 | 04 | Zeilenlänge einschl. Carriage Return |
| E057 | C6 | GOTO |
| E058 | 32 | 2 |
| E059 | 30 | 0 |
| E05A | 0D | Carriage Return |
| E05B | FF | Ende des Programms |

Jedem Schriftzeichen und Befehlswort wird also genau eine bestimmte Zahl zwischen 32 und 255 (hexadezimal: 20 und FF) zugeordnet, die ein Byte des Speichers füllt.

Dieser Code ist im Anhang A.2 zusammengestellt und stellt eine Erweiterung des ASCII dar (American Standard Code for Information Interchange).

Für die Kennung der Zeilennummer werden zwei Bytes in Anspruch genommen: Zur Darstellung der Zeilennummer n steht im ersten Byte die Zahl INT($n/256$) und im zweiten die Zahl $n-INT(n/256)*256$ ($= n \bmod 256$). Man sieht also unmittelbar, daß es vom Speicher-aufwand her günstiger ist, zwei Befehle (durch einen Doppelpunkt getrennt) in eine Zeile unterzubringen, statt sie in zwei getrennte Zeilen zu schreiben.

Ist eine Zeile zu Ende, so wird das mit einer 13. gekennzeichnet ($= \text{hex } OD$). Dieser Code entspricht damit genau dem ASCII-Zeichen für das Ende einer Eingabe (Carriage Return).

Der Anfang des Programms bei der Adresse E030 und sein Ende werden hingegen mit einer 255. ($= \text{hex } FF$) markiert. Wird das Programm gelöscht, so wird einfach nur die Programmende-Markierung direkt hinter den Anfang gelegt (Adresse E031), während das Programm selbst erhalten bleibt.

Gibt man also wieder das Kommando

POKE &E031, INT($n/256$)



ein, wobei n die erste Zeilennummer des Programms ist, so kann man das Programm wieder auflisten und ausführen, jedoch nicht mehr editieren. Dazu ist eine Zusatzmaßnahme erforderlich, die im folgenden Kapitel behandelt wird.

Ist die erste Zeilennummer des Programms, wie im allgemeinen der Fall, kleiner als 256, so vereinfacht sich das obenstehende Kommando zu

POKE &E031,0

2.6 Rettung gelöschter Programme

Die Funktion "MEM" gibt die Anzahl der noch freien Bytes an. Der Computer speichert die Adressen des Programmanfangs und des -endes und kann so den freien Speicher durch Subtraktion beider Adresse voneinander und Berechnung der Differenz zu der Gesamtzahl der verfügbaren Bytes bestimmen.

Bytes, die als Inhalt die Adressen anderer Bytes enthalten, werden Zeiger oder Pointer genannt. Da eine Adresse vier hexadezimale Ziffern, also 16 Bit, lang ist; ein Byte aber nur acht Bit speichern kann, werden zwei Bytes benötigt, um eine Adresse zu speichern. Insgesamt werden also vier Bytes in Anspruch genommen, nämlich FF01 - FF04 (65281. - 65284.).

Zeiger auf Programmanfang:

| Adresse | Inhalt | |
|---------|------------------|----------------|
| FF01 | 30 (48.) | (Low Byte) |
| FF02 | <u>E0 (224.)</u> | (High Byte) |
| | = E030 | (Startadresse) |

Die Endadresse ist dementsprechend in den Adressen FF03 und FF04 gespeichert. Ist das Programm beispielsweise bei der Adresse

F253 zu Ende, so sieht das so aus:

| Adresse | Inhalt | |
|---------|------------------|--------------|
| FF02 | 53 (83.) | (Low Byte) |
| FF03 | <u>F2 (242.)</u> | (High Byte) |
| | = F253 | (Endadresse) |

Will man ein gelösches Programm vollständig wiederherstellen (eine solche Prozedur wird oft als RENEW bezeichnet), so muß man auch diesen Zeiger auf das alte Programmende zurückstellen, da er beim Löschen auf die Adresse direkt hinter dem Programm-anfang gestellt worden ist. Dazu sollte man sich für jedes Programm den freien Speicher ("MEM") bei gelöschten Feldern ("CLEAR") notieren. Sei diese Zahl mit speicher bezeichnet und die erste Zeilennummer des Programms mit zeilennummer, so müssen folgende Kommandos ausgeführt werden (direkt in den Rechner einzugeben), um das Programm wieder vollständig zu restaurieren:

```
A=&FBOF-speicher  
POKE &E031,INT(zeilennummer/256)  
POKE &FF03,A-INT(A/256)*256,INT(A/256)
```

2.7 RENEW als Maschinenprogramm

Kennt man den freien Speicher des gelöschten Programms nicht, so läßt es sich trotzdem retten. Man kann es durch Eingeben und Ausführen des folgenden Maschinenprogramms wiederherstellen, welches die alte alte Programmende-Markierung selbstständig sucht und die gefundene Adresse in den Pointer einträgt.

Das Programm ist /2/ entnommen und für den PC-1403 umgeschrieben worden.

Assemblerlisting

| | | | |
|------|----------|------------|-------------------------------------|
| FB10 | 12 04 | LIP 04 | ;Adresse vom Programmanfang in das |
| FB12 | 10 FF 01 | LIDP FF01 | ;X-Register, Adresse vom -ende ins |
| FB15 | 00 03 | LII 03 | ;Y-Register bringen. |
| FB17 | 18 | MVWD | |
| FB18 | 07 | DY | ;Y=Y-1 |
| FB19 | 02 00 | LIA 00 | ;Null hinter ProgAnfang bringen |
| FB1B | 26 | IYS | |
| FB1C | 24 | IXL | ;Nach einer 13 suchen (Zeilenende) |
| FB1D | 67 0D | CPIA 0D | ;Eine 13 gefunden ? |
| FB1F | 29 04 | JRNZM FB1C | ;Nein: Weitersuchen |
| FB21 | 24 | IXL | ;Ja: Ist nachfolgendes Byte = 255 ? |
| FB22 | 67 FF | CPIA FF | |
| FB24 | 29 09 | JRNZM FB1C | ;Nein: Weitersuchen |
| FB26 | 10 FF 03 | LIDP FF03 | ;Ja: Programmende-Pointer auf die |
| FB29 | 12 04 | LIP 04 | ;gefundene Adresse stellen |
| FB2B | 00 01 | LII 01 | |
| FB2D | 19 | EXWD | |
| FB2E | 37 | RTN | ;Return |

Eingabe und Ausführung des Maschinenprogramms erfolgt mit den beiden folgenden, direkt in den Rechner einzugebenden Kommandos:

```
POKE &FB10,18,4,16,255,1,0,3,24,7,2,0,38,36,103,13,41  
POKE &FB20,4,36,103,255,41,8,16,255,3,18,4,0,1,25,55  
CALL &FB10
```

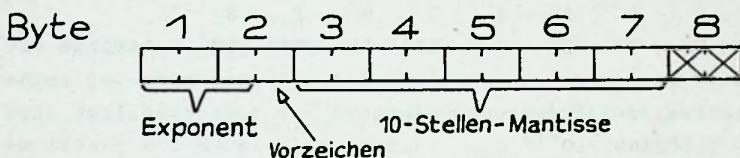
Es ist wichtig, das Maschinenprogramm wirklich nur dann auszuführen, wenn das BASIC-Programm tatsächlich mit "NEW" oder Druck auf 'ALL RESET' gelöscht worden ist, da ansonsten der Rechner abstürzen kann. Das bedeutet, daß er solange nicht mehr bedient werden kann (und auch nicht abgeschaltet werden kann !!!), bis die Taste 'ALL RESET' auf der Geräterückseite gedrückt wird.

2.8 Die Standardvariablen

Wie aus dem Speicheraufbau zu erkennen ist, besteht jede Variable aus acht Bytes. Jede der 26 Standardvariablen kann entweder als numerische Variable (z.B. E) oder Stringvariable (z.B. E\$) verwendet werden; andernfalls gibt es an der entsprechenden Stelle im Programm einen ERROR 9.

2.8.1 Format von numerischen Variablen

Bei der Speicherung einer Zahl wird folgendes Schema angewendet:



Es wird das sogenannte BCD-Verfahren angewendet (Binary Coded Decimal), d.h. jedes Halbbyte (Vier-Bit-Gruppe) repräsentiert genau eine dezimale Ziffer. Die zehn Ziffern der Mantisse finden in den fünf Bytes 3 bis 7 Platz. Das Vorzeichen befindet sich im niederwertigen Halbbyte (auch Nibble genannt) von Byte 2: Eine Null steht für eine nichtnegative, eine Acht für eine negative Zahl.

Der Exponent schließlich findet sich in Byte 1 und im höherwertigen Nibble von Byte 2 wieder, besteht also aus drei dezimalen Ziffern, die im folgenden Verfahren codiert werden.

Exponent

$$000 = 10^0$$

$$001 = 10^1$$

$$002 = 10^2$$

...

...

Exponent

$$999 = 10^{-1}$$

$$998 = 10^{-2}$$

$$997 = 10^{-3}$$

...

...

Damit lässt sich jede Zahl eindeutig beschreiben. Das letzte Byte dient nur während Berechnungen und wird zur Zahlendarstellung nicht herangezogen.

Es folgen einige Beispiele für die Speicherung von Zahlen:

(i) Darstellung der Ludolphschen Zahl

$$\pi = 3.141592654$$

| Byte | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|--------|----|----|----|----|----|----|----|----|
| Inhalt | 00 | 00 | 31 | 41 | 59 | 26 | 54 | 00 |

(ii) Darstellung der Vakuumlichtgeschwindigkeit

$$c_0 = 2.99792458 \cdot 10^8 \text{ m/s}$$

| Byte | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|--------|----|----|----|----|----|----|----|----|
| Inhalt | 00 | 80 | 29 | 97 | 92 | 45 | 80 | 00 |

(iii) Darstellung der Elektronenladung

$$e = -1.6021892 \cdot 10^{-19} \text{ C}$$

| Byte | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|--------|----|----|----|----|----|----|----|----|
| Inhalt | 98 | 18 | 16 | 02 | 18 | 92 | 00 | 00 |

(iv) Darstellung der magnetischen Feldkonstanten

$$\mu_0 = 1.256637061 \cdot 10^{-6} \text{ H/m}$$

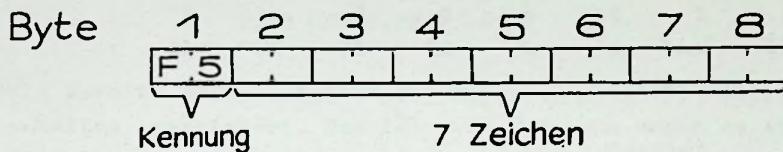
| Byte | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|--------|----|----|----|----|----|----|----|----|
| Inhalt | 99 | 40 | 12 | 56 | 63 | 70 | 61 | 00 |

Mit dem folgenden Programm kann jeder solche Beispiele selbst durchexerzierien. Es stellt den Inhalt der acht Bytes von Z bzw. Z\$ hexadezimal dar:

```
10:FOR I=1 TO 8
20:PRINT "BYTE ";STR$ I;" : ";HEX(PEEK(I+&FBOF))
30:NEXT I
```

2.8.2 Format von Stringvariablen

Wird eine Variable als Stringvariable (Textvariable) deklariert,
so wird sie folgendermaßen im Speicher formatiert:



Das erste Byte hat immer den Inhalt F5 (245.), um die Variable als Stringvariable kenntlich zu machen und sie von numerischen Variablen zu unterscheiden.

Die maximal sieben Zeichen füllen die sieben Bytes 2 bis 8, wobei derselbe Code benutzt wird, den wir schon im Kapitel 2.5 beim Speicherformat von Programmen kennengelernt haben und der im Anhang A.2 aufgeführt wird.

Enthält die Zeichenkette weniger als sieben Zeichen, so taucht ein Byte mit dem Inhalt Null als Schlußkennung auf.

Beispiel: Die Variable enthält die Zeichenkette "(Text)":

| | | | | | | | | |
|----------------|----|----|----|----|----|----|----|----|
| Byte | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Inhalt | F5 | 28 | 54 | 65 | 78 | 74 | 29 | 00 |
| Zeichen | (| T | e | x | t |) | | |

2.9 Der interne Code

Wir können nun sehr leicht eine Tabelle des internen Codes anzeigen bzw. ausdrucken lassen.

Wir deklarieren zunächst eine Variable als Stringvariable (hier Z\$), "poken" in einer FOR...NEXT-Schleife alle Zahlen des Codes (32 bis 255) nacheinander in das erste Byte der Zeichenkette und zeigen Z\$ dann an:

| Adresse | FB10 | FB11 | FB12 | FB13 | FB14 | FB15 | FB16 | FB17 |
|----------|------|-------|------|------|------|------|------|------|
| Byte | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Inhalt | F5 | Lauf- | 20 | 20 | 20 | 20 | 20 | 20 |
| variable | | | | | | | | |

Programm:

```
10:Z$=""      "
20:FOR I=32 TO 255
30:POKE &FB11,I
40:PRINT I;"    ";Z$
50:NEXT I
```

2.10 Das Passwort

Das Passwort ist beim PC-1403 in ähnlicher Weise gespeichert wie eine Stringvariable: Die maximal sieben Zeichen finden in den Adressen FFOA bis FF10 (65290. bis 65296.) Platz; bei weniger als sieben Zeichen taucht wieder eine Null als Schlußmarkierung auf.

Zusätzlich gibt es noch ein Byte (Adresse: FF14 (65300.)), welches anzeigt, ob das Passwort "scharf" ist oder nicht. Das Passwort bleibt nämlich auch dann gespeichert, wenn der Schutz aufgehoben worden ist und geht erst beim Einsatz eines neuen Passworts verloren.

Das Passwort ist dann und nur dann aktiv, wenn Bit 5 von FF14 gesetzt ist.

Dementsprechend erhält man folgende POKE-Formeln, mit denen man das Passwort ein- bzw. abschalten kann.

(i) Abschalten des Passwortschutzes:

POKE 65300,PEEK 65300 - 32

(ii) Einschalten des Passwortschutzes:

POKE 65300,PEEK 65300 + 32

Wie bereits gesagt, bleibt das Passwort auch nach seinem Abschalten gespeichert. Das folgende Programm macht es sichtbar, indem es die sieben Bytes in die Variable Z\$ überträgt:

```
5:POKE &FB10,245
10:FOR I=0 TO 6
20:POKE &FB11+I,PEEK (&FFOA+I)
30:NEXT I
40:PRINT Z$
```

2.11 Direktansteuerung der Flüssigkristallanzeige

Die Anzeige des PC-1403 besteht aus 24 Elementen, von denen jedes einzeln eine Matrix aus $5 \times 7 = 35$ Flüssigkristallpunkten darstellt. Durch Kombinieren dieser Punkte werden die jeweils darzustellenden Zeichen formiert.

Im folgenden soll beschrieben werden, wie man diese 840 Punkte einzeln ansteuern kann, um z.B. eigene Zeichen, Umlaute, Figuren etc. anzuzeigen.

Jedem Punkt der Anzeige ist ein Bit im Speicher des Rechners zugeordnet: Der jeweilige Punkt erscheint genau dann in der Anzeige, wenn das entsprechende Bit gesetzt ist. Die sieben Bits der sieben Punkte einer Spalte der Anzeige werden zu einem Byte zusammengefügt, wobei das überflüssige, höchstwertige Bit (Bit 7) gelöscht wird. Der n-te Punkt der Spalte (von oben nach unten gezählt) entspricht dann dem Bit (n-1) des jeweiligen Bytes.

Die einzelnen Bytes der einzelnen LCD-Spalten werden wie folgt im RAM #1 abgelegt:

| Element-Nr. | Adr. der linken Spalte | Adr. der rechten Spalte | links* | rechts* |
|-------------|------------------------|-------------------------|--------|---------|
| 1 | 3000 | 3004 | | + |
| 2 | 3005 | 3009 | | + |
| 3 | 300A | 300E | | + |
| 4 | 300F | 3013 | | + |
| 5 | 3014 | 3018 | | + |
| 6 | 3019 | 301D | | + |
| 7 | 302D | 3031 | | + |
| 8 | 3032 | 3036 | | + |
| 9 | 3037 | 303B | | + |
| 10 | 301E | 3022 | | + |
| 11 | 3023 | 3027 | | + |
| 12 | 3028 | 302C | | + |
| 13 | 306C | 3068 | | + |
| 14 | 3067 | 3063 | | + |
| 15 | 3062 | 305E | | + |
| 16 | 307B | 3077 | | + |
| 17 | 3076 | 3072 | | + |
| 18 | 3071 | 306D | | + |
| 19 | 305D | 3059 | | + |
| 20 | 3058 | 3054 | | + |
| 21 | 3053 | 304F | | + |
| 22 | 304E | 304A | | + |
| 23 | 3049 | 3045 | | + |
| 24 | 3044 | 3040 | | + |

Wie man sieht, erfolgt die Adressierung der einzelnen Spalten auf eine sehr komplizierte Art und Weise, wodurch die Programmierung von Spielen, die die Anzeige direkt ansteuern, erschwert. Was die Firma SHARP dazu bewog, die Adressierung derart chaotisch und anwenderfeindlich durchzuführen, bleibt unklar.

Zu der Fußnote *: "links" bedeutet, daß die Spalten nach links in aufsteigender Reihenfolge durchadressiert werden; "rechts" hat eine analoge Bedeutung.

Vor Benutzung der Anzeige muß man diese allerdings erst einschalten, indem man das niederwertigste Bit (Bit 0) des so genannten Control Ports setzt (siehe auch 2.13).

Dieses geschieht durch die folgende kurze Maschinensprach-Routine. Sie ist relokativ und läuft daher ohne Änderung in jedem beliebigen Adreßbereich im RAM.

Assemblerlisting

```
FB00 12 5F      LIP 5F          ;P-Reg zeigt auf CPU-RAM-Adr 5F
FB02 61 01      ORIM 01        ;Setze Bit 0 von (5F)
FB04 DF         OUTC           ;Schicke (5F) zum Control Port
FB05 37         RTN            ;Return
```

Das Einschalten der Anzeige erfolgt somit durch die folgenden Befehle:

```
POKE &FB00,18,95,97,1,223,55
CALL &FB00
```

Das Abschalten der Anzeige erfolgt ganz analog durch Löschen von Bit 0 des Control Ports. Dieses erfolgt durch folgende, ebenfalls relokative Routine:

Assemblerlisting

```
FB06 12 5F      LIP 5F          ;P-Reg zeigt auf CPU-RAM-Adr 5F
FB08 60 FE      ANIM FE        ;Lösche Bit 0 von (5F)
FB0A DF         OUTC           ;Schicke (5F) zum Control Port
FB0B 37         RTN            ;Return
```

Das Abschalten der Anzeige erfolgt somit durch die Befehle:

```
POKE &FB06,18,95,96,254,223,55
CALL &FB06
```

Das folgende kurze Programm verdeutlicht noch einmal die Vorgehensweise: Die Zeichen "Ä", "£" (Pfund) und "Ω" (Ohm) werden in die Anzeige gebracht:

| | | | |
|---------|---------------------|---------------------|---------------------|
| Inhalt | 79 14 12 14 79 | 68 7E 29 41 42 | 4E 71 01 71 4E |
| Bit-Nr. | 0 1 2 3 4 5 6 7 8 9 | 0 1 2 3 4 5 6 7 8 9 | 0 1 2 3 4 5 6 7 8 9 |
| Adresse | 3000 01 02 03 04 | 05 06 07 08 09 | 0A 0B 0C 0D 0E |
| | 65 43 21 0 | | |

```

Programm:      5:POKE &FB00,18,95,97,1,223,55,18,95,
                96,254,223,55
                10:WAIT 0:PRINT ""
                20:POKE &3000,&79,&14,&12,&14,&79
                30:POKE &3005,&68,&7E,&29,&41,&42
                40:POKE &300A,&4E,&71,&01,&71,&4E
                50:CALL &FB00
                60:FOR I=0 TO 50:NEXT I
                70:CALL &FB06
                80:FOR I=0 TO 50:NEXT I
                90:GOTO 50

```

In Zeile 5 werden die beiden obenstehenden Routinen in den Rechner geladen, in Zeile 10 wird die Anzeige gelöscht und die Zeilen 20 bis 40 poken die Bitmuster der darzustellenden Zeichen in den Display-Speicher. Der Abschnitt ab Zeile 50 läuft in einer Endlosschleife und lässt die Anzeige durch abwechselndes Aufrufen der beiden Routinen blinken.

Leider benutzt der Rechner den Speicherbereich des am weitesten rechts stehenden LCD-Elements als Zwischenspeicher während Berechnungen. Es kommt also zu einigen "Schmutzeffekten" am rechten Anzeigenrand, wenn Berechnungen durchgeführt werden.

Diese Effekte sind umso stärker, je komplizierter die Berechnungen sind. Man füge zum Beispiel in das obenstehende Demonstrationsprogramm ein

55:Y=SIN X

Man erkennt, daß nur die letzte Spalte betroffen ist, während z.B. bei

55:Y=HSN HCS HTN SIN COS TAN X

das gesamte 24. Element in Mitleidenschaft gezogen wird. Man muß diese Effekte in Kauf nehmen und kann sich vielleicht darüber hinwegtrösten, daß beim PC-1401 die gesamte linke Hälfte der Anzeige wegen solche Effekte nicht benutzbar ist.

2.12 Speicherung der Rechnermodus-Anzeige

Die Anzeigen der verschiedenen Rechnermodi sind in dem Display entweder als LCD-Schriftzüge (DEF, SHIFT, HYP etc.) oder als LCD-Striche, die am Anzeigenrand beschriftet sind (CAL, RUN, PRO etc.) vertreten.

Jedes dieser Anzeigenelemente wird durch ein Bit im RAM #1 gesteuert; es erscheint genau dann, wenn das entsprechende Bit gesetzt ist.

Die Speicherung erfolgt in den Adressen 303C, 303D und 307C, und zwar nach folgendem Schema:

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---|--------------|--------------|--------|--------|---------|-----|------|
| Adr | | | | | | | | |
| (303C) | - | ohne Bez. | ohne Bez. | MATRIX | STAT | SML σ | ♪' | ♪♪ σ |
| (303D) | - | CAL | RUNσ | PROσ | HYPσ | SHIFT σ | DEF | BUSY |
| (307C) | - | PRINTσ | DE ♀ | G ♀ | RAD σ♀ | () | M | E |

σ Beim Setzen dieser Bits erscheint nicht nur die Modusanzeige, sondern es wird auch in den entsprechenden Modus geschaltet.

Das ♪-Zeichen bedeutet, daß der Rechner auf den japanischen Zeichensatz umgeschaltet hat. Diese Zeichen (von denen jedes durch zwei Bytes, nämlich FE und ein Codebyte dargestellt wird) werden jedoch in der Anzeige nicht dargestellt. Hierfür ist ein Drucker erforderlich.

♀ Durch Setzen von jeweils zwei Bit gleichzeitig (DE+G oder G+RAD) wird in den entsprechenden Winkelmodus geschaltet (DEG oder GRAD). Bei den sinnlosen Kombination DE und G gilt der DEG-Modus (ASN 1 = 90), bei DEGRAD der GRAD-Modus (ASN 1 = 100) und bei DE RAD der RAD-Modus (ASN 1 = $\pi/2$).

2.13 Der Piezo-Summer

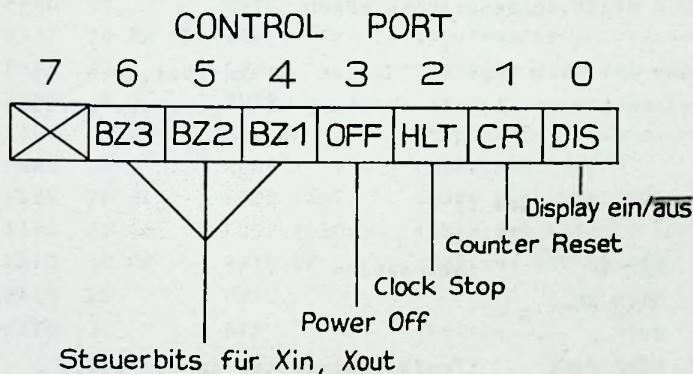
Leider läßt sich der Piezo-Summer des Rechners durch BASIC-Programme nur zu einem monotonen Piepen bewegen (BEEP-Befehl). Befehle, um mit ihm Melodien spielen zu können, gibt es im BASIC-Sprachschatz des PC-1403 wie auch seiner SHARP-Kollegen leider nicht.

Um dem abzuhelpfen, muß man auf die Hilfe kleiner Maschinenpro-

gramme zurückgreifen, mit denen der Summer gesteuert wird. Es würde den Rahmen dieses Buches sicherlich sprengen, die Maschinensprache des Rechners zu beschreiben; ich kann an dieser Stelle nur auf /3/ und /16/ verweisen.

