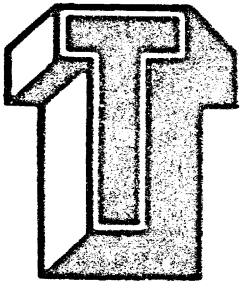
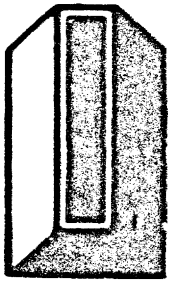
Kleinstrechner



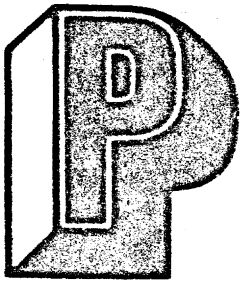
Tendenzen
und Theorien

Simulation



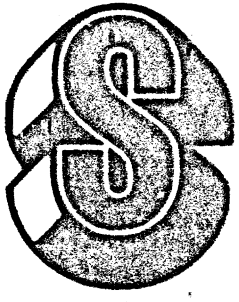
Informationen
und Ideen

Mikrorechner-
bausatz Z 1013



Programme
und Projekte

Kleincomputer
im Physikunterricht



Spaß
und Spiel

7

Kleinstrechner-TIPS

Heft 7

Mit 31 Bildern

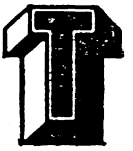
Herausgegeben von
Prof. Dr.-Ing. Hans Kreul
Doz. Dr.-Ing. Wilhelm Leupold
Doz. Dr. sc. techn. Thomas Horn

VEB Fachbuchverlag Leipzig

Inhalt

ISBN 3-343-00227-5

Zum Inhalt



Im ersten Beitrag des Heftes geht WAGENKNECHT auf rekursive Prozeduren ein und gibt Beispielprogramme für solche in BASIC an.

SCHWITALLA und MÜLLER stellen die Grundausbaustufe des Mikrorechnerbausatzes Z 1013 vor, der insbesondere dem Elektronikamateur einen kostengünstigen Weg für den Aufbau eines Mikrorechners ermöglicht.

Da Kleincomputer im zunehmenden Maße von Personen genutzt werden, die selbst nicht im Programmieren geübt oder ausgebildet sind, geben MEINHARDT und SASSE Hinweise, was ein Programmierer in diesem Falle beachten sollte.

LORENZ und SCHULZE behandeln im zweiten Teil der Veröffentlichung über die Simulation auf Mikrorechnern Simulationsmodelle von Bedienungssystemen.

JUPE und LÖHR stellen Möglichkeiten und Beispiele für den Einsatz von Kleincomputern im Physikunterricht an allgemeinbildenden Schulen vor.

Ein Programm zum Lösen beliebiger Gleichungen mit einer Variablen mittels Einschachtelungsverfahren gibt KREUL im Artikel »Gleichungen – vom Computer gelöst« an.

Herausgeber und Verlag danken den Lesern für das Interesse an den „Kleinstrechner-TIPS", das sich in zahlreichen Zuschriften und Veröffentlichungsangeboten äußert. Beim Einsenden von Artikeln bitten sie folgende Hinweise zu beachten:

- In den „Kleinstrechner-TIPS" werden Artikel aus den auf der 4. Umschlagseite angegebenen Gebieten veröffentlicht.
- Manuskripte sind zweizeilig mit schwarzem Farbband mit Schreibmaschine zu schreiben, ä, ö und ü dürfen nicht durch ae, oe und ue ersetzt werden.
- Bilder sollten – so klein wie möglich – auf getrennten Blättern gezeichnet sein. Eine Bildunterschriftenliste ist beizufügen.
- Rechnerausdrucke (z.B. Programme) sollen tiefschwarz mit guter Ausnutzung des Formats (engzeilig, kein zu breites Kommentarfeld usw.) gedruckt sein.
- Veröffentlichte Programme müssen auf Computern, die in der DDR produziert werden, lauffähig sein.
- Die Autorenangabe soll enthalten: akadem. Grad, Vorname, Name. Tätigkeit, Arbeitsstelle, Privatanschrift.

Manuskripte mit einem Umfang von nicht mehr als 15 Seiten (einschließlich Bilder und Programme) sind in zwei Exemplaren an einen der Herausgeber zu senden (Anschriften auf 3. Umschlagseite).

Wilhelm Leupold

REKURSION–
was, wie, wozu?

Einführung und Grundlagen

Wenn ich hier von einer guten Fee berichte, oder auch davon, wie man möglichst schnell die Scheiben eines »Turmes von Hanoi« nach bestimmten Regeln umstapelt, so stellt sich sehr berechtigt die Frage, was dies mit »Kleinstrechner-TIPS« zu tun hat. Im Falle des Scheibentransportes vermuten versierte Leser vielleicht schon eine interessante Programmieraufgabe.

Welchen Zusammenhang gibt es aber zwischen der Fee und Informatik? Jeder kennt verschiedene Märchen, in denen eine gute, alte und mütterliche Frau, meist »gute Fee« genannt, die einem Günstling, sagen wir, genau drei Wünsche nach freier Wahl gewährt, vorkommt. Diese Wahl fällt bekanntlich schwer, u.a. deshalb, weil nach Realisierung der Wünsche die Fee verschwindet und ihre Dienste für immer versagt.

Obwohl es moralisch sicher nicht hoch einzuschätzen ist, möchte ich zum Nachdenken aufrufen, wie man sogar vier, fünf, sechs und noch mehr dieser märchenhaften Dienstleistungen bei gleichem Maximalangebot – Erfüllung dreier Wünsche – in Anspruch nehmen kann.

Richtig, wir wünschen uns bei der dritten Möglichkeit, drei Wünsche frei zu haben. Damit entsteht ein Teilprogramm, auch Unterprogramm (UP) oder **Prozedur** (PR) genannt, wie in Bild 1 dargestellt.

```
PR »Drei Wünsche frei«
  Wunsch 1
  Wunsch 2
  »Drei Wünsche frei«
END
```

Bild 1

Offensichtlich ruft die Prozedur »Drei Wünsche frei« sich selbst (im dritten Wunsch) auf. Wir nennen daher solche Prozeduren auch **rekursive Prozeduren** (rekursiv – zurückgehend).

Von der Fee wird das Wörtchen END, mit dem jede Prozedur abzuschließen ist, wie eine Provokation empfunden werden, denn es entsteht eine unendliche Kette von Wünschen. Pro Aufruf erfüllt die Fee dann genau zwei frei wählbare Wünsche und gibt als dritten die Bereitschaft zu weiterer Wunscherfüllung an. Im konkreten Falle haben wir gegen die unendliche Wiederholung der Aufrufe nichts einzuwenden. Algorithmen zeichnen sich im Gegensatz dazu u.a. durch ihre Endlichkeit aus, d.h., sie stoppen nach einer gewissen Zeit. Man sagt auch, der Algorithmus **terminiert.** Um dies zu gewährleisten, muß eine **Abbruchbedingung** in die

Prozedur eingefügt werden. Diese bewirkt das Verlassen des **Zyklus,** falls sie erfüllt ist. So etwas könnte hier etwa dann eintreten, wenn Wunsch 1 oder 2 gegen die guten Sitten verstößt. Sehen wir uns das an einem einfachen Beispiel – dem Rückwärtszählen – an. Bild 2 gibt dazu eine rekursive Prozedur mit Abbruchbedingung wieder.

```
PR ABWAERTSZAEHLEN :N
  WENN : N = Ø
    DANN RUECKKEHR
  DRUCKE ZEILE : N
  ABWAERTSZAEHLEN :N – 1
END
```

Bild 2

Diese Prozedur ist nun schon in einer für den Computer verständlichen Sprache, der **Programmiersprache** (hier: LOGO), geschrieben. Dabei dient N als Platzhalter für genau die Zahl, mit der die Prozedur aufgerufen wird. Man nennt einen solchen Platzhalter einen **formalen Parameter.** Dieser wirkt **lokal,** d.h. nur in dieser Prozedur. Außerhalb kann der **Wert von N,** geschrieben als :N, ein ganz anderer sein. Beim Aufruf einer Prozedur erfolgt eine Parameterübergabe der Werte der **aktuellen** an die entsprechenden formalen **Parameter.**

Im Aufruf

ABWAERTSZAEHLEN 3

ist die Zahl 3 der Wert des aktuellen Parameters, und überall dort, wo :N in der Prozedur aus Bild 2 vorkommt, wird automatisch 3 eingesetzt.
Zu erklären ist noch das Kommando RUECKKEHR. Dazu »begeben« wir uns in die Situation des ersten Prozeduraufrufes, also

ABWAERTSZAEHLEN 3.

Da die Abbruchbedingung nicht erfüllt ist, wird der Wert von N, d.h. die Zahl 3, ausgegeben und durch

ABWAERTSZAEHLEN 2

die Prozedur rekursiv mit 2 aufgerufen. Dies geschieht, obwohl die Prozedur mit dem aktuellen Parameterwert 3 noch gar nicht beendet ist. Auf diese Art wird zunehmend in der sogenannten **Rekursionstiefe abgestiegen,** bis

ABWAERTSZAEHLEN Ø

erreicht ist und die Abbruchbedingung erfüllt wird. Nun kann wieder **aufgestiegen** werden, d.h., RUECKKEHR beendet die Prozedur mit :N = Ø ohne Erreichen des Druckbefehles und veranlaßt eben die Rückkehr in das nächsthöhere, auch als **rufendes** bezeichnete **Rekursionsniveau.** Dort wird die Befehlsabarbeitung genau mit der Anweisung fortgesetzt, die dem rekursiven Aufruf folgt, ebenso wie das auch bei anderen nicht rekursiven **Unterprogrammaufrufen** und in einer Vielzahl von Programmiersprachen üblich ist. Im konkreten Falle ist das der Befehl END, der den Abschluß der rufen-

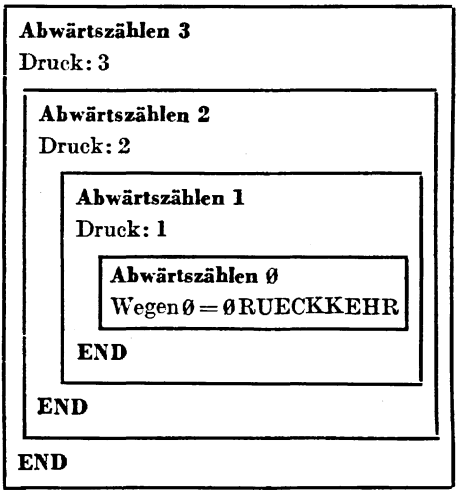


Bild 3

den Prozedur bewirkt und somit automatisch zur Fortsetzung im nächsthöheren Niveau führt. Bild 3 zeigt zusammenfassend den »Aufrufmechanismus« nach der Eingabe

ABWAERTSZAEHLEN 3.

Untereinander wird die Zahlenfolge 3 2 1 ausgegeben.
Nun nehmen wir eine äußerlich kleine Veränderung in der Prozedur aus Bild 2 vor, was einen sehr interessanten Effekt nach sich zieht. Diese Korrektur soll im Vertauschen der Reihenfolge der Zeilen 3 und 4 von PR ABWAERTSZAEHLEN :N bestehen, wonach sich die in Bild 4 dargestellte
PR ZAEHLEN :N ergibt.

```
PR ZAEHLEN :N
  WENN :N = Ø
    DANN RUECKKEHR
  ZAEHLEN :N − 1
  DRUCKE ZEILE :N
END
```

Bild 4

Wichtig ist, daß damit der rekursive Aufruf nicht mehr unmittelbar vor END steht, sondern es folgt noch eine, den Wert des auf dem jeweiligen Rekursionsniveau aktuellen Parameters verarbeitende Anweisung. Derartige Prozeduren heißen **echt rekursiv,** wogegen solche wie unser Einführungsbeispiel, aber auch die in Bild 2, als **endständige Rekursionen** (engl. tail recursion) bezeichnet werden. Die Abarbeitung erfolgt hier völlig analog.

Zur Eingabe

ZAEHLEN 3

wird allerdings die Zahlenfolge 1 2 3 untereinander ausgegeben. Das ist schon bemerkenswert. Aus Bild 5 geht die Wirkung klar hervor. Man beachte die jeweils lokalen, an das gerufene Rekursionsniveau gebundenen Werte für N.

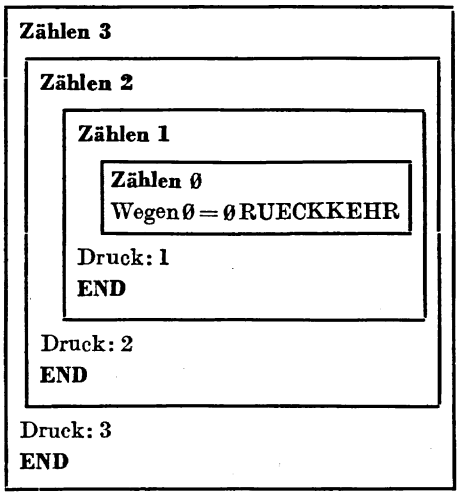


Bild 5

Rekursion und BASIC

Da nun die meisten Kleincomputer über BASIC verfügen, bereitet die Umsetzung rekursiver Algorithmen zunächst Probleme, denn im BASIC sind rekursive Prozeduren nicht formulierbar.
Doch halt! Bevor wir uns zu Anhängern einer so gewichtigen Aussage gegen BASIC erklären, soll mit dem in Bild 6 fixierten Programm experimentiert werden.
Offensichtlich erfolgt die Rückkehr aus der Rekursion völlig korrekt, denn die erwartete Ausgabe

5 4 3 2 1 Ø
GLUECKLICHE WIEDERKEHR!

für N = 5 wird geliefert.
Dennoch müssen wir einige Zusatzmaßnahmen vornehmen, die sich eigentlich direkt aus den bisher betrachteten Beispielen ergeben. So wird es möglich sein, allgemeine rekursive Prozeduren

```
  5 REM Hauptprogramm
 10 INPUT N
 20 GOSUB 500
 30 PRINT
    »GLUECKLICHE WIEDERKEHR«
 40 END

500 REM Unterprogramm
510 PRINT N;
520 IF N=0 THEN RETURN
530 N=N-1
540 GOSUB 500
550 RETURN
```

Bild 6

```
 10 REM Hauptprogramm
 20 INPUT N : TI=0
 30 GOSUB 50 : REM 1. UP-Aufruf
 40 END

 50 REM UP Zählen
 60 TI=TI+1 : N(TI)=N
 70 IF N(TI)=0 THEN
    TI=TI-1 : RETURN
 80 N=N(TI)-1 : GOSUB 50
 90 PRINT N(TI);
100 TI=TI-1 : RETURN
```

Bild 7

auch im BASIC mit Parametern zu simulieren.
Bekannt ist vorerst, daß Prozeduren als BASIC-Unterprogramme allerdings ohne echte Parameterübermittlung dargestellt werden können. Die genannte Einschränkung ergibt sich daraus, daß in dieser Programmiersprache keine **lokalen**, sondern nur **globale Variablen**, also nur solche, die überall im Programm mit dem ihnen zugewiesenen Wert gelten, verfügbar sind. Anstelle jedes in der Prozedur vorkommenden Parameters und jeder lokalen Variablen ist je ein Vektor – auch **Stapel** genannt – einzuführen, auf dessen Komponenten gemäß der Rekursionstiefe zugegriffen wird.
Die Rücksprungstellen werden automatisch gefunden. Allerdings muß vor dem Rücksprung (mit RETURN) die Tiefe um eins vermindert werden, d.h., aus END und RUECKKEHR wird TI = TI – 1: RETURN. Bei jedem Prozeduraufruf werden die Tiefe um eins erhöht und die Parameterwerte übergeben.
Diese hier erläuterten Maßnahmen sind – bis auf Dimensionierung der Vektoren – ausreichend und machen das in Bild 7 dargestellte BASIC-Programm für PR ZAEHLEN :N transparent.

Zur Eingabe der Zahl 3 liefert das Programm die Ausgabe
1 2 3.

Wir haben also gesehen, daß die Rekursion nicht nur in dem dafür besonders geeigneten System LOGO, sondern mit wenigen Zusatzmaßnahmen auch mit BASIC programmierbar ist. Das hat besonders dann Bedeutung, wenn eben nur BASIC beherrscht wird bzw. verfügbar ist.
Sehr häufig ist es offensichtlich, daß rekursive Abläufe sich in **iterative** umformen lassen bzw. daß einfach überschaubare iterative Abläufe zur Lösung der Aufgabe existieren. Für ZAEHLEN und ABWAERTSZAEHLEN ist das der Fall ebenso wie für die als Beispiel für die Rekursion angeführten Aufgaben zur Berechnung des GGT (größter gemeinsamer Teiler) oder der Fakultät. Es ist deshalb wichtig, ein Problem zu betrachten, welches besonders einfach durch einen rekursiven Algorithmus zu lösen ist und sonst aber Mühe bereitet. Dieser kann dann mit BASIC formuliert und vom Kleincomputer abgearbeitet werden. Das ist ganz wichtig! Ginge es nämlich nicht – und das wird eben manchmal vom BASIC behauptet –, so brauchten wir das mächtige Problemlösungsinstrumentarium »Rekursion« erst gar nicht

zu verwenden, und die Lösung mancher Aufgabenstellung bliebe für uns fast verborgen. Im folgenden Beispiel jedenfalls müßten wir bestimmt aufgeben.

