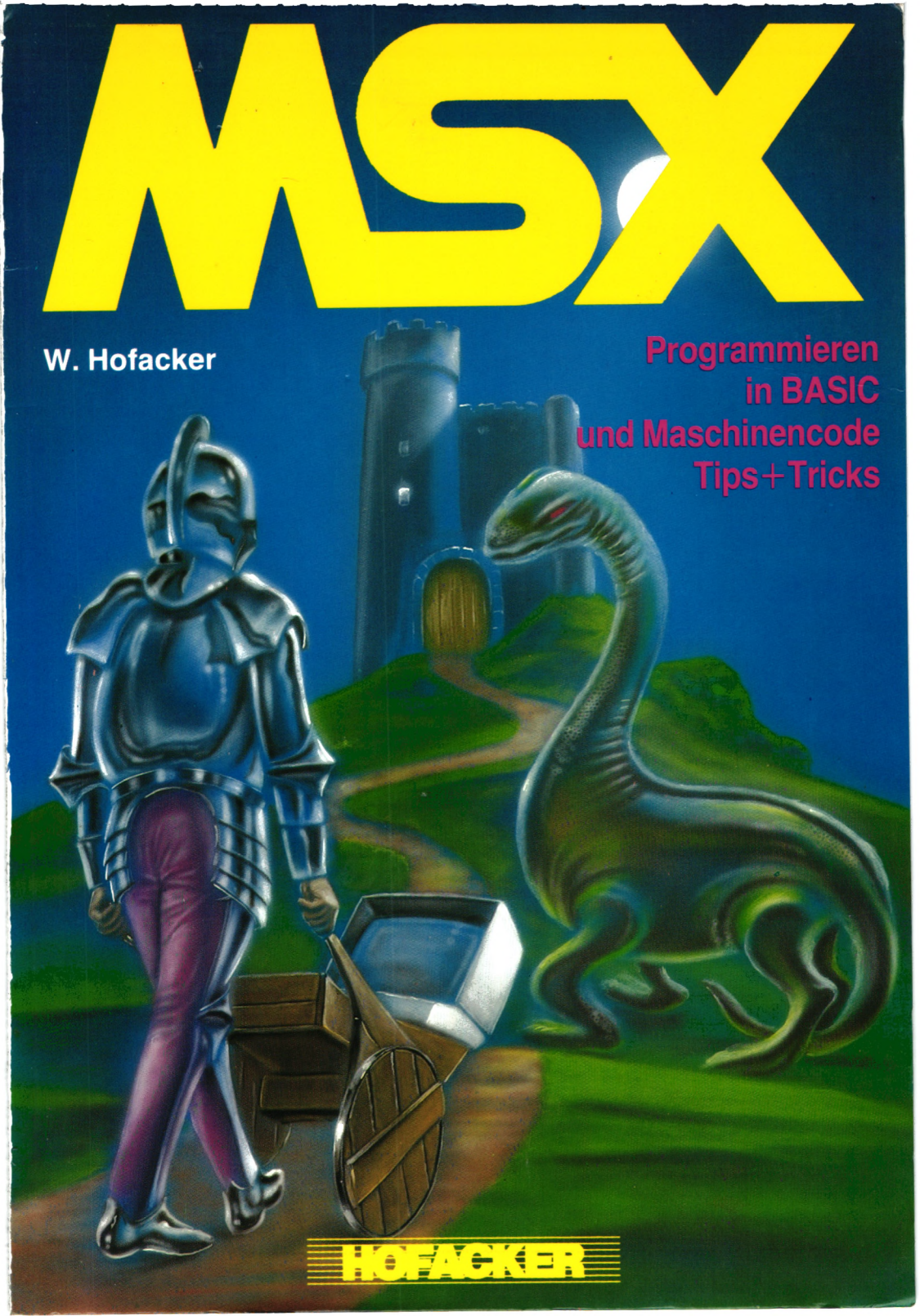
MSX

W. Hofacker

Programmieren
in BASIC
und Maschinencode
Tips+Tricks



HOFACKER

ISBN 3-88963-230-0

1. Auflage 1985

Gedruckt in der Bundesrepublik Deutschland – Printed in West-Germany – Imprime'en RFA.

W. Hofacker

MSX

Programmieren in BASIC und Maschinencode

Tips und Tricks

Inhaltsverzeichnis

Vorwort

MSX-Rechner sind seit ca. einem Jahr ein großer Erfolg in Japan. Nun werden diese Rechner auch auf unserem Markt zu finden sein.

Der große Unterschied zu den bisherigen Personal- und Home Computern liegt in der Standarisierung der Hard- und Software. MSX-Rechner von verschiedenen Herstellern werden sich zwar am äußeren Erscheinungsbild und an speziellen Zusätzen unterscheiden, sie sind aber voll Software kompatibel.

Damit ist es endlich vorbei, daß Software, entwickelt für ein bestimmtes Modell, beim Erscheinen des Folgemodells völlig wertlos wurde.

Das vorliegende Buch soll zeigen, wie man einen Einstieg in das Programmieren dieser Rechner erlangt. Dabei stehen nicht Spiele im Vordergrund, sondern Anwendungen, wie Lernen mit dem Computer, Listen generieren oder Rechnungen schreiben.

Es folgt eine Einführung in Maschinensprache, die Programmierung eine I/O Bausteines und des Tongenerators.

Der Vollständigkeit halber sind alle BASIC-Befehle zusammengestellt und teilweise mit Beispielen versehen.

Ich wünsche dem Leser viel Spaß mit MSX.

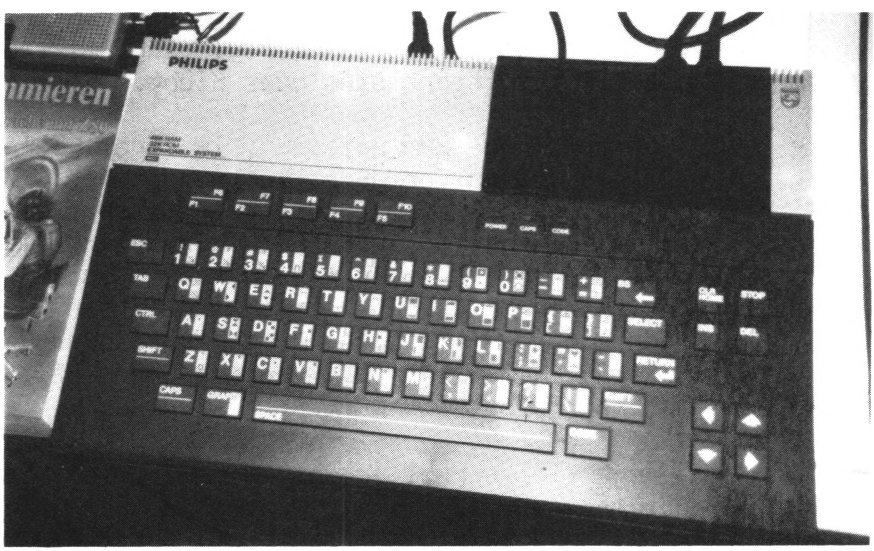
Holzkirchen, Sommer 1985. Winfried Hofacker

1 Programmieren in BASIC

1.1 Was heißt Programmieren ?

Ein MSX-Rechner wird für viele der erste Kontakt mit einem programmierbaren Rechner sein. Deshalb soll zuerst einmal die Frage geklärt werden, was es bedeutet, einen Rechner zu programmieren.

Ein Programm ist die Umsetzung eines Problems in eine Sprache, die der Rechner versteht. Zuerst muß also ein Problem, eine Aufgabe bestehen, die vom Rechner gelöst werden soll.



1.1 Ein MSX-Rechner

Diese Aufgaben können reine mathematische Aufgaben, wie zum Beispiel Auswertung von Formeln oder wirtschaftlichen Aufgaben der Mess- und Regeltechnik, wie zum Beispiel die laufende Überwachung einer Innen- und Außentemperatur sein.

Die Lösung solcher Aufgaben mit dem Rechner bedingt allerdings zweierlei. Zuerst muß ein Lösungsweg gesucht werden. Das Auffinden eines solchen Lösungsweges, auch Algorithmus genannt, ist nicht immer einfach.

Betrachten wir hierzu ein einfaches Beispiel:

Wir haben eine Liste mit vielen Namen. Aus dieser Liste wollen wir einen Namen aussuchen.

Was für Lösungsmöglichkeiten gibt es?

Für die Beantwortung dieser Frage überlegt man zuerst einmal, wie man diese Aufgabe ohne Rechner löst. Sind es nur wenige Namen, so wird man oben in der Liste anfangen, jeden Namen mit dem gesuchten vergleichen, solange, bis man den richtigen Namen gefunden hat. Dieses Suchen kann sehr schnell gehen, wenn der gesuchte Name ganz oben in der Liste steht, oder sehr lange, wenn er erst am Ende der Liste auftaucht. Dabei ist es aber gleichgültig, ob die Namen alphabetisch sortiert sind oder nicht.

Sind es viele Namen, wie zum Beispiel in einem Telefonbuch, dann dauert dieses Verfahren zu lange. Man wird dann einen anderen Weg versuchen. Sortiert man erst alle Namen nach dem Alphabet, so kann man hinterher viel schneller suchen. Man wird zuerst den gesuchten Namen mit einem Namen in der Mitte der Liste vergleichen, und somit feststellen, ob der gesuchte Name davor oder dahinter steht. Je nachdem wird man in der vorderen Hälfte oder in der hinteren Hälfte nachsuchen.

Dieses Suchen geht viel schneller, es bedingt aber, daß die Namen sortiert sind. Dieses Sortieren kann aber ebenfalls sehr lange dauern. Seit Beginn der programmierbaren Rechner sind viele Sortierverfahren entwickelt worden, wie zum Beispiel BUBBLE, RIPPLE, SHELL, HEAP oder

QUICK-Sort (2), um nur einige zu nennen. Ein Programmbeispiel dazu zeigt auch Programm TERMINKALENDER in Kapitel 3.

An diesem Beispiel kann aber auch eine Eigenschaft eines Computers zeigen, die sehr wichtig ist. Angenommen, wir suchen den Namen Schmidt in der Liste, dort ist aber nur ein Schmied vorhanden, so wird diesen Namen zwar ein Mensch, nicht aber der Computer finden. Dessen Auftrag lautete nämlich, Schmidt und nicht Schmied zu suchen.

Kann so ein Fall auftreten, so muß auch dafür eine Lösungsmöglichkeit gesucht werden. Eine wäre zum Beispiel als Suchwort Schmi* einzugeben. Das bedeutet, suche alle Namen, die mit Schmi anfangen. Dann wird der Rechner also Schmidt, Schmied, aber auch Schmielmayer finden.

Ein Rechner ist, entgegen allen anderen Behauptungen, nicht intelligent, sondern nur so intelligent, wie sein Programmierer. Ich halte es für absurd, wenn durch Einsatz eines Mikroprozessors, einem Rechner, aus einer Waschmaschine plötzlich eine intelligente Waschmaschine wird.

Bisher haben Programmierer die Tatsache, daß die breite Öffentlichkeit keine Ahnung vom Programmieren hat, dazu benutzt, eigene Fehler einfach dem Computer zuzuschreiben.

"Dieser blöde Computer schreibt für einen Schuldbetrag von 0,00 DM eine Mahnung aus."

Der Fehler liegt nicht beim Computer, sondern beim Programmierer.

Wenn man einen Lösungsweg gefunden hat, so braucht man ein zweites, eine Sprache in welcher dieser Lösungsweg beschrieben wird. Natürlich muß es eine Sprache sein, die der Rechner auch versteht. Unser MSX versteht "The Beginners All Purpose Symbolic Instruction Code" oder kurz BASIC. Von dieser, Anfang der 60er Jahre am College of Dartmounth entwickelten Sprache, gibt es viele Dialekte. Das MSX BASIC, von MICROSOFT entwickelt, ist

eine sehr umfangreiche BASIC Version, mit der sehr leicht programmiert werden kann. Es weicht an einigen Stellen von anderen BASIC-Dialekten ab, aber nicht so weit, daß es mit diesen nicht mehr vergleichbar wäre. BASIC ist eine interpretative Sprache, ein Interpreter.

Das bedeutet, daß eine direkte Anweisung wie

```
PRINT 3*4
```

gleich ausgeführt wird.

Ein Programm ist eine Folge von solchen Anweisungen, die nach Zeilennummern geordnet sind. Bei einem Programmablauf wird jede Zeile interpretiert, und sofort ausgeführt. Auf den Unterschied zu Compilersprachen soll hier nicht eingegangen werden. Auf die Sprachelemente von BASIC soll ebenfalls nicht eingegangen werden. Diese sind in Kapitel 5 alphabetisch zusammengestellt.

Fassen wir noch einmal zusammen:

Programmieren bedeutet

1. Lösungswege für ein Problem finden,
2. Beschreiben dieser Wege in einer Sprache, die der Computer versteht.

1.2 Programmieren in BASIC, ein Beispiel

In diesem Kapitel soll an einem Beispiel gezeigt werden, wie man von der Aufgabenstellung zum fertigen Programm fortschreitet.

Die Aufgabenstellung ist bewußt einfach gehalten, um den Aufbau des Programms leichter verständlich zu machen. Sie lautet: Planung der Ausgaben eines Haushalts.

Wie wir schon im ersten Kapitel gesehen haben, müssen wir unseren Rechner alle Schritte genau vorgeben, denn er kann ja nur das ausführen, was wir ihm eingegeben haben. Vor dem Programmieren machen wir eine sogenannte Systemanalyse.

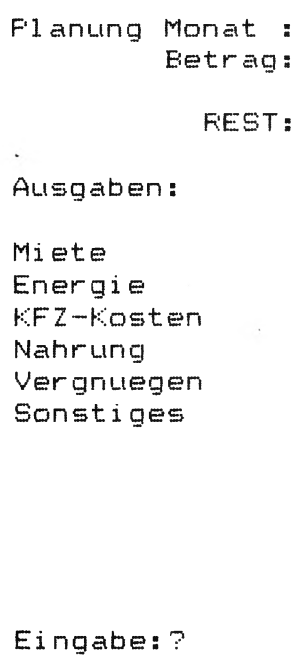
Folgende Fragen müssen dabei beantwortet werden:

1. Welches sind die Eingabedaten?
2. Wie werden diese verarbeitet?
3. Welche Daten sollen ausgegeben werden?
4. In welcher Form werden die Daten gespeichert?
5. Gibt es bei Berechnungen mehrdeutige Lösungen?
6. Welche mathematischen Verfahren werden angewendet?

Dieser Fragenkatalog kann sich bei einzelnen Problemen noch erheblich erweitern. Es kann zum Beispiel mehrere Lösungsmöglichkeiten geben. Dann ist zu entscheiden welche die beste ist. Wir werden später bei anderen Programmen verschiedene Lösungsmöglichkeiten mit ihren Vor- und Nachteilen betrachten. Nun zu unserem Problem, Planung der Ausgaben eines Haushalts.

1.3 Ein kleines Budgetplanungsprogramm

Diesem Programm liegt folgende Idee zu Grunde. Auf dem Bildschirm erscheint ein Blatt Papier, in das Eintragungen gemacht werden können. Damit soll eine Planung der Haushaltsausgaben durchgeführt werden. Die Abbildung 1.2 zeigt das leere Blatt.



1.2 Blatt zur Planung von Ausgaben

Für die Eingabe der Werte soll nicht eine der üblichen Menütechniken, sondern eine, dem Dialog des Menschen mit dem Computer besser angepasste Möglichkeit verwendet werden.

Einer der Kostenpunkte ist die Miete. Die Eingabe erfolgt nun in der Form, daß das Wort MIETE, gefolgt von

einem Leerzeichen und dem Betrag, eingegeben wird. Also z. B.

```
Miete 856.50
```

Der Computer sucht nun das Wort MIETE und addiert den eingegebenen Betrag zu den Ausgaben und zieht ihn vom vorhandenem Betrag ab. Auf diese Weise können die einzelnen Kostenpunkte in beliebiger Reihenfolge eingegeben werden.

Zuerst wird der Monat und dann der zur Verfügung stehende Bertrag mit

```
Monat Januar
Betrag 2200.00
```

eingegeben.

Die Abbildung 1.3 zeigt das Blatt nach einigen Eingaben.

```
Eingabe:? Monat Januar

Planung Monat : Januar
        Betrag:                 2200.00

          REST:                 1334.50

Ausgaben:

Miete                            865.50
Energie
KFZ-Kosten
Nahrung
Vergnuegen
Sonstiges

Eingabe:?
```

1.3 Blatt nach einigen Eingaben

Das Programm zeigt Abbildung 1.4.

Das Feld W$ ist das Wörterbuch, in das alle verwendeten Worte eingetragen sind und aus dem sie bei der Eingabe gesucht werden. Das Feld K ist das Kostenfeld. In beiden ist mit L=8 Platz für 8 Einträge geschaffen worden.

In den Zeilen 500 bis 570 wird das Blatt auf den Bildschirm ausgegeben. In Zeile 580 wartet das Programm auf eine Eingabe.

Ist diese erfolgt, so wird die eingegebene Zeichenkette untersucht. In den Zeilen 600 bis 620 wird nach dem, die beiden Worte trennenden Zeichen gesucht. Wird es gefunden, so wird die Zeichenkette A$ in den linken Teil AL$ und den rechten Teil AR$ zerlegt. Wird kein Leerzeichen gefunden, so wird in die Fehlerroutine ab Zeile 900 gesprungen.

Das Programm vergleicht nun das eingegebene Wort mit den in W$ gespeicherten Worten. Wird das Wort gefunden, so wird der Betrag in einen numerischen Wert gewandelt und eingetragen. Der bei der Suche verwendete Index I gibt gleichzeitig die Stelle an, wo der Betrag auf den Bildschirm ausgegeben wird.

```
100  SCREEN 0:WIDTH 40:KEY OFF
110  DEFINT I,L:V$=SPACE$(30)
120  L=8: DIM W$(L),K(L)
130  W$(1)="Monat"
140  W$(2)="Betrag"
150  W$(3)="Miete"
160  W$(4)="Energie"
170  W$(5)="KFZ-Kosten"
180  W$(6)="Nahrung"
190  W$(7)="Vergnuegen"
200  W$(8)="Sonstiges"
500  REM Maske
510  CLS
520  LOCATE 2,0:PRINT"Planung Monat :"
```

```
530  LOCATE 10,2:PRINT"Betrag:"
540  LOCATE 2,6:PRINT"Ausgaben:"
545  LOCATE 12,4:PRINT"Rest:"
550  FOR I=3 TO L
560    LOCATE 2,I+5:PRINT W$(I)
570  NEXT I
580  LOCATE 2,20:INPUT "Eingabe:";A$
590  L1=LEN(A$)
600  FOR I=1 TO L1
610    IF MID$(A$,I,1)=" " THEN 640
620  NEXT I
630  GOTO 900
640  AL$=LEFT$(A$,I-1)
650  AR$=RIGHT$(A$,L1-I)
700  REM Suchen
710  FOR I=1 TO L
720   IF AL$=W$(I) THEN 1000
730  NEXT I
900  REM Fehlerhafte Eingabe
920  LOCATE 2,20:PRINT V$
930  LOCATE 2,20:PRINT A$+" ?"
940  A$=INKEY$:IF A$="" THEN 940
950  LOCATE 2,20:PRINT V$
960  GOTO 580
1000 REM Eintragen
1010 IF I=1 THEN LOCATE 18,0:PRINT AR$:GOTO 950
1020 IF I>2 THEN 1100
1030 K(1)=VAL(AR$)
1040 LOCATE 30,2:PRINT USING "#####.##";K(1)
1050 GOTO 1115
1100 K(I)=VAL(AR$)
1110 LOCATE 30,I+5:PRINT USING "#####.##";K(I)
1115 R=0
1120 FOR I=3 TO L
1130   R=R+K(I)
1140 NEXT I
1150 R=K(1)-R
1160 LOCATE 30,4:PRINT USING "#####.##";R
1170 GOTO 950
```

Folgende Erweiterungen des Programms sollen als Programmierübung dienen.

Die Eingabe soll so vereinfacht werden, daß bei den Worten nur die ersten beiden Buchstaben eingegeben werden müssen, wie zum Beispiel So für Sonstiges.

Die Eingabe neuer Beträge soll zu den bisherigen Eingaben hinzuaddiert werden.

Das Feld K soll auf Kassette gespeichert werden.

1.4 Korrigieren und Ändern von Programmen

Die Programme des MSX-Rechners lassen sich auch sehr leicht editieren. Das heißt, fehlerhafte Zeilen abändern, löschen usw. Schon bei der Eingabe wird die richtige Schreibweise überwacht. Selbst wenn die Zeile richtig in das Programm übernommen wurde, so ist damit noch nicht gesagt, ob sie auch sachlich richtig ist. Kein Programm ist von Anfang an fehlerfrei. Fehler müssen gesucht und beseitigt werden.

Zum Editieren von Programmzeilen benötigt man hauptsächlich die Cursortasten, die INS- und die DEL-Taste.

Das Korrigieren und Ändern von MSX-Programmen ist sehr einfach, da der für BASIC verwendete Editor Bildschirm orientiert ist. Das bedeutet, daß der Cursor frei über den Bildschirm bewegt werden kann. Als Beispiel soll folgende Eingabe geändert werden:

```
590 K(M,M)=K(M,N)+W
```

Das zweite M in K(M,M) soll in N umgewandelt werden. Mit LIST 590 wird die Zeile auf dem Bildschirm angezeigt. Es können auch mehrere Zeilen auf dem Bildschirm sichtbar sein. Mit den Cusortasten wird der Cursor in die Zeile 590 über das M desetzt und an dieser Stelle N eingegeben.

Achtung! Wenn alle Fehler in einer Zeile beseitigt sind, muß unbedingt die RETURN Taste betätigt werden, damit die Zeile richtig in das Programm zurückgeschrieben wird. Geht man mit dem Cursor in eine andere Zeile, so ist die Änderung nur auf dem Bildschirm, nicht aber im Speicher vorgenommen worden. Beim Betätigen der RETURN Taste braucht der Cursor nicht am Ende des Textes stehen, er kann an einer beliebigen Stelle in der Zeile sein.

Wenn in eine Zeile Zeichen eingefügt werden sollen, so muß mit der INS Taste Platz für diese Zeichen geschaffen werden.

Oft kommt es in Programmen vor, daß mehrere Zeilen sich nur in Kleinigkeiten unterscheiden. Dann kann man sich die Schreibarbeit dadurch erleichtern, daß man diese Zeile einmal schreibt und mit RETURN speichert. Nun geht man mit dem Cusor zurück in die Zeile und ändert die Zeilennummer. Nach RETURN ist diese Zeile nochmals mit der geänderten Zeilennummer gespeichert und kann korrigiert werden.

1.5 Fehlersuche in Programmen

Im Prinzip kann man davon ausgehen, daß kein Programm auf Anhieb fehlerfrei ist. Durch die Syntaxprüfung sind Schreibfehler schon bei der Eingabe gefunden worden. Logische Fehler werden erst im Programmlauf gefunden.

Wenn ein Programm gestartet wurde, und es erscheint längere Zeit keine Ausgabe auf dem Bildschirm, so kann das Programm in einer unendlichen Schleife stecken. Das Programm kann dann durch CTRL STOP unterbrochen werden. Dabei wird die Zeilennummer ausgegeben bei welcher die Programmunterbrechung stattfand. In dieser Gegend muß man dann nach Möglichkeit einer unendlichen Schleifenbildung suchen. Meistens ist es ein GOTO-Befehl mit einer falschen Zeilennummer oder eine nicht erfüllte

IF-THEN-ELSE Bedingung wie im folgenden Beispiel:

```
10 LET I = 1
20 IF I = 100 THEN GOTO 50
30 I = I + 2
40 GOTO 20
50 STOP
```

Die Bedingung I = 100 ist nie erfüllt, da I immer eine ungerade Zahl ist. Nach I=99 wird I=101. Damit stellt dies eine unendliche Schleife dar. Richtig wäre in Zeile 20 die Abfrage gewesen.

```
20 IF I > = 100 THEN GOTO 50
```

Damit wäre die Schleife mit I = 101 verlassen worden.

Meistens ist aber nicht klar ersichtlich, wo im Programm so eine Schleife auftritt. Dann wird man versuchen, durch PRINT-Anweisungen oder durch Einfügen des STOP-Befehls den Fehler einzukreisen.

Ein anderer beliebter Fehler ist das unabsichtliche Ändern des Wertes einer Variablen. In dem Programm in Abb. 1.4 ist I eine Variable, deren Wert den Index für das Suchen des Leerzeichens ist. Dieser Wert darf während des Programmlaufes nicht geändert werden. Allzuleicht vergisst man dies und verwendet I als Laufparameter in einer Schleife. Die Ergebnisse sind dann reine Zufallszahlen.

BASIC ist eine interpretative Sprache. Das heißt, das eingegebene Programm oder Teile des Programms können sofort ausgeführt werden. Dies sollte man dazu benutzen, kleine Programmabschnitte (Module) sofort zu testen. Dies ist wesentlich einfacher, als ein vollständiges Programm auf Fehler zu untersuchen.

Leider führt dies aber auch dazu, daß Programme nur am Rechner entwickelt werden und nebenher keine Notizen zum Programmablauf gemacht werden. Bevor man mit dem Schreiben eines Programms beginnt, sollten die wesentlichen

Grundzüge des Programmablaufs schriftlich festgehalten werden. Dies erleichtert wesentlich die Fehlersuche.

Bei der Fehlersuche helfen die Befehle TRON und TROFF. Durch sie kann man feststellen, in welchen Bahnen sich das Prgramm bewegt. Bei der Fehlersuche sollten auch alle ON ERROR GOTO Befehle entfernt werden, da diese meist die eigentliche Fehlerzeile verschleiern.

Viel zu wenig wird beachtet, daß nach einem Anhalten des Programms mit einer Fehlermeldung, die augenblicklichen Werte der Variablen noch erhalten sind. Durch die direkte Eingabe eines PRINT Befehls lassen sich die Inhalte ausdrucken und man kann so feststellen, welche Variable den Fehler, zum Beispiel bei Überschreiten von Feldgrenzen, verursacht hat.

Auf eine weitere Fehlermöglichkeit, die Instabilität der angewendeten mathematischen Verfahren, soll hier nicht eingegangen werden.

Im Allgemeinen kann man sagen, daß die Fehlersuche mindestens ebenso lange dauert, wie das Schreiben des Programms.

Noch ein Wort in eigener Sache

Nach den Erfahrungen, die ich mit anderen Büchern gemacht habe, konnte ich folgendes feststellen. Die Zahl der echten Programmfehler war kleiner als 10. Dennoch bekam ich wütende Briefe und Anrufe, daß das Buch "voller Fehler" ist. Gerade der Computerneuling ist versucht, den Fehler überall woanders zu suchen, nur nicht bei sich selbst. Dabei treten die Fehler meistens beim Eintippen auf. Auf dem Bildschirm überliest man dies leicht. Besser lassen sich Fehler auf einem Ausdruck finden. Eine Möglichkeit solche Fehler zu finden ist folgende: Eine Person liest das Programm vom Bildschirm ab, während eine andere Person das Programm im Buch mitliest.

Die Programme in diesem Buch sind meistens schon auf anderen Rechnern gelaufen. Dabei sind möglichst alle

Schleifen und Eingaben getestet worden. Es kann aber dennoch sein, daß bei einer bestimmten Eingabesituation ein Fehler auftritt. Was aber nicht sein kann, das ist, daß ein Programm gar nicht läuft.

Also, wenn ein Programm nicht läuft:

1. Nach Schreibfehlern suchen
2. Handbuch zu Rate ziehen. Versuchen den Fehler einzukreisen.
3. Wenn alles nichts hilft, schriftliche Anfrage an Verlag. Wenn möglich, Programmausdruck beilegen.

2

Programmbeispiele

Die folgenden Programmbeispiele sollen eine Übersicht geben, wie man die MSX Rechner zur Lösung von Problemen einsetzen kann.

Dabei soll vor allem gezeigt werden, wie Daten gespeichert, wie die Funktionstasten programmiert werden und wie spezielle Programmiertechniken angewendet werden. Die Programme sind so geschrieben, daß sie leicht in andere Programme umgewandelt werden können. Der Terminkalender läßt sich leicht in eine Stichwortkartei umwandeln.

2.1 Vokabeln lernen

Mit dem Programm in Abbildung 2.1 kann man Vokabeln lernen und seine Lernleistung prüfen. Die Worte werden wie in ein Wörterbuch eingetragen. Erst die deutsche Bezeichnung, dann die englische (französische, lateinische usw.). Beim Lern- und Abhörteil wird durch eine Zufallszahl zwischen 1 und N, der Zahl der Einträge, ein Begriff ausgewählt. Vorher kann gewählt werden, ob dies ein deutscher oder englischer Ausdruck ist. Der andersprachige Begriff wird eingegeben und mit dem gespeicherten Wort verglichen. Stimmen beide überein, kann man weitermachen oder aufhören. Hat man aber ein falsches Wort oder eine falsche Schreibweise eingegeben, so wird eine nochmalige Eingabe verlangt. Nach der 3. falschen Eingabe wird das richtige Wort ausgegeben.

Alle gegebenen Begriffe können auf Bildschirm ausgegeben werden.

```
100 DIM D$(100):DIM E$(100)
105 MAXFILES=1:WIDTH 40
110 E$="Englisch: ":D$="Deutsch : "
115 N=0:L$=SPACE$(15)
120 CLS
130 LOCATE 10,2:PRINT"F1:Eingeben"
140 LOCATE 10,3:PRINT"F2:Lernen"
150 LOCATE 10,4:PRINT"F3:Ausgeben"
160 LOCATE 10,5:PRINT"F4:Laden/Speichern"
170 LOCATE 10,6:PRINT"F5:Ende"
180 ON KEY GOSUB 1000,2000,3000,4000,5000
190 KEY(1) ON:KEY(2) ON:KEY(3) ON
200 KEY(4) ON:KEY(5) ON
210 GOTO 210
1000 REM Eingeben
1010 CLS
1020 LOCATE 5,2:PRINT D$;
1030 LINE INPUT D$(N)
1040 LOCATE 5,4:PRINT E$;
1050 LINE INPUT E$(N)
1060 N=N+1:GOSUB 1070:GOTO 1010
1070 LOCATE 10,8:PRINT "MEHR ( /N)";
1080 A$=INKEY$:IF A$="" THEN 1080
1090 IF A$="N" OR A$="n" THEN 120 ELSE RETURN
2000 REM Lernen
2010 KEY(1) OFF:KEY(2) OFF
2020 CLS
2030 LOCATE 10,2:
2040 PRINT "F1:Deutsch"
2050 LOCATE 10,3
2060 PRINT "F2:Englisch"
2070 ON KEY GOSUB 2200,2500
2080 KEY(1) ON:KEY(2) ON:
2090 GOTO 2090
2200 REM Deutsch-Englisch
2205 L=0
2210 GOSUB 2400
2220 CLS:LOCATE 5,2:PRINT D$; D$(R)
2230 LOCATE 5,3:PRINT E$;
```

```
2240 LINE INPUT I$
2250 IF I$=E$(R) THEN LOCATE 5,5:PRINT "RICHTIG"
:GOSUB 1070:GOTO 2210
2260 LOCATE 5,5:PRINT"FALSCH"
2270 L=L+1:LOCATE 14,3
2280 IF L=3 THEN PRINT E$(R):GOSUB 1070 ELSE PRI
NT L$
2300 IF L=3 THEN 2205 ELSE 2230
2400 REM ZUFALLSZAHL
2405 I=TIME
2410 R=INT(RND(-I)*N)
2420 RETURN
2500 REM Englisch-Deutsch
2505 L=0
2510 GOSUB 2400
2520 CLS:LOCATE 5,3:PRINT E$; E$(R)
2530 LOCATE 5,2:PRINT D$;
2540 LINE INPUT I$
2550 IF I$=D$(R) THEN LOCATE 5,5:PRINT "RICHTIG"
:GOSUB 1070:GOTO 2510
2560 LOCATE 5,5:PRINT"FALSCH"
2570 L=L+1:LOCATE 14,2
2580 IF L=3 THEN PRINT D$(R):GOSUB 1070 ELSE PRI
NT L$
2590 IF L=3 THEN 2505 ELSE 2530
2900 REM :STOP
3000 REM Ausgeben
3010 CLS
3020 FOR I=0 TO N-1
3030 PRINT D$(I);TAB(20);E$(I)
3040 NEXT I
3050 PRINT"Ende"
3060 PRINT"Weiter mit beliebiger Taste"
3070 A$=INKEY$:IF A$="" THEN 3070 ELSE GOTO 120
4000 REM Laden/Speichern
4010 CLS
4020 KEY(1) OFF:KEY(2) OFF:
4030 LOCATE 10,2:PRINT "F1:Laden"
4040 LOCATE 10,3:PRINT "F2:Speichern"
4050 ON KEY GOSUB 4200,4500
4060 KEY(1) ON:KEY (2) ON
4070 GOTO 4070
4200 ' LADEN
```

```
4210 GOSUB 4800
4220 OPEN "CAS:VERB" FOR INPUT AS #1
4230 INPUT #1,N
4240 FOR I=0 TO N-1
4250  INPUT #1,E$(I),D$(I)
4260 NEXT I
4300 CLOSE #1:GOTO 120
4500 ' SPEICHERN
4510 GOSUB 4900
4520 OPEN "CAS:VERB" FOR OUTPUT AS #1
4530 PRINT #1,N
4540 FOR I=0 TO N-1
4550  PRINT #1,E$(I);",";D$(I)
4560 NEXT I
4600 CLOSE #1:GOTO 120
4700 LOCATE 3,4
4710 PRINT"Wenn fertig, beliebige Taste druecken
"
4720 A$=INKEY$:IF A$="" THEN 4720
4730 CLS:LOCATE 5,2
4740 PRINT"Laden/Speichern"
4780 RETURN
4800 CLS:LOCATE 3,2
4810 PRINT"Kassette einlegen und PLAY druecken"
4820 GOTO 4700
4900 CLS:LOCATE 3,2
4910 PRINT"Kassette einlegen und PLAY, REC druec
ken"
4920 GOTO 4700
5000 REM Ende
5010 CLS
5020 LOCATE 5,2
5030 PRINT"Sind Daten gespeichert (J/ )";:
5040 A$=INKEY$:IF A$="" THEN 5040
5050 IF A$="J" OR A$="j" THEN CLS:END ELSE GOTO
120
```

2.1 Vokabeln lernen

Programmbeschreibung:

D$(L)	Deutsche Vokabeln
E$(L)	Englische Vokabeln
L=100	Maximal 100 Worte werden gespeichert.
N	Zahl der gespeicherten Worte.

Zeile 120 bis 210	Hauptmenü
Zeile 1000 bis 1090	Eingeben der Worte.
Zeile 2000 bis 2590	Lernen
Zeile 2200 bis 2300	Deutsch-Englisch
Zeile 2500 bis 2590	Englisch-Deutsch
Zeile 2400 bis 2420	Erzeugen einer Zufallszahl
Zeile 3000 bis 3070	Bildschirmausgabe
Zeile 4000 bis 4929	Laden/Speichern
Zeile 4200 bis 4300	Laden von Kassette
Zeile 4500 bis 4600	Speichern auf Kassette
Zeile 5000 bis 5050	Programm beenden.

Für die Erzeugung der Zufallszahl R wird die Variable TIME als Ausgangszahl verwendet. Dadurch ist sichergestellt, daß nicht die gleiche Folge von Zufallszahlen erzeugt wird.

Beim Speichern der Worte auf Kassette wird zuerst die Anzahl N aufgezeichnet und dann die Werte von E$(I) und D$(I). Damit wird beim Einlesen zuerst N gelesen und in einer Schleife dann die Werte der Felder.

2.2 Terminkalender

Das Programm ist ein Demonstrationsprogramm zum Aufstellen, Sortieren und Suchen in Listen. Die Elemente dieses Programms sind Bestandteile vieler Programme.

Die Liste in diesem Programm ist das Feld A$(L). Mit L=50 können somit 50 Einträge mit max 255 Zeichen gespeichert werden. Die ersten 8 Zeichen sind das Schlüsselwort, nach welchen die Einträge sortiert werden.

Damit ist das Programm ziemlich universell einsetzbar. Als Beispiel ist ein Terminkalender angegeben. Dabei ist das Datum das Schlüsselwort.

Wird als Schlüsselwort eine Lagernummer gewählt, so kann das Programm ohne Änderung als Lagerverwaltung verwendet werden.

Das Programm wird mit den Funktionstasten gesteuert. Diese sind in den Zeilen 130 bis 230 programmiert und haben folgende Bedeutung:

F1: Eingeben
F2: Ausgeben
F3: Löschen
F4: Termine
F5: Laden/Speichern
F10: Ende

Die Eingabe erfolgt in den Zeilen 300 bis 370. Der neu eingegebene Begriff wird im Programmteil 910 bis 1080 in die vorhandene, schon alphabetisch sortierte Liste eingetragen.

Dazu wird der neue Begriff mit dem Eintrag in der Mitte der Liste verglichen. Je nach dem, ob er größer oder kleiner ist, wird in der oberen oder unteren Hälfte der Liste weitergesucht. Dabei wird diese Hälfte wiederum halbiert und der neue Eintrag wiederum mit diesem Eintrag verglichen. Dies wird solange fortgesetzt, bis nicht mehr halbiert werden kann. Das ist dann der Fall, wenn nur noch ein Eintrag vorhanden ist.

Dann wird durch Umspeichern in den Zeilen 1030 bis 1050 Platz für den neuen Eintrag geschaffen, der in Zeile 1060 an den gefundenen Platz geschrieben wird. Wird bei diesem Suchen ein gleicher Eintrag gefunden, so wird der alte Eintrag überschrieben.

In den Zeilen 240 bis 290 wird das Schlüsselwort eingegeben. In der Zeile 255 wird die Eingabe D$ mit Leerzeichen bis zur achten Stelle aufgefüllt.

Für das Suchen eines Eintrags in den Zeilen 1090 bis 1220 wird das gleiche Verfahren wie beim Einsortieren verwendet. Ebenso wird beim Löschen zuerst der Eintrag gesucht und danach durch seinen Nachfolger überschrieben.

```
100 SCREEN 0:WIDTH 40:MAXFILES=1
110 DIM A$(50):N=1:L$=SPACE$(10)
120 KEY OFF:CLS
130 ON KEY GOSUB 300,380,520,460,590,0,0,0,0,580
140 LOCATE 10,2:PRINT "Termine"
150 LOCATE 10,4:PRINT "F1:Eingeben"
160 LOCATE 10,5:PRINT "F2:Ausgeben"
170 LOCATE 10,6:PRINT "F3:Loeschen"
180 LOCATE 10,7:PRINT "F4:Termine"
190 LOCATE 10,8:PRINT "F5:Laden/Speichern"
200 LOCATE 10,9:PRINT "F10:Ende"
210 KEY(1) ON:KEY(2) ON:KEY(3) ON
220 KEY(4) ON:KEY(5) ON:KEY (10) ON
230 GOTO 230
240 LOCATE 5,5
250 INPUT"DATUM:TT.MM.JJ";D$
255 D$=LEFT$(D$+L$,8)
260 LOCATE 11,5
270 FOR I=1 TO 20:PRINT" ";:NEXT I
280 LOCATE 11,5:PRINT D$
290 RETURN
300 ' EINGEBEN
310 CLS:GOSUB 240
320 LOCATE 5,7:INPUT "Text ";E$
330 GOSUB 910
340 LOCATE 5,20:PRINT"Weitere Eintraege ( /N)?";
350 A$=INKEY$:IF A$="" THEN 350
360 IF ASC(A$)=13 THEN 300
370 IF A$="N" OR A$="n" THEN RETURN 120
380 ' AUSGABE
390 CLS:GOSUB 240
400 GOSUB 1090
410 IF B=1 THEN 120
420 LOCATE 5,7:PRINT "TEXT:"
430 L=LEN(A$(J))
440 PRINT RIGHT$(A$(J),L-8)
450 GOTO 1310
```

```
460 ' TERMINE
470 CLS
480 FOR J=1 TO N-1
490 PRINT LEFT$(A$(J),8)
500 NEXT J
510 GOTO 1310
520 ' LOESCHEN
530 CLS:GOSUB 240
540 GOSUB 1090
550 IF B=1 THEN 120
560 GOSUB 1230
570 RETURN 120
580 KEY ON:CLS:END
590 ' LADEN/SPEICHERN
600 CLS
610 KEY (1) OFF: KEY (2) OFF
620 LOCATE 10,4: PRINT"F1:Laden"
630 LOCATE 10,5: PRINT"F2:Speichern"
640 ON KEY GOSUB 770,840
650 KEY (1) ON:KEY (2) ON
660 GOTO 660
670 CLS:LOCATE 3,2
680 PRINT"Kassette einlegen und PLAY druecken"
690 GOTO 720
700 CLS:LOCATE 3,2
710 PRINT"Kassette einlegen und PLAY, REC drueck
en"
720 LOCATE 3,4
730 PRINT "Wenn fertig, beliebige Taste druecken
"
740 A$=INKEY$:IF A$="" THEN 740
750 CLS:LOCATE 5,2
760 PRINT "Laden/Speichern":RETURN
770 ' LADEN
780 GOSUB 670
790 OPEN "CAS:TERM" FOR INPUT AS #1
800 INPUT #1,N
810 FOR I=1 TO N-1
820  INPUT#1,A$(I)
830 NEXT I: CLOSE #1: GOTO 120
840 ' SPEICHERN
850 GOSUB 700
860 OPEN "CAS:TERM" FOR OUTPUT AS #1
```

```
870 PRINT #1,N
880 FOR I=1 TO N-1
890  PRINT #1,A$(I)
900 NEXT I: CLOSE #1: GOTO 120
910 ' EINSORTIEREN
920 J=1
930 IF N=1 THEN 1060
940 IF D$<LEFT$(A$(1),8) THEN 1020
950 J1=1:J2=N
960 J=INT((J1+J2)/2)
970 C$=LEFT$(A$(J),8)
980 IF C$=D$ THEN 1060
990 IF D$<C$ THEN J2=J ELSE J1=J
1000 IF J<> INT((J1+J2)/2) THEN 960
1010 J=J+1
1020 J1=N+1
1030 IF J1<J THEN 1060
1040 A$(J1)=A$(J1-1)
1050 J1=J1-1:GOTO 1030
1060 A$(J)=D$+E$
1070 N=N+1
1080 RETURN
1090 ' SUCHEN
1100 B=0:J=1
1110 IF N<>1 THEN 1140
1120 LOCATE 5,5:PRINT "Liste leer"
1130 GOTO 1310
1140 J1=1:J2=N
1150 J=INT((J1+J2)/2)
1160 C$=LEFT$(A$(J),8)
1170 IF C$=D$ THEN 1220
1180 IF D$<C$ THEN J2=J ELSE J1=J
1190 IF J<>INT((J1+J2)/2) THEN 1150
1200 LOCATE 5,5:PRINT "Kein Eintrag"
1210 GOTO 1310
1220 RETURN
1230 ' SUB LOESCHEN
1240 J1=J
1250 IF J1=N THEN 1300
1260 A$(J1)=A$(J1+1)
1270 J1=J1+1
1280 GOTO 1250
1290 N=N-1:IF N=0 THEN N=1
```

```
1300 RETURN
1310 LOCATE 5,20:B=1:PRINT" Weiter mit beliebige
r Taste"
1320 A$=INKEY$:IF A$="" THEN 1320
1330 GOTO 120
```

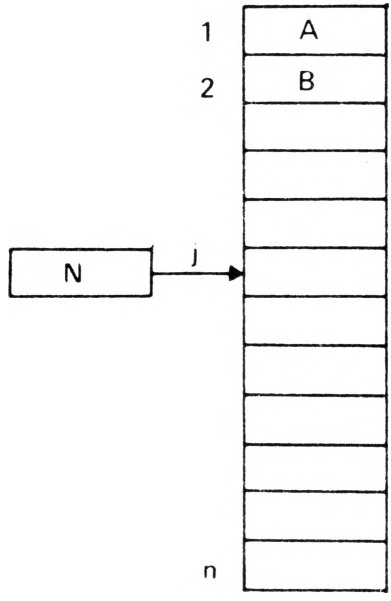
2.2 Programm Terminkalender

A$(L)	Feld der Einträge
LEFT$(A$(I),8)	Schlüsselwort
L=50	Platz für 50 Einträge
N	Zahl der Einträge

Zeile 130 bis 230	Hauptmenü
Zeile 240 bis 290	Eingabe des Schlüsselwortes
Zeile 300 bis 370	Eingabe
Zeile 400 bis 450	Ausgabe
Zeile 460 bis 510	Ausgabe der Schlüsselworte
Zeile 520 bis 570	Löschen
Zeile 580	Programm beenden
Zeile 590 bis 900	Laden/Speichern
Zeile 910 bis 1080	Einsortieren
Zeile 1090 bis 1220	Schlüsselwort suchen
Zeile 1230 bis 1300	Eintrag löschen

2.3 Verkettete Listen

Betrachtet man Adressverwaltungen, Lagerverwaltungen und ähnliche Programme zur Erfassung und Verwaltung von Daten, so stellt man meistens fest, daß die Datensätze selbständige und alleinstehende Einheiten sind, ohne Bezug auf den Vorgänger oder Nachfolger. Werden solche Datensätze nach einem Schlüsselwort (Key) sortiert, so müssen sehr große Datenmengen im Speicher bewegt werden. Betrachten wir dazu Abbildung 2.3.



2.3 Einfügen eines Datensatzes

Der Eintrag N soll an der Stelle j eingefügt werden.

Dazu müssen die Einträge j bis n um die Länge eines Datensatzes verschoben werden. Das sind n-j+1 Verschiebungen.

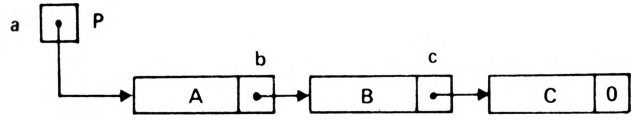
Ein Beispiel:

In einer Adressverwaltung ist ein Eintrag 128 Zeichen lang. Es sind bereits n = 400 Adressen gespeichert. Soll nun an der Stelle J = 150 ein Eintrag eingeschoben werden, so müssen (400 - 150 + 1) X 128 = 32128 Zeichen, das sind etwa 32K Byte, verschoben werden. Die Zeitdauer bei Sortiervorgängen ist also hauptsächlich durch die Zeit bestimmt, die für das Verschieben von Datensätzen benötigt wird.

Dieses Verschieben von Daten kann vermieden werden, wenn verkettete Listen verwendet werden.

Lineare Listen

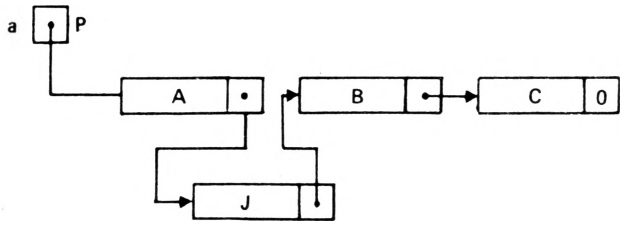
Bei einer verketteten Liste enthält ein Eintrag neben der Information (Adresse, Lagerbestand o.ä.) noch eine Angabe, wo der nächste Eintrag gespeichert ist. In Abbildung 2.4 ist eine verkettete Liste dargestellt. Jeder Eintrag enthält einen Zeiger (Pointer), welcher die Adresse des nächsten Eintrags enthält. Solche Datenstrukturen sind in PASCAL (Pointer Type) und auch in der Sprache C fest vereinbart.



2.4 Verkettete Liste

P in Abbildung 2.4 ist eine Zeigervariable. Sie enthält die Adresse des ersten Eintrags. Der letzte Eintrag enthält als Zeiger die Adresse Null. Dies ist die Angabe, daß kein weiterer Eintrag folgt. Enthält die Zeigervariable p den Wert 0, so ist die Liste leer.

Das Einfügen eines Elementes ist einfach (siehe Abbildung 2.5).



2.5 Einfügen eines Eintrags

Der Eintrag J soll zwischen A und B eingefügt werden. Der Eintrag wird irgendwo im Speicher abgelegt. In die Zeigervariable von A wird die Adresse des Eintrags J (j) geschrieben, während die Zeigervariable von J die Adresse von B (b) aufnimmt. Damit ist der Eintrag J logisch in die Kette eingefügt, obwohl er im Speicher irgendwo wahlfrei gespeichert ist. Das Programm in Abbildung 2.6 zeigt ein Beispiel.

In Zeile 100 wird ein Feld von Zeichenketten N$ und ein Zahlenfeld Z festgelegt, beide von der Länge N = 10. Das Feld N$ enthält die Daten (Namen), das Feld Z die Zeiger auf den nächsten Eintrag. Im Prinzip ist es gleichgültig, ob man die Daten sofort bei der Eingabe verkettet oder erst nach abgeschlossener Eingabe. Hier im Programm werden die Daten erst eingegeben (Subroutine 500 bis 530), dann verkettet (Unterprogramm 300 bis 390).

```
100 SCREEN 0:WIDTH 40
110 N=10
120 DIM N$(N),Z(N)
130 GOSUB 500
140 GOSUB 300
150 GOSUB 1000
```

```
160 END
300 ' VERKETTEN
310 P=0
320 IF P=0 THEN Z(1)=0:P=1
330 FOR J=2 TO N
340    I=P
350    IF N$(J)<N$(P) THEN Z(J)=P:P=J:GOTO 400
360    K=I:I=Z(I)
370    IF I=0 THEN Z(K)=J:Z(J)=0:GOTO 400
380    IF N$(J)<N$(I) THEN Z(K)=J:Z(J)=I:GOTO 400
390    GOTO 360
400 NEXT J:RETURN
500 ' EINGABE
510 FOR J=1 TO N
520 LINE INPUT N$(J)
530 NEXT J:RETURN
1000 CLS:FOR I=1 TO N
1010 LOCATE 0,I:PRINT N$(I)
1020 NEXT I
1030 I=P:J=1
1040 IF I=0 THEN RETURN
1050 LOCATE 20,J:PRINT N$(I):J=J+1
1060 I=Z(I):GOTO 1040
```

2.6 Lineare Liste

Die Zeigervariable P enthält zum Anfang Null als Zeichen für eine leere Liste. Die Verkettung beginnt mit P=1 und Z(1)=0. In Zeile 350 wird ein Eintrag vor dem ersten Eintrag, in Zeile 370 an das Ende und in Zeile 380 zwischen zwei Einträgen eingefügt. Die Variable k verweist auf den vorangehenden Eintrag.