Will man aber mit dem Summer etwas anfangen, so kommt man an solchen Programmen leider nicht vorbei. Vielleicht kommt dadurch auch der ein oder andere Leser auf den Geschmack, sich näher mit der Maschinensprache seines Rechners zu beschäftigen.

Die Steuerung zahlreicher Funktionen des Rechners erfolgt durch ein Byte in seiner CPU (Central Processing Unit), dem sogenannten Control Port:



Für uns interessant sind an dieser Stelle die Bits 4, 5, 6 (BZ1, BZ2, BZ3 genannt), mit denen unter anderem der Kanal Xout gesteuert wird, an dem über einen Transistor der Summer angeschlossen ist.

Die drei Bits können $2^3 = 8$ verschiedene Kombinationen aus Einsen und Nullen haben, von denen folgende für uns bedeutsam sind:

| | BZ3 | BZ2 | BZ1 | Bedeutung |
|-------|-----|-----|-----|-------------------------|
| (i) | 0 | 0 | 0 | Xout=0 (keine Spannung) |
| (ii) | 0 | 0 | 1 | Xout=1 (Spannung) |
| (iii) | 0 | 1 | 0 | 2 kHz an Xout |
| (iv) | 0 | 1 | 1 | 4 kHz an Xout |

Die erste Routine, die hier vorgestellt werden soll, nutzt die dritte Kombination aus: Dem Summer soll eine bestimmte Zeit lang ein 2 kHz-Ton entlockt werden; die Tondauer wird durch einen Parameter bestimmt (Zahl zwischen 0 und 255), der vor dem Maschinenprogrammaufruf in die Adresse FACF "geparkt" wird. Je höher diese Zahl ist, desto länger dauert der Ton. Der Aufruf des Programms erfolgt also mit dem Befehl

```
POKE &FACF,tondauer:CALL &FADO
```

wenn das Programm bei FADO beginnt. Es ist relokativ, d.h. es läuft auch in einem anderen Adressbereichen, ohne geändert werden zu müssen.

Assemblerlisting von Programm 1:

```
FADO 12 5F      LIP 5F          ;BZ2 setzen
FAD2 61 20      ORIM 20
FAD4 DF          OUTC
FAD5 10 FA CF    LIDP FACF     ;Tondauerparameter in den Stack
FAD8 57          LDD           ;bringen (für äußeren Loop)
FAD9 34          PUSH
FADA 02 FF       LIA FF         ;Beginn des inneren Loops
FADC 34          PUSH          ;(Warteschleife)
FADD 2F 01       LOOP FADD     ;Ende des inneren Loops
FADF 2F 06       LOOP FADA     ;Ende des äußeren Loops
FAE1 37          RTN           ;Return
```

Das zweite Programm ist mit dem ersten fast identisch. Nur wird hier nicht nur BZ2, sondern auch BZ1 gesetzt, womit die vierte Kombination gewählt wird und damit statt 2 kHz nun 4 kHz als

Frequenz benutzt wird.

Da BZ1 bei der Rückkehr ins BASIC-Programm im Gegensatz zu BZ2 nicht automatisch gelöscht wird, muß dieses im Maschinenprogramm geschehen, da sonst bei erneutem Aufruf des ersten Programms auch 4 kHz ertönen statt der erwarteten 2 kHz.

Auch dieses Programm ist relokatibel; der Aufruf lautet hier:

POKE &FACF,tondauer:CALL &FAE2

Assemblerlisting von Programm 2:

| | | | |
|------|----------|-----------|---------------------------------|
| FAE2 | 12 5F | LIP 5F | ;BZ1 und BZ2 setzen |
| FAE4 | 61 30 | ORIM 30 | |
| FAE6 | DF | OUTC | |
| FAE7 | 10 FA CF | LIDP FACF | ;Tondauerparameter in den Stack |
| FAEA | 57 | LDD | ;bringen (für äußeren Loop) |
| FAEB | 34 | PUSH | |
| FAEC | 02 FF | LIA FF | ;Beginn des inneren Loops |
| FAEE | 34 | PUSH | ;(Warteschleife) |
| FAEF | 2F 01 | LOOP FAEF | ;Ende des inneren Loops |
| FAF1 | 2F 06 | LOOP F5EC | ;Ende des äußeren Loops |
| FAF3 | 60 CF | ANIM CF | ;BZ1 und BZ2 löschen |
| FAF5 | DF | OUTC | |
| FAF6 | 37 | RTN | ;Return |

Das dritte Programm schließlich nutzt die beiden ersten Kombinationen aus. Durch abwechselndes Setzen und Löschen von BZ1 wird an Xout eine pulsierende Gleichspannung erzeugt, die, bei geeigneter Frequenz, vom Summer hörbar gemacht wird.

Durch Variation der Zeit zwischen dem Umschalten von BZ1 wird die Frequenz dieses Tons entsprechend geändert. Dazu wird an zwei Stellen im Programm (hier: FB05, FBOA) eine Zahl zwischen 0 und 255 gebracht. Je höher diese Zahl ist, desto länger ist die Periodendauer des Tons, d.h. desto niedriger ist seine Frequenz.

Auch dieses Programm ist relocatibel. Zu beachten ist jedoch, daß bei einer Verlegung des Programms auch die beiden Parameter-Adressen (FB05, FBOA) geändert werden müssen.

Aufruf:

```
POKE &FACF,tondauer:POKE &FB05,periodendauer:POKE &FBOA,peri-  
odendauer:CALL &FAF7
```

Assemblerlisting von Programm 3:

| | | | |
|------|----------|-----------|---------------------------------|
| FAF7 | 12 5F | LIP 5F | ;P-Register initialisieren |
| FCAF | 10 FA CF | LIDP FACF | ;Tondauerparameter in den Stack |
| FAPC | 57 | LDD | ;bringen (für äußeren Loop) |
| FAFD | 34 | PUSH | |
| FAFE | 02 FF | LIA FF | ;#255 in den Stack bringen |
| FBOO | 34 | PUSH | ;(für inneren Loop) |
| FB01 | 61 10 | ORIM 10 | ;BZ1 setzen |
| FB03 | DF | OUTC | |
| FB04 | 4E xx | WAIT xx | ;Halbe Periodendauer warten |
| FB06 | 60 EF | ANIM EF | ;BZ1 löschen |
| FB08 | DF | OUTC | |
| FB09 | 4E xx | WAIT xx | ;Halbe Periodendauer warten |
| FB0B | 2F 0B | LOOP FB01 | ;Ende des inneren Loops |
| FB0D | 2F 10 | WAIT FAFE | ;Ende des äußeren Loops |
| FBOF | 37 | RTN | ;Return |

Da Tondauer und Frequenz des Tons frei wählbar sind, kann man mit Hilfe dieser Routine Melodien spielen. Die dafür benötigten Parameter fasse man dabei am besten in DATA-Zeilen und lese sie mit dem READ-Befehl einzeln nacheinander ein , wobei man besondere Zahlen für Pause und das Ende der Melodie reserviert. Dieses geschieht unter anderem im auf der nächsten Seite folgenden kleinen Demonstrationsprogramm.

In den Zeilen 20 bis 50 werden abwechselnd die Programme 1 und 2 bei steigender Tondauer aufgerufen.

```

1: "A"
3: "Demonstrationsprogr
amm fuer drei Soundr
outinen
5: CLEAR
7: A=&FA00:B=&FAE2:C=&F
AF:D=&FB05:E=&FB0A:
F=&FACF
10: INPUT "LADEN ? (J/N)
": Q$
15: IF Q$="J" THEN GOSUB
900
17: PAUSE "OKAY, LOS GEH
TS"
20: FOR I=0 TO 50 STEP 5
30: POKE F,I:CALL A
35: CALL B
40: NEXT I
50: POKE F,255:CALL A:
CALL B
60: POKE F,8
70: FOR I=0 TO 100 STEP
2
80: POKE D,I:POKE E,I
90: CALL C
100: NEXT I
110: FOR I=100 TO 0 STEP
-4
120: POKE D,I:POKE E,I:
CALL C
130: NEXT I
140: FOR I=0 TO 20:NEXT I
150: RESTORE
160: READ I:READ J
170: IF I=300 THEN STOP
175: IF I=400 THEN FOR P=
0 TO 5:NEXT P:GOTO 1
60
180: POKE D,I:POKE E,I:
POKE F,J
190: CALL C
200: GOTO 160
210: DATA 50,1,45,1,40,1,
35,1,30,5,30,3,400,0
:25,1,25,1,25,1,25,1
,30,5,400,0,25,1
220: DATA 25,1,25,1,25,1,
30,5,400,0,35,1,35,1
,35,1,35,1,30,4,38,4
,400,0,30,1,30,1
230: DATA 30,1,38,1,40,4,
300,0
900: POKE &FA00,&12,&5F,&
61,&20,&DF,&10,&FA,&
CF,&57,&34,&02,&FF,&
34,&2F,&01,&2F
910: POKE &FAE0,&06,&37,&
12,&5F,&61,&30,&DF,&
10,&FA,&CF,&57,&34,&
82,&FF,&34,&2F
920: POKE &FAF0,&01,&2F,&
86,&60,&CF,&DF,&37,&
12,&5F,&10,&FA,&CF,&
57,&34,&02,&FF
930: POKE &FB00,&34,&61,&
10,&DF,&4E,&00,&60,&
EF,&DF,&4E,&00,&2F,&
03,&2F,&10,&37
940: RETURN

```

In den Zeilen 60 bis 130 wird bei konstantem Tondauerparameter Programm 3 bei sich ändernder Frequenz aufgerufen. Da der Tondauerparameter die Anzahl der Schwingungen des Summers bestimmt, ist zu beachten, daß trotz konstantem Parameter die Tondauer bei niedrigen Frequenzen länger ist als bei hohen Frequenzen.

In den Zeilen 150 bis 200 schließlich wird mit Programm 3 ein bekanntes Kinderlied gespielt. Die Frequenz- und Tondauerparameter stehen in den DATA-Zeilen 210 bis 230, wobei die Zahlen 400 für eine Pause und 300 für den Schluß stehen.

Die Zeilen 900 bis 940 schließlich stellen den BASIC-Lader dar, der die drei Maschinenprogramme in den Speicher lädt. Dazu muß vor Benutzung dieser Maschinenprogramme die am Anfang des BASIC-Programms stehende Frage "Laden ? (J/N)" in Zeile 10 mindestens einmal mit "J" beantwortet werden. Dieser Lader muß Bestandteil aller Programme sein, welche die drei Maschinenroutinen benutzen.

Man vergewissere sich, den BASIC-Lader korrekt eingegeben zu haben, bevor das Programm ausgeführt wird, da bereits eine falsche Zahl den Rechner zum Absturz und damit zum Verlust des Programms führen kann.

Falls es doch einmal passiert, muß man die 'ALL RESET'-Taste auf der Geräterückseite drücken und versuchen, das Programm mit den im Abschnitt 2.6 und 2.7 beschriebenen Methoden wiederzubeschaffen und zu ändern.

Falls das nicht klappt, bleibt einem nichts anderes übrig, als das Programm noch einmal zähneknirschend einzutippen. Wer über einen Cassetten-Recorder verfügt, hat hier überhaupt keine Probleme. Das Programm wird vor dem erstmaligen Ausführen auf Band genommen und kann dann jederzeit wieder eingelesen werden.

Die Adressen der Maschinenprogramme wurden hier so gewählt, daß sie am Ende des für BASIC-Programme reservierten RAM-Bereichs liegen.

Man darf also keine BASIC-Programme verwenden, die über die Adresse FACE hinausgehen. MEM muß dazu größer als 64 sein.

3. Matrizenrechnung

Der PC-1403 verfügt über ein breites Angebot interner Programme, mit denen sich die verschiedensten Matrizenoperationen sehr einfach durchführen lassen. Grund genug dafür, sich in diesem Buch etwas mit der Theorie der Matrizen zu beschäftigen, um auch demjenigen, der mit dem Begriff "Matrizen" wenig anfangen, dieses Programmangebot seines Rechners zu erschließen.

Selbstverständlich können die einzelnen Themen nur kurz ange schnitten werden; für genauere Studien empfehle ich /4/.

Über die Anwendung von Matrizen auf dem Gebiet der Analysis von Funktionen mehrerer Veränderlicher (auf die hier überhaupt nicht eingegangen werden soll) informiert u.a. /5/.

3.1. Was sind Matrizen ?

Matrizen sind Zahlenschemata von $n \times m$ Zahlen, die in einem matrixartigen Gitter mit n Zeilen und m Spalten eingetragen werden.

Die Matrizen werden mit großen Buchstaben (z.B. A) bezeichnet, ihre Elemente mit indizierten Kleinbuchstaben (z.B. a_{ij} , hierbei ist i der Zeilen- und j der Spaltenindex).

Dann wird durch

$$(1) \quad y_i = \sum_{j=1}^m a_{ij} x_j \quad \text{mit } i = 1, \dots, n$$

eine lineare Abbildung des aus den Komponenten x_j ($j=1 \dots m$) bestehenden Vektors \underline{x} auf den aus den Komponenten y_i ($i=1 \dots n$) bestehenden Vektors \underline{y} beschrieben.

Nennt man die aus den Elementen a_{ij} aufgebaute Matrix A, so kann man (1) durch die folgende Gleichung abkürzen:

$$(2) \quad \underline{y} = A\underline{x}$$

Da die Komponentendarstellung der Vektoren \underline{x} und \underline{y} basisabhängig ist (also abhängig ist vom gewählten Koordinatensystem), ist daher auch die Matrixdarstellung dieser linearen Abbildung basisabhängig.

Stammen \underline{x} und \underline{y} aus dem gleichen Vektorraum und haben damit die gleiche Dimension (dieses ist der häufigste Fall), so ist $n=m$, und die Matrix in (2) hat genauso viele Zeilen wie Spalten. Eine solche Matrix nennt man quadratisch.

Beispiele für lineare Abbildungen innerhalb des zweidimensionalen Vektorraums und ihre Matrixdarstellung bezüglich des gewöhnlichen cartesischen Koordinatensystems sind:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

mit

$$a_{11} \quad a_{12} \quad a_{21} \quad a_{22}$$

| | | | | |
|-------------------------------------|---------------|----------------|---------------|---------------|
| 1. Projektion auf die x-Achse | 1 | 0 | 0 | 0 |
| 2. Streckung um den Faktor α | α | 0 | 0 | α |
| 3. Identität | 1 | 0 | 0 | 1 |
| 4. Scherung in x-Richtung | α | 0 | 0 | 1 |
| 5. Spiegelung an der x-Achse | 1 | 0 | 0 | -1 |
| 6. Drehung um den Winkel θ | $\cos \theta$ | $-\sin \theta$ | $\sin \theta$ | $\cos \theta$ |

Als Anwendungsbeispiel möchten wir nun mit Hilfe des Rechners den Vektor (5,3) um 45° drehen. Dazu gehen wir folgendermaßen vor:

Mit 'CAL'-'SHIFT'-'↓' gehen wir in den Matrix-Mode über. Es erscheint die Anzeige "MATRIX: X(Ø_,Ø)". Wir legen die Zeilenzahl fest (hier: 2), drücken 'ENTER' und geben die Spaltenzahl ein (hier wieder 2). Auch diese Eingabe wird mit 'ENTER' abgeschlossen.

Es erscheint nun die Anzeige "X(1,1) Ø.". Es wird nun die Eingabe der Elemente der Operationsmatrix verlangt; in unserem Beispiel also die Elemente der Drehmatrix aus Beispiel 6 mit $\theta = 45^\circ$. Wir geben also nacheinander ein (im DEG-Modus):

"45"- 'cos' - 'ENTER' - "45"- 'sin' - '+/-' - 'ENTER' - "45"- 'sin' - 'ENTER' - "45"- 'cos' - 'ENTER' .

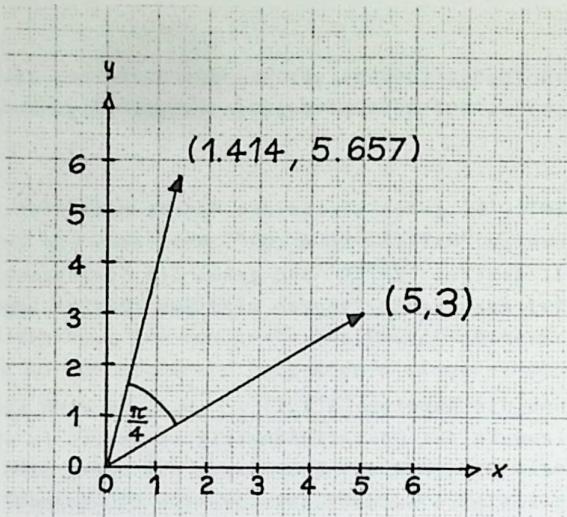
Nun erscheint die Anzeige "MATRIX: Y(Ø_,Ø)". Da wir einen Vektor formal als einspaltige Matrix schreiben können, geben wir nun dementsprechend die Zahlen 2 und 1 ein. Nach beiden Eingaben muß jeweils immer noch 'ENTER' gedrückt werden.

Nun werden die Komponenten des Vektors eingegeben. In unserem Beispiel sind dies also "5"- 'ENTER' - "3"- 'ENTER' .

Jetzt erscheint der Spruch "MATRIX OPERATION" in der Anzeige. Für das Anwenden von Matrizen auf Vektoren ist dieses die Multiplikation und somit drücken wir auf "*".

Durch Drücken auf "↓" können wir nun die Komponenten des um 45° gedrehten Vektors (5,3) ablesen: Es ist der Vektor $\sqrt{2}$ (1,4) = (1.414 , 5.657).

Das Ergebnis können wir leicht graphisch verifizieren:



Mit 'SHIFT' '↓' gelangen wir wieder in den normalen CAL-Modus zurück.

3.2 Matrizenmultiplikation

Werden zwei lineare Abbildungen hintereinander durchgeführt (also verkettet), so ist die Gesamtabbildung wiederum linear.

Folglich muß es eine Matrix C geben, so daß die Gleichung

$$(3) \quad \underline{y} = A(\underline{Bx})$$

mit A als eine $\ell \times n$ -Matrix und B als eine $n \times m$ -Matrix äquivalent ist zu der Gleichung

$$(4) \quad \underline{y} = \underline{Cx}$$

mit C als eine $\ell \times m$ -Matrix.

Die Elemente der Matrix C berechnen sich aus den Elementen der Matrizen A und B nach folgender Formel:

$$(5) \quad c_{ij} = \sum_{k=1}^n a_{ik} b_{kj} \quad i = 1, \dots, l \quad j = 1, \dots, m$$

Gleichung (5) wird dann folgendermaßen abgekürzt:

$$(6) \quad C = AB$$

Man nennt C das Matrizenprodukt von A und B und bezeichnet die Operation (5) als Matrizenmultiplikation.

Man beachte, daß die Spaltenanzahl der Matrix A gleich der Zeilenanzahl der Matrix B sein muß, um die Multiplikation ausführen zu können. Dieses ist deshalb notwendig, da die Dimension des Bildraumes von B gleich der Dimension des Definitionsräumes von A sein muß, um A und B verketten zu können (genauer gesagt: die zu den Matrizen A und B gehörenden linearen Abbildungen).

Die Durchführung einer Matrizenmultiplikation mit dem PC-1403 ist sehr einfach. Da die genaue Vorgehensweise bereits im vorangegangenen Kapitel eingehend beschrieben worden ist, folgt hier nur eine stichwortartige Zusammenfassung:

- Übergang vom CAL- in den Matrix-Mode mit 'SHIFT' '↓'
- Eingabe der Dimension der Matrix X
- Eingabe der Elemente der Matrix X
- Eingabe der Dimension der Matrix Y
- Eingabe der Elemente der Matrix Y
- Eingabe der Matrixoperation "*"
- Ausgabe der Ergebnismatrix XY in der Matrix X

Es sei an dieser Stelle nochmals betont, daß die Spaltenanzahl der Matrix X gleich der Zeilenanzahl der Matrix Y sein muß, um die Matrizenmultiplikation XY durchführen zu können. Sonst gibt der Rechner nach Eingabe der Operationsart "*" eine Fehlermeldung aus ("IMPOSSIBLE CALCULATION").

3.3 Matrizeninversion

Ist eine lineare Abbildung

$$(7) \quad \underline{y} = A\underline{x}$$

bijektiv (d.h. eineindeutig), so existiert genau eine inverse Abbildung

$$(8) \quad \underline{x} = B\underline{y}$$

Man nennt B die inverse Matrix von A und bezeichnet sie mit A^{-1} . Notwendige Voraussetzung für die Eineindeutigkeit von (7) ist, daß \underline{y} und \underline{x} die gleiche Dimension haben; A also quadratisch ist.

Jedoch sind nicht alle quadratischen Matrizen invertierbar, da die ihnen zugeordneten Abbildungen nicht eineindeutig sind. Solche Matrizen nennt man singulär; das Gegenteil heißt regulär.

Betrachtet man z.B. Beispiel 1 aus Kapitel 3.1 (Projektion auf die x-Achse), so sieht man unmittelbar ein, daß diese Abbildung nicht eindeutig umkehrbar ist, da alle Vektoren, deren Spitzen sich auf einer gemeinsamen Parallelen zur y-Achse befinden, dieselbe Projektion auf die x-Achse besitzen; also sämtlich den selben Bildvektor besitzen.

Setzt man (8) mit $B = A^{-1}$ in (7) ein, so erhält man

$$(9) \quad \underline{y} = AA^{-1}\underline{y}$$

Also repräsentiert die Matrix AA^{-1} die Identitätsabbildung.
Das Matrizenprodukt AA^{-1} ist also gleich der sogenannten Einheitsmatrix E.

$$(10) \quad AA^{-1} = A^{-1}A = E$$

Ist A eine $n*n$ -Matrix, so ist (10) äquivalent zu

$$(11) \quad \sum_{k=1}^n a_{ik} a^{-1}_{kj} = \delta_{ij},$$

wobei δ_{ij} das sogenannte Kroneckersymbol darstellt, welches folgendermaßen definiert ist:

$$(12) \quad \delta_{ij} = \begin{cases} 1 & \text{für } i=j \\ 0 & \text{für } i \neq j \end{cases}$$

Die Berechnung der Inversen einer Matrix ist sehr einfach, wenn man sie mit dem PC-1403 durchführt. Hier wieder die stichwortartige Zusammenfassung:

- Übergang vom CAL- in den Matrix-Modus mit 'SHIFT' '+'
- Eingabe der Dimension der Matrix X
- Eingabe der Elemente der Matrix X
- Eingabe der Matrizenoperation "1/x"
- Ausgabe der Elemente der Matrix X^{-1} in der Matrix X

Fehlermeldungen:

- (i) Matrix ist nicht quadratisch: "IMPOSSIBLE CALCULATION"
- (ii) Matrix ist singulär: "DIVISION BY ZERO"

3.4 Lösung linearer Gleichungssysteme

Mit Hilfe der eingebauten Matrizenrechnungsprogramme können wir auf eine sehr einfache Art und Weise ein lineares Gleichungssystem mit konstanten Koeffizienten durch den Computer lösen lassen, sofern es eindeutig lösbar ist.

Ein solches Gleichungssystem, bestehend aus n Gleichungen mit n Unbekannten x_i hat die folgende Form:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n &= y_1 && 1. \text{ Gleichung} \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n &= y_2 && 2. \text{ Gleichung} \\ \dots & && \text{usw.} \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n &= y_n && n. \text{ Gleichung} \end{aligned} \tag{13}$$

Faßt man die Koeffizienten a_{ij} zur Matrix A zusammen, die Unbekannten zum Vektor \underline{x} und die rechts vom Gleichheitszeichen stehenden Konstanten y_i zum Vektor \underline{y} , so läßt sich Gleichung (13) umschreiben zu

$$(14) \quad A\underline{x} = \underline{y},$$

welches man mit (1) leicht beweisen kann.

Ist das Gleichungssystem (13) eindeutig lösbar, so ist die Koeffizientenmatrix A regulär, d.h. man kann die Inverse A^{-1} bilden.

Damit lässt sich (14) explizit nach x auflösen:

$$(15) \quad \underline{x} = A^{-1} \underline{y}$$

und man kann so die Unbekannten (also die Komponenten des Vektors x) direkt berechnen.

Als Beispiel möchten wir das einfache Gleichungssystem

$$\begin{array}{rcl} x & + & y & + & z & = & 11 \\ 2x & - & 3y & + & 5z & = & -16 \\ & y & - & z & = & 8 \end{array}$$

lösen.

Wir schalten den Rechner zunächst wieder in den Matrix-Modus, dimensionieren die Koeffizientenmatrix X (3*3) und geben der Reihe nach ihre Elemente ein: 1,1,1,2,-3,5,0,1,-1.

Es folgt die Dimensionierung des Vektors Y (3*1) und die Eingabe seiner Komponenten (11,-16,8).