Türme von Hanoi

Das Spiel »Turm von Hanoi« wird gern für Programmierübungen zugrunde gelegt (Siehe auch Heft 5, SCHÖNFELDER: Computerspiele – mehr als eine Spielerei). Dieses berühmte Spiel hat einen historischen Ursprung, den ich aus der Literatur kurz wiedergeben möchte. In der Orginalversion »Turm von Brahma« stand vor den Priestern im Tempel der indischen Stadt Benares die Aufgabe, einen Turm aus 64 goldenen Scheiben von einem Ausgangsplatz auf einen Zielplatz umzulegen, wobei stets nur genau eine Scheibe transportiert werden und keine mit größerem Radius auf einer mit kleinerem zu liegen kommen durfte. Hierzu stand genau ein Hilfsplatz zur Verfügung.
Die Priester erhielten diese Aufgabe auf die Frage: »Wie lange wird die Welt bestehen?« mit der Antwort: »So lange ihr zur Lösung dieser Aufgabe braucht!«
Damit sind die Regeln schon vollständig genannt, und jeder wird die abgerüstete Version mit nur drei Scheiben schnell verstehen und bewältigen. Sehr geeignet sind Geldstücke für praktische Erprobungen, wie dies in Bild 8 dargestellt ist.
Zur Kontrolle sind die Zugnummern für die jeweils ganz oben aufliegenden Scheiben in Bild 8 mit angegeben. Es machen sich mindestens sieben Züge notwendig. Wie groß mag wohl die Mindestzuganzahl bei vier Scheiben sein?

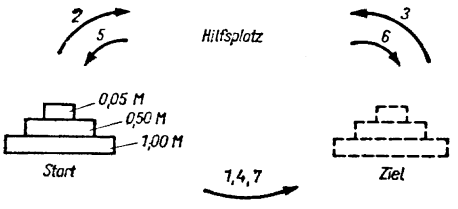


Bild 8

Bevor dies unter heftigen Diskussionen und möglicherweise Zurücknahmen von Umlegungen, wie ich vermute, versucht wird, möchte ich die Antwort auf diese Frage vorwegnehmen. Man schafft es mit 15 Zügen, und allgemein gilt für die Mindestzuganzahl z in Abhängigkeit von der Anzahl der Scheiben n

$$z = 2^n - 1 \quad (n \in N).$$

Es ist sofort einzusehen, daß bei größerem n, $n = 10$ und damit $z = 2^{10} - 1 = 1023$, schon eine Strategie zur Lösung der Aufgabe vorliegen muß, um die Übersicht zu wahren.
Diese Strategie, die wie angekündigt rekursiv sein soll, können wir aber schon bei $n = 4$ erkennen. Zur schnellen Verständigung sollen

S den Startplatz,
Z den Zielplatz und
H den Hilfsplatz

bezeichnen. Die Scheiben werden mit zunehmendem Radius mit den Nummern 1, 2, 3, . . ., $n - 1$, n versehen.
Ist nun ein aus vier Scheiben bestehender Turm von S nach Z zu bringen, so muß als unterste die Scheibe 4 auf Z zu liegen kommen. Der Zug »Scheibe 4 von S nach Z« ist aber nur dann ausführbar, wenn der Teilturm mit den Scheiben 1, 2, 3 (natürlich geordnet) auf H steht. Damit reduziert sich das Problem auf das Umsetzen eines Turmes aus drei Scheiben von S nach H, den Zug »Scheibe 4 von S nach Z« und den Transport o.g. Turmes von H nach Z auf die Scheibe 4.
Die Rekursivität dieses Umstapelalgorithmus besteht also darin, die Umlegung von k Scheiben auf das Umlegen von $k - 1$ Scheiben und den

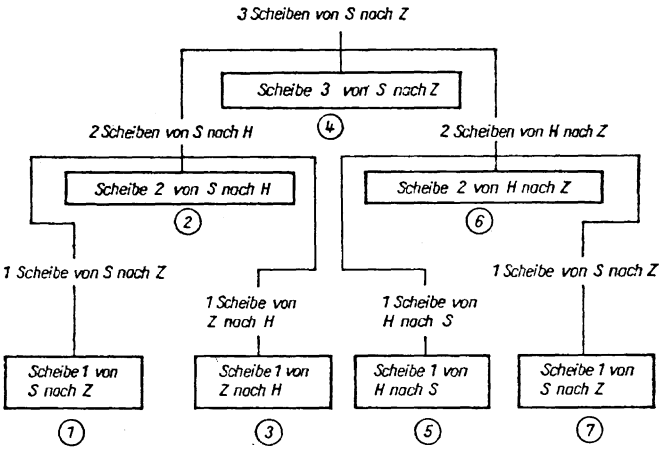


Bild 9

Elementarzug »Scheibe k nach Z« ($k = 1, 2, \ldots, n$) zu reduzieren. Bild 9 dokumentiert schematisch diese rekursive Strategie für $n = 3$.
In Bild 9 wurde jeweils angegeben, »von« welchem Platz »nach« welchem Platz der entsprechende Teilturm umzusetzen ist. Es versteht sich eigentlich von selbst, daß dann der dritte, nicht genannte Platz als Hilfsplatz für diese Teilaufgabe fungiert. Beispielweise ist bei »2 Scheiben von S nach H« eben Z der Hilfsplatz.
Hat man auch dies richtig verstanden, so ist die folgende LOGO-Prozedur in Bild 10 sofort zugänglich.

```
PR HANOI :ZAHL :VON :HILF
    :NACH
  WENN :ZAHL = 0
    DANN RUECKKEHR
  HANOI ( :ZAHL - 1)
    :VON :NACH :HILF
  DRUCKE ZEILE
    ( :ZAHL :VON :NACH)
  HANOI ( :ZAHL - 1)
    :HILF :VON :NACH
END
```

Bild 10

Nach dem Aufruf

HANOI 3 S H Z

werden folgende Züge ausgegeben:

1 S Z
2 S H
1 Z H
3 S Z
1 H S
2 H Z
1 S Z.

Das Nachvollziehen der nach diesem Programm im Computer auf- und absteigenden Rekursionen (z. B. für $n = 5$) ist für das Verständnis nicht von allzu großem Nutzen. Wichtig ist vielmehr, daß mit einer einzigen rekursiv formulierten Lösungsvorschrift der Computer »unbeirrt« einen verzwickten Ablauf realisiert, an dessen Durchführung fast jeder Mensch unweigerlich scheitert. In Anwendung der voranstehenden Erläuterungen erhält man ein BASIC-Programm wie in Bild 11, welches für die o.g. Plätze die Ziffern 1, 2 und 3 verwendet.

```
10 REM Hauptprogramm
20 CLS: DIM V(23), N(23), H(23),
   ZA(23)
30 INPUT
   »Anzahl der Scheiben:«; N
40 H1=1: H2=2: H3=3: TI=0
50 GOSUB 70: REM 1.
   Aufruf HANOI
60 END
```

```
70 REM UP HANOI
80 TI=TI+1
90 ZA(TI)=N: V(TI)=H1:
   H(TI)=H3: N(TI)=H2
100 IF ZA(TI)=0 THEN TI=TI-1:
   RETURN
110 N=ZA(TI)-1: H1=V(TI):
   H3=N(TI): H2=H(TI)
120 GOSUB 80:
   REM rekursiver Aufruf
130 PRINT ZA(TI); V(TI); N(TI)
140 N=ZA(TI)-1: H1=H(TI):
   H3=V(TI): H2=N(TI)
150 GOSUB 80:
   REM rekursiver Aufruf
160 TI=TI-1: RETURN
```

Bild 11

Für $n = 3$ erhält man die Ausgabe

1 1 2
2 1 3
1 2 3
3 1 2
1 3 1
2 3 2
1 1 2,

wobei bezüglich der Platznummern $1 \triangleq S$, $2 \triangleq Z$ und $3 \triangleq H$ gilt. Auch für $n = 13$ kann das Programm benutzt werden. Die Rechenzeit (ohne Druckzeit für die $8191 = 2^{13} - 1$ Tripel) beträgt beim hier verwendeten KC 85/1 immerhin etwa 7 min 15 s.

Angesichts dieser mit BASIC programmierten, rekursiven Lösung sei nochmals betont, daß damit ein Algorithmus umgesetzt werden konnte, der die rekursive Strategie, die in der Analyse des Problems gefunden wurde, vollständig wiedergibt.

Nur mit großer Mühe kann man das folgende, nicht rekursive Verfahren entdecken und so selbsterklärend wie das Programm in Bild 10 mit LOGO, BASIC oder irgendeiner anderen Programmiersprache formulieren.

Bei der Suche nach einer nicht rekursiven Lösung haben wir zwei Fragen zu beantworten:

1. In welcher Richtung (links/rechts vom aktuellen zum nachfolgenden Platz gesehen) wird die an der Reihe befindliche Scheibe bewegt?
2. Welche Scheibe ist im entsprechenden Zug zu bewegen?

Beide Antworten sollen ohne Beweis gegeben werden.

Die Bewegungsrichtung der Scheiben wechselt mit ihrer Nummer; z. B.

Scheibe 1 rechts herum,
Scheibe 2 links herum,
Scheibe 3 rechts herum, ...

In jeder Situation gibt es normalerweise drei Zugmöglichkeiten, da drei verschieden große Scheiben oben auf den Teiltürmen liegen:

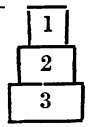
o H — von Z nach H
— von Z nach S
O S o Z — von H nach S.

Ein Zug scheidet aus, wenn er den Zug zuvor rückgängig macht und auch der Bewegungsrichtung der Scheibe widerspricht.

Ein Zug scheidet ebenfalls aus, wenn er dieselbe Scheibe wie im Vorgängerzug bewegt.

Zur Realisierung kann eine Dualzahldarstellung der Zugnummern vorgenommen werden, wie dies aus folgender Tabelle für $n = 3$ ersichtlich ist. Die in dieser Darstellung am weitesten rechts stehende Ziffer 1 bestimmt die im entsprechenden Zug zu bewegende Scheibe.

Scheiben-nummer	(4)	3	2	1		Zugfolge	
Zug-nummer							
1	0	0	0	1		1	r
2	0	0	1	0		2	l
3	0	0	1	1		1	r
4	0	1	0	0		3	r
5	0	1	0	1		1	r
6	0	1	1	0		2	l
7	0	1	1	1		1	r
(8)	(1)	0	0	0	→ ENDE		



Auf die Programmierung dieses Verfahrens wollen wir an dieser Stelle verzichten. Es gäbe keinerlei Gemeinsamkeiten mit dem Programm in Bild 11. Außerdem ist bereits von SCHÖNFELDER ein Assemblerprogramm mit einem ähnlichen, nichtrekursiven Algorithmus in Heft 5 abgedruckt.

Zum Schluß greifen wir nochmals die historisch zugrunde liegende und eingangs erwähnte indisch-vietnamesische Legende auf. Bei 64 Scheiben wären $2^{64} - 1$ Umlegungen nötig. Das ist eine Dezimalzahl mit 20 Stellen. Ihr Anfang lautet

$U = 1{,}84 \cdot 10^{19}$

Nimmt man für das Umlegen einer Scheibe genau 1 s an, (die Priester hätten mächtig flitzen müssen), so dauert der Prozeß

$$t = \frac{U}{60 \cdot 60 \cdot 24 \cdot 365} \approx 5{,}85 \cdot 10^{11} \text{ Jahre.}$$

So alt ist nach dem gegenwärtigen Wissen unser Kosmos noch nicht.
Insgesamt sollte durch diesen Beitrag deutlich werden, daß mit der Rekursion ein großartiges Mittel der Problemlösung zur Verfügung steht. Besonders mit LOGO und PASCAL aber auch mit BASIC lassen sich Programme, die rekursive Algorithmen realisieren, formulieren. Für BASIC gilt das wegen des intern in den Kleincomputern vorhandenen Rückkehrmechanismus (RETURN-Stapel). Bei FORTRAN und Großrechnern des ESER ist das nicht der Fall. Dann muß sogar dieser Stapel noch simuliert werden.

Autor:

Dr. rer. nat. *Christian Wagenknecht*
PH Dresden/Sektion Mathematik,
Wissenschaftsbereich Informatik

Rechentechnische Begriffe für den Laien erklärt

DRAM Dynamic RAM, RAM mit dynamischen Speicherelementen, die eines „Auffrischens" (Refresh) bedürfen; maximale Refreshperiode z.B. 2 ms. Schaltkreistypen: U 256 D (2116): 16K × 1 bit; U 2164 C: 64K × 1 bit; 21256: 256K × 1 bit.

SAM Serial Access Memory, RWM mit seriellem, i. allg. zeitlich geordneten Zugriff. Für kleine Informationsmengen (z.B. 1 Wort) wird das Schieberegister **SR,** Shift Register, verwendet.

CAM Content addressed Memory, inhaltsadressierter RWM mit einer durch einen Teil des Inhalts bestimmten Zugriffsordnung, auch als Assoziativspeicher bezeichnet.

Mikrorechnerbausatz Z 1013

1. Das Bausatzkonzept

Bei dem Bausatz handelt es sich nicht um einen Beutel mit Bauelementen, sondern um ein mehrere Baugruppen umfassendes Mikrorechnersystem. Es werden bestückte Leiterplatten angeboten, bei denen die Inbetriebnahme bereits beim Hersteller erfolgte. Der Nutzer des Bausatzes hat dann nur noch kleinere mechanische Arbeiten zu erledigen.

Es ist also augenscheinlich, wer als Nutzer angesprochen werden soll. Das sind vor allem die hardwareinteressierten Elektronikamateure. Anders als bei kompakten Heimcomputern wird der Bastler durch diese Bauform zum Eigenbau von Erweiterungen aufgefordert. Um das zu unterstützen, gibt es zu den einzelnen Baugruppen Handbücher mit ausführlichen Beschreibungen der Hardware.

Das Angebot des Z1013 – Sortimentes wird mit einer Grundausbaustufe eröffnet. Inhalt dieses Beitrages soll es nun sein, diese Baugruppe vorzustellen.

Es wird jedoch auf eine vollständige Beschreibung des Stromlaufplanes verzichtet. Auf Schaltungsdetails wird nur eingegangen, soweit diese für das Verständnis nötig sind.

Bei der Konzipierung der Grundausbaustufe wurde davon ausgegangen, daß der Käufer hiermit schon einen voll funktionstüchtigen Rechner mit allen zur Bedienung notwendigen Elementen erhält.

Im konkreten Fall heißt das, es ist eine Bildschirmsteuerung zum Anschluß eines Fernsehgerätes als Datensichtgerät und eine Tastatur zur Dateneingabe Bestandteil dieser Baugruppe. Demzufolge kann auch der an der Programmierung interessierte Computerfreund arbeiten.

2. Die Grundausbaustufe

Der Kunde erhält beim Kauf einer Grundausbaustufe

- eine bestückte und geprüfte Leiterplatte mit Zentraler Verarbeitungseinheit, Bildschirmsteuerung und Magnetbandsteuerung mit den Abmessungen 215 mm × 230 mm;
- Folienflachtastatur und ein Stück Bandkabel;
- Bedienungsanleitung.

Er hat außerdem die Möglichkeit, ein Handbuch mit ausführlicher Hard- und Softwarebeschreibung zu beziehen. Das ist aber für Erweiterungen der Hardware und Software unumgänglich.

2.1. Zentrale Verarbeitungseinheit

Die Zentrale Verarbeitungseinheit (ZVE) der Grundausbaustufe basiert

auf dem Mikroprozessor U 880. Die Beschaltung des Prozessors erfolgte entsprechend den bekannten Schaltungen, abgesehen von einigen Besonderheiten, auf die aber noch näher eingegangen wird.
Der Systemtakt wird von einem quarzstabilisierten Taktgenerator mit einer Frequenz von 8 MHz durch Teilung (Bild 1) gewonnen.

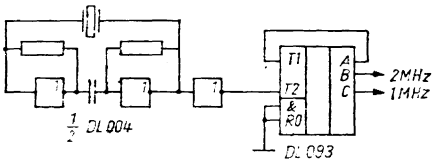


Bild 1. Takterzeugung

Es ist prinzipiell möglich, die ZVE mit 2 oder 1 MHZ zu betreiben. Die Funktion wird allerdings nur für 1 MHz vom Hersteller garantiert.
Die Grundausbaustufe ist mit dem 2 Kbyte ROM-Speicherschaltkreis U 2316, der das Betriebssystem enthält, und wahlweise mit 16 Kbyte dynamischen RAM (8 × K565RU3) oder 1 Kbyte statischen RAM (2 × U2114) als Arbeitsspeicher ausgerüstet. Die Speicherselektion erfolgt durch eine vollständig durchgeführte Adreßdekodierung, die acht Auswahl- (CS-) Signale liefert. Damit wird ein Adreßbereich von 8 Kbyte in 1 Kbyte-Schritten dekodiert. Dieser Bereich kann hardwaremäßig vom Anwender festgelegt werden. Damit sind dann die Adressen des Bildwiederholspeichers (BWS) und des Betriebsystems festgelegt (Bild 2).
Die Dekodierung für den DRAM erfolgt separat.
Das Betriebssystem der Grundausbaustufe beginnt ab Adresse F000H. Bekanntlich stellt sich jedoch der Befehlszähler der CPU nach aktivem RESET auf die Adresse 0000H ein. Eine spezielle Logik sorgt nun dafür,

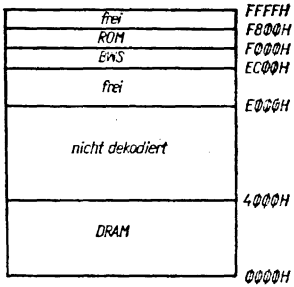


Bild 2. Speicherbereichaufteilung

daß nach RESET der Datenbus auf L-Pegel geschaltet wird. Die CPU liest also zuerst ein NOP und erhöht den Befehlszähler um eins. Ist schließlich die Adresse des Betriebssystems erreicht, wird mit dessen CS-Signal der Datenbus freigegeben.
Diese Lösung bietet den Vorteil, den RAM ab Adresse 0 zu legen. Da viele vorhandene Programme in den unteren Adreßbereichen arbeiten, wird so deren Implementierung erheblich erleichtert.
Für die Ein- und Ausgabe stehen eine PIO-Baugruppe (U 855) sowie ein spezielles Ausgabetor (Bild 3) zur Verfügung.