Abbildung 2.7 zeigt die Ein- und Ausgabe einer Namensliste. Das Zeigerfeld zeigt Abbildung 2.8. Die Zeigervariable P enthält die Adresse 6, als das erste Element der Liste. Dies ist der Name ANNI. Dessen Zeiger enthält die Adresse 10, die Adresse des nächsten Namens BERND. Das letzte Element der Liste ZAUSEL, enthält als Zeiger Null.

```
HANS                ANNI
OTTO                BERND
UWE                 EKKE
KARL                GEORG
GEORG               HANNI
ANNI                HANS
HANNI               KARL
ZAUSEL              OTTO
EKKE                UWE
BERND               ZAUSEL
```

2.7 Beispiel

```
P= 6

Z( 1 )= 4
Z( 2 )= 3
Z( 3 )= 8
Z( 4 )= 2
Z( 5 )= 7
Z( 6 )= 10
Z( 7 )= 1
Z( 8 )= 0
Z( 9 )= 5
Z( 10 )= 9
```

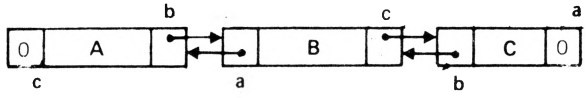
2.8 Zeigerliste

Dem Leser sei es überlassen, ein Unterprogramm zu schreiben, das einen Eintrag in eine schon bereits verkettete Liste einfügt.

Das Löschen eines Eintrags ist ebenfalls recht einfach. Der Zeiger von A(b) wird durch (c) ersetzt. Damit ist der Eintrag B aus der Liste entfernt.

2.4 Telefon-Verzeichnis

Das Programm in Abbildung 2.10, Telefon-Verzeichnis verwendet eine doppelt verkettete Liste. Diese ist in Abbildung 2.9 gezeigt. Bei einer solchen Liste werden zwei Zeiger R und L verwendet. Der Zeiger R zeigt auf den Nachfolger, der Zeiger L auf den Vorgänger eines Eintrags.



2.9 Doppelt verkettete Liste

Die Zeiger an den beiden Enden der Liste sind Null.

Bei dem Programm Telefon-Verzeichnis wird der Name als Schlüsselwort verwendet. Durch die doppelte Verkettung kann vorwärts und rückwärts durch die Liste geblättert werden.

Die Verkettung eines Eintrags sei an den fünf möglichen Fällen nochmals gezeigt.

1. Die Liste ist leer. Dann ist P=0
2. Ein Eintrag. Dann ist P=1, R(1)=0 und L(1)=0.
3. Der neue Eintrag J ist größer, als alle anderen. Dann ist R(J)=0, L(J)=K, R(K)=J. K war der bisher größte Eintrag.
4. Der neue Eintrag J ist kleiner als alle anderen. Dann ist R(J)=P, L(J)=0, L(P)=J und P=J.
5. Der neue Eintrag J muß zwischen die Einträge K und I eigefügt werden. Dann ist R(K)=J, L(J)=K, R(J)=I und L(I)=J.

Diese Verkettung ist im Programm in den Zeilen 300 bis 400 programmiert. Das Programm wird durch die Funktions-

tasten gesteuert und hat folgende Eingaben:

F1:Eingeben
F2:Suchen
F3:Laden/Speichern
F4:Liste
F5:Beenden

Beim Eingeben wird zuerst der Name, dann die Nummer eingegeben. Auf dem Bildschirm erscheint die Maske

Name:_____

und danach

Nummer:_____

Es werden jeweils 20 Stellen berücksichtigt. Da mit LINE INPUT auch die Unterstreichungen übernommen werden, so werden diese im Unterprogramm 1200 bis 1230 durch Leerzeichen ersetzt. Nach der Eingabe der Nummer erscheint am unteren Bildrand das Zeichen -->.

Das bedeutet, daß mit der Taste "Cursor Rechts" ein weiterer Eintrag gemacht werden kann. Alle anderen Tasten führen zum Menü zurück.

Nach der Eingabe kann mit F2 nach einem Eintrag gesucht werden. Ist die Variable C gleich Null, so müssen die Einträge noch verkettet werden. Danach kann der Name eingegeben werden. Wird er gefunden, so wird der Name und die Nummer auf den Bildschirm ausgegeben. Am unteren Bildrand erscheinen die Zeichen <-- -->. Mit den Cursortasten "Rechts" und "Links" kann durch die Liste geblättert werden. Wiederum führt eine beliebige Taste ins Menü zurück.

Mit F3 werden die Einträge auf Kassette gespeichert.
Eine sortierte Liste der Einträge wird mit F4 auf den Bildschirm ausgegeben und mit F5 wird das Programm beendet.

```
100 SCREEN 0:WIDTH 40:MAXFILES=1
105 CLEAR 4000
110 N=100:C=0:Z=0
120 DIM N$(N),E$(N),R(N),L(N)
125 L1$=STRING$(20,95)
130 DEFINT I-K
140 L$=SPACE$(20):KEY OFF
150 CLS
160 ON KEY GOSUB 1000,2000,3000,4000,5000
170 LOCATE 10,2:PRINT"F1:Eingeben"
180 LOCATE 10,3:PRINT"F2:Suchen"
190 LOCATE 10,4:PRINT"F3:Laden/Speichern"
200 LOCATE 10,5:PRINT"F4:Liste"
205 LOCATE 10,6:PRINT"F5:Beenden"
210 KEY (1) ON:KEY (2) ON:KEY (3) ON
220 KEY (4) ON:KEY (5) ON
230 GOTO 230
300 ' VERKETTEN
305 LOCATE 10,2:PRINT "Verkettung"
310 P=0
320 IF P=0 THEN R(1)=0:L(1)=0:P=1
330 FOR J=2 TO Z
340    I=P
350    IF N$(J)<N$(P) THEN R(J)=P:L(J)=0:L(P)=J:P
=J:GOTO 400
360    K=I:I=R(I)
370    IF I=0 THEN R(K)=J:R(J)=0:L(J)=K:GOTO 400
380    IF N$(J)<N$(I) THEN R(K)=J:R(J)=I:L(J)=K:L
(I)=J:GOTO 400
390    GOTO 360
400 NEXT J:C=1:RETURN
1000 ' EINGABE
1010 CLS:Z=Z+1
1015 LOCATE 10,0:PRINT"Eingabe"
1020 LOCATE 5,2:PRINT "Name:"+L1$
1030 LOCATE 10,2
1040 LINE INPUT N$(Z)
1045 A$=N$(Z):GOSUB 1200:N$(Z)=A$
1050 LOCATE 3,4:PRINT "Nummer:"+L1$
1055 LOCATE 10,4
1060 LINE INPUT E$(Z)
1065 A$=E$(Z):GOSUB 1200:E$(Z)=A$
1070 C=0
```

```
1075 LOCATE 30,20:PRINT"-->";
1080 A$=INPUT$(1)
1090 IF ASC(A$)=28 THEN 1010 ELSE 150
1200 FOR I=1 TO 20
1210 IF MID$(A$,I,1)="_" THEN MID$(A$,I,1)=" "
1220 NEXT I
1230 RETURN
2000 ' SUCHEN
2010 CLS
2020 IF C=0 THEN GOSUB 300
2025 LOCATE 10,0:PRINT"Suchen"
2030 LOCATE 5,2:PRINT "Name:"+L1$
2040 LOCATE 10,2
2050 LINE INPUT A$
2060 GOSUB 1200
2070 I=P
2080 IF I=0 THEN 2200
2090 IF N$(I)=A$ THEN 2100 ELSE I=R(I):GOTO 2080
2100 CLS
2110 LOCATE 10,2:PRINT N$(I)
2120 LOCATE 10,4:PRINT E$(I)
2130 LOCATE 26,20:PRINT "<-- -->";
2140 A$=INPUT$(1)
2150 IF ASC(A$)=29 THEN K=I:I=L(I)
2160 IF ASC(A$)=28 THEN K=I:I=R(I)
2170 IF ASC(A$)<>29 AND ASC(A$)<>28 THEN 150
2180 IF I=0 THEN LOCATE 10,6:PRINT" Ende der Lis
te";:I=K:GOTO 2140
2190 GOTO 2100
2200 CLS:LOCATE 5,20
2210 PRINT"Weiter mit beliebiger Taste"
2220 A$=INPUT$(1):GOTO 150
3000 ' LADEN/SPEICHERN
3010 CLS:KEY (1) OFF:KEY (2) OFF
3020 LOCATE 10,2:PRINT"F1:Laden"
3030 LOCATE 10,3:PRINT"F2:Speichern"
3040 ON KEY GOSUB 3100,3300
3045 KEY (1) ON: KEY (2) ON
3050 GOTO 3050
3100 ' LADEN
3110 GOSUB 3500
3120 OPEN "CAS:LISTE" FOR INPUT AS #1
3130 INPUT #1,Z
```

```
3140 FOR I=1 TO Z
3150 INPUT #1,N$(I),E$(I)
3160 NEXT I:CLOSE #1
3170 GOSUB 300:GOTO 150
3300 ' SPEICHERN
3310 GOSUB 3530
3320 OPEN "CAS:LISTE" FOR OUTPUT AS #1
3330 PRINT #1,Z
3340 FOR I=1 TO Z
3350 PRINT #1,N$(I);",";E$(I)
3360 NEXT I:CLOSE #1:GOTO 150
3500 CLS:LOCATE 3,2:PRINT"Kassette einlegen, PLA
Y druecken"
3510 GOTO 3542
3530 CLS:LOCATE 3,2
3540 PRINT"Kassette einlegen, PLAY REC druecken"
3542 LOCATE 3,4
3544 PRINT"Wenn fertig, beliebige Taste druecken
"
3546 A$=INPUT$(1)
3550 CLS:LOCATE 5,2
3560 PRINT"Laden/Speichern"
3570 RETURN
4000 ' LISTE
4010 CLS:IF C=0 THEN GOSUB 300
4020 I=P:IF I=0 THEN PRINT"Liste leer":GOTO 2210
4030 PRINT N$(I);TAB(19);E$(I)
4040 I=R(I):IF I=0 THEN PRINT"Ende der Liste":GO
TO 2210
4050 GOTO 4030
5000 ' BEENDEN
5010 CLS:LOCATE 3,2
5020 PRINT" Ist Liste gespeichert (J/ )";
5030 A$=INPUT$(1)
5040 IF A$="J" OR A$="j" THEN CLS:END
5050 GOTO 150
```

2.10 Programm Telefonverzeichnis

Programmbeschreibung:

N$(N)	Feld der Schlüsselworte
E$(N)	Feld der Einträge
N=100	Platz für 100 Einträge
R(N)	Zeiger für Nachfolger
L(N)	Zeiger für Vorgänger
P	Zeiger auf erstes Element
Z	Zahl der Einträge
I-K	Ganzzahlen
C	Merker für Verketten
Zeile 105	Platz für Zeichenketten
Zeile 150 bis 230	Hauptmenü
Zeile 300 bis 400	Verketten
Zeile 1000 bis 1090	Eingabe
Zeile 1200 bis 1230	"_" durch " " ersetzen.
Zeile 2000 bis 2090	Suchen
Zeile 2100 bis 2220	Durchblättern
Zeile 3000 bis 3570	Laden/Speichern
Zeile 4000 bis 4050	Ausgabe der Liste
Zeile 5000 bis 5050	Programm beenden.

2.5 Rechnungen schreiben

Mit dem Programm in Abbildung 2.12 können, wenn an den Rechner ein Drucker angeschlossen ist, Rechnungen geschrieben werden. Das Programm ist völlig linear aufgebaut.

Nach dem Starten des Programms wird das Datum und die Nummer der nächsten Rechnung eingegeben. Ebenso kann eine Buchungsnummer angegeben werden. Nach diesen Eingaben erfolgt die Aufforderung, den Drucker einzuschalten.

Für jede Rechnung können der Rabatt und die Versandkosten entweder von Hand eingegeben oder vom Rechner errechnet werden. Der Rabatt wird vom Rechner in den Zeilen 760 bis 790, die Versandkosten in den Zeilen 800 bis 840 be-

rechnet.

Nach der Eingabe der Bestellnummer des Kunden wird die Anschrift eingetragen. Danach erfolgt die Eingabe des Mehrwertsteuersatzes, die Initialen des Verkäufers, die Zahlungsweise und die Versandart.

Falls die Versandanschrift von der Rechnungsanschrift verschieden ist, kann diese nach der Anforderung eingegeben werden. An dieser Stelle druckt das Programm den Kopf der Rechnung aus und wartet auf die Eingabe der Rechnungsposten.

Für diese Eingabe gibt es zwei Möglichkeiten. Wird auf die Anfrage

Eingabe Manuell, Data, Ende (/D/E)

eine beliebige Taste gedrückt, so wird der Rechnungsposten mit Artikelnummer, Bezeichnung und Verkaufspreis, sowie bestellte und gelieferte Anzahl eingegeben.

Nach diesen Angaben wird die Zeile ausgedruckt. Häufig vorkommende Artikel können in DATA-Anweisungen am Ende des Programms angegeben werden und zwar in der Form

Nummer,Bezeichnung,Nettopreis

Diese DATA Anweisungen müssen mit

DATA 0,0,0

abgeschlossen sein, damit das Programm das Ende der Anweisungen erkennt. Wird bei der obigen Anfrage D eingegeben, so wird nach Eingabe der Artikelnummer diese in den DATA-Anweisungen gesucht und der gefundene Artikel auf dem Bildschirm angezeigt. Ist es der richtige Artikel, so wird nach Eingabe der bestellten und der gelieferten Menge die Zeile ausgegeben.

Mit der Eingabe von E wird das Schreiben der Rechnung beendet. Nach Abzug des Rabattes wird der Gesamtnettobetrag, der Anteil der Mehrwertsteuer (mit Versandkosten)

und der Endwert der Rechnung errechnet und ausgedruckt.

Gleichzeitig wird der Endwert auf den Bildschirm ausgegeben. Das Programm geht zur Eingabe der Buchungsnummer zurück und wartet auf die Eingabe einer weiteren Rechnung. Wird für die Buchungsnummer Null eingegeben, so wird das Programm beendet. Ein Beispiel für eine Rechnung zeigt die Abbildung 2.11

```
Hotacker-Verlag
Tegernseerstr.18                                        14.2.85
8150 Holzkirchen

  4                                                        1000

   Hans Meier                          Georg Meier
   Lindenstr.8                         Karlstr.8
   8150 Holzkirchen                    8000 Muenchen

          1234      ef      30 Tage Netto  Spetiteur          14.2.85

5       5 100           Hosentraeger              4.78          23.90
                              33  % Rabatt =                -    7.89

                  Vielen Dank fuer Ihren Auftrag

16.01     2.24     2.50                                        20.75
```

2.11 Beispiel Rechnung

```
100 REM ==== Rechnung Schreiben ====
110 SCREEN 0:WIDTH 40:CLS
120 BL$="                "
130 PRINT" *** Rechnung Schreiben *** "
140 PRINT
```

```
150 INPUT"Rechnungsdatum : ";RD$
160 INPUT"Erste Rechnungsnummer : ";I1
170 INPUT"Buchungsnummer : ";C1
180 IF C1 =0 THEN END
190 PRINT:PRINT"Drucker einschalten !! ":PRINT
200 AA$=INPUT$(1)
210 SC=1:DC=1:D=0:V=0
220 PRINT"Rabattrechnung Automatisch (J/N)";
222 AA$=INPUT$(1): PRINT AA$
225 IF AA$="N" THEN DC=0
230 IF AA$<>"N" AND AA$<>"J" THEN 220
240 IF DC = 0 THEN INPUT "Wieviel % ";D:D=D/100
250 PRINT"Versandkosten-Berechnung"
255 PRINT"Automatisch (J/N)";
257 AA$=INPUT$(1):PRINT AA$
260 IF AA$="N" THEN SC=0
265 IF AA$<>"N" AND AA$<>"J" THEN 250
270 IF SC= 0 THEN INPUT"Versandkosten ";V
280 INPUT"Kundenauftragsnummer : ";CD$
290 PRINT
300 PRINT"Name des Kunden : "
305 INPUT S1$
310 PRINT"Strasse und Hausnummer :"
315 INPUT S2$
320 PRINT"Postleitzahl,Stadt : "
325 INPUT S3$
330 PRINT:INPUT"Mehrwertsteuer (%) ";TR:TR=TR/10
0
340 PRINT
350 INPUT"Initialen des Verkaeufers ";SM$:PRINT
360 PRINT"Zahlungsweise : "
370 PRINT"1 = Netto innerhalb 30 Tagen"
372 PRINT"2 = Nachnahme"
374 INPUT"3 = Vorauszahlung";TE
380 IF TE=1 THEN TE$ = "30 Tage Netto"
390 IF TE=2 THEN TE$ = " Nachnahme "
400 IF TE=3 THEN TE$ = " Vorauszahlung"
410 IF TE <1 OR TE> 3 THEN 370
420 PRINT:PRINT"Zustellung per"
430 PRINT"1 = Post"
432 PRINT"2 = Paketservice"
434 INPUT"3 = Spediteur";SV
440 IF SV=1 THEN SV$=" Post"
```

```
450 IF SV=2 THEN SV$="Paket Service"
460 IF SV=3 THEN SV$="Spediteur"
470 IF SV<1 OR SV>3 THEN 430
480 CLS
490 PRINT S1$:PRINT S2$:PRINT S3$
500 PRINT:PRINT"Lieferung an dieselbe Adresse ?
(J/N)":Q$=INPUT$(1)
510 IF Q$="J" OR Q$="j" THEN 570
520 IF Q$<>"N" AND Q$<>"n" THEN 500
530 PRINT
535 PRINT"Versandadresse":PRINT"Name"
538 INPUT H1$
540 PRINT"Strasse und Hausnummer:"
542 INPUT H2$
544 PRINT"Postleitzahl und Stadt:"
546 INPUT H3$
550 GOTO 580
560 REM === Eigene Anschrift ===
570 H1$=S1$:H2$=S2$:H3$=S3$
580 FOR Q=1 TO 9:LPRINT:NEXT Q
590 LPRINT SPC(8);"Hofacker-Verlag"
600 LPRINT SPC(8);"Tegernseerstr.18  ";SPC(36);R
D$
610 LPRINT SPC(8);"8150 Holzkirchen "
620 LPRINT:LPRINT
630 LPRINT SPC(10);C1;SPC(53);I1
640 LPRINT:LPRINT
650 LPRINT SPC(10);S1$;SPC(35-LEN(S1$));H1$
660 LPRINT SPC(10);S2$;SPC(35-LEN(S2$));H2$
670 LPRINT SPC(10);S3$;SPC(35-LEN(S3$));H3$
680 LPRINT:LPRINT:LPRINT
690 LPRINT:LPRINT
700 LPRINT SPC(17);CO$;SPC(9-LEN(CO$));SM$;SPC(6
);TE$;SPC(15-LEN(TE$));SV$;SPC(19-LEN(SV$));RD$
710 LPRINT:LPRINT:LPRINT
720 GOSUB 1230
730 GOSUB 880
740 GOTO 720
750 S1=I
760 IF DC=0 THEN 800
770 IF (S1>0) AND (S1<6) THEN D=.25
780 IF (S1>5) AND (S1<11) THEN D=.33
790 IF S1>10 THEN D=.4
```

```
800 IF S1=0 THEN 850
810 IF S1<15 THEN V=2.5
820 IF (S1>14) AND (S1<30) THEN V=3
830 IF (S1>29) AND (S1<50) THEN V=5
840 IF S1>49 THEN V=10
850 I1=I1+1
860 R=INT(D*100+.5)
870 GOTO 1010
880 C=S1*D
890 LPRINT SPC(2);
900 LPRINT USING"#######";Z1;
910 LPRINT " ";
920 LPRINT USING"#######";S1;
930 LPRINT SPC(1); : LPRINT USING "\            \"
;IN$;
940 LPRINT SPC(2); : LPRINT USING "\            \"
;E$;
950 LPRINT SPC(11);
960 LPRINT USING"######.##";D;
970 LPRINT SPC(9);
980 LPRINT USING"######.##";D*S1
990 C=S1*D:T=T+C
1000 RETURN
1010 IF D=0 THEN ZS=ZS-2:D1+0:GOTO 1070
1020 D1=D*T
1030 D1=INT(D1*100+.5)/100
1040 LPRINT
1050 LPRINT SPC(35);R;" % Rabatt =    ";SPC(16);"
-";
1060 LPRINT USING"######.##";D1
1070 FOR P=1 TO (18-ZS):LPRINT:NEXT P
1080 LPRINT SPC(23);"Vielen Dank fuer Ihren Auft
rag"
1090 LPRINT:LPRINT:LPRINT
1100 M1=(T-D1+V)*TR:M2=T-D1+M1+V
1110 LPRINT SPC(4)
1120 LPRINT USING"######.##";T-D1;
1130 LPRINT SPC(1)
1140 LPRINT USING"####.##";M1;
1150 LPRINT SPC(2);
1160 LPRINT USING"###.##";V;
1170 LPRINT SPC(42);
1180 LPRINT USING"######.##";M2
```

```
1190 PRINT:PRINT"              Gesamtbetrag : ";
1200 PRINT USING"######.##";M2
1210 LPRINT:LPRINT:LPRINT
1220 I=0:T=0:PRINT"-------- Naechste Rechnung ---
-----";PRINT:GOTO 170
1230 DD$="":INPUT "Eingabe Manuel, Data , Ende (
 /D/E) ";DD$
1240 IF DD$ = "E" OR DD$ ="e" THEN 750
1250 IF DD$<>"D" AND DD$<>"d" THEN 1350
1260 INPUT"Artikelnummer : ";IN$
1270 RESTORE
1280 READ N$,E$,O
1290 IF N$="0" THEN PRINT"Artikelnummer nicht ge
funden":GOTO 1230
1300 IF N$=IN$ THEN 1320
1310 GOTO 1280
1320 PRINT "** ";IN$,E$,O;" **"
1330 ZZ$="":INPUT"Richtig ( /N)";ZZ$
1340 IF ZZ$="N" OR ZZ$="n" THEN 1230 ELSE 1390
1350 PRINT:INPUT"Artikelnummer : ";IN$
1370 INPUT"Beschreibung  : ";E$
1380 INPUT"Verkaufspreis : ";O
1390 PRINT
1400 INPUT"Bestellmenge ";Z1
1410 INPUT"Liefermenge  ";S1
1420 I=I+S1:ZS=ZS+1
1430 RETURN
1440 DATA 100,Hosentraeger,4.78
1450 DATA 101,Socken,1.89
1460 DATA 102,Stiefel,59.80
1470 DATA 0,0,0
```

2.12 Programm: Rechnung schreiben

Programmbeschreibung:

Die wichtigsten Variablen:

RD$	Rechnungsdatum
I1	nächste Rechnungsnummer
C1	Buchungsnummer
D	Rabatt
V	Versandkosten
CO$	Auftragsnummer Kunde
TR	Mehrwertsteuer
S(I)$	Rechnungsanschrift
H(I)$	Versandanschrift
O	Nettopreis
S1	Stückzahl
T	Netto-Gesamtpreis

Zeile 130 bis 180	Eingabe Datum, Nummer
Zeile 200 bis 240	Rabatt?
Zeile 250 bis 270	Versandkosten?
Zeile 280 bis 325	Kundenanschrift
Zeile 360 bis 410	Zahlungsbedingung
Zeile 420 bis 470	Versandart
Zeile 500 bis 550	Lieferanschrift
Zeile 590 bis 610	Eigene Anschrift
Zeile 570 bis 710	Rechnungskopf drucken
Zeile 720 bis 740	Ausgabe eines Postens
Zeile 760 bis 790	Rabattbestimmung
Zeile 800 bis 840	Versandkosten
Zeile 880 bis 1000	Berechnung und Ausgabe einer Zeile
Zeile 1010 bis 1210	Berechnung und Ausgabe des Gesamtbetrages
Zeile 1260 bis 1310	Suchen eines Artikels
Zeile 1350 bis 1390	Eingabe eines Artikels
Zeile 1400 bis 1430	Eingabe der Menge
Zeile 1440 bis	Gespeicherte Artikel

2.6 Laufschrift

Die beiden Programme in den Abbildungen 2.13 und 2.14 lassen einen Schriftzug über den Bildschirm laufen.

```
100 ' LAUFSCHRIFT
110 CLS
120 A$="Ich bin der MSX Computer von PHILIPS "
130 L=LEN(A$)
140 FOR K=1 TO L-1
150  LOCATE 36-K,10:PRINT LEFT$(A$,K)
160  GOSUB 500
170 NEXT K
300 FOR K=1 TO L
310  LOCATE 0,10:PRINT RIGHT$(A$,L-K)
320  GOSUB 500
330 NEXT K
499 END
500 FOR J=1 TO 60:NEXT J:RETURN
```

2.13 Laufschrift 1

```
100 ' LAUFSCHRIFT
110 CLS
120 A$="MSX Computer von PHILIPS "
130 L=LEN(A$)
140 FOR K=1 TO L-1
150  LOCATE 36-K,10:PRINT LEFT$(A$,K)
160  GOSUB 500
170 NEXT K
200 FOR K=36-L TO 1 STEP -1
210  LOCATE K,10:PRINT A$
220  GOSUB 500
230 NEXT K
300 FOR K=1 TO L
310  LOCATE 1,10:PRINT RIGHT$(A$,L-K)
320  GOSUB 500
330 NEXT K
340 GOTO 140
499 END
500 FOR J=1 TO 60:NEXT J:RETURN
```

2.14 Laufschrift 2

Notizen

3 Grafische Spielereien

Die folgenden grafischen Spielereien lassen sich alle auf die Form

$$x = f(t) \cdot \cos(w_1 t + \varphi_1)$$
$$y = g(t) \cdot \cos(w_2 t + \varphi_2)$$

zurückführen. Hierbei können die Funktionen f(t) und g(t) ebenfalls periodisch oder aber konstant sein.

Die Programme unterscheiden sich nur unwesentlich bei den Eingabezeilen und bei den Funktionen.

3.1 Polarplot

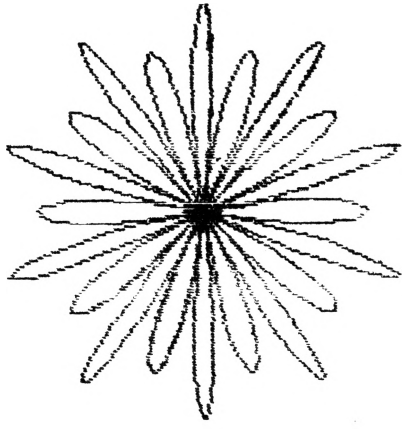
Das Programm in Abbildung 3.1 zeichnet die Funktionen

$$x = r(t) * \sin t$$
$$y = r(t) * \cos t \quad \text{mit} \quad r(t) = k1 * \sin k2 * t$$

Das Ergebnis mit K1=2.5 und K2=5 zeigt Abbildung 3.2.

```
100 PI=3.14159
110 INPUT"K1=";K1
120 INPUT"K2=";K2
125 XO=125:YO=85
140 SCREEN 2:COLOR 1,11,6
150 A=.1:S=50:W=0
160 FOR I=0 TO 45 STEP A
170   T=I/(2*PI)
175   R=COS(K1*SIN(K2*T))
180   X= R*SIN(T)
190   Y= R*COS(T)
195   Y=INT(Y*S)+YO:X=INT(X*S)+XO
200   PSET(X,Y),6
210 NEXT I
220 GOTO 220
```

3.1 Programm Polarplot



3.2 Polarplot

Die Abbildung 3.3 zeigt nochmals das Programm Polarplot. Hier werden jedoch keine Punkte, sondern Linien gezeichnet.

```
100 PI=3.14159
110 INPUT"K1=";K1
120 INPUT"K2=";K2
125 X0=125:Y0=85
140 SCREEN 2:COLOR 1,11,6
150 A=.1:S=50:W=0
160 FOR I=0 TO 45 STEP A
170   T=I/(2*PI)
175   R=COS(K1*SIN(K2*T))
180   X= R*SIN(T)
190   Y= R*COS(T)
195   Y=INT(Y*S)+Y0:X=INT(X*S)+X0
200 IF I<>0 THEN LINE (XA,YA)-(X,Y),6
205 XA=X:YA=Y
210 NEXT I
220 GOTO 220
```

3.3 Polarplot mit Linien

3.2 Reguläre Spirale

Eine reguläre Spirale hat die Form

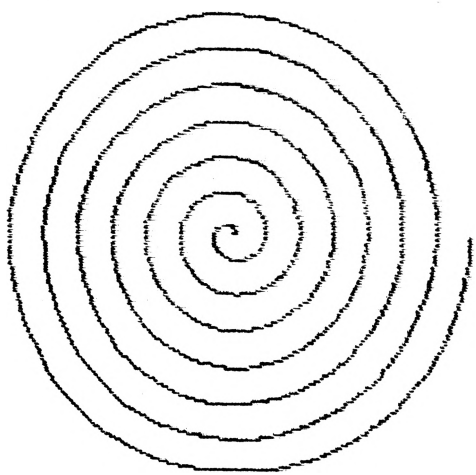
$$x = k1 * \varphi * \cos a \cdot \varphi$$
$$y = k2 * \varphi * \sin b \cdot \varphi$$

Das Programm in Abbildung 3.4 zeichnet die Spirale in Abbildung 3.5 mit K1=2, A=1, K2=2 und B=1. (Ohne Zeile 260)

```
100 PI=3.14159
110 R=7:RD=2*PI*R
120 S=PI/32
130 INPUT"K1=";K1
140 INPUT  "A=";A
150 INPUT"K2=";K2
```

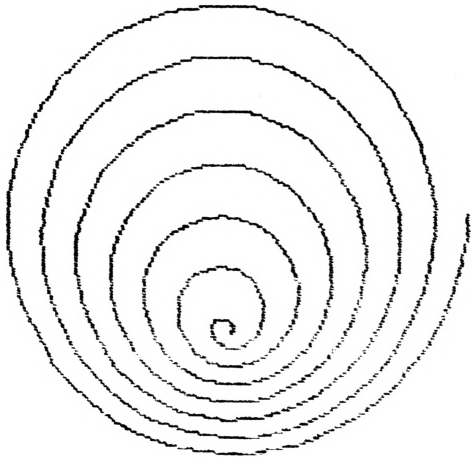
```
160 INPUT "B=";B
170 X0=125:Y0=50:Z=.1
200 SCREEN 2:COLOR 1,11,6
210 PSET(X0,Y0),6
220 XA=X0:YA=Y0
230 FOR X=0 TO RD STEP S
240   XP=X0+K1*X*COS(A*X)
250   YP=Y0+K2*X*SIN(B*X)
260   Z=Z+.1:YP=YP+Z
270   LINE (XA,YA)-(XP,YP),6
280   XA=XP:YA=YP
290 NEXT X
300 FOR I=0 TO 50
310   BEEP
320 NEXT I
330 GOTO 330
```

3.4 Programm Reguläre Spirale

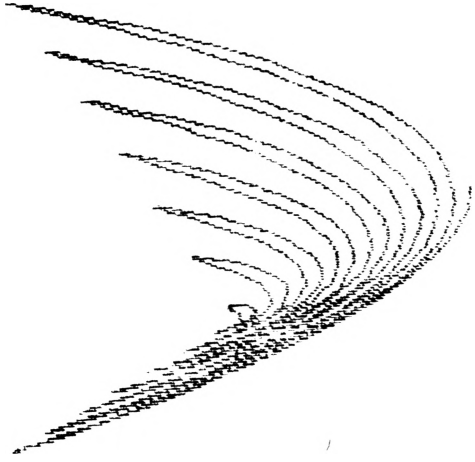


3.5 Reguläre Spirale

Überlagert man die Y-Richtung mit einer Verschiebung Z, so entsteht ein räumlicher Eindruck, wie in Abbildung 3.6.



3.6 Reguläre Spirale mit Verschiebung



3.7 Verzerrte Spirale

Die Abbildung 3.7 entstand mit den Eingaben K1=2, K2=2, A=2 und B=1. (mit Zeile 260)

3.3 Exponentielle Spirale

Die exponentielle Spirale hat die Form

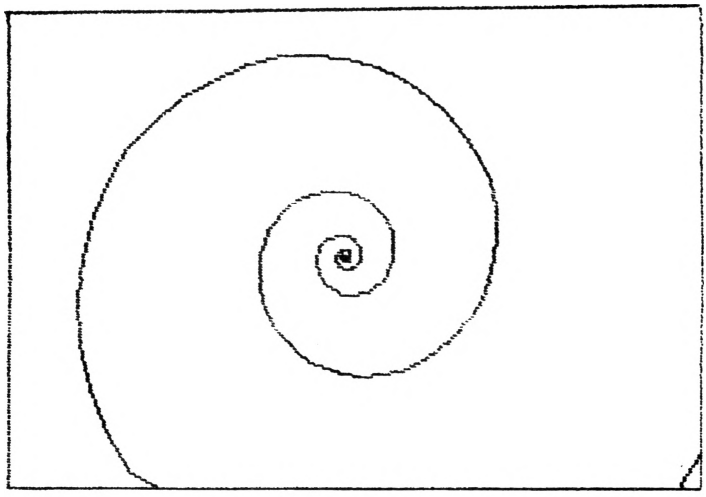
$$x = c \cdot e^{k1 \cdot \varphi} \cdot \cos a \cdot \varphi$$
$$y = c \cdot e^{k2 \cdot \varphi} \cdot \sin b \cdot \varphi$$

Im Programm in Abbildung 3.4 wurden die Zeilen 120, 130, 240 und 250 in

```
110 R=2:RD=2*PI*R
120 S=PI/100
240    XP=XO+.02*EXP(K1*X)*COS(A*X)
250    YP=YO+.02*EXP(K2*X)*SIN(B*X)
```

geändert. Die Zeile 260 wird herausgelassen.

Die Abbildung 3.8 entstand mit den Eingaben K1=0.9, K2=0.9, A=5 und B=5. Beim Zeichnen dieser Abbildung kann man das exponentielle Verhalten gut beobachten, denn für längere Zeit ist nur ein Punkt zu sehen, ehe die Spirale anwächst.



3.8 Exponentielle Spirale

3.4 Logarithmische Spirale

Die logarithmische Spirale hat die Form

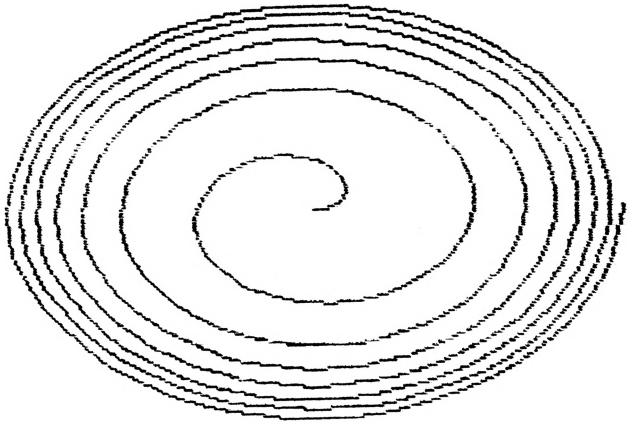
$$x = k1 * \ell n\,(1 + \varphi) * \cos a \cdot \varphi$$
$$y = k2 * \ell n\,(1 + \varphi) * \sin b \cdot \varphi$$

Folgende Zeilen wurden geändert:

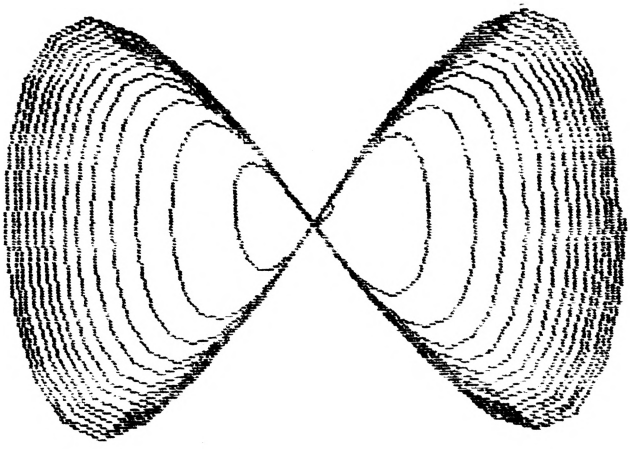
```
110 R=7:RD=2*PI*R
120 S=PI/32
240    XP=X0+K1*LOG(1+X)*COS(A*X)
250    YP=Y0+K2*LOG(1+X)*SIN(B*X)
```

Zwei Beispiele zeigen die Abbildungen 3.9 und 3.10. Diese wurden mit den Eingaben K1=30, K2=20, A=1 und B=1, sowie mit K1=30, K2=20, A=2 und B=4 gezeichnet.

Eingabedaten:
k1 = 30, a = 1,
k2 = 20, b = 1



3.9 Logarithmische Spirale 1



Eingabedaten:
k1 = 30, a = 2,
k2 = 20, b = 4

3.10 Logarithmische Spirale 2

3.5 Kardioide

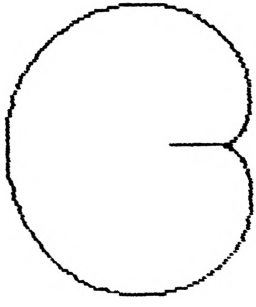
Die Gleichung einer Kardioide lautet:

$$x = k1\,(a \cdot \cos\varphi - a \cdot \cos 2\varphi)$$
$$y = k2\,(b \cdot \sin\varphi - b \cdot \sin 2\varphi)$$

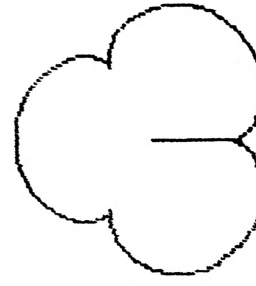
Folgende Zeilen wurden geändert:

```
110 R=2:RD=2*PI*R
120 S=PI/100
240    XP=XO+K1*(A*COS(X)-COS(A*X))
250    YP=YO+K2*(B*SIN(X)-SIN(B*X))
```

Die Abbildungen 3.11 und 3.12 entstanden mit den Eingaben K1=20, K2=20, A=2 und B=2, sowie mit K1=10, K2=10, A=4 und B=4.



3.11 Kardioide 1



3.12 Kardioide 2

3.6 Zykloide

Die Gleichung einer Zykloide lautet:

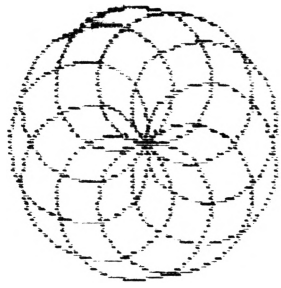
$$x = k1 \cdot \cos \varphi - k2 \cdot \cos k3 \cdot \varphi$$
$$y = k4 \cdot \sin \varphi - k5 \cdot \cos k6 \cdot \varphi$$

Physikalisch ist die Zykloide die Abrollkurve eines Punktes auf einem Kreis. Das Programm zeigt Abbildung 3.13.

```
100 PI=3.14159
110 RD=2*PI
120 S=PI/100
130 INPUT"K1=";K1
140 INPUT"K2=";K2
150 INPUT"K3=";K3
160 INPUT"K4=";K4
162 INPUT"K5=";K5
164 INPUT"K6=";K6
170 X0=125:Y0=75
200 SCREEN 2:COLOR 1,11,6
210 PSET(X0,Y0),6
220 XA=X0:YA=Y0
230 FOR X=0 TO RD STEP S
240   XP=X0+K1*COS(X)-K2*COS(K3*X)
250   YP=Y0+K4*SIN(X)-K5*SIN(K6*X)
270   LINE (XA,YA)-(XP,YP),6
280   XA=XP:YA=YP
290 NEXT X
300 FOR I=0 TO 50
310   BEEP
320 NEXT I
330 GOTO 330
```

3.13 Programm Zykloide

Zwei Beispiele zeigen die Abbildungen 3.14 und 3.15. Dazu wurden fogende Eingaben gemacht: K1=25, K2=25, K3=10, K4=25, K5=25 und K6=10, sowie K1=50, K2=25, K3=10, K4=20, K5=10 und K6=5.



3.14 Zykloide 1

3.15 Zykloide 2

3.7 Rosetten

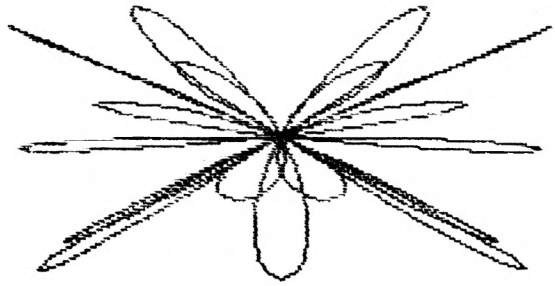
Rosetten werden durch die Gleichungen

$$x = k1 \cdot \sin k2 \cdot \varphi + k1 \cdot \cos k3 \cdot \varphi$$
$$y = k4 \cdot \cos k5 \cdot \varphi + k4 \cdot \sin k6 \cdot \varphi$$

gekennzeichnet. Folgende Zeilen im Programm Kardioide wurden geändert:

```
240    XP=XO+K1*SIN(K2*X)+K1*COS(K3*X)
250    YP=YO+K4*COS(K5*X)+K4*SIN(K6*X)
```

Die Rosette in Abbildung 3.16 wurde mit den Eingaben K1=50, K2=10, K3=7, K4=25, K5=8 und K6=9 gezeichnet.



3.16 Rosette

3.8 Sphärische Koordinaten

Räumliche Gebilde lassen sich mit den Funktionen

$$x = k1 \cdot \sin \varphi * \cos (k2 * \varphi)$$
$$y = k3 \cdot \sin \varphi * \sin (k4 * \varphi)$$
$$z = k5 \cdot \sin \varphi$$

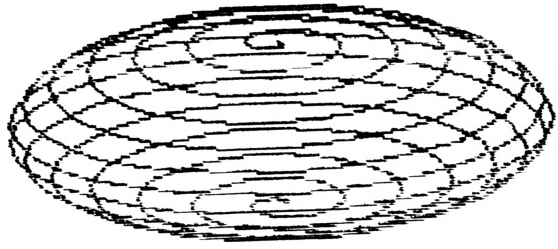
zeichnen. Das Programm zeigt Abbildung 3.17.

```
100 PI=3.14159
110 R=7:RD=2*PI*R
120 S=PI/32
130 INPUT"K1=";K1
140 INPUT"K2=";K2
150 INPUT"K3=";K3
160 INPUT"K4=";K4
162 INPUT"K5=";K5
170 XO=125:YO=75
180 Z=0
200 SCREEN 2:COLOR 1,11,6
```

```
210 PSET(XO,YO),6
220 XA=XO:YA=YO
230 FOR X=0 TO RD STEP S
240   XP=XO+K1*SIN(X/(2*R))*COS(K2*X)
250   YP=YO+K3*SIN(X/(2*R))*SIN(K4*X)
255   Z=Z+K5*SIN(X/(2*R))
260   YP=YP+Z
270   LINE (XA,YA)-(XP,YP),6
280   XA=XP:YA=YP
290 NEXT X
300 FOR I=0 TO 50
310   BEEP
320 NEXT I
330 GOTO 330
```

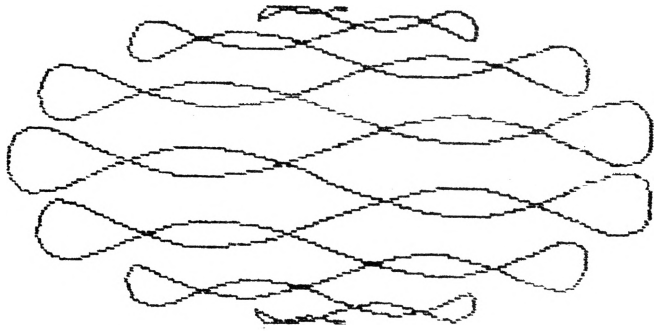
3.17 Programm Sphärische Koordinaten

Mit den Eingaben K1=100, K2=2, K3=20, K4=2 und K5=0.2 entsteht das Ellipsoid in Abbildung 3.18.



3.18 Ellipsoid

Das verzerrte Ellipsoid entsteht mit den Eingaben K1=100, K2=1, K3=10, K4=4 und K5=0.4.



3.19 Verzerrtes Ellipsoid

3.9 Darstellung von Funktionen

Mit dem Programm in Abbildung 3.20 können Funktionen der Form y=f(x) auf dem Bildschirm dargestellt werden. Zum Zeichnen dieser Funktionen wird ein Achsenkreuz festgelegt. Hierbei wird der Nullpunkt XO, YO und die Längen der Achsen XL und YL fesgelegt. Bei beiden Eingaben ist die Einheit 1 Bildpunkt (Pixel). Im Beispiel liegt der Ursprung des Koordinatensystems bei XO=126 und YO=87. Die Länge der X-Achse ist 260, die der Y-Achse 160 Bildpunkte.

Es soll die Spaltfunktion y=sin(x)/x gezeichnet werden. Der X-Bereich geht von -10 bis 10, der Y-Bereich von -1 bis 1. Die X-Achse soll 10, die Y-Achse 4 Teilstriche enthalten.

Für die Eingabe der Werte für das Koordinatensystem gibt es zwei Möglichkeiten, einmal durch Eingabe in den Zeilen 2020 bis 2060 oder als DATA-Anweisungen im Unterprogramm 3000. Dementsprechend ist die Zeile 100 in

```
100 GOSUB 3000:GOSUB 2200
```

zu ändern.

Mit diesen Angaben werden die Maßstabsfaktoren berechnet und das Koordinatensystem gezeichnet. Da der Nullpunkt der Bildpunkte in der linken oberen Ecke liegt, muß die Achslänge YL negativ sein (Zeile 2115) damit positive Y Werte nach oben gezeichnet werden.

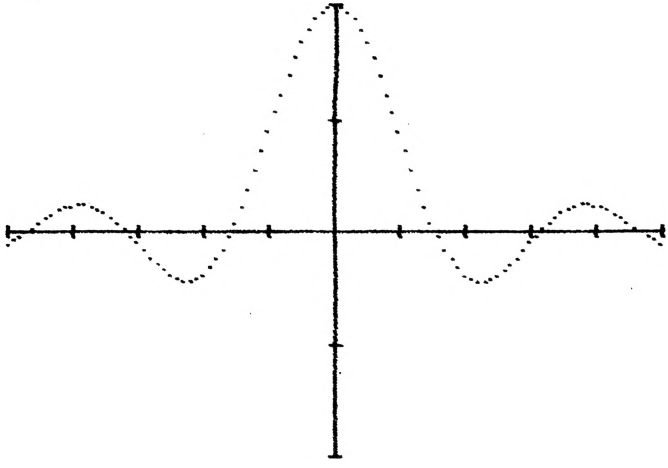
Die Funktion wird in den Zeilen 1000 bis 1070 programmiert. Als Schrittweite wird 0.1 vorgegeben. An das Unterprogramm zum Zeichnen werden die Variablen X und Y übergeben.

```
100 GOSUB 3000:GOSUB 2200
1000 ' FUNKTION
1010 S=.1
1020 FOR X=XA TO XE STEP S
1030 IF X<>0 THEN Y=SIN(X)/X
1040 GOSUB 2400
1050 NEXT X
1060 BEEP
1070 GOTO 1070
2000 ' EINGABE ACHSENKREUZ
2010 CLS
2020 INPUT"Nullpunkt    XO,YO:";XO,YO
2030 INPUT"Achslaengen  XL,YL:";XL,YL
2040 INPUT"X-Bereich    XA,XE:";XA,XE
2050 INPUT"Y-Bereich    XA,YE:";YA,YE
2060 INPUT"Teilung      XT,YT:";XT,YT
2070 XG=ABS(XA)+ABS(XE)
2080 YG=ABS(YA)+ABS(YE)
2100 X1=XL/XG*XA+XO
2110 X2=XL/XG*XE+XO
2115 YL=-YL
2120 Y1=YL/YG*YA+YO
2130 Y2=YL/YG*YE+YO
2140 MX=XL/XG:MY=YL/YG
2150 XT=XL/XT:Y1=YL/YT
2200 SCREEN 2:COLOR 1,11,6
2210 LINE(X1,YO)-(X2,YO),6
2220 LINE(XO,Y1)-(XO,Y2),6
2230 FOR X=X1 TO X2 STEP XT
2240 LINE (X,YO-2)-(X,YO+2)
```

```
2250 NEXT X
2260 FOR Y=Y1 TO Y2 STEP YT
2270 LINE (XO-2,Y)-(XO+2,Y)
2280 NEXT Y
2290 RETURN
2400 XP=X*MX+XO:YP=Y*MY+YO
2405 XP=INT(XP):YP=INT(YP)
2410 PSET(XP,YP),6
2420 RETURN
3000 ' DATEN ACHSENKREUZ
3010 READ XO,YO,XL,YL,XA,XE,YA,YE,XT,YT
3020 GOTO 2070
3100 DATA 126,87,240,160
3110 DATA -10,10,-1,1
3120 DATA 10,4
```

3.20 Zeichen von Funktionen

Ein Beispiel zeigt Abbildung 3.21.