Als Matrizenoperation drücken wir auf '1/x' zwecks Inversenbildung und direkt anschließend auf "*" zwecks Berechnung von (15).
(Anmerkung: In Anführungsstriche auftretende Zeichen bedeuten hier druckbare Schriftzeichen, in Apostrophe eingeschlossene Zeichen meinen Funktionstasten mit entsprechender Bezeichnung).

Die in X stehenden Komponenten geben uns die Lösung des Systems:

$$x = 5 \quad , \quad y = 7 \quad , \quad z = -1$$

Einsetzen dieser Werte in das Gleichungssystem bestätigt die Richtigkeit dieses Ergebnisses.

Ist das Ergebnis nicht eindeutig, weil die n Gleichungen linear

voneinander abhängen, so läßt sich das Problem auf diese Art nicht lösen, da die Koeffizientenmatrix in diesem Fall singulär ist. Die Folge ist, daß beim Versuch, A zu invertieren, die Fehlermeldung "DIVISION BY ZERO" in der Anzeige erscheint. Dasselbe passiert, wenn die Gleichungen sich gegenseitig widersprechen; das Gleichungssystem also überhaupt nicht lösbar ist.

3.5 Eigenwerte und Eigenvektoren

Abschließend sollen in diesem Abschnitt noch einige Worte über die sogenannten Eigenwertgleichungen verloren werden. Das Problem ist, daß man nach Vektoren sucht, die parallel zu ihrem Bildvektor bezüglich einer bestimmten linearen Abbildung liegen. Nennt man die dieser Abbildung entsprechende Matrix A, so soll also folgende Gleichung gelöst werden:

$$(16) \quad A\mathbf{x} = \lambda\mathbf{x}$$

mit skalarem λ .

Diejenigen Vektoren \mathbf{x} , die (16) erfüllen, nennt man Eigenvektoren, die entsprechenden Werte für λ werden als Eigenwerte bezeichnet.

Ein anschauliches Beispiel für die Lösung eines Eigenwertproblems in der Natur ist die freie Rotation eines Körpers (siehe dazu auch z.B. /6/, /7/).

Der Drehimpulsvektor \underline{L} und der in Richtung der momentanen Drehachse weisende Kreisfrequenzvektor $\underline{\omega}$ hängen über eine lineare Abbildung zusammen:

$$(17) \quad \underline{L} = \theta\underline{\omega}$$

zusammen. Die Matrix Θ wird Trägheitstensor genannt und stellt eine für den rotierenden Körper charakteristische Größe dar.

Freie Drehungen sind nur dann möglich, wenn \underline{L} und $\underline{\omega}$ parallel zueinander stehen. Folglich werden die Drehachsen, um die freie Drehungen möglich sind, durch die Eigenvektoren $\underline{\omega}_i$ zu Θ festgelegt; die entsprechenden Eigenwerte λ_i nennt man Hauptträgheitsmomente.

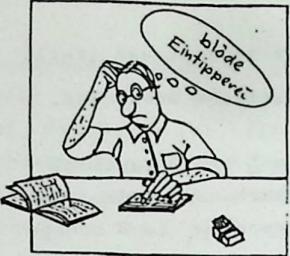
Das Lösen eines Eigenwertproblems (16) ist etwas kompliziert. Man bildet die Matrix A^* , deren Elemente folgendermaßen definiert werden:

$$(18) \quad a_{ij}^* := a_{ij} - \lambda \delta_{ij},$$



wobei δ_{ij} wieder das in (12) definierte Kronecker-Symbol darstellt. Nun berechnet man die Determinante von A^* und erhält auf diese Weise ein Polynom von λ , dessen Nullstellen λ_k bestimmt werden. Diese λ_k sind die möglichen Eigenwerte. Setzt man sie einzeln statt λ in (18) ein und betrachtet die so gebildete Matrix A_{ik}^* als Koeffizientenmatrix eines homogenen Gleichungssystems, so bilden dessen Lösungen die Eigenvektoren zum Eigenwert λ_k .

Sehr einfach ist die Überprüfung eines Vektors auf Eigenvektoreigenschaft. Dieses geschieht im folgenden BASIC-Programm, mit dem wir diesen theoretischen Exkurs verlassen wollen und uns im nächsten Abschnitt wieder mehr dem Rechner selbst zuwenden.

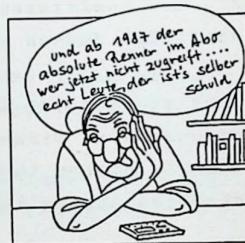


```

5:CLEAR
10:INPUT "DIMENSION: ";
N
15:N=N-1
20:DIM M(N,N),V(N),B(N)
30:PAUSE "MATRICEINGABE"
*
40:FOR I=0 TO N
50:FOR J=0 TO N
60:POKE 57525,(49+I):
POKE 57527,(49+J)
70:INPUT "(3/3) := ";M(I,
,J)
80:NEXT J
90:NEXT I
100:PAUSE "VEKTOREINGABE"

105:Z=1
110:FOR I=0 TO N
120:POKE 57613,(49+I)
130:INPUT "V(I) := ";V(I)
135:IF V(I)<>0 THEN LET
Z=0
140:NEXT I
145:IF Z THEN BEEP 1:
PAUSE "NULLVEKTOR IS
T":PAUSE "KEIN EIGEN
VEKTOR":GOTO 100
147:E=1:R=0
150:FOR I=0 TO N
155:B(I)=0
160:FOR J=0 TO N
170:B(I)=B(I)+M(I,J)*V(J
)
180:NEXT J
190:NEXT I
200:PAUSE " BILDVEKTOR
"
210:FOR I=0 TO N
220:PRINT "<";STR$ (I+1)
;">= ";STR$ B(I)
230:IF V(I)=0 THEN IF B(
I)>0 THEN LET E=0:
GOTO 250
240:IF 1-R THEN IF V(I)<
>0 THEN LET R=1:LET
Q=B(I)/V(I):GOTO 250
248:IF B(I)<>Q*V(I) THEN
LET E=0
250:NEXT I
260:IF E THEN PAUSE " EI
GENVEKTOR !!":PRINT
"EIGENWERT ";STR$ Q
:GOTO 100
270:PRINT "KEIN EIGENVEK
TOR":GOTO 100

```



4.1 Verzicht auf den IF-Befehl

Logische Ausdrücke (Ausdrücke, die entweder wahr oder falsch sind wie z.B. $A > B$, $C = 0$ etc.) lassen sich in Programmen wie numerische Ausdrücke verwenden: Ist der Ausdruck wahr, so repräsentiert er den Wert Eins, ist er falsch, so repräsentiert er hingegen die Zahl Null. Beispiel:

LET A=(B>15)

Nach Ausführung dieses Befehls steht in der Variablen A eine Eins, falls $B > 15$ ist, und eine Null, falls $B \leq 15$. Somit er-setzt dieser einfache Befehl die beiden IF-Zeilen

IF B>15 THEN LET A=1
IF B<=15 THEN LET A=0

Man kann also auf IF-Befehle verzichten und mehrere IF-Zeilen in einem Befehl obenstehender Form zusammenfassen, wodurch das Programm insgesamt kürzer wird. Einige Beispiele hierfür sind in der folgenden Tabelle gegenübergestellt:

mit IF-Befehl

IF B<0 THEN LET A=24
IF B=0 THEN LET A=53

IF P>5 THEN PRINT "\$"
IF P=5 THEN PRINT "*"
IF P<5 THEN PRINT ":"

IF P>=87 THEN LET A=A+11
IF S=5 THEN LET A=A-52

ohne IF-Befehl

$A = (B < 0) * 24 + (B = 0) * 53 + (B > 0) * A$

PRINT CHR\$((P>5)*36+(P=5)*42+
(P<5)*58)

$A = A + 11 * (P >= 87) - 52 * (S = 5)$

mit IF-Befehl

```
IF A=4 AND B<6  
THEN LET C=8  
IF A<>4 OR B>=6  
THEN LET C=2
```

ohne IF-Befehl

```
C=(A=4)*(B<6)*6+2
```

Diese neue Methode ist zwar im allgemeinen kürzer als die konventionelle, jedoch auch unübersichtlicher. Es hat daher keinen Sinn, auf IF-Befehle völlig zu verzichten.

Man sollte vielmehr von Fall zu Fall entscheiden, ob sich der Verzicht auf den IF-Befehl wirklich lohnt und das Programm wesentlich verkürzt.

Da logische Ausdrücke, wie wir eben gesehen haben, im Grunde genommen auch nur numerische sind, kann man letztere natürlich auch umgekehrt in IF-Statements einsetzen.

So wird der PRINT-Befehl

```
IF E THEN PRINT "HALLO"
```

genau dann ausgeführt, wenn E größer als Null ist.

Die Befehlsfolge

```
E=(P=14)  
IF E THEN LET Z=A
```

ist damit bis auf die Veränderung der Variablen E äquivalent zu dem Einzelbefehl

```
IF P=14 THEN LET Z=A
```

4.2 Rechenun genauigkeit

Einige Programme laufen nur deshalb nicht richtig, weil eine Bedingung, die theoretisch wahr ist, vom Rechner aufgrund ungenauer Rechnungen als falsch angesehen wird.

Zweifellos ist zum Beispiel $(2^{(1/50)})^{50} = 2$. Der PC-1403 berechnet aber für $(2^{(1/50)})^{50}$ den Wert $2 - 1.02E-09$. Diese Differenz ist zwar gering, hat aber zur Folge, daß der Rechner den Ausdruck $(2^{(1/50)})^{50} = 2$ als falsch ansieht und entsprechende IF-Befehle nicht ausführt. Um diesem Übel abzuheilen, muß man Toleranzstreifen (z.B. 1E-08) um das zu erzielende Ergebnis zulassen.

Der logische Ausdruck

```
zahl1 = zahl2
```

muß danach ersetzt werden durch den Ausdruck

```
ABS (zahl1-zahl2) < toleranz
```

d.h. der Betrag der Differenz zwischen zahl1 und zahl2 muß kleiner als die erlaubte Toleranz sein, um zahl1 und zahl2 als gleich zu betrachten.

So wird zum Beispiel der Ausdruck $ABS((2^{(1/50)})^{50}-2)<1E-08$ vom PC-1403 als wahr angesehen. Auf diese Weise eingeführte Toleranzstreifen sind jedoch im allgemeinen nur dann erforderlich, wenn das zu erzielende Ergebnis (zahl2) entweder als Dezimalzahl nicht vollständig darstellbar ist (irrationale Zahlen wie π oder periodische Brüche wie $1/3$) oder gleich Null ist.

4.3 Die RND-Funktion

Die RND-Funktion des PC-1403 gibt eine Zufallszahl aus, die sich innerhalb des mit dem Funktionsargument gewähltem Bereich bewegt (siehe Bedienungsanleitung).

Natürlich ist beim Rechner nichts vom Zufall abhängig; vielmehr wird die Zahl nach einem komplizierterem Algorithmus berechnet. Jedoch erfüllen diese Zahlen zwei wichtige Kriterien:

- (i) Sie sind vom Benutzer nicht vorhersagbar.
- (ii) Sie sind innerhalb des gewählten Bereichs gleichmäßig verteilt.

Daher können sie als Zufallszahl betrachtet werden.

Es sollen nun zwei Programme vorgestellt werden, die sich im wesentlichen auf die RND-Funktion stützen.

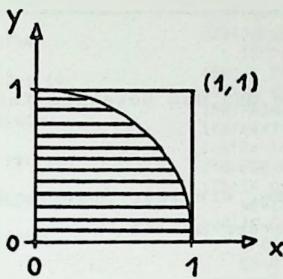
Das erste Programm soll anhand der Berechnung von π demonstrieren, wie man die unvorhersagbaren Zufallszahlen für recht genaue Näherungsrechnungen einsetzen kann.

Das zweite Programm steht stellvertretend für viele Spielprogramme, die die RND-Funktion als "Würfel" einsetzen; Mit ihm kann man das Kartenspiel 17+4 spielen.

4.3.1 Bestimmung von π nach der Monte-Carlo-Methode

Das Prinzip des Programms ist sehr einfach: Es bestimmt immer Paare von Zufallszahlen x, y zwischen 0 und 1, die man in der xy -Ebene als Punkte betrachten kann.

Diese Punkte befinden sich im eingezeichneten Quadrat mit der



Kantenlänge 1. Es werden alle Punkte gezählt, die sich innerhalb des schraffierten Viertelkreises mit dem Radius 1 befinden. (Bedingung: $x^2 + y^2 < 1$).

Da die Zufallszahlen gleichmäßig innerhalb des Bereichs zwischen Null und Eins verteilt sind, nähert sich das Verhältnis der Anzahl der Punkte innerhalb des Viertelkreises zu der Gesamtzahl der Punkte bei steigender Punktzahl dem Verhältnis der Fläche des Viertelkreises zu der des Quadrates, also der Zahl $\pi/4$.

Auf diese Weise kann man sich der Zahl π mit beliebiger Genauigkeit nähern. Selbstverständlich ist das Ergebnis umso genauer, je mehr Punkte bestimmt werden.

Mit der gleichen Methode kann man natürlich auch Flächen unter Funktionsgraphen bestimmen, d.h. Funktionen integrieren.

```

65:BEEP 1
70:PAUSE "π = ";S/N*4
75:PAUSE "Abweichung: "
; USING "#####.##";(
S/N*4/X-1)*100;" %":
USING
80:WAIT 15:PRINT "(J/H)
":WAIT
90:IF INKEY$ = "J" GOTO
20
100:IF INKEY$ <>"N" GOTO
80
110:END
1:"A"
5:"Bestimmung von X nach der Monte-Carlo-Methode
10:CLEAR :RANDOM
20:S=0:INPUT "Punktzahl
h": ";N
30:FOR I=1 TO N
40:X=RND 0:Y=RND 0
50:S=S+(X*X+Y*Y<1)
60:NEXT I

```

4.3.2 Spiel 17+4

Mit diesem Programm kann man das bekannte Kartenspiel 17+4 spielen.

Zunächst muß man eingeben, wieviele Spieler teilnehmen und wieviele Runden gespielt werden sollen.

Mit Hilfe der RND-Funktion werden sodann dem jeweils spielenden Teilnehmer (alle Teilnehmer bekommen eine Nummer verpasst und kommen der Reihe nach dran) Karten zugeteilt. Neben dem Kartennamen werden hierbei gleichzeitig der Wert dieser Karte und die Summe der Punkte der bisherigen Karten (einschließlich der gerade gezogenen) angezeigt.

Die einzelnen Karten haben folgende Werte (in Punkten):

| | | |
|-------------|-------------|------------|
| Bube 2 | Sieben .. 7 | Zehn .. 10 |
| Dame 3 | Acht 8 | As 11 |
| König ... 4 | Neun 9 | |

Anschließend wird der Spieler gefragt, ob er noch eine Karte haben möchte. Er muß sie mit "J" oder "N" beantworten.

Überschreitet der Spieler die Summe 21, so verfällt sein Konto. Erhält er zwei Asse, so zählen diese zusammen 21 Punkte (und ^{etw} nicht 22 !!!), womit er das Optimum an Punkten erreicht hat.

Am Ende jeder Runde werden der oder die Sieger ausgegeben, also die Spieler, die in dieser Runde die meisten Punkte erreicht haben und deshalb einen Punkt mehr in der Gesamtliste bekommen.

Am Ende des Spiels werden für jeden Spieler die Punktzahl der Gesamtliste angezeigt.

```

1: "A"
2: "Kartenspiel 17+4
5:CLEAR
10:PRINT " ==> 17+4 <
      ==
20:INPUT "SPIELERANZAHL
      ?,A
30:DIM P(A),L(A),K$(11)
      ,D(A)
35:FOR N=2 TO 11:READ K
      $(N):NEXT N
40:INPUT "RUHDENANZAHL
      ?,Z
45:RANDOM
50:FOR N=1 TO Z
55:PRINT N;" Runde"
60:FOR M=1 TO A
65:X=0:P(M)=0
70:PRINT "ES SPIELT ";
      ISTR$ M
80:K= INT (10*(RND .5))
      +2
82:IF (K=5) OR (K=6)
      GOTO 80
85:IF (K=11) AND (X<>-1
      ) THEN LET X=X+1
87:IF K>11 THEN LET X=
      -1
90:P(M)=P(M)+K:IF X=2
      THEN LET P(M)=21
100:PRINT K$(K);"; "
      STR$ K;" ";STR$ P(
      M)
110:IF P(M)>21 THEN
      PRINT "DAS TUT MIR L
      EID":LET P(M)=0:GOTO
      160
115:Y$=" "
120:INPUT "NOCH EINE KAR
      TE?",Y$
130:IF Y$="J" GOTO 80
140:IF Y$(<)>"N" GOTO 120
150:PRINT "OKAY"
160:NEXT M
165:C=-1:E=0
170:FOR M=1 TO A
175:IF C=P(M) THEN LET E
      =E+1:LET D(E)=M
180:IF C>P(M) THEN LET C
      =P(M):LET E=0:LET D(
      E)=M
190:NEXT M
195:FOR I=0 TO E
200:PRINT "SIEGER: ";
      STR$ D(I)
205:L(D(I))=L(D(I))+1
210:NEXT I
220:NEXT N
230:PRINT "ENDSTAND:"
240:FOR N=1 TO A
250:PRINT "#";ISTR$ N;;
      ";STR$ L(N);" PKT.
      .
260:NEXT N
270:PRINT "### TSCHUESS
      $$$"
280:DATA "BUBE","DAME",""
      KOENIG","","","SIE
      BEN","ACHT","NEUN",""
      ZEHN","AS"
290:END

```

4.4 Anzeige von Indizes beim INPUT-Befehl

Es gibt zwar die Möglichkeit, bei INPUT-Befehlen einen beliebigen Text auszugeben, wie zum Beispiel in

```
INPUT "Parameter: ";P
```

Doch sind nur wörtliche Texte zugelassen; der Versuch, den Text mit einer Stringvariable darzustellen, wie z.B. in

```
B$="Parameter: "
INPUT B$;P
```

führt unweigerlich zu einem Syntaxfehler (ERROR 1) in der INPUT-Zeile.

Liest man aber die Elemente eines Feldes nacheinander ein, so ist es natürlich wünschenswert, beim INPUT-Befehl die Nummer (den Index) des gerade einzulesenden Feldelements mitzugeben.

Ein Trick hilft uns hier weiter: In dem Kapitel 2.5 haben wir das Format kennengelernt, in dem Programme gespeichert werden. Mit dem POKE-Befehl haben wir nun die Möglichkeit, im laufenden Programm das Programm selbst zu verändern. Man kann dadurch auch laufende Indizes in den wörtlichen Text eines INPUT-Befehls bringen.

```
Beispiel: 10:DIM B$(9)
           20:FOR I=0 TO 9
           30:POKE 57436,48+I
           40:INPUT "B$( ): ";B$(I)
           50:NEXT I
```

Bei der Ausführung wird der Code der in I stehenden Zahl in die Leerzeichen zwischen den beiden Klammern im Text des INPUT-Befehls zugeordneten Adresse "gepoket". Diese Adresse erhält man folgendermaßen:

Man tippt das Programm bis zu der Stelle in den Rechner, wo das zu verändernde Zeichen hinkommen soll, wobei man für die Adresse im POKE-Befehl zunächst fünf beliebige Zeichen einsetzt. Dieses zu verändernde Zeichen wird hierbei nicht mitgetippt; man macht also im oberen Beispiel nach dem "(" -Zeichen Schluß.

Die Adresse des zu verändernden Zeichens berechnet man dann nach folgender Formel:

&FBOE - MEM

Nach Einsetzen dieser Adresse in den POKE-Befehl kann man dann

das Programm zu Ende tippen.

Es ist selbstverständlich darauf zu achten, daß sich durch nachträgliches Einfügen bzw. Löschen einzelner Zeichen oder gar ganzer Zeilen im Programmteil oder dem zu verändernden Zeichen sich auch dessen Adresse verändert. Diese muß entsprechend korrigiert werden, damit man später nicht "danebenpoket".

4.4.1 Berechnung der Inversen einer Matrix

Mit diesem Programm lassen sich n*n-Matrizen invertieren.

Die Elemente der Matrix werden zeilenweise (von links nach rechts, von oben nach unten) eingelesen; die Elemente der Inversen dieser Matrix werden in entsprechender Weise ausgegeben.

Falls die eingegebene Matrix singulär und damit nicht invertierbar ist, wird eine Fehlermeldung ausgegeben und das Programm abgebrochen.

```
5:CLEAR
10:INPUT "Rang: ";N
20:IF N>5 GOTO 10
30:H=N-1:M=2*N+1
35:DIM M(N,M),E(M):P=N
40:FOR I=0 TO N
50:FOR J=0 TO N
60:POKE 57518,(49+I):
    POKE 57520,(49+J)
70:INPUT "(3/3):= ";M(I
    ,J)
80:NEXT J
90:NEXT I
100:FOR I=0 TO N
110:FOR J=N+1 TO M
120:M(I,J)=(J=I+N+1)
130:NEXT J
140:NEXT I
145:Z=1E-8
150:FOR J=0 TO N-1
160:FOR I=J+1 TO N
172:IF ABS M(J,J)<Z AND
    ABS M(P,J)>Z GOTO 45
    0
174:IF ABS M(J,J)>Z GOTO 0
    177
175:P=P-1:IF P=J THEN
    LET M(N,N)=0:GOTO 24
    0
176:GOTO 172
177:P=N
178:IF ABS M(I,J)<Z GOTO
    220
180:F=M(I,J)/M(J,J)
190:FOR K=J TO M
200:M(I,K)=M(I,K)-F*M(J,
    K)
210:NEXT K
220:NEXT I
230:NEXT J
240:IF ABS M(N,N)<Z THEN
    BEEP 2:PRINT "Matrix
    ist singulaer":END
250:FOR J=N TO 1 STEP -1
260:FOR I=J-1 TO 0 STEP
    -1
270:IF ABS M(I,J)<Z GOTO
    320
280:F=M(I,J)/M(J,J)
290:FOR K=J TO M
300:M(I,K)=M(I,K)-F*M(J,
    K)
310:NEXT K
320:NEXT I
330:NEXT J
340:FOR I=0 TO N
350:FOR J=N+1 TO M
360:M(I,J)=M(I,J)/M(I,I)
370:NEXT J
380:NEXT I
390:FOR I=0 TO N
400:FOR J=N+1 TO M
410:PRINT "(";STR$(I+1);
    ";" USING "####.##";
    ;M(I,J)
420:NEXT J
430:NEXT I
440:END
450:FOR Y=0 TO M
460:E(Y)=M(J,Y)
470:M(J,Y)=M(P,Y)
480:M(P,Y)=E(Y)
490:NEXT Y
500:GOTO 177
```

Es gibt viele Verfahren, ein aus Zahlen oder Strings bestehendes Feld der Reihenfolge nach zu sortieren. Im allgemeinen werden dabei immer je zwei Zahlen (bzw. Texte) miteinander verglichen, um ihnen dann, je nach Ergebnis des Vergleichs, neue Plätze im Feld zuzuordnen. Über die unterschiedliche Effizienz verschiedener Sortierverfahren gibt es Abhandlungen in /8/ und /9/.

Leider haben die sehr schnellen Sortierverfahren den Nachteil, sehr lang und kompliziert zu sein. Auf der anderen Seite sind die sehr kurzen Programme auch die langsamsten.

Man muß also hinsichtlich der Länge der Programme und seiner Effizienz einen Kompromiß eingehen.

Deshalb möchte ich in diesem Kapitel zwei Sortierprogramme vorstellen: Ein kurzes, langsames sowie ein langes, schnelles Programm zum Sortieren von Strings.

Glücklicherweise lassen sich zwei Strings `string1` und `string2` nicht nur auf Gleichheit und Ungleichheit untersuchen. Vielmehr hat auch der Ausdruck

`string1 < string2`

beim PC-1403 einen Sinn. Diese Aussage ist genau dann wahr, wenn der Text in `string1` in der alphabetischen Reihenfolge vor dem Text in `string2` auftritt. Entsprechend sind die anderen Vergleichsoperatoren "`>`", "`<=`", "`>=`" definiert.

Genauer gesagt dient als Ordnungskriterium nicht das Alphabet, sondern der gesamte ASCII-Code, in dem allerdings alle 26 Buchstaben in alphabetischer Reihenfolge auftreten.

Ebenso treten auch die Ziffern 0 bis 9 geordnet in diesem Code

auf. Daher eignen sich beide Programme auch zum Ordnen von Zahlen, wobei beachtet werden muß, daß alle Zahlen dieselbe Stellenanzahl besitzen. Ist die höchste auftretende Zahl beispielsweise 1987, so muß z.B. eine Sieben als "0007" oder " 7" (mit drei führenden Leerzeichen) eingegeben werden.