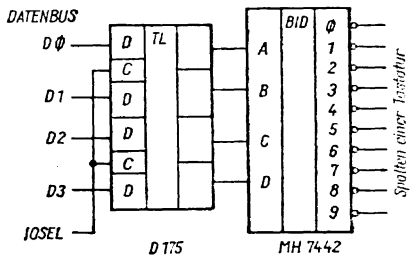


Bild 3. Tastaturspaltentreiber

Davon sind der PIO-Port B mit der Magnetbandein- und -ausgabe und in Verbindung mit dem Tastaturspaltentreibertor mit dem Tastaturanschluß in der Grundausbaustufe belegt. Der Port A steht dem Anwender zur Verfügung.

Die Dekodierung der E/A-Kanaladressen gestattet die Selektion besagter Kanäle in Tetradenschritten. Sie ist unvollständig, aber mittels einer von außen zugänglichen Leitung bis zur vollständigen Dekodierung expandierbar (Bild 4).

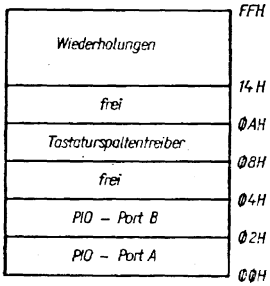


Bild 4. Aufteilung des E/A-Bereiches

2.2. Die Bildschirmsteuerung

Aus einer Zählkaskade werden die Synchron- und Bildaustastimpulse, die Inhalt des dem Fernsehgerät zuzuführenden Signals sind, dekodiert. Das ist bei der Eingangsfrequenz von 8 MHz besonders einfach. Diese Frequenz stellt außerdem auch die Bildpunktfrequenz dar. Die Zählerausgänge liefern direkt die Spalten- und Zeilenadressen der Zeichen, dessen Kodes im BWS abgelegt sind. Der BWS ist ein 1 Kbyte RAM (2 × U2114), der von der CPU beschrieben und gelesen werden kann. Es werden 32 Zeilen je Bild und 32 Zeichen je Zeile dekodiert.
Die Datenausgänge des BWS bilden die acht höherwertigen Adreßeingänge des 2 Kbyte Zeichengenerators (ZG). Der ZG stellt ein dem Zeichenkode entsprechendes Bitmuster zur Verfügung, das anschließend noch parallel-seriell gewandelt wird und so die Hell-Dunkel-Information des Bildsignals liefert.
Der Zeichengenerator enthält:

- Groß- und Kleinbuchstaben
- Ziffern 0 · · · 9
- Sonderzeichen
- etwa 128 Grafikzeichen, einschließlich Schachfiguren

Das Mischen der Bildinformation mit den Synchron- und Austastimpulsen ergibt das BAS-Signal, das dem Fernsehgerät entweder direkt oder über einen HF-Modulator zugeführt werden kann.
Die Bildschirmsteuerung arbeitet also absolut autonom und beansprucht keine Rechenzeit der CPU. Alle Signale werden zyklisch erzeugt. Nur bei einem Zugriff der CPU auf den BWS wird dieser Zyklus gestört.

2.3. Die Tastatur

Die Auswahl der Tastatur erfolgte nach dem Gesichtspunkt eines günstigen Preis-Leistungs-Verhältnisses. Es wird deshalb eine im Angebot befindliche Folienflachtastatur eingesetzt, die den Erfordernissen eines Minirechners genügt.
Die Grundausbaustufe erlaubt den Anschluß einer Tastaturmatrix von maximal 5 Zeilen mal 10 Spalten, 5 Eingabeleitungen des PIO-Port B und die 10 Ausgänge des Tastaturspaltentreibers. Für unsere Tastatur, die über Bandkabel mit der Grundplatine verbunden ist, werden 4 Zeilenleitungen und 8 Spaltenleitungen benötigt (Bild 5). Sie besitzt 32 Tasten, so daß sowohl hexadezimale als auch alphanumerische Eingaben möglich sind.
Das Umschalten geschieht durch spezielle Monitorkommandos (vgl. 3.3.)
Die Tastatur liegt beim Verkauf lose in der Verpackung und muß vom Anwender an das Bandkabel angelötet werden und dieses an die Grundplatine.

2.4. Das Kassetteninterface

Mit der bisher vorgestellten Konfiguration können wir Programme erarbeiten.

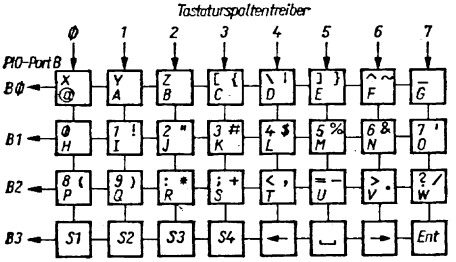


Bild 5. Anschluß und Kennzeichnung der Tastatur

Um diese beim Abschalten der Stromversorgung vor Verlust zu schützen, wurde in die Grundausbaustufe eine Anschlußelektronik für Magnetband- oder Kassettengeräte integriert.
Die Ausgabe der Daten erfolgt seriell über die PIO-Leitung B7 mit einer maximalen Übertragungsrate von 1200 bit/s. Der Sender des Kassetteninterfaces besteht aus einem einfachen Spannungsteiler, der den TTL-Pegel auf 60 ··· 125 mV absenkt und dem Diodeneingang eines Magnetbandgerätes zuführt. Der Empfänger ist ein Operationsverstärker, mit dem die Pegel des vom Magnetbandgerät kommenden Signals in den TTL-Bereich transformiert werden.

2.5. Die Stromversorgung

Die Lösung für die Stromversorgung der Leiterplatte erfordert vom Anwender nur minimalen Aufwand. Auf der Grundplatine befinden sich die Gleichrichterschaltung, die Ladekondensatoren und die Spannungsstabilisatorschaltung für die drei benötigten Spannungen + 5 V, — 5 V und + 12 V. Von außen muß demzufolge nur noch eine Wechselspannung von etwa 12 V bei einer Stromaufnahme von etwa 1 A zugeführt werden, was z. B. ein handelsüblicher Heiztrafo liefert.
Die Reserven des Netzteiles sind eng begrenzt, so daß bei Erweiterungen des Systems es unbedingt nötig ist, diese mit einer externen Stromversorgung zu betreiben.

2.6. Das Bussystem

Das Format der Grundausbaustufe des Mikrorechnerbausatzes ist so gewählt, daß sie in ein K 1520-Gefäß eingeschoben werden kann. Die Realisierung der wichtigsten Signale des K 1520-Systembusses erlaubt dann unter Beachtung des unter 2.5. gesagten die Verwendung von K 1520-Platinen zur Erweiterung. Es besteht außerdem die Möglichkeit, über einen entsprechenden Adapter die Erweiterungsmodule des Kleincomputers KC 85/1 zu nutzen.
Gegenüber der Systemsteckverbinderseite kann über eine Diodenbuchse das Magnetbandgerät angeschlossen werden, über eine HF-Buchse das BAS-Signal für den Anschluß eines Fernsehgerätes entnommen werden und über einen 15poligen Steckverbinder der PIO-Port A erreicht werden. Die Tastatur wird ebenfalls an dieser Seite angelötet.

3. Der Monitor des Z 1013

3.1. Die Speicherbelegung der Systemsoftware

Das Monitorprogramm des Z 1013 umfaßt 2 Kbyte. Es steht ab Adresse F000H. Der Monitor benötigt als Arbeitsspeicher den RAM-Bereich ab 0003H bis 00B0H. In der Z 1013-Dokumentation werden alle wichtigen Adressen von Systemzellen genannt, um ein rationelles Arbeiten mit dem Monitorprogramm zu ermöglichen.
Anwenderprogramme können ab 00B0H stehen. Ein 3-Kbyte-BASIC-Interpreter wird als Listing mitgeliefert und muß ab Adresse 0100H eingegeben werden. Unterstützt wird eine

fehlerfreie Eingabe durch Vergleich mit den in der Liste angegebenen Prüfsummen. Er kann so schrittweise auf Magnetbandkassette aufgebaut werden.

3.2. Das Arbeitsprinzip des Monitors

Das Monitorprogramm wurde unter dem Gesichtspunkt eines minimalen Speicherplatzbedarfes, sicherer Funktion, einfacher Handhabbarkeit, der Bereitstellung günstiger Testmöglichkeiten und der Mehrfachnutzung oft verwendeter Grundroutinen geschaffen. Alle Eingaben im Monitormodus erfolgen in hexadezimaler Form. Sie werden auf dem Bildwiederholspeicherbereich zwischengespeichert, können bis zur Betätigung der ENTER-Taste korrigiert werden, und erst nach ENTER erfolgt die Umwandlung der Zeichenketten in das interne Format.
Nach ENTER wird die entsprechende Programmfunktion abgearbeitet, wobei z.B. Bildschirmausschriften, Veränderungen von Arbeitsbereichen und das Lesen bzw. Schreiben des Kassetteninterfaces usw. erfolgen. Ist das Kommando vollständig abgearbeitet, meldet sich der Monitor wieder mit Eingabebereitschaft und erwartet ein neues Kommando.
Für die problemlose Benutzung von Grundroutinen des Monitors wurde der RST 20H festgelegt. Dabei muß hinter dem RST 20H eine weitere Spezifizierung über eine Datenbytedefinition erfolgen, z.B. bedeutet:

```
E7      RST 20H
00      DB 0
```

die Ausgabe eines Zeichens, welches im A-Register bereitgestellt werden muß (siehe auch 3.4.).
Im Monitorprogramm werden die über RST 20H nutzbaren Grundroutinen ebenfalls in dieser Form genutzt.

3.3. Die Kommandos des Monitorprogramms

Nach dem Einschalten und RESET wird das Monitorprogramm des Z 1013 auf der Adresse F000H gestartet. Der Monitor meldet sich mit der Ausschrift »Z 1013« auf dem Bildschirm. Auf der folgenden Zeile erscheint ein Doppelkreuz als Quittungssymbol (auch Promptsymbol genannt) und der Cursor als Zeichen für die Eingabebereitschaft.
Nun wird ein Kommando in der allgemeinen Form:

XY aaaa bbbb cccc ENTER

erwartet.
Für X können die im folgenden erläuterten ASCII-Zeichen als Abkürzungen mit fest zugeordneten Monitorfunktionen benutzt werden. Y muß dann ein Leerzeichen (Space) sein. Es ist aber auch möglich, die Monitorfunktionen durch zusätzliche Programmteile im RAM selbst zu erweitern.
Will man diese erweiterten Funktionen aus der Kommandoschleife des Monitors heraus benutzen, so muß für X das Zeichen @, gefolgt von einem zweiten vorher selbst festgelegtem ASCII-Zeichen und eventuell notwendigen Argumenten, eingegeben werden. Die als ASCII-Zeichen verwendete Abkürzung entspricht meist einem Schlüsselbegriff aus der englischen Sprache.
Für aaaa, bbbb und cccc können drei maximal vierstellige Argumente eingegeben werden, die dann eine spezielle Bedeutung je Kommando haben. Die Eingabe führender Nullen ist nicht erforderlich. Zwischen den Argumenten ist die Eingabe eines Leerzeichens als Trennzeichen notwendig. Mehrere Leerzeichen zwischen den Argumenten sind erlaubt.
Die bereits eingegebene Kommandozeile kann mit den Tasten Cursor

links/Cursor rechts korrigiert werden. Vor Drücken der ENTER-Taste muß der Cursor hinter dem letzten gültigen Zeichen stehen. Nach ENTER erfolgt die Analyse der Kommandozeile. Ist das eingegebene ASCII-Zeichen nicht in der Kommandotabelle des Monitors enthalten, so erscheint auf der Folgezeile ein Fragezeichen. Sind die abgespeicherten Argumente von vorherigen Kommandos noch gültig, so kann X: eingegeben werden.
Die möglichen Kommandos des Monitors werden in alphabetischer Reihenfolge erläutert:

A (Alpha)

Umschaltung der Tastaturkodetabelle auf alphanumerische Zeichen in der Shift-Ebene Null, wenn vorher das Kommando H (siehe dort) benutzt wurde.

B aaaa (Breakpoint-Eingabe)

aaaa ist eine Adresse im RAM, auf die ein Softwarebreakpoint gesetzt wird. Zur Kontrolle werden automatisch der eingetragene Breakpoint BP:..., die Breaksequenz BS:... und alle aktuellen Registerinhalte angezeigt.

C aaaa bbbb cccc (Compare)

Es wird der Speicherinhalt ab Adresse aaaa mit dem ab Adresse bbbb in einer Länge von cccc byteweise miteinander verglichen. Bei auftretenden Differenzen werden die entsprechenden Adressen und Byte angezeigt.

D aaaa bbbb (Display Memory, Dump)

Der Speicherbereich ab Adresse aaaa wird bis zu Adresse bbbb in der Form

Anfangsadresse	8 Datenbyte	3stellige Prüfsumme der 8 Datenbyte

ausgegeben.

E aaaa (Execute Machine Program)

Ein ab Adresse aaaa stehendes Maschinenkodeprogramm wird unter Breakpointkontrolle gestartet. Das Programm wird von aaaa bis zum Breakpoint abgearbeitet.

F aaaa bbbb aa bb cc ... (Find)

Ab der Adresse aaaa wird die Bytefolge aa bb cc ... mit der Byteanzahl bbbb gesucht. Wird sie gefunden, erfolgt ein Übergang zum M-Kommando, sonst die Ausschrift »NOT FOUND«.

G (Go On)

Dieses Kommando wird zur Fortsetzung eines Programmes nach erfolgtem Softwarebreakpoint angewendet. Wurde ein neuer Breakpoint gesetzt, so wird dieser aktiviert.

H (Hexadezimal)

Nach Einschalten befindet sich die Tastatur im Alpha-Modus, wie nach Kommando A. Mit dem Kommando H werden in der Shift-Ebene Null die Tasten H ... W auf die Zeichen 0 ··· 7, 8 ··· ? umgeschaltet.

I (Initialize)

Mit diesem Kommando werden die Register im Registerrettebereich gerettet. Die weitere Programmfolge ist wie bei Neustart des Monitors mittels RESET.

J aaaa (Jump)

Sprung zum Anwenderprogramm ab Adresse aaaa. Es erfolgt keine Breakpointaktivierung.

K aaaa bbbb cc (Kill)

Der Speicher wird von Adresse aaaa bis Adresse bbbb mit dem Byte cc gefüllt.

L aaaa bbbb (Load From Cassette)

In den Adreßbereich von aaaa bis bbbb wird ein Programm bzw. Daten von Kassette geladen.

M aaaa (Modify and Inspect Memory)

Mit diesem Kommando kann der RAM-Speicher ab Adresse aaaa byteweise angezeigt und verändert werden. Nach ENTER wird die aktuelle Adresse um eins erhöht.

N (Next Instruction)

Das N-Kommando kann nur nach vorheriger Breakpointeingabe und einem Programmstart über das E-Kommando ausgeführt werden. Es wird nach Eingabe des Kommandos N bei Betätigung der ENTER-Taste der nächste Befehl abgearbeitet, und es werden erneut alle Register angezeigt.

R XY (Register Display and Modify)

Mit diesem Kommando lassen sich die Inhalte aller Registerpaare XY aus dem Registerrettebereich anzeigen und verändern.

S aaaa bbbb (Save to Cassette)

Der Speicherbereich von Adresse aaaa bis zur Adresse bbbb wird über das Magnetbandinterface auf Kassette ausgegeben.

T aaaa bbbb cccc (Transfer)

Es werden cccc Byte von der Adresse aaaa nach der Adresse bbbb transportiert.

W aaaa bbbb (Window)

Festlegung eines Rollfensters im Bildschirmbereich. Das Fenster kann nur für Zeilen eingestellt werden.

3.4. Die RST 20H-Routinen

Die über RST 20H erreichbaren Monitorroutinen sind nach der Größe des zu spezifizierenden Datenbytes geordnet:

OUTCH

Ausgabe des im A-Register stehenden Zeichens über den Videotreiber.

INCH

Eingabe eines Zeichens von der Tastatur in das A-Register.

PRST7

Die dem RST 20H folgende Zeichenkette wird ausgegeben, bis das 7. Bit gesetzt ist.

INHEX

Die Routine wandelt eine im ASCII-Kode eingegebene Zeichenfolge in das interne Format.

INKEY

Abfrage der Tastatur nach dem Polling-Prinzip.

INLIN

Eingabe einer Zeile mit führendem Promptsymbol.

OUTHX

Ausgabe des A-Registers hexadezimal. Es werden pro Byte zwei Zeichen ausgegeben.

OUTHL

Ausgabe des HL-Registers hexadezimal. Es werden vier Zeichen ausgegeben.

CSAVE

Auslagern von MC-Programmen auf Kassette.

CLOAD

Laden eines Programmes von Kassette.

MEM

Entspricht M-Kommando

WIND

Entspricht W-Kommando

OTHLS

Ausgabe von zwei Byte hexadezimal entsprechend der Adresse im HL-Register.