4 Programmieren in Maschinencode

Das Programmieren in Maschinensprache ist wesentlich schwieriger, als das Programmieren in BASIC. Es setzt eine erhebliche Kenntnis des Prozessors und dessen Befehlsvorrat voraus. Dieses Kapitel soll keine Anleitung zum Lernen dieser Programmierung sein, sondern nur zeigen, wie man Maschinenprogramme mit dem MSX ausführen kann. Diejenigen, die sich mehr mit der Programmierung in Maschinencode befassen wollen, seien auf das Literaturverzeichnis im Anhang hingewiesen. Hier nur eine ganz kurze Einführung.

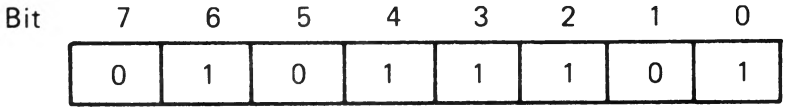
4.1 Einführung in die Programmierung mit Maschinencode

Die kleinste Recheneinheit eines Computers ist ein Bit, eine Größe, die zwei Werte annehmen kann, 0 oder 1. 8 Bit werden zu einem Byte zusammengefasst. Mit einem Byte lassen sich die Zahlen 0 bis 255 (2 -1, mit N = 8) darstellen.

Diese Zahl kann nun im Rechner verschiedenes bedeuten. Es kann eine echte Zahl sein. Es kann ein Zeichen sein. Die Zahl 38 stellt im MSX den Buchstaben A dar (siehe Handbuch Anhang A), oder es ist ein Befehl für den Prozessor. Auch die BASIC-Befehle werden durch eine solche

Zahl dargestellt. Der BASIC-Befehl THEN entspricht der Zahl 218.

Für die Schreibweise eines Bytes gibt es zwei Formen. Einmal die Angabe durch eine Dezimalzahl, zum zweiten als sogenannte Hexadezimalzahl. Diese Bezeichnung hat sich aus dem angelsächsischen Sprachgebrauch eingebürgert. In dieser Schreibweise werden die 8 Bit eines Bytes in zwei 4 bit Gruppen aufgeteilt. (Abbildung 4.1).



4.1 Ein Byte

Mit diesen 4 Bit lassen sich die Zahlen 0 bis 15 darstellen, wobei die Zahlen 0 bis 9 normal bezeichnet werden, die Zahlen 10 bis 15 durch Buchstaben A - F (Abbildung 4.2).

0 0 0 0	0
0 0 0 1	1
0 0 1 0	2
0 0 1 1	3
0 1 0 0	4
0 1 0 1	5
0 1 1 0	6
0 1 1 1	7
1 0 0 0	8
1 0 0 1	9
1 0 1 0	A
1 0 1 1	B
1 1 0 0	C
1 1 0 1	D
1 1 1 0	E
1 1 1 1	F

4.2 Hexadezimalzahlen

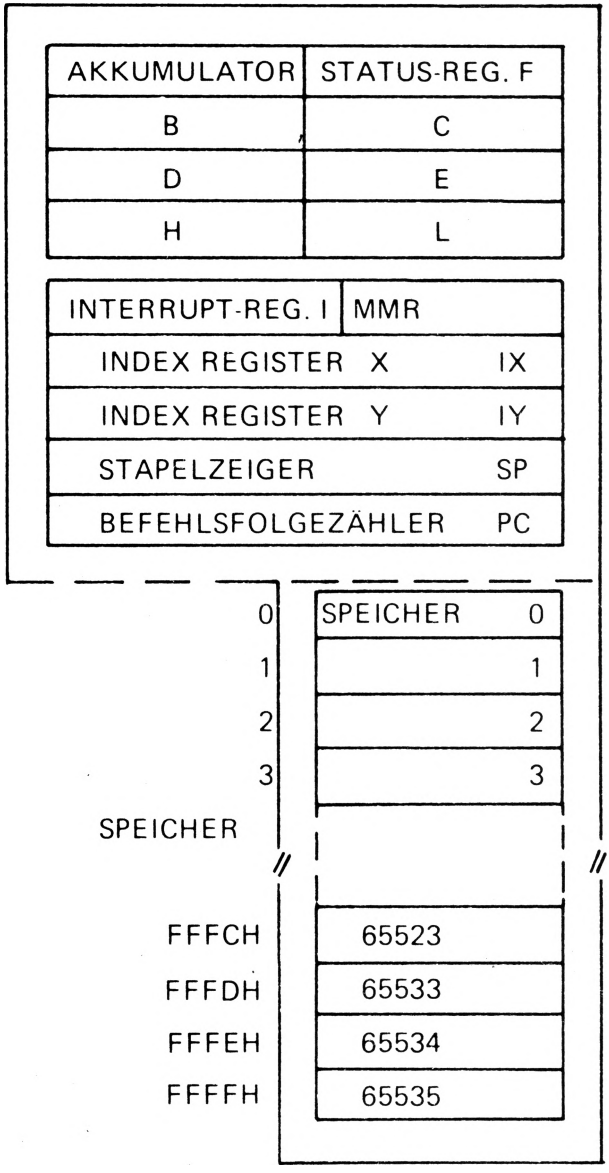
Die Zahl in dem oben angegegenen Byte entspricht einer Dezimalzahl 93 oder einer Hexadezimalzahl 5DH. Um eine Hexadezimalzahl von einer Dezimalzahl unterscheiden zu können, wird hinter die Zahl ein H gesetzt. Das obige Bitmuster ist die Zahl 5DH oder 93 dezimal. Davon abweichende Schreibweisen gibt es in BASIC. Hier wird eine Hex-Zahl durch Voranstellen von &H gekennzeichnet. 5DH ist in BASIC &H5D. Vielfach wird in der Literatur eine Hex-Zahl durch Voranstellen des $-Zeichens ($5D) gekennzeichnet. In seltenen Fällen wird eine Binärzahl als Bitmuster angegeben. Dies soll durch ein vorangestelltes %-Zeichen gekennzeichnet werden.

Nun wollen wir uns ein vereinfachtes Programmiermodell eines Rechners mit einer Z80 CPU anschauen (Abbildung 4.3). Im oberen Bildteil ist der, dem Programmierer, zugängliche Teil der Z80 CPU gezeigt. Dieser besteht aus dem Akkumulator, dem Statusregister F und den Registern B, C, D, E, H und L. Alle diese Register sind 8-bit Register, wobei allerdings die Register BC, DE und HL zu 16-bit Registern zusammengefasst werden.

Die Verwendung dieser Register werden wir gleich an einigen Beispielen zeigen.

Es folgt noch ein spezieller Registersatz, mit den Indexregistern, dem Stapelzeiger, dem Befehlsfolgezähler, Interrupt und MMR Register.

Im Rahmen der Möglichkeiten, mit der wir uns hier mit dem CPU beschäftigen wollen, sind von diesen Registern nur der Stapelzeiger und der Befehlsfolgezähler interessant. Der Befehlsfolgezähler enthält immer die Adresse der Speicherzelle, deren Inhalt als nächster Befehl ausgeführt werden soll. Der Stapelzeiger zeigt euf einen besonderen Speicherbereich, dem Stapel. Hier werden von der CPU Daten, die augenblicklich nicht gebraucht werden, zwischengespeichert. Im Blockschaltbild folgt noch der Speicher. Die Speicherzellen sind 8-bit Speicher. Die Adressen (Hausnummern) gehen von 0 bis 65535 (0 bis $FFFF).



4.3 Programmiermodell Z80

Dabei belegt das Betriebssystem und BASIC den Bereich 0 bis 32767 (0 bis 7FFFH). Aus diesen Speicherzellen kann nur gelesen, aber nichts hineingeschrieben werden (ROM - Read Only Memory). Ab 32768 (8000H) beginnt der Arbeitsspeicher. In diesem kann geschrieben und wieder ausgelesen werden (RAM - Random Access Memory, Speicher mit wahlfreiem Zugriff).

Was für Befehle kann nun eine CPU ausführen.

Dies sind einmal Transportbefehle. Mit diesen Befehlen können Daten vom Speicher zu den Registern, von den Registern zum Speicher und zwischen den Registern transportiert werden.

Beispiele:

LD A,(NN)	Lade den Akkumulator mit dem Inhalt der Speicherzelle, mit der Adresse NN. Die Angabe von NN in runden Klammern zeigt an, daß der Inhalt der Zelle NN gemeint ist. Diese Bezeichnungsweise ist aber bei der Angabe des Befehlssatzes nicht konsequent eingehalten worden.
LD C,N	Lade das Register C mit der Zahl N.
LD B,A	Lade das Register B mit dem Inhalt des Registers A.
LD BC,NN	Lade die Register B und C mit der 16 bit Zahl NN.

Desweiteren muß eine CPU Rechenoperationen ausführen können. Das eigentliche Rechenregister ist der Akkumulator. Das Ergebnis einer Rechenoperation wird dort gespeichert.

Beispiele:

ADD A,B Addiere zum Inhalt des Akkumulators den Inhalt des B-Registers. Das Ergebnis ist in A gespeichert.

ADD A, A Verdopple den Inhalt des Akkumulators.

Es gibt noch eine Vielzahl von Befehlsarten. Dies sind z.B. Vergleichsbefehle, mit denen die Inhalte von Registern oder Speicherzellen verglichen werden können. Bitbefehle, mit denen einzelne Bit gesetzt oder abgefragt werden können. Sprungbefehle, mit denen ein Programm an einer anderen Stelle weitergeführt werden kann. Blockbefehle, mit denen durch einen einzigen Befehl ganze Datenblöcke im Speicher verschoben werden können, usw. Auf eine ausführliche Befehlsdarstellung sei auf die angegebene Literatur und auf die Befehlsliste im Anhang verwiesen.

Der Maschinencode Monitor MONI.

Moni ist ein einfacher, in BASIC geschriebener Monitor zum Anzeigen und Ändern des Inhaltes von Speicherzellen. Mit ihm können folgende Funktionen durchgeführt werden.

Nach dem Starten des Programms mit RUN erscheint das Menü mit

F1: Zellen ändern
F2: Inhalt von Zellen
F3: Programm starten
F4: Ende

Ändern von Speicherinhalten.

Durch Drücken der Funktionstaste F1 wird das Programm zum Ändern von Speicherinhalten aufgerufen. Als nächstes muß die Adresse eingegeben werden. Diese Eingabe kann sowohl in dezimaler, wie in hexdezimaler Schreibweise erfolgen. Bei hexdezimaler Eingabe muß nach der Adresse ein H

folgen. Also 53248 oder D000H. Auf dem Bildschirm wird die Adresse in dezimaler Form und der Inhalt als Hexadezimalzahl angezeigt. Also zum Beispiel

```
53248 21 -->
```

An dieser Stelle kann nun eine neue Zahl in diese Zelle geschrieben werden. Diese Zahl muß eine Hexadezimalzahl sein. Geben wir 3E ein, so erscheint dann folgende Ausgabe

```
53248 21 --> 3E
53249 00 -->
```

Am unteren Ende des Bildschirms steht die Zeile

```
X: ENDE      RET:VOR      R: Zurück
```

Wird jetzt X (RETURN) eingegeben, so ist der Programmteil Ändern beendet und es erfolgt ein Rücksprung in das Menü. Wird nur RETURN eingegeben, so wird der Inhalt der angezeigten Zeile nicht verändert und die Adresse und der Inhalt der nächsten Zelle angezeigt.

```
53248 21 --> 3E
53249 00 -->
53250 00 -->
```

Wird R (RETURN) eingegeben, so wird die vorangegangene Zelle angezeigt. Auf diese Weise können Programme in Maschinencode in die Zellen des MSX eingegeben werden. Vom Menü aus kann das Programm mit F3 gestartet werden. Die Eingabe der Startadresse erfolgt wie im Programmteil "Ändern".

In BASIC wird das Programm mit

```
DEF USR (N)
A=USR(0)
```

gestartet, wobei N die eingegebene Adresse ist. Ist das

Maschinenprogramm mit RET (C9) abgeschlossen, so wird nach der Ausführung des Programms nach BASIC zurückgesprungen.

Mit F2 wird derInhalt von Zellen angezeigt. Nach der Eingabe der Adresse wird der Inhalt dieser und der folgenden 7 Zellen ausgegeben.

```
100  GOTO 1000
500  REM Hex Byte
510  B=PEEK(L)
520  B1=INT(B/16)+48
530  GOSUB 900
540  B1=B-INT(B/16)*16+48
550  GOTO 900
700  REM Hex Adresse
710  H=4096
720  B1=INT(B/H)+48
730  GOSUB 900
740  B=B-INT(B/H)*H:H=H/16
750  IF H>=1 THEN GOTO 720
760  PRINT "H";
770  RETURN
900  REM Hex Drucken
910  IF B1>57 THEN B1=B1+7
920  PRINT CHR$(B1);
930  RETURN
1000 SCREEN 0:KEY OFF:WIDTH 40
1010 CLS:LOCATE 3,3:PRINT "Mini Monitor"
1020 ON KEY GOSUB 2000,3000,4000,5000
1030 LOCATE 5,5:PRINT "F1:Zellen aendern"
1040 LOCATE 5,6:PRINT "F2:Inhalt von Zellen"
1050 LOCATE 5,7:PRINT "F3:Programm starten"
1060 LOCATE 5,8:PRINT "F4:Ende"
1070 KEY(1) ON:KEY(2) ON:KEY (3) ON:KEY (4) ON
1080 GOTO 1080
1500 REM Adresseneingabe
1510 CLS:LOCATE 3,3:INPUT "Adresse :";A$
1520 IF RIGHT$(A$,1)="H" THEN D=0 ELSE D=1:N=VAL
(A$):RETURN
1550 N=0
1560 FOR X=1 TO LEN (A$)-1
```

```
1570    B=ASC(MID$(A$,X,1))-48
1580    IF B>9 THEN B=B-7
1590    N=N*16+B
1600 NEXT X:RETURN
2000 REM Zelleninhalt aendern
2010 GOSUB 1500
2020 L=N:I=6
2030 LOCATE 1,20:PRINT "X:Ende RET:Vor R:Zurueck
"
2040 LOCATE 5,I: IF D=1 THEN PRINT L;
2050 B=L:IF D=0 THEN GOSUB 700
2060 PRINT "    ";:GOSUB 500
2070 PRINT " --> ";
2080 A$="":INPUT "";A$
2090 L=L+1:I=I+1
2100 IF I>18 THEN I=6
2110 IF A$="" THEN GOTO 2040
2120 IF A$="R" THEN L=L-2:GOTO 2040
2130 IF A$="X" THEN RETURN 1010
2140 IF LEN(A$)>2 THEN GOTO 2080
2150 N=0
2160 FOR X=1 TO 2
2170 B=ASC(MID$(A$,X,1))-48
2180 IF B>9 THEN B=B-7
2190 N=N*16+B
2200 NEXT X
2210 POKE L-1,N
2220 GOTO 2040
3000 REM Zelleninhalte ausgeben
3010 GOSUB 1500
3020 L=N:I=6
3030 LOCATE 1,20:PRINT "M:Zurueck ins Menue    W:
Weiter"
3040 LOCATE 0,I:B=L
3050 IF D=1 THEN PRINT L; ELSE GOSUB 700
3060 X=L
3070 FOR K=X TO X+7
3080   PRINT" ";:GOSUB 500:L=L+1
3090 NEXT K
3100 PRINT " ";
3120 FOR K=X TO X+7
3130 J=PEEK(K)
3140 IF J>13 THEN PRINT CHR$(J); ELSE PRINT ".";
```

```
3150 NEXT K
3160 A$=INKEY$:IF A$="" THEN 3160
3170 I=I+2:IF I>18 THEN I=6
3180 IF A$="W" THEN GOTO 3040
3190 IF A$="M" THEN RETURN 1010
3200 GOTO 3160
4000 GOSUB 1500
4010 DEF USR=N
4020 A=USR(0)
4030 RETURN 1010
5000 END
```

4.4 Maschinencode Monitor MONI

Maschinencode Monitor MSXMON.

Der Maschinencode Monitor MSXMON ist ein in Maschinencode geschriebener Monitor zur Ausgabe des Inhaltes von Speicherzellen auf Bildschirm oder Drucker. Ferner kann der Inhalt einer Zelle oder eines Registers geändert werden und es kann ein Hexdump mit Prüfsumme ausgegeben werden.

Einlesen des MSXMON von Kassette:

```
SCREEN 0
CLEAR 100,&HEC00
MAXFILES=0
BLOAD"CAS:",R
```

Auf dem Bildschirm muß nach kurzer Zeit

```
FOUND:MSXMON
```

angezeigt werden. Nach etwa 20 Sekunden meldet sich das System mit:

```
MSX MONITOR REV 1.0
*
```

Der * zeigt an, daß der Monitor auf eine Eingabe wartet.

Die Ausgabe eines Hexdumps mit dem Befehl:

D[AADRE[,EADR]]

Beispiel: DEC00,EC20

Der Speicherbereich zwischen EC00 und EC20 wird auf den Bildschirm ausgegeben.

```
EC00 C3 22 EC F3 01 05 00 11 C"ls....
EC08 0D FE 21 17 EC ED B0 21 .~!.lm0!
EC10 FF EB 22 7A F3 18 0B F1 .k"zs..q
EC18 C3 1C EC 00 E5 CD 22 EC C.l.eH"l
EC20 E1                      a
```

4.5 Hexdump 1

Wird keine Endadresse EADR angegeben, so werden 16 Zeilen auf dem Bildschirm angezeigt.

```
EC00 C3 22 EC F3 01 05 00 11 C"ls....
EC08 0D FE 21 17 EC ED B0 21 .~!.lm0!
EC10 FF EB 22 7A F3 18 0B F1 .k"zs..q
EC18 C3 1C EC 00 E5 CD 22 EC C.l.eH"l
EC20 E1 C9 F3 ED 73 3E F3 31 alsms>s1
EC28 3E F3 FD E5 DD E5 E5 D5 >s)eJeeU
EC30 C5 F5 21 00 EC 22 40 F3 Eu!.l"@s
EC38 3A 9A F2 B7 C2 ED EF 32 :r7Bmo2
EC40 9D F2 3D 32 9A F2 2A 3E r=2r*>
EC48 F3 22 9B F2 FB 21 56 EC s"(!Vl
EC50 CD 7D F1 F3 18 17 0D 0A M)qs....
EC58 4D 53 58 20 4D 6F 6E 69 MSX Moni
EC60 74 6F 72 20 20 52 65 76 tor  Rev
EC68 20 31 2E 30 00 2A 38 F3  1.0.*8s
EC70 22 CA F2 01 05 00 11 DD "Jr....J
EC78 F2 21 E4 FE ED B0 3E C3 r!d~m0>C
```

4.6 Hexdump 2

Wird keine Anfangsadresse angegeben, so wird ab der augenblicklichen Adresse, im obigen Beispiel ab EC80 der Zelleninhalt ausgegeben.

Mit D wird also der Inhalt der Zellen in hexadezimaler und in ASCII Darstellung angezeigt. Durch den Zusatz S wird eine Prüfsumme gebildet.

DS[AADR[,EADR]]

Beispiel:DSEC00

```
EC00 C3 22 EC F3 01 05 00 11 : DB
EC08 0D FE 21 17 EC ED B0 21 : ED
EC10 FF EB 22 7A F3 18 0B F1 : 8D
EC18 C3 1C EC 00 E5 CD 22 EC : 8B
EC20 E1 C9 F3 ED 73 3E F3 31 : 5F
EC28 3E F3 FD E5 DD E5 E5 D5 : 8F
EC30 C5 F5 21 00 EC 22 40 F3 : 1C
EC38 3A 9A F2 B7 C2 ED EF 32 : 4D
EC40 9D F2 3D 32 9A F2 2A 3E : F2
EC48 F3 22 9B F2 FB 21 56 EC : 00
EC50 CD 7D F1 F3 18 17 0D 0A : 74
EC58 4D 53 58 20 4D 6F 6E 69 : AB
EC60 74 6F 72 20 20 52 65 76 : C2
EC68 20 31 2E 30 00 2A 38 F3 : 04
EC70 22 CA F2 01 05 00 11 DD : D2
EC78 F2 21 E4 FE ED B0 3E C3 : 93
```

4.7 Hexdump mit Prüfsumme

Am Ende jeder Zeile wird eine Prüfsumme ausgegeben. Dazu werden alle Byte addiert, von dem Ergebnis werden aber nur die letzten beiden Stellen angezeigt. Diese Prüfsumme ist bei der Eingabe längerer Hexdumps eine große Hilfe.

Ändern von Speicherinhalten.

Der Inhalt von Speicherzellen wird mit

```
SAADR
```

geändert.

```
Beispiel: SD500
          D500 7D 3E
          D501 00 FF
          D502 00
          D503 .
          *
```

Mit SD500 wird das Ändern von Speicherzellen eingeleitet. Der Rechner zeigt die Adresse und den Inhalt an und wartet auf eine Eingabe.

```
D500 7D __
```

Wird nun eine Hexadezimalzahl eingegeben, so wird diese, im Beispiel 3E, in der Zelle gespeichert. Wird keine Eingabe gemacht, sondern nur die RETURN Taste betätigt, so wird der bisherige Inhalt der Zelle beibehalten. Diese Betriebsart wird durch Eingabe eines Punktes "." beendet. Mit dem Zeichen ^ wird zur vorherigen Zelle zurückgesprungen.

```
SD500
D500 3E
D501 FF
D502 00 ^
D501 FF .
*
```

Ändern und Anzeigen des Inhaltes der Register.

Mit dem Befehl

```
X[R]
```

wird der Inhalt des Registers angezeigt.

```
XA
A =00 __
```

Mit XA wird der Inhalt des Akkumulators angezeigt. Das Programm wartet auf eine Eingabe. Wird eine Hexadezimalzahl eingeben, so wird diese in dem Register gespeichert. Der Registerinhalt bleibt erhalten, wenn nur die RETURN Taste betätigt wird.

Wird nur X ohne Registername eingegeben, so wird der Inhalt aller Register in folgender Form angezeigt:

```
                SZ H PNC
A =00   F =40(01000000)
BC=0000 DE=FE12 HL=EBFF
IX=FFFF IY=FFFF SP=EA87 PC=EC00
```

Die Bit des FLAG-Registers werden einzeln angezeigt, so ist es leichter festzustellen, welche Status-Bits gesetzt sind.

Starten eines Programms.

Ein Maschinenprogramm wird mit

```
G[AADR][,BADR]
```

bei der Anfangsadresse AADR gestartet. Die Adresse BADR ist die Adresse, bei welcher das Programm angehalten wird. Dabei wird der Inhalt aller Register angezeigt.

Rückkehr zu BASIC.

Mit B wird in den BASIC Interpreter gesprungen. Von dort wird mit CMD oder CMD MON der Monitor wieder aufgerufen.

```
100 '
110 ' MSX MONITOR
120 '   REV 1.0
130 '
200 CLEAR 100,&HEC00
210 DEF USR=&HEC03
220 I=&HEC00
230 READ A$: IF A$="*" THEN 300
240 POKE I,VAL("&H"+A$): I=I+1
```

```
250 GOTO 230
300 A=USR(0)
310 END
1000 DATA C3,22,EC,F3,01,05,00,11
1010 DATA 0D,FE,21,17,EC,ED,B0,21
1020 DATA FF,EB,22,7A,F3,18,0B,F1
1030 DATA C3,1C,EC,00,E5,CD,22,EC
1040 DATA E1,C9,F3,ED,73,3E,F3,31
1050 DATA 3E,F3,FD,E5,DD,E5,E5,D5
1060 DATA C5,F5,21,00,EC,22,40,F3
1070 DATA 3A,9A,F2,B7,C2,ED,EF,32
1080 DATA 9D,F2,3D,32,9A,F2,2A,3E
1090 DATA F3,22,9B,F2,FB,21,56,EC
1100 DATA CD,7D,F1,F3,18,17,0D,0A
1110 DATA 4D,53,58,20,4D,6F,6E,69
1120 DATA 74,6F,72,20,20,52,65,76
1130 DATA 20,31,2E,30,00,2A,38,F3
1140 DATA 22,CA,F2,01,05,00,11,DD
1150 DATA F2,21,E4,FE,ED,B0,3E,C3
1160 DATA 21,9C,EF,32,E4,FE,22,E5
1170 DATA FE,31,32,F3,AF,32,9D,F2
1180 DATA 32,C9,F2,3E,DF,32,D7,F2
1190 DATA 32,DA,F2,FB,CD,74,F1,CD
1200 DATA 2C,ED,78,B7,28,E3,7E,23
1210 DATA FE,4C,20,0B,3E,FF,32,9D
1220 DATA F2,7E,23,FE,44,20,4C,FE
1230 DATA 44,20,11,7E,FE,53,20,06
1240 DATA 23,3E,FF,32,C9,F2,22,9F
1250 DATA F2,C3,FA,ED,22,9F,F2,FE
1260 DATA 53,CA,E8,EE,FE,58,CA,25
1270 DATA F0,FE,47,CA,34,EF,FE,42
1280 DATA 28,0B,FE,52,CA,B1,F1,2B
1290 DATA 22,9F,F2,18,16,F3,01,05
1300 DATA 00,11,E4,FE,21,DD,F2,ED
1310 DATA B0,AF,32,9A,F2,ED,7B,9B
1320 DATA F2,FB,C9,AF,32,9D,F2,CD
1330 DATA 74,F1,3E,3F,CD,A2,00,CD
1340 DATA 6A,F1,2A,9F,F2,7E,B7,28
1350 DATA 0A,FE,2C,28,06,CD,A2,00
1360 DATA 23,18,F2,CD,C0,00,C3,89
1370 DATA EC,AF,18,07,3E,2A,CD,A2
1380 DATA 00,3E,FF,32,9E,F2,CD,56
1390 DATA 01,06,00,21,A1,F2,22,9F
```

```
1400 DATA F2,CD,9F,00,FE,61,38,06
1410 DATA FE,7B,30,02,E6,5F,FE,0D
1420 DATA 28,6B,FE,1B,28,04,FE,03
1430 DATA 20,0D,78,B7,28,E3,3E,7F
1440 DATA CD,A2,00,10,FB,18,D2,FE
1450 DATA 08,20,0D,78,B7,28,D2,3E
1460 DATA 7F,CD,A2,00,05,2B,18,C9
1470 DATA FE,0C,20,0D,3A,9E,F2,B7
1480 DATA 28,BF,3E,0C,CD,A2,00,18
1490 DATA A3,FE,01,20,15,CD,9F,00
1500 DATA FE,41,38,0E,FE,60,38,A9
1510 DATA FE,61,38,A5,FE,7B,30,02
1520 DATA E6,5F,FE,20,38,9B,FE,7F
1530 DATA 28,97,4F,77,23,04,78,FE
1540 DATA 28,38,04,2B,05,0E,07,79
1550 DATA CD,A2,00,18,84,36,00,21
1560 DATA A1,F2,C9,DD,2A,9F,F2,0E
1570 DATA 00,61,69,DD,7E,00,B7,28
1580 DATA 23,FE,2C,28,1D,D6,30,FE
1590 DATA 0A,38,0C,D6,07,FE,0A,DA
1600 DATA 03,ED,FE,10,D2,03,ED,29
1610 DATA 29,29,29,B5,6F,0C,DD,23
1620 DATA 18,D9,DD,23,47,DD,22,9F
1630 DATA F2,C9,CD,C3,ED,79,B7,28
1640 DATA 03,22,CA,F2,78,B7,28,0F
1650 DATA CD,C3,ED,79,B7,CA,03,ED
1660 DATA 78,B7,C2,03,ED,18,07,2A
1670 DATA CA,F2,11,7F,00,19,22,CC
1680 DATA F2,3A,9D,F2,B7,C4,74,F1
1690 DATA 06,00,78,B7,20,13,DD,21
1700 DATA CE,F2,DD,36,00,00,CD,74
1710 DATA F1,2A,CA,F2,CD,86,F1,0E
1720 DATA 00,CD,6A,F1,2A,CA,F2,7E
1730 DATA CD,8B,F1,7E,81,4F,7E,FE
1740 DATA 20,38,08,FE,7F,28,04,FE
1750 DATA FF,20,02,3E,2E,DD,77,00
1760 DATA DD,23,DD,36,00,00,23,22
1770 DATA CA,F2,2B,04,ED,5B,CC,F2
1780 DATA B7,ED,52,30,2A,78,FE,08
1790 DATA 20,C7,CD,B4,EE,CD,9C,00
1800 DATA 28,A6,CD,9F,00,FE,0D,CA
1810 DATA 89,EC,CD,56,01,CD,9C,00
1820 DATA 28,FB,CD,9F,00,FE,0D,CA
```

```
1830 DATA 89,EC,CD,56,01,18,89,78
1840 DATA FE,08,28,0A,C5,06,03,CD
1850 DATA 6E,F1,C1,04,18,F1,CD,B4
1860 DATA EE,C3,89,EC,3A,C9,F2,B7
1870 DATA 28,0F,CD,6A,F1,3E,3A,CD
1880 DATA 9E,F1,CD,6A,F1,79,C3,8B
1890 DATA F1,3A,9D,F2,B7,20,10,3A
1900 DATA B0,F3,FE,25,D8,20,08,CD
1910 DATA DF,EE,3E,08,C3,A2,00,CD
1920 DATA 6A,F1,21,CE,F2,C3,7D,F1
1930 DATA CD,C3,ED,79,B7,CA,03,ED
1940 DATA 78,B7,C2,03,ED,22,CC,F2
1950 DATA CD,74,F1,CD,86,F1,CD,6A
1960 DATA F1,7E,CD,8B,F1,CD,6A,F1
1970 DATA CD,29,ED,78,B7,28,19,3D
1980 DATA 20,0A,7E,FE,2E,CA,89,EC
1990 DATA FE,5E,28,12,CD,C3,ED,B7
2000 DATA C2,03,ED,7D,2A,CC,F2,77
2010 DATA 2A,CC,F2,23,18,C7,2A,CC
2020 DATA F2,2B,18,C1,CD,C3,ED,ED
2030 DATA 5B,40,F3,79,B7,20,01,EB
2040 DATA 22,CC,F2,78,B7,28,2F,CD
2050 DATA C3,ED,79,B7,CA,03,ED,E5
2060 DATA 78,B7,28,18,CD,C3,ED,79
2070 DATA B7,CA,03,ED,E5,78,B7,C2
2080 DATA 03,ED,E1,22,DB,F2,7E,32
2090 DATA DA,F2,36,DF,E1,22,D8,F2
2100 DATA 7E,32,D7,F2,36,DF,2A,CC
2110 DATA F2,22,40,F3,F3,31,32,F3
2120 DATA ED,5B,40,F3,2A,3E,F3,2B
2130 DATA 72,2B,73,22,3E,F3,F1,C1
2140 DATA D1,E1,DD,E1,FD,E1,ED,7B
2150 DATA 3E,F3,FB,C9,E5,D5,21,08
2160 DATA 00,39,5E,23,56,1B,3A,D7
2170 DATA F2,FE,DF,28,17,2A,D8,F2
2180 DATA B7,ED,52,28,14,3A,DA,F2
2190 DATA FE,DF,28,08,2A,DB,F2,B7
2200 DATA ED,52,28,05,D1,E1,C3,DD
2210 DATA F2,F3,D1,E1,F1,F1,ED,73
2220 DATA 3E,F3,31,3E,F3,FD,E5,DD
2230 DATA E5,E5,D5,C5,F5,2A,3E,F3
2240 DATA 5E,23,56,23,22,3E,F3,1B
2250 DATA ED,53,40,F3,FB,3A,D7,F2
```

```
2260 DATA FE,DF,28,0F,2A,D8,F2,77
2270 DATA 3A,DA,F2,FE,DF,28,04,2A
2280 DATA DB,F2,77,AF,32,9D,F2,21
2290 DATA 1C,F0,CD,7D,F1,2A,40,F3
2300 DATA 22,CA,F2,CD,86,F1,CD,74
2310 DATA F1,C3,B4,F0,0D,0A,42,72
2320 DATA 65,61,6B,20,00,2A,9F,F2
2330 DATA 7E,B7,CA,B4,F0,1E,20,57
2340 DATA 23,7E,B7,28,07,5F,23,7E
2350 DATA B7,C2,03,ED,01,00,10,21
2360 DATA 3A,F1,7E,BA,23,20,04,7E
2370 DATA BB,28,08,23,23,0C,10,F2
2380 DATA C3,03,ED,2B,CD,74,F1,CD
2390 DATA 7D,F1,3E,3D,CD,A2,00,06
2400 DATA 00,79,FE,08,30,21,21,32
2410 DATA F3,09,7E,CD,8B,F1,E5,CD
2420 DATA 6A,F1,CD,29,ED,78,B7,CA
2430 DATA 89,EC,CD,C3,ED,B7,C2,03
2440 DATA ED,7D,E1,77,C3,89,EC,D6
2450 DATA 08,87,4F,21,32,F3,09,23
2460 DATA 7E,CD,8B,F1,2B,E5,7E,CD
2470 DATA 8B,F1,CD,6A,F1,CD,29,ED
2480 DATA 78,B7,CA,89,EC,CD,C3,ED
2490 DATA B7,C2,03,ED,EB,E1,73,23
2500 DATA 72,C3,89,EC,CD,74,F1,06
2510 DATA 0E,CD,6E,F1,21,31,F1,CD
2520 DATA 7D,F1,CD,74,F1,11,3D,F1
2530 DATA CD,0E,F1,2A,32,F3,E5,7C
2540 DATA CD,8B,F1,06,03,CD,6E,F1
2550 DATA 11,3A,F1,CD,0E,F1,E1,7D
2560 DATA CD,8B,F1,3E,28,CD,A2,00
2570 DATA 06,08,26,18,29,7C,CD,A2
2580 DATA 00,10,F7,3E,29,CD,A2,00
2590 DATA CD,74,F1,11,55,F1,21,34
2600 DATA F3,06,03,CD,19,F1,06,04
2610 DATA CD,19,F1,C3,89,EC,EB,CD
2620 DATA 7D,F1,EB,13,3E,3D,C3,A2
2630 DATA 00,CD,0E,F1,D5,5E,23,56
2640 DATA 23,EB,CD,86,F1,EB,D1,10
2650 DATA 03,C3,74,F1,CD,6A,F1,18
2660 DATA E8,53,5A,20,48,20,50,4E
2670 DATA 43,00,46,20,00,41,20,00
2680 DATA 43,20,00,42,20,00,45,20
```

```
2690 DATA 00,44,20,00,4C,20,00,48
2700 DATA 20,00,41,46,00,42,43,00
2710 DATA 44,45,00,48,4C,00,49,58
2720 DATA 00,49,59,00,53,50,00,50
2730 DATA 43,00,3E,20,18,30,CD,6A
2740 DATA F1,10,FB,C9,3E,0D,CD,9E
2750 DATA F1,3E,0A,18,21,7E,B7,C8
2760 DATA CD,9E,F1,23,18,F7,7C,CD
2770 DATA 8B,F1,7D,F5,0F,0F,0F,0F
2780 DATA CD,94,F1,F1,E6,0F,FE,0A
2790 DATA 38,02,C6,07,C6,30,F5,3A
2800 DATA 9D,F2,B7,20,04,F1,C3,A2
2810 DATA 00,F1,CD,A5,00,DA,89,EC
2820 DATA C9,CD,C3,ED,78,B7,C2,03
2830 DATA ED,E5,FD,E1,3A,AF,FC,B7
2840 DATA 20,32,21,EA,F1,CD,7D,F1
2850 DATA 2A,4A,FC,CD,86,F1,3E,2D
2860 DATA CD,9E,F1,2A,7A,F3,CD,86
2870 DATA F1,CD,6A,F1,3E,29,CD,9E
2880 DATA F1,C,74,F1,CD,1A,F2,C3
2890 DATA 89,EC,0D,0A,46,72,65,65
2900 DATA 20,28,20,00,21,FD,F1,CD
2910 DATA 7D,F1,C3,89,EC,0D,0A,53
2920 DATA 63,72,65,65,6E,20,6E,6F
2930 DATA 74,20,34,30,58,32,34,20
2940 DATA 74,65,78,74,20,6D,6F,64
2950 DATA 65,00,21,00,20,18,0A,EB
2960 DATA 3E,2D,CD,9E,F1,CD,86,F1
2970 DATA EB,CD,4A,00,23,5F,CD,4A
2980 DATA 00,23,57,CD,4A,00,23,4F
2990 DATA CD,4A,00,23,47,B1,B2,B3
3000 DATA C8,E5,FD,E5,E1,19,EB,21
3010 DATA 75,F2,CD,7D,F1,EB,CD,86
3020 DATA F1,EB,E1,78,B1,28,C8,E5
3030 DATA 2A,4A,FC,2B,B7,ED,52,E1
3040 DATA 30,1B,E5,2A,7A,F3,B7,ED
3050 DATA 52,E1,38,11,CD,4A,00,12
3060 DATA 13,23,0B,18,DE,0D,0A,4C
3070 DATA 6F,61,64,20,00,CD,74,F1
3080 DATA EB,CD,86,F1,21,8D,F2,CD
3090 DATA 7D,F1,C3,89,EC,20,20,4C
3100 DATA 6F,61,64,20,65,72,72,6F
3110 DATA 72,00,00,D6,D0,00,FF,A2
```

```
3120 DATA F2,42,00,54,00,00,00,00
3130 DATA 00,00,00,00,00,00,00,00
3140 DATA 00,00,00,00,00,00,00,00
3150 DATA 00,00,00,00,00,00,00,00
3160 DATA 00,00,00,00,00,00,00,00
3170 DATA 00,00,FF,D4,00,00,00,00
3180 DATA 00,00,00,00,00,00,00,DF
3190 DATA 00,00,DF,00,00,C9,C9,C9
3200 DATA C9,C9,00,00,00,00,00,00
3210 DATA 00,00,00,00,00,00,00,00
3220 DATA 00,00,00,00,00,00,67,0F
3230 DATA 67,0F,A8,0D,01,00,3F,01
3240 DATA 7F,04,EC,FB,E1,FB,69,0D
3250 DATA F3,0C,04,00,02,00,00,00
3260 DATA 8A,2E,90,E9,04,46,A8,7F
3270 DATA FF,00,75,03,DE,F3,90,0D
3280 DATA 20,00,F8,0B,17,03,FD,10
3290 DATA 42,01,E2,F2,A2,F2,44,ED
3300 DATA A2,EC,37,7C,34,7C,DB,39
3310 DATA FF,D4,04,00,02,00,D6,D0
3320 DATA 00,EC,00,00,00,00,00,00
3330 DATA 00,00,00,00,00,00,00,00
3340 DATA 00,00,00,00,00,00,00,00
3350 DATA 00,00,00,00,00,00,00,00
3360 DATA 00,00,00,00,00,00,00,00
3370 DATA 00,00,00,00,00,00,50,7E
3380 DATA 64,01,C8,00,CF,7C,00,00
3390 DATA 00,00,FF,D4,00,00,00,00
3400 DATA *
```

4.8 MSXMON

Der Assembler MSXASM.

Der Assembler MSXASM übersetzt Programme, die in memnotechnischer Schreibweise geschrieben sind, in ausführbaren Z80 Maschinencode.

Auf eine Beschreibung der Befehle des Z80 wird hier verzichtet, da hierfür schon ausführliche Literatur besteht.

Einlesen des MSXASM von Kassette:

```
SCREEN 0
CLEAR 100,&HD470
MAXFILES=0
BLOAD"CAS:",R
```

Hiermit wird sowohl der Assembler als auch der Monitor eingelesen. Mit dem Befehl

```
CMD MON
```

wird der Monitor aufgerufen. Ein in Assembler geschriebenes Programm wird mit

```
CMD ASM
```

assembliert. Mit diesem Befehl können noch Optionen angegeben werden, auf die später noch eingegangen wird.

Schreiben eines Programmes.

Zum Schreiben eines Assembler Programmes wird der BASIC Editor verwendet. Die Assembler Befehle werden in REM Anweisungen angegeben.

Beispiel:

```
1000 ' LD HL,CD
```

Alle Befehle sind mit Zeilennummern versehen. Damit können später leicht Änderungen und Erweiterungen durchgeführt werden. Nach der Zeilennummer steht die Anweisung REM bzw. deren Abkürzumg '.

Es folgt eine, bei den meisten Assemblern übliche Schreibweise. Steht in dieser Zeile ein Befehl, so ist davor ein Leerzeichen. Beginnt die Zeile mit einer Marke, so folgt diese unmittelbar auf das Zeichen '.

```
1000 '; Programmanfang
1010 'START:
1020 ' LD HL,CD
```

In Zeile 1000 steht ein Kommentar. Dieser beginnt mit einem ; , um ihn von einer Programmzeile zu unterscheiden. Ein Kommentar kann auch in einer Zeile mit einem Befehl stehen. Auch hier wird er durch einen ; vom Befehl getrennt.

Die nächste Zeile enthält eine Marke START: . Diese beginnt an der ersten Stelle nach dem ' und ist mit einem Doppelpunkt abgeschlossen. Bei diesem Assembler wird nicht zwischen Groß- und Kleinbuchstaben unterschieden. Intern werden alle Buchstaben als Großbuchstaben gelesen.

Die Zeile 1020 enthält einen Assemblerbefehl, mit einem vorausgehenden Leerzeichen.

Um sich die Schreibarbeit zu vereinfachen, wird man zwei Funktionstasten folgendermaßen belegen:

```
KEY 1,"'_"
KEY 3,"'_LD_"
```

Das Unterstreichungszeichen _ soll anzeigen, daß an dieser Stelle ein Leerzeichen steht. Der Befehl LD kommt so häufig vor, daß man damit eine Funktionstaste belegt.

Beim Schreiben eines Assembler Programms werden nicht nur die Befehle der Z80 CPU benötigt, sondern auch sogenannte Pseudo-Befehle. Folgende Pseudo-Befehle stehen beim MSX-ASM zur Verfügung:

```
1000 ' ORG 0D400H
```

Der ORG Befehl legt den Anfang des Maschinenprogramms im Speicher fest. In der obigen Anweisung wird festgelegt, daß der Code des folgenden Programms bei der Adresse D400H beginnt. Um eine Zahl von einer Marke zu unterscheiden, wird die Ziffer 0 vorangestellt.

```
1010 'GETCH EQU 009FH
```

Mit der EQU (EQUATE) Anweisung wird einer Marke eine Adresse oder eine Zahl zugewiesen. In der Zeile 1010

wird die Marke GETCH gleich der Zahl 009FH gesetzt. Nun kann, anstatt JP 009FH, JP GETCH geschrieben werden.

```
1020 ' END
```

Diese Anweisung kennzeichnet das Ende des Assemblerprogamms.

Wird in einem Programm Konstante oder auch freier Speicherplatz benötigt, so können dafür die DEF Anweisungen verwendet werden.

```
1030 ' DEFB 100,'A',0F9H
```

Die Anweisung DEFB (DEFine Byte) speichert die angegebenen Zahlen in einem Byte. Durch die obige Anweisung werden die Zahlen 64, 41 und F9 in drei aufeinander folgenden Speicherzellen abgelegt.

```
1040 ' DEFW 100,'AB',02F9H
```

Die Anweisung DEFW (DEFine Word) speichert eine Zahl in zwei aufeinander folgenden Speicherzellen. Mit der obigen Anweisung werden die Werte 64,00,41,42,02,F9 im Speicher abgelegt.

```
1050 ' DEFM 'HALLO'
```

Die Anweisung DEFM (DEF Memory) speichert die folgende Zeichenkette in aufeinanderfolgenden Bytes ab. Mit der obigen Anweisung werden die Werte 48,41,4C,4C,4F gespeichert.

```
1060 ' DEFS 10
```

Die Anweisung DEFS (DEFine Storage) reserviert Speicherzellen, ohne daß der Inhalt dieser Zellen festgelegt wird. Die Anweisung DEFS 10 hält 10 Bytes im Speicher frei.

Das Arbeiten mit dem Assembler soll an einem Beispiel gezeigt werden.

Folgende Aufgabe soll in Maschinencode programmiert wer-

den:

Über das Tastenfeld wird eine Zahl mit 20 Stellen eingegeben. Dabei wird mitgezählt, wie oft die 1, die 2 usw. vorkommt. Das Ergebnis wird ausgedruckt.

Die folgende Abbildung zeigt das mit dem BASIC Editor geschriebene Programm.

```
1000 ';
1010 '; Testprogamm
1020 ';
1022 ' ORG 0D400H
1024 ';
1030 'chget equ 009fh
1040 'chput equ 00a2h
1050 ';
1060 'ff equ 0ch
1070 'cr equ 0dh
1080 'lf equ 0ah
1090 ';
1100 'start:
1110 ' ld a,ff
1120 ' call chput
1130 ';
1140 ' ld hl,ipnot
1150 ' xor a
1160 ' ld b,10
1170 'mclr:
1180 ' ld (hl),a
1190 ' inc hl
1200 ' djnz mclr
1210 ';
1220 ' ld b,20
1230 'loop1:
1240 ' call chget
1250 ' cp '0'
1260 ' jr c,loop1
1270 ' cp '9'+1
1280 ' jr nc,loop1
1290 ' call chput
```

```
1300 ' sub '0'
1310 ' ld d,0
1320 ' ld e,a
1330 ' ld hl,ipnot
1340 ' add hl,de
1350 ' ld a,(hl)
1360 ' add a,1
1370 ' daa
1380 ' ld (hl),a
1390 ' djnz loop1
1400 ';
1410 ' call crlf
1420 ' ld hl,ipnot
1430 ' ld b,10
1440 'loop2:
1450 ' ld a,10
1460 ' sub b
1470 ' add a,'0'
1480 ' call chput
1490 ' ld a,'='
1500 ' call chput
1510 ' ld a,(hl)
1520 ' call putbcd
1530 ' inc hl
1540 ' call crlf
1550 ' djnz loop2
1560 ' ret
1570 ';
1580 '; Unterprogramme
1590 ';
1600 'putbcd:
1610 ' push af
1620 ' rrca
1630 ' rrca
1640 ' rrca
1650 ' rrca
1660 ' and 0fh
1670 ' or '0'
1680 ' call chput
1690 ' pop af
1700 ' and 0fh
1710 ' or '0'
1720 ' call chput
```

```
1730 ' ret
1740 ';
1750 'crlf:
1760 ' ld a,cr
1770 ' call chput
1780 ' ld a,lf
1790 ' call chput
1800 ' ret
1810 ';
1820 '; reservierter Speicher
1830 ';
1840 'ipnot: defs 10
1850 ';
1860 ' end
```

4.9 Quellcode des Beispiels

Programmbeschreibung:

Als Anfangsadresse für das Assemblerprogramm wird D400H gewählt. Das Unterprogramm chget beginnt bei Adresse 9FH. Dieses Programm wartet, bis eine Taste betätigt wird. Dann wird dieses Programm, mit dem ASCII Zeichen in Register A verlassen. Das Registerpaar AF wird dabei geändert.

Das Unterprogramm chput gibt den Inhalt des Registers A als ASCII Zeichen auf den Bildschirm aus.

Es werden drei Zeichen mit Namen versehen. Das Zeichen 0CH erhält den Namen FF (Form Feed= Neue Seite), 0DH wird CR (Cariage Return= Wagenrücklauf) und 0AH wird LF (Line Feed= Zeilenvorschub).

Das Programm beginnt mit der Marke start:. In den Zeilen 1030 bis 1040 wird das Zeichen für eine neue Seite ausgegeben.

Am Ende des Programms ist mit dem Namen ipnot Speicherplatz für 10 Bytes reserviert worden. In diese Zellen

wird die Häufigkeit der Ziffern bei der Eingabe der Zahlen mitgezählt. In den Zeilen 1140 bis 1200 werden diese Zellen mit Nullen gefüllt. Das HL Register enthält die Anfangsadresse und das B-Register die Länge (10) von ipnot.

Der Befehl XOR A löscht das Register A. In der folgenden Schleife wird der Inhalt von A in die in HL enthaltene Adresse gespeichert. Anschließend wird HL um Eins erhöht und mit djnz mclr zurückgesprungen. Dieser Befehl dekrementiert das B-Register und wird solange ausgeführt, solange der Inhalt des B-Registers nicht Null ist.

In den Zeilen 1220 bis 1390 erfolgt die Eingabe der Zahl und die Bestimmung, wie oft eine Ziffer vorkommt.

Die Länge der Zahl (20) wird in das B-Register geschrieben. Dann wartet das Programm mit call chget auf die Eingabe eines Zeichens. Ist dies erfolgt, so wird geprüft, ob es eine gültige Ziffer ist. Das Zeichen muß größer "0" und kleiner "10" sein. Wenn dies nicht der Fall ist, wird nach loop1 zurückgesprungen.

Das Zeichen wird auf den Bildschirm ausgegeben und anschließend wird das Zeichen "0" vom Inhalt des Registers A abgzogen. Wenn die Taste 4 gedrückt wurde, so enthält das A-Register den Wert 34H. Davon wird "0"=30H abgezogen und der Inhalt von A ist somit 4. Dieser Wert wird in das DE-Register übertragen. Die Anfangsadresse von ipnot wird in das HL-Register geholt und dazu der Inhalt des DE-Registers addiert. Im HL-Register ist nun die Adresse der Zelle, in welcher die Häufigkeit der Ziffer 4 gezählt wird. Der Inhalt wird geholt und dazu eine 1 addiert.

Der Befehl DAA (Decimal Adjust) wandelt den Inhalt des Registers A in eine zweistellige Dezimalzahl.

Danach wird der Inhalt des Registers A nach der im HL-Register gespeicherten Adresse geschrieben. Mit DJNZ LOOP1 wird, wenn das B-Register noch nicht Null ist, zurückgesprungen.