4.5.1 Bubble Sort

Das Bubble-Sort-Verfahren ist ein sehr kurzes Sortierverfahren: Es besteht nur aus zwei ineinander verschachtelten FOR-NEXT-Schleifen.

Ein Sortierprogramm, daß sich dieses Verfahren bedient, sieht folgendermaßen aus:

```
10:DIM B$(50),C$(0)
20:I=0
30:INPUT B$(I)
40:IF B$(I)<>"*" THEN LET I=I+1:GOTO 30
50:N=I-1
60:FOR I=0 TO N-1
70:FOR J=I+1 TO N
80:IF B$(I)>B$(J) THEN LET C$(0)=B$(J):LET B$(J)=B$(I):
    LET B$(I)=C$(0)
90:NEXT J
100:NEXT I
110:FOR I=0 TO N
120:PRINT B$(I)
130:NEXT I
```

Die zu sortierenden Strings (bis zu 50) werden zunächst ungeordnet eingelesen, wobei ein "*" als Schlußzeichen gewertet wird. Nach dem Sortiervorgang werden die Texte in geordneter Reihenfolge ausgegeben.

Der eigentliche Sortieralgorithmus befindet sich in den Zeilen 60 bis 100.

Die Variable I zeigt zunächst auf das erste Element des Feldes. J läuft nun von I+1 bis N durch. Dabei wird jedesmal geprüft, ob das i-te Element größer als das j-te Element ist.

Ist das der Fall, so werden beide Elemente ausgetauscht, so daß am Ende das Minimum aller Elemente im ersten Feldelement seinen endgültigen Platz gefunden hat.

Nun wird I um 1 erhöht und zeigt somit auf das zweite Element. J läuft nun wieder von I+1 bis N durch, dabei kommt das Minimum dieser verbliebenen Elemente zum schluß der J-Schleife wieder auf die i-te Position (auf die zweite position also).

Das ganze wird solange wiederholt, bis I bis N-1 durchgelaufen ist. Dann ist das Feld vollständig sortiert.

Der Name "Bubble Sort" röhrt daher, daß das Minimum gewissermaßen wie eine Blase (engl. bubble) zum Vorschein kommt und nach oben auf seine Position kommt.

Das Programm ist zwar sehr kurz, aber auch sehr langsam. Will man insbesondere möglichst viele Strings sortieren, sollte man auf das folgende Programm zurückgreifen.

4.5.2 Sortierprogramm mit Min-Max-Suche

Der wesentliche Unterschied zwischen diesem Programm und Bubble Sort ist, daß hier bei jedem Durchlauf nicht nur das Minimum, sondern auch das Maximum der durchsuchten Elemente auf ihre endgültigen Plätze verwiesen wird. Es liegt auf der Hand, daß sich dadurch die Dauer des Sortiervorgangs wesentlich verkürzt. Die Handhabung des Programms ist identisch mit der von Bubble Sort; daher erübriggt sich eine erneute Beschreibung.

```

1: "A"
3: "Sortierprogramm fue
   r Woerter und Zeiche
   n
5: CLEAR
7: WAIT 100
10: DIM F$(55), R(1), M$(1)
    )
20: N=0
30: N=N+1
40: INPUT F$(N)
50: IF F$(N)="" GOTO 3
     0
60: A=1: E=N-1
70: M$(1)=F$(A): R(1)=A
80: M$(0)=M$(1): R(0)=R(1)
    )
110: FOR M=A TO E
120: IF F$(M)>M$(1) THEN
      LET M$(1)=F$(M): LET
      R(1)=M
130: IF F$(M)<M$(0) THEN
      LET M$(0)=F$(M): LET
      R(0)=M
140: NEXT M
150: F$(R(0))=F$(A)
152: F$(A)=M$(0)
155: A=A+1
157: IF F$(R(0))=M$(1)
      THEN LET R(1)=R(0)
160: F$(R(1))=F$(E)
162: F$(E)=M$(1)
165: E=E-1
170: IF A<E GOTO 90
175: BEEP 1
180: PRINT "GEORDNET:"
190: FOR M=1 TO N-1: PRINT
      F$(M): NEXT M
200: END

```



5. Programme

Zum Abschluß dieses Buches folgen einige BASIC- und Maschinenprogramme aus den unterschiedlichsten Anwendungsgebieten.

Die BASIC-Programme sind im Prinzip auf allen BASIC-Rechnern lauffähig; evtl. müssen einige spezielle Befehle geändert werden.

5.1 Monatskalender

Zu diesem Programm ist wenig zu sagen: Wer im Besitz eines Druckers ist, kann sich damit den Kalender für einen beliebigen Monat zwischen 1901 und 2099 ausdrucken lassen.

Es folgen einige Beispiele, mit denen man die korrekte Eingabe des Programms kontrollieren kann.

Juni 1987

```
#####
Mo Di Mi Do Fr Sa So
-----
1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30
#####
#
```

September 1988

```
#####
Mo Di Mi Do Fr Sa So
-----
1 2 3 4
5 6 7 8 9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30
#####
#
```

```
1;"A"
3;"Ausdruck eines beliebigen Monatskalenders zwischen 1901 und 2099
5:CLEAR :RESTORE
7:DIM M$(0)*10,Z$(0)*2
1
8:USING "#####"
10:INPUT "Monat: ";M
15:IF (M<1) OR (M>12)
    THEN PRINT "NUR
    1901-2099":GOTO 20
30:D=J-1901+ INT ((J-19
    01)/4)
40:FOR I=1 TO M
50:READ M$(0)
60:READ H
66:H=H+((I=2) AND ((J
    AND 3)=0))
70:D=D+H
80:NEXT I
90:D=D-H
100:D=- INT (7*(D/7- INT
    (D/7))+1)
105:IF D<-5 THEN LET D=D
    +7
110:LPRINT LEFT$ (""
    ,11-(LEN M$(0)
    +5)/2);M$(0);J
120:LPRINT "#####"
    #####
130:LPRINT " Mo Di Mi Do
    Fr Sa So"
140:LPRINT "-----"
150:Z$(0)=""
160:FOR I=1 TO 7
170:IF D<1 THEN LET Z$(0
    )=Z$(0)+" ":"GOTO 2
    00
180:IF D>H GOTO 220
190:Z$(0)=Z$(0)+LEFT$ (""
    ,(D<(10)+1)+STR$ D
    )
200:D=D+1
210:NEXT I
220:LPRINT Z$(0)
222:IF D<H GOTO 150
230:LPRINT "-----"
240:LPRINT "#####
    #####"
250:END
260:DATA "Januar",31,"Fe
    bruar",28,"Maerz",31
    ,"April",30,"Mai",31
    ,"Juni",30,"Juli"
270:DATA 31,"August",31,
    "September",30,"Okto
    ber",31,"November",3
    0,"Dezember",31
```

5.2 Berechnung beweglicher Feiertage

Dieses Programm berechnet die beweglichen Festtage sowie die Wochentage datumsfester Feiertage (wie z.B. 1. Mai) für ein beliebiges Jahr zwischen 1901 und 2099.

Das Programm wird mit "RUN" oder 'DEF' "A" gestartet. Die Berechnung nimmt ca. 12 Sekunden in Anspruch; anschließend werden der Reihenfolge nach folgende zehn Feiertage ausgegeben:

NEUJAHR, ASCHERmittwoch, OSTERN (Ostersonntag), 1. MAI, Christi HIMMELfahrt, PFINGSTsonntag, 17. JUNI, BUSTAG, ADVENT (1. Advent), 24. DEZember.

(Die im Programm verwendeten Abkürzungen sind mit Großbuchstaben hervorgehoben.)

Der Ausgabevorgang kann mit 'DEF' "D" wiederholt werden. Wird vorher das Kommando "PRINT=LPRINT" eingegeben, erfolgt die Ausgabe über den angeschlossenen Drucker.

Zur Kontrolle erscheinen hier die Daten für die nächsten vier Jahre:

| | 1987 | 1988 | 1989 | 1990 |
|---------------------|--------|--------|--------|--------|
| Neujahr | Do | Fr | So | Mo |
| Aschermittwoch | 4.3. | 17.2. | 8.2. | 28.2. |
| Ostersonntag | 19.4. | 3.4. | 26.3. | 15.4. |
| 1. Mai | Fr | So | Mo | Di |
| Christi Himmelfahrt | 28.5. | 12.5. | 4.5. | 24.5. |
| Pfingstsonntag | 7.6. | 22.5. | 14.5. | 3.6. |
| 17. Juni | Mi | Fr | Sa | So |
| Buß- und Bettag | 18.11. | 16.11. | 22.11. | 21.11. |
| 1. Advent | 29.11. | 27.11. | 3.12. | 2.12. |
| 24. Dezember | Do | Sa | So | Mo |

```

1:"A"
3:"Berechnung der Feie
rtage eines beliebig
en Jahres zwischen 1
901 und 2099
5:CLEAR
10:DIM F$(9),W$(0)*14,M
    $(0)*24
20:W$(0)="MoDiMiDoFrSaS
    o"
30:M$(0)="3128313031303
    13130313031"
40:INPUT "Jahr: ";J
50:IF J<1981 OR J>2099
    THEN PRINT " NUR 190
    1-2099";GOTO 40
60:D=1+(J-1981)+C INT (
    (J-1981)/4))
70:D=D- INT (D/7)*7
80:S=(J AND 3)=0
85:IF S THEN LET M$(0)=
    LEFT$ (M$(0),3)+"9"+
    RIGHT$ (M$(0),20)
90:T=D+1:IF T>7 THEN
    LET T=T-7
100:I=0:N$="Neujahr":
    GOSUB "WT"
110:A=J- INT (J/19)*19
120:Z=204-11*A
130:C=Z- INT (Z/30)*30
140:IF C=28 OR C=29 THEN
    LET C=C-1
150:Z=J+ INT (J/4)+C-13
160:E=28+C-(Z- INT (Z/7)
    *7)
170:T=E+59+S
180:I=2:N$="Ostern ":
    GOSUB "DAT"
190:T=T-46
200:I=1:N$="Ascher ":
    GOSUB "DAT"
210:T=T+85
220:I=4:N$="Himmel ":
    GOSUB "DAT"
230:T=T+10
240:I=5:N$="Pfingst ":
    GOSUB "DAT"
250:T=D+2+S:IF T>7 THEN
    LET T=T-7
260:I=3:N$="1.Mai  ":
    GOSUB "WT"
270:T=D+7+S:IF T>7 THEN
    LET T=T-7
280:I=6:N$="17.Juni ":
    GOSUB "WT"
290:T=D+1+S:IF T>7 THEN
    LET T=T-7
300:I=9:N$="24.Dez ":
    GOSUB "WT"
310:X=T
320:T=358-X-21+S+7*(X=7)
330:I=8:N$="Advent ":
    GOSUB "DAT"
340:T=T-11
350:I=7:N$="Busstag ":
    GOSUB "DAT"
360:D"
365:BEEP 1
370:PRINT " ****";STR$
    J;"***"
380:FOR I=0 TO 9
390:PRINT F$(I)
400:NEXT I
410:END
420:"WT"
430:F$(I)=N$+" "+MID$ (
    W$(0),(T-1)*2+1,2)
440:RETURN
450:"DAT"
460:U=T
470:M=0
480:Y=U
490:U=VAL MID$ (M$(0),
    2*M+1,2)
500:M=M+1
510:IF U>0 GOTO 480
520:F$(I)=N$+" "+STR$ Y
    +"."+STR$ M+"."
530:RETURN

```

S
H
ALLES
FÜR
COMPUTER

5.3 Berechnung von Sonnenauf- und -untergang

Dieses Programm berechnet nach Eingabe des Datums und der geographischen Breite des Beobachtungsortes die Auf- und Untergangszeit der Sonne in der MOZ (mittlere Ortszeit) des Beobachtungspunktes.

Die berechneten Zeiten weichen je nach geographischer Breite um einige Minuten von den tatsächlichen Zeiten ab (siehe Vergleich auf der nächsten Seite). Zusätzlich ist noch eine weitere Korrektur erforderlich, um die MOZ in die MEZ bzw. Sommerzeit umzurechnen. Hierfür sind $4*(15-\lambda)$ Minuten zu der MOZ zu addieren, um die mitteleuropäische Zeit zu erhalten (bei Sommerzeit noch eine Stunde zusätzlich).

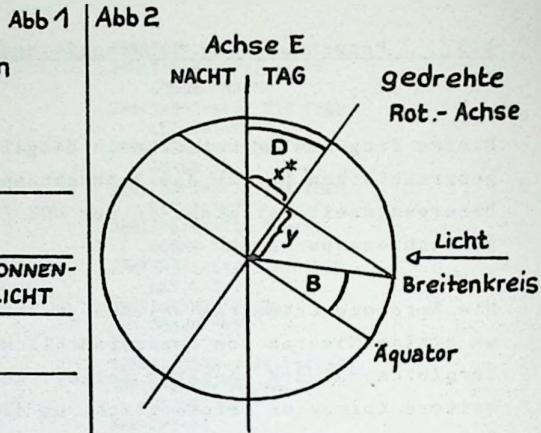
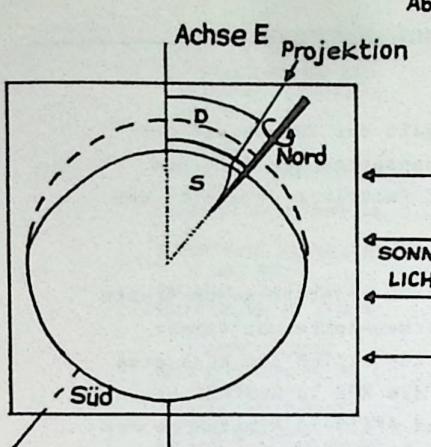
λ ist hierbei die geographische Länge des Beobachtungsortes in Grad östlich von Greenwich.

Im folgenden soll erklärt werden, wie das Programm arbeitet:

Bekanntlich ist die Äquatorebene der Erde leicht gegen die Ekliptikebene (Ebene der Erdumlaufbahn) geneigt. Der Neigungswinkel S beträgt 23.45 Grad.

Da die Rotationsachse der Erde raumstabil ist, treffen im Verlauf eines Jahres die Sonnenstrahlen unter einem sich ständig ändernden Winkel auf der Erde. Dieser Winkel ist die Deklination der Sonne D und kann definiert werden als Winkel zwischen der senkrecht auf der Ekliptik stehenden Achse E und der Projektion der Erdrotationsachse auf diejenige Ebene, auf welcher sich Sonne und Erde befinden und die senkrecht auf der Ekliptikebene steht (siehe Abb. 1).

Dieser Winkel D ist eine Sinusfunktion der Zeit mit S als Amplitude (Zeile 60), da von einem Bezugssystem, welches sich mit der Erde um die Sonne dreht, aus betrachtet sich die



Rotationsachse der Erde im Laufe eines Jahres einmal vollständig mit konstanter Winkelgeschwindigkeit um die Achse E dreht, sofern man eine kreisförmige Erdumlaufbahn voraussetzt.

Zur weiteren Berechnung betrachte man Abb. 2, welche den Schnitt der um die Verbindungsachse Sonnenmittelpunkt-Erdmittelpunkt gedrehten Erde mit der in Abb. 1 eingezeichneten Ebene darstellt.

Der Radius der Erdkugel sei auf 1 normiert, um die Rechnung zu vereinfachen. Interessant ist für uns die Strecke X^* .

Man sieht:

$$X^* = y \tan D$$

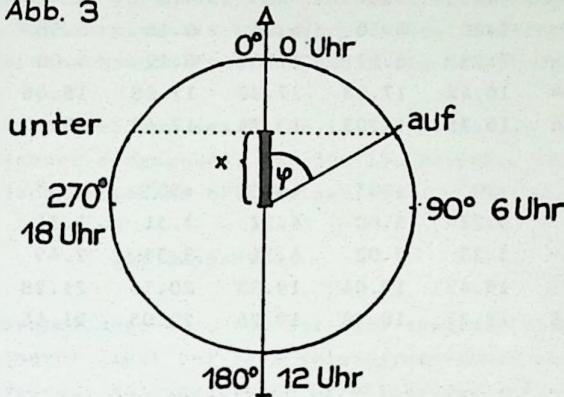
$$= \sin B \tan D$$

Nun soll X^* normiert werden, in dem wir X^* durch den Radius des Breitenkreises des Beobachtungsortes teilen ($= \cos B$).

Man erhält: $X := X^*/\cos B = \tan D \sin B/\cos B = \tan D \tan B$
(Zeile 70)

Wir sind jetzt praktisch fertig. Wir betrachten nun in Abb. 3 den normierten Breitenkreis:

Abb. 3



Ist x größer als 1, so liegt der Breitenkreis vollständig in der beleuchteten Erdhälfte (Polartag, Zeile 80); ist x kleiner als -1, so haben wir Polarnacht (Zeile 90).

Sonst berechnen wir den Winkel ϕ als $\arccos X$ (Zeile 100), der sich sofort als Auf- und Untergangszeit der Sonne in wahrer Ortszeit (Kulmination der Sonne um 12.00 Uhr) umrechnen lässt.

Zur Umrechnung in mittlere Ortszeit wird die monatliche Zeitgleichungstabelle in der DATA-Zeile 220 benutzt.

Aufgrund dieser groben Umrechnung und einiger Vereinfachungen (wir haben eine kreisförmige Umlaufbahn vorausgesetzt; in Wirklichkeit ist sie elliptisch) ergeben sich Ungenauigkeiten bei der Rechnung. Deshalb werden der Einfachheit halber alle Monate mit 30 Tagen Länge festgesetzt.

Trotzdem ist es interessant, einmal die Auf- und Untergangszeiten für verschiedene Jahreszeiten und Breiten durchzuspielen.

Vergleich zwischen berechneten und wahren (/10/) Zeiten.

Beispiel: 23. Juni

| Breite: | -50° | -40° | -30° | -20° | -10° | 0° | |
|---------|-------|-------|-------|-------|-------|-------|--------|
| auf: | 8.00 | 7.22 | 6.56 | 6.34 | 6.16 | 5.58 | (wahr) |
| auf: | 8.04 | 7.25 | 6.57 | 6.36 | 6.17 | 6.00 | (ber.) |
| unter: | 16.04 | 16.42 | 17.08 | 17.30 | 17.48 | 18.06 | (wahr) |
| unter: | 15.56 | 16.35 | 17.03 | 17.24 | 17.43 | 18.00 | (ber.) |

| Breite: | +10° | +20° | +30° | +40° | +50° | +60° | |
|---------|-------|-------|-------|-------|-------|-------|--------|
| auf: | 5.41 | 5.22 | 5.00 | 4.32 | 3.51 | 2.36 | (wahr) |
| auf: | 5.42 | 5.23 | 5.02 | 4.34 | 3.55 | 2.45 | (ber.) |
| unter: | 18.23 | 18.42 | 19.04 | 19.33 | 20.13 | 21.28 | (wahr) |
| unter: | 18.18 | 18.37 | 18.58 | 19.26 | 20.05 | 21.15 | (ber.) |

```

1: "A"
3: "Berechnung von Sonn
enau- und -untergaa
ngen an beliebigen T
agen und Orten
5: CLEAR
6: RESTORE
7: DEGREE
8: DIM Z(11)
9: FOR I=0 TO 11: READ Z
   (I): NEXT I
10: INPUT "Tag: "; T
15: IF T<1 OR T>31 THEN
    GOTO 10
20: INPUT "Monat: "; M
25: IF M<1 OR M>12 THEN
    GOTO 20
30: INPUT "Breite: "; B
35: IF B<-90 OR B>90
    THEN GOTO 30
37: IF ABS B=90 THEN LET
    B=SGN B*89.99
40: S=23.45
50: J=(M-1)*30+T-81
60: D=S*SIN J
70: X=TAH D*TAN B
80: IF X>1 THEN PRINT "
   POLARTAG": GOTO 13
90: IF X<-1 THEN PRINT "
   POLARHACHT": GOTO
131
100: A= INT (ACS X/360*14
40)
110: U=1440-A
115: A=A-Z(M-1): U=U-Z(M-1
)
116: IF U>=1440 THEN LET
    U=U-1440
117: IF A<0 THEN LET A=A+
1440
120: I=A: Y$="AUF": GOSUB 2
   00
130: I=U: Y$="UNTER": GOSUB
   200
131: Y$=""
132: INPUT "NOCHMAL ? "; Y
   $
135: IF Y$="J" GOTO 10
137: IF Y$<>"N" GOTO 132
140: END
200: PRINT " "; Y$; ":";
      USING "###.##"; DMS (
      (I+.5)/60)
210: RETURN
220: DATA -10,-14,-9,0,4,
   , -6,-4,5,14,15,5

```

Dieses Programm berechnet für jedes beliebige Datum die Lage des Mondterminators (Grenze zwischen beleuchteter und unbeleuchteter Mondhälfte) und damit die Mondphase.

Dazu wird einfach nur das Datum in Tag/Monat/Jahr getrennt in den Rechner eingegeben. Für den 15. Dezember 1987 erhält man als Rechenergebnis beispielsweise:

(24W Copernicus

Das "("-Symbol sagt aus, daß der Mons abnimmt (zunehmend = ")"), der Terminator liegt bei 24°W selenographischer Länge (E=Ost (East), W=West, astronautische Orientierung) und verläuft damit durch das Gebiet des Ringgebirges Copernicus, welches sich deshalb an diesem Tag gut für Beobachtungen eignet.

Für genauere Formationsangaben studiere man entsprechende Karten oder Atlanten (z.B. /11/).

Das Programm benutzt als Bezugspunkt die Mondphase vom 31. Dezember 1984 und nimmt einen konstanten Zeitabstand zwischen zwei aufeinanderfolgende Neumondstellungen an (dieser Abstand wird als synodischer Monat bezeichnet), was leider nicht der Fall ist.

Trotzdem arbeitet das Programm recht genau; die Ungenauigkeiten sind geringer als der im Verlauf eines Tages vom Terminator zurückgelegte Winkel.

Zur Veranschaulichung folgen einige Beispiele für Juni 1987; die als "wahr" bezeichneten Angaben stammen wieder, wie schon im Kapitel 5.3, aus dem Buch /10/.

| Datum | Lage des Terminators ber. | | Formation wahr |
|---------------|------------------------------|--------|----------------------|
| 01. Juni 1987 | 36°E | 40.0°E | Mare Nectaris |
| 02. Juni 1987 | 24°E | 27.8°E | Mare Tranquillitatis |
| 03. Juni 1987 | 12°E | 15.6°E | Mare Serenitatis |
| 04. Juni 1987 | 0° | 3.3°E | Mare Vaporum |
| 05. Juni 1987 | 12°W | 8.9°W | Mare Nubium |
| 06. Juni 1987 | 24°W | 21.1°W | Copernicus |
| 07. Juni 1987 | 36°W | 33.3°W | Mare Nubium |
| 08. Juni 1987 | 48°W | 45.5°W | Aristarchus |
| 09. Juni 1987 | 60°W | 57.7°W | Grimaldi |
| 10. Juni 1987 | 72°W | 68.9°W | Riccioli |

```

1;"A"
3;"B. Rechnung des Mond
terminators fuer jed
es beliebige Datum
5:CLEAR :RESTORE
18:DIM M$(0)*24,F$(17)*
18
20:FOR I=0 TO 17
30:READ F$(I)
40:NEXT I
50:V=29.53058912;"DAUER
DES SYNOUDISCHEN MON
ATS"
60:X=11;"MORGENTERMINAT
OR AM 31.12.1984"
70:M$(0)="0031283130313
03131303130"
80:INPUT "Tag: ";T
90:IF T>31 GOTO 80
100:INPUT "Monat: ";M
110:IF M>12 GOTO 100
120:INPUT "Jahr: ";J
140:J=J-1985
150:IF (J AND 3)=3 THEN
    LET M$(0)=LEFT$ (M$(
    0),5)+"9"+RIGHT$ (M$(
    0),18)
160:D=0
170:FOR I=1 TO M
180:D=D+VAL MID$ (M$(0),
-1+I*2,2)
190:NEXT I
200:D=J*365+ INT (J/4)+D
+T
210:P=(D/V- INT (D/V))*3
60+X
220:IF P>360 THEN LET F
P=360
230:N=(P>270) OR (P<=90)
240:IF N THEN IF P>270
    THEN LET P=P-360
250:IF 1-N THEN LET P=P-
180
260:PRINT CHR$ (40+N);"
";STR$ INT ABS P;
CHR$ (87-(P<0)*18);"
";F$C INT ((P+90)/1
0))
270:GOTO 70
280:DATA "Mare Marginis"
,"Gauss","Lansrenus"
,"Mare Crisium","Mar
e Fecunditatis"
290:DATA "Mare Nectaris"
,"Mare Tranquill.","
Mare Serenitatis","M
are Vaporum"
300:DATA "Ptolemaeus","M
are Nubium","Coperni
cus","Mare Humorum",
"Aristarchus"
310:DATA "Schickard","Gr
imaldi","Riccioli",""
Einstein"

```

5.5 Berechnung komplexer Wurzeln

Eine komplexe Zahl $z = a + ib$ hat genau n verschiedene n -te Wurzeln, welche sich durch folgende Formel berechnen lassen (siehe dazu auch /12/ und /13/):

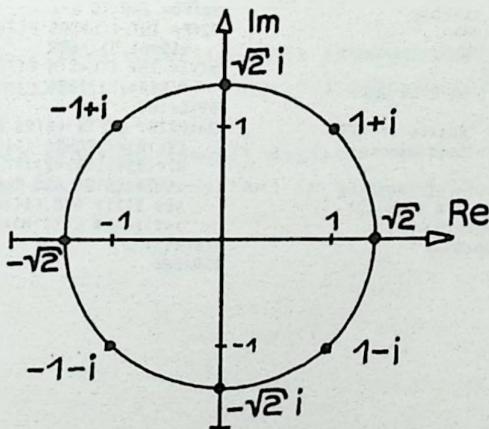
$$\sqrt[n]{z} = \sqrt[n]{|z|} \exp(i(\phi + k2\pi)/n), \quad \text{mit } k = 0, \dots, n-1 \\ \text{und } \phi := \arg z$$

Hierbei gilt: $\exp i\xi \equiv \cos \xi + i \sin \xi$ (Eulersche Identität)

Geometrisch sind diese n Zahlen viel einfacher zu bestimmen, wenn man sie graphisch als Punkte in der Gaußschen Zahlenebene darstellt.