OUTDP

Ausgabe eines Doppelpunktes

OUTSP

Ausgabe eines Leerzeichens

TRANS

Entspricht T-Kommando

INSTR

Eingabe einer Zeichenkette bis ENTER

KILL

Entspricht K-Kommando

HEXUM

Entspricht H-Kommando

ALFA

Entspricht A-Kommando

3.5. Das Minimal-BASIC

Da das BASIC aus Kostengründen in Papierform angeboten werden muß, mußte ein Kompromiß zwischen Leistungsfähigkeit und Länge gefunden werden. Das hier mitgelieferte BASIC ist vor allem zum Einarbeiten in diese Programmiersprache gedacht, bietet aber auch dem versierten Programmierer zahlreiche Möglichkeiten.
An dieser Stelle soll nur eine Übersicht über Kommandos und Befehle gegeben werden:

- Kommandos:
  BYE, END, LIST, NEW, RUN
- Befehle:
  BYTE, CALL, CLOAD, CSAVE, FOR, GOSUB, GOTO, IF, INPUT, I$, LET, NEXT, OUT, OUTCHAR, O$, POKE, PRINT, REM, RETURN, STOP, TAB, WORD
- Funktionen:
  ABS, CSTS, HEX, IN, INCHAR, LEN, PEEK, RND, SIZE, TOP
- Vergleichsoperanden:
  =, <, >, <=, > =

4. Ausblick

Die Grundausbaustufe des Mikrorechnerbausatzes Z 1013 wird bereits seit Ende 1985 beim VEB Robotron-Vertrieb Erfurt verkauft.
Ab dem 3. Quartal 1987 soll dieses Angebot durch Erweiterungsbaugruppen ergänzt werden. Dazu gehören:

1. Busverstärkerbaugruppe
   Diese Baugruppe wird am Systemsteckverbinder der Grundausbaustufe angesteckt. Sie enthält die Logik zur Verstärkung der Systemsignale und vier Steckplätze für Erweiterungsmodule. Diese Steckplätze sind steckerkompatibel zu den Erweiterungsbaugruppen des KC 85/1.

2. Stromversorgungsmodul
   Das Modul stellt alle benötigten Spannungen mit den nötigen Reserven für das Betreiben von Erweiterungen zur Verfügung.
3. ROM-Modul
   Auf dieser Baugruppe befinden sich vier Steckplätze für ROM-Speicherschaltkreise, deren Kapazität mit Hilfe von Brücken festgelegt werden kann.
4. E/A-Modul
   Diese Baugruppe beinhaltet zwei PIO U855, von deren vier Ports drei dem Anwender zur Verfügung stehen und das vierte für das Bilden einer seriellen Schnittstelle genutzt wird.

Dem Bedarf an umfangreicherer und leistungsfähigerer Software wird durch das Angebot einer Softwarekassette mit einem größeren BASIC-Interpreter, der weitestgehend mit dem HC-BASIC übereinstimmt und von dem VEB Meßelektronik »Otto Schön« Dresden stammt, und einem editierenden Assembler für U880 Rechnung getragen.

Autoren:

Dipl.-Phys. Dietmar Schwitalla
Dipl.-Ing. Eberhard Müller

Mitarbeiter F/E im
VEB Robotron-Elektronik Riesa

Computer-Zeitschriften in der VR Polen

In der VR Polen haben Klein- und Mikrorechner sowie Personalcomputer in den letzten Jahren eine starke Verbreitung erfahren. In großen Stückzahlen gelangten Rechner der Typen ZX81, ZX Spectrum und ZX Spectrum Plus, deren Endmontage in der VR Polen erfolgt, sowie die Eigenentwicklungen MERITUM I und II zum Einsatz.

Die genannten Computertypen werden zum größeren Teil von Wissenschaftlern, Studenten oder Amateuren individuell genutzt, aber auch in Betrieben und Schulen verwendet. Vorzugsweise für die jüngeren Nutzer erscheint seit 1985 die Zeitschrift **Bajtek**, Z MIKROKOMPUTEREM NA TY (Mit dem Mikrocomputer auf Du). Der Titel stellt eine Wortbildung aus „Byte" und der Endsilbe für männliche Vornamen „... ek" dar. Sie erscheint monatlich, mehrfarbig und im Format A4 und wendet sich an die Schüler der Grundschulen. Sie vermittelt einführende Kenntnisse in die Computertechnik und ihre Anwendungen und stellt einfache Lehr- und Spielprogramme für verschiedene Mikrocomputertypen vor. Diese sind inhaltlich z. T. so angelegt, daß sie bereits von Schülern der untersten Klassenstufen, ja ggf. von Vorschulkindern bei Unterstützung durch die Erzieher, angewendet werden können.

Der Unterstützung der Einführung von Computern in der Volkswirtschaft und im Bildungswesen sowie bei den individuellen Nutzen dient die Zeitschrift **KOMPUTER**, POPULARNY MIESIECZNIK INFORMATYCZNY (Populäre Informatikmonatsschrift). Seit 1986 im Format A4, mehrfarbig, monatlich erscheinend, wendet sich diese Zeitschrift an die Erwachsenen und an Schüler der Oberstufe. Sie berichtet über Technik und Software von Kleincomputern, vermittelt Hinweise und Erfahrungen beim Erarbeiten von Programmen sowie Programme für die hauptsächlich zum Einsatz kommenden Computertypen selbst. Auch Artikel zur Vermittlung von Grundkenntnissen der Informatik sind enthalten. Es werden Algorithmen und daraus erarbeitete Programme vorgestellt. Über neue Rechnertypen, Ergänzungsbaugruppen, Programme, Ersatzteile usw. informieren polnische Importfirmen und Herstellerbetriebe.

Dozent Dr.-Ing. *W. Leupold*, Technische Universität Dresden
Dr.-Ing. *A. Stepien*, Polytechnische Hochschule Wroclaw

Programmieren für den »non sophisticated user« – Empfehlungen

1. Zielstellung

Die Mikroelektronik machte es möglich: Wir leben in einer Zeit, in der der Computer zum Spiel-, Schul-, Heim-, Büro- oder Arbeitsplatzcomputer wird, d.h. zum »persönlichen Computer« PC, der sich für die verschiedenartigsten Aufgaben nützen läßt. Damit ist öfter der Fall verbunden, daß er für Nutzer programmiert werden muß, die keine Informatiker sind und es auch nicht werden wollen. Das kann der Geographielehrer sein, der Lehrsoftware für seinen Unterricht verwendet, die Arzthelferin, die ein Patientenregister führt, die Köchin, welche ihre Vorräte und Speisepläne mit dem Kleincomputer verwaltet usw. usw. Verwenden wir für diesen hier betrachteten, sich äußerst differenzierten Personenkreis die Bezeichnung »non sophisticated user«[1] (NSU), ohne darin eine Abwertung zu erblicken; der NSU benötigt seine Intelligenz, Arbeitskraft und Aufmerksamkeit für den Aufgabenkreis, den er zu lösen hat. Der PC ist für ihn nichts als ein Werkzeug, von dem er einfachste, sinnfällige Bedienbarkeit, vollständige Transparenz, Schutz gegen Fehlhandlungen sowie Hilfe in der Lernphase und bei Störungen, aber keine zusätzlichen Anstrengungen erwartet. Zumindest fordert er zu Recht diesbezüglich einen angemessenen Kompromiß.

Die NSU kann man in zwei Hauptklassen einteilen, die grob den klassischen Nutzertypen »Teilhaber« und »Teilnehmer« analog sind:

- Nutzer NSU 1, die **ein** Applikationsprogrammsystem für ihren festen, exakt definierbaren Aufgabenkreis gebrauchen (Beispiel: Fahrkartenverkäufer an der computergestützten Schaltermaschine);
- Nutzer NSU 2, denen ein bausteinförmiges Instrumentarium (z.B. Editor, Compiler, Monitor, Dateiverwaltung und Applikationssoftware von allgemeinem Charakter wie Textsystem, Statistikpaket, Buchhaltung, Registerführung u.ä.) zur Verfügung gestellt wird und die damit ihre relativ veränderlichen Aufgaben lösen (Beispiel: Analysentätigkeit im Gesundheits- und Arbeitsschutz). In einer gehobenen Stufe schreiben sie sogar – z.B. mit BASIC – zusätzliche Applikationsprogramme selbst.

Will man für solche Nutzer programmieren und implementieren, sind gewisse Unterschiede zu beachten, welche durch die Nutzerklassen bedingt sind. Für beide Fälle aber gilt: Die Software, die man erarbeitet und zur Ver-

[1]) Das heißt: nicht speziell (in Informatik) ausgebildeter Nutzer.

fügung stellt, hat zwei wesentliche Komponenten:

- den »Arbeitskern«, der die gestellten Anwendungsaufgaben löst, und
- die nutzerfreundliche »Verpackung«, welche die problemlose Nutzung des Arbeitskerns und des PC selbst ermöglicht.

Anliegen des Beitrages ist es, relevante Aspekte zur zweiten Komponente in kompakter Form darzustellen, ohne daß die Autoren den Anspruch erheben, diese selbst erfunden zu haben. Tatsächlich handelt es sich im einzelnen um wohlbekannte Dinge. Neu ist, daß sie für die PC-Programme, welche von NSU genutzt werden sollen, in wesentlich höherer Qualität und Komplexität berücksichtigt werden müssen, als es bisher in der »großen Datenverarbeitung« nötig und möglich war.

2. Zur Erleichterung der Lern- und Trainingsphase

Über die Effektivität des Einsatzes eines PC entscheidet vorrangig, wie der NSU damit zurecht kommt und wie dieser seine Arbeitsprozesse unterstützt. Dagegen sind solche Aspekte wie benötigte Speicherkapazität, Reaktionszeit und zeitliche Auslastung des PC oft sehr wenig relevant, und man sollte sich hüten, an diesen Stellen zu sparen, wenn dadurch die Effektivität in den Arbeitsprozessen auch nur im mindesten angegriffen wird.

Für die erste Bekanntschaft mit dem PC, d. h. aus der Anlern- und Trainingsproblematik, lassen sich Empfehlungen formulieren, die in Tafel 1 zusammengestellt sind. Hierin liegt ein wesentlicher Faktor für einen erfolgreichen Anfang, welcher bekanntlich viel entscheidet.

3. Zur Gestaltung der Mensch-Computer-Kommunikation

Tätigkeit am PC ist vorrangig Dialogarbeit mit den Kommunikationsmitteln Tastatur und Display. Die Text- oder Bildstrukturen auf dem Bildschirm sollen als »Masken« bezeichnet werden. Sie werden von drei Verfahren genutzt:

a) Mit dem Führungsverfahren **Menütechnik** bietet man mittels fester Maske dem NSU in Textform Möglichkeiten an, aus denen er eine durch Drücken einer im Text genannten Taste auswählen kann. Damit wird er zu der von ihm gewünschten Teilaufgabe geführt.

b) Bei dem Eingabeverfahren **Formulartechnik** sind die Masken wie Papierformulare gestaltet, in die in vorgeschriebene Felder mittels der Tastatur variable Daten wie Bezeichnungen, Namen, Adressen, Zahlen, Abkürzungen usw. eingetragen werden.

c) **Hilfefunktionen** stellen eine computerinterne Form des Bedienerhandbuches dar. Die festen Masken sind »Seiten« des Buches, in dem »geblättert« werden kann, eventuell sehr zielstrebig unter Nutzung der Menütechnik.

Mit der aufgabenangepaßten Kombination der drei Verfahren kann man einen hohen Grad von »Bedienerfreundlichkeit« erzielen. Für die Gestaltung der Masken und ihre programmtechnische Umgebung sollte man die Empfehlungen der Tafel 2 beachten.

Auf drei Aspekte sei gesondert hingewiesen:

a) Bei einem umfangreichen Menü mit logischen Abhängigkeiten zwischen den Teilaufgaben (Verzweigungen) muß man es in Teilmenüs zerlegen. Der NSU hangelt sich dann mittels Blättern über mehrere Masken zum gewünschten Ziel durch. Ist er aber trainiert, empfindet er diesen narrensicheren Weg als zu umständlich.

Tafel 1. Empfehlungen zur Minimierung der Lern- und Trainingsphase

1. Gestalte das System so, daß das Anlernen und Trainieren des Nutzers Stunden, nicht Tage oder gar Wochen erfordert!

2. Vorteilhaft ist es, wenn sich der PC vollständig selbst erklärt, so daß sich der NSU in kurzer Zeit allein einarbeiten kann.

Das reduziert Probleme, wenn ein Nutzer plötzlich ausfällt und durch einen nicht eingearbeiteten Mitarbeiter ersetzt werden muß.

3. Bevorzuge die residente Abspeicherung der Betriebssystem- und der unveränderlichen Applikationssoftware im ROM-Bereich des Arbeitsspeichers!

Reicht die interne Speicherkapazität dafür nicht aus, so organisiere ein automatisches Laden oder – wenn das nicht möglich ist – ein einfaches, vom Bildschirmtext geführtes Laden (z.B. nach Bild 1) von den externen Speichern! Verfahre analog mit der Bedienung sämtlicher peripherer Geräte (z.B. Drucker)!

4. Verzichte auf die traditionelle, durch Anglizismen und Sonderzeichen geprägte EDV-Mnemonik! Verwende für die notwendigen Symbole sinnfällige und selbsterklärende deutsche Bezeichnungen! Spare nicht mit den Buchstaben!

5. Mehr als ein Blatt A 4 mit Bedieneranleitung sollte nicht über dem PC hängen! Ein leicht verständliches (dünnes!) Bedienerhandbuch kann man eventuell für den NSU 2 vorsehen.

6. Verachte nicht die Computerspiele! Sie sind hervorragend geeignet, Nutzer in elementare Bedienvorgänge des PC einzuführen, sie zu interessieren und erste Erfolgserlebnisse zu verschaffen. Man muß nur rechtzeitig mit dem Spielen aufhören! Aus manchem »Spieler« wurde schon ein hervorragender Hobby- oder Berufsinformatiker. Dies gilt insbesondere für den NSU 2.

LEGEN SIE BITTE DIE SYSTEMDISKETTE IN LAUFWERK 0 UND DRUECKEN SIE
DIE ENTER-TASTE

DAS ERSCHEINEN DER FEHLERAUSSCHRIFT »DISKETTEN-FEHLER« KANN FOLGENDE URSACHEN HABEN:
LAUFWERK NICHT RICHTIG VERRIEGELT
DISKETTE FALSCH EINGELEGT
DISKETTE DEFEKT
DISKETTE IST KEINE SYSTEMDISKETTE
GERÄTETECHNISCHER FEHLER

VIEL ERFOLG BEI DER ARBEIT!

Bild 1

Deshalb sollte man einen Parallelmodus vorsehen, mit dem über einen Suchbegriff oder eine spezielle Funktionstaste direkt zum Ziel gesprungen wird.

b) Sehr kleine Menüs können auch direkt auf Funktionstasten dargestellt werden, so daß eine Tastenbetätigung zum Ziel führt. Falsche Sparsamkeit aber ist, eine mehr oder minder will-

Tafel 2. Empfehlungen zur Maskengestaltung

1. Strukturiere die Masken so, daß sie übersichtlich und schnell verständlich sind! Benütze ein einheitliches Design!

2. Nutze maximal nur etwa 50% der Zeichenplätze auf dem Bildschirm und bringe nicht mehr als 30 bis 40 Zeichen auf der Zeile unter!

3. Weise alphabetische Feldinhalte linksbündig, numerische rechtsbündig aus!

4. Programmiere für die Formularausfüllung den Cursor so, daß er automatisch nach Bestätigung einer Feldeingabe (z.B. mit der ENTER-Taste) zum Beginn des nächsten Eintragungsfeldes springt!

5. Sind in Eintragungsfelder verschiedene Angaben (z.B. Staatsangehörigkeiten) einzutasten, so erkläre die häufigste davon (z.B. »DDR«) zur Standardannahme, welche automatisch eingetragen wird, wenn ohne Feldeingabe die ENTER-Taste gedrückt wird!

6. Muß das Formular gerollt oder geblättert werden, so automatisiere dieses über die Bestätigung der letzten Feldeingabe auf der Maske!

7. Nutze die Möglichkeiten der Bild- und Farbdarstellungen, wo sie dem Applikationsfall angepaßt sind! Ein Diagramm spricht oft schneller an als eine Zahlenkolonne. Farben haben Signalwirkungen.

kürliche Zuordnung von alphanumerischen Tasten zu Menüalternativen als »second function« vorzunehmen. Zweit- oder gar Drittbelegungen von Tastaturen lassen die Bedienungsfehler stark anwachsen.

c) Die Hilfefunktionen können den NSU unterstützen mit

- Erläuterungen zu Funktionen, Befehlen, Kommandos;
- Hinweisen zu Fehlerursachen und Fehlerkorrekturen;
- Ausgabe zusätzlicher Sachinformationen;
- Syntaxerklärungen;
- Hinweisen zur Bedienung und Handhabung der Gerätetechnik.

Der Aufruf kann über eine Funktionstaste oder ein geeignetes Symbol (»?«) erfolgen.

Die Erstellung eines Gerüstes von Hilfefunktionen erfordert eine ausführliche Systemanalyse und ist mit hohem Programmieraufwand verbunden. Er wird in den Applikationsprogrammen kaum sichtbar und deshalb mit der Begründung »Programmierökonomie« leider noch zu oft eingespart.