In der nächsten Schleife werden die gespeicherten Zahlen ausgegeben. In das B-Register wird eine 10 geschrieben. Innerhalb der Schleife wird das A-Register mit 10 geladen und davon der Inhalt des B-Registers abgezogen. Beim ersten Schleifendurchlauf ist das Ergebnis Null, beim zweiten Durchlauf Eins usw., da am Ende der Schleife immer vom B-Register Eins abgezogen wird.

Zum Inhalt des Registers A wird "0"=30H addiert. Somit enthält das A-Register das der Zahl entsprechende ASCII-Zeichen. Dieses, gefolgt von dem Zeichen "=" wird auf den Bildschirm ausgegeben.

Der Inhalt der Zelle wird durch das Unterprogramm putbcd ausgegeben. Danach wird der Inhalt des HL-Registers um Eins erhöht und nach Ausgabe der Zeichen Wagenrücklauf, Zeilenvorschub wird DJNZ LOOP2 zurückgesprungen.

Im Unterprogramm putbcd wird der Inhalt einer Speicherzelle als Dezimalzahl ausgegeben. Das AF-Register wird auf dem Stapel zwischengespeichert. Dann wird der Inhalt des A-Registers vier Mal nach rechts geschoben. Damit werden die höheren 4 Bit in die unteren 4 Bit übertragen. Durch and 0fh werden die oberen 4 Bit der Speicherzelle gelöscht. Mit or "0" wird 30H addiert und die Zahl in das entsprechende ASCII-Zeichen gewandelt. Nach der Ausgabe des Zeichens, werden die unteren 4 Bit auf die gleiche Weise ausgegeben.

Nachdem das Programm in den Rechner eingegeben ist, wird es mit CMD ASM assembliert. Diese Assemblierung findet in zwei Schritten statt. Im ersten Durchlauf (pass) wird die Tabelle mit den Namen und Marken angelegt, die Befehle dekodiert und die Adressen der Marken und Namen bestimmt. Im zweiten Durchlauf werden dann die Adressen eingesetzt und das Programm auf den Bildschirm ausgegeben und an der mit der ORG Anweisung angegeben Adresse gespeichert.

Mit dem CMD ASM können folgende Optionen übergeben werden:

CMD ASM"XXX"	Es werden nur Fehlermeldungen auf den Bildschirm ausgegeben.

CMD ASM"PXX"	Es wird das übersetzte Programm auf Drucker ausgegeben.
CMD ASM"PPX"	Es wird das übersetzte Programm und die Tabelle mit den Marken und den Namen auf den Drucker ausgegeben.

Der Assembler kennt noch einen weiteren CMD Befehl. Mit

CMD S "<Zeichenkette>"

wird im Quelltext nach der <Zeichenkette> gesucht.

Die folgende Abbildung zeigt nun das assemblierte Programm mit den eingetragenen Adressen.

```
     MSXASM   Assembler  Rev 2.5     PAGE     1


1000:                    ;
1010:                    ; Testprogamm
1020:                    ;
1022:   D400              ORG 0D400H
1024:                    ;
1030:   009F =           chget equ 009fh
1040:   00A2 =           chput equ 00a2h
1050:                    ;
1060:   000C =           ff equ 0ch
1070:   000D =           cr equ 0dh
1080:   000A =           lf equ 0ah
1090:                    ;
1100:   D400             start:
1110:   D400 3E0C         ld a,ff
1120:   D402 CDA200       call chput
1130:                    ;
1140:   D405 216FD4       ld hl,ipnot
1150:   D408 AF           xor a
1160:   D409 060A         ld b,10
1170:   D40B             mclr:
1180:   D40B 77           ld (hl),a
```

```
1190:   D40C 23         inc hl
1200:   D40D 10FC       djnz mclr
1210:                   ;
1220:   D40F 0614       ld b,20
1230:   D411            loop1:
1240:   D411 CD9F00     call chget
1250:   D414 FE30       cp '0'
1260:   D416 38F9       jr c,loop1
1270:   D418 FE3A       cp '9'+1
1280:   D41A 30F5       jr nc,loop1
1290:   D41C CDA200     call chput
1300:   D41F D630       sub '0'
1310:   D421 1600       ld d,0
1320:   D423 5F         ld e,a
1330:   D424 216FD4     ld hl,ipnot
1340:   D427 19         add hl,de
1350:   D428 7E         ld a,(hl)
1360:   D429 C601       add a,1
1370:   D42B 27         daa
1380:   D42C 77         ld (hl),a
1390:   D42D 10E2       djnz loop1
1400:                   ;
1410:   D42F CD64D4     call crlf
1420:   D432 216FD4     ld hl,ipnot
1430:   D435 060A       ld b,10
1440:   D437            loop2:
1450:   D437 3E0A       ld a,10
1460:   D439 90         sub b
1470:   D43A C630       add a,'0'
1480:   D43C CDA200     call chput
1490:   D43F 3E3D       ld a,'='
1500:   D441 CDA200     call chput
1510:   D444 7E         ld a,(hl)
1520:   D445 CD4FD4     call putbcd
1530:   D448 23         inc hl
1540:   D449 CD64D4     call crlf
1550:   D44C 10E9       djnz loop2
1560:   D44E C9         ret
1570:                   ;
1580:                   ; Unterprogramme
1590:                   ;
1600:   D44F            putbcd:
1610:   D44F F5         push af
```

```
1620:   D450 0F          rrca
1630:   D451 0F          rrca
1640:   D452 0F          rrca
1650:   D453 0F          rrca
1660:   D454 E60F        and 0fh
1670:   D456 F630        or '0'
1680:   D458 CDA200      call chput
1690:   D45B F1          pop af
1700:   D45C E60F        and 0fh
1710:   D45E F630        or '0'
1720:   D460 CDA200      call chput
1730:   D463 C9          ret
1740:                    ;
1750:   D464             crlf:
1760:   D464 3E0D        ld a,cr
1770:   D466 CDA200      call chput
1780:   D469 3E0A        ld a,lf
1790:   D46B CDA200      call chput
1800:   D46E C9          ret
1810:                    ;
1820:                    ; reservierter Speicher
1830:                    ;
1840:   D46F             ipnot: defs 10
1850:                    ;
1860:   D479             end
```

```
009F  CHGET     00A2  CHPUT
000C  FF        D46F  IPNOT
D437  LOOP2     D40B  MCLR

000D  CR        D464  CRLF
000A  LF        D411  LOOP1
D44F  PUTBCD    D400  START
```

4.10 Assembliertes Programm

Nachdem beim Assemblieren keine Fehler mehr aufgetreten sind, wird mit CMD MON der Monitor aufgerufen und dort das Programm mit GD400 gestartet. Die folgende Abbildung zeigt ein Beispiel.

```
12343456123456123456
0=00
1=03
2=03
3=04
4=04
5=03
6=03
7=00
8=00
9=00
```

4.11 Beispiel

Im ROM des Monitors ist von 0000H bis 0160H eine Sprungtabelle gespeichert, über welche die wichtigsten System-Monitor Prgramme angesprungen werden können. Es folgt eine Liste der am häufigsten gebrauchten Unterprogramme.

009C	CHSNS	Testet das Tastenfeld, ob eine Taste gedrückt wurde. Dann ist das Z Bit gesetzt. AF-Reg ist verändert.
009F	CHGET	Das Unterprogramm wartet, bis eine Taste gedrückt wird. Das ASCII-Zeichen ist im A-Reg. AF-Reg. sind verändert.
00A2	CHPUT	Inhalt des A-Reg. wird auf den Bildschirm ausgegeben.
00A5	LPTOUT	Inhalt des A-Reg. wird aus den Drucker ausgegeben. C Bit ist gesetzt, wenn Drucken abgebrochen wird.

00A8	LPTSTT	Testet Drucker. A-Reg.= FF und Z Bit nicht gesetzt, wenn Drucker bereit. A-Reg.= 0 und Z Bit gesetzt, wenn Drucker nicht bereit.
00AE	PINLIN	Speichert eine Zeile vom Tastenfeld. Mit RETURN wird Unterprogramm verlassen. Anfangsadresse im HL-Register. C Bit ist gesetzt, wenn STOP Taste gedrückt wurde.
00B7	BREAKX	Testet, ob ^STOP Taste gedrückt wurde. Dann ist C Bit gesetzt.
00C0	BEEP	Ausgabe eines Tones.
00C3	CLS	Löscht Bildschirm, wenn Z Bit gesetzt.
00C6	POSIT	Setzt Cursor. Nummer der Spalte im H-Reg., Nummer der Zeile im L-Reg. AF-Reg. verändert.

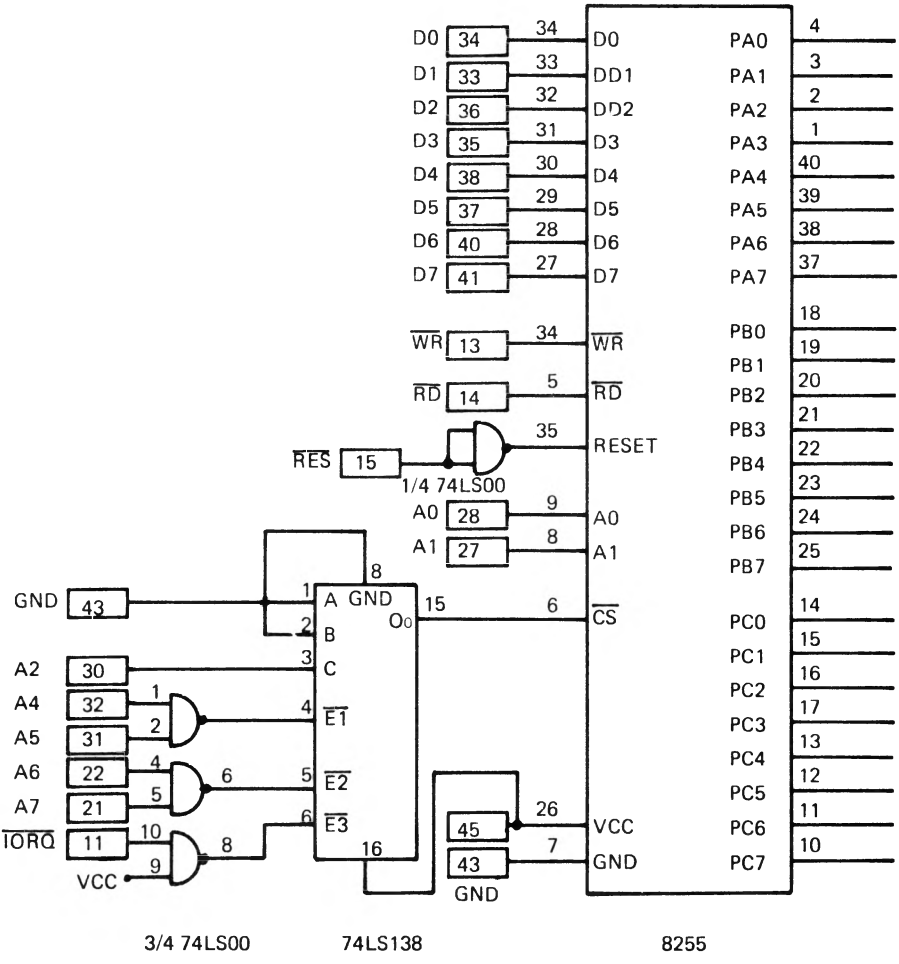
Ein- und Ausgabe mit dem PPI 8255

Für die MSX-Rechner ist eine Interface-Platine entwickelt worden, mit deren Hilfe Signale über den PPI 8255 ausgegeben und eingelesen werden können. Ein ausführliches Datenblatt dieses Bausteins ist im Anhang angegeben.

Das Schaltbild zeigt Abbildung 4.12

Die Adressdekodierung erfolgt mit drei 74LS00 und einem 74LS138. Die Eingänge des 74LS138 sind wie folgt belegt:

A = GND
B = GND
C = A2
E1= A4 & A5
E2= A6 & A7
E3= IORQ



4.12 Schaltbild

Wird als Auswahlleitung der Ausgang O0 verwendet, dann ist dieser aktiv, wenn folgende Adresskombination vorliegt:

```
7 6 5 4 3 2 1 0   Adressbit
1 1 1 1 x 0 x x
```

Das entspricht den Adressen F0 bis F3 und F8 bis FB. Die

Adressen A0 und A1 werden am 8255 benötigt, um die 4 Register auszuwählen. Diese haben folgende Adressen:

F0 = Tor A
F1 = Tor B
F2 = Tor C
F3 = Steuerregister.

Mit den BASIC-Befehlen INP(I) und OUT I,3 können Daten eingelesen oder ausgegeben werden. Nach dem Einschalten des Rechners sind alle Tore in Mode 0 auf Eingang geschaltet. Sollen alle Tore Ausgänge sein, so muß folgender Befehl an das Steuerregister gegeben werden:

```
OUT &HF3,&H&80
```

Dann können mit

```
OUT &HF0, &HBB
OUT &HF1, &HBB
OUT &HF2, &HBB
```

Daten andere Tore ausgegeben werden. Die folgende Tabelle in Abbildung 4.13 zeigt die Steuerworte für die verschiedenen Möglichkeiten.

D7	D6	D5	D4	D3	D2	D1	D0	TOR A	TOR B	TOR CL	TOR CU	HEX
1	0	0	0	0	0	0	0	AUS	AUS	AUS	AUS	80
1	0	0	0	0	0	0	1	AUS	AUS	EIN	AUS	81
1	0	0	0	0	0	1	0	AUS	EIN	AUS	AUS	82
1	0	0	0	0	0	1	1	AUS	EIN	EIN	AUS	83
1	0	0	0	1	0	0	0	AUS	AUS	AUS	EIN	88
1	0	0	0	1	0	0	1	AUS	AUS	EIN	EIN	89
1	0	0	0	1	0	1	0	AUS	EIN	AUS	EIN	8A
1	0	0	0	1	0	1	1	AUS	EIN	EIN	EIN	8B
1	0	0	1	0	0	0	0	EIN	AUS	AUS	AUS	90
1	0	0	1	0	0	0	1	EIN	AUS	EIN	AUS	91
1	0	0	1	0	0	1	0	EIN	EIN	AUS	AUS	92
1	0	0	1	0	0	1	1	EIN	EIN	EIN	AUS	93
1	0	0	1	1	0	0	0	EIN	AUS	AUS	EIN	98
1	0	0	1	1	0	0	1	EIN	AUS	EIN	EIN	99
1	0	0	1	1	0	1	0	EIN	EIN	AUS	EIN	9A
1	0	0	1	1	0	1	1	EIN	EIN	EIN	EIN	9B

4.13 Tabelle der Steuerworte

Mit

```
OUT &HF3, &H90
```

kann Tor A als Eingang, Tor B und Tor C als Ausgang benutzt werden.

Dazu noch eine allgemeine Bemerkung.

Bis jetzt haben wir uns nicht darum gekümmert, wie ein Rechner eine 1 oder 0 intern darstellt. Wenn wir aber z. B. die Binärzahl 0 1 0 1 1 1 0 1 über das Tor eingeben wollen, so müssen wir wissen, welchen Spannungswert einer 1 und welcher Spannungswert einer 0 entspricht. Denn es sind ja Spannungen, die an die Leitungen angelegt werden. Bei der Kennzeichnung dieser Spannungswerte hat sich die Bezeichnung TTL-kompatibel durchgesetzt. TTL ist die Transistor-Transistor-Logik, mit der eine der umfangreichsten Familien von integrierten Schaltungen aufgebaut wurde. Durch sie wurden folgende Pegel festgelegt.

1. Für die Ausgabe vom Daten:

   Eine 1 entspricht mindestens 2,4 Volt,
   eine 0 entspricht maximal 0,4 Volt.

2. Für die Eingabe von Daten:

   Eine 1 entspricht mindestens 2,0 Volt,
   eine 0 entspricht maximal 0,8 Volt.

In beiden Fällen darf eine Spannung von 5,0 Volt nicht überschritten werden.

Wird in einem Datenblatt angegeben, daß die Ein- oder Ausgänge TTL-kompatibel sind, so werden obige Spannungspegel eingehalten. Das gilt dann auch für die Belastbarkeit. Ein Ausgang kann durch mindestens einen TTL-Eingang belastet werden. Dies sind aber Ströme von ca. 1 mA, so daß wir für das Schalten größerer Lasten Verstärker zwischen Tor und Last einschalten müssen.

Beim Experimentieren mit der I/O-Platine muß sorgfältig mit dem MSX-Computer und den integrierten Schaltkreisen

umgegangen werden. Vor dem Berühren von Daten- und Adressleitungen, auch bei ausgeschaltetem Rechner, sollte eine statische Aufladung durch Berühren einer Masseleitung abgeleitet werden.

Die Platine sollte niemals bei eingeschaltetem Rechner abgezogen oder aufgesteckt werden. Ferner ist bei Anschluß externer Geräte auf eine gute Masseverbindung zu achten.

Auf der Platine ist auch Platz für ein 2764 EPROM freigehalten worden. Dieses kann für eigene Programme verwendet werden.

Notizen

Befehlsübersicht 5

Verzeichniss der MSX-BASIC Befehle.

Erläuterung zur Syntax.

Angaben, die in eckigen Klammern [ ] stehen, brauchen nicht unbedingt eingegeben werden.

Beispiel:

```
AUTO [Anfang[,Erhöhung]]
```

Folgende Eingaben sind möglich:

```
AUTO
AUTO 1000
AUTO 1500,20
```

Folgende Abkürzungen werden verwendet:

```
AADR = Anfangsadresse
EADR = Endadresse
SADR = Startadresse
ZNR  = Zeilennummer
```

X,Y können Zahlen oder arithmetische Ausdrücke sein.
I ist eine ganze Zahl.
C ist eine Konstante oder ein Zeichen.
A$ ist ein Zeichen, eine Zeichenkette oder eine Zeichenvariable.

Steht eine Bezeichnung in spitzen Klammern < >, so ist diese einschließlich der Klammern durch einen Ausdruck zu ersetzen.

Beispiel:

```
BLOAD "<Name>",R
```

Mit <Name> gleich MSXMON lautet der Befehl

```
BLOAD "MSXMON",R
```

ABS (X)

Die ABS Funktion bildet den Absolutwert von X.

```
PRINT ABS (-10)
 10
Ok
```

X AND Y

UND-Funktion. Wenn X und Y ungleich Null sind, dann ist X AND Y gleich 1. Ist eine der beiden Variablen Null, dann ist X AND Y ebenfalls 0.

```
PRINT 3 AND 5
 1
Ok
PRINT 0 AND 5
 0
Ok
```

AS #I

Zuweisung einer Kanalnummer. Siehe auch OPEN.

ASC (A$)

Die ASC Funktion liefert den ASCII Wert des ersten Zeichens von A$.

```
PRINT ASC("EMIL")
 69
Ok
```

Die ASCII Werte sind im Anhang in der ASCII Tabelle angegeben.

ATN [X]

Die Funktion ATN berechnet den Arkustangens von X im Bogenmaß. Der Wert liegt immer zwischen -PI/2 und PI/2.

```
PRINT ATN(100)
 1.5607966601082
Ok
```

AUTO[Zeilennummer[,Erhöhung]]

Die Zeilennummer wird automatisch erzeugt und erhöht.

```
AUTO 100,20
```

Die Zeilennummerierung beginnt mit 100 und wird jeweils um 20 erhöht. Mit ^C wird diese Betriebsart beendet. Mit einem * wird angezeigt, wenn eine Zeilennummer schon vorhanden ist. Erscheint nach RET z. B. die Anzeige 120*, so ist die Zeilennummer 120 schon im Programm vorhanden.

BASE[I]

Mit A=BASE(I) wird die Basisadresse eines Registers des Videoprozessors der Variablen A zugewiesen. I ist eine Zahl zwischen 0 und 12 und hat folgende Bedeutung

0	Namenliste	(Textmodus 1)
2	Mustertabelle	(Textmodus 1, Grafikmodus 1)
5	Namenliste	(Textmodus 2)
6	Farbtabelle	(Textmodus 2)
7	Mustertabelle	(Textmodus 2)
10	Namenliste	(Grafikmodus 1)
11	Farbtabelle	(Grafikmodus 1)
15	Namenliste	(Grafikmodus 2)
17	Mustertabelle	(Grafikmodus 2)

BEEP

Ausgabe eines Tones (800 Hz,0.25 sec) an den Lautsprecher. Kann auch durch PRINT CHR$(7) ersetzt werden.

BIN$(X)

Die Variable X wird als Ganzzahl interpretiert und in eine binäre Zeichenkette gewandelt.

```
PRINT BIN$(100)
 1100100
Ok

PRINT BIN$(100.5)
 1100100
Ok
```

Anmerkung: Eine Binärzahl ist eine Zahl, die nur durch die Ziffern 0 und 1 dargestellt wird.

BLOAD "<Name>"[,R]

Die Maschinencode-Datei <Name> wird von Kassette eingelesen. Das Programm wird ab der in BSAVE angegebenen Adresse gespeichert. Wenn die Option R angeben ist, wird das Programm automatisch gestartet. Die Startadresse ist AADR, wenn keine Adresse SADR in BSAVE angeben ist.

BSAVE "<Name>",AADR,EADR[,SADR][,2]

Der Speicherinhalt von AADR bis EADR wird unter dem Namen <Name> auf Kassette aufgezeichnet. Ist die Adresse SADR angegeben, so wird das Maschinenprogramm an dieser Stelle automatisch gestartet. Die normale Aufzeichnungsgeschwindigkeit ist 1200 BAUD. Mit der Option ,2 wird mit 2400 Baud aufgezeichnet.

```
BSAVE "TEST",&H9000,&H92FF
```

schreibt den Inhalt von 9000H bis 92FFH auf Kassette.

CDBL(X)

Die Variable X wird in eine doppelt lange Variable gewandelt.

CHR$(I)

Der ASCII Code I wird in das entsprechende Zeichen gewandelt. Dabei ist I eine Ganzzahl zwischen 0 und 255.

```
100 FOR I=0 TO 255
110 PRINT CHR$(I);
120 NEXT I
```

CINT(X)

Wandelt die Variable X in eine Ganzzahl zwischen -32786 und 32767 um. Dabei werden die Stellen hinter dem Dezimalpunkt abgeschnitten. In anderen BASIC Dialekten wird auf- oder abgerundet.

```
PRINT CINT(12.34)
 12
Ok
PRINT CINT(123.54)
 123                    (MSX BASIC, sonst 124 !)
Ok
```

CIRCLE (X,Y),R[,Farbe [,Anfang,Ende [,Proportion]]]

Dieser Befehl zeichnet eine Ellipse auf den Bildschirm. X, Y sind die Koordinaten des Mittelpunktes, R der Radius der größten Hauptachse. Diese Angaben müssen gemacht werden. Zusätzliche Angaben betreffen die Farbe der Linie, den Anfang und das Ende der Linie (Vielfaches von PI), sowie eine Verzerrung der X- und Y-Achse.

```
100 SCREEN 2
110 CIRCLE (100,100),40,1
120 GOTO 120
```

zeichnet eine Ellipse etwa in Bildschirmmitte.

Mit

```
110 CIRCLE (100,100),40,1,,,1.4
```

wird diese Ellipse zu einem Kreis verzerrt.

CLEAR [I[,ADR]]

Dieser Befehl löscht den Inhalt aller Variablen und alle Dimensionierungsangaben für Felder. Die zusätzliche Ganzzahl I reserviert Speicherplatz am oberen Ende des Speichers. Dort sind bei der Initialisierung schon 200 Bytes für Zeichenketten freigehalten. Ist die Adresse ADR angegeben, so ist dies die höchste Adresse des BASIC Bereichs. Dieser Befehl kann auch für eine dynamische Dimensionierung von Feldern verwendet werden.

```
100 CLEAR
110 INPUT"N=";N
120 DIM A(N)
130 GOTO 100
```

In Zeile 120 wird das Feld A in Abhängigkeit von der Eingabe N dimensioniert. Vor der Zeile 100 dürfen keine Vereinbarungen gemacht werden.

CLOAD "<Name>"

Dieser Befehl sucht das Programm <Name> auf der Kassette und liest es, wenn es gefunden wurde, in den Speicher. Besteht <Name> nur aus einem Lehrzeichen (<Name = " "), dann wird das nächste gefundene Programm gelesen.

CLOAD?"<Name>"

Mit dieser Anweisung wird ein auf Kassette aufgezeichnetes Programm mit dem im Speicher befindlichen Programm verglichen. Wird dabei eine Abweichung gefunden, so wird die Fehlermeldung

VERIFICATION ERROR

ausgegeben.

CLOSE #I

Dieser Befehl schließt einen geöffneten Datenkanal und gibt den zugehörigen Pufferspeicher frei. I ist die Nummer des Kanals.

CLS

Mit CLS wird der Bildschirm gelöscht. Dieser Befehl gilt in allen Bildschirmbetriebsarten.

COLOR I[,J[,K]]

Durch COLOR wir die Farbe der Schrift (I), des Hintergrunds (J) und des Randes (K) ausgewählt. Es gilt die folgende Farbtabelle:

0	Transparent
1	Schwarz
2	Grün
3	Hellgrün
4	Dunkelblau

5	Hellblau
6	Dunkelrot
7	Zyanblau
8	Rot
9	Hellrot
10	Dunkelgelb
11	Hellgelb
12	Dunkelgrün
13	Magentarot
14	Grau
15	Weiß

```
1000 COLOR 15,1,1:SCREEN 2
1005 MAXFILES=1
1010 OPEN "GRP:" FOR OUTPUT AS #1
1020 PRESET (10,5)
1030 PRINT #1,"EIN GUT EINGESTELLTER FERNSEHER"
1040 PRESET (10,15)
1050 PRINT #1,"ZEIGT 15 FARBEN UND SCHWARZ"
1060 CLOSE #1
1070 FOR G%=2 TO 15
1080 COLOR G%
1090 LINE (G%*16-15,30)-(G%*16,184),,BF
1100 NEXT G%
1110 GOTO 1110
```

CONT

Mit CONT wird ein durch STOP angehaltenes Programm wieder fortgesetzt.

COS (X)

Der Kosinus von X wird berechnet.

CSAVE "<Name>"[,2]

Das Programm <Name> wird auf Kassette aufgezeichnet. Mit derOption 2 wird das Programm mit 2400 Baud aufgezeichnet.

CSNG(X)

Damit wird eine doppelt lange Zahl in eine einfach lange Zahl gewandelt.

```
100 A=12.34+1E-10
110 PRINT A
120 PRINT CSNG(A)
run
 12.3400000001
 12.34
```

CSRLIN

Eine Systemvariable, welche die Nummer der Zeile enthält, in welcher der Cursor sich augenblicklich befindet. Die oberste Zeile des Bildschirms hat die Nummer 1.

DATA C[,C]....

Diese Anweisung kennzeichnet eine Zeile, daß sie Daten enthält. Dies können Zahlen, Zeichen oder Zeichenketten sein. Durch eine READ Anweisung werden die Daten gelesen und den entsprechenden Variablen zugewiesen.

Mit dieser Anweisung können eigene Funktionen definiert werden.

Beispiel:

```
100 DEF FN A(X,Y)=X*X+Y*Y
110 PRINT FN A(2,3)
run
 13
Ok
```

DEFtype Buchstabe [-Buchstabe]

[,Buchstabe [-Buchstabe]]

Mit dieser Definition werden Variablen, die mit einem festgelegten Buchstaben beginnen, einem Datentyp zugeordnet. Dies sind

INT Ganzzahl
SNG einfachlange Zahl
DBL doppeltlange Zahl
STR Zeichenkette.

Mit

```
DEFINT I-K
```

sind alle Variablen, die mit den Buchstaben I, J und K beginnen, Ganzzahlvariable.

DEFUSR[I]=AADR

Mit dieser Anweisung wird die Anfangsadresse eines Maschinenprogramms festgelegt, daß mit USR aufgerufen wird. Wird die Angabe I (0-9) weggelassen, so wird 0 eingesetzt. Das Programm wird mit A=USRÄIÜ aufgerufen.

DELETE I[,J]

Mit dieser Anweisung werden die Programmzeilen I bis J gelöscht.

DIM F[I[,J]...)

Mit dem DIM Befehl werden Felder (Zahlen oder Zeichenketten) dimensioniert.

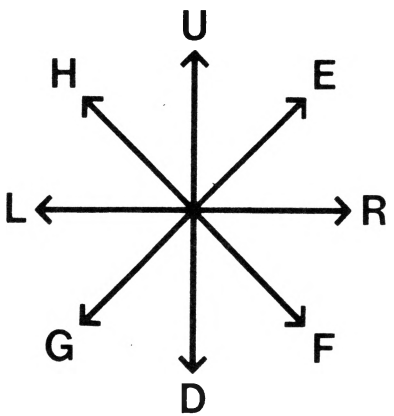
Beispiel:

```
DIM F(3,4)
```

dimensioniert eine zweidimensionale Matrix mit 3 Zeilen und 4 Spalten.

DRAW "<Zeichenkette>"

In der Zeichenkette werden die Richtung und die Länge der Linie angegeben. Die unten stehende Abbildung zeigt die Abkürzungen.



Das folgende Beispiel zeichnet einen Rhombus.

```
100 SCREEN 2
110 PSET(50,100),15
120 DRAW"E50F50G50H50"
130 GOTO 130
```

END

Diese Anweisung kennzeichnet das Ende eines Programms.

EOF(I)

Diese Funktion ist "TRUE", wenn das Ende einer Datei auf Kassette erreicht ist. Dabei ist I die Dateinummer der Datei, die gerade eingelesen wird.

ERASE Feldname [,Feldname]...

Im Gegensatz zu CLEAR löscht ERASE nur Felder. Diese Anweisung kann wie CLEAR für die dynamische Dimensionierung von Feldern verwendet werden. Dabei wird aber der Inhalt von Variablen nicht verändert. Beispiel wie mit CLEAR.

ERR
ERL

Diese beiden Systemvariablen enthalten die Nummer (ERR) und die Zeilennummer (ERL), wenn im Programm ein Fehler auftritt.

ERROR I

I ist eine Ganzzahl im Bereich 0 bis 255. Diese Zahl kennzeichnet die Fehlermeldung, welche beim Auftreten eines Fehlers ausgegeben wird.

Die Eingabe von

```
ERROR 1
```

bringt die Fehlermeldung

```
NEXT without FOR.
```

Die ERROR Anweisung kann auch zur Ausgabe eigener Fehlermeldungen verwendet werden. Dazu folgendes Beispiel: Bei einer Eingabe soll die Fehlermeldung "Keine negative Zahl eingeben" ausgegeben werden, wenn dies der Fall ist.

```
100 ON ERROR GOTO 200
110 INPUT "N=";N
120 IF N<0 THEN ERROR 210
200 IF ERR=210 THEN PRINT"KEINE NEGATIVE ZAHL EI
NGEBEN"
210 IF ERL=120 THEN RESUME 110
```

EXP(X)

Diese Funktion berechnet den Wert e^X. X muß kleiner als 145.06266 sein.

FIX(X)

Diese Funktion schneidet die Stellen hinter dem Dezimalpunkt ab.

```
X=123456.12
Ok
PRINT  X,FIX(X)
123456.12  123456 Ok
X=-123.45
Ok
PRINT FIX(X),INT(X)
-123 -124 Ok
```

Die zweite Eingabe zeigt den Unterschied zwischen der Funktion FIX(X) und INT(X). Die Funktion INT(X) bringt die nächst kleinere ganze Zahl.

FOR...NEXT...STEP

Mit diesen Anweisungen werden in einem Programm Schleifen gebildet. Diese haben folgende Form:

```
FOR <Variable>=X TO Y [STEP Z]
.
.
.     <Anweisungen>
.
.
NEXT <Variable>
```

Die Anweisungen, die sich zwischen FOR und NEXT befinden, werden mit den Werten von X bis Y berechnet. Ist STEP Z nicht angegeben, so wird als Schrittweite 1 angenommen. Innerhalb einer Schleife kann eine weitere Schleife programmiert werden. Die Schleifen können geschachtelt werden.

Eine Schleife wird immer mindestens einmal durchlaufen, da die Abbruchbedingung am Ende mit NEXT geprüft wird.

FRE(X)
FRE(X$)

Mit PRINT FRE(x) wird der freie Speicherraum, der für BASIC zur Verfügeung steht, ausgegeben, während PRINT FRE(X$) den freien Speicherplatz für Zeichenketten ausgibt.

```
NEW
PRINT FRE(X)
28815
Ok
PRINT FRE(X$)
200
Ok
A$="TRALALA":PRINT FRE(X$)
193
Ok
```

GOSUB...RETURN

Mit der Anweisung GOSUB erfolgt ein Sprung in ein Unterprogramm. Die Rückkehr in das Hauptprogramm wird mit der Anweisung RETURN ausgeführt. Im Gegensatz zu anderen BASIC-Dialekten kann in MSX-BASIC nach RETURN eine Zeilennummer angegeben werden. Damit wird das Programm nicht in der üblichen Weise fortgesetzt, sondern es erfolgt ein unbedingter Sprung zu dieser Zeile.

Dazu folgendes Beispiel:

```
100 PRINT "HALLO"
110 GOSUB 200
120 GOSUB 300
130 END
200 PRINT "HIER BIN ICH"
210 RETURN
300 PRINT "ENDE"
310 RETURN
```

Ohne Zeilennummer nach RETURN wird das Programm in der herkömmlichen Weise durchlaufen. Es wird der Text

```
HALLO
HIER BIN ICH
ENDE
```

ausgegeben.

Wird die Zeile 210 durch

```
210 RETURN 100
```

ersetzt, so entsteht eine endlose Programmschleife, die den Text

```
HALLO
HIER BIN ICH
```

ausgibt.

GOTO I

Sprungbefehl. Das Programm wird bei der Zeilennummer I weitergeführt.

HEX$(X)

Die Funktion HEX$(X) wandelt die Variable X in eine Hexadezimalzahl um. Eine Hexadezimalzahl besteht aus den Zahlen 0 bis 9 und den Buchstaben A bis F.

```
PRINT HEX$(1023)
3FF
Ok
```

Ist X eine Gleitkommazahl, so wird nur der ganzzahlige Teil gewandelt.

IF...THEN...ELSE

Diese Anweisung hat die Form

IF <Bedingung> THEN <Anweisungen A> [ELSE <Anweisungen B>]

Ist die Bedingung erfüllt, so werden die Anweisungen A ausgeführt. Danach wird das Programm mit der nächsten Zeilennummer fortgesetzt. Ist die Bedingung nicht erfüllt, so werden die Anweisungen B nach ELSE durchlaufen. Das Programm wird ebenfalls mit der nächsten Zeilennummer fortgesetzt.

In dieser Anweisung kann der Teil mit ELSE entfallen.

Beispiel für IF...THEN...ELSE...

```
100 INPUT "A=";A
110
IF A>=0 THEN PRINT "POSITIV" ELSE PRINT "NEGATIV"
120 INPUT"NOCHMAL ( /N)";A$
130 IF A$<>"N" GOTO 100 ELSE END
```

Ist in Zeile 110 die Bedingung A>=0 erfüllt, so wird POSITIV ausgedruckt. Wie Zeile 130 zeigt, kann das THEN auch durch GOTO ersetzt werden. Andererseits braucht man das GOTO nach THEN nicht unbedingt angeben. Die Zeile 130 kann genauso

```
130 IF A$<>"N" THEN 100 ELSE END
```

lauten.

INKEY$

Mit der Funktion INKEY$ wird das Tastenfeld abgefragt, ob eine Taste gedrückt wird. Als Funktionswert wird ein "leeres Zeichen" oder das ASCII Zeichen der gedrückten Taste übergeben.

```
100 PRINT"BITTE EINE TASTE DRUECKEN"
110 IF INKEY$="" THEN 110
120 PRINT"JETZT GEHTS WEITER"
```

Soll die Tasteneingabe zu einer Programmverzweigung führen, so muß der Wert von INKEY$ einer Variablen zugewiesen werden.

```
100 PRINT"BITTE EINE TASTE DRUECKEN"
110 A$=INKEY$:IF A$="" THEN 110
120 IF ASC(A$)=13 THEN 100
```

Wird jetzt in Zeile 110 die RETURN Taste gedrückt, so beginnt das Programm wieder in Zeile 100

INP(I)

Mit der Funktion INP wird der Inhalt eines Tores übernommen. Dabei ist I eine Adresse zwischen 0 und 255.

INPUT ["<Hinweis>";] X [,Variable]...
INPUT ["<Hinweis>";] A$ [,Zeichenkette]..

Mit der Funktion INPUT kann eine Zahl oder eine Zeichenkette vom Tastenfeld eingelesen und einer Variablen zugewiesen werden. Die Eingabe wird mit der RETURN Taste abgeschlossen. Vor der Eingabe kann noch ein Hinweis ausgegeben werden, welcher die Art der Eingabe kennzeichnet. Gleichzeitig wird vom Rechner noch ein ? ausgegeben.

```
100 INPUT "LAENGE DER SEITE ";L
```

Bei der Eingabe einer Zeichenkette muß folgende Besonderheit beachtet werden. Ist einer Zeichenkette ein Wert zugewiesen worden und wird danach ein neuer Wert für diese Zeichenkette eingegeben, so bleibt bei einer "leeren" Eingabe (nur RETURN Taste) der Inhalt der

Zeichenkette erhalten. Bei anderen BASIC Dialekten wird in diesem Fall der Inhalt gelöscht. Dies gilt nicht nur für die Eingabe von Zeichenketten, sondern auch für die Eingabe von Zahlen.

```
100 A$="HAHA"
110 INPUT A$
120 PRINT A$
run
?
HAHA
Ok
```

Nach dem Fragezeichen wurde nur die RETURN Taste betätigt. Sollen mit der INPUT Anweisung mehrere Variable eingegeben werden, so werden diese, durch Komma getrennt, eingegeben.

INPUT$(I[,#N])

Mit der Funktion INPUT$ werden I Zeichen vom Tastenfeld oder von einer Datei mit der Nummer N eingelesen.

```
100 PRINT " Mit W Weiter, mit E Ende des Programms"
110 A$=INPUT$(1)
120 IF A$="E" THEN END
130 IF A$="W" THEN GOTO 200
140 GOTO 100
200 REM HIER GEHTS WEITER
```

Mit der INPUT$ Funktion können auch Steuerzeichen, die mit der INPUT Funktion nicht gelesen werden, in den Rechner eingegeben werden. Deshalb sollte die INPUT$ Funktion vor allem bei Datenübertragungen von anderen Rechnern verwendet werden. Bei einer solchen Übertragung können alle Zeichen von Bedeutung sein.

INPUT#I,VARIABLE[,VARIABLE]...

Die INPUT# Anweisung liest Daten von der sequentiellen Datei I und weist sie den in der Variablenliste aufgeführten Variablen zu.

INSTR([I,]X$,Y$)

Die INSTR Funktion sucht das erste Auftreten der Zeichenkette Y$ in der Zeichenkette X$ und gibt die Anfangsposition aus. Es kann eine Ganzzahl I (0-255) zusätzlich mitgegeben werden. Dann beginnt die Suche ab dieser Position in der Zeichenkette X$. Wenn Y$ nicht gefunden wird, wenn I größer als die Länge von X$ oder wenn X$ leer ist, so wird von INSTR Null übergeben. Ist Y$ leer, so wird 1 oder I übergeben.

```
100 A$="Froehliche Weihnachten"
110 PRINT INSTR(A$,"acht")
120 PRINT INSTR(A$,"ACHT")
130 PRINT INSTR(5,A$,"e")
run
17
0
10
Ok
```

INT(X)

Diese Funktion bildet aus der Gleitkommazahl X eine ganze Zahl. Dabei wird aber immer nach der nächst kleineren Zahl abgerundet.

```
PRINT INT (123.45)
 123
Ok

PRINT INT (-123.45)
-124
Ok
```

INTERVAL ON
INTERVAL OFF
INTERVAL STOP
ON INTERVAL=X GOSUB ZNR

Die Anweisung INTERVAL wird durch einen internen Zeitgeber gesteuert. Durch einen Hardware Interrupt wird der Inhalt einer Zelle (2 Bytes) alle 1/50 Sekunden um Eins erhöht. Das folgende Beispielprogramm zeigt die Verwendung dieses Befehls. Alle 5 Sekunden wird der Inhalt der Timerzelle ausgedruckt.

```
100 ON INTERVAL=250 GOSUB 200
110 INTERVAL ON
120 GOTO 120
200 LPRINT TIME:RETURN
 24511
 24761
 25011
 25261
 25511
 25761
 26011
```

Damit ist auch ein einfaches "Multiprogramming" möglich. Im folgenden Beispiel wird im oberen linken Bildschirmrand die Zeit eingeblendet, wie lange das Programm ab Zeile 1000 schon läuft. Die Zeile 120 ist durch

```
120 GOTO 1000
```

zu ersetzen. Alle 10 Sekunden wird die Zeitangabe eingeblendet.

```
 90 GOSUB 250
100 ON INTERVAL=500 GOSUB 200
110 INTERVAL ON
120 GOTO 120
200 SEC=SEC+10
210 IF SEC=60 THEN SEC=0:MIN=MIN+1
220 IF MIN=60 THEN MIN=0:HR=HR+1
230 IF HR=24 THEN HR=0
250 LOCATE 25,0:PRINT USING "##:";HR,MIN,SEC
260 RETURN
```

KEY ON
KEY OFF
KEY LIST

Mit KEY ON wird die Belegung der Funktionstasten mit Text am unteren Bildschirmrand angezeigt. Diese Zeile wird mit KEY OFF ausgeschaltet. Die Textbelegung wird mit KEY LIST auf den Bildschirm ausgegeben.

Der Text für eine Funktionstaste wird mit

```
KEY I,"<Text>"
```

geändert. Beispiel:

```
KEY 1,"Csave"+CHR$(34)
```

belegt die Funktionstaste 1 mit dem Text Csave". Soll bei Betätigung der Taste ein RETURN ausgeben werden, so muß +CHR$(13) an den Text angehängt werden.

KEY (I) ON
KEY (I) OFF
ON KEY GOSUB ZNR[,ZNR]...

Mit diesen Anweisungen werden die Funktionstasten zur Programmverzweigung verwendet. Mit dem Verteiler ON KEY GOSUB werden nach Betätigen einer Funktionstaste die entsprechenden Unterprogramme angesprungen.

Beispiel:

```
100 KEY OFF
110 ON KEY GOSUB 200,300,400
120 KEY(1) ON:KEY(2) ON:KEY(3) ON
122 GOTO 130
125 PRINT "GEDRUECKT"
130 GOTO 130
200 PRINT "F1 ":: RETURN 125
```

```
300 PRINT "F2 ":: RETURN 125
400 PRINT "F3 ":: RETURN 125
500 ON ERROR GOTO 130
```

Mit den Funktionstasten F1, F2 und F3 wird in die Programmzeilen 200,300 und 400 gesprungen. In der Zeile 130 wartet das Programm auf das Drücken einer der drei Funktionstasten. Diese sind in Zeile 120 eingeschaltet worden. Die Rückkehr aus den Unterprogrammen geschieht einheitlich nach Zeile 125. Dies ist ein Beispiel für ein RETURN mit Zeilennummer.

LEFT$(A$,I)

Die Funktion LEFT$ übergibt die I am weitesten links stehenden Zeichen. Beispiel:

```
1000 A$="MSX-COMPUTER"
1010 FOR I=1 TO LEN(A$)
1020 PRINT LEFT$(A$,I)
1030 NEXT I
M
MS
MSX
MSX-
MSX-C
MSX-CO
MSX-COM
MSX-COMP
MSX-COMPU
MSX-COMPUT
MSX-COMPUTE
MSX-COMPUTER
```

LEN(A$)

Die Funktion LEN übergibt die Länge der Zeichenkette A$.

```
100 A$="MSX-COMPUTER"
110 PRINT LEN(A$)
run
 12
Ok
```

[LET] X=Y

Mit LET wird der Variablen X der Wert Y zugewiesen. Die Anweisung selbst kann weggelassen werden.

LINE [(X1,Y1)]-[STEP](X2,Y2)[,I[,B[F]]]

Die Funktion LINE zeichnet eine Linie. X1, X2, Y1 und Y2 sind dabei die absoluten oder relativen Koordinaten der Bildpunkte auf dem Bildschirm. Die Anweisung

```
LINE (125,10)-(75,75)
```

zeichnet eine Linie von X1=125, Y1=10 nach X2=75 und Y2=75. In dieser Form sind die angegebenen Werte absolute Koordinaten. Mit

```
LINE (125,10)-STEP(75,75)
```

wird eine Linie von X1=125, Y1=10 nach X2=200 und Y2=85 gezogen. In diesem Fall sind die Angaben X2,Y2 relative Koordinaten. Mit

```
LINE -(75,75)
```

wird eine Linie vom augenblicklichen Bildpunkt zum Punkt 75, 75 gezeichnet.

Folgende Optionen können verwendet werden. I ist die Nummer der Farbe, mit welcher die Linie gezeichnet wird. Mit der Option B wird ein Rechteck gezeichnet. Dabei bezeichnen die Koordinaten X2,Y2 die Ecke, die X1, Y1 diagonal gegenüber liegt. Ist schließlich noch die Option F angegeben, so wird das Rechteck mit der angegebenen Farbe ausgefüllt.

```
LINE (125,10)-STEP(75,75),15,BF
```

zeichnet ein Rechteck in Weiß.

```
1000 SCREEN 2
1010 LINE(125,10)-STEP(75,75),15,BF
1020 GOTO 1020
```

```
1000 COLOR 15,1,1:SCREEN 2
1010 FOR F%=1 TO 100
1020 LINE (RND(1)*250,RND(1)*190)-
     (RND(1)*250,RND(1)*190),RND(1)*14+2,BF
1030 NEXT F%
```

LINE INPUT [<"Hinweis">;] A$

Mit LINE INPUT wird eine ganze Zeile, bis zu 254 Zeichen vom Tastenfeld übernommen. Ist ein Hinweis angegeben, so wird dieser vor der Eingabe auf dem Bildschirm angezeigt.

LINE INPUT #I,A$

Mit dieser Aweisung werden solange Zeichen über den Kanal I eingelesen und in A$ gespeichert, bis ein RETURN Zeichen (13) erkannt wird.

LIST [[ZNR1][-[ZNR2]]]

Mit der Anweisung LIST werden die Zeilen eines Programms auf den Bildschirm ausgegeben. Dabei bestehen folgende Möglichkeiten:

LIST	Ausgabe aller Zeilen.
LIST ZNR	Ausgabe nur der Zeile ZNR
LIST ZNR1-ZNR2	Ausgabe aller Zeilen von ZNR1 bis ZNR2.
LIST -ZNR	Ausgabe aller Zeilen vom Anfang bis ZNR.
LIST ZNR-	Ausgabe aller Zeilen von Zeile ZNR bis zum Ende des Programms.

LLIST [[ZNR1][-[ZNR2]]]

LLIST hat die gleiche Bedeutung wie LIST, das Ausgabegerät ist jedoch der Drucker.

LOAD "<Datei>"

Mit dieser Anweisung werden Dateien von Diskette gelesen. Dabei hat <Datei> den folgenden Aufbau:

<Datei> = ÄGÜ<Name>Ä.TypÜ

Hierbei ist

G	die Bezeichnung des Laufwerks,
<Nane>	der Name der Datei mit maximal 8 Zeichen und
Typ	eine Kennzeichnung der Datei mit maximal 3 Zeichen.

Beispiel: LOAD "A:TEST.BAS"

Anmerkung:
Das Laden einer Datei von Kassette mit LOAD "CAS:<Name>" ist nicht möglich.

LOCATE H,V[,I]

Diese Anweisung plaziert den Cursor an der Stelle V,H. Dies geschieht jedoch erst bei der nächsten PRINT Anweisung. V ist die vertikale, H ist die horizontale Position des Cursors. Dabei kann V die Werte 0 bis 22 und H die Werte 0 bis 39 (mit WIDTH=40) annehmen. Für die linke obere Ecke gelten die Angaben V=0 und H=0.

Mit der Option I kann der Cursor ein- oder ausgeschaltet werden. Normalerweise ist der Cursor während des Progamm-ablaufs nur bei Eingaben sichtbar. Mit

```
LOCATE,,1
```

kann dieser jedoch eingeschaltet werden. Dies zeigt das folgende Programm. Es zeigt ebenfalls, wie in die Zeile 23 geschrieben werden kann.

```
100 KEY OFF
110 LOCATE 1,23
120 PRINT "A"
130 LOCATE ,,1
140 GOTO 140
```

Es wird in die Zeile 23 in die erste Stelle ein A geschrieben. Der Cursor ist in der nächsten Stelle sichtbar.

LOG (X)

Diese Funktion berechnet den natürlichen Logarithmus (zur Basis e=2.718281826...).

```
PRINT LOG(10)
 2.302585092994
Ok
PRINT LOG(2.718281826)
 .99999999909531
Ok
```

LPRINT [V] [;]
LPRINT USING "<Format>"; V [;]

Ausgabe der Daten V auf den Drucker. LPRINT USING Beschreibung siehe PRINT USING.

MAXFILES=I

Durch diese Anweisung wird festgelegt, wieviele Dateien maximal gleichzeitig eröffnet werden können.

MERGE "<Name>"

Das Programm <Name> wird mit dem im Speicher befindlichen Programm gemischt. Dazu muß das Programm <Name> als ASCII Datei mit

```
SAVE "CAS:<Name>",A
```

auf Kassette geschrieben worden sein.

```
100 ' MERGE DEMO
120 PRINT"EINS"
140 PRINT"DREI"
160 PRINT"FUENF"
```

```
110 ' MERGE DEMO
130 PRINT"ZWEI"
150 PRINT"VIER"
170 END
```

```
100 ' MERGE DEMO
110 ' MERGE DEMO
120 PRINT"EINS"
130 PRINT"ZWEI"
140 PRINT"DREI"
150 PRINT"VIER"
160 PRINT"FUENF"
170 END
```

MID$(A$,I,J)

Die Funktion MID$ übergibt eine Zeichenkette, welche bei I beginnt und J Zeichen lang ist.