Dort bilden die n verschiedenen Lösungen die Ecken eines regelmäßigen n -Ecks mit dem Außenradius $\sqrt[n]{|z|}$, wobei für eine Ecke w gilt: $\arg w = (\arg z)/n$.

Die nachfolgende Abbildung stellt auf diese Weise alle acht komplexen Lösungen der Gleichung $z^8 = 16$ in der Gaußschen Zahlenebene dar.



Es folgen einige Anwendungsbeispiele. Die Angabe der Lösungen erfolgt dabei im selben Format, in der die Zahlen auch vom Programm ausgegeben werden.

#1: Wurzelexponent: 8, Realteil: 16, Imaginärteil: 0.

Hierfür gibt es folgende Lösungen (siehe Abbildung auf der vorhergehenden Seite):

1.414 1+i +1.414i -1+i -1.414 -1-i -1.414i 1-i

#2: Wurzelexponent: 16, Realteil: 2, Imaginärteil: 2.

Hierfür gibt es folgende 16 Lösungen:

1.066 + 0.052i 0.965 + 0.456i 0.717 + 0.791i 0.36 + 1.005i
-0.052 + 1.066i -0.456 + 0.965i -0.791 + 0.717i -1.005 + 0.36i
-1.066 - 0.052i -0.965 - 0.456i -0.717 - 0.791i -0.36 - 1.005i
0.052 - 1.066i 0.456 - 0.965i 0.791 - 0.717i 1.005 - 0.36i

```
1;"A"
3:"Berechnung aller Wu
rzeln einer komplexe
n Zahl
5:CLEAR :RADIAN
7:DIM P(20)
10:INPUT "Wurzelexponen
t: ";E
15:IF E<1 OR E>20 GOTO
    18
20:INPUT "Realteil: ";R
30:INPUT "Imaginärteil
: ";I
40:P(0)=ATN (I/(R+10^-1
    0)):IF R<0 THEN LET
        P(0)=I+P(0)
45:P(0)=P(0)/E
50:L=f(R*R+I*I):L=L^(1/
    E)
60:FOR J=1 TO E-1
70:P(J)=P(J-1)+2*I/E
80:NEXT J
90:FOR J=0 TO E-1
92:R= INT ((L*COS P(J))
    *1000+.5)/1000
97:I= INT ((L*SIN P(J))
    *1000+.5)/1000
99:Q=105
100:PRINT LEFT$ (STR$ R,
    (R<>0)*7);CHR$ ((I<>
    0)*(43+(I<>2)));
    LEFT$ (STR$ ABS I,(((
    ABS I)>1) AND (I<>0)
    )*7);CHR$ ((I<>0)*Q)
110:NEXT J
120:END
```

5.6 Umwandlung arabischer Zahlen in das römische Zahlensystem

Das römische Zahlensystem ist ein sogenanntes additives Zahlensystem, bei dem die den einzelnen Zahlzeichen zugeordneten Zahlenwerte addiert werden müssen, um die dargestellte Zahl zu berechnen.

Folgende Werte sind hierbei den folgenden Zeichen zugeordnet:

$$I = 1 , V = 5 , X = 10 , L = 50 , C = 100 , D = 500 , M = 1000$$

Dabei treten die einzelnen Zahlzeichen in der Reihenfolge ihres Zahlenwerts auf (das Symbol mit dem größeren Wert kommt vor dem mit dem kleineren).

$$\text{Somit ist beispielsweise } LXXXVII = 50+10+10+10+5+1+1 = 87.$$

Treten einmal vier gleiche Zahlzeichen auf, wie z.B. in $VIII = 5+1+1+1 = 9$, so wird aus Platzgründen eine subtraktive Schreibweise bevorzugt, indem man das Symbol mit dem kleineren Wert vor das mit dem größeren stellt.

$$\text{Man erhält dann } IX = 10-1 = 9.$$

Beide Verfahren treten natürlich auch in einer Zahl gemischt auf. So ist $MCMLXXXVII = 1000+1000-100+50+10+10+10+5+1+1 = 1987$.

Das nachfolgende Programm wandelt umgekehrt beliebige arabische Zahlen (herkömmliches Zahlensystem) in römische Zahlen um.

Die folgenden Beispiele können zur Kontrolle der korrekten Eingabe des Programms herangezogen werden.

| | | |
|-------------------|-------------------|--------------------|
| 12 = XII | 1888=MDCCLXXXVIII | 1987 = MCMLXXXVII |
| 503 = DIII | 509 = DIX | 520 = DXX |
| 1666 = MDCCDXCVI | 90 = XC | 2222 = MMCCXXII |
| 49 = XLIX | 1234 = MCCXXXIV | 1988 = MCMLXXXVIII |
| 4321 = MMMMCCCXXI | 1999 = MCMXCIX | 1001 = MI |
| 5555 = MMMMMMDLV | 1989 = MCMLXXXIX | 253 = CCLIII |

```

1;"A"
3:"Darstellung beliebi-
ger Zahlen im roemis-
chen Zahlensystem
5:CLEAR
10:DIM R$(0)
20:Z$="IVXLCDM"
25:DATA 1000,500,100,50
,10,5,1
30:INPUT "Zahl: ";X
32:IF X=0 GOTO 30
35:RESTORE
40:P=8:R$(0)=``
42:READ D:P=P-1:F=0
45:F=C>0
47:C=0
50:IF X<D GOTO 42
60:R$(0)=R$(0)+MID$(Z$
,P,1)
70:C=C+1
80:X=X-D
85:Y=0
90:IF C=4 AND P>7 THEN
LET R$(0)=LEFT$(R$(0),
0),LEN R$(0)-4-F)+_
MID$(Z$,P,2):LET Y=
1
95:IF Y AND F THEN LET
R$(0)=LEFT$(R$(0),
LEN R$(0)-1)+MID$(Z$
,P+2,1)
100:IF X>0 GOTO 50
110:PRINT R$(0)
120:GOTO 30

```



5.7 Integration einer Funktion

Dieses kurze Programm eignet sich zur Berechnung des bestimmten Integrals einer beliebigen Funktion innerhalb beliebiger Grenzen.

Die Funktion wird in Zeile 60 gespeichert (im folgenden Listing ist es die Funktion $y=x^2$).

Nach dem Starten des Programms müssen dann nur noch die untere und die obere Integrationsgrenze eingegeben werden.

Der Integrationsbereich wird in 100 Streifen aufgeteilt, deren Flächen berechnet und aufsummiert werden. Anschließend wird die Gesamtfläche F auf drei Stellen hinter dem Komma gerundet ausgegeben.

Es folgen einige Beispiele für bestimmte Integrale der Form

$$\begin{array}{lll} b & & \\ F = \int_a^b f(x)dx : & f(x)=x^2, a=0, b=1 : & F = 1/3 \\ a & f(x)=\sin x, a=0, b=\pi : & F = 2 \\ & f(x)=\cos x, a=0, b=\pi : & F = 0 \\ & f(x)=e^x, a=0, b=1 : & F = e-1 \\ & f(x)=\arcsin x, a=0, b=1 : & F = \pi/2-1 \end{array}$$

```
1;"A"
3:"Integration einer beliebigen Funktion innerhalb beliebiger Grenzen
5:CLEAR
10:INPUT "Untere Grenze
      :,U,"Obere Grenze:"
      ,0
20:D=(U-U)/100
30:X=U-D/2
40:X=X+D
60:Y=X*X
70:F=F+Y*D
80:IF (X+D)<0 GOTO 40
85:BEEP 2
90:PRINT "F = ",.001*
      INT (1000*F+.5)
100:END
```

5.8 Primzahloberechnung nach Eratosthenes

Mit diesem Verfahren lassen sich Primzahlen auf eine zugleich einfache und schnelle Art und Weise berechnen.

Man schreibe alle Zahlen von 2 bis zu der höchsten Zahl auf, die man auf Primzahleneigenschaft untersuchen möchte. Nun nimmt man die 2 und streicht alle Vielfachen von 2 durch, da diese Zahlen als Primzahlen nicht mehr in Frage kommen.

Anschließend geht man zur nächsten Zahl über und schaut, ob sie durchgestrichen worden ist. Ist das nicht der Fall, so muß es sich um eine Primzahl handeln. Ist sie dagegen durchgestrichen worden, so ist sie ein Vielfaches einer anderen Zahl, somit durch diese Zahl ganzzahlig teilbar und dementsprechend keine Primzahl.

Nachdem auch hier wieder alle Vielfachen durchgestrichen worden sind, fährt man mit der nächsten Zahl fort, prüft diese auf Primzahleneigenschaft etc.

Das Programm benutzt den größten Teil des Standardvariablen-speichers als Zahlenschema, wo jeder ungeraden Zahl zwischen 3 und 337 ein Byte zugeordnet wird.

Die geraden Zahlen wurden bereits entfernt, da sie keine Primzahlen sind (abgesehen von der 2, die gesondert aufgenommen wird).

Der Inhalt der Bytes ist anfangs durchgehend Null; dafür sorgt der "CLEAR"-Befehl, welcher die Variablen löscht. Anschließend werden "durchgestrichene" Bytes mit einer Eins gekennzeichnet.

Mit dem Programm werden dann folgende 68 Primzahlen bestimmt:

| | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|
| 2 | 3 | 5 | 7 | 11 | 13 | 17 |
| 19 | 23 | 29 | 31 | 37 | 41 | 43 |
| 47 | 53 | 59 | 61 | 67 | 71 | 73 |
| 79 | 83 | 89 | 97 | 101 | 103 | 107 |
| 109 | 113 | 127 | 131 | 137 | 139 | 149 |
| 151 | 157 | 163 | 167 | 173 | 179 | 181 |
| 191 | 193 | 197 | 199 | 211 | 223 | 227 |
| 229 | 233 | 239 | 241 | 251 | 257 | 263 |
| 269 | 271 | 277 | 281 | 283 | 293 | 307 |
| 311 | 313 | 317 | 331 | 337 | | |

```
1;"A"
3;"Bestimmung der Prim
zahlen zwischen 2 un
d 338 mit dem Sieb d
es Eratosthenes
5:CLEAR
7:C=&FB10:D=2
8:WAIT 60
9:PRINT " A : 2."
10:FOR A=3 TO 337 STEP
    2
15:B=0
20:IF PEEK (C+(A-1)/2)=
    0 THEN PRINT " ";
    STR$ D;" : ";A:B=1
    :D=D+1
30:IF B AND A<170 THEN
    FOR B=A TO 338 STEP
        2*A:POKE (C+(B-1)/2)
        ,1:NEXT B
40:NEXT A
50:END
```



Beim Tic-Tac-Toe-Spiel geht es darum, in einem Quadrat aus neun Feldern durch abwechselndes Setzen von Steinen drei eigene Steine in eine Zeile, Spalte oder Diagonalen zu bekommen.

Es gibt verschiedene Verfahren, dem Computer das Tic Tac Toe beizubringen. Einen Vergleich verschiedener Vorgehensweisen zeigt /14/.

Für das folgende Programm zeichne man eine 3*3-Matrix und nummeriere die neun Felder folgendermaßen durch:

| | | |
|---|---|---|
| 8 | 3 | 4 |
| 1 | 5 | 9 |
| 6 | 7 | 2 |

Man beachte, daß man mit dieser, anfangs etwas obskur erscheinenden Art der Nummerierung ein magisches Quadrat erhält. Die Summe aller Zeilen, Spalten und Diagonalen ist immer gleich 15.

Dadurch ist es für den Rechner einfacher festzustellen, ob bereits zwei oder sogar drei Steine desselben Spielers in einer Zeile, Spalte oder Diagonale stehen.

```

1: "A"
5: "Tic Tac Toe
7: CLEAR
10: DIM F(9), X(4), Y(4)
20: PAUSE "Wer faengt an
    ?"
30: INPUT "Spieler/Computer (S/C) "; S$
40: IF A$ = "C" GOTO 200
50: IF A$ <> "S" GOTO 20
60: GOSUB "ENDE"
70: INPUT "Feld-Nr.: "; I
80: IF F(I) = 0 THEN LET F
    (I) = 1: LET X(H) = I: LET
    N = H + 1: GOTO 100
90: PRINT "Feld bereits
    besetzt": GOTO 70
100: GOSUB "ENDE"
200: SUC = 0
205: FOR I = 0 TO M - 2
210: FOR J = i + 1 TO M - 1
215: IF SUC GOTO 240
220: B = 15 - Y(I) - Y(J)
230: IF B > 0 AND B <= 9 THEN
    IF F(B) = 0 THEN LET S
    UC = 1: LET Y(M) = B: LET
    M = M + 1: LET F(B) = 1
240: NEXT J
250: NEXT I
255: IF SUC GOTO 400
260: FOR I = 0 TO M - 2
270: FOR J = i + 1 TO M - 1
275: IF SUC GOTO 300
280: B = 15 - X(I) - X(J)
290: IF B > 0 AND B <= 9 THEN
    IF F(B) = 0 THEN LET S
    UC = 1: LET Y(M) = B: LET
    M = M + 1: LET F(B) = 1
300: NEXT J
310: NEXT I
320: IF SUC GOTO 400
330: IF F(5) = 0 THEN LET S
    UC = 1: LET Y(M) = 5: LET
    M = M + 1: LET F(5) = 1:
    GOTO 400
340: FOR I = 2 TO 8 STEP 2
345: IF SUC GOTO 360
350: IF F(I) = 0 THEN LET S
    UC = 1: Y(M) = I: LET M = M +
    1: LET F(I) = 1
360: NEXT I
365: IF SUC GOTO 400
370: FOR I = 1 TO 9 STEP 2
375: IF SUC GOTO 390
380: IF F(I) = 0 THEN LET S
    UC = 1: LET Y(M) = I: LET
    M = M + 1: LET F(I) = 1
390: NEXT I
400: PRINT "Ich setze auf
    Feld "; STR$ Y(M - 1);
    "."
405: GOTO 60
410: "ENDE"
420: FOR I = 0 TO M - 3
430: FOR J = I + 1 TO M - 2
440: FOR K = J + 1 TO M - 1
450: IF X(I) * X(J) * X(K) < 0
    THEN IF X(I) + X(J) + X(
    K) = 15 THEN BEEP 1:
    PRINT "Du hast gewo
    nnen.": END
460: NEXT K: NEXT J: NEXT I
470: FOR I = 0 TO M - 3
480: FOR J = I + 1 TO M - 2
490: FOR K = J + 1 TO M - 1
500: IF Y(I) * Y(J) * Y(K) < 0
    THEN IF Y(I) + Y(J) + Y(
    K) = 15 THEN BEEP 1:
    PRINT "Ich habe gewo
    nnen.": END
510: NEXT K: NEXT J: NEXT I
520: S = 0
530: FOR I = 1 TO 9: S = S + F(I
    ): NEXT I
540: IF S = 9 THEN BEEP 1:
    PRINT "Unentschieden
    .": END
550: RETURN

```

Als magisches Quadrat wird eine quadratische Matrix natürlicher Zahlen bezeichnet, welche die besondere Eigenschaft hat, daß die Summenbildung über alle Spalten, Zeilen und Diagonalen immer zu ein und demselben Ergebnis führt.

Am bekanntesten ist wohl dasjenige magische Quadrat, welches Albrecht Dürer im Jahre 1514 in seinem Kupferstich "Melancolia I" aufgenommen hat:

| | | | |
|----|----|----|----|
| 16 | 3 | 2 | 13 |
| 5 | 10 | 11 | 8 |
| 9 | 6 | 7 | 12 |
| 4 | 15 | 14 | 1 |

Die gemeinsame Summe aller Diagonalen, Spalten und Zeilen ist 34.

Das folgende Programm erzeugt ein magisches Quadrat mit einer beliebigen ungeraden Zeilenanzahl. Es benutzt dazu den folgenden Algorithmus:

"Schreibe eine 1 in die Mitte der obersten Zeile und scheibe die darauffolgenden Zahlen jeweils immer in das Feld, welches der vorhergegangenen Zahl nach links oben angrenzt. Trifft man auf den Rand, so mache man beim gegenüberliegenden Rand weiter. Trifft man auf ein bereits besetztes Feld, so schreibe man die nächste Zahl nicht nach links oben, sondern in das nach unten angrenzende Feld."

Ein nach diesem Algorithmus erzeugtes, aus fünf Zeilen und Spalten bestehendes magisches Quadrat sieht dann folgendermaßen aus:

| | | | | |
|----|----|----|----|----|
| 15 | 8 | 1 | 24 | 17 |
| 16 | 14 | 7 | 5 | 23 |
| 22 | 20 | 13 | 6 | 4 |
| 3 | 21 | 19 | 12 | 10 |
| 9 | 2 | 25 | 18 | 11 |

Die gemeinsame Summe ist 65. Man kann mit diesem Beispiel auch gleich überprüfen, ob man das Programm korrekt eingegeben hat.

```

1:"A"
3:"Erzeugung eines mag
ischen Quadrats mit
ungerader Zeilenanzahl
h1
5:CLEAR
10:INPUT "ZEILEANZAHL:
";N
20:N=ABS N
30:IF N AND 1=0 THEN
    PRINT "FEHLER. N IST
    GERADE.":GOTO 10
35:DIM F(N-1,N-1)
80:S=2:I=0:J= INT ((N-1
    )/2)
90:F(I,J)=1:Q=J:F(N-1,Q
    )=N
100:IF S>N/2+1 GOTO 20
    7
110:IF I<1 THEN LET K=N-
        1:GOTO 130
120:K=I-1
130:IF JK<1 THEN LET M=N-
        1:GOTO 150
140:M=J-1
150:IF F(K,M)<>0 THEN
    LET I=I+1- INT ((I+1
    )/N)*N:GOTO 190
160:I=K
170:IF J<1 THEN LET J=N-
        1:GOTO 190
180:J=J-1
190:F(I,J)=S
195:F(2*Q-I,2*Q-J)=N*N+1
    -S
200:S=S+1
205:GOTO 100
207:BEEP 2
210:PRINT "MAGISCHES QUA
    DRAT":PRINT "ZEILENA
    NZAHL: ";STR$ N
220:FOR I=0 TO N-1
230:FOR J=0 TO H-1
240:PRINT "(";STR$ (I+1)
    ;",";STR$ (J+1);")"
    = "STR$ F(I,J)
250:NEXT J
260:NEXT I
270:END

```



Dieses Programm stellt eine kleine Spielerei dar:
Es vertauscht die Buchstaben innerhalb eines beliebigen
Wortes und gibt alle möglichen Kombinationen aus.

Hat ein Wort n Buchstaben, so gibt es $n!$ (sprich: n Fakultät)
verschiedene Vertauschungsmöglichkeiten. Hierbei gilt:
 $n! = 1*2*3* \dots * (n-1)*n$.

Maximal zugelassen sind sieben Buchstaben als Wortlänge.
Dem entsprechen $1*2*3*4*5*6*7 = 5040$ verschiedene Kombinationen.

Im unteren Beispiel sind die $5! = 120$ verschiedenen Permutationen der Buchstaben des Wortes "SHARP" aufgeführt.

```
1:"A"
3;"Ausgabe aller Kombinationen der Buchstaben (bis zu 7) eines beliebigen Wortes
5:CLR
7:WAIT 100
10:INPUT "WORT: ";W$
15:E=LEN W$-1
20:DIM W$(E)*1,0$(E)*1,
    Z(E)
25:Q=1
30:FOR I=0 TO E
40:W$(I)=MID$ (W$,I+1,1
    )
45:Z(I)=1
50:NEXT I
60:FOR I=0 TO E
70:0$(I)=""
80:NEXT I
90:FOR I=0 TO E
100:K=-1
110:FOR J=1 TO Z(I)
120:K=K+1
130:IF 0$(K)<>"" GOTO 12
    0
140:NEXT J
150:0$(K)=W$(I)
160:NEXT I
170:P$="."
180:FOR I=0 TO E
190:P$=P$+0$(I)
200:NEXT I
210:PRINT USING "#####";
    Q;" ";P$
215:Q=Q+1
220:I=E
225:I=I-1
227:IF I<0 GOTO 255
230:Z(I)=Z(I)+1
240:IF Z(I)>E+1-I THEN
        LET Z(I)=I:GOTO 225
250:GOTO 60
255:BEEP 3
260:PRINT "DAS WAR ALLES
    !"
270:END
```



| | | | | | |
|----|-------|----|--------|-----|-------|
| 1 | SHARP | 41 | RSPHA | 81 | RHASP |
| 2 | SHAPR | 42 | PSRHA | 82 | PHASR |
| 3 | SHRAP | 43 | ASRPH | 83 | RHPSA |
| 4 | SHPAR | 44 | ASPRH | 84 | PHRSA |
| 5 | SHRPA | 45 | RSAPH | 85 | ARHSP |
| 6 | SHPRA | 46 | PSARH | 86 | APHSR |
| 7 | SAHRP | 47 | RSPAII | 87 | RAHSP |
| 8 | SAHPR | 48 | PSRAH | 88 | PAHSP |
| 9 | SRIAP | 49 | HASRP | 89 | RPHSA |
| 10 | SPIAR | 50 | HASPR | 90 | PRHSA |
| 11 | SRHPA | 51 | HRSAP | 91 | ARPSH |
| 12 | SPHRA | 52 | HPSAR | 92 | APRSH |
| 13 | SARHP | 53 | HRSPA | 93 | RAPSH |
| 14 | SAPHR | 54 | HPSRA | 94 | PARSH |
| 15 | SRAHP | 55 | AHSRP | 95 | RPASH |
| 16 | SPAHR | 56 | AHSPR | 96 | PRASH |
| 17 | SRPRA | 57 | RHSAP | 97 | HARPS |
| 18 | SPRHA | 58 | -SAR | 98 | HAPRS |
| 19 | SARPH | 59 | RHSPA | 99 | HRAPS |
| 20 | SAPRH | 60 | PHSRA | 100 | HPARS |
| 21 | SRAPH | 61 | ARSHP | 101 | HRPAS |
| 22 | SPARH | 62 | APSHR | 102 | HPRAS |
| 23 | SRPAH | 63 | RASHP | 103 | AHRPS |
| 24 | SPRH | 64 | PASHR | 104 | AHPRS |
| 25 | HSARP | 65 | RPSHA | 105 | RHAPS |
| 26 | HSAPR | 66 | PRSHA | 106 | PHARS |
| 27 | HSRAP | 67 | ARSPH | 107 | RPHAS |
| 28 | HSPAR | 68 | APSRH | 108 | PHRAS |
| 29 | HSRPA | 69 | RASPH | 109 | ARHPS |
| 30 | HSRPA | 70 | PASRH | 110 | APHRS |
| 31 | ASHRP | 71 | RPSAH | 111 | RAHPS |
| 32 | ASHPR | 72 | PRSAH | 112 | PAHRS |
| 33 | RSIAP | 73 | HARSP | 113 | RPHAS |
| 34 | PSHAR | 74 | HAPSR | 114 | PRHAS |
| 35 | RSHPA | 75 | HRASP | 115 | ARPHS |
| 36 | PSHRA | 76 | HPASR | 116 | APRHS |
| 37 | ASRHP | 77 | HRPSA | 117 | RAPHS |
| 38 | ASPHR | 78 | HPRSA | 118 | PARHS |
| 39 | RSAPR | 79 | AHRSP | 119 | RPAHS |
| 40 | PSAHR | 80 | AHPSR | 120 | PRAHS |

DAS WAR ALLES !