4. Zur Gewährleistung der Betriebssicherheit

Betriebssicherheit soll hier im allgemeinsten Sinne verstanden werden, wie es auch die Empfehlungen in Tafel 3 ausweisen. Ein System, das durch Unzulänglichkeiten die Qualität und den Fortschritt der Arbeit herabsetzt oder gar zu längeren Ausfällen der Arbeitsfähigkeit führt, wird bald in die Ecke gestellt werden und dort verstauben.

Demzufolge kann auch die Verantwortung des Systementwicklers nicht mit der Übergabe an den NSU, sondern erst dann enden, wenn das System alle seine Aufgaben stabil erfüllt.

Tafel 3. Empfehlungen für die Gewährleistung der Betriebssicherheit

1. Erarbeite vor der Systemkonzipierung eine Analyse der Fehlermöglichkeiten, z. B. falsche Zuordnungen, fehlerhafte Kodierungen (auch Abkürzungen), falsche Bedienhandlungen, technische Störungen usw.! Füge in das Systemkonzept auch ein möglichst dichtes Netz für das Erkennen und Korrigieren dieser Fehler ein!

2. Informiere Dich über die Datenschutzforderungen des Applikationsfalles! Beachte sie insbesondere

- bei der Zugriffsverwaltung für Dateien,
- bei der Gestaltung von Ausdrucken für die einzelnen unterschiedlichen Adressaten,
- bei der Gestaltung der Masken, wenn unterschiedliche Nutzer mit dem PC arbeiten müssen,
- bei der Gestaltung der Möglichkeiten für den direkten Zugriff zu Arbeits- und externen Speichern (Speicherabzug!)!

3. Schütze das System gegen alle voraussehbaren »Systemzusammenbrüche«! Treten sie dennoch auf, muß ein einfaches Wiederanlaufen vorbereitet sein.

4. Gib die Tastatur nur dann frei, wenn Eingaben oder Steuerfunktionen zu betätigen sind! Schließe dabei alle unzulässigen Tasten von der Freigabe aus!

5. Prüfe alle Feldeingaben in Formularmasken und auch sonst automatisch auf syntaktische Richtigkeit und Plausibilität (auf zugelassene Zeichenklassen, Prüfziffern, Grenzwerte von Zahlen usw.), am besten unmittelbar nach Betätigung der ENTER-Taste! Sieh eine geeignete Korrekturmöglichkeit vor! Du hast folgende Möglichkeiten bei Eingabefehlern:

- automatisches Löschen des Feldes und Rückführen des Cursors;
- Hervorheben fehlerhafter Zeichen (z. B. durch Blinken) und Korrektur mit tastaturgesteuerter Cursorrückführung;
- Ausgabe einer Fehlerinformation (aber nicht eines Fehlerkodes, der erst mit einer Liste entschlüsselt werden muß!) – nur bei komplizierteren, nicht unmittelbar einsichtigen Fehlern zu benützen.

6. Sichere, daß notwendige Eingaben auch durchgeführt werden müssen, z. B. indem der Übergang zum nächsten Feld ohne erfolgte Eingabe blockiert wird.

7. Gib dem NSU eine Möglichkeit, zu prüfen, in welchem Status und bei welcher Aufgabe sich das System gerade befindet! Er kann in der Arbeit gestört werden, möchte aber seine Aufgabe unmittelbar fortsetzen können.

8. Sichere, daß zu erhaltende Datenbestände auf externen Speichern und die Programme nicht durch beabsichtigten oder unbeabsichtigten Nutzereingriff zerstört werden können!

9. Nutze die Hilfefunktionen, um die Bediensicherheit des NSU zu erhöhen!

10. Wähle nur solche technischen Komponenten aus, deren Betriebssicherheit den Forderungen an das System entspricht!

11. Nutze die Möglichkeit, durch Testprogramme die Arbeitsfähigkeit der technischen Komponenten und der Software nachprüfen zu können!

12. Organisiere Havarielösungen für Teil- oder Gesamtausfälle des Systems, insbesondere wenn es als bestimmendes Arbeitsmittel in einen nicht anzuhaltenden Prozeß eingefügt ist!

13. Regele die rechtzeitige und kontinuierliche Bevorratung mit Hilfsmaterialien (Papier, Formulare, Disketten usw.)!

14. Regele rechtzeitig den Service für die periodische Wartung und bei Störungen!

15. Dokumentiere die von Dir erarbeitete Software gründlich und für andere verständlich!

Tafel 4. Empfehlungen für die Motivierung der Nutzer

1. Gewinne das Vertrauen und die aktive Mitarbeit der zukünftigen NSU! Ziehe sie bei der Problemanalyse und der Systemkonzeption zu Rate! Analysiere und berücksichtige alle Verästelungen und Konsequenzen ihrer Arbeitstätigkeit!

2. Sorge dafür, daß die NSU einen deutlichen Nutzen und Arbeitserleichterungen durch die Einführung des Systems verspüren, z. B. durch

- leichteren und schnelleren Zugriff zu den benötigten Informationen,
- Wegfall **aller** mehrfachen Schreibarbeiten,
- automatisierte Dokumentation und Archivierung,
- bessere Auskunftsmöglichkeiten,
- verbessertes Ordnungssystem,
- automatisiertes Anfertigen von Rationalisierungsmitteln (z. B. Klebeetiketten mit Adressen), Registern, zusammenfassenden Berichten, Analysen usw.,
- Anbieten neuer Arbeitsmöglichkeiten (z. B. Aufbau und Nutzung persönlicher Datenbanken, Nutzung eines Textverarbeitungssystems).

3. Präsentiere die Resultatdaten auf dem Bildschirm oder im Druckbild so, daß sie ohne Schwierigkeiten verwertet werden können, d. h.:

- Präsentiere dem »Endverbraucher« der Daten nur solche, die er für seine Entscheidungen oder seine Arbeit braucht!
- Ordne die Daten in geeigneter Form (z. B. sachlich oder nach Bedeutung)!
- Bezeichne die Daten verständlich und eindeutig!
- Wähle die dem Entscheidungs- oder Arbeitsprozeß am besten angepaßte Darstellungsform aus!

4. Schaffe Nutzeffekte gemäß Pkt. 2 auch für die Werktätigen, die nun nach neuen Vorschriften Daten an das System liefern müssen oder/und deren Tätigkeit organisatorische Änderungen erfährt!

5. Bedenke, daß dabei bei allen Betroffenen persönliche Vorbehalte und Kompetenzstreitigkeiten, aber auch Ängste bezüglich der »komplizierten« Technik entstehen können! Baue sie rechtzeitig und gründlich ab!

6. Beachte die physische und psychische Belastung an den Rechnerarbeitsplätzen (Bildschirmarbeit!)! Sichere, daß die entsprechenden Vorschriften des Gesundheitsschutzes eingehalten werden können! Sorge für eine abwechslungsreiche Arbeit durch Kombination mit anderen Tätigkeiten!

7. Schaffe eine ausgeglichene Arbeitsatmosphäre ohne Hektik! Sorge für eine helle, freundliche, säubere und geräuschgedämpfte Raumgestaltung!

8. Konzipiere Dein System so, daß es erweiterungsfähig ist! Der Appetit des NSU kann mit dem Essen wachsen.

9. Last not least: Es ist die Ausnahme und nicht die Regel, daß neu eingeführte Systeme alle Nutzer befriedigen. Viele Unzulänglichkeiten zeigen sich erst im Betrieb. Auch können dabei berechtigte Zusatzforderungen entstehen. Plane genügend Zeit für Verbesserungen in der Erprobungsphase ein! Obwohl die NSU lernen müssen, auf eigenen Füßen zu stehen, lasse sie nicht zu früh im Stich!

5. Zur Motivierung der Nutzer

Interessiertheit und Kooperationsbereitschaft des NSU beeinflussen sehr stark, wie rasch und in welchem Maße die Möglichkeiten des PC genützt werden. Keinesfalls darf man als Regelfall ansehen, daß der NSU das neue Arbeitsmittel sehnlich erwartet. Eher rechnet er mit Schwierigkeiten, unbequemen Zusatzforderungen, einschneidenden Änderungen liebgewonnener und bewährter Arbeitsweisen, aber auch mit Ansprüchen, denen er sich nicht gewachsen fühlt. Hier rechtzeitig allen Konfrontationen die Spitze zu nehmen und Vorbehalte abzubauen, ist eine wichtige, leider oft vergessene Aufgabe des Systementwicklers, der beim PC meist auch Problemanalysator und Implementator ist.

Tafel 4 erinnert deshalb an Möglichkeiten, welche das kollegiale Zusammenarbeiten von Entwickler und Nutzer sowie das Akzeptanzverhalten fördern können. Ignoriert man sie, kann es zu erheblichen Hemmungen kommen.

6. Schlußbemerkungen

Natürlich ist es viel einfacher, eine umfangreiche Liste von Empfehlungen aufzustellen, als sie bei der Systementwicklung umzusetzen. Alle sind eigentlich so geartet, daß sie zusätzliche Arbeiten neben und bei der Programmentwicklung verlangen, wobei insgesamt ein Mehrfaches der Aufwendungen herauskommen kann, die für die reine Applikationslösung notwendig sind. Trotzdem erscheint es zweckmäßig, bereits die ersten Konzeptionen an Hand der Empfehlungen[1]) zu überprüfen und sie gegebenenfalls zu erweitern. Eine Nachbesserung nach der Praxiseinführung eines Systems ist in der Regel aufwendiger als die vorherige Realisierung, welche zudem noch Mißerfolge und Ärger ersparen kann. Daß sowohl die personellen und technischen Möglichkeiten als auch die Spezifik der Applikationsfälle zu Kompromissen zwingen können, bleibt unbestritten. Es schadet aber sicher nicht, wenn die Kompromisse nach allen Seiten hin durchdacht und weitgehend reduziert sind.

Schließlich soll man darauf hinarbeiten, daß mit jedem neuen Applikationsfall auch standardisierbare Methoden und Prozeduren entwickelt werden, die bei Folgefällen nachgenutzt werden können. So kann der Aufwand im Laufe der Zeit sinken. Dafür muß man weitsichtig konzipieren, systematisch dokumentieren und die Weiterverbreitung organisieren. Die Autoren werden es deshalb auch sehr begrüßen, wenn die Leser ihre Erfahrungen und Lösungen zur dargestellten Problematik mitteilen würden, um mitzuhelfen, diese wichtige Seite der Kleinrechentechnik weiterzuentwickeln. Solches anzuregen, ist erklärtes Ziel dieses Beitrags.

[1]) Sicher sind noch weitere Empfehlungen möglich, welche den Autoren entgangen sind.

Autoren:

Prof. Dr. sc. techn. Jürgen Meinhardt
Informatik-Zentrum des Hochschulwesens
an der Technischen Universität Dresden

Dr.-Ing. Holger Sasse
Humboldt Universität Berlin
Bereich Charitè

Simulation auf Mikrorechnern: Spiele und Experimente (Teil 2)

Simulationsmodelle von Bedienungssystemen

In Teil 1 (Heft 6) wurden Methoden zur Erzeugung und Transformation von Zufallszahlen behandelt. Sie bilden Werkzeuge zur Schaffung von Simulationsmodellen, in denen Prozesse der Realität auf Computern nachgebildet werden.
Simulationsmodelle werden entwickelt und benutzt,
- um die Planung und Projektierung komplexer Systeme zu unterstützen,
- um Steuerungsstrategien solcher Systeme zu bewerten und zu vergleichen und
- um Bediener komplizierter Anlagen zu trainieren.

Die Benutzung von Modellen ist billiger, weniger gefährlich und kann in anderem Zeitmaßstab erfolgen als Experimente oder Übungen am Original.
Man kann die auf Computern realisierten Simulationsmodelle in zwei Klassen untergliedern:
Kontinuierliche Modelle haben Prozesse zum Gegenstand, in denen alle auftretenden Variablen stetige Funktionen der Zeit sind. In ihnen gibt es keine sprunghaften Veränderungen. Die Differential- und Integralrechnung liefert die theoretische Basis für kontinuierliche Simulationsmodelle.
Diskrete Modelle haben Prozesse zum Gegenstand, die durch sprunghafte Änderungen der Werte von Variablen oder prozeßbeschreibenden Größen gekennzeichnet sind. Aus heutiger Sicht ist die Schaffung derartiger Modelle weniger mit mathematischen als mit programmierungstechnologischen Problemen verbunden.
Viele praktisch interessante Prozesse sind sowohl durch kontinuierliche als auch durch sprunghafte Wert- oder Zustandsänderungen charakterisiert. Für derartige Prozesse entwickelt und benutzt man kombinierte, also kontinuierlich-diskrete Modelle.
Der vorliegende Beitrag beschränkt sich auf diskrete Modelle. In der Praxis findet man breite Anwendungsfelder derartiger Modelle. Die Autoren haben zum Beispiel Modelle
- von leitbahngesteuerten Flurfördersystemen,
- von Fertigungs- und Produktionssteuerungssystemen,
- von Gurtbandförder- und Verteilsystemen,
- von Krantransport- und Steuerungssystemen und
- von informationsverarbeitenden Systemen

entwickelt und zur Lösung von Planungs-, Projektierungs- und Bewertungsaufgaben angewandt.

Unter den diskreten Modellen haben Bedienungsmodelle eine besondere Bedeutung. Bedienungsmodelle treten als Teilmodelle in vielen Komplexmodellen auf. Die Bedienungstheorie, eine Teildisziplin der Wahrscheinlichkeitstheorie, hat für viele Klassen von Bedienungssystemen mathematisch-analytische Lösungen entwickelt, die ohne Computerhilfe anwendbar sind. Dennoch sind einfache Bedienungsmodelle klassische Demonstrationsobjekte für die Entwicklung von Simulationsmodellen. Auch in diesem Beitrag sollen sie zur Erläuterung der für die Simulation diskreter Prozesse grundlegenden Algorithmen und Datenstrukturen dienen. Wie in Teil 1 wird BASIC als die auf Mikrocomputern gegenwärtig meistgenutzte Sprache angewandt, obwohl darauf hinzuweisen ist, daß andere Programmiersprachen besser für die Simulation geeignet sind.

1. Bedienungssysteme

Jeder hat sich schon einmal in eine Warteschlange eingereiht, z. B. um im Sommer eine gekühlte Limonade oder ein Eis zu kaufen. Betrachtet man den Verkauf von Eis an einem Kiosk als einen Bedienprozeß, so verkörpern die Kunden sogenannte *Forderungen*. Sie bilden eine *Warteschlange* vor dem Kiosk, der als *Bedienungseinrichtung* bezeichnet wird.
In komplizierten Bedienungssystemen durchlaufen die Forderungen ganze Netze von Bedienungseinrichtungen. Das gilt z. B. für die Bearbeitung von Reparaturaufträgen, die in einer Werkstatt eintreffen und die individuell verschiedene Folgen von Bearbeitungsstationen durchlaufen. Wenn neue Bedienungssysteme entworfen oder projektiert werden, benutzt man Modelle, mit denen man experimentiert. Die Resultate dieser Experimente beeinflussen Entscheidungen zur Gestaltung komplizierter Bedienungssysteme und ihrer Steuerung.
In einer Warteschlange sind die Forderungen eines Bedienungssystems nach einem bestimmten Aspekt geordnet. Am Kiosk stellen sich die Kunden entsprechend ihres Eintreffens »hinten« an die Schlange an. Ein »Vordrängeln« wird nicht gestattet. Als Ordnungsprinzip gilt: Wer zuerst kommt, mahlt zuerst, die sogenannte FIFO-Disziplin (First In/First Out). Aus dem Sportunterricht ist uns ein anderes Prinzip bekannt. Hier stellen sich die Schüler der Größe nach auf und beginnen in dieser Reihenfolge ihre Übungen am Turngerät. Die Elemente ordnen sich hier nach einem bestimmten Merkmal, der Körpergröße, wobei ein Hineindrängeln beim Aufstellen zulässig ist.
Eine weitere häufig verwendete Ordnungsregel ist das sog. LIFO-Prinzip (Last In – First Out). Es wird häufig in Lagern angewandt, wo zuletzt geliefertes zuerst wieder entnommen wird. Die *Bedienungsdisziplin* ist ein weiteres, Bedienungssysteme charakterisierendes Merkmal.
Um zu erklären, wie man Simulationsprogramme für Bedienungssysteme entwirft, soll im folgenden Abschnitt ein Beispiel eines einfachen Bedienungssystems in Verbindung mit einem entsprechenden BASIC-Programm vorgestellt werden. In Abschnitt 3. wird auf die Arbeit mit Listen eingegangen. Das ist die Voraussetzung dafür, ein allgemeines Simulationskonzept für Bedienungssysteme zu entwerfen, das den Gegenstand des Abschnittes 4. bildet. Im 5. Abschnitt werden Ergebnisdarstellung und -auswertung eines Simulationsexperimentes an einem Beispiel erläutert. Abschnitt 6. gibt Hinweise zum weiteren Ausbau des allgemeinen Konzeptes und zu seiner Nutzung.

2. Simulationsmodell eines einfachen Bedienungssystems

Für das zu modellierende System seien die folgenden Angaben bekannt:
Forderungen treffen in zufälligen, gleichmäßig im Intervall von 100 bis 300 Sekunden verteilten Zeitabständen ein. Die Bedienungsdauer ist gleichmäßig im Intervall von 100 bis 270 Sekunden verteilt. Die Bedienungsdisziplin ist FIFO. Gesucht sind Angaben über mittlere Wartezeit und mittlere Aufenthaltsdauer einer Forderung.