Beispiel:

```
100 A$="MSX-COMPUTER"
110 B$=MID$(A$,3,4)
120 PRINT B$
run
X-CO
Ok
```

MID$(A$,I[,J])=Y$

Ein Teil der Zeichen von A$ werden ab der Stelle I durch J Zeichen der Zeichenkette Y$ ersetzt. Ist J nicht angegeben, so werden alle Zeichen von Y$ eingesetzt.

Beispiel:

```
100 A$="MSX-COMPUTER"
105 Y$="TEXTER"
110 MID$(A$,3,4)=Y$
120 PRINT A$
run
MSTEXTMPUTER
```

I MOD J

Diese Funktion gibt den Rest an, der bei der Ganzzahl-Division I/J entsteht.

Beispiel:

```
PRINT 5 MOD 3
 2
Ok

PRINT 5 MOD 5
 0
Ok
```

MOTOR ON
MOTOR OFF

Schaltet den Kassettenrekorder ein oder aus.

NEW

Diese Anweisung löscht ein Programm im Speicher.

NOT

NOT ist die logische Umkehrfunktion NICHT.

Beispiel:

```
100 INPUT "J/N";A$
110 IF NOT A$="N" THEN PRINT "JA" ELSE
    PRINT "NEIN"

run
(J/N)?N
NEIN
```

OCT$ (I)

Diese Funktion wandelt die Dezimalzahl I in eine Oktalzahl.

```
PRINT OCT$(1000)
 1750
Ok
```

Anmerkung: Eine Oktalzahl ist eine Zahl im 8-System. Als Ziffern kommen nur die Zahlen 0 bis 7 vor.

ON ERROR GOTO ZNR

Tritt nach dieser Anweisung ein Fehler auf, so wird das Programm bei der Zeilennummer ZNR fortgesetzt.

ON I GOTO ZNR1,ZNR2[,ZNR3]...
ON I GOSUB ZNR1,ZNR2,[ZNR3]...

Dies ist ein Verteiler. Wenn I gleich 1 ist, so wird ZNR1 gesprungen oder das bei ZNR1 beginnende Unterprogramm aufgerufen. Ist I gleich 2, so wird ZNR2 angesprungen. Wenn I gleich 0 oder größer als die Zahl der Verteiler ist, wird der nächste ausführbare Befehl ausgeführt.

OPEN "<Geraet:Name>" [FOR <Richtung>] AS #I

Durch diese Anweisung wird ein Ein- bzw Ausgabekanal vom oder zum <Gerät> eröffnet. Dort werden Daten unter dem Dateinamen <Name> abgelegt. Die Richtung des Datentransfers wird in <Richtung> angegeben.

Beispiel:

```
OPEN "CAS:TEST" FOR OUTPUT AS #1
PRINT #1,X
```

Auf die Kassette wird, unter dem Namen TEST, der Wert von X über den Kanal 1 aufgezeichnet. Von dort kann er zum Beispiel mit

```
OPEN "CAS:TEST" FOR INPUT AS #2
INPUT #2,X
```

wieder gelesen werden.

Am Ende eines Programms, oder wenn die Dateien nicht mehr gebraucht werden, sollen alle eröffneten Kanäle mit CLOSE geschlossen werden.

OUT I,J

Ausgabe der Zahl J (0-255) an das Tor mit der Adresse I

PAINT [I,J][,F][,G]

Sind I,J die Koordinaten eines Punktes innerhalb einer gezeichneten Figur, so wird diese mit der Vordergrundfarbe ausgefüllt. Hat der Startpunkt diese Farbe, so wird die Figur nicht ausgefüllt. Wird zusätzlich eine Farbzahl F und die Farbe der Umrandung G angegeben, so wird die Figur, die diese Farbe als Rand hat, eingefärbt.

```
100 SCREEN 2
110 LINE (105,10)-(75,75),B
120 PAINT (90,60)
130 GOTO 130
```

PEEK (ADR)

Mit PEEK (ADR) wird der Inhalt eines Bytes mit der Adresse ADR übergeben. Der Wert liegt zwischen 0 und 255.

PLAY "<Zeichenkette>"

Mit dieser Funktion werden die in der Zeichenkette angegebenen Noten gespielt. Dabei können Tonlage, Tondauer und Pausen durch Unterkommandos programmiert werden. Der folgende Befehl spielt die C-Dur Tonleiter.

```
PLAY"O5CDEFGABO6C"
```

POINT (I,J)

Die Funktion POINT übergibt die augenblickliche Farbe des Punktes mit den Koordinaten I,J.

Beispiel siehe PSET,PRESET

POS (I)

Diese Funktion übergibt die augenblickliche Stellung des Cursors.

```
100 LOCATE 10,10: PRINT "A";
110 PRINT POS(I)
```

Die zweite PRINT-Anweisung liefert als Ergebnis 11.

PRESET[I,J]
PESET(I,J)[,F]

Die Funktion PESET setzt einen Punkt mit den Koordinaten I,J und der Farbe F. Ist F nicht angegeben, so wird der Punkt in der Vordergrundfarbe gemalt. Die Funktion PRESET löscht einen Punkt an den Koordinaten I,J.

```
100 SCREEN 2
110 LINE (105,10)-(75,75),15,B
120 PAINT(90,60)
125 IF POINT (90,60)<>0 THEN PRESET (90,60)
130 GOTO 130
140 END
```

Durch die Anweisung in Zeile 125 wird die Farbe des Punktes (90,60) zurückgesetzt.

PRINT [X[,X]]... [;]

Mit PRINT werden die Werte X auf den Bildschirm ausgegeben. Wird PRINT ohne weitere Angaben verwendet, so wird eine Leerzeile ausgegeben. Bei positiven Zahlen wird statt des Vorzeichens ein Leerzeichen ausgedruckt. Nach einer Zahl wird ein Leerzeichen ausgegeben. Sind die Variablen durch Komma getrennt, so werden die Zahlen an der nächsten Druckposition ausgegeben. Eine Druckposition ist 14 Leerzeichen breit. Sind die Variablen durch Strichpunkte getrennt, so werden sie unmittelbar hintereinander gedruckt. Wird eine PRINT Anweisung mit einem Strichpunkt beendet, so wird an dieser Druckposition mit der nächsten PRINT Anweisung weitergefahren. Andernfalls wird ein Wagenrücklauf, Zeilenvorschub ausgegeben.

```
100 A=-10:B=30:A$="SSSS"
110 PRINT A,B,A$
run

        -10   30.123 SSSS

110 PRINT A;B;A$
run

  -10              30.123        SSSS
```

Statt dem Wort PRINT kann für die Ausgabe auch das Fragezeichen ? verwendet werden.

```
110 ? A;B;A$
```

PRINT USING V$; X[,X]... [;]

Mit PRINT USING können Werte in einem vorgegebenen Format gedruckt werden. V$ ist eine Zeichenkette, welche spezielle Steuerzeichen zum Drucken enthält. Bei der Ausgabe von Text können drei Steuerzeichen verwendet werden.

!	Nur das erste Zeichen wird gedruckt
\ n Leerzeichen \	n+2 Zeichen werden gedruckt.
&	Der Text wird ohne Veränderung gedruckt.

Beispiele:

```
100 A$="GUTEN":B$="TAG"
110 PRINT USING "!";B$;
120 PRINT USING "\ \";A$
130 PRINT USING "&";A$;" ";B$



TGUT
GUTEN TAG
```

Bei der Ausgabe von Zahlenwerten können folgende Steuerzeichen verwendet werden:

#	Gibt eine Zahlenstelle an. Diese Zahlenstellen werden immer gedruckt. Hat eine Zahl weniger Stellen als angegeben, so wird sie rechtsbündig gedruckt.
.	Ein Dezimalpunkt kann an beliebiger Stelle eingesetzt werden.
+	Ein Pluszeichen zu Beginn der Formatierung gibt an, daß die Vorzeichen immer mit ausgegeben werden.
-	Ein Minuszeichen am Ende der Formatierung gibt an, daß Minuszeichen am Ende der Zahl gedruckt werden.
**	Mit diesen Zeichen zu Beginn der Formatierung werden führende Nullen durch * ersetzt.

Beispiele:

```
PRINT USING "##.##";.99
 0.99

PRINT USING "###.##";123.456
 123.46

PRINT USING "##.##  ";10.2,5,4.55
 10.20  5.00   4.55

PRINT USING "+### ";-10,5,-46
 -10   +5  -46

PRINT USING "###- ";-10,5,-46
 10-   5   46-

PRINT USING "**#####.## ";-10,5,-46
***-10.00 *****5.00 ***-46.00
```

$$	Mit diesem Zeichen vor der Formatierung wird ein $ Zeichen vor die Zahl gesetzt
**$	Führende Nullen werden durch * ersetzt und es wird ein $ Zeichen vor der Zahl ausgegeben.
,	Ein Komma links neben dem Dezimalpunkt gibt an, daß nach jeder 3. Stelle links vom Dezimalpunkt ein, eingesetzt wird.
^	Die Zahl wird in exponentieller Schreibweise ausgegeben.

Beispiele:

```
PRINT USING "$$###.##";123.456
 $123.46

PRINT USING "#####,.##";1234567
%1,234,567.00
```

Ein Prozentzeichen vor der Zahl gibt an, daß die Zahl größer als das angegebene Format ist.

```
PRINT USING "##.##^^^^";234.56
 2.35E+02
```

PRINT #I
PRINT USING #I

Datenausgabe über den Kanal I.

PUT SPRITE P,(I,J),F,N

Diese Anweisung setzt den SPRITE mit der Nummer N und der Farbe F an die Koordinaten I,J des Bildschirms. Die Priorität ist eine Zahl zwischen 0 und 31. Befinden sich zwei SPRITES an der gleichen Position, so der SPRITE mit der höheren Priorität sichtbar.

```
1000 COLOR 15,1,1:SCREEN 2,3
1010 S$="":FOR F%=0 TO 31:READ A$
1020 S$=S$+CHR$(VAL("&H"+A$)):NEXT F%
1030 SPRITE$(0)=S$
1040 DATA 0,1,6,1D,2A,2A,2A,1F,4C,F7
1045 DATA F0,18,7,2,3E,FE,18,8F,65
1047 DATA 11,C9,A9,81
1050 DATA F3,7F,9F,31,41,81,81,F9,FD
1060 X%=128:Y%=96
1070 XS%=1:YS%=1
1080 PUT SPRITE 0,(X%,Y%),13
1090 X%=X%+XS%:Y%=Y%+YS%
1100 IF X%<0 OR X%>210 THEN XS%=-XS%
1110 IF Y%<0 OR Y%>156 THEN YS%=-YS%
1120 GOTO 1080
```

In diesem Beispiel wird gezeigt, wie ein SPRITE mit PUT bewegt wird und an den Rändern reflektiert wird.

READ V[,V]...
RESTORE

Mit READ werden Daten, die in DATA Anweisungen programmiert sind, eingelesen und den Variablen V zugewiesen. Bei jeder READ Anweisung wird ein Zeiger auf den nächsten Datenwert gesetzt. Sollen mehr Daten gelesen werden, als vorhanden sind, so erfogt die Fehlermeldung "OUT OF DATA ERROR". Sollen die Daten ein zweites Mal gelesen werden, so ist dieser Zeiger mit RESTORE zurückzusetzen.

REM

Mit der REMark Anweisung können Bemerkungen zum Programm eingefügt werden. Diese Zeilen werden bei der Programmausführung übergangen. REM kann durch ' ersetzt werden.

RENUM [ZNR1] [,ZNR2 [,INC]]

Ein Programm, bzw ein Teil eines Programms wird mit neuen Zeilenzahlen versehen. Werden keine Angaben gemacht, so beginnt die Zeilenzahl mit 10 und wird jedesmal um 10 (INC) erhöht. Es kann eine Anfangszeilenzahl ZNR1, eine Endzeilenzahl ZNR2 und eine Erhöhung der Zeilenzahl INC angegeben werden.

RESUME [OPT]

Mit dieser Anweisung kann ein Programm nach einem Fehler fortgesetzt werden. Mit ON ERROR GOTO wird ein Sprung in einen Programmteil durchgeführt, in welchem der Fehler untersucht wird. Kann das Programm trotz des Fehlers fortgesetzt werden, so geschieht dies mit RESUME.

RESUME RESUME 0	Programm wird mit der Zeile fortgesetzt, in welcher der Fehler auftrat.
RESUME ZNR	Programm wird mit der Zeile ZNR fortgesetzt.
RESUME NEXT	Programm wird mit der nächsten, auf den Fehler folgenden Anweisung, fortgesetzt.

RIGHT$ (A$,I)

Diese Funktion übergibt die I am weitesten rechts stehenden Zeichen aus.

Beispiel:

```
100 A$="MSX-COMPUTER"
110 FOR I=LEN(A$) TO 1 STEP -1
120 PRINT RIGHT$(A$,I)
130 NEXT I
```

```
MSX-COMPUTER
SX-COMPUTER
X-COMPUTER
-COMPUTER
COMPUTER
OMPUTER
MPUTER
PUTER
UTER
TER
ER
R
```

RND(I)

Mit RND werden Zufallszahlen zwischen 0 und 1 erzeugt.

I=1	Immer gleiche Folge von Zufallszahlen.
I=0	Immer gleiche Zufallszahl.
I=-K	Nächste Zufallszahl hängt von K ab.

Mit Z=INT(RND(1)*(N+1)) werden Zufallszahlen zwischen 0 und N erzeugt.

RUN [ZNR]

Mit RUN wird ein Programm gestartet. Ohne weitere Angaben wird das Programm mit der kleinsten Zeilennummer begonnen. Andernfalls wird das Programm bei der angegebenen Zeilennummer begonnen.

SAVE "<Datei>"

Speichern eines Programms auf Diskette. <Datei> hat die Form:

[G]<Name>[.Typ][,OPT]

G	die Bezeichnung des Laufwerks, CAS: Kassette CRT: Textbildschirm GRP: Grafilbildschirm LPT: Drucker
<Name>	der Name der Datei mit maximal 8 Zeichen und
Typ	eine Kennzeichnung der Datei mit maximal 3 Zeichen.
OPT	ist eine zusätzliche Option.

Als Option kann A angegeben werden. Dann wird das Programm nicht als übersetztes Basicprogramm sondern als Textdatei aufgezeichnet. So kann es mittels Telekommunikation zu anderen Rechnern übertragen werden.

Mit

```
SAVE "CAS:<Name>",A
```

kann ein Programm als Textdatei auch auf Kassette aufgezeichnet werden.

Mit SCREEN werden die verschiedenen Bildschirmdarstellungen ausgewählt.

SCREEN 0

Dies ist die erste von zwei Textarten. Das Format ist 24 Zeilen zu je 40 Zeichen. Jeder Buchstabe wird mit 6X8 Bildpunkten dargestellt. Intern sind die Buchstaben mit 8x8 Zeichen gespeichert. Hier werden jedoch bei der Darstellung rechts zwei Spalten abgeschnitten. Zwei Farben, eine für Vordergund und eine für Hintergrund, können gewählt werden.

SCREEN 1

Hier können in einer Reihe bis zu 32 Zeichen angezeigt werden. Beim Umschalten auf SCREEN 1 wird der Zeichensatz von ROM in RAM kopiert. Somit ist es möglich, eigene Zeichen zu definieren. Das folgende Programm zeigt alle möglichen Zeichen auf dem Bildschirm.

```
1000 COLOR 15,4,4:SCREEN 1
1010 FOR F%=0 TO 19
1020 FOR J%=0 TO 13
1030 IF F%*14+J%<256 THEN VPOKE &H1842+F%*32+
     J%*2,F%*14+J%
1040 NEXT J%,F%
```

SCREEN 2

Mit SCREEN 2 wird die hochauflösende Grafik ausgewählt. Der Bildschirm ist in 256x192 Bildpunkte aufgeteilt. Mit den Grafikbefehlen können Bilder gezeichnet werden.

SCREEN 3

In SCREEN 3 ist der Bildschirm genau wie in SCREEN 2 aufgeteilt. Hier wird jedoch ein Punkt aus 4x4 Bildpunkten

zusammengesetzt.

Mit der J Option wird die Größe der SPRITES bestimmt.

Hierbei ist

0=kleine SPRITES, Format 8x8
1=kleine SPRITES, vergrößert auf 16x16
2=große SPRITES, Format 16x16
3=große SPRITES, vergrößert auf 32x32

Mit der K Option wird die Erzeugung eines Tones bei der Betätigung einer Taste ein- oder ausgeschaltet.

0=kein Ton
X (beliebige Zahl)=Ton

Mit der L Option wird die BAUD-Zahl der Kassette vorgegeben.

1=1200 Baud
2=2400 Baud

Mit der M Option wird der Druckertyp angegeben.

1=MSX Drucker
X (beliebige Zahl)= Anderer Drucker.

SGN (X)

Die Signum Funktion übergibt eine 1, wenn X positiv und -1, wenn X negativ ist.

```
PRINT SGN(-2)
-1
Ok
PRINT SGN(2)
 1
Ok
```

SIN (X)

Mit SIN (X) wird der Sinuswert von X berechnet. Der Wert von X ist im Bogenmaß angegeben.

SOUND I,J

Mit dieser Anweisung werden die Register des PSG direkt beschrieben. Dabei ist I die Nummer des Registers und J eine Zahl (0-255) die in dieses Register geschrieben werden soll. Das folgende Programm lässt einen Ton in der Lautstärke anschwellen.

```
100 FOR N=0 TO 15
110 SOUND 0,&H18
120 SOUND 1,&H1
130 SOUND 7,&HFE
140 SOUND 8,N
150 SOUND 11,&H46
160 SOUND 12,&H0
170 SOUND 13,&H8
180 FOR J=1 TO 100
190 NEXT J,N
200 END
```

A$=SPACE$(I)

Mit SPACE$ werden der Variablen A$ I Leerzeichen zugewiesen.

```
100 A$=SPACE$(10):B$="*"
110 PRINT B$;A$;B$
run
*          *
```

SPC(I)

Die SPC Funktion fügt I Leerzeichen in einer PRINT Anweisung ein.

```
100 FOR I=1 TO 5
110 PRINT SPC(I);I
120 NEXT I
  1
   2
    3
     4
      5
```

SPRITE$(I)=A$
SPRITE ON
SPRITE OFF
SPRITE STOP
ON SPRITE GOSUB ZNR

Mit diesen Funktionen werden die SPRITES (wörtliche Übersetzung: ELFEN) über den Bildschirm bewegt. Mit

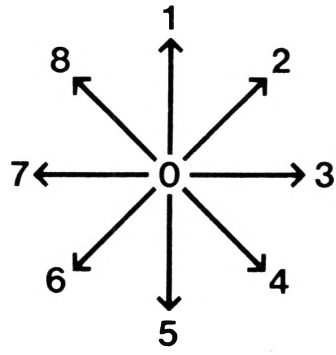
```
SPRITE (I)=A$
```

wird dem SPRITE I das in A$ gespeicherte Bitmuster zugewiesen. Bei einer Kollision von zwei SPRITES wird mit ON SPRITE GOSUB ZNR zur Zeilennummer ZNR gesprungen.

SQR (X)

Die Funktion SQR berechnet die Wurzel der Variablen X

Die Funktion gibt die Richtung an, in welcher ein Steuerknüppel zeigt. Die Zahlenwerte zeigt die folgende Abbildung.



I hat folgende Bedeutung:

0 = Die Cursortasten werden als Joystick verwendet.
1 = Joystick in Anschluß 1
2 = Joystick in Anschluß 2

Mit dem folgenden Programm kann mit den Cursortasten oder mit einem Joystick ein laufender Punkt über den Bildschirm gesteuert werden.

```
1000 COLOR 15,1,1
1010 SCREEN 2
1020 X%=128:Y%=96
1030 PSET (X%,Y%)
1040 C%=STRIG(0)
1050 IF C%<0 THEN CLS
1060 M%=STICK(0)
1070 IF M%=0 THEN 1040
1080 IF M%=1 THEN Y%=Y%-1
1090 IF M%=2 THEN Y%=Y%-1:X%=X%+1
1100 IF M%=3 THEN X%=X%+1
1110 IF M%=4 THEN Y%=Y%+1:X%=X%+1
1120 IF M%=5 THEN Y%=Y%+1
1130 IF M%=6 THEN Y%=Y%+1:X%=X%-1
1140 IF M%=7 THEN X%=X%-1
1150 IF M%=8 THEN Y%=Y%-1:X%=X%-1
1160 GOTO 1030
```

STOP

Die Anweisung STOP hält ein Programm an. Es kann mit CONT, wenn keine Änderungen gemacht wurden, fortgesetzt werden. STOP wird hauptsächlich bei der Fehlersuche verwendet.

STOP ON
STOP OFF
STOP STOP
ON STOP GOSUB ZNR

Mit diesen Anweisungen kann die STOP Taste zu Programmverzweigung benutzt werden.

```
100 ON STOP GOSUB 200
110 STOP ON
120 GOTO 120
200 PRINT"HALT..HALT...":RETURN
```

Damit ist aber die ^STOP Taste nicht mehr zum Programmabbruch zu verwenden. Obiges Programm kann nur durch Ausschalten des Rechners abgebrochen werden.

STRIG (I)
STRIG ON
STRIG OFF
STRIG STOP
ON STRIG GOSUB ZNR

Mit diesen Anweisungen wird die Aktionstaste des Steuerknüppels programmiert. Ist die Taste gedrückt, so ist der Wert von STRIG gleich -1, andernfalls 0. Mit I=0 wird die Leertaste als Aktionstaste verwendet.

STR$(I)

Diese Funktion wandelt den Zahlenwert I (0 bis 255) in das entsprechende ASCII Zeichen.

```
100 FOR I=40 TO 50
110 PRINT CHR$(I);" ";
120 NEXT I
( ) * + , - . / 0 1 2
```

A$=STRING$(I,J)

Die Zeichenvariable A$ enthält I mal das ASCII Zeichen mit dem Dezimalwert J.

```
100 A$=STRING$(10,45)
110 B$="BERICHT"
120 PRINT A$;B$;A$

----------BERICHT----------
```

SWAP X,Y

SWAP vertauscht die Werte von zwei Variablen.

```
100 A$="PETER":B$="LIESE"
110 PRINT A$;" LIEBT ";B$
120 SWAP A$,B$
130 GOTO 110

PETER LIEBT LIESE
LIESE LIEBT PETER
PETER LIEBT LIESE
LIESE LIEBT PETER
PETER LIEBT LIESE
```

TAB (I)

Innerhalb einer PRINT Anweisung wird der Cursor auf die absolute Position I gesetzt.

```
90 ON ERROR GOTO 300
100 PRINT"NAME";TAB(15);"NUMMER"
110 READ A$,B$
120 PRINT A$;TAB(15);B$
130 GOTO 110
200 DATA "HILLRAINER","089/12345
210 DATA "PETERS","0711/65432
300 END
```

```
NAME            NUMMER
HILLRAINER      089/12345
PETERS          0711/65432
```

TAN(X)

Der Tangens von X wird berechnet. Der Wert von X muß im Bogenmaß angegeben werden.

TIME

TIME ist eine Variable, deren Inhalt alle 1/50 tel Sekunde erhöht wird. Sie kann zur Messung vom Programmlaufzeiten oder ähnlichen Messungen verwendet werden.

Diese beiden Anweisungen werden benutzt, um den Programmablauf sichtbar zu machen. Sie werden hauptsächlich bei der Fehlersuche verwendet.

In dem folgenden Beispiel wird TRON und TROFF direkt eingegeben.

```
100 K=10
110 FOR J=1 TO 2
120 L=K+10
130 PRINT J;K;L
140 K=K+10
150 NEXT
160 END

 1   10   20
 2   20   30


TRON
Ok
run
[100][110][120][130] 1 10 20
[140][150][120][130] 2 20 30
[140][150][160]
Ok
TROFF
Ok
```

Die durchlaufenen Zeilennummern werden in eckigen Klammern ausgedruckt. Soll nur ein Teil eines Programms untersucht werden, so können diese Anweisungen mit Zeilennummern in das Programm eingesetzt werden.

VAL(A$)

Diese Funktion wandelt eine Zahl, die als Zeichenkette in A$ gespeichert ist, in eine numerische Zahl um.

```
100 ON ERROR GOTO 300
110 READ A$
120 PRINT VAL(A$);
130 GOTO 110
200 DATA "-2","10.45","KARL","10A"
300 END

-2   10.45   0   10
```

Die Zahlen werden mit Vorzeichen und Dezimalpunkt gewandelt. Ist A$ keine Zahl sondern eine Zeichenkette, so wird der Wert 0 ausgegeben. Die Zeichenkette wird solange gewandelt, bis ein Zeichen gefunden wird, das keine Ziffer darstellt.

VARPTR(X)

Mit der Funktion VARPTR wird die Adresse angegeben, bei welcher der Wert der Variablen X gespeichert ist.

```
100 I%=10000
110 A=VARPTR(I%)
120 PRINT A
130 FOR J=0 TO 1
140 PRINT HEX$(PEEK(A+J));
150 NEXT J


-32690
1027
```

Der Wert von 10000 ist bei der Adresse -32690 (804EH) gespeichert. 2710H ist dezimal 10000.

VPEEK(I)
VPOKE I,J

Diese Anweisungen entsprechen den PEEK und POKE Anweisungen. Die Werte werden aber aus dem Bildschirmspeicher geholt, bzw. nach dorthin gespeichert.

WIDTH I

Durch WIDTH wird die Anzahl I der Zeichen in einer Zeile festgelegt.

Notizen

6

MSX der neue Standard für Heimcomputer

Wer die Entwicklung des Heim-und Personal Computermarktes von Anfang an verfolgt hat und gesehen hat welches Spiel die PC Hersteller mit den Anwendern geführt haben, der wird sicher verstehen, warum es zwangsläufig zu einem Standard kommen mußte. Jeden Monat wurden auf dem Markt neue Computer vorgestellt und jedesmal wurde ein anderes Betriebssystem, eine andere Computersprache, ein anderes Kassetten- und Diskettenformat oder eine andere Bildschirmaufteilung gewählt. Nichts war mit nichts mehr kompatibel.Die neuen Systeme waren zwar oft leistungsfähiger als deren Vorgänger, aber die vom Anwender bis dahin erworbene oder erstellte Software war wertlos.

Stellen Sie sich einmal vor, es käme heute eine Firma und stellte Schallplatten und Plattenspieler mit 49 Umdrehungen pro Minute vor, die in allen Eigenschaften die herkömmlichen Systeme mit 33 und 45 U/min übertreffen und wirklich besser und leistungsfähiger sind. Diese Geräte dürften sogar billiger sein, ich bin jedoch sicher, daß sie niemand kaufen würde.

Sie sehen daraus, daß es nicht immer das Neuere und Leistungsfähigere sein muß. Es muß einfach Standard sein.

Apple Computer sollte an dieser Stelle lobenswert erwähnt werden, da sie als erste Firma das Prinzip der Abwärtskompatibilität über die Apple II Serie einhielten. Bei

den Personal Computern hat IBM mit einem Paukenschlag den Standard einfach vorgeschrieben. Ein Industriegigant wie IMB, der weltweit mehr Gewinn erwirtschaftet als sein nächster Mitbewerber Umsatz macht, kann eine solche Standardisierung erzwingen. Kein anderer Hersteller ist alleine in der Lage eine solche Standardisierung in so kurzer Zeit auf dem Markt durchzuführen.

Es ist gar nicht so lange her, da war das Wort Personal Computer beinahe ein schmutziges Wort. Man wurde in die Nähe von Bastlern und elektronischen Flickschustern gerückt, wenn man dieses Wort verwendete. Zwei Zeitschriften Redakteure wollten das Wort Personal Computer sogar abschaffen. Mit der Begründung: Sie hätten herausgefunden, daß die meisten Leute bei dem Wort Personal Computer an einen Computer in der Personal Abteilung denken würden. Bei einem Meeting in einem Chinarestaurant in München wurde beschlossen, das Wort Personal Computer offiziell abzuschaffen und diese Geräte fortan nur noch mit Tischcomputer zu bezeichnen.

Heute,nachdem IBM den Namen Personal Computer mit oder PC salonfähig gemacht hat, käme niemand mehr in diesem Lande auf die Idee diese Bezeichnung auch nur im geringsten anzuzweifeln. An diesen Ausführungen wollte ich Ihnen demonstrieren was ein Standard bewirken kann, wenn er mit genügend Kraft durchgesetzt wird. Ich bin sicher, daß die oben erwähnten Herren heute das Wort Personal Computer öfter verwenden als das Wort Tischcomputer. Der Standard hat sie also bekehrt!

Ähnliches wird auch mit all den anderen geschehen, die heute noch nicht glauben können, daß in einigen Monaten MSX der neue Standard im Heimcomputer Bereich sein wird. Hier haben wir es einmal nicht mit IBM zu tun aber dafür mit:

Stand 6.Januar 1985

Spectravideo (eigentlicher Katalysator von MSX)
Philips

Samsung Korea
Goldstar Korea
Daewoo Korea
Sony mit HIT BIT
Hitachi
Toshiba
Fujitsu
Matsushita
JVC
Canon
Mitsubishi
Yamaha
General Paxon
Pioneer
Sanyo
Kyocera

Im Gespräch sind folgende Firmen, von denen man annimmt, daß sie auch Computer nach dem MSX Standard fertigen werden.

ITT
Siemens
Olympia

Gerüchte gehen sogar in der Branche um, daß IBM eventuell auch Heimcomputer nach dem MSX Standard fertigen will.

MSX ist seit ca 1 Jahr in Japan ein voller Erfolg. Dort wird bereits die zweite Generation von MSX Maschinen verkauft.Ca 300.000 MSX Rechner wurden in Japan verkauft. Im Herbst 1984 wurden MSX Rechner in England, Italien, Benelux und Frankreich eingeführt. Mit großem Erfolg. Einige englische Zeitschriften sträubten sich mit Händen und Füßen gegen die neuen Rechner und mußten sich von Lesern einseitige Berichterstattung vorwerfen lassen.

Die MSX Welle war nicht aufzuhalten. In Deutschland ist der offizielle Auftakt für das Frühjahr 1985 vorgesehen. Die Sterne stehen gut. Jeder verantwortungsbewußte Händler sollte jetzt seinem Kunden keinen Rechner mehr

anbieten, von dem er weiß, daß sein Kunde in ein paar Monaten ohne ein umfassendes, von vielen Herstellern getragenes Softwareangebot dasteht.

Ich habe selbst erlebt, wie Heimcomputer Besitzer sich beklagt haben, daß gerade ihr Computer eingestellt wurde und es jetzt keine Software mehr für ihren Computer gibt. (Ohio Scientific, Dragon, alter PET und CBM, TRS-80 Model 1, Sorcerer um nur einige wenige zu nennen)

Diese Zeiten werden mit MSX endgültig vorbei sein. Kein MSX Besitzer braucht sich in Zukunft zu fürchten, daß sein Rechner in ein paar Monaten veraltet ist und es keine Software mehr gibt.

Ein Riesenangebot an Peripherie wird kaum Wünsche offen lassen.Die MSX Hersteller wollen sich sogar untereinander absprechen, wer welchen Zusatzbaustein produziert. So werden die Kräfte zu Gunsten des Kunden besser koordiniert und während der eine eine 80 Zeichenkarte entwickelt, die auf allen MSX Rechnern arbeitet, kann der andere Hersteller einen Lichtgriffel produktionsreif machen. So gelangt der Anwender in kürzester Zeit zu einer optimalen Auswahl. Wenn wir das Wort "veraltet" hören, so werden viele Gegner von MSX sagen: Sie haben recht, MSX kann nicht mehr veralten, es ist bereits eine veraltete Technologie. Vielleicht hat man hier das Wort "veraltet" mit "ausgereift" oder "erprobt" verwechselt.

Alle MSX Computer werden nach einer bestimmten Spezifikation hergestellt die wie folgt aussieht:

Die 8 Bit Orginal Spezifikation:

- -Z 80 A CPU Prozessor mit 3.57 MHz Taktfrequenz.
- -Videoprozessor von Texas Instruments TMS 9918 A.
- -Tonerzeugungsbaustein AY-3-8910. Dieser ist identisch mit dem AY-3-8912 (Gehäuse ist verschieden). Der gleiche Baustein war auch im Color Genie von TCS.

-32 kByte ROM mit Microsoft Extended BASIC
-Mindestens 8k Byte RAM -max 1MByte adressierbar.

-Bildschirm Aufbau: 40 Zeichen pro Zeile mit 24 Zeilen pro Bildschirm
-(39x24)/16 Farben oder 256x192 Pixel
-Standard Erweiterungs Stecker mit 40 Anschlüssen
-Kassetten Interface Anschluß FSK moduliert 1200 oder 2400 baud
-Joystickstecker mindestens einen Stecker
-Möglichkeiten für den MSX DOS Ausbau
-Ein/Ausgabe Baustein Intel 8255 oder äquivalent

Optionale Erweiterungen:

-80 Zeichen Karte
-Echtzeituhr mit CMOS backup
-RS 232 Schnittstelle
-MS-DOS kompatibles Diskettenbetriebssystem
-8 Bit parallele Drucker Schnittstelle

Bisherige Erfahrungen mit MSX Computern haben gezeigt, daß es sich um eine solide Entwicklung mit einem gut durchdachten und einfach für den Programm Entwickler zu handhabendes Betriebssystem handelt. Das Betriebssystem ist leicht durchschaubar und hat keine Haken und Ösen. Exotische Programmiertechniken wie wir diese vom Atari 800 oder von dem Commodore 64 oder VC 20 her kennen, finden wir hier nicht vor. Das BASIC ist eines der leistungsfähigsten Microsoft Versionen. MSX wird in großem Maße vom Microsoft in den Markt gedrückt werden und bald den unteren Heim Computer Markt voll beherrschen.

An wen wendet sich der MSX Standard in erster Linie?

MSX Computer werden sicher nicht in erster Linie an den Hacker und Freak verkauft werden, der den Computer nach Anzahl der CPU Bitbreite, Bildschirmauflösung u.s.w aussucht. Diese Kunden sind sicherlich mit einem IBM PC Ju-

nior, einem ATARI 130ST oder ATARI 520ST besser bedient.

MSX wendet sich in erster Linie an denjenigen Kunden, der sich bisher überhaupt noch nicht mit dem Gedanken an einen Computerkauf beschäftigt hat. MSX Computer werden im Rahmen einer Heim-Unterhaltungs Anlage, bestehend aus Fernsehgerät, Videorecorder, Laserdisk, Telefonmodem und Computer in einer kompletten Anwendung erscheinen. Der Schwerpunkt wird dann in erster Linie bei der Anwendung des Computers liegen und der Kunde interessiert sich auch nur für das Ergebnis, welches von der Heim Unterhaltungs Anlage geliefert wird.

Und diese Ergebnisse werden gigantisch sein. In diesem Falle interessiert es den Kunden und Anwender in keiner Weise welcher Computer sich in der Anlage befindet und wieviel Bit dieser hat. Er ist von der technischen Leistung fasziniert und oft wird er den Computer nur als kleinen Teil seiner Anlage sehen. Daß er dann auch noch in BASIC programmieren kann und mit dem Computer als solchem spielen kann, ist in diesem Falle sekundär.

Bisher gestaltete sich der Heimcomputer Markt gerade in der umgekehrten Richtung. Der Freak oder Interessierte kaufte einen Heimcomputer in erster Linie einmal um so ein Gerät überhaupt zu besitzen. Der Heimcomputer als Selbstzweck. Dann begann man BASIC zu lernen und mit dem Computer zu spielen. Erst nach alledem überlegte man sich, was man eigentlich sinnvolles mit dem Heimcomputer anfangen könnte. Sie sehen hier, daß man bei MSX von einer ganz anderen Seite an den Markt und den Kunden herangeht.

Aus diesem Grunde sind die MSX Computer von der Konstruktion so aufgebaut, daß sie zu Stereoanlagen und Laser Disk Systemen in einen Aufbau hineinpassen und so mit diesen Geräten eine Einheit bilden. Die Heim-Unterhaltungs-Station der Zukunft.

Ein Pioneer MSX Computer zusammen mit einer Laserdisk.

Wer einmal einen MSX Computer mit angeschloßener Laserdisk gesehen hat und welche Spiele und Grafikspektakel damit veranstaltet werden können, der wird begeistert sein. Es ist auch später möglich, daß man einen MSX Computer zusammen mit Bildschirmtext und einer Laserdisk als Heimkaufhaus verwenden kann. Der Kunde sitzt in seinem Wohnzimmer und hat eine Laserdisk die einen kompletten Kaufhauskatalog enthält. Über die Laserdisk werden alle Produkte und im Reiseteil Hotels, Strände und die Umgebung vom Urlaubsort als Film auf den Bildschirm gebracht.

Der Computer bringt über Bildschirmtext alle Informationen über Preis, Verfügbarkeit, Buchungen etc. mit auf den Bildschirm. Im Dialog mit dem Hersteller kann dann abgefragt, weitere Informationen anfordert oder

sogar gebucht werden. Dies ist ein einfaches Beispiel für eine Heim Unterhaltungs Anlage, wie wir sie in einigen Jahren in vielen Wohnungen vorfinden werden. Dieser Markt wird sehr groß sein und den der Heimcomputer Käufer zum Selbstzweck, um ein Vielfaches übertreffen. Genau in diesen Markt zielt MSX und die große Anzahl der Hersteller wird für den Erfolg garantieren.

Wo wird MSX verkauft werden?

In allen Fachgeschäften, Kaufhäusern und im Rundfunk und Fernsehhandel.
In den Duty Free Shops auf den Flughäfen.
In den FOTO Geschäften und Ketten

Zukünftige Entwicklungen:

Kay Nishi von Microsoft hat versichert, daß alle neuen Computer, die im MSX Standard folgen werden, abwärtskompatibel sein werden. D.h. vorhandene Software können Sie mit auf Ihren neuen Computer "mitnehmen".

Kay Nishi (links im Bild) hier zusammen mit dem Verfasser dieses Buches.

In diesem Bereich haben die Hersteller mit ihren Kunden in den letzten Jahren viel Schindluder getrieben. Nahezu alle haben jedoch heute dazugelernt. Beispiel: Commodore 128 PC ist kompatibel mit C-64.Auch der neue ATARI 65XE ist zu seinem Vorgänger 800 XL voll kompatibel.

Die 8 Bit MSX Computer:

In Zukunft ist ein MSX-2 geplant. Ein neuer Video Baustein mit besserer Graphik und besseren Anschlußmöglichkeiten. Prototypen sollen Ende 1985 oder später auf den Markt kommen.

Die MSX Hersteller sind gerade dabei alle Bausteine, die man zum Aufbau eines MSX Computers benötigt auf einem Chip zu integrieren. (CPU, Video Chip, Sound,RAM ROM, I/O usw.). Dies bringt erhebliche Preisvorteile bei der Produktion, verkleinert die Platine und das Gehäuse und bietet eine höhere Betriebssicherheit. Weniger Anschlüsse auf der Platine ergibt eine höhere Zuverlässigkeit. Auch der Verbraucher profitiert von dieser Entwicklung.

16 Bit MSX - Wann werden wir es zu sehen bekommen?

Wenn wir einmal den IBM PC Junior und den Sinclair QL ausklammern, so wird der Heimcomputer Markt auch heute noch von 8 Bit Rechnern beherrscht. Auch der neue ATARI 800 XL Nachfolger, ATARI 65XE, arbeitet mit einer 6502 8-Bit CPU. Der Commodore 64, der auch weiterhin Commodores "low end" Maschine bleiben wird, begnügt sich mit einer 8-Bit CPU.

Der Schritt in Richtung 16 Bit und 32 Bit wird sicherlich auch bei MSX nicht ausbleiben, da die Konkurrenz wie Apple IIx und ATARI 130 ST mit ihren Auflösungen des Bildschirms und dem riesigen adressierbaren Speicherbereich neue Maßstäbe im mittleren PC Bereich setzen werden. Ein 16 Bit MSX existiert bereits und ähnelt sehr

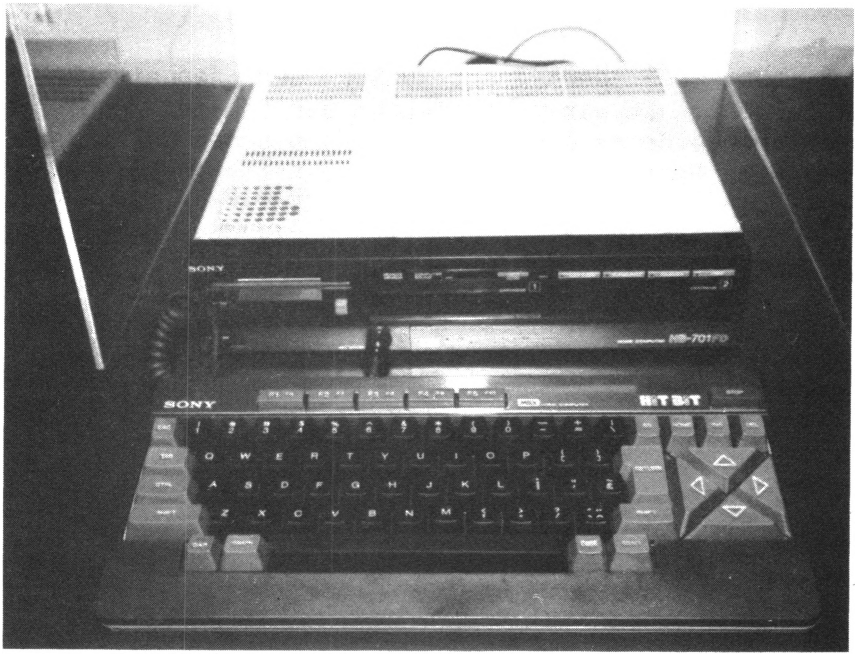
dem MS-DOS von Microsoft. Ein großer Vorteil dieses MSX-DOS ist, daß es auch CP/M Files lesen kann. Genauso wie an den 16 Bit PCs arbeitet man auch an 32 Bit MSX Maschinen. Bis zur Markteinführung wird jedoch noch eine Weile vergehen. Viele Insider behaupten sogar, daß der amerikanische Markt für MSX nur über die 16 und 32 Bit Systeme zu erobern sein wird.

-In Europa jedoch werden 8 Bit MSX Rechner im Jahre 1985 einen großen Marktanteil erobern können.

Die Zukunft des Standards.

Kay Nishi, der Präsident von ASCII Corporation in Japan, hat angekündigt, daß in naher Zukunft eine Verbesserung der MSX Maschinen erfolgen wird. Die Video Ausgabe soll wesentlich verbessert werden, z.B. 80 Zeichen pro Zeile und eine Auflösung von 512x212 bei 256 möglichen farbigen Bildpunkten sowie NAPLPS Standard Grafik. Man weiß auch schon, daß die Maschinen dieser neuen Generation nicht alle Software der jetzigen Maschinen verarbeiten werden, aber die Commodore 128 und ATARI 65 weit übertreffen werden. Die Verbesserungen im Videobereich werden in erster Linie durch einen neuen Videokontroller mit 64kRAM on board erreicht. Die neuen Bausteine sollen es sogar erlauben, daß man Bilder von angeschlossenen Videorecordern oder Fernsehgeräten manipulieren kann. (Bildverarbeitung)

Nicht alle Japaner unterstützen MSX. Sharp und NEC sind weltweit und in Japan auch mit ihren eigenen Maschinen erfolgreich. Fujitsu promoted seine FM 7 Heimcomputer, die in den gleichen Bereich wie MSX fallen, wesentlich stärker als seine MSX Maschinen. Philips und Sony sind sehr stark in der Bundesrepublik Deutschland. So wie die Entwicklung jetzt verlaufen ist, werden Sony und Philips in Deutschland wohl als erste den Markt bearbeiten.



Hier der SONY Hit Bit mit 3,5 Zoll Diskettenstation

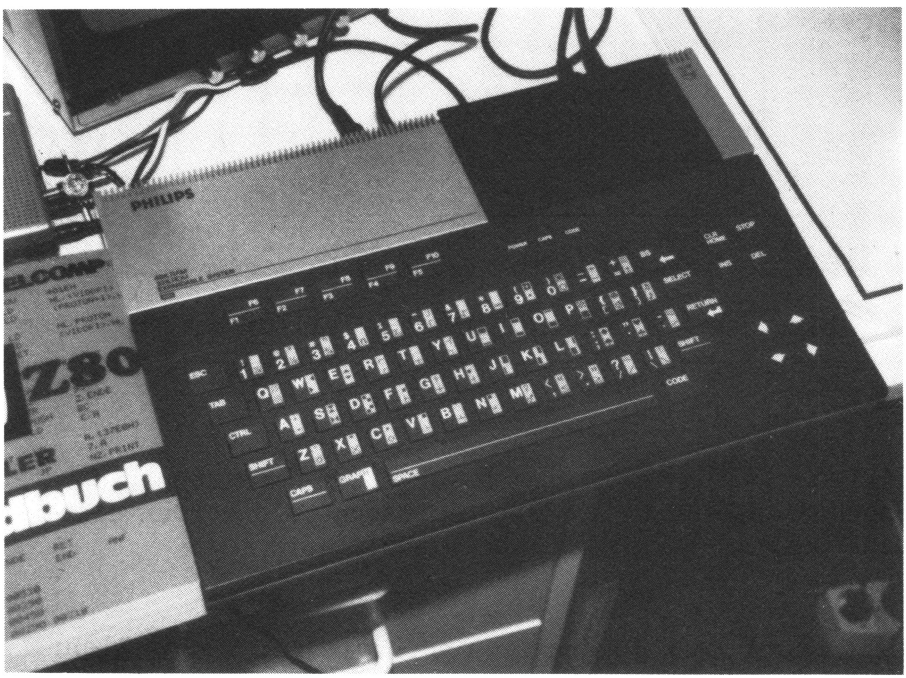
Einige Händler haben sich sogar MSX Maschinen der 2. Generation bereits aus Japan besorgt und werden diese in den nächsten Tagen und Wochen anbieten.

Die Generationen lassen sich grob wie folgt einteilen:

1.Generation: MSX Computer ohne eingebauten Kassetten Recorder.
2.Generation: MSX Computer mit eingebautem Kassetten Recorder, IBM ähnlicher Tastatur und Video Recorder oder Laserdisk Anschluß.
3.Generation: MSX Computer mit 3,5 Zoll Floppy Disk, VCR Anschluß , Compact Disk Anschluß Laser Disk usw.

MSX und die Firmen.

Sony ist wohl der größte unter den japanischen MSX Herstellern. Er war auch in den vergangenen Monaten der dominierende Hersteller auf dem japanischen Markt. Durch den guten Namen und die ausgezeichneten Vertriebskanäle können wir Sony als die Nr.1 der MSX Hersteller bezeichnen. Sony ist in keiner Weise traurig über den Eintritt von Philips in die MSX Arena.



Hier der Philips MSX Computer der ersten Generation.

Im Gegenteil, man ist bei Sony zu jeder Zusammenarbeit beim Anschluß von Laser Disk und Compact Disk an die MSX

Maschinen bereit. Auch in Deutschland hörte man von MSX zuerst im Zusammenhang mit dem Namen Sony. Der HIT BIT wurde im letzten Monat von nahezu allen deutschen Fachzeitschriften getestet.

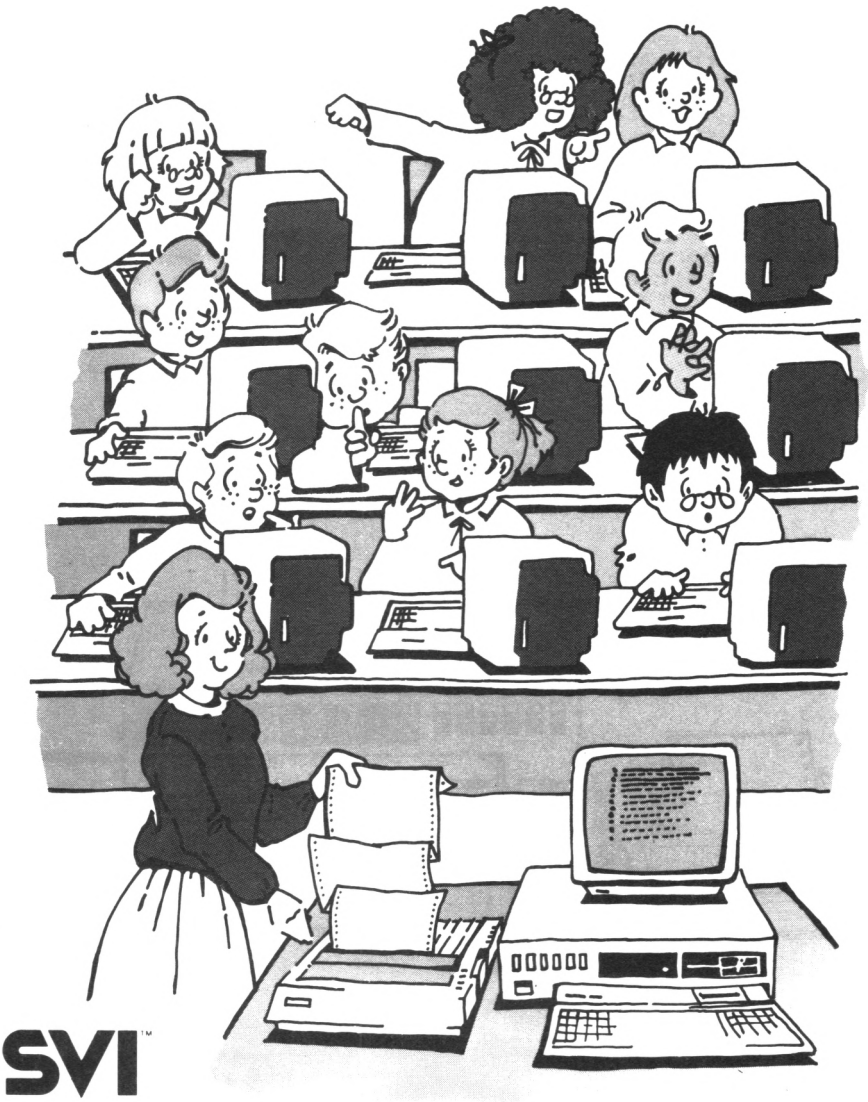
Der Spectra Video SVI 7728 MSX

Spectravideo, der eigentliche Katalysator des MSX Standards, hat in den letzten Monaten erhebliche Marketingprobleme durchgestanden. Bondwell hat mittlerweile 55% der Firmenanteile von Spectravideo erworben. Die Herren Weiss und Fox, die als die eigentlichen Erfinder von MSX gelten, sind dabei, neue Produkte in einem anderen Marketing Konzept anzubieten. Die neuen peripheren Geräte und Communications Systeme von Spectravideo sind beachtenswert. Sehr interessant ist die Stringy Floppy (viele kennen diese vom TRS-80 her) und ein preiswertes

lokales Netzwerk. Dieses LAN erlaubt den Verbund von max. 32 Computern und könnte für Schulen und kleine Betriebe recht interessant werden (Übertragungsrate 230 kBit pro Sekunde bis zu Entfernungen von 300 m). Die Spectravideo Systeme sind zur Zeit die einzigen MSX Rechner mit einer Anzeige von 80 Zeichen pro Zeile und CP/M 2.2.