Dieses Programm wird vor allem Assemblerprogrammierer interessieren, da es bei der Fehlersuche in Maschinenprogrammen behilflich sein kann.

Nach Erscheinen des "\$"-Prompts (Bereitschaftszeichen) muß eines der folgenden Kommandos eingegeben werden; das Ergebnis gibt der Rechner in hexadezimaler Form aus.

(X^1 und X^2 sind Bytes in hexadezimaler Darstellung,
n ist eine ganze Dezimalzahl mit $|n| < 10.$)

| Kommandosyntax | Bedeutung |
|------------------------|----------------------------------------|
| $X^1 \text{ NOT}$ | $\overline{X^1}$ (Negation) |
| $X^1 \text{ AND } X^2$ | $X^1 \wedge X^2$ (Und-Verknüpfung) |
| $X^1 \text{ OR } X^2$ | $X^1 \vee X^2$ (Oder-Verknüpfung) |
| $X^1 \text{ XOR } X^2$ | $X^1 \nabla X^2$ (Exklusiv-Oder) |
| $X^1 \text{ ADD } X^2$ | $X^1 + X^2$ (Addition) |
| $X^1 \text{ SUB } X^2$ | $X^1 - X^2$ (Subtraktion) |
| $X^1 \text{ ROT } n$ | Rotation von X^1 um n Bits * |
| $X^1 \text{ SHF } n$ | Arithmetic Shift von X^1 um n Bits † |
| $X^1 \text{ NAD } X^2$ | $\overline{(X^1 \wedge X^2)}$ |
| $X^1 \text{ NOR } X^2$ | $\overline{(X^1 \vee X^2)}$ |
| $X^1 \text{ NEX } X^2$ | $\overline{(X^1 \nabla X^2)}$ |
| EXIT | Programmende |

* Ist n positiv, so wird X^1 um n Bits nach links rotiert.

(Bit i := Bit (i-1) ; Bit 0 := Bit 7)

Ist n negativ, so wird X^1 um -n Bits nach rechts rotiert.

(Bit i := Bit (i+1) ; Bit 7 := Bit 0)

+ Ist n positiv, so wird X^1 um n Bits nach links geschoben.

(Bit i := Bit (i-1) ; Bit 0 := 0)

Ist n negativ, so wird X^1 um -n Bits nach rechts geschoben.

(Bit i := Bit (i+1) ; Bit 7 := Bit 7)

```

1;"A"
3;"Bitoperationen an H
ez-Bytes
5:CLEAR
10:DATA " NOT"," AND","  

    OR "," XOR"," ADD",
    " SUB"," ROT"," SHF",
    " NAD"," NOR"," HEX
    "
20:DIM B$(1)*2,C$(0)*9,
    B(1)
30:INPUT "$ "+C$(0)
40:IF C$(0)="EXIT" GOTO
    "EXIT"
45:RESTORE
47:I=0
50:IF I=11 GOTO 90
60:READ C$
70:IF MID$(C$(0),3,4)=
    C$ GOTO 95
80:I=I+1
85:GOTO 50
90:PRINT "SYNTAX FEHLER
    ":GOTO 30
95:B$(0)=LEFT$(C$(0),2
    ):B$(1)=RIGHT$(C$(0
    ),2)
100:FOR N=0 TO 1
105:B(N)=0
110:FOR P=1 TO 2
120:Z$=MID$(B$(N),P,1)
130:IF Z$<>"A" THEN LET B
    (N)=B(N)+(ASC Z$-ASC
    "0")*16^(2-P)
140:IF Z$>"9" THEN LET B
    (N)=B(N)+(ASC Z$-ASC
    "7")*16^(2-P)
150:NEXT P
160:NEXT N
170:gosub C$
180:IF E<0 THEN LET E=25
    6+E
190:B$(0)=""
200:Y= INT (E/16):E=E-16
    *Y
210:B(0)=Y:B(1)=E
220:FOR P=0 TO 1
230:IF B(P)<10 THEN LET
    B$(0)=B$(0)+CHR$(B(
    P)+ASC "0")
240:IF B(P)>9 THEN LET B
    $(0)=B$(0)+CHR$(B(P
    )+ASC "7")
245:NEXT P
250:PRINT B$(0)
260:GOTO 30
270:"EXIT"
275:END
280:" AND"
290:E=B(0) AND B(1)
300:RETURN
310:" OR "
320:E=B(0) OR B(1)
330:RETURN
340:" XOR"
350:E=(B(0) OR B(1))-(B(
    0) AND B(1))
360:RETURN
370:" ADD"
380:E=B(0)+B(1)
390:IF E>255 THEN LET E=
    E-256
400:RETURN
410:" SUB"
420:E=B(0)-B(1)
430:RETURN
440:" ROT"
444:E=B(0)
447:R=VAL B$(1)
450:IF R<0 GOTO 510
460:IF R=0 THEN RETURN
470:FOR N=1 TO R
480:E=E*2
490:IF E>255 THEN LET E=
    E-256:LET E=E OR 1
495:NEXT N
500:RETURN
510:FOR N=1 TO ABS R
520:E=E/2
530:IF E<> INT E THEN
    LET E= INT E:LET E=E
    OR 128
535:NEXT N
540:RETURN
550:" NOT"
560:E=255-B(0)
570:RETURN
580:" SHF"
590:R=VAL B$(1)
600:E= INT (B(0)*2^R+.1)
610:E=E OR ((R(0)*B(0))
    =128)*(256-2^ INT (
    LOG (E+.1)/LOG 2)+E
    <1>))
620:E=E AND 255
630:RETURN
640:" NAD"
650:gosub " AND"
660:GOTO " NOT"
670:" NOR"
680:gosub " OR"
690:GOTO " NOT"
700:" HEX"
710:gosub " XOR"
720:GOTO " NOT"
730:" NOT"
740:E=255-E
750:RETURN

```

Wer keinen Drucker hat, sollte dieses Kapitel überblättern und sich dem nächsten Programm annehmen.

Das folgende Programm soll nämlich den Verlauf einer beliebigen Funktion innerhalb eines beliebigen x-Intervalls ausdrucken.

Das dargestellte y-Intervall reicht von -ymax bis +ymax, das darzustellende x-Intervall von x0 bis xl.

Delta-x ist die Schrittweite in x-Richtung pro Druckzeile.

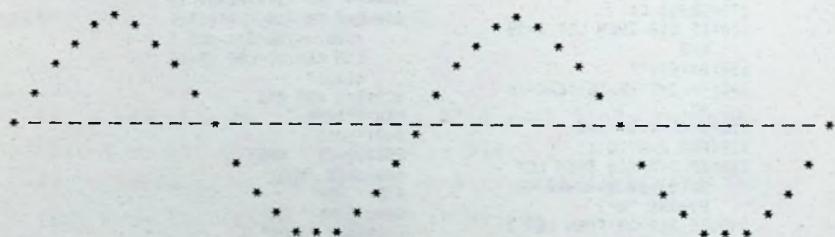
Alle vier Parameter (ymax, x0, xl, Delta-x) sind hierbei frei wählbar.

Die Funktion selbst steht in Zeile 80; im Listing ist es die Funktion $y = \sin x$.

Gibt man außerdem als Parameter ein:
(im RAD-Modus)

| |
|-----------------|
| ymax := 1 |
| x0 := 0 |
| xl := 4*π |
| Delta-x := π/10 |

so müßte der Drucker folgendes ausgeben:

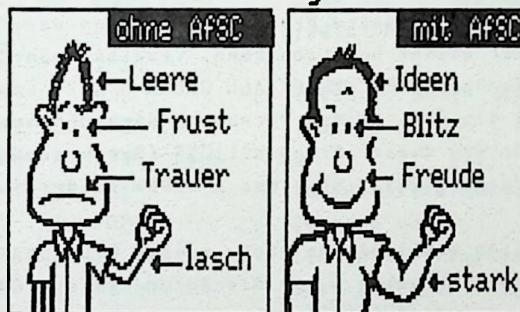


```

1;"A"
3;"Ausdruck des Verlauf
   fs einer Funktion
5:CLEAR
7:DIM F$(0)*25,M$(0)*2
4
10:INPUT "Ymax: ";M
20:INPUT "X0: ";A:~ 24
30:INPUT "X1: ";B:~ 40:INPUT "Delta-X: ";D
50:M$(0)=" "
60:X=A
80:Y=SIN X
85:IF ABS Y<=M THEN LET
   L=13+ INT (Y/M*11):
   LET F$(0)=LEFT$ (M$(0),L-1)+"+"+RIGHT$ (
   M$(0),24-L):GOTO 100
90:F$(0)=M$(0)
100:LPRINT F$(0)
110:IF X>B THEN END
120:X=X+D:GOTO 80

```

AfSC immer richtig informiert



Das folgende Maschinenprogramm wurde in Versionen für andere SHARP Pocket Computer bereits in /2/ und /15/ veröffentlicht.

Nach dem Start des BASIC-Programms werden zunächst das Maschinenprogramm und die Bitmuster der einzelnen Ziffern für deren Darstellung in der Anzeige in den Speicher geladen.

Nach Ertönen eines akustischen Signals wird nun die Zeit als Zahl des Formats hhmmss eingegeben (15:50:48 beispielsweise als 155048).

Anschließend zerlegt der Rechner diese Zahl in ihre Ziffern, legt diese in einem Zeitbuffer ab und meldet sich daraufhin mit der Aufforderung "Bitte auf ENTER druecken". Sofort nach Drücken dieser Taste beginnt die Uhr zu laufen.

Das Programm kann nur durch Druck auf 'ALL RESET' abgebrochen werden. Man sollte während dieses Drucks gleichzeitig eine weitere Taste auf der Gerätевorderseite drücken, damit man das Programm anschließend sofort wieder benutzen kann. Vergisst man das, so meldet sich der Rechner mit "MEMORY ALL CLEAR O.K.?" und gibt einem noch einmal die Chance, das Versäumte nachzuholen. Erst durch die Beantwortung dieser Frage mit "Y" (Druck genügt) wird das Programm gelöscht, bleibt aber durch RENEW wiedereinsetzbar.

Bei längerem Betrieb empfiehlt es sich, zwecks Schonung der eingebauten Batterie den Rechner z.B. an die Recorder-Interface/Drucker-Einheit CE-129P zu schließen.

| | | |
|-------------|------|----------------|
| Zeitbuffer: | F45A | Stundenzehner |
| | F45B | Stundeneiner |
| | F45C | Minutenzehner |
| | F45D | Minuteneiner |
| | F45E | Sekundenzehner |
| | F45F | Sekundeneiner |

Assemblerlisting

| | | | |
|------|----------|------------|-----------------------------------|
| F460 | 12 5F | LIP 5F | ;LCD einschalten |
| F462 | 61 01 | ORIM 01 | |
| F464 | DF | OUTC | |
| F465 | 78 F4 C0 | CALL F4C0 | ;Zeit anzeigen |
| F468 | 78 F5 10 | CALL F510 | ;1 sec warten |
| F46B | 12 05 | LIP 05 | ;Initialisierung des |
| F46D | 02 F4 | LIA F4 | ;X-Registers als Adreßpointer |
| F46F | DB | EXAM | ;für den Zeitbuffer |
| F470 | 51 | DECP | ;(#&F460 nach X) |
| F471 | 02 60 | LIA 60 | |
| F473 | DB | EXAM | |
| F474 | 78 F4 A0 | CALL F4A0 | ;Sekunden weiterzählen |
| F477 | 78 F4 A0 | CALL F4A0 | ;Minuten weiterzählen |
| F47A | 25 | DXL | ;Pointer zeigt auf Stundeneiner |
| F47B | 42 | INCA | ;Stundeneiner erhöhen |
| F47C | 52 | STD | |
| F47D | 67 04 | CPIA 04 | ;Stundeneiner = 4 ? |
| F47F | 28 04 | JRNZP F484 | |
| F481 | 78 F4 90 | CALL F490 | ;Ja: Ist es Mitternacht ? |
| F484 | 78 F4 83 | CALL F483 | ;Nein: Std.-Einer evtl. erhöhen |
| F490 | 25 | DYL | ;Pointer zeigt auf Stundenzehner |
| F491 | 67 02 | CPIA 02 | ;Ist es 24 Uhr ? |
| F493 | 2A 06 | JRNCP F49A | |
| F495 | 5B | POP | ;Nein: Stack Pointer zurück- |
| F496 | 5B | POP | ;setzen und |
| F497 | 79 F4 65 | JP F465 | ;ins Hauptprogramm zurückspringen |
| F49A | 02 00 | LIA 00 | ;Ja: Std-Zehner auf Null setzen |
| F49C | 52 | STD | |
| F49D | 04 | IX | |
| F49E | 52 | STD | ;Std-Einer auf Null setzen |
| F49F | 37 | RTN | ;Rückkehr ins Hauptprogramm |
| F4A0 | 25 | DXL | ;Adreßpointer zeigt auf Einer |
| F4A1 | 42 | INCA | ;Einer erhöhen |
| F4A2 | 52 | STD | |

| | | | |
|------|----------|------------|-----------------------------------|
| F4A3 | 67 0A | CPIA 0A | ;Einer größer als 9 ? |
| F4A5 | 2A 06 | JRNCP F4AC | |
| F4A7 | 5B | POP | ;Nein: Stack Pointer zurücksetzen |
| F4A8 | 5B | POP | ;und ins Hauptprogramm |
| F4A9 | 79 F4 65 | JP F465 | ;zurückspringen |
| F4AC | 02 00 | LIA 00 | ;Ja: Einer auf Null setzen |
| F4AE | 52 | STD | |
| F4AF | 25 | DXL | ;Pointer zeigt auf Zehner |
| F4B0 | 42 | INCA | Zehner erhöhen |
| F4B1 | 52 | STD | |
| F4B2 | 67 06 | CPIA 06 | ;Zehner größer als 5 ? |
| F4B4 | 2A 06 | JRNCP F4BB | |
| F4B6 | 5B | POP | ;Nein: Stack Pointer zurück- |
| F4B7 | 5B | POP | ;setzen und |
| F4B8 | 79 F4 65 | JP F465 | ;ins Hauptprogramm zurückspringen |
| F4BB | 02 00 | LIA 00 | ;Ja: Zehner auf Null setzen |
| F4BD | 52 | STD | |
| F4BE | 37 | RTN | ;Rückkehr ins Hauptprogramm |
| | | | |
| F4C0 | 00 04 | LII 04 | ;#4 nach I (für MVWD-Befehl) |
| F4C2 | 12 05 | LIP 05 | ;Initialisierung des |
| F4C4 | 02 F4 | LIA F4 | ;X-Registers als Adreßpointer |
| F4C6 | DB | EXAM | ;für den Zeitbuffer |
| F4C7 | 51 | DECPL | ;(#&F459 nach X) |
| F4C8 | 02 59 | LIA 59 | |
| F4CA | DB | EXAM | |
| F4CB | 12 0A | LIP 0A | ;P-Reg zeigt auf (10) im CPU-RAM |
| F4CD | 02 03 | LIA 03 | ;#3 nach A |
| F4CF | 34 | PUSH | ;A in den Stack retten |
| F4D0 | 02 03 | LIA 03 | ;#3 nach A |
| F4D2 | 34 | PUSH | ;A in den Stack retten |
| F4D3 | 67 03 | CPIA 03 | ;A = #3 ? |
| F4D5 | 3A 05 | JRCP F4DB | |
| F4D7 | 02 0A | LIA 0A | ;Ja: Code für Doppelpunkt nach A |
| F4D9 | 2C 02 | JRP F4DC | ;Nächsten Befehl überspringen |
| F4DB | 24 | IXL | ;Nein: Nächste Ziffer nach A |
| F4DC | 10 F4 E6 | LIDP F4E6 | ;DP zeigt auf (&F4E6) |
| F4DF | D1 | RC | ;Carry-Flag löschen |

| | | | |
|------|----------|------------|--------------------------------------------------------------------------|
| F4E0 | 5A | SL | ;Inhalt des Akkus verachten |
| F4E1 | 5A | SL | |
| F4E2 | 5A | SL | |
| F4E3 | 52 | STD | |
| F4E4 | 10 F4 xx | LIDP F4 xx | ;Die fünf Bitmuster der aktuellen |
| F4E7 | 18 | MVWD | Ziffer in den CPU-RAM bringen |
| F4E8 | 5B | POP | ;A aus dem Stack zurückholen |
| F4E9 | 43 | DECA | ;A=A-1 |
| F4EA | 7C F4 D2 | JPNZ F4D2 | ;Falls A größer 0 nach &F4D2 springen |
| F4ED | 5B | POP | ;Sonst A erneut aus dem Stack holen |
| F4EE | 43 | DECA | ;A=A-1 |
| F4EF | 7C F4 CF | JPNZ F4CF | ;Falls A größer 0 nach &FACF springen |
| F4F2 | 00 27 | LII 27 | ;Die ersten 40 Bytes ab der |
| F4F4 | 10 30 05 | LIDP 3005 | ;CPU-RAM-Adresse &0F werden in |
| F4F7 | 12 OF | LIP OF | den Bereich ab der Adresse &3005 |
| F4F9 | 19 | EXWD | ;kopiert (Display-Bereich) |
| F4FA | 37 | RTN | ;Rückkehr ins Hauptprogramm |
| F510 | 02 FF | LIA FF | ;"FOR A=255 TO 1 STEP -1" |
| F512 | 4E BF | WAIT BF | ;191 Cycles warten |
| F514 | 4E FF | WAIT FF | ;255 Cycles warten |
| F516 | 4E FF | WAIT FF | ;255 Cycles warten |
| F518 | 4E FF | WAIT FF | ;255 Cycles warten |
| F51A | 43 | DECA | |
| F51B | 29 0A | JRNZM F512 | ;"NEXT A" |
| F51D | 02 00 | LIA 00 | ;Das letzte LCD-Element löschen |
| F51F | 10 30 40 | LIDP 3040 | |
| F522 | 00 04 | LII 04 | |
| F524 | 1F | FILD | |
| F525 | 02 D0 | LIA D0 | ;Akku von 208 auf 0 runterzählen |
| F527 | 43 | DECA | |
| F528 | 29 02 | JRNZM F527 | |
| F52A | 4E 10 | WAIT 10 | ;Nochmals 16 Cycles warten (Hier kann die Uhr nachgeeicht werden.) |
| F52C | 37 | RTN | ;Rückkehr ins Hauptprogramm |

Bitmuster der darzustellenden Zeichen

| Adr | Bitmuster (hex) | Zeichen |
|------|-----------------|---------|
| F400 | 3E 41 41 41 3E | "0" |
| F408 | 00 04 02 7F 00 | "1" |
| F410 | 62 51 49 49 46 | "2" |
| F418 | 22 41 49 49 36 | "3" |
| F420 | 18 14 12 7F 10 | "4" |
| F428 | 2F 45 45 45 39 | "5" |
| F430 | 3E 49 49 49 32 | "6" |
| F438 | 03 61 19 05 03 | "7" |
| F440 | 36 49 49 49 36 | "8" |
| F448 | 26 49 49 49 3E | "9" |
| F450 | 00 36 36 36 00 | |

```

1:"A"
3:"Digitaluhr
5:CLEAR
10:POKE &F400,&3E,&41,&
    41,&41,&3E
20:POKE &F408,&00,&04,&
    02,&7F,&00
30:POKE &F410,&62,&51,&
    49,&49,&46
40:POKE &F418,&22,&41,&
    49,&49,&36
50:POKE &F420,&10,&14,&
    12,&7F,&10
60:POKE &F428,&2F,&45,&
    45,&45,&39
70:POKE &F430,&3E,&49,&
    49,&49,&32
80:POKE &F438,&03,&61,&
    19,&05,&03
90:POKE &F440,&36,&49,&
    49,&49,&36
100:POKE &F448,&26,&49,&
    49,&49,&3E
110:POKE &F450,&00,&36,&
    36,&36,&00
120:POKE &F460,&12,&5F,&
    61,&01,&DF,&78,&F4,&
    C0,&78,&F5,&10,&12,&
    05,&02,&F4,&DD
130:POKE &F470,&51,&02,&
    60,&DB,&78,&F4,&A0,&
    78,&F4,&A0,&25,&42,&
    52,&67,&04,&28
140:POKE &F480,&04,&78,&
    F4,&90,&78,&F4,&A3
150:POKE &F490,&25,&67,&
    02,&2A,&06,&5B,&5B,&
    79,&F4,&65,&02,&00,&
    52,&84,&52,&37
160:POKE &F4A0,&25,&42,&
    52,&67,&08,&2A,&06,&
    5B,&5B,&79,&F4,&65,&
    02,&00,&52,&25
170:POKE &F4B0,&42,&52,&
    67,&06,&2A,&06,&5B,&
    5B,&79,&F4,&65,&02,&
    00,&52,&37
180:POKE &F4C0,&00,&04,&
    12,&05,&02,&F4,&DB,&
    51,&02,&59,&DB,&12,&
    0A,&02,&03,&34
190:POKE &F4D0,&02,&03,&
    34,&67,&03,&3A,&05,&
    02,&8A,&2C,&02,&24,&
    10,&F4,&E6,&D1
200:POKE &F4E0,&5A,&5A,&
    5A,&52,&10,&F4,&00,&
    18,&5B,&43,&7C,&F4,&
    D2,&5B,&43,&7C
210:POKE &F4F0,&F4,&CF,&
    00,&27,&10,&30,&05,&
    12,&0F,&19,&37
220:POKE &F510,&02,&FF,&
    4E,&BF,&4E,&FF,&4E,&
    FF,&4E,&FF,&43,&29,&
    0A,&02,&00,&00
222:POKE &F520,&04,&10,&
    30,&40,&1F,&02,&D0,&
    43,&29,&02,&4E,&10,&
    37
230:BEEP 2
240:INPUT A
250:B=&F45F
260:FOR C=1 TO 6
270:POKE B,A- INT (A/10)
    *10
280:A= INT (A/10)
285:B=B-1
290:NEXT C
300:PRINT "Bitte auf ENT
    ER druecken":WAIT 0:
    PRINT **:CALL &F460

```

5.15 Strategiespiel AWELE

Awele ist ein Strategiespiel für zwei Personen, welches ursprünglich von der Elfenbeinküste stammt.

Das Spielfeld wird in zwei Hälften à fünf Fächer aufgeteilt, in die zu Beginn des Spiels je drei Muschelschalen oder Körner gelegt werden.

In dem folgenden Programm übernimmt der Rechner, der hier als Gegenspieler auftritt, die linke Spielhälfte und der Benutzer die rechte.

Bei jedem Zug nimmt der Spieler den Inhalt eines nicht-leeren Fachs seines Feldes und legt die Körner einzeln nacheinander in die nach rechts angrenzenden Fächer. Hierbei zählt das erste Fach als rechter Nachbar des zehnten Fachs. Der Benutzer hat hierbei nur die gewünschte Fach-Nr. (6-10) anzugeben; den Rest übernimmt der Rechner automatisch.

Beispiel: vorher: 2.3.0.1.3./1.2.1.3.1.
 Fach-Nr.: 9
 nachher: 3.4.0.1.3./1.2.1.0.2.

Jetzt wird es kompliziert:

Kommt das letzte Korn in ein gegnerisches Fach, in dem vorher genau ein Korn gelegen hat, so kann der Spieler den Inhalt dieses Fachs als Gewinn einstreichen. Dasselbe gilt für alle davorliegenden, zusammenhängenden Fächer, die, einschließlich des eben hineingelegten, höchstens zwei Körner enthalten. Jedes so gewonnene Korn zählt einen Punkt.

Beispiel: vorher: 1.2.1.1.0./1.2.0.5.4.
 Fach-Nr.: 10
 nach dem Zug: 2.3.2.2.0./1.2.0.5.0.
 Inhalt der Fächer 3, 4 gewonnen. (+ 4 Punkte)
 nachher: 2.3.0.0.0./1.2.0.5.0.

Auch diese Prozedur wird vom Rechner durchgeführt.

Der Sinn des Spiels besteht somit darin, dem Gegner möglichst viele Körner abzuknöpfen. Es ist jedoch verboten, dem Gegner alles wegzunehmen. sonst wird das Spiel sofort beendet und das Konto des Missetäters um den Betrag des vermeintlichen Gewinns vermindert.

Beispiel: vorher: 1.1.0.0.0./0.0.0.3.1.
Fach-Nr.: 9
nach dem Zug: 2.2.0.0.0./0.0.0.0.2.
Fächer 1 und 2 scheinen gewonnen zu sein.
nachher: 0.0.0.0.0./0.0.0.0.2.
Punktverlust: 4 Punkte

Das Spiel wird ebenfalls beendet, sobald einer der beiden Spieler mehr als 14 Punkte erreicht hat oder beide zusammen mehr als 21 Punkte haben.