Programmbeispiel 1

```
10 !  SIMULATIONSMODELL EINES
20 !  EINFACHEN BEDIENUNGSSYSTEMS
30 !
40 !  A PAUSENZEIT ZWISCHEN FORDERUNGEN
50 !  B  BEDIENUNGSDAUER
60 !  T1 ANKUNFTSZEITPUNKT
70 !  T2 BEDIENUNGSBEGINNZEITPUNKT
80 !  T3 BEDIENUNGSENDZEITPUNKT
90 !  W  WARTEZEIT EINER FORDERUNG
100 ! SW SUMME DER WARTEZEITEN
110 ! D  AUFENTHALTSDAUER JE FORDERUNG
120 ! SD SUMME DER AUFENTHALTSZEITEN
140 ! NR LAUFENDE NR.EINER FORDERUNG
150 !
160 INPUT"ANZAHL DER FORDERUNGEN:";N
200 ! Z Y K L U S
210 !
220 A=100+200*RND(1)
230 T1=T1+A: NR=NR+1
240 IF T1<T3 THEN GOSUB300 ELSE GOSUB400
250 GOTO500
260 !
300 !BEDIENUNGSSTATION BESETZT
310 !
320 T2=T3: W=T3-T1: SW=SW+W: RETURN
330 !
400 !BEDIENUNGSSTATION FREI
410 !
420 T2=T1: RETURN
430 !
500 ! B E D I E N U N G
510 !
520 B=100+70*RND(1): SB=SB+B: T3=T2+B
530 D=T3-T1: SD=SD+D
540 IF NR<N THEN 200
550 !
600 ! A U S W E R T U N G
610 !
620 PRINT"MITTLERE WARTEZEIT:";SW/N
630 PRINT"MITTL.AUFENTHALTSZEIT:";SD/N
640 END
```

Dieses einfache, relativ leicht überschaubare Programm hat den Nachteil, daß es sich kaum verallgemeinern und z.B. nicht für Bedienungsnetze anwenden läßt, in denen das gegenseitige Überholen von Forderungen möglich ist. Aus diesem Grund soll in den folgenden Abschnitten das angekündigte, allgemein nutzbare Simulationskonzept entwickelt werden. Es stützt sich auf die Arbeit mit Listen, den Gegenstand des folgenden Abschnittes 3.

3. Arbeit mit Listen

Warteschlangen kann man mit Hilfe von Listen nachbilden. Unter einer Liste soll eine Folge von Elementen verstanden werden, die unter einem bestimmten Aspekt geordnet sind, d.h., es existiert eine Ordnungsregel. Zu dieser Liste können Elemente hinzugefügt werden, die sich dann entsprechend der Ordnungsregel einreihen, und es können Elemente entfernt werden.
Einige Programmiersprachen, wie z.B. PASCAL, bieten zur Nachbildung von Listen spezielle Sprachelemente an.
Im folgenden Programmbeispiel wird gezeigt, wie in BASIC Listen nachgebildet werden können. Da BASIC keine speziellen Datenstrukturen für Listen besitzt, werden zur Nachbildung Matrizen verwendet.
Die Liste soll maximal 10 Elemente aufnehmen können, und als Ordnungsregel gilt das FIFO-Prinzip. Jedes Element wird durch eine beliebige positive Nummer dargestellt.

Die verwendete Matrix besitzt zwei Spalten und 10 Zeilen.
Spalte 1 beinhaltet die Nummer eines Elementes und Spalte 2 einen sogenannten Verweis, Zeiger oder auch Pointer: die Zeilennummer des nächsten Elementes der Liste. Wird nur der Pointer für das nachfolgende Listenelement, den Nachfolger, gespeichert, so bezeichnet man diese Art der Verkettung als »Vorwärtsverkettung«. Speichert man zusätzlich noch den Pointer für den Vorgänger, so ergibt sich eine »Vorwärts- und Rückwärtsverkettung«.

	Spalte 1	Spalte 2
Zeile 1	10	3
Zeile 2	5	0
Zeile 3	6	2

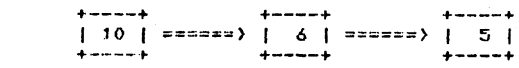


Bild 1
Vorwärtsverkettung von 3 Elementen

In Bild 1 ist ein Beispiel einer Vorwärtsverkettung von 3 Elementen einer Liste dargestellt. Zusätzlich zu den Pointern, die auf das nächste Element verweisen, wird noch ein Frontpointer (Kopfzeiger) benötigt. Dieser Frontpointer FP verweist auf das erste Element der Liste. Zur Verkürzung der Rechenzeit beim Verarbeiten von Listen empfiehlt es sich, auch einen Backpointer (Endezeiger) zu verwenden, der auf das letzte Element der Liste verweist. Sonst kann das Ende einer Liste ermittelt werden, indem die Liste Element für Element durchsucht wird. Das Ende ist erreicht, wenn der Pointer eines Elementes keinen Verweis auf ein weiteres Element besitzt. In diesem Fall hat der Pointer den Wert 0.
Eine leere Liste wird durch FP = 0 angezeigt.
Das Programmbeispiel 2 demonstriert die Arbeitsweise mit Listen in BASIC. Man kann zwischen den folgenden Programmfunktionen wählen:

- Einordnen
- Herausnehmen
- Listen
- Beenden

Programmbeispiel 2

```
10 REM Beispiel fuer Listenverarbeitung
20 REM KET(10,2) Matrix zur Verwaltung
30 REM Sp. 1  : Nr. des Elementes
40 REM Sp. 2  : Zeilennummer des naechsten Elementes
50 REM FP     : Frontpointer
60 REM BP     : Backpointer
70 REM Initialisierung
80 DIM KET(10,2)
90 FP=0: BP=0: FOR I=1 TO 10: KET(I,1)=-1: NEXT
100 REM Auswahl der Programmfunktionen
110 PRINT"Funktionen"
120 PRINT "Einordnen.................:1"
130 PRINT "Herausnehmen..............:2"
140 PRINT "Listen....................:3"
```

```
150 PRINT "Beenden.....:4"
160 INPUT "Welche Funktion ? .....:";F
170 ON (F) GOTO 190,290,350,520
180 PRINT "Unzulaessige Eingabe": GOTO 110
190 REM Einordnen eines Elementes nach FIFO
200 INPUT "Nr. des Elementes :";NR
210 REM Existiert schon ein Listenelement ?
220 IF FP=0 THEN FP=1: BP=1: I=1: GOTO 270
230 REM Suche einen freien Platz in KET
240 FOR I=1 TO 10: IF KET(I,1)<0 THEN GOTO 270
250 NEXT I: PRINT "Liste belegt": GOTO 110
260 REM Eintragen von Nummer und Zeiger
270 KET(I,1)=NR: KET(BP,2)=I: KET(I,2)=0: BP=I
280 GOTO 110
290 REM Entfernen des ersten Elementes
300 REM Ist die Liste leer ?
310 IF FP=0 THEN PRINT "Liste ist leer": GOTO 110
320 NR=KET(FP,1): PRINT "Entfernt wurde Nummer";NR
330 REM Frontpointer weitersetzen
340 I=FP: FP=KET(I,2): KET(I,1)=-1: GOTO 110
350 REM Ausgabe der Liste
360 REM Ist die Liste leer ?
370 IF FP=0 THEN PRINT"Liste ist leer": GOTO 110
380 PRINT TAB(6);"Reihenfolge"
390 PRINT TAB(11);"*"
400 PRINT TAB(2);"PHYSISCH * LOGISCH"
410 STE$="*":FOR I=1 TO 22: STE$=STE$+"*":NEXT I
420 PRINT STE$
430 I=1: J=FP
440 WE2=KET(J,1)
450 WE1=KET(I,1)
460 PRINT WE1;TAB(11);"*";TAB(17);WE2
470 IF I=10 THEN GOTO 500
480 I=I+1: NP=KET(J,2): IF NP=0 THEN WE2=-1: GOTO 450
490 J=NP: GOTO 440
500 PRINT "Frontpointer :";FP: PRINT "Backpointer :";BP
510 GOTO 110
520 REM Ende
530 END
```

In Bild 2 ist die physische Reihenfolge (Anordnung in der Matrix) und die logische Reihenfolge für eine Beispielkette dargestellt. Diese Darstellung wird mittels der Programmfunktion Listen erreicht.

```
          *
 PHYSISCH *  LOGISCH
**********************
     6    *     3
   - 1    *     4
     3    *     5
     4    *     6
     5    *   - 1
   - 1    *   - 1
   - 1    *   - 1
   - 1    *   - 1
   - 1    *   - 1
   - 1    *   - 1
```

Bild 2.
Anordnung von Elementen in einer Liste

Die in Bild 2 gezeigte Reihenfolge erhält man, indem man die Elemente 1 bis 5 in die Liste einordnet, die ersten beiden Elemente wieder herausnimmt und anschließend das Element mit der Nummer 6 einordnet.

Zeilen in der Matrix AKT, die noch nicht zur Abbildung von Aktivatoren verwendet wurden, werden durch einen Wert von —1 in der Spalte 1 markiert.

4. Allgemeines Simulationskonzept für Bedienungssysteme

4.1. Dynamische Modellelemente

In Anlehnung an SIMDIS, eine in der DDR verbreitete Simulationssprache, werden die Forderungen eines Bedienungssystems als Aktivatoren bezeichnet. Aktivatoren sind bewegliche Modellelemente. So verlassen sie zum Beispiel eine Warteschlange, werden von der Einrichtung bedient, verlassen diese, treten in eine neue Warteschlange ein und warten hier wieder auf eine Bedienung.
Für die Realisierung der Bewegung der Aktivatoren gibt es zwei Möglichkeiten.
Zum einen können die Aktivatoren »physisch« bewegt werden. Alle Informationen, die ein Aktivator besitzt, wie z.B. Nummer und Ereigniszeitpunkt, benötigen Speicherplatz und müssen bei jeder Bewegung des Aktivators mit transportiert werden. Günstiger ist es, die Aktivatoren mit ihren Informationen nur einmal abzuspeichern und die Bewegung nur »virtuell«, d.h. scheinbar, zu realisieren, Der Aktivator muß dafür zusätzliche Informationen tragen, die beinhalten,

- in welche Liste er eingetreten ist,
- auf welcher Position innerhalb der Liste er sich befindet.

Unter Verwendung von Frontpointern und Backpointern für jede Liste wird nur die Information »Wer ist der nächste Aktivator in der Liste« benötigt. Dieser Zusammenhang soll an dem folgenden Beispiel erläutert werden.
10 Aktivatoren sind auf 3 Listen LIS1, LIS2 und LIS3 verteilt. Die logische Reihenfolge der Aktivatoren in den Listen sei bekannt.

```
LIS1      1,3,7
LIS2      8,2,4
LIS3      5,6,9,10
```

Die Aktivatoren sind mit ihren Informationen in einer Matrix AKT von 10 Zeilen und 2 Spalten abgespeichert, wobei die Nummer des Aktivators einer Zeilennummer entspricht. Von den 2 Spalten wird nur die erste betrachtet. Sie enthält die Nummer des nächsten Aktivators in der betreffenden Liste. Die zweite Spalte

Matrix AKT

Zeilen	Spalten 1	2
1	3	
2	4	
3	7	
4	0	
5	6	
6	9	
7	0	
8	2	
9	10	
10	0	

enthält ein Merkmal des Aktivators, das für die Bewegung der Aktivators nicht benötigt wird. Zur Beschreibung der Reihenfolge in den Listen wird für jede Liste ein Kopfzeiger benötigt.
Der Kopfzeiger von LIS1, der ersten Liste, verweist auf den Aktivator mit der Nummer 1. Dieser Aktivator wird in der 1. Zeile der Matrix AKT beschrieben.
Aus Zeile 1 und Spalte 1 der Matrix AKT ergibt sich die nächste Aktivatornummer zu 3. In Zeile 3 wird auf den nächsten Aktivator mit der Nummer 7 verwiesen. Der Zeiger auf den folgenden Aktivator ist 0, d.h., hier ist das Ende der Liste LIS1 erreicht.
Für die anderen Listen ergibt sich eine analoge Vorgehensweise.

Frontpointer für die Listen:

```
LIS1 : 1
LIS2 : 8
LIS3 : 5
```

In dem hier vorzustellenden Simulationssystem werden alle Aktivatoren in einer Aktivatorenmatrix AKT abgebildet. In dieser Matrix sind die nötigen Informationen für einen Aktivator in einer Zeile abgelegt. Die Anzahl der Zeilen der Matrix wird bestimmt durch die maximale Anzahl der im Simulationsmodell zu verwendenden Aktivatoren. Alle Aktivatoren sind numeriert, wobei die Aktivatorennummer mit der Zeilennummer übereinstimmt, in welcher die Informationen über den Aktivator gespeichert sind.
Die Spalten der Matrix AKT sind wie folgt belegt:

- Spalte 1
  Nummer des nächsten Aktivators in der Liste

  In dieser Spalte wird der Zeiger gespeichert, der auf den nächsten Aktivator der Liste verweist.
- Spalte 2
  Merkmal eines Aktivators

  Jeder Aktivator kann eine Zusatzinformation tragen, die eine bessere Beschreibung des Bedienprozesses ermöglicht. Für das vorliegende System könnte z.B. vermerkt werden, wieviel Limonaden ein Kunde kaufen möchte.

Der Programmausschnitt 1 zeigt ein Unterprogramm zum Einordnen eines Aktivators in eine Liste nach dem FIFO-Prinzip und ein Unterprogramm zum Entfernen des ersten Aktivators aus einer Liste. Als Eingangsparameter dieses Unterprogramms dienen FP und NR.

Programmausschnitt 1

```
1000 REM Modul zum Einordnen in eine Liste nach FIFO
1010 REM FP=Frontpointer ; BP=Backpointer ; NR=Aktivatornummer
1020 IF FP=0 THEN FP=NR: AKT(NR,1)=0: BP=NR: RETURN
1030 AKT(BP,1)=NR: AKT(NR,1)=0: BP=NR: RETURN


1200 REM Modul zum Loesen des ersten Elementes aus einer Liste
1210 REM FP=Frontpointer ; BP=Backpointer ; NR=Aktivatornummer
1220 NR=FP: FP=AKT(NR,1): IF FP=0 THEN BP=0
1230 RETURN
```

4.2. Ereignisse

Leute, die am Kiosk vorbei gehen, werden erst dann zu Elementen des Bedienungssystems, wenn sie sich entschlossen haben, eine Limonade zu kaufen, und sich am Kiosk anstellen. Zur Nachbildung dieses Vorganges ist es erforderlich zu wissen, zu welchem Zeitpunkt sich ein Vorbeigehender entschließt, Kunde zu werden, d.h. zu welchem Zeitpunkt ein neues Element in das Bedienungssystem aufgenommen werden muß.
Dieser Vorgang soll als Ereignis vom Typ 1 bezeichnet werden. Innerhalb des Bedienungssystems treten noch weitere Ereignisse auf, die folgenden Typen zugeordnet werden:

Ereignis 1
Erzeugen eines Aktivators

Ereignis 2
Eintritt eines Aktivators in eine Warteschlange

Ereignis 3
Austritt eines Aktivators aus einer Warteschlange

Ereignis 4
Belegen einer Einrichtung durch einen Aktivator

Ereignis 5
Verlassen einer Einrichtung

Ereignis 6
Vernichten eines Aktivators

Bei der Nachbildung des Bedienprozesses kommt es nun darauf an, diese Ereignisse zeitlich in derselben Reihenfolge zu realisieren wie im realen Prozeß. Diese Methode wird auch als Ereignisfolgesimulation bezeichnet. Die Steuerungsfunktion wird innerhalb des Simulationsmodells einem Steuerteil übertragen.
Die Ereignisse werden in der Matrix EVEN gespeichert, wobei jedem Ereignis eine Zeile zugeordnet ist. Spalte 1 kennzeichnet, ob das Ereignis aktiv ist, d.h. zu einem bekannten Zeitpunkt realisiert werden soll. Der Ereigniszeitpunkt selbst wird in Spalte 2 abgelegt.
In Abhängigkeit von dem jeweiligen Ereignis und den damit verbundenen Zuständen in dem Bedienungssystem wird entschieden, ob weitere Ereignisse zu aktivieren sind. Diese differenzierte Vorgehensweise wird bei der Darstellung der erforderlichen Operationen gezeigt.

Ereignis 1 – Erzeugen eines Aktivators

Das Ereignis 1 veranlaßt das Erzeugen eines Aktivators, der sich dann durch das Simulationsmodell bewegt. Die Aktivatoren werden mit ihren Informationen in der Matrix AKT gespeichert. Bei der Realisierung des Ereignisses 1 laufen folgende Operationen ab:

1. *Suche freien Platz in der Matrix AKT*
Freie Plätze sind durch einen negativen Wert in der 1. Spalte markiert. Kann kein freier Platz gefunden werden, so wird der Simulationslauf vorzeitig beendet, und es wird eine entsprechende Mitteilung ausgegeben.

Es ist in diesem Fall zu prüfen, ob die Matrix AKT größer dimensioniert werden kann.

2. *Zuweisen von Informationen an den Aktivator*
In der Spalte 2 können dem Aktivator Zusatzinformationen zugewiesen werden. Für das vorliegende Beispiel wird hier eine bei 1 beginnende Zählnummer für Aktivatoren (XGES) verwendet.