Die neue MSX Generation von SVI

Das SVI-Netzwerk



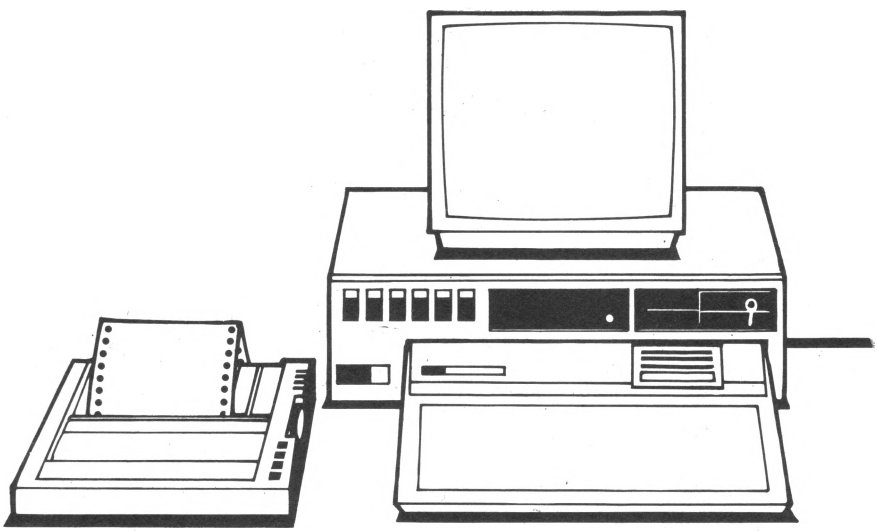
CP/M, MSX-DOS und Basic – komplette Ausbildung mit dem SVI-Netzwerk

Als Ausbilder kennen Sie die Arbeitserleichterungen, die ein Netzwerk (d. h. der Zusammenschluß mehrerer Computer unter der Regie einer Zentraleinheit) bei der täglichen Ausbildung bieten kann. Natürlich wissen Sie auch um die immense Speicherkapazität und den extrem schnellen Datenzugriff eines Festplatten-Laufwerkes, und schließlich wissen Sie auch, daß die Realisierung eines derartigen Projektes für die meisten Bildungseinrichtungen unerschwinglich ist. Bis jetzt!

Der Preis des SVI-Netzwerkes ermöglicht es erstmals auch Schulen mit kleinem Budget, den entscheidenden Schritt in Richtung Mehrplatz-Computer-System zu wagen. Durch die Ausbaufähigkeit und den Anschluß von max. 32 Rechnern sowie die Möglichkeit, CP/M-, MSX-DOS- und BASIC-Programme zu betreiben, ist die Entscheidung für das SVI-Netzwerk eine zukunftssichere Lösung.

Das SVI-Netzwerk erlaubt es, die Möglichkeiten eines Festplatten-Laufwerkes voll auszunutzen. Jeder Schüler einer Klasse kann darauf zurückgreifen; gleichzeitig steht es aber auch für schulinterne Aufgaben zur Verfügung (Stunden-, Raum-Planung). Durch die mitgelieferte System-Software ist sowohl die Inbetriebnahme und die Kontrolle des Netzwerkes durch den Ausbilder als auch die Bedienung der angeschlossenen Rechner (Terminals) und der Zugriff auf den zentralen Massenspeicher (10 MegaByte) durch die Schüler völlig unproblematisch.

Die Installation und die Inbetriebnahme der Zentraleinheit und der anzuschließenden Rechner kann vom Ausbilder ohne Schwierigkeiten vorgenommen werden.



Die Zentraleinheit

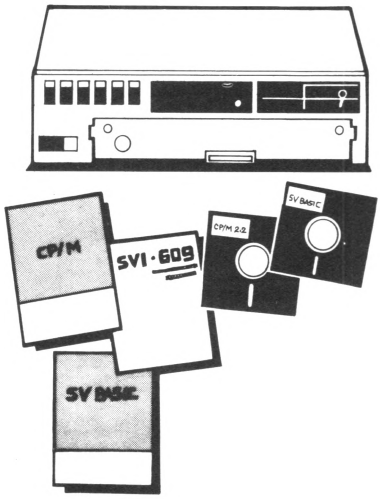
Die Zentraleinheit

Die Zentraleinheit SVI-609 koordiniert den Datenverkehr zwischen den Terminals. Der Ausbilder wird in die Lage versetzt, die Aktivitäten des einzelnen Schülers auf seinem Monitor zu beobachten.

Zum Lieferumfang gehören:

- ein 10-MegaByte-Festplattenlaufwerk
- ein 320 KByte 5 1/4 Zoll-Disketten-Laufwerk
- eine 80-Zeichenkarte SVI-806
- eine parallele Druckerschnittstelle (Centronics-kompatibel)
- 64 K RAM-Speichererweiterung
- RS 232 C-Schnittstelle (seriell)
- SVI-Netzwerk-Schnittstelle
- SVI-Netzwerk-Software

Durch den Einsatz des im Lieferumfang enthaltenen Betriebssystems CP/M ist die Nutzung von PASCAL oder anderen Programmiersprachen möglich. Da auch ein leistungsstarkes Microsoft-BASIC zur Verfügung steht, werden alle Anforderungen, die an ein Computersystem für den Einsatz in Schulen gestellt werden, erfüllt.

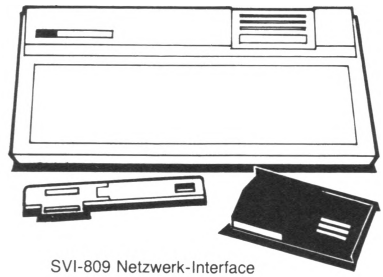


SVI-609 SVI-Netzwerk

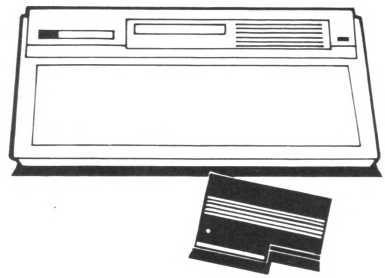
Die Terminals

Jeder Terminal-Rechner kann entweder ein SVI-328, ein SVI-728 oder jeder andere MSX-Rechner mit mindestens 64 K RAM sein. Da es die Netzwerk-Software ermöglicht, MSX-DOS und CP/M zur gleichen Zeit zu betreiben, können die Terminals eines SVI-Netzwerkes sowohl aus MSX-Rechnern und SVI-328 gemeinsam bestehen.

Jeder Terminal-Rechner muß mit einem Netzwerk-Interface ausgestattet sein. Das Interface steht in zwei Ausführungen zur Verfügung: das SVI-809 für den SVI-328 und das SVI-709 für den SVI-728 und andere MSX-Rechner. Das SVI-809 wird zusammen mit dem Mini-Expander SVI-602 geliefert und am Erweiterungsbus des SVI-328 angeschlossen. Das SVI-709 wird in den Cartridge-Schacht des SVI-728 oder anderer MSX-Computer gesteckt. Durch ein im Lieferumfang enthaltenes Kabel werden die Schnittstellen dann mit der Zentraleinheit SVI-609 verbunden.



SVI-809 Netzwerk-Interface
(einschl. Mini-Expander)
und SVI-328 Computer



SVI-709 MSX-Netzwerk-Interface
und SVI-728 Computer

Die Software

Die im Lieferumfang enthaltene Software ist der Schlüssel zur vielfältigen Nutzung der Möglichkeiten, die das SVI-Netzwerk bietet. Die durch die Software abzudeckenden Aufgaben umfassen

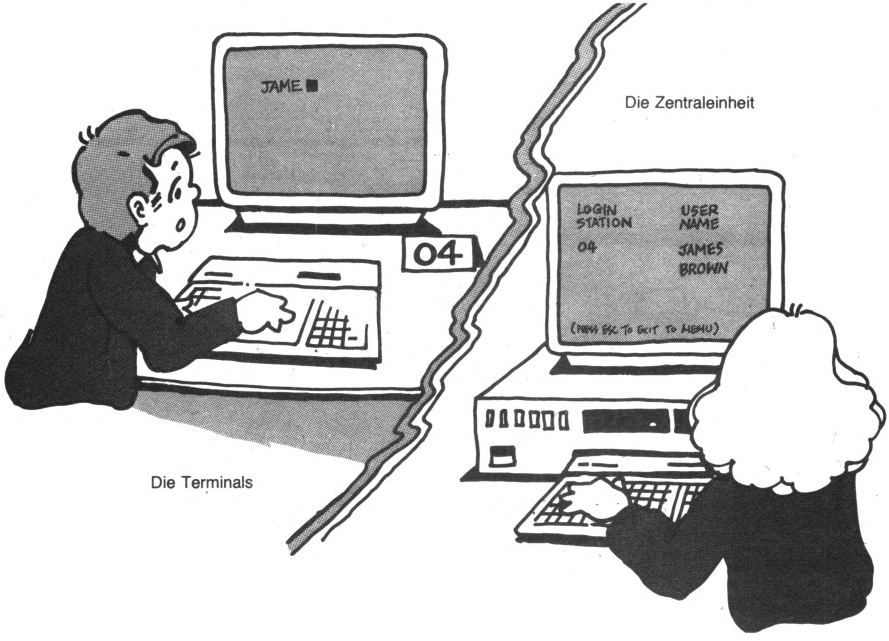
- Initialisieren des Festplatten-Laufwerks
- Kopieren von durch den Anwender erstellten Dateien
- Pufferung des Druckers
- Echtzeit-Koordination der Terminals mit der Zentraleinheit

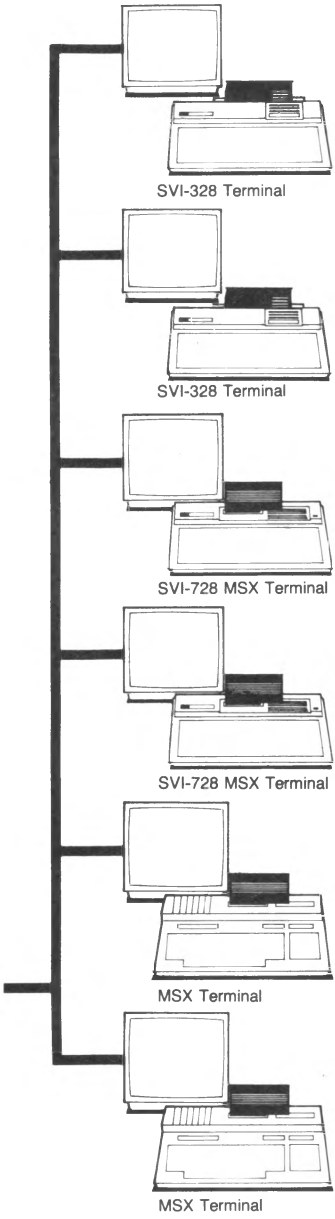
Der von der Software belegte Speicher ist in drei Bereiche untergliedert. Der Systembereich fungiert als Programm-Bibliothek; die hierin befindlichen Programme können simultan von allen Anwendern genutzt werden. Der Zugriff umfaßt nur das Schreiben und Lesen innerhalb dieses Speicherbereichs; das Löschen von Dateien kann aus Sicherheitsgründen ausschließlich von dem Ausbilder an der Zentraleinheit vorgenommen werden.

Der Kommunikationsbereich dient als Zwischenspeicher, über den sowohl Nachrichten als auch Programme zwischen den Terminals ausgetauscht werden können.

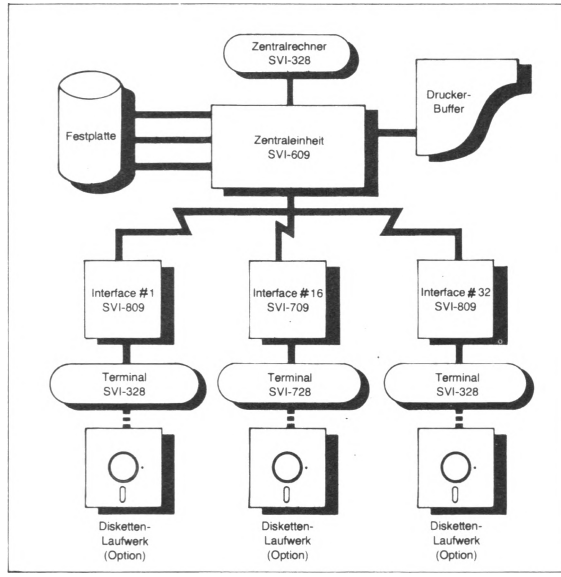
In dem Anwenderbereich speichert der Schüler Dateien ab, auf die kein gemeinsamer Zugriff gewünscht ist. Dem Ausbilder ist der Zugang zu diesen Dateien selbstverständlich nicht verwehrt.

Zusätzliche, nur durch den Ausbilder zu nutzende Programme umfassen die komfortable Überwachung aller Aktivitäten innerhalb des SVI-Netzwerkes, das Kopieren von kompletten Anwender-Bereichen auf die Hard-Disk und das Verwalten der dort befindlichen Dateien.





Soft- und Hardware gemeinsam nutzen



SVI-NETZWERK-Aufbau

Jeder mit einem SVI-Netzwerk-Interface ausgestattete MSX-Rechner (SVI-728) oder jeder SVI-328 kann die Aufgabe eines Terminals innerhalb des SVI-Netzwerkes übernehmen. Die Zentraleinheit ermöglicht die gemeinsame Nutzung des Festplatten-Laufwerks und angeschlossener Peripherie (Drucker) für bis zu 32 dieser Rechner.

Die Zentraleinheit arbeitet zusammen mit einem SVI-328 und kontrolliert den Zugriff zum Festplatten-Laufwerk sowie den gesamten Datenfluß innerhalb des SVI-Netzwerkes.

Jedes Terminal kann einen reservierten Teil der 10 000 000-Byte-Kapazität der Festplatte für eigene Zwecke nutzen, während alle Terminals auf den gemeinsamen Speicher und die darin befindliche Software zurückgreifen.

Spezifikationen

HARDWARE

ZENTRALEINHEIT

- SVI-328-Computer (nicht im Lieferumfang enthalten)
- SVI-609 Expander
- Winchester-Festplatte, 10 MegaByte (formatiert)
- Diskettenlaufwerk, 320 KiloByte (formatiert), 40 Track, DS/DD
- Drucker-Schnittstelle parallel (Centronics-kompatibel)
- SVI-807 64 K RAM-Speichererweiterung
- SVI-805 RS 232 C-Schnittstelle (seriell)
- SVI-806 80-Zeichenkarte
- RS 422 Netzwerk-Schnittstelle

TERMINAL

Als Terminal können folgende Rechner in beliebiger Zusammenstellung dienen:

1. SVI-328-Computer
2. SVI-728 MSX-Computer
3. jeder MSX-Computer mit mind. 64 K RAM

Der Anschluß an das SVI-Netzwerk erfolgt bei dem SVI-328 über die Netzwerk-Schnittstelle SVI-809 (SVI-602 Mini-Expander ist im Lieferumfang enthalten) und bei den MSX-Rechnern über die SVI-709-Schnittstelle.

ANSCHLUSSDATEN:

Kabelverbindung: Telefonkabel
max. Kabellänge: ca. 300 m
Übertragungsgeschwindigkeit: 230 KBaud/Sek.
Netzwerk-Organisation: polling
max. Terminal: 32

SOFTWARE

ZENTRALEINHEIT

Die folgende Software ist im Lieferumfang der Zentraleinheit enthalten:

- CP/M 2.2 BETRIEBSSYSTEM
- DISKETTENVERWALTUNG
  zum Formatieren und Aufteilen der Festplatte
- NETZWERK-VERWALTUNG
  zum Verteilen der Winchester- und Druckerkapazität
- DRUCKER-BUFFER
  16 K Drucker-Buffer zur Zwischenspeicherung der für den Drucker bestimmten Daten; dadurch erhebliche Steigerung der Arbeitsgeschwindigkeit
- KOPIERPROGRAMM
  komfortables Programm zum Kopieren der Hard-Disk-Dateien auf eine Diskette
- FESTPLATTEN-ZWISCHENSPEICHER
  96 K Zwischenspeicher zum Erhöhen der Zugriffgeschwindigkeit auf die Dateien des Festplatten-Laufwerkes

TERMINAL

Je nach gewähltem Terminal stehen verschiedene Betriebssysteme zur Verfügung:
SVI-328-RECHNER

- CP/M 2.2
- Disk-BASIC

MSX-RECHNER

- MSX-Disk-BASIC
- MSX-DOS

Jedes angeschlossene Terminal kann zusätzlich mit bis zu 2 Diskettenlaufwerken ausgestattet werden.

GEMEINSAME MEDIEN

MASSENSPEICHER

- 9 MegaByte frei verfügbar auf Festplatte

DRUCKER

- Jeder Standard-Centronics-Drucker (Option)

SVI™

Generalimporteur: Bernd Jöllenbeck GmbH Internationale Industrievertretungen Import Export 2730 Weertzen Telefon (0 42 87) 6 91- 5 Telex 2 49 635

1/85 2 2E

C-307

Philips.

Philips hatte an einem eigenen Homecomputer gearbeitet und machte in letzter Minute eine Entscheidung für MSX. Sicher sind die japanischen Hersteller über diesen Entschluß sehr erfreut, denn ein Name wie Philips ist sicher eine Gewähr für einen besseren Start dieses Standards in Europa. Philips hat seinen VG 8010 MSX zuerst in Deutschland, Österreich, Belgien und in Italien eingeführt und trotz einer Gummitastatur an den ersten Geräten, recht gute Verkaufserfolge erzielt. Philips ging mit seinen Geräten hier in erster Linie in die Radio und Fernsehgeschäfte. Interessant ist, daß die Philips MSX Computer in Frankreich produziert wurden.

MSX und die Zukunft

In Japan diskutiert man zur Zeit einen Home-Bus. Dies sind Kabelanlagen, welche die heutige elektrische Installation und die Telefonleitungen ersetzen sollen.

Dieser Home-Bus soll alle nur erdenklichen Energie und Kommunikationsleitungen beinhalten. Z.B. Computer Netzwerke, Glasfaser Verbindungen in zwei Richtungen für Audio und Video, elektrischer Strom für jedes Zimmer und Telefonleitungen, die über Satelit mit jedem anderen Haus in der freien Welt kommunizieren können. Der Mensch kann so von jedem Punkt der Erde über ein Überalltelefon zu Hause anrufen und z.B. die Heizung, das Licht oder den Elektroherd ein und ausschalten oder Informationen abrufen und zuweisen. Die Unterhaltung im Heim wird dann Dimensionen und Formen annehmen, die heute noch jenseits unseres Vorstellungvermögens liegen.

Zusammenfassung

Als Zusammenfassung meiner kleinen Einführung in die Philosophie von MSX kann gesagt werden, daß die MSX Technology zwar nichts revolutionierendes in sich hat. Die Technik ist, wie bereits gesagt, alt. Sie arbeitet aber ausgezeichnet und sehr zuverlässig. MSX ist nicht

dazu da, um ein Super Computer zu sein. In erster Linie wird MSX preiswert und von hoher Qualität sein. Das weiß der Handel und der Verbraucher zu schätzen.

Weiterhin wird es ein Riesenangebot an Software und Hardwarezusätzen geben. Wir alle wissen, daß es die Software ist, die hilft die Computer zu verkaufen. Eine große Anzahl von Herstellern bietet dem Anwender die Gewißheit, daß er nicht von einem einzigen marktbeherrschenden Lieferanten erpresst wird, immer wieder neue Computer zu kaufen, deren Software untereinader nicht kompatibel ist. Auch der einmal gekaufte Drucker lässt sich an den nächsten Heimcomputer jetzt noch anschliessen. Die Tatsache, daß es einen Standard gibt, wird sich auch auf das Verhalten der anderen Computer Hersteller auswirken. Auch wenn sie keine MSX Maschinen produzieren werden, so werden sie sich sicher hüten ein Nachfolgemodell vorzustellen, welches nicht mehr kompatibel zum Vorgänger ist.
Auf diese Weise hilft MSX sogar denjenigen, die keinen MSX Computer kaufen.

Wichtige Hersteller - Eine Auswahl.

Auf Messen wurde ich immer wieder nach einer Übersicht über die wichtigsten MSX Computer gefragt. Aus diesem Grunde möchte ich an dieser Stelle eine kleine Zusammenstellung der bei uns zu erwartenden MSX Computer geben. Die Fabrikate der einzelnen Hersteller stimmen zwar in den Grundspezifikationen überein, haben aber meistens eine ganz bestimmte Eigenschaft oder Sonderfunktionen, die ein anderer MSX Rechner nicht aufweist.

So ist zum Beispiel der Yamaha MXS Computer mit einem besonderen Musiksyntisizer ausgerüstet, der HIT BIT von SONY hat dafür eine eingebaute Adressenverwaltung, Terminkalender und Datenbank in ROM, der CASIO MSX Rechner ist besonders preiswert und hat eine Tastatur ähnlich wie der Sinclair Spectrum. Hier hat der Anwender die Auswahlmöglichkeit, sich den Rechner zu kaufen, der seinen Wünschen am nächsten kommt.

Hier sehen sie z.B. den Mirwald MSX mit eingebautem Cassetten Recorder. Die Tastatur ist in der Art der IBM Tastatur gehalten. Eine Supergrafik, ähnlich dem Apple Macintosh ist eingebaut. Mit dem Joystick lassen sich über Ikonen und Symbole Grafiken erstellen und mit Text mischen. Ein Zoom erlaubt ein Verändern der Zeichen usw. Weiterhin ist ein Musiksynthisizer und ein Maschinensprachen Monitor in ROM enthalten.

Der Sanyo MSX hat einen eigebauten Lichtgriffel. Dieses Gerät dürfte deshalb für die Schule und für Kinder besonders interessant werden.

Den Philips Computer hatten wir Ihnen schon gezeigt.

Nachteil ist die schlechte Tastatur und die fehlende parallele Druckerschnittstelle. Der Nachkauf ist zwar nicht teuer, aber zu einem Computer gehört nun einmal heute auch ein Drucker. Hinzu kommt, daß hier bei eingestecktem Druckerinterface ein ROM Steck Platz belegt ist und somit verloren ist.

Hier im Bild ein PANASONIC MSX Rechner mit Disketten Station. Bei den MSX Rechnern mit Diskettenstation ist hier der Rechner in einem 19 Zoll Gehäuse untergebracht. In diesem Gehäuse befinden sich die CPU und zwei ROM Steckplätze. Sie sehen im Bild, wie in den einen ROM Steckplatz eine zusätzliche Diskettenstation eingesteckt ist. Der Rechner passt hier im Design gut zu einem Videorecorder, Laserdisksystem oder Stereoanlage. Die Tastatur ist in diesem Falle eine reine Tastatur, ohne Computerinhalt. Im Gegensatz zu den anderen MSX Rechnern, bei denen Tastatur und Rechner in einer Console zusammengefaßt sind.

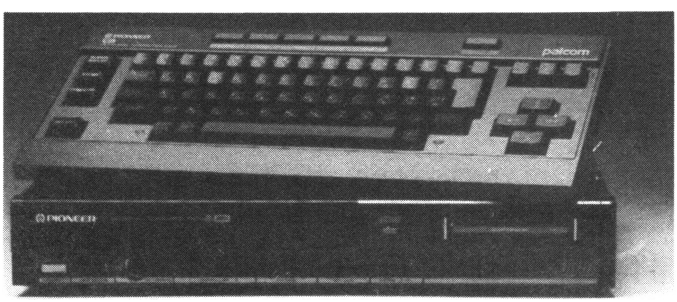
Hier noch einmal der PANASONIC MSX mit Grafiktablett.

Ein weiterer interessanter MSX Rechner ist der YAMAHA Computer. Er ist besonders den Anwendern zu empfehlen, bei denen der Schwerpunkt der Anwendung auf dem Musikbereich liegt. Mit einem preiswerten Keyboard zusammen

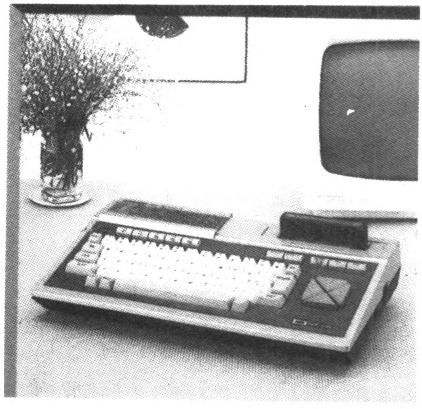
können Sie sich hier eine echte Heimorgel zusammenstellen.

Im nachfolgenden Bild sehen Sie den MSX Computer von TOSHIBA.

Der PIONEER - PALCOM Computer hier als Tastatur mit getrenntem Computer System und eingebauter 3,5 Zoll Floppy.



Der MITSUBISHI MSX Computer



Der DAEWOO MSX Computer

Der Cannon MSX Computer

Der CASIO MSX Computer

Notizen

7

Töne und Geräusche mit dem AY–3–8912

Der PSG (Programmable Sound Generator) AY-3-8912 erzeugt Töne und Geräusche durch Mischen von 3 programmierbaren Rechteckfrequenzen oder einen Rauschgenerator. Über einen D/A-Wandler erscheinen die erzeugten Frequenzen an 3 Ausgangskanälen, die entweder getrennt oder zusammen an eine Verstärkerendstufe geführt werden können. Die Hüllkurve der 3 Ausgangssignale kann über einen programmierbaren Hüllkurvengenerator beeinflußt werden,

Alle Funktionen werden über 16 Register gesteuert.

REGISTER \ BIT		B7	B6	B5	B4	B3	B2	B1	B0
R0	Channel A Tone Period	8-BIT Fine Tune A							
R1		▨	▨	▨	▨	4-BIT Coarse Tune A			
R2	Channel B Tone Period	8-BIT Fine Tune B							
R3		▨	▨	▨	▨	4-BIT Coarse Tune B			
R4	Channel C Tone Period	8-BIT Fine Tune C							
R5		▨	▨	▨	▨	4-BIT Coarse Tune C			
R6	Noise Period	▨	▨	▨	5-BIT Period Control				
R7	Enable	IN/OUT		Noise			Tone		
		IOB	IOA	C	B	A	C	B	A
R8	Channel A Amplitude	▨	▨	▨	M	L3	L2	L1	L0
R9	Channel B Amplitude	▨	▨	▨	M	L3	L2	L1	L0
R10	Channel C Amplitude	▨	▨	▨	M	L3	L2	L1	L0
R11	Envelope Period	8-BIT Fine Tune E							
R12		8-BIT Coarse Tune E							
R13	Envelope Shape/Cycle	▨	▨	▨	▨	CONT	ATT	ALT	HOLD
R14	I/O Port A Data Store	8-BIT PARALLEL I/O on Port A							
R15	I/O Port B Data Store	8-BIT PARALLEL I/O Port B							

7.1 Die Register des PSG

Die Tonerzeugung erfolgt durch Frequenzteilung. Eine von außen angelegte Taktfrequenz wird erst durch 16 und dann durch einen 12-Bit-Zähler geteilt. Dieses 12 Bit Wort wird für den Kanal A in die Register 0 (8 Bit) und 1, die verbleibenden 4 Bit gespeichert.

Für eine gegebene Taktfrequenz kann die Tonperiode TP folgendermaßen berechnet werden:

$$TP = \frac{f_T}{f \times 16}$$

mit f als gewünschte Frequenz und fT der angelegten Taktfrequenz.

Beispiel:

Kammerton A f=440 Hz , fT=1.7898*10^6 (Taktfrequenz MSX)

TP=1.7898*10^16/(440*16)=254.2

Die Zahl 254 in eine 12 Bit Binärzahl gewandelt, ergibt (in Hex-Darstellung) 0FE.

Mit FE als Registerinhalt von R0 und 0 als Registerinhalt von R1 entsteht an einem Kanal ein Rechtecksignal von 440 Hz. Durch die Rundung von TP entsteht natürlich ein Fehler, so daß die sich tatsächlich einstellende Frequenz etwas höher ist und bei 440.35 Hz liegt.

Für die Tonerzeugung muß über die angegebene Formel die Hexadezimalzahl berechnet werden. Diese Berechnung kann mit dem folgenden Programm in Abbildung 7.2 durchgeführt werden.

```
100 LINE INPUT"F=";F$
105 F=VAL(F$)
110 IF F<0 THEN END
120 FT=1789600!
130 FF=16
140 TP=FT/(F*FF)
150 MSD=INT(TP/256)
160 TP=TP-MSD*256
170 NSD=INT(TP/16)
180 LSD=INT(TP-NSD*16+.5)
190 FI=FT/((MSD*256+NSD*16+LSD)*FF)
200 IF MSD>9 THEN MSD=MSD+7
210 MSD=MSD+48:A$=CHR$(MSD)
220 IF NSD>9 THEN NSD=NSD+7
230 NSD=NSD+48:B$=CHR$(NSD)
240 IF LSD>9 THEN LSD=LSD+7
250 LSD=LSD+48:C$=CHR$(LSD)
260 PRINT F;" ";A$;B$;C$;" ";INT(FI*100)/100
270 GOTO 100
```

7.2 Berechnung der Tonperiode

Nach der Eingabe gewünschter Frequenz f werden die Registerinhalte für RL und RH berechnet.

Beispiel: f = 440 Hz (Kammerton a)
Ausgabe : 440 0FE 440.35

Für die Datenübergabe wird der SOUND Befehl verwendet. Mit

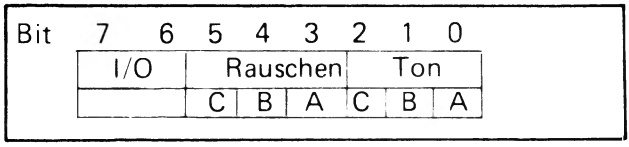
```
SOUND I,J
```

wird die Zahl J in das Register I des PSG geschrieben. Mit

```
100 SOUND 0,&HFE
110 SOUND 1,0
```

wird der Kanal A zur Ausgabe eines Tones mit 440 Hz programmiert. Mit dem Register 7 können die Kanäle ein-

oder ausgeschaltet werden. Die Abbildung 7.3 zeigt die Belegung des Registers R7.



7.3 Belegung des Registers R7

Über die drei Kanäle kann entweder Rauschen oder ein Ton ausgegeben werden. Eine Null im entsprechenden Bit schaltet den Kanal ein. Mit 00111110=3EH wird der Kanal A ausgewählt. Nun kann noch die Lautstärke des Kanal festgelegt werden. Für den Kanal A bestimmt das Register R8 die Lautstärke. Sie kann in 16 Stufen eingestellt werden. Dies zeigt Abbildung 7.4.

Vollste Lautstärke entspricht 0FH. Mit dem SOUND Befehl erhalten wir dann folgendes Programm zur Ausgabe des Tones A.

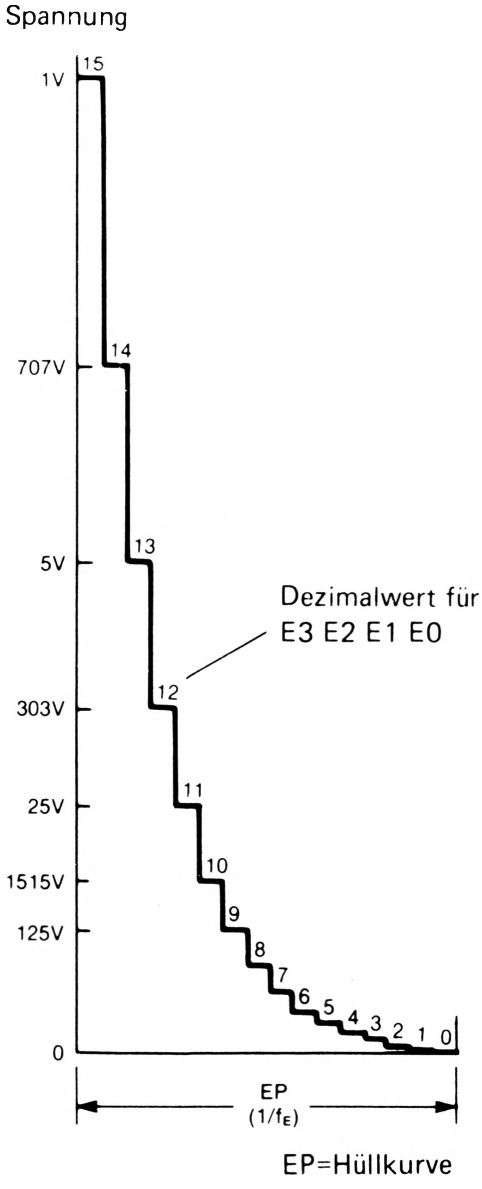
```
100 SOUND 0,&HFE
110 SOUND 1,0
120 SOUND 7,&H3E
130 SOUND 8,&H0F
```

Mit den Registern R2 bis R5 werden die Tonperioden für die Kanäle B und C programmiert.

Das Register R6 dient zur Programmierung des Rauschgenerators. Dabei werden aber nur die unteren 5-Bit verwendet. Die tiefste Rauschfrequenz wird durch 1FH (alle 5 Bit=1) und die höchste Rauschfrequenz durch 01H erzeugt. Die Taktfrequenz wird wiederum erst durch 16 und dann durch das 5-Bit Wort geteilt.

Die Rauschperiode NP berechnet sich dann nach der Formel

$$NP = \frac{f_t}{16 \times f_n}$$



Bei einer Taktfrequenz von 1 MHz kann Rauschen im Bereich von 3.4 kHz -100 kHz erzeugt werden.

Wird in einem der Register 8 bis 10 Bit 5 zu 1 gesetzt, so wird die Amplitude dieses Kanals durch den Hüllkurvengenerator bestimmt.

Dieser wird über die Register R11, R12 und R13 programmiert. R11 und R10 bilden einen 16-Bit-Zähler zur Erzeugung der Periodenlänge der Hüllkurve. Die Taktfrequenz wird erst durch 256, dann durch den Wert der Registerinhalte R11 und R12 geteilt, wobei R12 das LSB ist.

Bei einer Taktfrequenz von 1.7 MHz können Hüllkurvenperioden von 0.1 Hz bis 6800 Hz erzeugt werden. Die Berechnung der Periodendauer erfolgt durch

$$EP = \frac{f_t}{256 \times f_e}$$

Der 16-stellige Binärwert für EP wird in die Register R11 und R12 geschrieben. Die Berechnung der Periodenlänge kann dem Programm in Abbildung 7.2 mit FF=256 in Zeile 130 erfolgen.

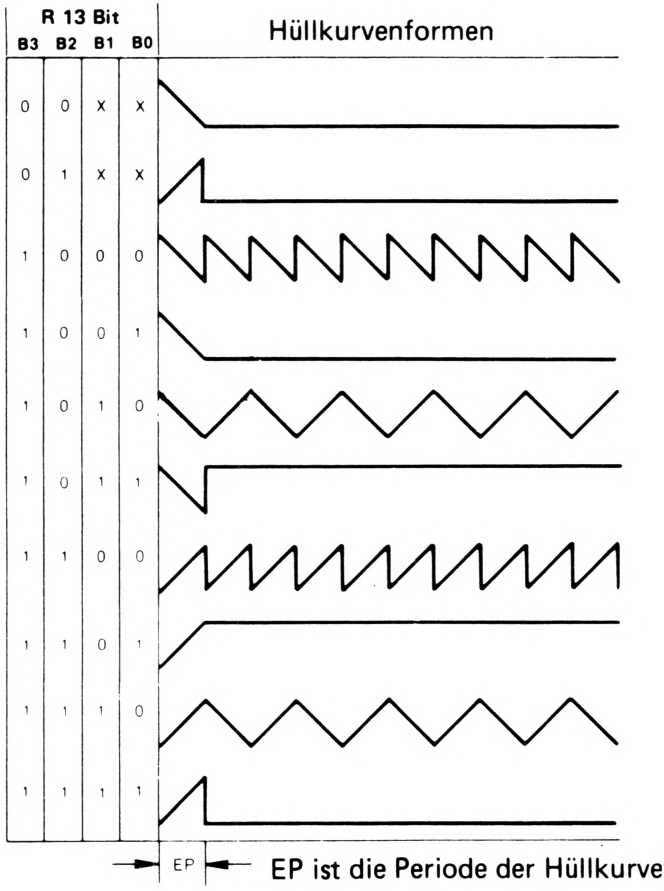
Die untersten 4 Bit des Registers R13 bestimmen die Form der Hüllkurve. Eine Zusammenstellung zeigt Abbildung 7.5.

Die zweite Kurvenform, mit R13 = 04 erzeugt einen mit der Periodendauer EP anschwellenden Ton, dessen Lautstärke dann schlagartig auf Null zurückgeht.

Als Beispiel für die Verwendung des Rauschgenerators und der Hüllkurven soll ein Schuß simuliert werden. Hierzu wird eine mittlere Rauschfrequenz auf allen Kanälen mit abnehmender Lautstärke programmiert. Die Periodendauer der Hüllkurve sei ungefähr 1 Sekunde.

```
100  SOUND  6,&H10  ; mittlere Rauschfrequenz
110  SOUND  7,7     ; auf allen Kanälen
120  SOUND  8,&H10  ; Kanal A
130  SOUND  9,&H10  ; Kanal B
140  SOUND 10,&H10  ; Kanal C
150  SOUND 11,0     ; Periodendauer der
160  SOUND 12,&H10  ; Hüllkurve
170  SOUND 13;0     ; Form der Hüllkurve
```

Für diese beiden Geräusche soll auch noch ein Assembler Programm angegeben werden. Dieses Programm zeigt Abbildung 7.6.



7.5 Hüllkurvenformen

```
1000:                    ; SOUND DEMO
1010:                    ;
1020:                    ; WRTPSG (0093H)
1030:                    ;
1040:                    ; BEIM EINSPRUNG
1050:                    ;       REG#  IN A
1060:                    ;       DATEN IN E
1070:                    ;KEINE AENDERUNG DER REGISTER
1080:                    ;
1090:    D400              ORG 0D400H
1100:                    ;
1110:    0093 =          WRTPSG EQU 0093H
1120:                    ;
1130:                    ;KAMMERTON A
1140:    D400            START:
1150:    D400 3E00        LD A,0
1160:    D402 1EFE        LD E,0FEH
1170:    D404 CD9300      CALL WRTPSG
1180:    D407 3E01        LD A,1
1190:    D409 1E00        LD E,0
1200:    D40B CD9300      CALL WRTPSG
1210:    D40E 3E07        LD A,7
1220:    D410 1E3E        LD E,03EH
1230:    D412 CD9300      CALL WRTPSG
1240:    D415 3E08        LD A,8
1250:    D417 1E0F        LD E,0FH
1260:    D419 CD9300      CALL WRTPSG
1265:    D41C C9          RET
1270:                    ;
1280:                    ;SCHUSS
1290:    D41D             S:
1300:    D41D 2131D4      LD HL,TAB
1310:    D420 3E00        LD A,0
1320:    D422 1E0E        LD E,0EH
1330:    D424            LOOP:
1340:    D424 D5          PUSH DE
1350:    D425 5E          LD E,(HL)
1360:    D426 CD9300      CALL WRTPSG
1370:    D429 D1          POP DE
1380:    D42A 3C          INC A
1390:    D42B 23          INC HL
1400:    D42C 1D          DEC E
1410:    D42D C224D4      JP NZ,LOOP
```

```
1420:    D430 C9          RET
1430:                     ;
1450:    D431 00000000 TAB: DEFB 00,00,00,00,00,00
1460:    D437 10071010      DEFB 10H,7,10H,10H
1470:    D43B 10001000      DEFB 10H,0,10H,0
1480:                     ;
1490:    D43F             END
```

```
D424  LOOP       D41D  S
0093  WRTPSG

D400  START      D431  TAB
```

7.6 Ton A und Schuß in Assembler

Für die Übergabe der Daten an den PSG wird das Unterprogramm WRTPSG mit der Anfangsadresse 0093H verwendet. Dabei ist die Nummer des PSG-Registers in Register A und die Daten in Register E. Das Unterprogramm ändert keine Register des Prozessors.

Beim Kammerton A werden die Daten direkt in die Register geladen, während beim Schuß die Daten in einer Tabelle angegeben sind, die in einer Schleife in den PSG eingelesen wird.

Notizen

Anhang –

Datenblatt 8255

Programmierbarer peripherer Schnittstellen-Baustein

SAB 8255A

SAB 8255A-5 kompatibel mit dem System SAB 8085

24 programmierbare E/A-Anschlüsse

Vollständig TTL-kompatibel

Völlig kompatibel mit allen Siemens Mikroprozessoren

Direkte Bit-Setz- und Rücksetz-Möglichkeiten zur Vereinfachung der Schnittstellen bei Steuerungsanwendungen

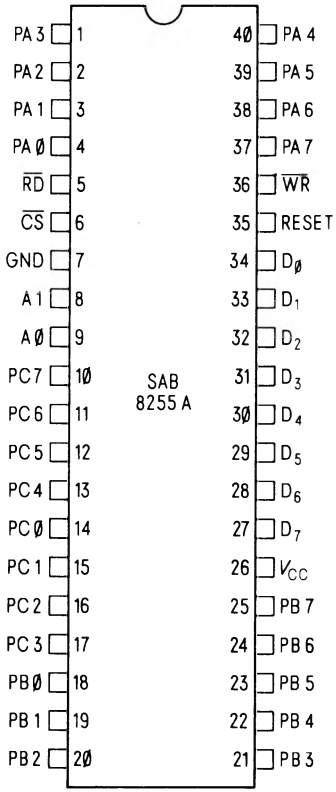
Verbesserte Ausgangstreiberleistung

Verbessertes Zeitverhalten

Nur eine Versorgungsspannung (5 V)

Der SAB 8255A ist ein programmierbarer Mehrzweck-E/A-Baustein für Siemens-Mikroprozessoren. Er hat 24 E/A-Anschlüsse, die in zwei Gruppen von je zwölf Anschlüssen getrennt programmiert und im wesentlichen in drei Betriebsarten benutzt werden können. In der ersten Betriebsart (Betriebsart Ø) kann jede Gruppe von 12 E/A-Anschlüssen in Abschnitten von 4 Anschlüssen als Eingang oder Ausgang programmiert werden. In der zweiten Betriebsart (Betriebsart 1) können acht Leitungen jeder Gruppe als Eingang oder Ausgang programmiert werden. Von den verbleibenden vier Anschlüssen werden drei für den Austausch von Quittungen und für Unterbrechungs-Steuersignale verwendet. Die dritte Betriebsart (Betriebsart 2) kann als Zweiweg-Bus-Betriebsart bezeichnet werden, bei der acht Anschlüsse für einen Zweiweg-Bus eingesetzt werden. Fünf weitere Anschlüsse, von denen einer zur anderen Gruppe gehört, werden in diesem Fall für den Quittungsaustausch benutzt.

Anschlußbelegung

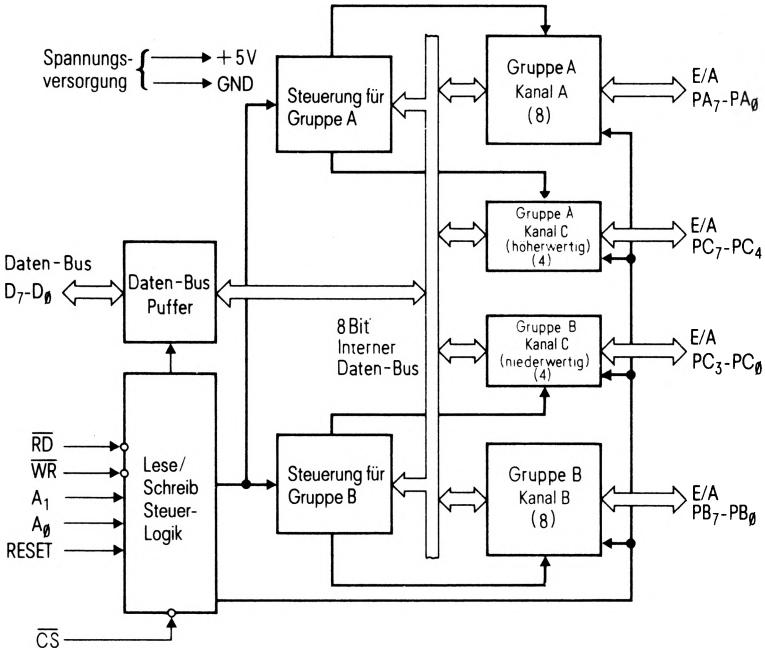


Abmessungen am Schluß des Buches

Anschlußbezeichnungen

$D_Ø - D_7$	Daten-Bus (Zweiweg)
RESET	Rücksetz-Eingang
$\overline{CS}$	Baustein-Auswahl
$\overline{RD}$	Lese-Eingang
$\overline{WR}$	Schreib-Eingang
AØ, A1	Kanal-Adresse
$PA_Ø - PA_7$	Kanal A (Bit Ø bis 7)
$PB_Ø - PB_7$	Kanal B (Bit Ø bis 7)
$PC_Ø - PC_7$	Kanal C (Bit Ø bis 7)
V_{cc}	Versorgungsspannung (+5V)
GND	Masse (0 V)

Blockschaltbild



Funktionsbeschreibung

Allgemeines

Der SAB 8255A ist ein programmierbarer peripherer Schnittstellen-Baustein (PPS) für Siemens-Mikrocomputer-Systeme. Er verbindet als Mehrzweck-E/A-Baustein periphere Geräte mit dem System-Daten-Bus. Die funktionellen Eigenschaften des SAB 8255A werden durch Software bestimmt, so daß normalerweise keine zusätzlichen Logik-Bausteine erforderlich sind, um periphere Geräte oder Schaltungen anzuschließen.

Daten-Bus-Puffer

Ein 8 Bit breiter Zweigweg-Puffer mit drei Ausgangszuständen (tri-state) verbindet den SAB 8255A mit dem System-Daten-Bus. Daten werden bei der Ausführung der Befehle Eingabe (IN) und Ausgabe (OUT) vom Puffer ausgegeben oder empfangen. Steuerwerte und Zustandsinformationen werden ebenfalls durch den Daten-Bus-Puffer übertragen.

Schreib-/Lese- und Steuerlogik

Mit diesem Schaltungsteil werden alle internen und externen Übertragungen von Daten- und Steuer- oder Zustandsworte vorgenommen. Er übernimmt Informationen vom Adreß- und Steuer-Bus des Prozessors und gibt entsprechende Befehle an die Steuerlogik der beiden Gruppen.

($\overline{CS}$)

Baustein-Auswahl (Chip Select): Ein L-Pegel an diesem Eingang ermöglicht den Informationsaustausch zwischen dem SAB 8255A und dem Prozessor.

($\overline{RD}$)

Lesen (Read): Bei einem L-Pegel an diesem Eingang kann der SAB 8255A Daten oder Zustandsinformationen über den Daten-Bus an den Prozessor senden.

($\overline{WR}$)

Schreiben (Write): Ein L-Pegel an diesem Eingang ermöglicht dem Prozessor Daten oder Steuerworte in den SAB 8255A einzuschreiben.

(RESET)

Rücksetzen (Reset): Ein H-Pegel an diesem Eingang setzt alle internen Register einschließlich des Steuerregisters zurück und bringt alle Kanäle (A, B, C) in die Betriebsart Ø Eingabe.

(A_0 und A_1)

Kanalauswahl Ø und Kanalauswahl 1: In Zusammenarbeit mit den $\overline{RD}$ und $\overline{WR}$ Eingängen steuern diese Eingangssignale die Auswahl eines der drei Kanäle oder des Steuerwort-Registers. Normalerweise sind sie mit den niederwertigen Bits (A_0 und A_1) des Adressen-Bus verbunden.