Der Benutzer kann während des Spiels ständig den aktuellen Punktestand einsehen, indem er auf die Frage "Fach-Nr.:" mit "Ø" (Null) antwortet.

```
13:INPUT "Fach-Nr.:";I
14:IF I=0 THEN 900
15:IF B(I)=0 OR I<6 OR
   I>10 THEN 11
20:"TRI*B(0)=B(I):C=0
22:IF B(I)+I<10 LET A=
   I+1:B=B(I)+I:GOTO "A
   V1"
24:IF I=10 AND B(I)<=10
   LET A=1:B=B(I):GOTO
   "AV1"
26:IF I=10 AND B(I)>10
   LET A=1:B=10:C=1:
   GOSUB "AV1":A=1:B=B(
   0)-10:GOTO "AV2"
28:IF B(I)+I>10 LET A=I
   +1:B=10:C=1:GOSUB "A
   V1"
29:IF B(0)-10+I<=10 LET
   A=1:B=B(0)-10+I:GOTO
   "AV2"
30:IF B(0)-10+I<=20 LET
   A=1:B=10:C=1:GOSUB "
   AV2":A=1:B=B(0)-20+I
   :GOTO "AV2"
40:"AV1*B(I)=0
41:"AV2"
42:FOR N=A TO B
44:B(N)=B(N)+1
46:NEXT N
47:N=B
48:IF C=1 LET C=0:
   RETURN
50:PRINT B(1);B(2);B(3)
   ;B(4);B(5);"/";B(6);
   B(7);B(8);B(9);B(10)
60:"TPR"
62:IF B(N)>2 THEN "TOU
   .
63:IF B(N)=2 AND N<6
   AND I<6 THEN "TOU"
64:IF B(N)=2 AND N>5
   AND I>5 THEN "TOU"
70:"PRI"
71:Z=0
72:FOR N=B TO S STEP -1
74:IF B(N)>2 LET N=S:
   GOTO 84
76:IF S=6 LET Q=Q+B(N):
   Z=Z+B(N)
```

```

78:IF S=1 LET W=W+B(N):
    Z=Z+B(N)
02:B(N)=0
04:NEXT N
05:IF B(6)=0 AND B(7)=0
    AND B(8)=0 AND B(9)=
    0 AND B(10)=0 LET Q=
    Q-Z:BEEP 1:GOTO 900
06:IF B(1)=0 AND B(2)=0
    AND B(3)=0 AND B(4)=
    0 AND B(5)=0 LET W=W
    -Z:BEEP 1:GOTO 900
90:PRINT B(1);B(2);B(3);
    ;B(4);B(5);";";B(6);
    B(7);B(8);B(9);B(10)
95:IF Q>14 OR W>14 OR Q
    +W>21 BEEP 2:GOTO 90
    0
100:TOU"
102:E=E+1
103:S=(E/2- INT (E/2))
104:IF S<.5 LET S=1:
    GOTO "JOU"
106:S=6
200:IF B(1)=0 AND B(2)=0
    AND B(3)=0 AND B(4)=
    0 AND B(5)=0 BEEP 1:
    GOTO "TOU"
201:PAUSE "Ich bin dran.
    "
202:D=0:G=0
204:FOR D=6 TO 10
206:FOR P=5 TO 1 STEP -1
208:IF B(0)=1 AND B(P)+P
    =0 LET G=P
210:IF B(P)=1 AND B(0)+0
    -10=P LET D=P
212:NEXT P:NEXT D
213:IF G>0 THEN "SI0"
214:IF D>0 LET I=D:GOTO
    "TRI"
302:K:=0:=M=0
304:FOR P=1 TO 5
306:IF B(P)+P>9 LET M=P
308:IF B(P)=1 AND K<0
    LET L=P
310:IF B(P)=1 AND K=0
    LET K=P
312:NEXT P
313:IF M<0 LET I=M:GOTO
    "TRI"
314:IF K>0 AND B(K+1)=0
    AND K>5 THEN "SI2"
316:IF K>0 LET I=K:GOTO
    "TRI"
400:IF E=1 LET I=RND 3:
    GOTO "TRI"
500:"DPN*S=0
502:FOR K=1 TO 5
504:IF B(K)>0 THEN "SI1
    "
505:IF B(K)=0 LET S=S+1
506:NEXT K
508:IF V=0 THEN "TRI"
510:IF V>0 AND U=0 AND
    S<4 THEN "TRI"
511:IF V>0 AND U=0 AND
    S>4 LET I=Y:GOTO "TR
    I"
512:IF V>0 AND U>0
    THEN "SI3"
600:"SI1"
602:FOR I=1 TO 10
604:C(I)=B(I)
606:NEXT I
610:C(K)=0
612:FOR I=K+1 TO B(K)+K
614:C(I)=C(I)+1
616:NEXT I
622:FOR I=1 TO 5
624:FOR J=6 TO 10
626:IF C(I)=1 AND C(J)+J
    -10=I AND V=0 LET V=
    I:Y=K
627:IF C(I)=1 AND C(J)+J
    -10=I AND V>0 AND U
    =0 AND K>Y LET U=I:X
    =K
628:NEXT J
630:NEXT I
632:IF V=0 LET I=K:K=5:
    GOTO 506
633:IF V>0 AND U=0 AND
    K>Y LET I=K:K=5:GOTO
    506
634:GOTO 506
700:"SI2"
702:R=0
704:FOR J=10 TO 6 STEP -
    1
706:IF K+1=B(J)+J-10 LET
    J=6:R=1
708:NEXT J
710:IF R=0 LET I=K:GOTO
    "TRI"
712:IF L>0 AND L<>5 AND
    B(L+1)=0 LET K=L:
    GOTO 702
714:IF L>0 LET I=L:GOTO
    "TRI"
716:GOTO "DPN"
800:"SI3"
802:FOR I=1 TO 10
804:C(I)=B(I)
806:NEXT I
810:Z=V
812:FOR I=Z TO 1 STEP -1
814:C(I)=C(I)+1
816:IF C(I)>0 AND C(I)<3
    LET G=G+C(I)
818:NEXT I
820:IF H=0 LET V=U:H=G:G
    =0:GOTO 802
822:IF HK=G LET I=Y:GOTO
    "TRI"
824:I=X:GOTO "TRI"
850:"SI0"
852:FOR I=5 TO 10
854:C(I)=B(I)
856:IF I<G+B(G) LET C(I)
    =C(I)+1
858:NEXT I
860:FOR I=G+B(G) TO 5
    STEP -1
862:IF C(I)>2 LET I=5:
    GOTO 866
864:C(I)=0
866:NEXT I
868:IF C(6)=0 AND C(7)=0
    AND C(8)=0 AND C(9)=
    0 AND C(10)=0 THEN 2
    14
870:I=G:GOTO "TRI"
900:PAUSE "---ERGEBNIS--"
    -:PRINT "Du hast ";
    STR$ W;" Punkte.":
    PRINT "Ich habe ";
    STR$ Q;" Punkte."
901:IF I=0 PRINT B(1);B(
    2);B(3);B(4);B(5);"/
    ";B(6);B(7);B(8);B(9
    );B(10):GOTO 13
902:END

```

Dieses Programm hat kein herkömmliches Glücksspiel wie Roulett, Black Jack o.ä. als Vorbild, sondern stellt eine Art Zahlentippspiel dar.

Zunächst wird die Anzahl der Spieler in den Rechner gegeben und jedem Spieler ein Betrag von \$ 50 000.- zur Verfügung gestellt.

Jeder Spieler wird nun einzeln vom Rechner aufgerufen (zu diesem Zweck bekommt jeder Teilnehmer eine Nummer) und wird nach seinem Einsatz befragt. Der Spieler darf anschließend eine Ziffernkombination eingeben, die nur aus Einsen und Zweien besteht und bis zu sechs Ziffern lang sein darf. Die Länge der Kombination muß ebenfalls eingegeben werden.

Nun wird ausgelost: Der Rechner springt intern immer zwischen den Zahlen 1 und 2 hin und her, bis er vom Spieler mit einem Druck auf "*" gestoppt wird. Diejenige Ziffer, bei der der Rechner zuletzt stand, ist damit ausgelost worden.

Auf diese Weise wird eine Ziffernkombination ermittelt, welche mit der vorhergesagten verglichen wird.

Sind sie verschieden, so verliert der Spieler seinen Einsatz, sind sie dagegen gleich, so gewinnt er das $(2^n - 1)$ fache seines Einsatzes ($n = \text{Anzahl der Ziffern in der Kombination}$). 2^{-n} ist nämlich gerade die Wahrscheinlichkeit, die richtige Kombination erwischt zu haben.

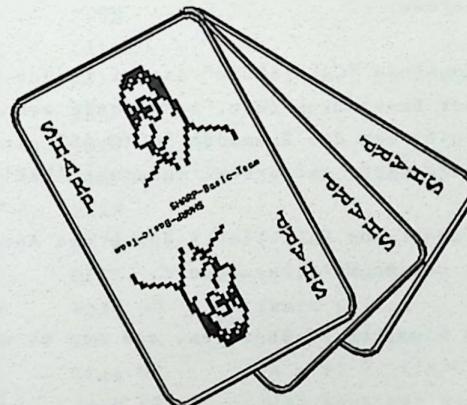
Gerät ein Spieler in die roten Zahlen, so muß er ausscheiden und wird vom Rechner auch nicht mehr aufgerufen.

Das Spiel wird mit 'DEF' "F" beendet, wobei die Kontostände aller Spieler noch einmal aufgeführt werden.

```

1: "A"
3: "Gluecksspielprogramm"
4:
7: CLEAR
10: INPUT "SPIELERZAHL":
    ";N"
15: DIM S(N): FOR I=1 TO
    N:S(I)=50000:NEXT I
20: PAUSE "ENDE MIT DEF
    F"
30: FOR I=1 TO N
35: IF S(I)<=0 THEN NEXT
    I: GOTO 30
40: PRINT "ES SPIELT ";I;
    STR$ I;"."
45: PRINT "KONTO: $ ";S(
    I)
50: INPUT "EINSATZ: $ ";
    E
60: INPUT "ZIFFERNZAHL
    : ";Z
70: IF Z<1 OR Z>6 THEN
    PAUSE "WITZBOLD !!":
    GOTO 60
75: INPUT "ZAHLENTIP: ";
    T
78: IF LEN STR$ T>Z
    THEN PRINT STR$ Z;""
    ZIFFER":CHR$ ((Z>1)*
    78)+" ":"GOTO 60
80: PAUSE "OKAY... ":
    WAIT 30:PRINT "AUSLO
    SUNG JETZT"
85: FOR K=1 TO Z
90: IF INKEY$ = "*" GOTO
    98
92: FOR J=1 TO 1000:IF
    INKEY$ = "*" GOTO 97
95: NEXT J:GOTO 98
97: J=(J AND 1)+1
100: PRINT STR$ J:IF STR$
    J;>MIDS (STR$ T,K,1)
    GOTO 310
110: NEXT K
120: WAIT :PRINT "GLUECKW
    UNSCH !":PRINT "GEWI
    NH: $ ";E*2^Z-E
125: S(I)=S(I)+E*2^Z-E
130: PRINT "KONTO: $ ";S(
    I)
140: NEXT I
150: GOTO 30
310: WAIT :PRINT "DAS TUT
    MIR LEID."
320: PRINT "VERLOREN !"
325: S(I)=S(I)-E
330: PRINT "KONTO: $ ";S(
    I)
340: IF S(I)<=0 THEN
    PRINT "PLEITE! TSCHU
    ESS"
350: NEXT I
360: GOTO 30
400: "F":PAUSE "ENDSTAND:
    "
410: FOR I=1 TO N
420: PRINT " ";STR$ I;" ";
    $ ";S(I)
430: NEXT I
450: PAUSE "AUF WIEDERSEH
    EN"

```



Die Handhabung dieses Programms ist sehr einfach: Nach dem Starten mit "RUN" wartet der Rechner einige Sekunden (die Wartezeit hängt von der RND-Funktion ab und ist daher unregelmäßig und unvorhersagbar) und fängt dann an, einen 4 kHz-Ton von sich zu geben.

Der Benutzer muß nun auf dieses Signal hin blitzschnell auf die rote 'C/CE'-Taste drücken. Die benötigte Zeit wird auf Millisekunden genau angezeigt und das Programm wieder von neuem gestartet.

Das Messen der Zeit übernimmt ein Maschinenprogramm, welches den eingebauten Tongenerator einschaltet und solange zählt, bis die 'C/CE'-Taste gedrückt wird. Diejenige Zahl, bis zu der der Rechner beim Zählen gekommen ist, wird berechnet und daraus die verflossene Zeit bestimmt.

Zur Umrechnung dient die in Zeile 100 stehende Zahl 5018.5. Sie gibt an, bis zu welcher Zahl der Rechner innerhalb einer Sekunde zählen kann und muß gegebenenfalls für jeden Rechner leicht korrigiert werden.

Dazu gibt man das Kommando "CALL &FOOO" direkt in den Rechner ein und drückt nach einer bestimmten Zeit, z.B. exakt zehn Minuten auf 'C/CE'. Danach gibt man das Kommando "GOTO 65" ein, wartet, bis die Zeit angezeigt wird, und drückt dann auf 'BRK'.

Man teilt nun den Inhalt der Variablen T durch die Anzahl der Sekunden und erhält so den Umrechnungswert.

Um das Programm als Stoppuhr zu benutzen, muß man es wie folgt ändern.

Zeilen 50 und 60 löschen

Zeile 110 erweitern mit ":CALL &FO00"

sowie "PAUSE" durch "PRINT" ersetzen

Zeile 115 löschen

Zeile 120:GOTO 65

Nach der Anzeige der Zeit wird die Uhr mit 'ENTER' erneut gestartet und mit Druck auf 'C/CE' gestoppt, worauf wieder die Zeitanzeige folgt.

Benutzt man als Aufrufadresse &FO05 statt &FO00, so bleibt man während der Zeitmessung von dem nervenden Sinuston verschont.

Assemblerlisting

| | | | |
|------|-------|-----------|-------------------------------------|
| F000 | 12 5F | LIP 5F | |
| F002 | 61 30 | ORIM 30 | ;Tongenerator einschalten (4 kHz) |
| F004 | DF | OUTC | |
| F005 | 02 FF | LIA FF | :#255 in den Stack (LOOP 1) |
| F007 | 34 | PUSH | |
| F008 | 02 FF | LIA FF | :#255 in den Stack (LOOP 2) |
| F00A | 34 | PUSH | |
| F00B | 02 FF | LIA FF | :#255 in den Stack (LOOP 3) |
| F00D | 34 | PUSH | |
| F00E | 12 5C | LIP 5C | |
| F010 | 02 01 | LIA 01 | :#1 nach &5C (CPU-RAM-Adresse) |
| F012 | DB | EXAM | |
| F013 | 50 | INCP | |
| F014 | 02 00 | LIA 00 | :#0 nach &5D |
| F016 | DB | EXAM | |
| F017 | 5D | OUTA | ;(&5C) nach Port IA |
| F018 | DD | OUTB | ;(&5D) nach Port IB |
| F019 | 4E 00 | WAIT 00 | ;kurz warten |
| F01B | 4C | INA | ; (Port IA) nach A |
| F01C | 67 02 | CPIA 02 | ;A = #2 ? ('C/CE'-Taste gedrückt ?) |
| F01E | 38 07 | JRZP F026 | ;Ja: Raus aus den Loops |
| F020 | 2F 13 | LOOP F00E | ;LOOP 3 |
| F022 | 2F 18 | LOOP F00B | ;LOOP 2 |
| F024 | 2F 1D | LOOP F008 | ;LOOP 1 |

| | | |
|------|-------|----------------------------------|
| F026 | 12 06 | LIP 06 |
| F028 | 02 4F | LIA 4F |
| F02A | DB | EXAM ;#&FO4F nach Y |
| F02B | 50 | INCP |
| F02C | 02 F0 | LIA F0 |
| F02E | DB | EXAM |
| F02F | 00 03 | LII 03 ;"FOR I=0 TO 3" |
| F031 | 5B | POP ;(R) nach A, R=R+1 |
| F032 | 06 | IY ;Y=Y+1, Y nach DP |
| F033 | 52 | STD ;A nach (DP) |
| F034 | 41 | DECI |
| F035 | 29 05 | JRNZM F031 ;"NEXT I" |
| F037 | 12 5F | LIP 5F |
| F039 | 60 CF | ANIM CF ;Tongenerator abschalten |
| F03B | DF | OUTC |
| F03C | 37 | RTN ;Return |

Versionen dieses Programms für andere PC-Typen befinden sich in /1/ und /2/.

```

1:"A"
5:"Reaktionstestprogra
  am mit akustischer A
  uslöesung
7:RANDOM
10:POKE &F000,&12,&5F,&
  61,&30,&DF,&02,&FF,&
  34,&02,&FF,&34,&02,&
  FF,&34,&12,&5C
20:POKE &F010,&02,&01,&
  DB,&50,&02,&00,&DB,&
  5D,&DD,&4E,&00,&4C,&
  67,&02,&38,&07
30:POKE &F020,&2F,&13,&
  2F,&18,&2F,&1D,&12,&
  06,&02,&4F,&DB,&50,&
  02,&F0,&DB,&00
40:POKE &F030,&03,&5B,&
  06,&52,&41,&29,&05,&
  12,&5F,&60,&CF,&DF,&
  37
45:POKE &F200,&12,&5F,&
  61,&01,&DF,&37
50:"START":FOR I=0 TO
  RND 1000:NEXT I
60:CALL &F000
65:T=0
70:FOR I=0 TO 2
  80:T=T+(255-PEEK (&F050
    +I))*256*I
  90:NEXT I
100:SEC=T/5018.5:MIN=SEC
  /60:SEC=SEC-60* INT
  MIN
110:PAUSE " Zeit: ";
  STR$ INT MIN;"";
  CHR$ (48*(SEC<10));
  INT ((SEC*1000+.5)/10
  00)
115:CALL &F200
120:GOTO "START"

```

Das letzte Programm in diesem Buch soll dem Gedächtnistraining dienen.

Das Prinzip des Programms besteht darin, daß der Benutzer ein vom Rechner vorgegebenes und bei jedem Durchgang um einen Buch-Buchstaben ergänztes Zufallswort aus dem Gedächtnis einzugeben hat.

Damit man das Wort wenigstens aussprechen kann, wird es abwechselnd aus Konsonanten und Vokalen aufgebaut.

```

1: "A"
5: "Gedächtnistest"
9: CLEAR :RANDOM
10: DIM B$(0)*24,C$(0)*2
    4
20: S$="AEIOU":Z=0
30: B$(0)=""
40: IF Z AND 1 THEN LET
    B=RND 5:LET B$(0)=B$
    (0)+MID$ (S$,B,1):
    GOTO 60
50: B=RND 26+64:I=1
53: IF CHR$ B=MID$ (S$,I
    ,1) GOTO 59
55: I=I+1:IF I<6 GOTO 53
57: B$(0)=B$(0)+CHR$ B
60: PAUSE "Das Wort laut
    et:"
70: WAIT 100:PRINT B$(0)
    :WAIT
80: PAUSE "Wie lautet da
    s Wort ?"
90: INPUT C$(0)
100: PAUSE "Das Wort laut
    et:"
110: PAUSE B$(0) GOTO
    200
130: PRINT "Richtig."
140: Z=Z+1
145: IF Z=24 THEN PRINT "
    Alles geschafft !":END
150: PAUSE "Jetzt ";STR$
    (Z+1); " Buchstaben."
160: GOTO 40
200: PAUSE "Das sind dane
    ben."
210: PRINT "Ende mit ";
    STR$ (Z+1); " Buchsta
    ben"
220: END

```

A.1 Literaturverzeichnis

- /1/ I. Laue: "Anwendungs-Handbuch zum SHARP Taschencomputer PC-1245, PC-1251, PC-1260, PC-1261", Fischel GmbH, Berlin, 1986.
- /2/ I. Laue/S. Kuhlmann: "Maschinenspracheprogrammsammlung für die SHARP Taschencomputer PC-1401, PC-1402, PC-1421", Fischel GmbH, Berlin, 1986.
- /3/ "SHARP PC-1250/1251 (1250A) Machine Language Reference Manual", EA 1251 T, SHARP Corporation, Osaka, 1983.
- /4/ H. J. Kowalsky: "Lineare Algebra", 9. Aufl., Walter de Gruyter, Berlin, 1979.
- /5/ H. Grauert/W. Fischer: "Differential- und Integralrechnung", Band II, 3. Aufl., Springer, Berlin/Heidelberg, 1978.
- /6/ L. Bergmann/C. Schaefer: "Lehrbuch der Experimentalphysik", Band 1: "Mechanik", 9. Auflage, Walter de Gruyter, Berlin, 1974.
- /7/ L. D. Landau/E. M. Lifschitz: "Lehrbuch der Theoretischen Physik", Band I: "Mechanik", 11. Aufl., Akademie-Verlag, Berlin, DDR, 1984.
- /8/ J. L. Bentley: "How to sort", Communications of the A.C.M. 27, 4 (April 1984) 287.

- /9/ E. Horowitz/S. Sahni: "Fundamentals of Data Structures in Pascal", 2. Aufl., Computer Science Press, Rockville, Maryland, 1987.
- /10/ "The Astronomical Almanac for the Year 1987", U.S. Government Printing Office/Her Majesty's Stationery Office, Washington D.C./London, 1986.
- /11/ A. Rükl: "Mond, Mars, Venus", Werner Dausien, Hanau, 1977.
- /12/ I. N. Bronstein/K. A. Semendjajew: "Taschenbuch der Mathematik", 22. Aufl., B. G. Teubner, Leipzig, 1985.
- /13/ G. Joos/E. Richter: "höhere Mathematik für den Praktiker", 12. Aufl., Harri Deutsch, Thun/Frankfurt am Main, 1979.
- /14/ E. Rich: "Artificial Intelligence", McGraw Hill, New York, 1983.
- /15/ I. Laue: "Uhrenprogramm für PC-1245/51", Alles für SHARP Computer 3, 5 (Mai 1985) 23.
- /16/ S. Kuhlmann: "Maschinensprachehandbuch zum SHARP Taschencomputer PC-1403", Fischel GmbH, Berlin, 1987.
- /17/ "SHARP PC-1403 Service Manual", Code: 00ZPC1403SM/E, SHARP Corporation, Osaka, Juni 1986.
- /18/ "SHARP CE-140F Pocket Disk Drive Product Information", 2Y5E, SHARP Corporation, Osaka, 1987.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|----|--------|--------|--------|--------|---------|---------|-------|-------|
| 20 | SPACE | ! | " | # | \$ | % | & | ' |
| 28 | (|) | * | + | , | - | . | / |
| 30 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 38 | 8 | 9 | : | ; | < | = | > | ? |
| 40 | @ | A | B | C | D | E | F | G |
| 48 | H | I | J | K | L | M | N | O |
| 50 | P | Q | R | S | T | U | V | W |
| 58 | X | Y | Z | [| \ |] | ^ | - |
| 60 | ' | a | b | c | d | e | f | g |
| 68 | h | i | j | k | l | m | n | o |
| 70 | p | q | r | s | t | u | v | w |
| 78 | x | y | z | { | | } | ~ | |
| 80 | MDF | REC | POL | ROT | DECI | HEX | TEN | RCP |
| 88 | SQU | CUR | HSN | HCS | HTN | AHS | AHC | AHT |
| 90 | FACT | LN | LOG | EXP | SQR | SIN | COS | TAN |
| 98 | INT | ABS | SGN | DEG | DMS | ASN | ACS | ATN |
| A0 | RND | AND | OR | NOT | ASC | VAL | LEN | PEEK |
| A8 | CHR\$ | STR\$ | MID\$ | LEFT\$ | RIGHT\$ | INKEY\$ | PI | MEM |
| B0 | RUN | NEW | CONT | PASS | LIST | LLIST | CSAVE | CLOAD |
| B8 | MERGE | ~ | ~ | OPEN | CLOSE | SAVE | LOAD | ~ |
| C0 | RANDOM | DEGREE | RADIAN | GRAD | BEEP | WAIT | GOTO | TRON |
| C8 | TROFF | CLEAR | USING | DIM | CALL | POKE | ~ | ~ |
| D0 | TO | STEP | THEN | ON | IF | FOR | LET | REM |
| D8 | END | NEXT | STOP | READ | DATA | PAUSE | PRINT | INPUT |
| E0 | GOSUB | AREAD | LPRINT | RETURN | RESTORE | CHAIN | ~ | ~ |
| E8 | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
| FO | ♣ | ■ | □ | π | ✓ | ♠ | ♥ | ♦ |
| F8 | ◆ | ■ | □ | π | ✓ | | | |

(QUELLE: /17/)

| | | KEY SIGNAL INPUT | | | | | | | |
|--------------------------|-------|------------------|-------|------------|-------|-------|-------|-----------|-------|
| <u>KEY STROBE OUTPUT</u> | | I A 1 | I A 2 | I A 3 | I A 4 | I A 5 | I A 6 | I A 7 | I A 8 |
| VOM | K01 | 7 | 8 | 9 | ÷ | x → M | | | |
| GATE ARRAY | K02 | 4 | 5 | 6 | X | R M | SHIFT | DEF | SML |
| CHIP | K03 | 1 | 2 | 3 | - | M + | Q | A | Z |
| LZ 92 K32 | K04 | Ø | +/- | * | + | = | W | S | X |
| | K05 | hyp | sin | cos | tan | | E | D | C |
| | K06 | HEX | DEG | ln | log | | R | F | V |
| | K07 | EXP | y^x | \sqrt{x} | x^2 | | T | G | B |
| VON DER | I A 1 | C-CE | ↑ | FSE | ↓ | Y | H | N | |
| CPU | I A 2 | |) | $1/x$ | ↑ | U | J | M | |
| SC61860 A | I A 3 | | | (| ◀ | I | K | SPC | |
| | I A 4 | | | | ► | O | L | EN TER | |
| | I A 5 | | | | | P | , | BASIC | |
| | I A 6 | | | | | | | CAL | |

Der 'ALL RESET'-Schalter auf der Geräterückseite wird benutzt, wenn ein Problem auftritt, welches weder mit 'C/CE', 'BRK' oder irgendeine andere Tastenbetätigung behoben werden kann.