3. *Planung der nächsten Ereignisse*
Das Ereignis 1 aktiviert Ereignisse zweier Typen. Zum einen soll sich der erzeugte Aktivator in die Warteschlange einordnen und zum anderen muß die Erzeugung des nächsten Aktivators vorbereitet werden.
Zur Bestimmung des Zeitpunktes, zu dem der nächste Aktivator erzeugt werden soll, wird die Zwischenerzeugungszeit verwendet. Sie gibt die Zeit an, die zwischen der Erzeugung zweier Aktivatoren vergehen soll. Wenn ein reales System nachgebildet werden soll, benötigt man dazu Informationen, die durch statistische Messungen am realen System oder an ähnlichen Systemen auszuführen sind. Für das vorliegende Beispiel ist als Zwischenerzeugungszeit eine gleichverteilte Zufallsgröße im Intervall [OAG – OBG, OAG + OBG] angesetzt worden. Dabei sind OAG und OBG Operanden, die in der Initialisierungsphase eingegeben werden. OAG ist der Mittelwert, der Operand OBG stellt die maximale Abweichung von diesem Mittelwert dar. Das nächste Ereignis vom Typ 1 wird somit aktiviert, und der Ereigniszeitpunkt ergibt sich aus der Zwischenerzeugungszeit und der aktuellen Zeit TIME.
Der durch Ereignis 1 erzeugte Aktivator soll ohne Zeitverlust in die Warteschlange eingeordnet werden. Das Ereignis 2 (Eintritt in eine Warteschlange) muß zum selben Zeitpunkt realisiert werden. Aus diesem Grund wird bei der Aktivierung des Ereignisses 2 als Ereigniszeitpunkt die aktuelle Zeit eingetragen.

Der Programmausschnitt 2 zeigt die erforderlichen Anweisungen.

Programmausschnitt 2

```
300 REM Realisiere Ereignis 1 - Erzeugen eines Aktivators
310 FOR I=1 TO XMAX: IF AKT(I,1)<0 THEN XNR=I: GOTO 325
320 NEXT I: PRINT "FEHLER 1": GOTO 3000
325 XGES=XGES + 1: AKT(XNR,2)=XGES
330 REM Plane naechste Ereignisse
340 ZWZ=OAG - OBG + 2*RND*OBG: EVEN(1,1)=1: EVEN(1,2)=TIME + ZWZ
350 EVEN(2,1)=1: EVEN(2,2)=TIME: ENR=2: GOTO 250
```

Ereignis 6 – Vernichten eines Aktivators

Ereignis 6 bewirkt das Vernichten eines Aktivators, d.h., eine Forderung verläßt das Bedienungssystem. Bei der Realisierung von Ereignis 6 werden folgende Operationen ausgeführt:

1. *Freigeben des Speicherplatzes in der Matrix AKT*
In der Spalte 1 der Matrix AKT wird an der betreffenden Zeile XNR ein negativer Wert eingetragen.

2. *Zählung der vernichteten Aktivatoren*
Jeder vernichtete Aktivator wird in der Größe XTER gezählt. In Abhängigkeit von der Anzahl der vernichteten Aktivatoren kann der Abbruch der Simulation erfolgen.

Im Programmausschnitt 3 sind die erforderlichen Anweisungen aufgelistet.

Programmausschnitt 3

```
800 REM Realisiere Ereignis 6 - Vernichten eines Aktivators
810 AKT(XNR,1)=-1: PRINT AKT(XNR,2);"vernichtet"
820 EVEN(6,1)=0: XTER=XTER + 1: IF XTER=ANZ THEN GOTO 3000
830 GOTO 200
```

4.3. Warteschlangen

Verwaltung von Warteschlangen

Ein Ergebnis unserer Untersuchung sollen Aussagen über die Größe des Stauraumes sein, in dem die Kunden auf ihre Bedienung warten können. Es ist daher notwendig, die Kenngrößen der sich eventuell bildenden Warteschlange statistisch zu erfassen.
In der Statistik für eine Warteschlange werden erfaßt:

Statistische Groessen	in Matix QUE
Nummer der Warteschlange	Zeilennummer
Frontpointer fuer die Warteschlange	Spalte 1
Backpointer fuer die Warteschlange	Spalte 2
Anzahl der Eintritte	Spalte 3
Maximale Laenge der Warteschlange	Spalte 4
Aktuelle Laenge der Warteschlange	Spalte 5
Letzter Ereigniszeitpunkt	Spalte 6
Letzter Wert des Zeitintegrales	Spalte 7

Zur Verwaltung der Warteschlangen mit ihren Informationen wird eine Matrix QUE verwendet. Die Zeilennummer entspricht der laufenden Nummer der Warteschlange im Simulationssystem. Die Spalten enthalten die oben aufgezeigten Größen. Die Anzahl der Zeilen ergibt sich aus der Anzahl der im Simulationsmodell verwendeten Anzahl von Warteschlangen.
Die Spalten 6 und 7 müssen im Zusammenhang betrachtet werden. Sie dienen zur Berechnung der mittleren Verweilzeit eines Aktivators in der Warteschlange und der mittleren Warteschlangenlänge. In Bild 3 ist der zeitliche Verlauf der aktuellen Länge einer Warteschlange dargestellt.
Die abgebildete Fläche wird Zeitintegral genannt. Zur Berechnung des Gesamtflächeninhaltes wird die Gesamtfläche in mehrere Teilflächen eingeteilt, deren Höhe sich aus der aktuellen Länge der Warteschlange und deren Breite sich aus der Zeitdauer ergibt, in der die aktuelle Länge konstant bleibt.
Für die Fläche A_i gilt:

$$A_i = (t_2 - t_1) \cdot l.$$

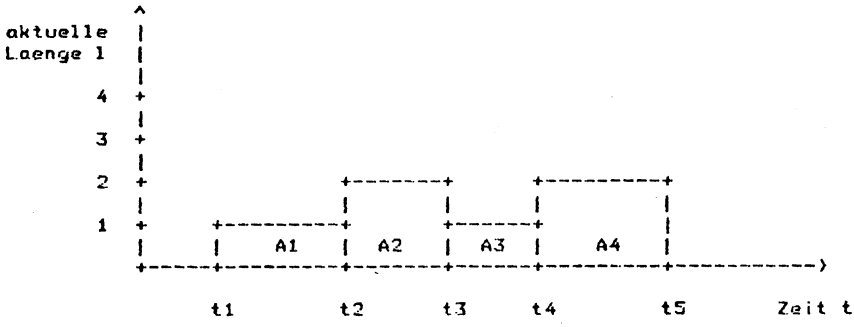


Bild 3. Zeitlicher Verlauf einer Warteschlange

Eine beliebige Fläche A_i berechnet sich wie folgt:

$A_i = (t_{i+1} - t_i) \cdot l\,(t_i)$.

Allgemein läßt sich der Wert des Zeitintegrales Z zum Zeitpunkt t_r wie folgt ermitteln:

$$Z\,(t_r) = \sum_{i=1}^{r-1} (t_{i+1} - t_i) \cdot l\,(t_i). \qquad (1)$$

Folgende statistische Kenngrößen lassen sich für eine Warteschlange zum Zeitpunkt t ableiten:

- mittlere Warteschlangenlänge
  Division des Wertes des Zeitintegrals durch den Wert des aktuellen Zeitpunktes
- mittlere Wartezeit eines Aktivators in der Warteschlange
  Division des Wertes des Zeitintegrales durch die Anzahl der eingetretenen Aktivatoren

Ereignis 2 – Eintritt in eine Warteschlange

Das Ereignis 2 bewirkt den Eintritt eines Aktivators in eine Warteschlange und die damit verbundene Berechnung der statistischen Kenngrößen. Bei der Realisierung werden folgende Operationen ausgeführt:

1. *Einordnen in eine Warteschlange*

Der Aktivator wird nach dem FIFO-Prinzip in die Warteschlange eingeordnet. Die erforderlichen Angaben werden der Matrix QUE entnommen.

2. *Berechnung der statistischen Kenngrößen*

- Ermittlung der Dauer des letzten Zustandes, d.h. der Differenz aus dem aktuellen Zeitpunkt TIME und dem Zeitpunkt der letzten Zustandsänderung QUE (1,6).
- Das Produkt aus der Dauer und der augenblicklichen Warteschlangenlänge wird zum Wert des Zeitintegrals addiert.
- Der aktuelle Ereigniszeitpunkt wird vermerkt.
- Die Anzahl der eingetretenen Aktivatoren wird um 1 erhöht.
- Die aktuelle Warteschlangenlänge wird um 1 erhöht und die maximale Warteschlangenlänge bestimmt.

3. *Planung der nächsten Ereignisse*

Wenn die hinter der Warteschlange befindliche Einrichtung frei ist, kann der eingetretene Aktivator die Warteschlange sofort wieder verlassen und die Einrichtung belegen. In Abhängigkeit vom Belegungszustand der Einrichtung wird das Ereignis 3 (Austritt aus einer Warteschlange) aktiviert. Als Ereigniszeitpunkt gilt der aktuelle Zeitpunkt TIME.
Kann die Einrichtung nicht belegt werden, so muß das nächste Ereignis gewählt werden. Die Auswahl trifft die später zu beschreibende Ablaufsteuerung.
Der Programmausschnitt 4 zeigt die erforderlichen Anweisungen.

Programmausschnitt 4

```
400 REM Realisierung Ereignis 2 - Eintritt in eine Warteschlange
405 FP=QUE(1,1): BP=QUE(1,2): NR=XNR:GOSUB 1000
410 QUE(1,1)=FP: QUE(1,2)=BP
415 REM Statistik
420 DIF=TIME - QUE(1,6): QUE(1,7)=QUE(1,7) + DIF*QUE(1,5)
425 QUE(1,6)=TIME: QUE(1,3)=QUE(1,3) + 1: QUE(1,5)=QUE(1,5) + 1
430 IF QUE(1,4)<QUE(1,5) THEN QUE(1,4)=QUE(1,5)
435 REM Plane naechste Ereignisse
440 EVEN(2,1)=0: IF FAC(1,1)>0 THEN GOTO 200
445 EVEN(3,1)=1: EVEN(3,2)=TIME: ENR=3: GOTO 250
```

Ereignis 3 – Austritt aus einer Warteschlange

Das Ereignis 3 realisiert den Austritt eines Aktivators aus einer Warteschlange und veranlaßt die Berechnung der notwendigen statistischen Kenngrößen. Bei der Abarbeitung des Ereignisses 3 werden folgende Operationen ausgeführt:

1. *Austritt aus einer Warteschlange*

Der erste Aktivator aus der Warteschlange wird entfernt und der Frontpointer auf den Folgeaktivator übertragen. Die erforderlichen Angaben zur Verwaltung der Liste werden der Matrix QUE entnommen.

2. *Berechnung der statistischen Kenngrößen*

- Ermittlung der Dauer des letzten Zustandes, d. h. der Differenz aus dem aktuellen Zeitpunkt und dem Zeitpunkt der letzten Zustandsänderung.
- Das Produkt aus der Dauer und der augenblicklichen Warteschlangenlänge wird zum Wert des Zeitintegrals addiert.
- Der aktuelle Ereigniszeitpunkt wird vermerkt.
- Die aktuelle Warteschlangenlänge wird um 1 vermindert.

3. *Planung der nächsten Ereignisse*

Ein Aktivator kann eine Warteschlange nur verlassen, wenn die nachfolgende Einrichtung nicht belegt ist. Folglich muß sich als nächstes Ereignis die Belegung der Einrichtung zum aktuellen Zeitpunkt anschließen. Das Ereignis 4 wird aktiviert.

Die erforderlichen Anweisungen sind im Programmausschnitt 5 aufgezeigt.

Programmausschnitt 5

```
500 REM Realisierung Ereignis 3 - Austritt aus einer Warteschlange
510 FP=QUE(1,1): BP=QUE(1,2): GOSUB 1200
520 XNR=NR: QUE(1,1)=FP: QUE(1,2)=BP
530 REM Statistik
540 DIF=TIME - QUE(1,6): QUE(1,7)=QUE(1,7) + DIF*QUE(1,5)
550 QUE(1,6)=TIME: QUE(1,5)=QUE(1,5)-1
560 REM Plane naechste Ereignisse
570 EVEN(3,1)=0: EVEN(4,1)=1: EVEN(4,2)=TIME: ENR=4: GOTO 250
```

4.4. Einrichtungen

Verwaltung von Einrichtungen

Einrichtungen sind Elemente in einem Bediensystem, die zu jedem Zeitpunkt nur von einem Aktivator belegt sein können.
Die erforderlichen Informationen zur Verwaltung der Einrichtungen sind in der Matrix FAC gespeichert. Aus der folgenden Übersicht ergibt sich die Struktur der Matrix:

Informationen ueber Einrichtungen	*	in Matrix FAC
Nummer der Einrichtung	*	Zeilennummer
Nummer des belegenden Aktivators	*	Spalte 1
Anzahl der Eintritte	*	Spalte 2
Letzter Ereigniszeitpunkt	*	Spalte 3
Letzter Wert des Zeitintegrals	*	Spalte 4

Die Anzahl der Zeilen ergibt sich aus der im Simulationsmodell verwendeten maximalen Anzahl von Einrichtungen. Das Zeitintegral wird nach Formel 1 berechnet, wobei der Faktor l zwei Werte annehmen kann, 0 für nicht belegt und 1 für belegt.
Die mittlere Auslastung ergibt sich aus dem Wert des Zeitintegrals, dividiert durch die aktuelle Zeit. Aus dem Quotienten von dem Wert des Zeitintegrals und der Anzahl der Eintritte berechnet sich die mittlere Verweilzeit eines Aktivators in der Einrichtung.

Ereignis 4 – Belegen einer Einrichtung

Das Ereignis 4 bewirkt das Belegen einer Einrichtung durch einen Aktivator und die erforderliche Berechnung der statistischen Kenngrößen. Bei der Realisierung werden folgende Operationen ausgeführt:

1. *Belegen der Einrichtung*
In die Spalte 1 der Matrix FAC wird die Nummer des belegenden Aktivators eingetragen

2. *Berechnung der statistischen Kenngrößen*
- Der aktuelle Ereigniszeitpunkt wird vermerkt
- Die Anzahl der eingetretenen Aktivatoren wird um 1 erhöht

3. *Planung der nächsten Ereignisse*
Der Kunde wird von dem Verkäufer bedient und verläßt dann den Kiosk. Diese Bedienzeit wird auch als Verweilzeit des Aktivators in der Einrichtung nachge-

bildet. Für das vorliegende Beispiel wird die Verweilzeit als gleichverteilte Zufallsgröße im Intervall [OAS — OBS, OAS + OBS] betrachtet. Die beiden Operanden OAS und OBS werden in der Initialisierungsphase eingegeben.
Der Zeitpunkt, zu dem der Aktivator die Einrichtung wieder verläßt, ergibt sich aus dem aktuellen Zeitpunkt TIME und der berechneten Verweilzeit. Als nächstes Ereignis wird das Ereignis 5 (Verlassen einer Einrichtung) aktiviert.

Der Programmausschnitt 6 zeigt die notwendigen Anweisungen.

Programmausschnitt 6

```
600 REM Realisierung Ereignis 4 - Eintritt in eine Einrichtung
610 FAC(1,1)=XNR: FAC(1,2)=FAC(1,2) + 1: FAC(1,3)=TIME
620 REM Plane Folgeereignisse
630 EVEN(4,1)=0: ZWZ=OAS - OBS + 2*RND*OBS
640 EVEN(5,1)=1: EVEN(5,2)=TIME + ZWZ: GOTO 200
```

Ereignis 5 – Verlassen einer Einrichtung

Das Ereignis 5 veranlaßt das Verlassen einer Einrichtung durch einen Aktivator und die Berechnung der statistischen Kenngrößen für die Einrichtung. Bei der Realisierung laufen folgende Operationen ab:

1. *Freigeben der Einrichtung*
In die Spalte 1 der Matrix FAC wird der Wert 0 eingetragen.

2. *Berechnung der statistischen Kenngrößen*
- Ermittlung der Dauer des letzten Zustandes
- Berechnung des aktuellen Wertes des Zeitintegrals
- Der aktuelle Ereigniszeitpunkt wird vermerkt.

3. *Planung der nächsten Ereignisse*
Hat ein Aktivator eine Einrichtung verlassen, so kann sie von einem anderen Aktivator wieder belegt werden. Bedingung dafür ist, daß sich Aktivatoren in der entsprechenden Warteschlange befinden. Das Ereignis 3 (Austritt aus einer Warteschlange) wird nur dann aktiviert, wenn sich in der betreffenden Warteschlange Aktivatoren befinden. Der Ereigniszeitpunkt für dieses Ereignis ist die aktuelle Zeit TIME!
Für das vorliegende Bedienungssystem wird angenommen, daß der Aktivator nach dem Verlassen der Einrichtung vernichtet werden soll. Es wird somit das Ereignis 6 aktiviert.

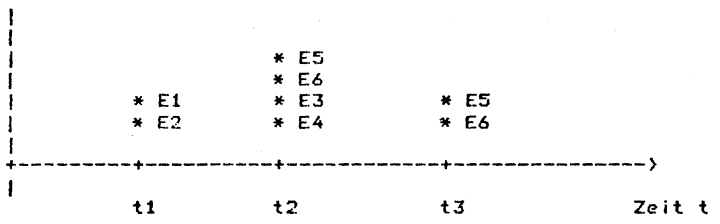
Der Programmausschnitt 7 zeigt die notwendigen Anweisungen.