Prinzipielle Betriebsarten

A1	AØ	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	Eingabe (Lesen)
Ø	Ø	Ø	1	Ø	Kanal A → Daten-Bus
Ø	1	Ø	1	Ø	Kanal B → Daten-Bus
1	Ø	Ø	1	Ø	Kanal C → Daten-Bus
					Ausgabe (Schreiben)
Ø	Ø	1	Ø	Ø	Daten-Bus → Kanal A
Ø	1	1	Ø	Ø	Daten-Bus → Kanal B
1	Ø	1	Ø	Ø	Daten-Bus → Kanal C
1	1	1	Ø	Ø	Daten-Bus → Steuerlogik
					Funktionen nicht ausgewählt
x	x	x	x	1	Daten-Bus → hochohmiger Zustand
1	1	Ø	1	Ø	ungültige Bedingung
x	x	1	1	Ø	Daten-Bus → hochohmiger Zustand

Steuerlogik der Gruppen A und B

Die Funktion jedes einzelnen Kanals ist durch Software zu programmieren. Dies geschieht durch Senden eines Steuerwortes an den SAB 8255A, das Informationen, wie „Betriebsart", „Bit setzen", „Bit rücksetzen" und andere Informationen enthält, die die funktionellen Eigenschaften des SAB 8255A bestimmen.

Jeder der Steuerblöcke (Gruppe A und Gruppe B) übernimmt „Befehle" von der Schreib-/Lese- und Steuerlogik, empfängt „Steuerworte" vom internen Daten-Bus und gibt die entsprechenden Befehle an die dazugehörigen Kanäle aus.

Steuerlogik Gruppe A – Kanal A und Kanal C, höherwertige Bits (C7–C4)
Steuerlogik Gruppe B – Kanal B und Kanal C, niederwertige Bits (C3–CØ)

In das Steuerwortregister kann **nur** geschrieben werden. Das Lesen des Steuerwortregisters ist nich möglich.

Kanäle A, B und C (Ports A, B und C)

Der SAB 8255A enthält drei 8-Bit-Kanäle (A, B und C). Sie können durch entsprechende Software-Programmierung, verschiedene Funktionen erfüllen. Darüber hinaus besitzt jeder spezielle Merkmale, die den Anwendungsbereich und die Flexibilität des SAB 8255A weiter vergrößern.

Kanal A: Ein 8-Bit-Zwischenspeicher für Dateneingabe und ein 8-Bit-Zwischenspeicher für Datenausgabe.

Kanal B: Ein 8-Bit-Zwischenspeicher für Dateneingabe oder Datenausgabe.

Kanal C: Ein 8-Bit-Datenausgabe-/Zwischenspeicher-Puffer (keine Zwischenspeicherung für die Eingabe). Dieser Kanal kann durch Steuerung der Betriebsart in zwei 4-Bit-Kanäle aufgeteilt werden. Jeder 4-Bit-Kanal besteht aus einem 4-Bit-Zwischenspeicher und kann für die Steuersignal-Ausgänge in Verbindung mit den Kanälen A und B verwendet werden.

Ausführliche Betriebsbeschreibung

Wahl der Betriebsart

Drei wesentliche Betriebsarten können durch die System-Software festgelegt werden:

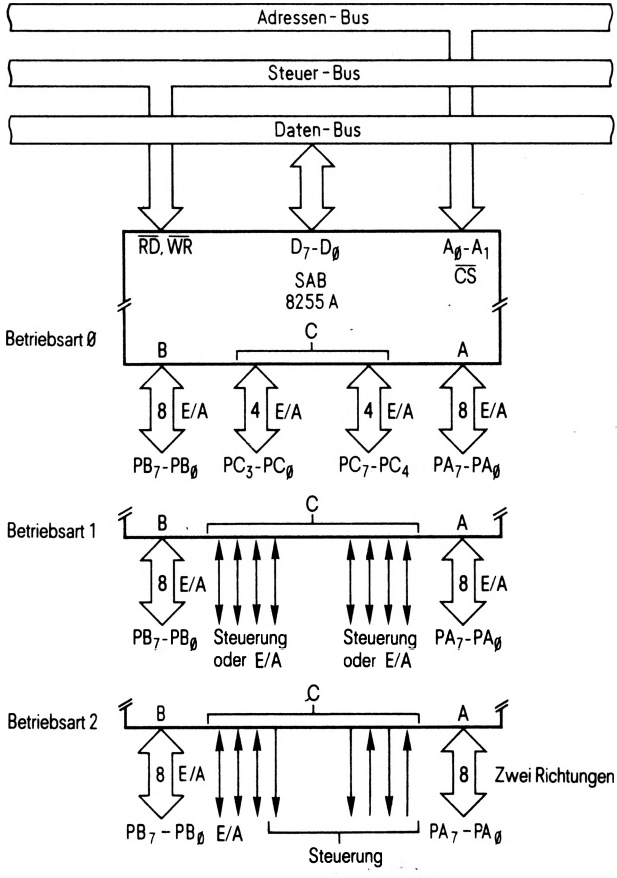
Betriebsart Ø: Einfache Ein-/Ausgabe
Betriebsart 1: Getastete Ein-/Ausgabe
Betriebsart 2: Zweiweg-Bus

Liegt der Rücksetz-Eingang (Reset) auf H-Pegel, werden alle Kanäle in den Eingabezustand gebracht (d.h. die 24 Leitungen haben einen hohen Eingangswiderstand). Nach Ende des Rücksetzsignals bleibt der SAB 8255A im Eingabezustand, ohne daß zusätzliche Einstellungen notwendig sind. Jede der anderen Betriebsarten kann während der Ausführung eines Systemprogramms mit einem einfachen Ausgabe-Befehl ausgewählt werden. Damit kann ein einzelner SAB 8255A verschiedene periphere Geräte mit einem einfachen Software-Verwaltungs-Programm bedienen.

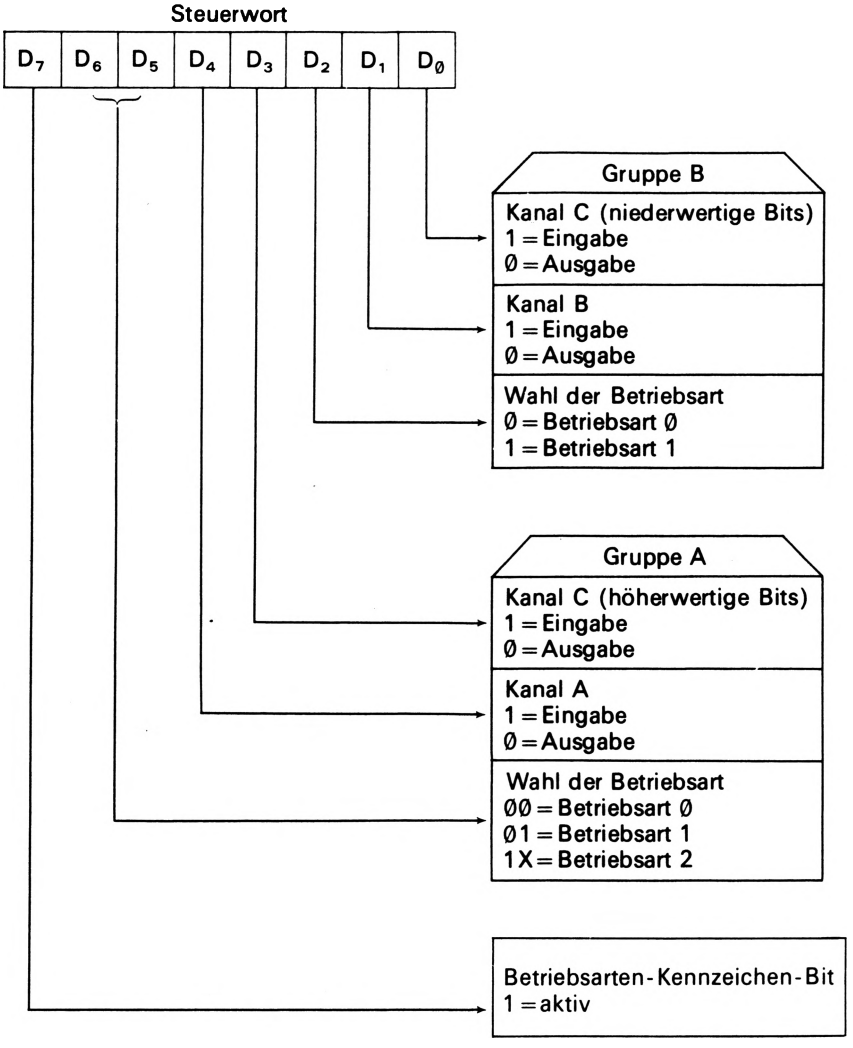
Die Betriebsarten der Kanäle A und B können unabhängig voneinander definiert werden, während Kanal C entsprechend den Erfordernissen der Kanäle A und B in zwei Teile aufgeteilt wird. Wird die Betriebsart gewechselt, werden alle Ausgaberegister einschließlich des Zustands-Flipflops zurückgesetzt. Betriebsarten können kombiniert werden, so daß ihre funktionelle Definition praktisch auf jede E/A-Struktur hin „maßgeschneidert" werden kann. Zum Beispiel kann die Gruppe B für die Betriebsart Ø programmiert sein, um das Schließen von Schaltern zu überwachen oder Rechenergebnisse anzuzeigen, während die Gruppe A für die Betriebsart 1 programmiert sein könnte, um eine Tastatur oder einen Lochstreifenleser durch eine Unterbrechungssteuerung zu überwachen.

Die möglichen Kombinationen von Betriebsarten, erscheinen auf den ersten Blick verwirrend. Aber schon nach einem kurzen Überblick über die gesamte Arbeitsweise des Bausteins wird die einfache und einleuchtende E/A-Struktur erkennbar.

Definition der Betriebsarten und der Bus-Schnittstelle



Format-Definition für die Betriebsart-Wahl



Einzelbit-Setzen/Rücksetzen

Jedes der 8 Bit des Kanals C kann durch einen Ausgabebefehl (OUT) an das Steuerwortregister gesetzt oder rückgesetzt werden. Diese Eigenschaft verringert den Software-Aufwand in regelungstechnischen Anwendungen.

Wird Kanal C für Zustands- und Steuerzwecke für Kanal A oder B verwendet, können die Bits durch die Operation „Bit Setzen/Rücksetzen", wie bei einem Daten-Ausgabekanal, gesetzt oder rückgesetzt werden. Die Bits D_1–D_3 des Steuerworts werden zur Bit-Auswahl benutzt, während Bit $D_\emptyset$ das ausgewählte Bit von Kanal C setzt oder rücksetzt.

Unterbrechungs-Steuerungs-Funktionen

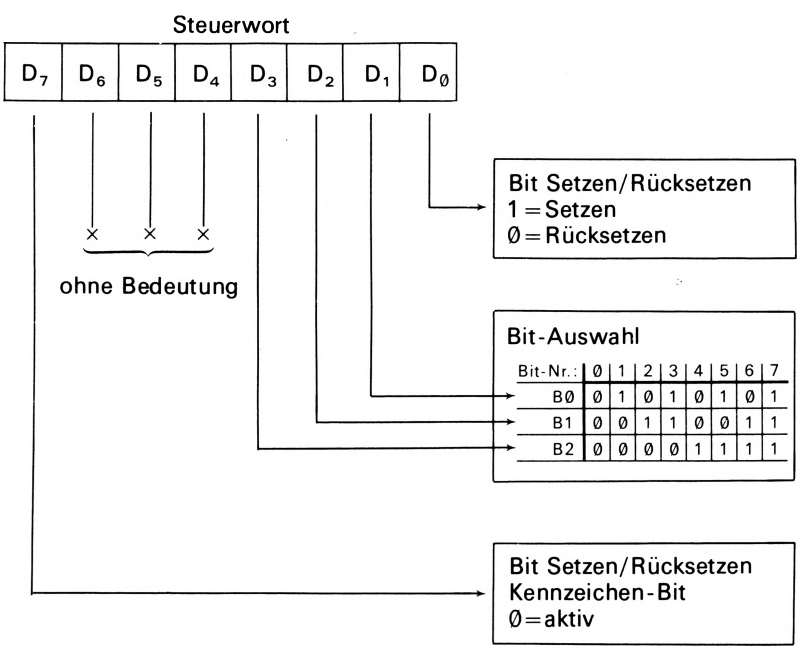
Ist der SAB 8255A für Betriebsart 1 oder 2 programmiert, stehen Steuersignale zur Verfügung, die als Unterbrechungs-Anforderungs-Signale für den Prozessor benutzt werden können. Die vom Kanal C erzeugten Unterbrechungs-Anforderungs-Signale können durch Setzen oder Rücksetzen des dazugehörigen INTE-Flipflops gesperrt oder freigegeben werden, indem die Funktion „Bit Setzen/Rücksetzen" des Kanals C angesprochen wird.

Definitionen für das INTE-Flipflop:

(Bit-SET) – INTE ist gesetzt – Unterbrechung freigegeben
(Bit-RESET) – INTE ist rückgesetzt – Unterbrechung gesperrt

Anmerkung: Alle Maskierungs-Flipflops werden bei der Auswahl der Betriebsart und beim Rücksetzen des Bausteins automatisch rückgesetzt.

Format für Bit Setzen/Rücksetzen



Betriebsarten

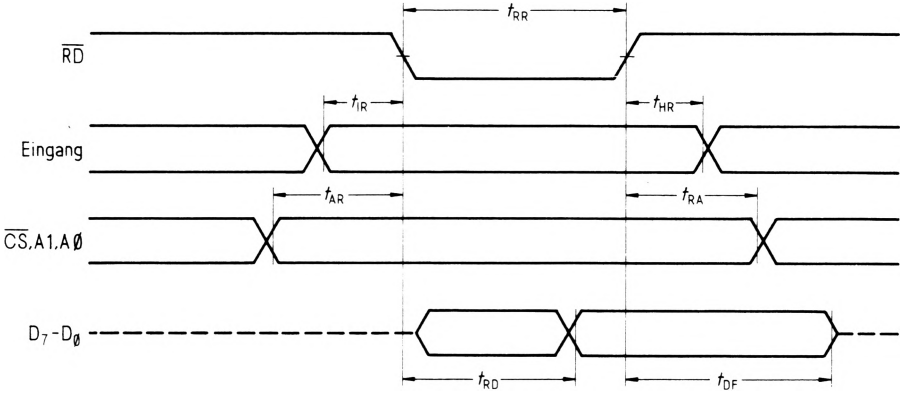
Betriebsart Ø (einfache Ein-/Ausgabe)

Diese Funktions-Anordnung ermöglicht eine einfache Eingabe und Ausgabe für jeden der drei Kanäle. Es ist kein Quittungsaustausch erforderlich, denn Daten werden einfach in den ausgewählten Kanal geschrieben oder aus ihm gelesen.

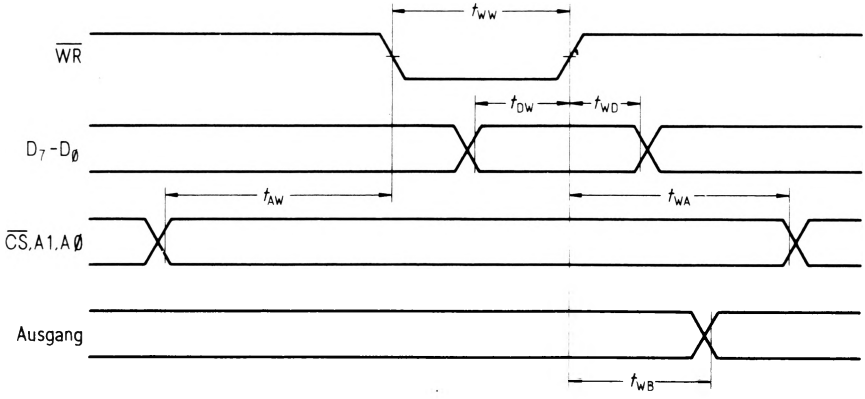
Prinzipielle Funktionsdefinitionen der Betriebsart Ø:

- Zwei 8-Bit-Kanäle und zwei 4-Bit-Kanäle
- Jeder Kanal kann Eingang oder Ausgang sein
- Ausgänge haben Zwischenspeicher
- Eingänge arbeiten ohne Zwischenspeicher
- 16 verschiedene Ein-/Ausgabe Kombinationen sind bei dieser Betriebsart möglich.

Prinzipieller Zeitverlauf der Eingabe (D_7-$D_Ø$ folgen dem Eingang, keine Zwischenspeicherung)



Prinzipieller Zeitverlauf der Ausgabe (Ausgänge mit Zwischenspeicherung)



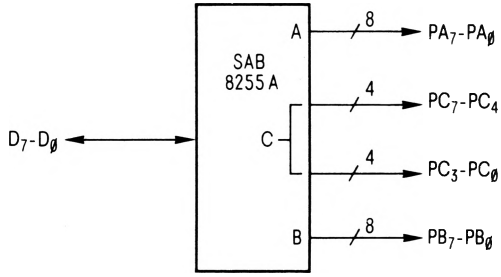
Definition der Kanäle für die Betriebsart Ø

A		B		Gruppe A			Gruppe B	
D_4	D_3	D_1	$D_Ø$	Kanal A	Kanal C (höher-wertige Bits)	Nr.	Kanal B	Kanal C (nieder-wertige Bits)
Ø	Ø	Ø	Ø	Ausgang	Ausgang	Ø	Ausgang	Ausgang
Ø	Ø	Ø	1	Ausgang	Ausgang	1	Ausgang	Eingang
Ø	Ø	1	Ø	Ausgang	Ausgang	2	Eingang	Ausgang
Ø	Ø	1	1	Ausgang	Ausgang	3	Eingang	Eingang
Ø	1	Ø	Ø	Ausgang	Eingang	4	Ausgang	Ausgang
Ø	1	Ø	1	Ausgang	Eingang	5	Ausgang	Eingang
Ø	1	1	Ø	Ausgang	Eingang	6	Eingang	Ausgang
Ø	1	1	1	Ausgang	Eingang	7	Eingang	Eingang
1	Ø	Ø	Ø	Eingang	Ausgang	8	Ausgang	Ausgang
1	Ø	Ø	1	Eingang	Ausgang	9	Ausgang	Eingang
1	Ø	1	Ø	Eingang	Ausgang	1Ø	Eingang	Ausgang
1	Ø	1	1	Eingang	Ausgang	11	Eingang	Eingang
1	1	Ø	Ø	Eingang	Eingang	12	Ausgang	Ausgang
1	1	Ø	1	Eingang	Eingang	13	Ausgang	Eingang
1	1	1	Ø	Eingang	Eingang	14	Eingang	Ausgang
1	1	1	1	Eingang	Eingang	15	Eingang	Eingang

Anordnungen in der Betriebsart Ø:

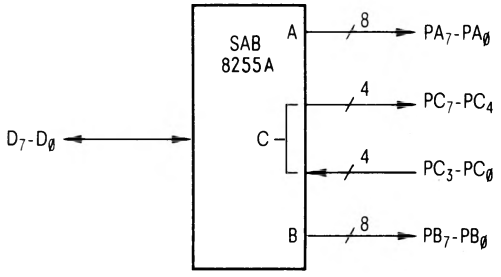
Steuerwort Nr. Ø

D_7	D_6	D_5	D_4	D_3	D_2	D_1	$D_Ø$
1	Ø	Ø	Ø	Ø	Ø	Ø	Ø



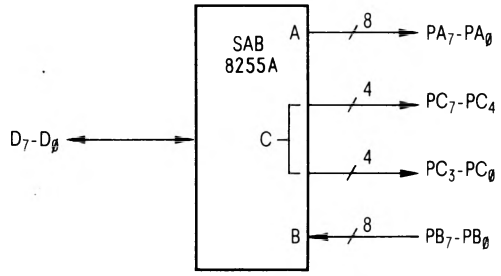
Steuerwort Nr. 1

D_7	D_6	D_5	D_4	D_3	D_2	D_1	$D_Ø$
1	Ø	Ø	Ø	Ø	Ø	Ø	1



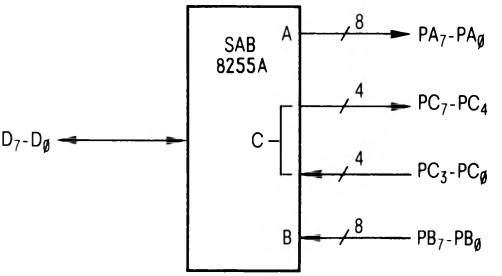
Steuerwort Nr. 2

D_7	D_6	D_5	D_4	D_3	D_2	D_1	$D_Ø$
1	Ø	Ø	Ø	Ø	Ø	1	Ø



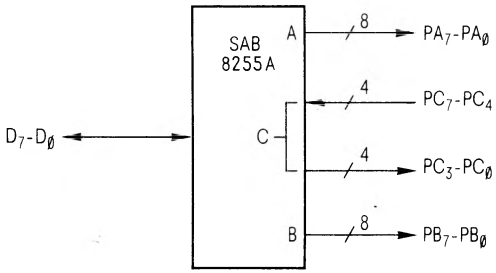
Steuerwort Nr. 3

D_7	D_6	D_5	D_4	D_3	D_2	D_1	$D_Ø$
1	Ø	Ø	Ø	Ø	Ø	1	1



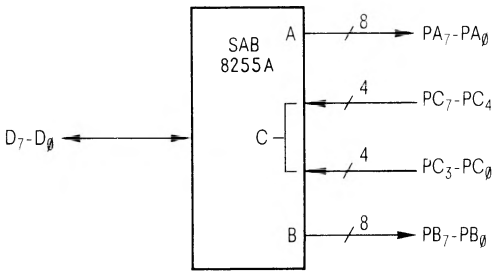
Steuerwort Nr. 4

D_7	D_6	D_5	D_4	D_3	D_2	D_1	$D_\emptyset$
1	Ø	Ø	Ø	1	Ø	Ø	Ø



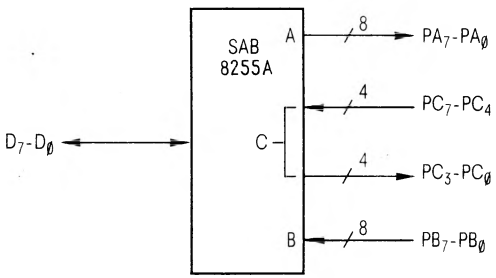
Steuerwort Nr. 5

D_7	D_6	D_5	D_4	D_3	D_2	D_1	$D_\emptyset$
1	Ø	Ø	Ø	1	Ø	Ø	1



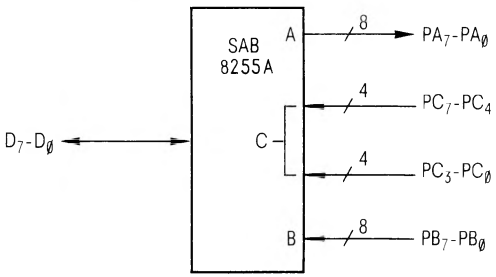
Steuerwort Nr. 6

D_7	D_6	D_5	D_4	D_3	D_2	D_1	$D_\emptyset$
1	Ø	Ø	Ø	1	Ø	1	Ø



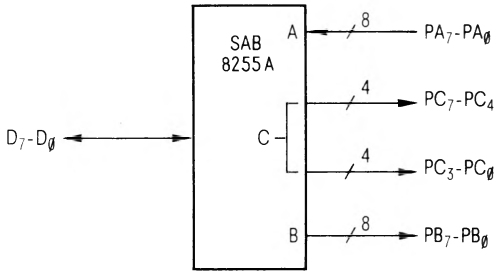
Steuerwort Nr. 7

D_7	D_6	D_5	D_4	D_3	D_2	D_1	$D_\emptyset$
1	Ø	Ø	Ø	1	Ø	1	1



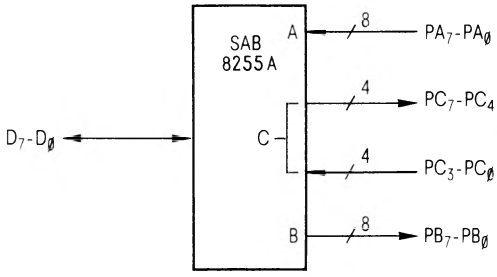
Steuerwort Nr. 8

D_7	D_6	D_5	D_4	D_3	D_2	D_1	$D_\emptyset$
1	Ø	Ø	1	Ø	Ø	Ø	Ø



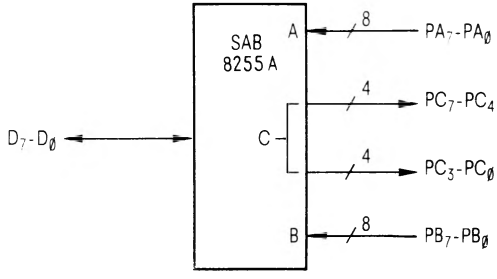
Steuerwort Nr. 9

D_7	D_6	D_5	D_4	D_3	D_2	D_1	$D_\emptyset$
1	Ø	Ø	1	Ø	Ø	Ø	1



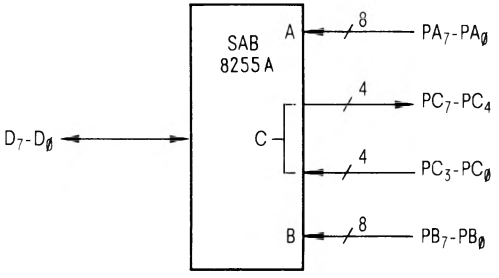
Steuerwort Nr. 10

D_7	D_6	D_5	D_4	D_3	D_2	D_1	$D_\emptyset$
1	Ø	Ø	1	Ø	Ø	1	Ø



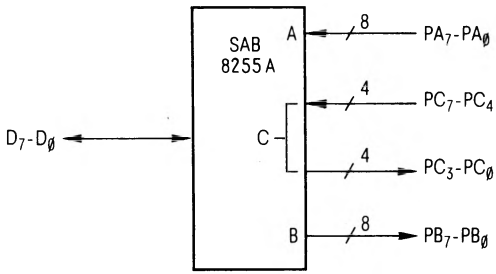
Steuerwort Nr. 11

D_7	D_6	D_5	D_4	D_3	D_2	D_1	$D_\emptyset$
1	Ø	Ø	1	Ø	Ø	1	1



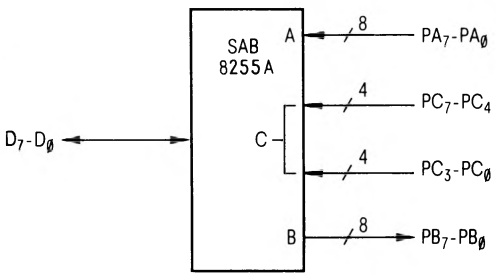
Steuerwort Nr. 12

D_7	D_6	D_5	D_4	D_3	D_2	D_1	$D_\emptyset$
1	Ø	Ø	1	1	Ø	Ø	Ø



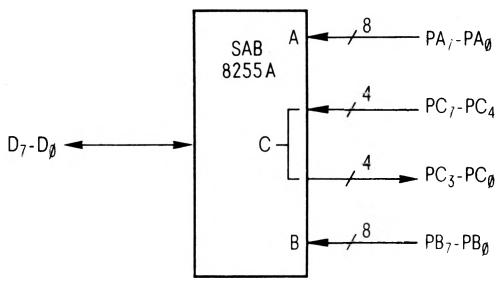
Steuerwort Nr. 13

D_7	D_6	D_5	D_4	D_3	D_2	D_1	$D_\emptyset$
1	Ø	Ø	1	1	Ø	Ø	1



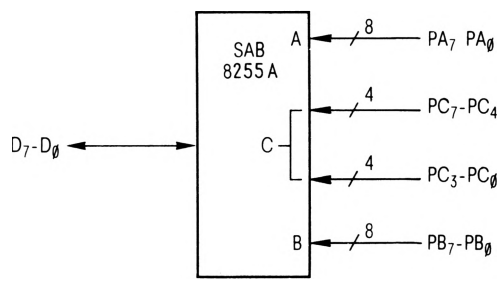
Steuerwort Nr. 14

D_7	D_6	D_5	D_4	D_3	D_2	D_1	$D_\emptyset$
1	Ø	Ø	1	1	Ø	1	Ø



Steuerwort Nr. 15

D_7	D_6	D_5	D_4	D_3	D_2	D_1	$D_\emptyset$
1	Ø	Ø	1	1	Ø	1	1



Betriebsart 1 (getastete Ein-/Ausgabe)

Diese Funktions-Anordnung dient zum Austausch von E/A-Daten zu oder von einem ausgewählten Kanal in Verbindung mit Abtastimpulsen oder „Quittungs"-Signalen. Kanal A und Kanal B benützen in der Betriebsart 1 die Leitungen des Kanals C, um diese „Quittungs"-Signale zu erzeugen oder zu empfangen.

Prinzipielle Funktionsdefinitionen der Betriebsart 1:

- Zwei Gruppen (Gruppe A und Gruppe B)
- Jede Gruppe umfaßt einen 8-Bit-Datenkanal und einen 4-Bit-Steuer-/Datenkanal.
- Der 8-Bit-Datenkanal kann entweder als Eingang oder Ausgang verwendet werden. Die Ein- und Ausgangsdaten werden zwischengespeichert.
- Der 4-Bit-Kanal wird für Steuer- und Zustandszwecke für die 8-Bit-Datenkanäle benutzt. Anschlüsse, die nicht für Steuer- und Zustandszwecke benötigt werden, sind für Ein-/Ausgabe-Operationen frei. Eingabedaten werden in Kanal C nicht zwischengespeichert.

Definition der Eingangs-Steuersignale

$\overline{\text{STB}}$ (Übernahmesignal-Eingang)

L-Pegel an diesem Eingang bewirkt, daß Daten in den Eingangs-Zwischenspeicher geladen werden.

IBF (Eingabe-Puffer-Flipflop voll)

H-Pegel an diesem Ausgang zeigt an, daß die Daten in den Eingangs-Zwischenspeicher geladen wurden; dies entspricht einer Quittung. IBF wird durch die fallende Flanke des STB-Eingangs gesetzt und durch die ansteigende Flanke des RD-Eingangs rückgesetzt.

INTR (Unterbrechungs-Anforderung)

H-Pegel an diesem Ausgang kann dazu ausgenutzt werden, das Hauptprogramm des Mikroprozessors zu unterbrechen, wenn ein Eingabegerät bedient werden soll. INTR wird durch die ansteigende Flanke von $\overline{\text{STB}}$ gesetzt, falls IBF auf „Eins" und INTE auf „Eins" gesetzt sind. Es wird rückgesetzt durch die fallende Flanke von $\overline{\text{RD}}$. Dieser Vorgang ermöglicht es dem Eingabegerät, vom Mikroprozessor, durch einfaches Eintasten seiner Daten in den Kanal bedient zu werden.

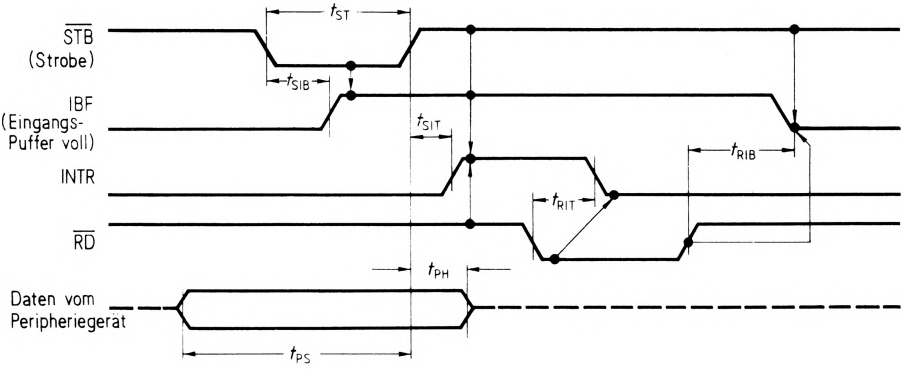
INTE A

Wird gesteuert durch Bit setzen/rücksetzen von PC_4.

INTE B

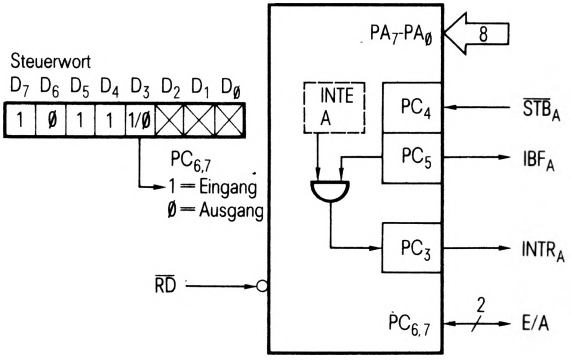
Wird gesteuert durch Bit setzen/rücksetzen von PC_2.

Impulsdiagramm für Betriebsart 1, Eingabe

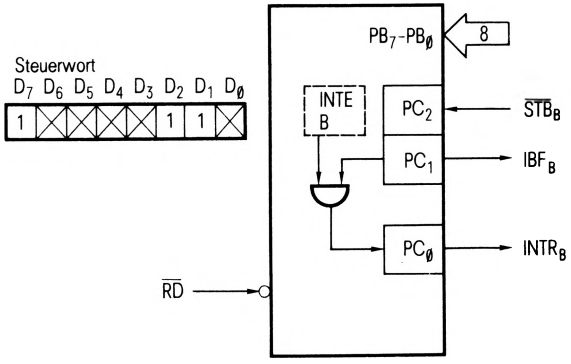


Betriebsart 1 – Eingabe

Kanal A



Kanal B



Definition der Ausgangs-Steuersignale

$\overline{OBF}$ (Ausgabe-Puffer-Flipflop voll)

Der $\overline{OBF}$-Ausgang nimmt L-Pegel an, wenn der Mikroprozessor Daten in den ausgewählten Kanal geschrieben hat. Das $\overline{OBF}$-Flipflop wird von der ansteigenden Flanke des $\overline{WR}$-Eingangs gesetzt und von der fallenden Flanke des $\overline{ACK}$-Signals rückgesetzt.

$\overline{ACK}$ (Quittungs-Eingang)

L-Pegel an diesem Eingang zeigt dem SAB 8255A an, daß Daten von Kanal A oder B übernommen werden, d.h., dieses Signal ist eine Antwort des Peripherie-Gerätes, welches den Empfang des vom Mikroprozessor ausgegebenen Datums bestätigt.

INTR (Unterbrechungs-Anforderung)

H-Pegel an diesem Ausgang kann zum Unterbrechen des Mikroprozessors verwendet werden, wenn das Ausgabegerät die vom Mikroprozessor ausgesendeten Daten übernommen hat. INTR wird durch die ansteigende Flanke von $\overline{ACK}$ gesetzt, wenn gleichzeitig $\overline{OBF}$ auf „Eins" und INTE auf „Eins" gesetzt sind. Es wird mit der fallenden Flanke von $\overline{WR}$ zurückgesetzt.

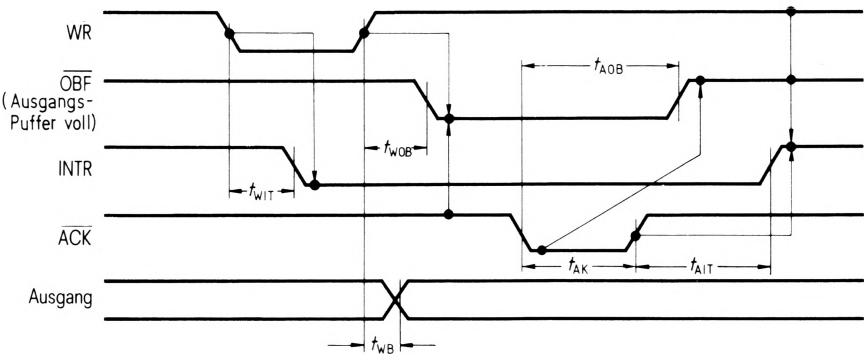
INTE A

Wird gesteuert durch Bit setzen/rücksetzen von PC_6.

INTE B

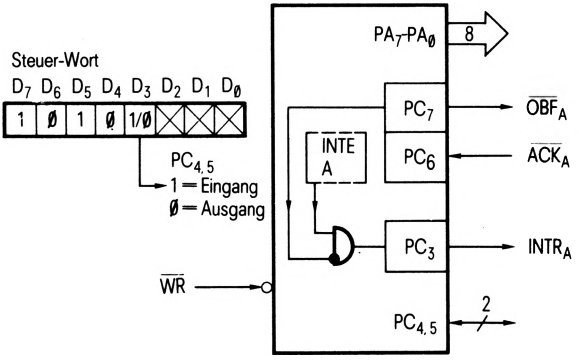
Wird gesteuert durch Bit setzen/rücksetzen von PC_2.

Impulsdiagramm für Betriebsart 1, Ausgabe

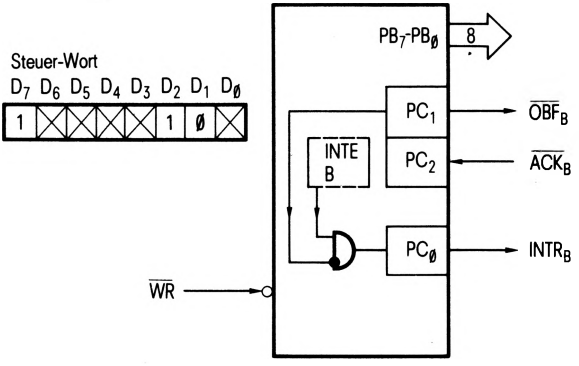


Betriebsart 1 – Ausgabe

Kanal A

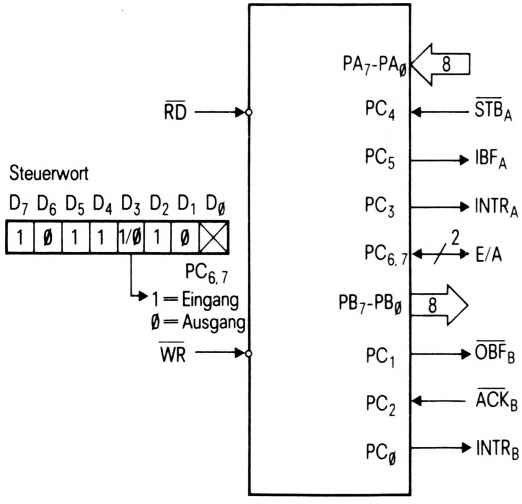


Kanal B

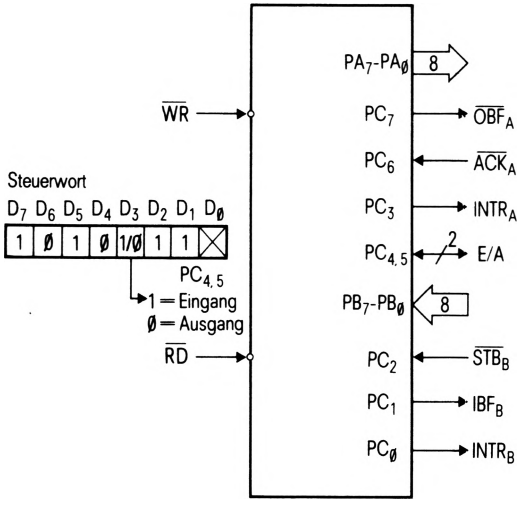


Kombinationen in der Betriebsart 1

Kanal A und B können in der Betriebsart 1 unabhängig voneinander als Eingänge oder Ausgänge definiert werden; dadurch ist eine Vielzahl getasteter E/A-Anwendungen möglich.



Kanal A (getasteter Eingang)
Kanal B (getasteter Ausgang)



Kanal A (getasteter Ausgang)
Kanal B (getasteter Eingang)

Betriebsart 2 (getastete Zweiweg-Bus-Ein-/Ausgabe)

Diese funktionelle Anordnung ermöglicht den Datenaustausch mit einem Peripherie-Gerät oder einer Schaltung auf einem einzigen 8-Bit Bus, über den Daten gesendet und empfangen werden (Zweiweg-Bus-Ein-/Ausgabe). Der richtige Datenfluß auf dem Bus wird in ähnlicher Weise wie bei der Betriebsart 1 durch Quittungs-Signale gewährleistet. Die Erzeugung von Unterbrechungen und die Funktionen Sperren/Freigeben stehen ebenfalls zur Verfügung.

Prinzipielle Funktionsdefinitionen der Betriebsart 2:

- **Nur** in Gruppe A verwendet
- Ein 8-Bit-Zweiweg-Bus-Kanal (Kanal A) und ein 5-Bit-Steuerkanal (Kanal C)
- Eingänge und Ausgänge verfügen über Pufferspeicher.
- Der 5-Bit-Steuerkanal (Kanal C) wird für Steuer- und Zustandszwecke für den 8-Bit-Zweiweg-Bus-Kanal (Kanal A) verwendet.

Definition der Steuersignale für die Zweiweg-Bus-Ein-/Ausgabe

INTR (Unterbrechungs-Anforderung)

H-Pegel an diesem Ausgang kann bei Eingaben und Ausgaben zum Unterbrechen des Hauptprogramms benützt werden.

Ausgabebetrieb

$\overline{\text{OBF}}$ (Ausgabepuffer-geladen)

Der $\overline{\text{OBF}}$ Ausgang nimmt L-Pegel an, wenn der Mikroprozessor Daten in den Kanal A geschrieben hat.

$\overline{\text{ACK}}$ (Quittung)

L-Pegel an diesem Eingang gibt den Tri-state-Ausgabepuffer des Kanals A zum Senden von Daten frei. Sonst befindet sich der Ausgabepuffer im hochohmigen Zustand.

INTE 1 (INTE-Flipflop im Zusammenhang mit $\overline{\text{OBF}}$)

Wird gesteuert durch Bit setzen/rücksetzen von PC_6.

Eingabebetrieb

$\overline{STB}$ (Tasteingang)

L-Pegel an diesem Eingang lädt Daten in den Eingabe-Zwischenspeicher.

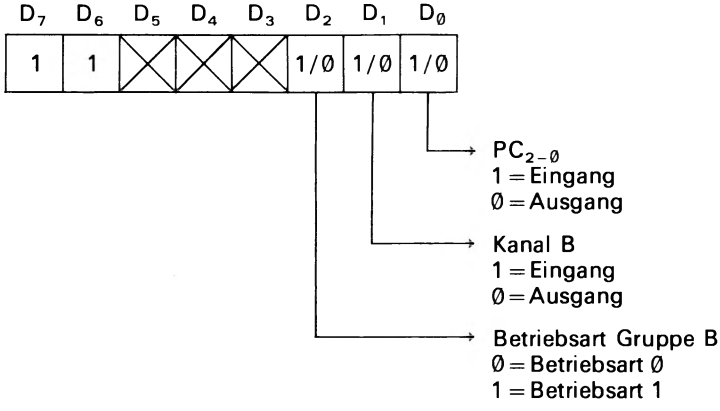
IBF (Eingabepuffer-Flipflop voll)

H-Pegel an diesem Ausgang zeigt an, daß Daten in den Eingabe-Zwischenspeicher geladen wurden.

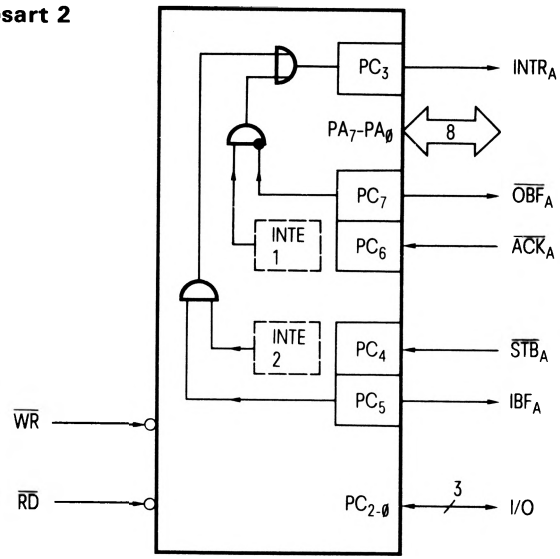
INTE 2 (INTE-Flipflop im Zusammenhang mit IBF)

Wird gesteuert durch Bit setzen/rücksetzen von PC_4.

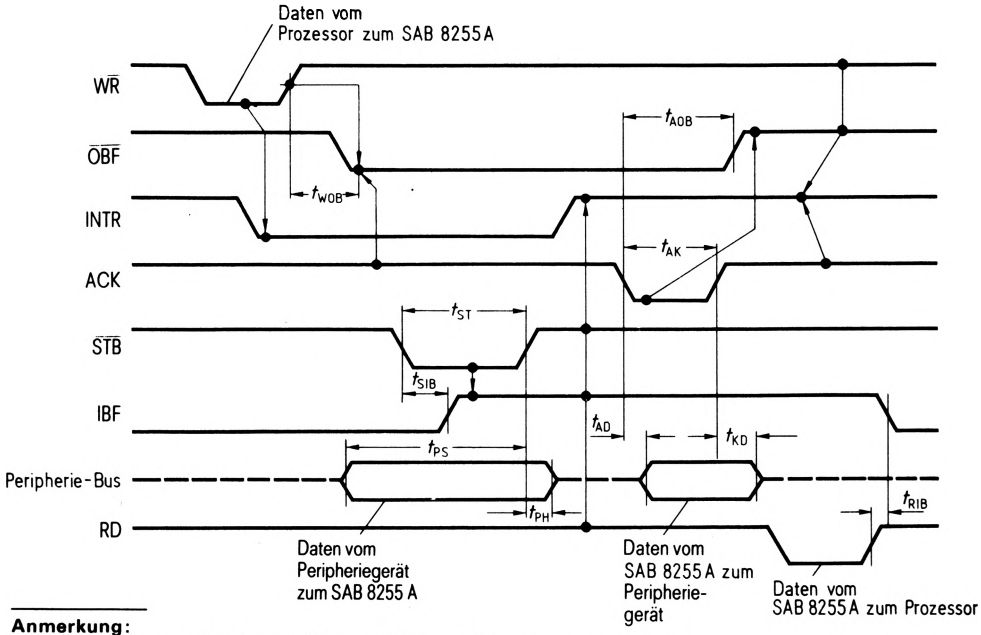
Steuerwort für Betriebsart 2



Betriebsart 2



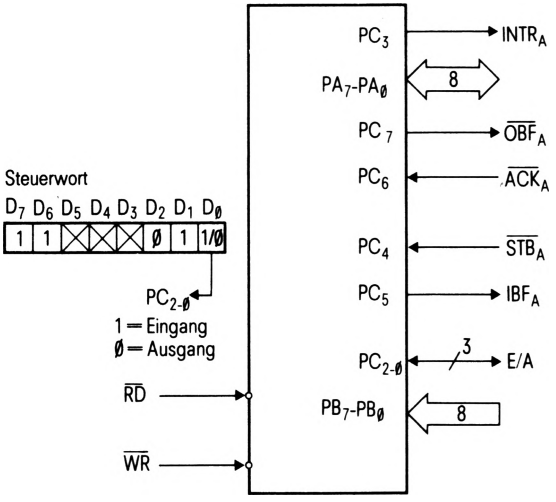
Impulsdiagramm für Betriebsart 2 (Zweiweg-Bus)



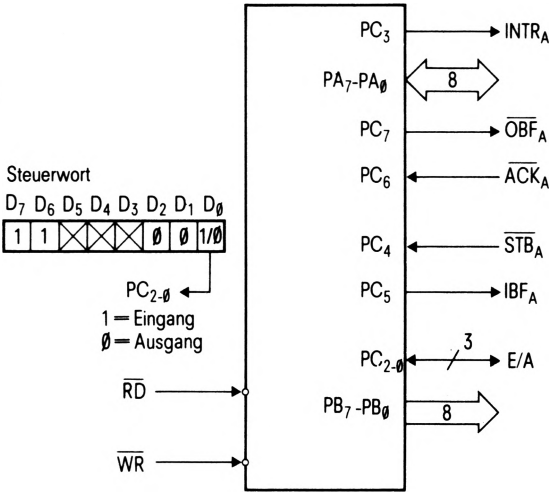
Anmerkung:
Jede Signalfolge bei der $\overline{WR}$ vor $\overline{ACK}$ und $\overline{STB}$ vor $\overline{RD}$ erscheint, ist zulässig.
($INTR = IBF \cdot \overline{MASK} \cdot \overline{STB} \cdot \overline{RD} + \overline{OBF} \cdot \overline{MASK} \cdot \overline{ACK} \cdot \overline{WR}$)

Betriebsart 2 – Kombinationen

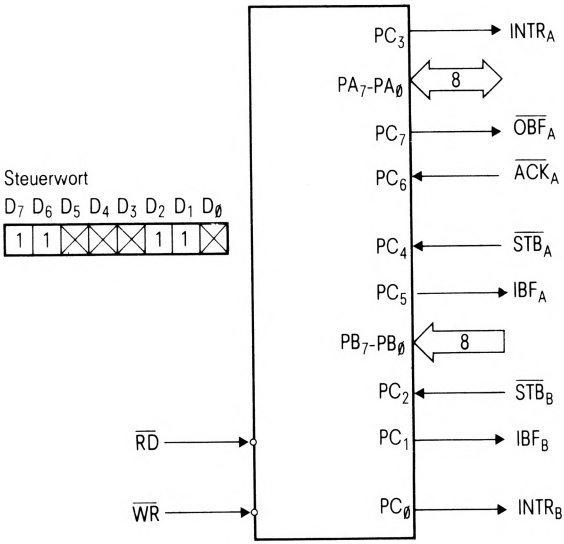
Betriebsarten 2 und Ø (Eingabe)



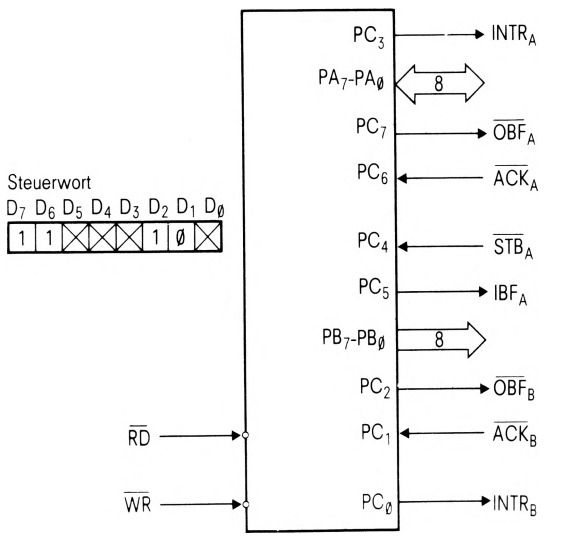
Betriebsarten 2 und Ø (Ausgabe)



Betriebsarten 2 und 1 (Eingabe)



Betriebsarten 2 und 1 (Ausgabe)



Übersichtstabelle der Betriebsartendefinition

	Betriebsart Ø		Betriebsart 1		Betriebsart 2	
	Eingang	Ausgang	Eingang	Ausgang	Nur Gruppe A	
$PA_Ø$	Eingabe	Ausgabe	Eingabe	Ausgabe	↔	
PA_1	Eingabe	Ausgabe	Eingabe	Ausgabe	↔	
PA_2	Eingabe	Ausgabe	Eingabe	Ausgabe	↔	
PA_3	Eingabe	Ausgabe	Eingabe	Ausgabe	↔	
PA_4	Eingabe	Ausgabe	Eingabe	Ausgabe	↔	
PA_5	Eingabe	Ausgabe	Eingabe	Ausgabe	↔	
PA_6	Eingabe	Ausgabe	Eingabe	Ausgabe	↔	
PA_7	Eingabe	Ausgabe	Eingabe	Ausgabe	↔	
$PB_Ø$	Eingabe	Ausgabe	Eingabe	Ausgabe	–	Nur Betriebsarten Ø und 1
PB_1	Eingabe	Ausgabe	Eingabe	Ausgabe	–	
PB_2	Eingabe	Ausgabe	Eingabe	Ausgabe	–	
PB_3	Eingabe	Ausgabe	Eingabe	Ausgabe	–	
PB_4	Eingabe	Ausgabe	Eingabe	Ausgabe	–	
PB_5	Eingabe	Ausgabe	Eingabe	Ausgabe	–	
PB_6	Eingabe	Ausgabe	Eingabe	Ausgabe	–	
PB_7	Eingabe	Ausgabe	Eingabe	Ausgabe	–	
$PC_Ø$	Eingabe	Ausgabe	$INTR_B$	$\overline{INTR}_B$	E/A	
PC_1	Eingabe	Ausgabe	IBF_B	$\overline{OBF}_B$	E/A	
PC_2	Eingabe	Ausgabe	$\overline{STB}_B$	$\overline{ACK}_B$	E/A	
PC_3	Eingabe	Ausgabe	$INTR_A$	$INTR_A$	$INTR_A$	
PC_4	Eingabe	Ausgabe	$\overline{STB}_A$	E/A	$\overline{STB}_A$	
PC_5	Eingabe	Ausgabe	IBF_A	E/A	IBF_A	
PC_6	Eingabe	Ausgabe	E/A	$\overline{ACK}_A$	$\overline{ACK}_A$	
PC_7	Eingabe	Ausgabe	E/A	$\overline{OBF}_A$	$\overline{OBF}_A$	

Hinweise für Kombinationen spezieller Betriebsarten

Bei verschiedenen Kombinationen von Betriebsarten werden nicht alle Bits des Kanals C für Steuer- oder Zustandszwecke verwendet. Die übrigen Bits können wie folgt eingesetzt werden:

Bei Programmierung als Eingang:

Der Zugriff zu allen Eingabeleitungen ist während einer normalen Kanal C-Ladeoperation möglich.