Um den Rechner wieder benutzen zu können, drückt man irgendeine Taste auf der Gerätewerterseite und gleichzeitig den 'ALL RESET'-Schalter. Dadurch bleiben Programme und Daten erhalten.

Wenn man den 'ALL RESET'-Schalter betätigt, so drücke man ihn mindestens zwei oder drei Sekunden. Bei kürzeren Zeiten kann es passieren, daß der Hardware Reset nicht aktiviert wird.

Der Schaltet sollte mit einem spitzen Gegenstand wie beispielsweise einem Kugelschreiber gedrückt werden. Benutzen Sie keine brüchigen Gegenstände wie Bleistifte oder Nadeln. Benutzen Sie ebenso keine Gegenstände, die dicker als das Loch für die Taste sind.

Falls Sie keine Reaktion auf Tastendrücke nach der oben beschriebenen Operation erkennen können, so drücken Sie nur die 'ALL RESET'-Taste. Anschließend wird folgende Mitteilung in der Anzeige erscheinen:

MEMORY ALL CLEAR O.K.?

Drücken Sie nun 'ENTER' oder "Y" oder "=" . Wenn einer dieser Tasten ca. zwei Minuten nach Erscheinen der obenstehenden Mitteilung betätigt wird, so schaltet sich der Rechner automatisch ab.

Anschließend sind Programme und Daten gelöscht; sie können jedoch eventuell durch RENEW (siehe 2.7) wiederbeschafft werden.

A.5 Abspeichern von Maschinenprogrammen auf Cassette

Für das Aufnehmen und Einlesen von Maschinenprogrammen auf Band gibt es Erweiterungen der Befehle "CSAVE" und "CLOAD".

- (i) Das Abspeichern eines Maschinenprogramms, welches bei der Adresse anfangsadr anfängt und bei der Adresse endadr aufhört, geschieht mit folgendem Kommando:

```
CSAVE M"programmname";anfangsadr,endadr
```

Will man auf einen Programmnamen verzichten, so lautet das Kommando

```
CSAVE M anfangsadr,endadr
```

- (ii) Das Einlesen eines Maschinenprogramms, welches bei der Anfangsadresse anfangsadr beginnen soll, geschieht mit dem folgenden Kommando:

```
CLOAD M"programmname";anfangsadr
```

Verzichtet man auf den Programmnamen und gibt nur

```
CLOAD M anfangsadr
```

ein, so liest der Rechner das erste Maschinenprogramm ein, daß auf dem Band erscheint.

Prozessor: 8-Bit-CMOS-CPU SC61860
Speicherkapazität: System-ROM: 72 kB
System-RAM: ca. 1.1 kB
User-RAM: Standardvariablen: 208 Bytes
Programme/Daten: 6878 Bytes
Stack: Unterprogramme: 10 Stacks
Funktionen: 16 Stacks
FOR-NEXT-Loops: 5 Stacks
Daten: 8 Stacks
Rechengenauigkeit: 10 Ziffern in der Mantisse
2 Ziffern im Exponenten
Editierfunktionen: Cursor links, Cursor rechts, Zeile hoch,
Zeile runter, Zeichen einfügen,
Zeichen löschen
Anzeige: 24-Zeichen-Flüssigkristallanzeige (LCD)
Jedes Zeichen besteht aus 5*7 Punkten
Abmessungen (in mm): 2.7 Breite
5.2 Höhe
Auto Power Off: Automatische Abschaltung ca. 11 Minuten
nach dem letzten Tastendruck
Tastatur: 77 Tasten
Spannungsversorgung: 2 Lithium-Zellen à 6.0 V
Typ: CR-2032
Verbrauch: ca. 30 mW
Lebensdauer der
Batterien: ca. 120 Betriebsstunden bei normalen Be-
dingungen (pro Stunde 10 Minuten rechnen
und 50 Minuten anzeigen bei 20°C). Je nach
Benutzung des Rechners und der Batterie-
typen kann diese Zeit schwanken.
Abmessungen (in mm): 170 Länge, 72 Breite, 9.5 Tiefe
Gewicht: Einschl. Hard Cover und Batterien ca. 150 g

Die enorme Vergrößerung der Kapazität von Pocket Computer RAMs verlangt nach einem schnellen und zuverlässigen externen Speichermedium.

Das CE-140F 2.5" Diskettenlaufwerk (für double-sided Disketten) genügt dieser Anforderung und bietet Kompatibilität mit vielen unterschiedlichen SHARP Pocket Computern. Es ist durch BASIC-Kommandos einfach zu steuern und erlaubt zahlreiche I/O-Operationen.

Das CE-140F ist 118*39*145 mm groß, wiegt 650 g und bietet 128 kB (64 kB pro Diskettenseite) Speicherkapazität. Es wird von fünf 1.5 V Batterien gespeist; man kann auch den Netzadapter EA-160 benutzen.

Das Gerät eignet sich für die SHARP Pocket Computer PC-1403, PC-1360, PC-1425, PC-1460, PC-1280 und zukünftige Modelle.

Über eine 11-Pin-Buchse kann man gleichzeitig z.B. den Drucker CE-126P anschließen.

Befehle: FILES, LFILES, INIT, KILL, COPY, NAME, SET, MERGE, CHAIN, LOAD, SAVE, LOADM, SAVEM, INPUT#, PRINT#, CLOSE, OPEN.

Funktionen: EOF, DSKF, LOC, LOF.

Quelle: /18/



Alles für **SHARP** Computer
Fischel GmbH
Kaiser-Friedrich-Straße 54a
1000 Berlin 12 - Tel. 030/3236029

ATARI, CASIO, HEWLETT-PACKARD, PSION, SHARP

Super-Bestellschein

Hiermit bestätige ich:

Anzahl: Buch:

Atari

PC Portabla Anwendungshandbuch
ISBN 3-80374-046-5, VK = 48,- DM

Casio FX-850P
FX-850P Anwendungshandbuch
ISBN 3-80374-000-7, VK = 48,- DM
DX-850P in Deutsch/Karib.
ISBN 3-80374-020-1, VK = 48,- DM

Casio PB-1000 / 2000

PEB-1000 Tipps und Tricks Programmhandbuch

ISBN 3-80374-007-1, VK = 48,- DM

PB-1000 Anwendungshandbuch
ISBN 3-80374-080-4, VK = 48,- DM

PB-1000 Intern
ISBN 3-80374-028-7, VK = 48,- DM

PB-1000 Power Software
ISBN 3-80374-044-9, VK = 48,- DM

PB-1000 Systemhandbuch
ISBN 3-80374-047-3, VK = 48,- DM

PB-2000 Anwendungshandbuch
ISBN 3-80374-042-2, VK = 48,- DM

PB-2000 Intern
ISBN 3-80374-037-8, VK = 48,- DM

Hewlett Packard

HP-262/G Anwendungsprogramme

ISBN 3-80374-029-5, VK = 48,- DM

HP-262/G Programmammlung

ISBN 3-80374-041-4, VK = 48,- DM

PSION

Organizer II Anwendungshandbuch
ISBN 3-80374-048-1, VK = 48,- DM

Sharp PC-1500(A)/PC-1600

PC-1500/PC-1600 Hardvershandbuch
ISBN 3-80374-134-0, VK = 48,- DM

PC-1500 Tips und Tricks
ISBN 3-80374-12-2, VK = 48,- DM

Ergebnisbericht zum PC-1500A Maschinen sprache handbuch
ISBN 3-80374-173-1, VK = 48,- DM

PC-1500 Intern von Schmitt
ISBN 3-80374-110-1, VK = 48,- DM

PC-1600 Systemhandbuch
ISBN 3-80374-31-9, VK = 48,- DM

PC-1600 Anwendungshandbuch
ISBN 3-80374-55-9, VK = 48,- DM

PC-1600 Maschinen sprache handbuch
ISBN 3-80374-001-5, VK = 48,- DM

PC-1600 Tipps und Tricks Programmhandbuch
ISBN 3-80374-054-8, VK = 48,- DM

Die besten Programme für den PC-1600
ISBN 3-80374-040-8, VK = 48,- DM

Sharp PC-1401/02/03/21/50/75

PC-1401/02 Systemhandbuch
ISBN 3-924237-01-7, VK = 38,- DM

PC-1401/02 Maschinen sprache handbuch
ISBN 3-924237-11-1, VK = 48,- DM

PC-1402 Systemhandbuch
ISBN 3-924237-66-4, VK = 30,- DM

PC-1403 Anwendungshandbuch
ISBN 3-924237-65-3, VK = 48,- DM

PC-1403 Maschinen sprache handbuch
ISBN 3-924237-73-4, VK = 48,- DM

Die besten Programme für den PC-1403
ISBN 3-80374-039-2, VK = 48,- DM

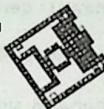
PC-1450 Anwendungshandbuch
ISBN 3-80374-18-1, VK = 48,- DM

PC-1450 Maschinen sprache handbuch
ISBN 3-924237-22-4, VK = 48,- DM

PC-1401/02/03 Tipps und Tricks Programmhandbuch
ISBN 3-924237-33-4, VK = 48,- DM

PC-1401/02/03/21/50/75/100/150/200/250/300/350/400/450/500/550/600/650/700/750/800/850/900/950/1000/1050/1100/1150/1200/1250/1300/1350/1400/1450/1500/1550/1600/1650/1700/1750/1800/1850/1900/1950/2000/2050/2100/2150/2200/2250/2300/2350/2400/2450/2500/2550/2600/2650/2700/2750/2800/2850/2900/2950/3000/3050/3100/3150/3200/3250/3300/3350/3400/3450/3500/3550/3600/3650/3700/3750/3800/3850/3900/3950/4000/4050/4100/4150/4200/4250/4300/4350/4400/4450/4500/4550/4600/4650/4700/4750/4800/4850/4900/4950/5000/5050/5100/5150/5200/5250/5300/5350/5400/5450/5500/5550/5600/5650/5700/5750/5800/5850/5900/5950/6000/6050/6100/6150/6200/6250/6300/6350/6400/6450/6500/6550/6600/6650/6700/6750/6800/6850/6900/6950/7000/7050/7100/7150/7200/7250/7300/7350/7400/7450/7500/7550/7600/7650/7700/7750/7800/7850/7900/7950/8000/8050/8100/8150/8200/8250/8300/8350/8400/8450/8500/8550/8600/8650/8700/8750/8800/8850/8900/8950/9000/9050/9100/9150/9200/9250/9300/9350/9400/9450/9500/9550/9600/9650/9700/9750/9800/9850/9900/9950/10000/10050/10100/10150/10200/10250/10300/10350/10400/10450/10500/10550/10600/10650/10700/10750/10800/10850/10900/10950/11000/11050/11100/11150/11200/11250/11300/11350/11400/11450/11500/11550/11600/11650/11700/11750/11800/11850/11900/11950/12000/12050/12100/12150/12200/12250/12300/12350/12400/12450/12500/12550/12600/12650/12700/12750/12800/12850/12900/12950/13000/13050/13100/13150/13200/13250/13300/13350/13400/13450/13500/13550/13600/13650/13700/13750/13800/13850/13900/13950/14000/14050/14100/14150/14200/14250/14300/14350/14400/14450/14500/14550/14600/14650/14700/14750/14800/14850/14900/14950/15000/15050/15100/15150/15200/15250/15300/15350/15400/15450/15500/15550/15600/15650/15700/15750/15800/15850/15900/15950/16000/16050/16100/16150/16200/16250/16300/16350/16400/16450/16500/16550/16600/16650/16700/16750/16800/16850/16900/16950/17000/17050/17100/17150/17200/17250/17300/17350/17400/17450/17500/17550/17600/17650/17700/17750/17800/17850/17900/17950/18000/18050/18100/18150/18200/18250/18300/18350/18400/18450/18500/18550/18600/18650/18700/18750/18800/18850/18900/18950/19000/19050/19100/19150/19200/19250/19300/19350/19400/19450/19500/19550/19600/19650/19700/19750/19800/19850/19900/19950/20000/20050/20100/20150/20200/20250/20300/20350/20400/20450/20500/20550/20600/20650/20700/20750/20800/20850/20900/20950/21000/21050/21100/21150/21200/21250/21300/21350/21400/21450/21500/21550/21600/21650/21700/21750/21800/21850/21900/21950/22000/22050/22100/22150/22200/22250/22300/22350/22400/22450/22500/22550/22600/22650/22700/22750/22800/22850/22900/22950/23000/23050/23100/23150/23200/23250/23300/23350/23400/23450/23500/23550/23600/23650/23700/23750/23800/23850/23900/23950/24000/24050/24100/24150/24200/24250/24300/24350/24400/24450/24500/24550/24600/24650/24700/24750/24800/24850/24900/24950/25000/25050/25100/25150/25200/25250/25300/25350/25400/25450/25500/25550/25600/25650/25700/25750/25800/25850/25900/25950/26000/26050/26100/26150/26200/26250/26300/26350/26400/26450/26500/26550/26600/26650/26700/26750/26800/26850/26900/26950/27000/27050/27100/27150/27200/27250/27300/27350/27400/27450/27500/27550/27600/27650/27700/27750/27800/27850/27900/27950/28000/28050/28100/28150/28200/28250/28300/28350/28400/28450/28500/28550/28600/28650/28700/28750/28800/28850/28900/28950/29000/29050/29100/29150/29200/29250/29300/29350/29400/29450/29500/29550/29600/29650/29700/29750/29800/29850/29900/29950/30000/30050/30100/30150/30200/30250/30300/30350/30400/30450/30500/30550/30600/30650/30700/30750/30800/30850/30900/30950/31000/31050/31100/31150/31200/31250/31300/31350/31400/31450/31500/31550/31600/31650/31700/31750/31800/31850/31900/31950/32000/32050/32100/32150/32200/32250/32300/32350/32400/32450/32500/32550/32600/32650/32700/32750/32800/32850/32900/32950/33000/33050/33100/33150/33200/33250/33300/33350/33400/33450/33500/33550/33600/33650/33700/33750/33800/33850/33900/33950/34000/34050/34100/34150/34200/34250/34300/34350/34400/34450/34500/34550/34600/34650/34700/34750/34800/34850/34900/34950/35000/35050/35100/35150/35200/35250/35300/35350/35400/35450/35500/35550/35600/35650/35700/35750/35800/35850/35900/35950/36000/36050/36100/36150/36200/36250/36300/36350/36400/36450/36500/36550/36600/36650/36700/36750/36800/36850/36900/36950/37000/37050/37100/37150/37200/37250/37300/37350/37400/37450/37500/37550/37600/37650/37700/37750/37800/37850/37900/37950/38000/38050/38100/38150/38200/38250/38300/38350/38400/38450/38500/38550/38600/38650/38700/38750/38800/38850/38900/38950/39000/39050/39100/39150/39200/39250/39300/39350/39400/39450/39500/39550/39600/39650/39700/39750/39800/39850/39900/39950/40000/40050/40100/40150/40200/40250/40300/40350/40400/40450/40500/40550/40600/40650/40700/40750/40800/40850/40900/40950/41000/41050/41100/41150/41200/41250/41300/41350/41400/41450/41500/41550/41600/41650/41700/41750/41800/41850/41900/41950/42000/42050/42100/42150/42200/42250/42300/42350/42400/42450/42500/42550/42600/42650/42700/42750/42800/42850/42900/42950/43000/43050/43100/43150/43200/43250/43300/43350/43400/43450/43500/43550/43600/43650/43700/43750/43800/43850/43900/43950/44000/44050/44100/44150/44200/44250/44300/44350/44400/44450/44500/44550/44600/44650/44700/44750/44800/44850/44900/44950/45000/45050/45100/45150/45200/45250/45300/45350/45400/45450/45500/45550/45600/45650/45700/45750/45800/45850/45900/45950/46000/46050/46100/46150/46200/46250/46300/46350/46400/46450/46500/46550/46600/46650/46700/46750/46800/46850/46900/46950/47000/47050/47100/47150/47200/47250/47300/47350/47400/47450/47500/47550/47600/47650/47700/47750/47800/47850/47900/47950/48000/48050/48100/48150/48200/48250/48300/48350/48400/48450/48500/48550/48600/48650/48700/48750/48800/48850/48900/48950/49000/49050/49100/49150/49200/49250/49300/49350/49400/49450/49500/49550/49600/49650/49700/49750/49800/49850/49900/49950/50000/50050/50100/50150/50200/50250/50300/50350/50400/50450/50500/50550/50600/50650/50700/50750/50800/50850/50900/50950/51000/51050/51100/51150/51200/51250/51300/51350/51400/51450/51500/51550/51600/51650/51700/51750/51800/51850/51900/51950/52000/52050/52100/52150/52200/52250/52300/52350/52400/52450/52500/52550/52600/52650/52700/52750/52800/52850/52900/52950/53000/53050/53100/53150/53200/53250/53300/53350/53400/53450/53500/53550/53600/53650/53700/53750/53800/53850/53900/53950/54000/54050/54100/54150/54200/54250/54300/54350/54400/54450/54500/54550/54600/54650/54700/54750/54800/54850/54900/54950/55000/55050/55100/55150/55200/55250/55300/55350/55400/55450/55500/55550/55600/55650/55700/55750/55800/55850/55900/55950/56000/56050/56100/56150/56200/56250/56300/56350/56400/56450/56500/56550/56600/56650/56700/56750/56800/56850/56900/56950/57000/57050/57100/57150/57200/57250/57300/57350/57400/57450/57500/57550/57600/57650/57700/57750/57800/57850/57900/57950/58000/58050/58100/58150/58200/58250/58300/58350/58400/58450/58500/58550/58600/58650/58700/58750/58800/58850/58900/58950/59000/59050/59100/59150/59200/59250/59300/59350/59400/59450/59500/59550/59600/59650/59700/59750/59800/59850/59900/59950/60000/60050/60100/60150/60200/60250/60300/60350/60400/60450/60500/60550/60600/60650/60700/60750/60800/60850/60900/60950/61000/61050/61100/61150/61200/61250/61300/61350/61400/61450/61500/61550/61600/61650/61700/61750/61800/61850/61900/61950/62000/62050/62100/62150/62200/62250/62300/62350/62400/62450/62500/62550/62600/62650/62700/62750/62800/62850/62900/62950/63000/63050/63100/63150/63200/63250/63300/63350/63400/63450/63500/63550/63600/63650/63700/63750/63800/63850/63900/63950/64000/64050/64100/64150/64200/64250/64300/64350/64400/64450/64500/64550/64600/64650/64700/64750/64800/64850/64900/64950/65000/65050/65100/65150/65200/65250/65300/65350/65400/65450/65500/65550/65600/65650/65700/65750/65800/65850/65900/65950/66000/66050/66100/66150/66200/66250/66300/66350/66400/66450/66500/66550/66600/66650/66700/66750/66800/66850/66900/66950/67000/67050/67100/67150/67200/67250/67300/67350/67400/67450/67500/67550/67600/67650/67700/67750/67800/67850/67900/67950/68000/68050/68100/68150/68200/68250/68300/68350/68400/68450/68500/68550/68600/68650/68700/68750/68800/68850/68900/68950/69000/69050/69100/69150/69200/69250/69300/69350/69400/69450/69500/69550/69600/69650/69700/69750/69800/69850/69900/69950/70000/70050/70100/70150/70200/70250/70300/70350/70400/70450/70500/70550/70600/70650/70700/70750/70800/70850/70900/70950/71000/71050/71100/71150/71200/71250/71300/71350/71400/71450/71500/71550/71600/71650/71700/71750/71800/71850/71900/71950/72000/72050/72100/72150/72200/72250/72300/72350/72400/72450/72500/72550/72600/72650/72700/72750/72800/72850/72900/72950/73000/73050/73100/73150/73200/73250/73300/73350/73400/73450/73500/73550/73600/73650/73700/73750/73800/73850/73900/73950/74000/74050/74100/74150/74200/74250/74300/74350/74400/74450/74500/74550/74600/74650/74700/74750/74800/74850/74900/74950/75000/75050/75100/75150/75200/75250/75300/75350/75400/75450/75500/75550/75600/75650/75700/75750/75800/75850/75900/75950/76000/76050/76100/76150/76200/76250/76300/76350/76400/76450/76500/76550/76600/76650/76700/76750/76800/76850/76900/76950/77000/77050/77100/77150/77200/77250/77300/77350/77400/77450/77500/77550/77600/77650/77700/77750/77800/77850/77900/77950/78000/78050/78100/78150/78200/78250/78300/78350/78400/78450/78500/78550/78600/78650/78700/78750/78800/78850/78900/78950/79000/79050/79100/79150/79200/79250/79300/79350/79400/79450/79500/79550/79600/79650/79700/79750/79800/79850/79900/79950/80000/80050/80100/80150/80200/80250/80300/80350/80400/80450/80500/80550/80600/80650/80700/80750/80800/80850/80900/80950/81000/81050/81100/81150/81200/81250/81300/81350/81400/81450/81500/81550/81600/81650/81700/81750/81800/81850/81900/81950/82000/82050/82100/82150/82200/82250/82300/82350/82400/82450/82500/82550/82600/82650/82700/8

**LISTE der BASIC-BEFEHLE
des SHARP-CE-140F ,
des superschnellen Dis-
kettenlaufwerkes für die
SHARP-Taschencomputer .**



Dieses Diskettenlaufwerk
eignet kann mit den nach-
folgend aufgeführten neu-
en Taschencomputern ver-
wendet werden :

SHARP-PC-1280, PC-1403,
PC-1425/60/75, PC-1460,
sowie weiteren zukünfti-
gen Modellen.

Befehl/Anweisung * Erklärung (P=programmgesteuert / M=manuell)

CHAIN

[P] - lädt ein Programm von der Diskette und startet es
automatisch. Im Computer gespeicherte Programme
werden dabei gelöscht.
> CHAIN 'X:Dateiname' [,Zeilen-Nr. o. Marke] <

CLOSE

[M,P] - schliesst eine (mehrere) Datei(en). (Nummern zwi-
schen 2...7 .)
> CLOSE [# Nummer, # Nummer...] <

COPY

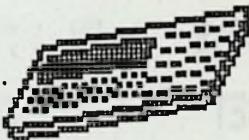
[M] - kopiert den Inhalt einer Diskette auf eine andere
oder einen anderen Bereich auf dieselbe Diskette.
> COPY 'X:Dateiname1' TO 'Y:Dateiname2'
[oder 'X:...'] <

DSKF

[M,P] - zeigt den freien Speicherbereich der Diskette in
Bytes auf dem Display an.
> DSKF(1) <

EOF

[M,P] - findet das Ende einer Datei auf.
> EOF (Datei-Nr.) <



FILES

[M] - zeigt die Namen und Merkmale der angegebenen Da-
teien auf dem Display an.
> FILES ['X:[Dateiname]'] <

INPUT #

[M,P] - weist Daten von der Diskette einer angegebenen
Variablen zu.
> INPUT #Datei-Nr.,Variable [,Variable,...] <

INIT

[M] - initialisiert (formatiert) eine Diskette.
> INIT 'X:' <

KILL

[M] - löscht eine Datei.
> KILL 'X:Dateiname' <

LFILES

[M] - drückt die Namen und Merkmale der angegebenen Da-
teien aus.
> LFILES ['X:[Dateiname]'] <

Sharp Microcomputer

..... Fischel GmbH

Kaiser-Friedrich-Str. 54 a

D - 1000 Berlin 12

..... Tel. 030 / 323 60 29

Mo - Fr 10 - 18.30, Sa - 14 h

ISBN 3-924327-56-4

SCHMID SEEFELD

71 08 37