Bei Programmierung als Ausgang:

Der Zugriff zu den höherwertigen Bits des Kanals C (PC_7–PC_4) muß einzeln mit der Bit setzen/rücksetzen-Funktion erfolgen.

Der Zugriff zu den niederwertigen Bits des Kanals C (PC_3–$PC_Ø$) kann mit der Bit setzen/rücksetzen-Funktion oder durch Schreiben in den Kanal C erfolgen.

Ausgangsbelastbarkeit der Kanäle B und C:

Jeder Satz von acht Ausgabepuffern, die aus Kanal B und C wahlfrei ausgewählt werden, kann bei 1,5 V 1 mA liefern. Diese Eigenschaft ermöglicht es dem SAB 8255A direkt Darlington-Treiber und Hochspannungsanzeigeeinheiten zu betreiben, die solche Ströme benötigen.

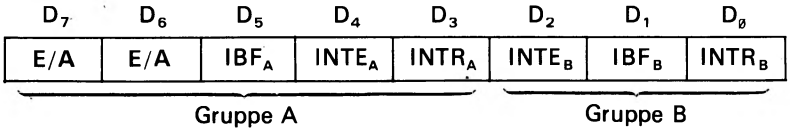
Lesen des Zustands von Kanal C

Kanal C überträgt in der Betriebsart Ø Daten von oder zu Peripherie-Geräten. Wenn der SAB 8255A für die Betriebsarten 1 oder 2 programmiert ist, erzeugt oder empfängt der SAB 8255A Quittungs-Signale der Peripherie-Geräte. Der Programmierer kann durch Auswerten des Inhalts von Kanal C, den Zustand jedes peripheren Gerätes testen oder überprüfen, um den Programmablauf entsprechend zu ändern.

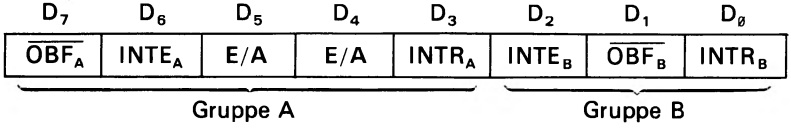
Das Lesen der Zustandsinformation von Kanal C wird ohne einen speziellen Befehl vorgenommen. Ein normaler C-Kanal-Lesevorgang führt diese Funktion aus.

Format des Zustandswortes für die Betriebsarten Ø oder 1

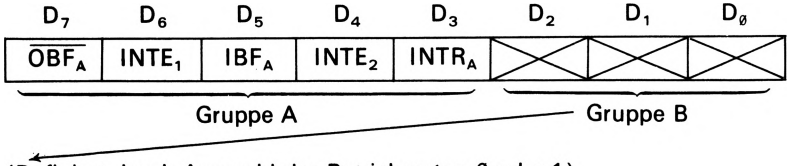
Eingangs-Anordnungen



Ausgangs-Anordnungen



Format des Zustandswortes für die Betriebsart 2



(Definiert durch Auswahl der Betriebsarten Ø oder 1)

Grenzdaten [1])

Betriebstemperatur	0 bis + 70 °C
Lagertemperatur	− 65 bis +150 °C
Spannung an jedem Anschluß gegen Masse	−0,5 bis +7 V
Verlustleistung	1 W

Statische Kenndaten und Betriebsbedingungen

T_U = 0 bis 70 °C; V_{CC} = +5 V ± 5%; GND = 0 V (wenn nicht anders angegeben)

Symbol	Bezeichnung	Grenzwerte		Einheit	Prüfbedingung
		min.	max.		
V_{IL}	L-Eingangsspannung	−0,5	0,8	V	–
V_{IH}	H-Eingangsspannung	2,0	V_{CC}		
V_{OL}(DB)	L-Ausgangsspannung (Datenbus)	–	0,45		I_{OL} = 2,5 mA
V_{OL}(PER)	L-Ausgangsspannung (Peripherie-Port)				I_{OL} = 1,7 mA
V_{OH}(DB)	H-Ausgangsspannung (Datenbus)	2,4	–		I_{OH} = −400 µA
V_{OH}(PER)	H-Ausgangsspannung (Peripherie-Port)				I_{OH} = −200 µA
I_{DAR} [2])	Darlington-Treiberstrom	−1,0	−4,0	mA	R_{EXT} = 750 Ω; V_{EXT} = 1,5 V
I_{CC}	Stromaufnahme, V_{CC}	–	120		
I_{IL}	Eingangs-Laststrom		±10	µA	V_{IN} = V_{CC} bis 0 V
I_{OFL}	Ausgangs-Leckstrom im hochohmigen Zustand				V_{OUT} = V_{CC} bis 0 V

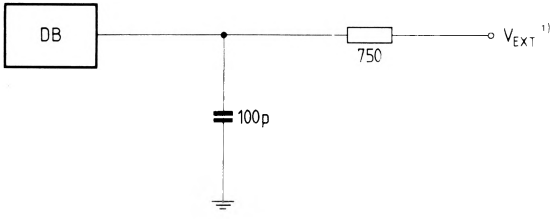
[1]) Die Überschreitung der angegebenen Grenzdaten kann zu Dauerschäden am Baustein führen.
[2]) Verfügbar an jedem Anschluß von Kanal B und C.

Kapazitäten

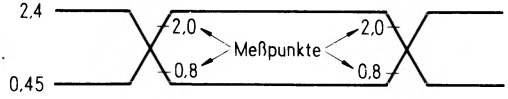
$T_U = 25\,°C$; $V_{CC} = GND = 0\,V$

Symbol	Bezeichnung	Grenzwerte (max.)	Einheit	Prüfbedingung
C_{IN}	Eingangskapazität	10	pF	$fc = 1$ MHz
$C_{I/O}$	Ein-/Ausgangs-Kapazität	20		freie Anschlüsse auf Masse gelegt

Meßschaltung für den Datenbus



Spannungswerte für die Schaltzeitbestimmung



1) V_{EXT} ist eine Anzahl von Testspannungen, um die Spezifikationen zu überprüfen.

Schaltzeiten

$T_U = 0$ bis 70 °C; $V_{CC} = +5\ V \pm 5\%$; GND = 0 V

System-Bus Werte

Lesen

Symbol	Bezeichnung	Grenzwerte				Einheit
		8255A		8255A-5 [2])		
		min.	max.	min.	max.	
t_{AR}	Adresse stabil vor $\overline{RD}\downarrow$	0	–	0	–	ns
t_{RA}	Adresse stabil nach $\overline{RD}\uparrow$					
t_{RR}	$\overline{RD}$-Impulsbreite	300		300	–	
t_{RD}	Gültige Daten nach $\overline{RD}\downarrow$ [1])	–	250	–	200	
t_{DF}	Datenbus hochohmig nach $\overline{RD}\uparrow$	10	150	10	100	
t_{RV}	Zeit zwischen $\overline{RD}$- oder $\overline{WR}$-Signalen	850	–	850	–	

Schreiben

Symbol	Bezeichnung	Grenzwerte				Einheit
		8255A		8255A-5 [2])		
		min.	max.	min.	max.	
t_{AW}	Adresse stabil vor $\overline{WR}\downarrow$	0	–	0	–	ns
t_{WA}	Adresse stabil nach $\overline{WR}\uparrow$	20		20		
t_{WW}	$\overline{WR}$-Impulsbreite	400		300		
t_{DW}	Gültige Daten vor $\overline{WR}\uparrow$	100		100		
t_{WD}	Gültige Daten nach $\overline{WR}\uparrow$	30		30		

[1]) Nach dem Einschalten der Versorgungsspannung muß am RESET-Eingang wenigstens 50 µs H-Pegel anliegen. Nachfolgende RESET-Impulse mussen mindestens 500 ns lang sein.

[2]) Die Spezifikationen für den SAB 8255A-5 sind noch nicht endgültig, einige Parameter können noch geändert werden.

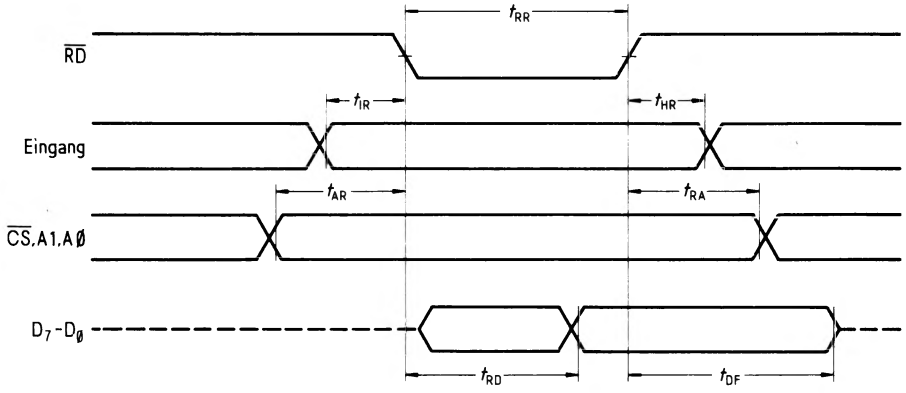
Andere Schaltzeiten

$T_U = 0$ bis 70 °C; $V_{CC} = +5\,V \pm 5\%$; GND = 0 V

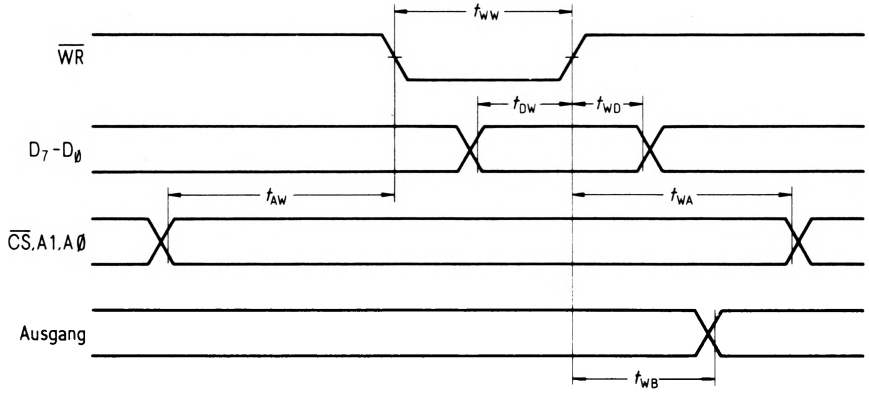
Symbol	Bezeichnung	Grenzwerte				Einheit
		8255A		8255A-5 [2]		
		min.	max.	min.	max.	
t_{WB}	Zeit von $\overline{WR}=1$ bis Ausgabe [1]	–	350	–	350	
t_{IR}	periphere Daten vor $\overline{RD}\downarrow$	0	–	0	–	
t_{HR}	periphere Daten nach $\overline{RD}\uparrow$					
t_{AK}	$\overline{ACK}$-Impuls Breite	300		300		
t_{ST}	$\overline{STB}$-Impuls Breite	500		500		
t_{PS}	periphere Daten vor $\overline{STB}\uparrow$	0		0		
t_{PH}	periphere Daten nach $\overline{STB}\uparrow$	180		180		
t_{AD}	Zeit von $\overline{ACK}=0$ bis Ausgabe [1]	–	300	–	300	
t_{KD}	Zeit von $\overline{ACK}=1$ bis Ausgang hochomig	20	250	20	250	ns
t_{WOB}	Zeit von $\overline{WR}=1$ bis $\overline{OBF}=0$ [1]	–	650	–	650	
t_{AOB}	Zeit von $\overline{ACK}=0$ bis $\overline{OBF}=1$ [1]		350		350	
t_{SIB}	Zeit von $\overline{STB}=0$ bis IBF = 1 [1]		300		300	
t_{RIB}	Zeit von $\overline{RD}=1$ bis IBF = 0 [1]		300		300	
t_{RIT}	Zeit von $\overline{RD}=0$ bis INTR = 0 [1]		400		400	
t_{SIT}	Zeit von $\overline{STB}=1$ bis INTR = 1 [1]		300		300	
t_{AIT}	Zeit von $\overline{ACK}=1$ bis INTR = 1 [1]		350		350	
t_{WIT}	Zeit von $\overline{WR}=0$ bis INTR = 0 [1]		850		850	

[1]) Test-Bedingung: SAB 8255A: $C_L = 100$ pF; SAB 8255A-5: $C_L = 150$ pF.
[2]) Die Spezifikationen für den SAB 8255A-5 sind noch nicht endgültig, einige Parameter können noch geändert werden.

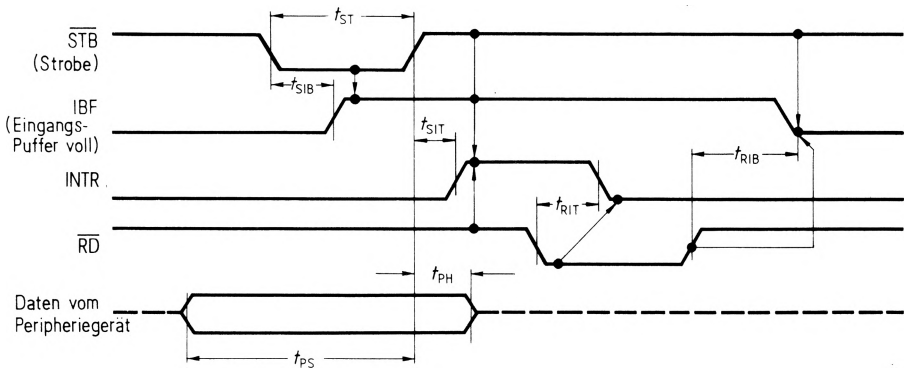
Impulsdiagramm Betriebsart Ø (einfache Eingabe)



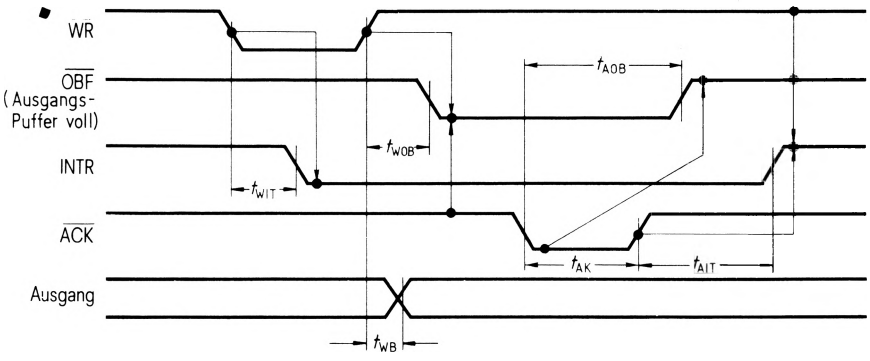
Impulsdiagramm Betriebsart Ø (einfache Ausgabe)



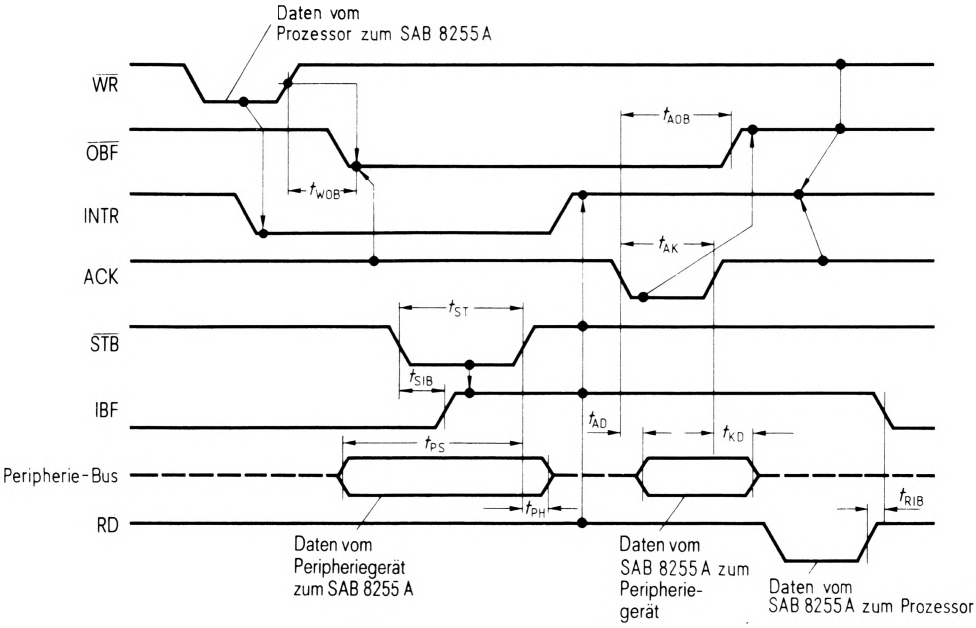
Impulsdiagramm Betriebsart 1 (getastete Eingabe)



Impulsdiagramm Betriebsart 1 (getastete Ausgabe)



Impulsprogramm Betriebsart 2 (Zweiweg-Bus)



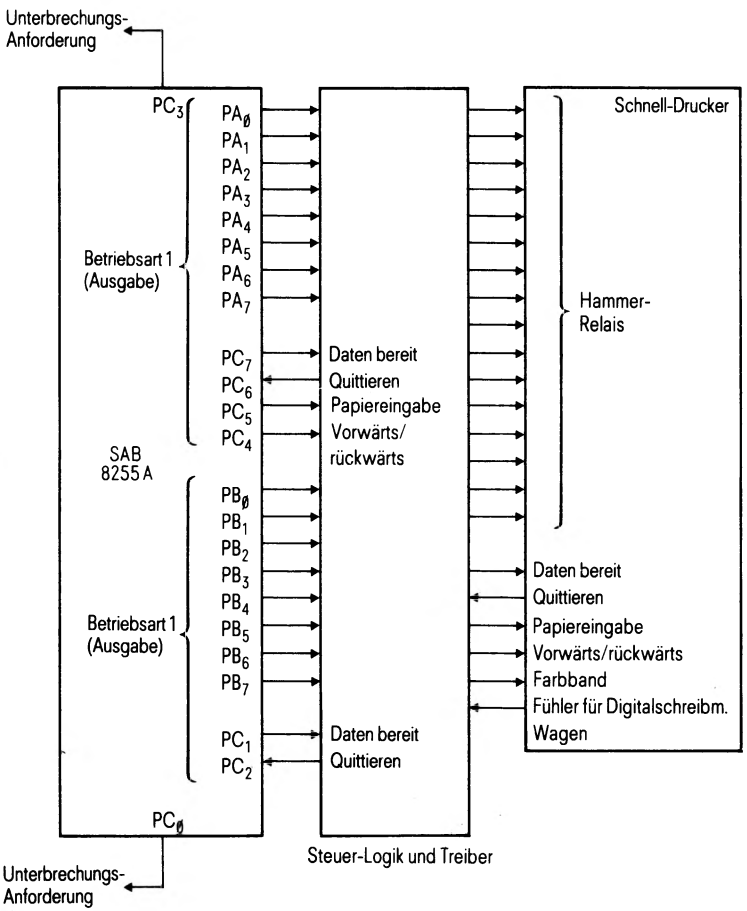
Anwendungen des SAB 8255A

Der SAB 8255A ist ein sehr leistungsfähiger Baustein, mit dem Peripherie-Geräte an Mikroprozessor-Systeme angeschlossen werden. Er nutzt die vorhandenen Anschlüsse optimal aus und ist flexibel genug, daß viele E/A-Geräte ohne zusätzlichen Schaltungsaufwand angeschlossen werden können.

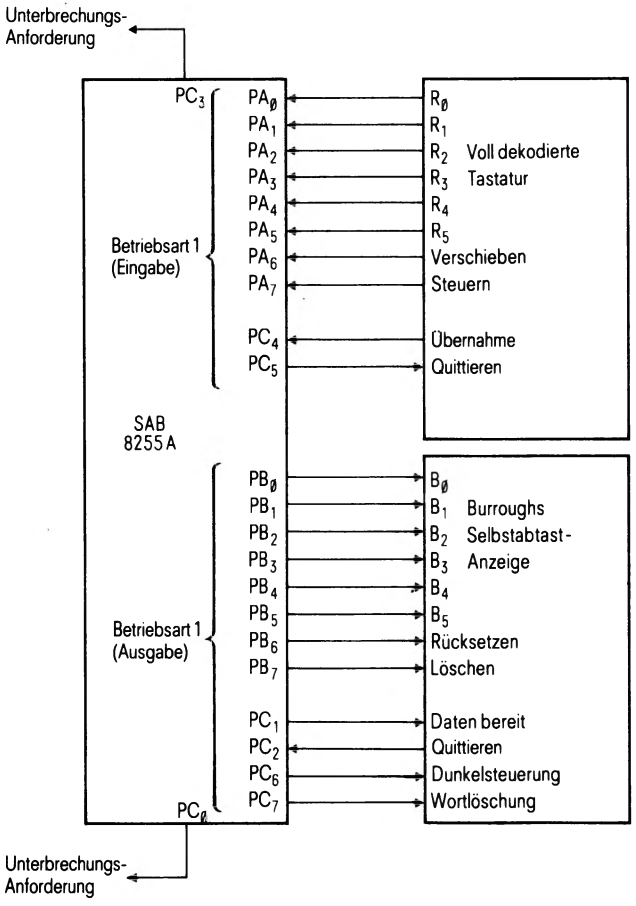
Jedem Peripherie-Gerät ist normalerweise im Mikrocomputersystem ein „Dienstprogramm" zugeordnet. Dieses Programm verwaltet die Software-Schnittstelle zwischen Gerät und Mikroprozessor. Die funktionelle Festlegung des SAB 8255A wird durch das E/A-Dienstprogramm programmiert und stellt eine Erweiterung der Systemsoftware dar.

Einige typische Anwendungen des SAB 8255A sind hier aufgeführt.

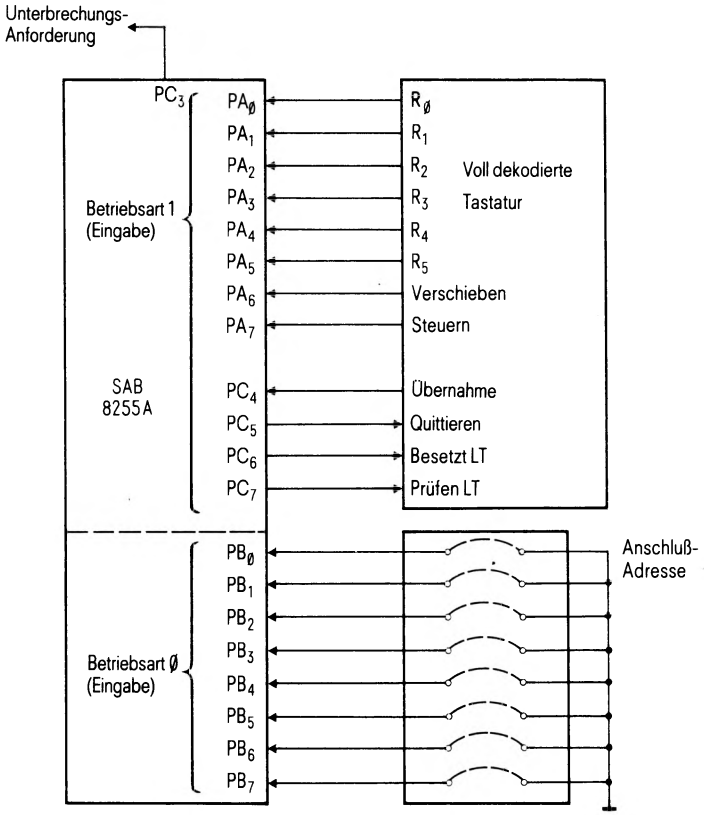
Schnittstelle für Drucker



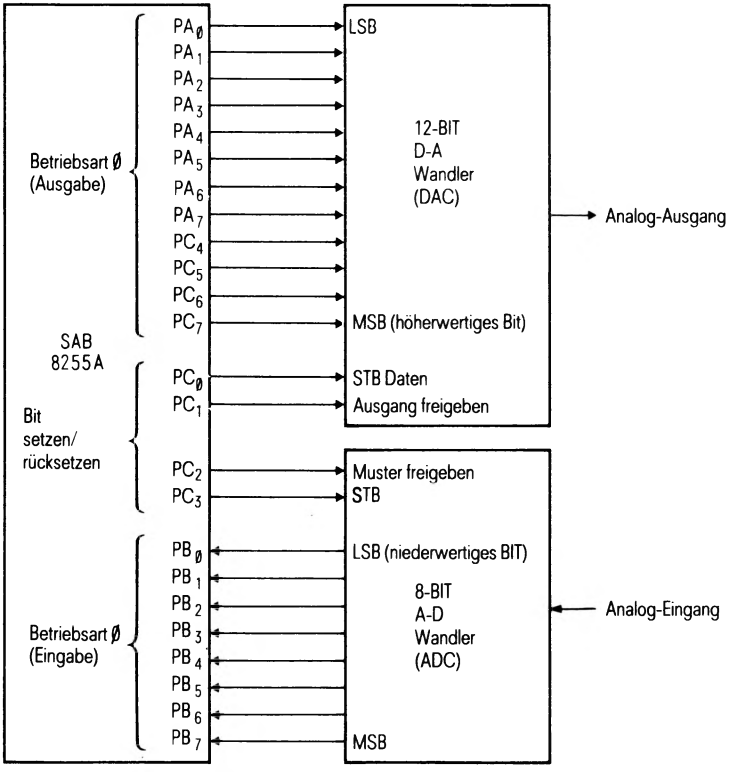
Schnittstelle zu Tastatur- und Anzeigegerät



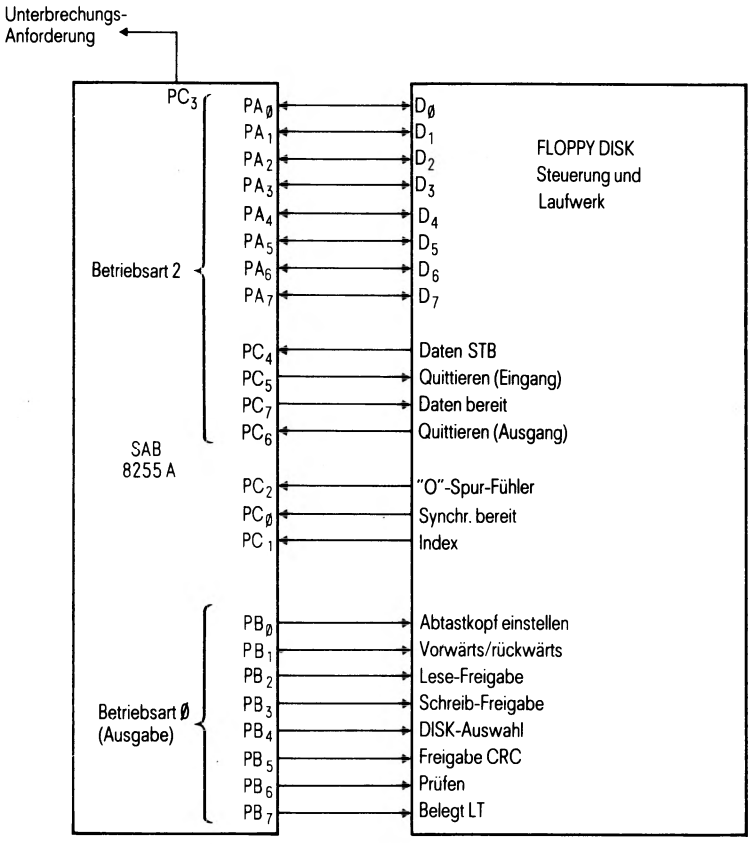
Schnittstelle zu Tastatur und Anschlußadressen



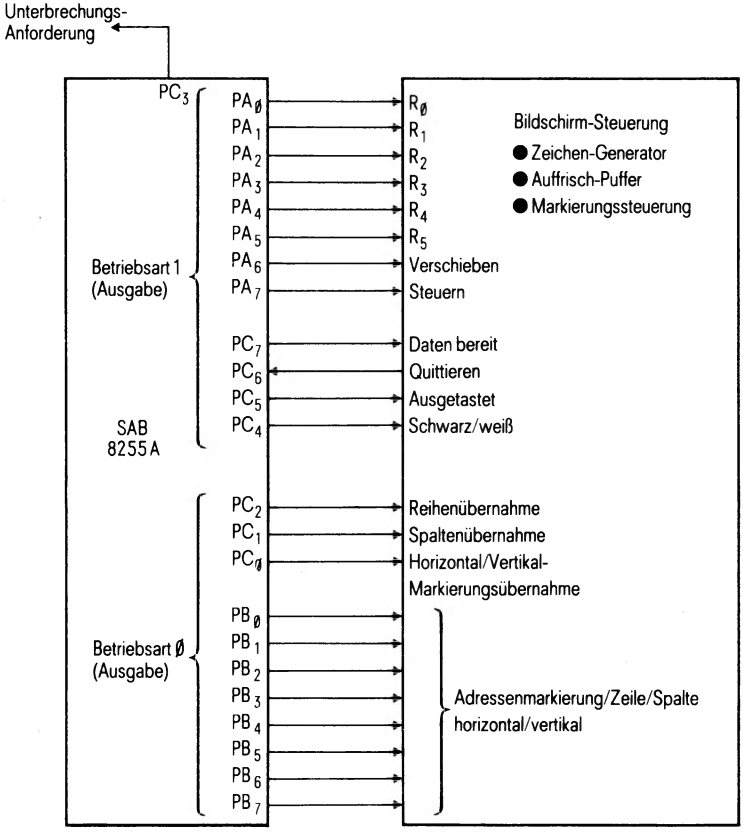
Schnittstelle zu Digital-/Analog- und Analog-/Digitalwandler



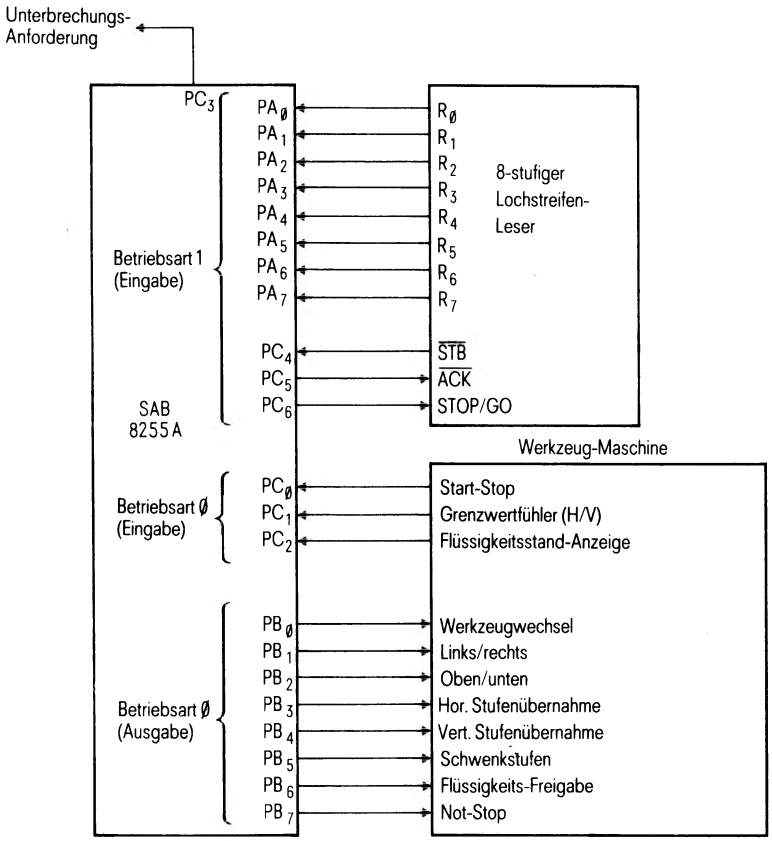
Schnittstelle zu Floppy-Disk-Steuerung und Laufwerk



Schnittstelle zu Bildschirmsteuerung



Schnittstelle zu einer Werkzeugmaschine



MSXASM
Editor/Assembler Z80 / MSX

Der ZN Editor/Assembler ist ein komplettes Programmentwicklungssystem zur Erstellung und Analyse von Z80 Maschinenprogrammen auf MSX-Rechnern. Das Paket enthält einen symbolischen Assembler, einen Disassembler, einen Texteditor, sowie einen Maschinencode-Monitor.
Der komplette Quelltext des Assemblers wird jedem Käufer mitgeliefert. Dies erlaubt eine eingehende Studie des Assemblers und ermöglicht eigene Modifikationen und Erweiterungen. Alle Befehle sind in einer einzigen Kommandostruktur zusammengefaßt. Daraus resultiert ein kompaktes und sehr effizientes System.

- Standard Z80 SYNTAX
- über 400 Zeilen pro Minute Assemblergeschwindigkeit
- Symbole in praktisch beliebiger Länge
- Symbolsortiermöglichkeit
- Dezimal-, Hex- oder Octal-Zahlen
- Ausgabe auf parallelen Druckerport oder RS-232
- Voller symbolischer Zweipaß-Disassembler
- kompletter Editor mit Suchfunktionen und Lese- und Schreibfunktionen auf den Massenspeicher

Der eingebaute Monitor erlaubt das Ansehen von Speicherzellen und des Ein-/Ausgabe-Ports. BREAK-Points können gesetzt und Registerinhalte ausgegeben werden.

Die Lieferung erfolgt mit englischer Kurzbeschreibung. Wir empfehlen Ihnen zu diesem Assembler die Bücher Nr. 119 und 8029.

Best.-Nr. 3006 **(Cassette)** **149,– DM**

LOGO – Turtle-Graphics Interpreter

LOGO ist die Computersprache, die sich in den letzten Jahren immer mehr unter den Schülern und Anfängern verbreitet hat.
Programmieren in LOGO ist eine kreative und hochinteressante Aufgabe. Rechenprozeduren sind eingebaut.

Best.-Nr. 3008 **(Cassette)** **149,– DM**

WORDPRO Wortprozessor for MSX

Dieses Textverarbeitungsprogramm auf Cassette ist ein kompaktes Paket mit professionellem Qualitäts-Level. Sie können Texte eingeben, editieren und diese dann auf einen Drucker ausgeben. Die Editierung ist voll bildschirmorientiert. Globales Suchen, Verschieben und Ersetzen von Text ist selbstverständlich. Ausgabe-Formattierungskommandos wie:

- Untersteichen
- Seitennumerierung
- Seitenüberschriften
- horizontaler Tabulator
- automatische Paragraphenerstellung
  u. v. a. mehr

Ein Handbuch in englischer Sprache ist im Lieferumfang enthalten.

Best.-Nr. 3011 **(Cassette)** **199,– DM**

FORTH für MSX

Die Programmiersprache FORTH hat in den letzten Jahren immer mehr an Bedeutung gewonnen. Diese FORTH-Version entspricht voll der FIG-FORTH-Spezifikation und enthält sogar eine Fließkomma-Arithmetik. Ideal für den Anfänger und Einsteiger, der FORTH zuerst einmal kennenlernen möchte und für den FORTH-Experten, der das System sogar selbst erweitern kann. Im Lieferumfang sind zwei deutschsprachige Bücher enthalten.

Best.-Nr. 3005 **(Cassette)** **249,– DM**

Kommunikationspaket (RS232C)

Jeder, der ein Modem oder einen Drucker mit serieller Schnittstelle an seinen MSX-Computer anschließen will, braucht dieses Paket.
RS232 Parameter können unter Programmkontrolle eingestellt werden. Ideal für "UP- und DOWN-loaden" von Datenbanken.

Best.-Nr. 3041 **(Cassette)** **149,– DM**

MSX

Es lebe der Standard !

Software ★ Hardware Zusätze
Bücher für Ihren MSX-Computer

Kaufen Sie MSX-Programme bei der Nr. 1 in Deutschland !

Hiermit bestelle ich unter Anerkennung Ihrer Liefer- und Zahlungsbedingungen folgende Artikel:

BÜCHER IN DEUTSCHER SPRACHE

....	24	Progr.i.Maschinenspr. Z80,T.2	29,80
....	31	57 Programme in BASIC	39,00
....	119	Progr. i. Masch.Spr. Z80,T. 1	39,00
....	113	BASIC Programmierhandbuch	19,80
....	122	BASIC für Fortgeschrittene	39,00
....	217	Künstliche Intelligenz	19,80
....	220	Tabellenkalk.f.blutige Laien	19,80
....	230	MSX – Tips und Tricks	29,80
....	252	Z-80 Referenzkarte (handl.Karte m.a.Z-80-Bef.)	5,00
...	3001	GI-3-8910 Handbuch (ausführl. Datenblatt m.Anwend.-Beisp.	19,80
...	3002	8255 Datenheft (Der 8255 ist der Ein-/Ausgabe-Baustein im MSX)	19,80
...	3003	TMS 9929A Datenblatt (Der TMS 9929 ist der Video-controller i.MSX-Computer)	9,80
...	8029	Z80 Assemblerhandbuch	29,80

SOFTWARE

...	3000	View Data	149,00
...	3005	FORTH für MSX	249,00
...	3006	Editor/Assembler f. MSX	149,00
...	3007	Tabellenkalkulation	199,00
...	3008	LOGO für MSX	149,00
...	3009	Sprach-Synthese	379,00
...	3010	Datenbank	149,00
...	3011	Wortprozessor	199,00
...	3012	Home Budget	129,00

BÜCHER IN ENGLISCHER SPRACHE

...	3037	Starting with MSX	49,00
...	3038	Starting Mach.Code with MSX	49,00
...	3039	MSX Red Book	79,00
...	3040	Behind the Screens of MSX	79,00

PLATINE

...	3004	Ein-/Ausgabeplatine f. MSX-Computer. Kann als 8K ROM-Platine oder als 8255 I/O Platine oder kombiniert verwendet werden. Ideal für I/O Experimente, Druckeranschluß u.v.a.	79,00

SPIELE

...	3013	Eric and the Floaters	39,00
...	3014	Driller Tanks	49,00
...	3015	Fire Rescue	49,00
...	3016	Dog Fighter	49,00
...	3017	Hyper Viper	49,00
...	3018	Spooks + Ladders	49,00
...	3019	Coco in the Castle	49,00
...	3031	Holdfast	79,00
...	3032	Superchess	79,00
...	3033	Stop the Express	49,00
...	3094	Ninja	39,00
...	3035	Star Avenger	49,00
...	3036	Mean Streets	49,00
...	3037	Cribbage	49,00

Programme auf Cassette, Lieferung per Vorkasse auf Postscheckkonto München Nr. 15 994-807 oder Nachnahme (+ DM 6,50 NN-Gebühr). Angebot freibleibend. Zwischenverkauf vorbehalten.

Name: ..... Vorname: .....

Straße: ..... Ort: .....

☐ Ich habe das Geld bereits überwiesen (+ DM 2,50 für Versandkosten)

☐ Ich bitte um Lieferung per Nachnahme

HOFACKER

HOFACKER-VERLAG
Ing. W. Hofacker GmbH
Tegernseer Str. 18
D-8150 Holzkirchen
Tel.: 08024 / 7331 • Tlx.: 526973

Weitere interessante Bücher von Hofacker:

BÜCHER in deutscher Sprache

Best.-Nr.	Titel	Preis DM
1	Transistor Berechnungs- und Bauanleitungsbuch-1	29.80
2	Transistor Berechnungs- und Bauanleitungsbuch-2	19.80
3	Elektronik im Auto	9.80
4	IC-Handbuch. TTL. CMOS. Linear	19.80
5	IC-Datenbuch. TTL. CMOS. Linear	9.80
6	IC-Schaltungen. TTL. CMOS. Linear	19.80
7	Elektronik Schaltungen	19.80
8	IC-Bauanleitungs-Handbuch	19.80
9	Feldeffekttransistoren	9.80
10	Elektronik und Radio	19.80
12	Beispiele integrierter Schaltungen (BIS)	19.80
13	HEH. Hobby Elektronik Handbuch	9.80
16	CMOS Teil 1. Einführung. Entwurf. Schaltbeispiele	19.80
17	CMOS Teil 2. Entwurf und Schaltbeispiele	19.80
18	CMOS Teil 3. Entwurf und Schaltbeispiele	19.80
19	IC-Experimentier Handbuch	19.80
20	Operationsverstärker	19.80
21	Digitaltechnik Grundkurs	19.80
22	Mikroprozessoren. Eigenschaften und Aufbau	19.80
23	Elektronik Grundkurs. Kurzlehrgang Elektronik	9.80
24	Programmieren in Maschinensprache mit Z80. II	29.80
25	68000 Microcomputer Einführung	39.00
26	Mikroprozessor. Teil 2	19.80
27	BASIC-M für 6800 09 68000 (Motorola)	29.80
28	Lexikon u. Worterbuch f. Elektr. u. Mikroprozessor	29.80
30	Floppy Disk Selbstbau-Handbuch (i.V.)	49.00
31	57 Praktische Programme in BASIC	39.00
32	ATARI BASIC. für Selbststudium und Praxis	39.00
33	Microcomputer Programmierbeispiele	19.80
34	TINY-BASIC Handbuch	19.80
35	Der freundliche Computer	29.80
102	Mathematische + Wissenschaftliche Progr. i. BASIC	29.80
103	Oszillographen-Handbuch	19.80
104	Portable Computer Handbuch (i.V.)	39.00
108	Rund um den Spectrum (Progr., Tips und Tricks)	29.80
109	6502 Microcomputer Programmierung	29.80
111	Programmieren mit TRS-80 (GENIE)	29.80
112	PASCAL-Programmier-Handbuch	29.80
113	BASIC-Programmier-Handbuch (mit BASIC-Kurs)	19.80
114	Der Microcomputer im Kleinbetrieb	39.80
116	Einführung 16-Bit Microcomputer	29.80
118	Programmieren in Maschinensprache mit dem 6502	49.00
119	Programmieren in Maschinensprache mit Z80. I	39.00
120	Anwenderprogramme fur TRS-80 und GENIE	29.80
121	Microsoft BASIC-Handbuch	29.80
122	BASIC für Fortgeschrittene	39.00
123	IEC-Bus Handbuch	19.80
124	Progr. in Maschinensprache mit Commodore-64	29.80
127	Einführung i.d. Microcomputer-Progr. mit 6800	49.00
128	Programmieren mit dem CBM	29.80
130	Programmierbeispiele für CBM	9.80
132	CP-M Handbuch	19.80
133	Handbuch für MS-DOS. IBM-PC	29.80
136	Funktionsanalyse	79.00
137	FORTH – Grundlagen. Einführung. Beispiele	49.00
139	BASIC für blutige Laien (speziell f. TRS-80. Genie)	19.80
140	Progr. i. BASIC u. Maschinencode mit dem ZX81	29.80
141	Progr. f. VC-20 (Spiele. Utilities. Erweiterungen	29.80
143	35 Programme für den ZX81	29.80
144	33 Programme für den ZX-Spectrum	29.80
145	64 Programme für den Commodore-64	39.00
146	Hardware Erweiterungen für den Commodore-64	39.00
147	Beherrschen Sie Ihren Commodore-64	19.80
148	Programmierhandbuch für SHARP	49.00
149	Programme für TI 99 4A	49.00
175	Astrologie auf dem ATARI 800	49.00
187	Mehr als 29 Programme für den Commodore-64	29.80
188	Statistik in BASIC	39.00
189	Editor + Assembler. Macrofire	19.80
190	Das große Spielebuch für ATARI 600XL 800XL. I	29.80
200	FORTH-Anwendungsbeispiele	49.00
202	UNIX – Grundlagen und Anwendungen	39.00
204	Grafik und Ton mit dem Commodore-64	29.80
205	Das große Spielebuch für ATARI 600XL 800XL. II	29.80
210	Superprogramme für IBM-PC	29.80
212	Geschäftsprogramme für Commodore-64	39.00
213	Technische Gleichungssysteme in BASIC	49.00
217	Künstliche Intelligenz	19.80
218	Produktivität rauf mit OPEN-ACCESS	39.00
220	Tabellenkalkulation für blutige Laien (i.V.)	19.80
221	SYMPHONY-Anwendungen	29.80
222	Praktische Anwendungen mit dem Sinclair QL	29.80
223	MODULA-2 Anwender Handbuch	49.00
224	Anwenderprogramme für Apple II c. und II e	19.80
227	Die große Starparade	39.00
8029	Z-80 Assembler-Handbuch	29.80

BÜCHER in englischer Sprache

Best.-Nr.	Titel	Preis DM
29	Hardware Handbook	49.00
52	Small Business Programs for the IBM PC	29.80
151	8K Microsoft BASIC Reference Manual	9.80
152	Expansion Handbook for 6502 and 6800	19.80
156	Small Business Programs	29.80
158	The Second Book of Ohio Scientific	19.80
159	The Third Book of Ohio Scientific	29.80
160	The Fourth Book of Ohio Scientific	29.80
161	The Fifth Book of Ohio Scientific	19.80
162	ATARI Games in BASIC	19.80
164	ATARI-BASIC Learning by Using	19.80
166	Programming in 6502 Machinelanguage PET CMB	49.00
169	How the Progr. y. ATARI i. 6502 Machinelanguage	29.80
170	FORTH on the ATARI – Learning by Using	29.80
171	See the Future with your ATARI (Astrology)	49.00
172	Hackerbook I (Tricks + Tips for your ATARI)	29.80
173	PD-Program Descriptions (ATARI)	9.80
174	ZX-81 TIMEX Progr. i. BASIC a. Machine Lang.	5.00
176	Programs + Tricks for VIC's	29.80
177	CP/M – MBASIC and the OSBORNE	5.00
178	The APPLE in Your Hand	39.00
182	The Great Book of Games Vol. I – Games f. the C-64	29.80
183	More on the Sixtyfour (Commodore-64)	39.00
184	How to Progr. your C-64 i. 6502 10 Machinelang.	29.80
185	Commodore-64 Tune-up	39.00
186	Small Business Programs for the Commodore-64	29.80

Der HOFACKER-Verlag produziert und vertreibt neben einer sehr großen Auswahl an Fachbüchern für Elektronik und Microcomputertechnik noch:

- Leerplatinen und Bauanleitungen für Zusatzeinrichtungen für Ihren Personalcomputer. sowie
- Programme (Software) und Leercassetten (C-10) für die bedeutenden Personalcomputer.

(i.V. bedeutet: Buch ist in Vorbereitung)

HOFACKER

HOLZKIRCHEN SINGAPORE LOS ANGELES

ISBN 3-88963-230-0