Programmausschnitt 7

```
700 REM Realisierung Ereignis 5 - Austritt aus einer Einrichtung
710 XNR=FAC(1,1)
720 REM Statistik
730 DIF=TIME - FAC(1,3): FAC(1,4)=FAC(1,4) + DIF
740 FAC(1,1)=0: FAC(1,3)=TIME
750 REM Plane naechste Ereignisse
760 IF QUE(1,5)>0 THEN EVEN(3,1)=1: EVEN(3,2)=TIME
770 EVEN(5,1)=0: EVEN(6,1)=1: EVEN(6,2)=TIME: ENR=6: GOTO 250
```

4.5. Steuerung des Ablaufs

Aufgabe der Steuerung ist es, die Realisierung der möglichen Ereignisse in der richtigen Reihenfolge zu veranlassen. Die Ereignisse müssen in ihrer zeitlichen Reihenfolge realisiert werden. Man kann sich diese Folge wie eine Ereigniskette auf dem Zeitstrahl vorstellen.



Zum Zeitpunkt t_1 wird ein Aktivator erzeugt (Ereignis 1), und er reiht sich zum gleichen Zeitpunkt in die Warteschlange ein (Ereignis 2). Die Einrichtung kann nicht belegt werden, da sie durch einen anderen Aktivator blockiert ist.
Zum Zeitpunkt t_2 verläßt ein Aktivator die Einrichtung (Ereignis 5) und wird zum gleichen Zeitpunkt vernichtet (Ereignis 6). Da nun die Einrichtung frei ist, kann ein Aktivator aus der Warteschlange entfernt werden (Ereignis 3) und die Einrichtung belegen (Ereignis 4).
Es tritt der Fall ein, daß zu einem Zeitpunkt mehrere Ereignisse realisiert werden müssen. Obwohl die Ereignisse zum gleichen Zeitpunkt stattfinden, muß bei der Realisierung eine bestimmte Reihenfolge eingehalten werden. So darf zum Ereigniszeitpunkt t_1 das Ereignis 2 nicht vor dem Ereignis 1 realisiert werden.
Durch ein Ereignis wird die Bewegung eines Aktivators veranlaßt. Kann ein neues Ereignis, bezogen auf diesen Aktivator, zum gleichen Zeitpunkt realisiert werden, so wird dieses Ereignis als Folgeereignis ausgeführt. Diese Abarbeitung wird so lange vollzogen, bis dieser Aktivator zum aktuellen Zeitpunkt nicht mehr bewegt werden kann. Zum Ereigniszeitpunkt t_1 kann der Aktivator die Warteschlange betreten, er kann sie aber nicht verlassen, da die Einrichtung belegt ist und der Aktivator auf einen späteren Zeitpunkt warten muß, zu dem die Einrichtung frei wird.
Zum Zeitpunkt t_2 verläßt ein Aktivator die Einrichtung (Ereignis 5) und veranlaßt während der Realisierung dieses Ereignisses eine Kennzeichnung in der Spalte 1 der Matrix EVEN, daß zu diesem Zeitpunkt auch das Ereignis 3 realisiert werden soll. Für den Aktivator, der die Einrichtung verlassen hat, existiert das Folgeereignis 6, welches im Anschluß an das Ereignis 5 ausgeführt werden muß. Dieses Folgeereignis wird zum aktuellen Ereignis und wird unmittelbar ausgeführt. Nach der Realisierung von Ereignis 6 existiert kein Folgeereignis, und es muß geprüft werden, ob zum aktuellen Zeitpunkt t_2 weitere Ereignisse zu realisieren sind. Durch das Ereignis 5 wurde das Ereignis 3 als realisierbar gekennzeichnet, welches nun ausgeführt wird. Aus dem Ereignis 3 ergibt sich das Folgeereignis 4. Da keine weiteren Ereignisse mehr zum Zeitpunkt t_2 realisiert werden müssen, muß das nächste Ereignis mit dem nächstfolgenden Ereigniszeitpunkt ausgewählt werden. Hier ist es das Ereignis 5 zum Zeitpunkt t_3.
Der Algorithmus für die Steuerung wird als PAP in Bild 4 dargestellt.

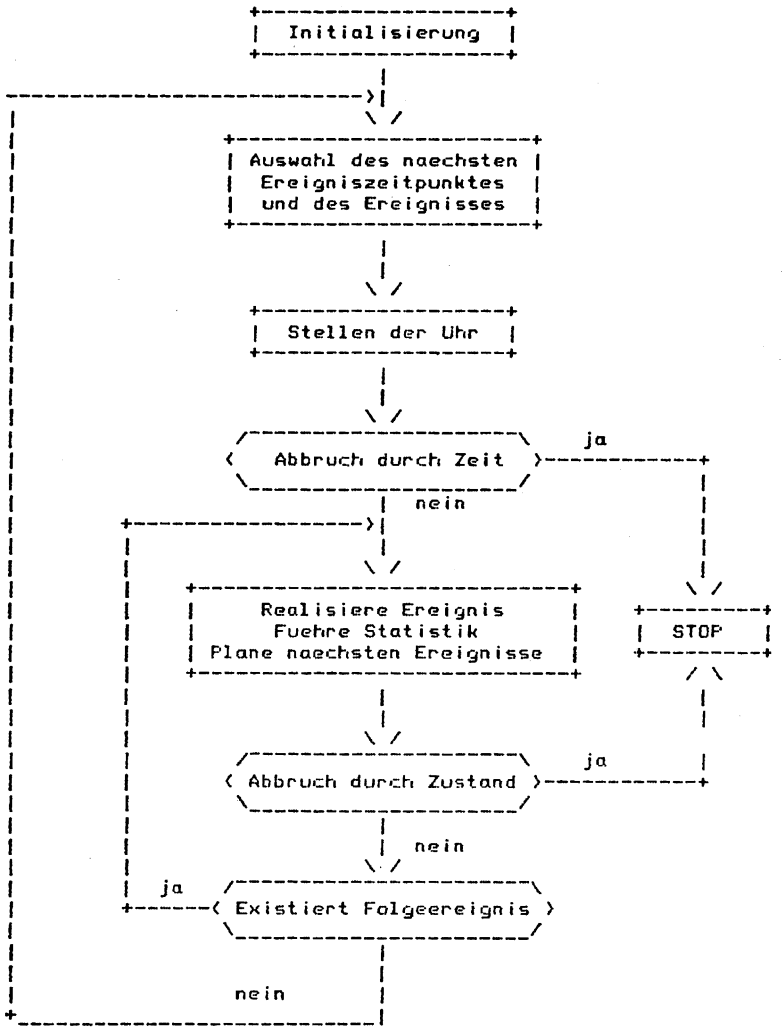


Bild 4. Steuerung des Ablaufes

Bei der Umsetzung in die BASIC-Anweisungen wurden folgende Variablen verwendet:

• aktueller Ereigniszeitpunkt	TIME
• aktuelle Ereignismummer	ENR
• Ereigniszeitpunkt des i-ten Ereignisses	EVEN (i, 2)
• Zeitpunkt des Abbruches der Simulation	ZP

Der Programmausschnitt 8 zeigt die notwendigen BASIC-Anweisungen zur Auswahl des nächsten Ereignisses.

Programmausschnitt 8

```
200 REM Auswahl eines Ereignisses
210 MIN=32000: FOR I=1 TO 6
220 IF EVEN(I,1)>0 AND EVEN(I,2)<MIN THEN MIN=EVEN(I,2): ENR=I
230 NEXT I: TIME=EVEN(ENR,2): IF TIME>ZP THEN GOTO 3000
240 REM Realisierung in Abhaengigkeit vom Ereignistyp
250 ON (ENR) GOTO 300,400,500,600,700,800
```

4.6. Initialisierungsphase des Simulationssystems

Die Initialisierungsphase hat die Aufgaben:

- die Matrizen in Abhängigkeit von der Größe und den Aufgaben des Bediensystems zu dimensionieren,
- Anfangswerte zu setzen und
- Werte für die Operanden und Abbruchbedingungen einzulesen.

Das betrachtete Simulationssystem verwendet die Matrizen AKT, QUE, FAC und EVEN mit fest vorgegebenen Spaltenanzahlen. Die Anzahl der Zeilen muß den gegebenen Bedingungen angepaßt werden. Für die Matrix AKT wird die Anzahl der Zeilen vom Nutzer eingegeben und richtet sich nach der Anzahl der Aktivatoren im System.
Die Anzahl der Zeilen für die Matrizen QUE und FAC wurden mit 1 festgelegt, da nur jeweils eine Warteschlange und eine Einrichtung verwendet werden. Für die Matrix EVEN ergeben sich 6 Zeilen, da in dem System 6 verschiedene Ereignistypen definiert wurden. Den einzelnen Elementen der Matrizen werden die entsprechenden Anfangswerte zugewiesen.
Der Nutzer muß den Startzeitpunkt für das Simulationssystem eingeben. Dieser Zeitpunkt bildet den ersten Ereigniszeitpunkt für das Ereignis 1, welches aktiviert wird.
Zur Steuerung des Abbruchs der Simulation kann der Nutzer zwischen dem Abbruch nach einem eingegebenen Zeitpunkt oder dem Abbruch nach der eingegebenen Anzahl der vernichteten Aktivatoren wählen.
Der Programmausschnitt 9 zeigt die erforderlichen BASIC-Anweisungen.

Programmausschnitt 9

```
10 REM Einfaches Bediensystem
20 INPUT "Anzahl der Aktivatoren im System :";XMAX
30 DIM AKT(XMAX,2): REM Aktivatorenmatrix
40 DIM QUE(1,7)   : REM Verwaltung von Warteschlangen
50 DIM FAC(1,4)   : REM Verwaltung von Einrichtungen
55 DIM EVEN(6,2)  : REM Verwaltung von Ereignissen
60 REM Initialisierung
70 FOR I=1 TO XMAX: AKT(I,1)=-1: NEXT I
80 FOR I=1 TO 7: QUE(1,I)=0: NEXT I
90 FOR I=1 TO 4: FAC(1,I)=0: NEXT I
100 FOR I=2 TO 6: EVEN(I,1)=0: NEXT I
110 INPUT "Startzeitpunkt :";TIME
120 EVEN(1,1)=1: EVEN(1,2)=TIME
130 XGES=0: XTER=0: ZP=32000: ANZ=32000
140 REM Operanden fuer Ankunft- und Bedienzeiten
150 PRINT "Ankunftszeit": INPUT"OAG :";OAG: INPUT"OBG :";OBG
160 PRINT "Bediendauer": INPUT"OAS :";OAS: INPUT"OBS :";OBS
170 REM Abbruch der Simulation
```

```
175 INPUT "Abbruch ueber Zeit (Z) oder Anzahl (A)";F$
180 IF F$="z" OR F$="Z" THEN INPUT"Zeitpunkt";ZP:GOTO 200
185 IF F$="A" OR F$="a" THEN INPUT"Anzahl";ANZ:GOTO 200
190 GOTO 175
```

4.7. Ergebnisdarstellung und -auswertung

Welche Ergebnisse und in welcher Form, d.h. auf Drucker oder Terminal, die Ergebnisse aufbereitet werden, das soll dem Nutzer selbst überlassen bleiben. Die erforderlichen Angaben zur Berechnung der statistischen Kenngrößen sind im Kapitel 4. gegeben worden. An Abschnitt 5. sind Beispiele für die Gestaltung von Drucklisten aufgezeigt.
Bei der Programmierung der Ergebnisausgabe sollte beachtet werden, daß sie mit der Zeilennummer 3000 beginnt, da beim Abbruch der Simulation zu dieser Zeilennummer gesprungen wird.

5. Beispiel eines Simulationsexperimentes

Verbindet man die angegebenen Programmausschnitte miteinander und ergänzt diese um eine Ergebnisaufbereitung, so entsteht ein BASIC-Programm zur Simulation einfacher Bedienungssysteme. Mit diesem Simulationsmodell können Experimente auf dem Computer vollzogen werden.
Folgende Eingabeparameter charakterisieren das Simulationsexperiment:

- Zwischenankunftszeit der Forderungen

Die Zwischenankunftszeit ist gleichverteilt im Intervall von 0 bis 80 Zeiteinheiten,

- Bediendauer

Die Bediendauer ist gleichmäßig im Intervall von 15 bis 55 Zeiteinheiten verteilt.

- Abbruch der Simulation

Die Simulation soll nach 100 bedienten Forderungen beendet werden.

Für das Simulationsexperiment wurde folgende Ausgabe programmiert.

Eingabeparameter

```
Ankunftszeit        OAG :  40              OBG : 40
Bedienzeit          OAS :  35              OBS : 20
Abbruch der Simulation
            ZP  : 32000          ANZ : 100
```

Resultate

Zeitpunkt: 3998

Die Simulation wurde nach 3998 Zeiteinheiten beendet.

```
AKTIVATORENMATRIX   AKT

AKTIVATOR-      NAECHSTER       MERKMAL 1
NUMMER          AKTIVATOR

     1              -1              100
     2               3              102
     3               4              103
     4               0              104
     5               2              101
     6              -1               33
     7              -1               34
     8              -1                0
     9              -1                0
    10              -1                0
```

Zum Zeitpunkt des Abbruches der Simulation befinden sich 4 Aktivatoren im System.

```
EREIGNISMATRIX  EVEN

EREIGNIS        AKTIV          EREIGNIS-
TYP                            ZEITPUNKT

   1              1             4031
   2              0             3969
   3              1             3998
   4              0             3956
   5              0             3998
   6              0             3998
```

Entsprechend der Ablaufsteuerung könnte zum Zeitpunkt 3998 das Ereignis 3 (Austritt aus einer Warteschlange) realisiert werden. Dieses Ereignis wird nicht mehr realisiert, da das Abbruchkriterium schon erreicht wurde.

```
WARTESCHLANGE VOR DER EINRICHTUNG

5   2   3   4
```

Vor der Einrichtung warten die Aktivatoren in der aufgezeigten Reihenfolge. Diese Reihenfolge ist auch aus der Matrix AKT abzulesen.

```
STATISTISCHE ANGABEN ZUR EINRICHTUNG

      BELEGENDER AKTIVATOR            : 0
      ANZAHL DER EINTRITTE            : 100
      MITTLERE VERWEILZEIT EINES
      AKTIVATORS IN DER EINRICHTUNG : 33.71
      MITTLERE AUSLASTUNG             : 0.84
```

In diesem Experiment wurde für die Bedieneinrichtung eine mittlere Auslastung von 84 % ermittelt.

```
STATISTISCHE ANGABEN ZUR WARTESCHLANGE

      AKTUELLE LAENGE DER WARTESCHLANGE   : 4
      MAXIMALE LAENGE DER WARTESCHLANGE   : 6
      MITTLERE LAENGE DER WARTESCHLANGE   : 1.16
      ANZAHL DER EINTRITTE                : 104
      MITTLERE VERWEILZEIT EINES AKTIV.   : 44.5
```

6. Hinweise zur Nutzung und allgemeinen Verwendbarkeit

In dem behandelten Beispiel besteht das Bedienungssystem aus einer Bedieneinrichtung, in der die ankommenden Forderungen bedient werden. Treffen die Forderungen auf eine belegte Einrichtung, so werden sie in eine Warteschlange eingereiht. Einschließlich dem Erzeugen und Vernichten der Forderungen werden zur Modellierung dieses Prozesses 6 Ereignistypen verwendet.
Betrachtet man Bedienungssysteme, die mehrere Einrichtungen und Warteschlangen aufweisen, so wird sich die Anzahl der Ereignistypen erhöhen. Als Beispiel für ein derartiges Bedienungssystem kann eine Kaufhalle herangezogen werden.

Die Kaufhalle bestehe aus einem Fleischstand, einer Flaschenrücknahme und zwei Kassen als Einrichtungen, vor denen sich Warteschlangen bilden können. Zusätzlich befinden sich noch 3 Regalreihen in der Kaufhalle, an denen der Kunde sich selbst bedienen kann. Jeder Kunde besitzt einen Einkaufszettel, auf dem vermerkt ist, in welcher Reihenfolge er die verschiedenen Einrichtungen anläuft und welche Aktivitäten dort durchzuführen sind. Die Kaufhalle besitzt einen Eingang und einen Ausgang, durch den das Erzeugen bzw. das Vernichten der Forderungen nachgebildet wird, und es sind ausreichend Körbe für alle Kunden vorhanden. Der Einkaufszettel eines Kunden ist in den Merkmalen des Aktivators nachzubilden. Bild 5 zeigt die Kaufhalle mit den entsprechenden Einrichtungen.

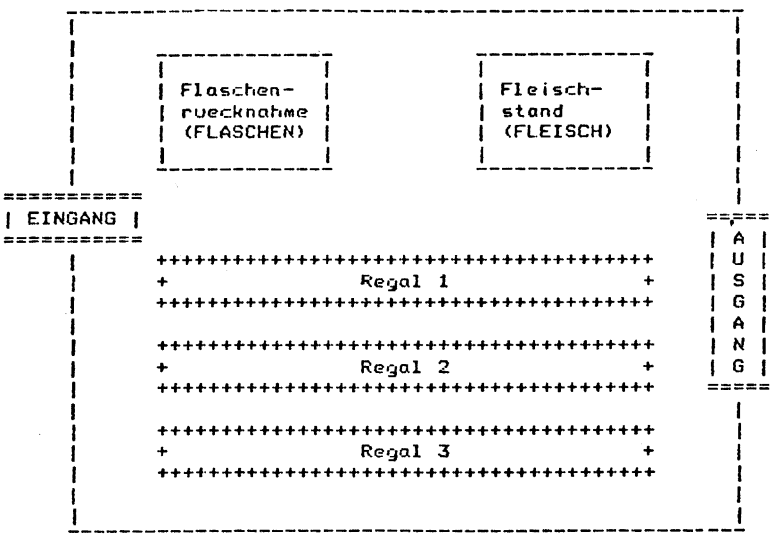


Bild 5. Bedienungssystem Kaufhalle

Zur Modellierung dieses Bedienungssystems sind folgende Ereignistypen zu berücksichtigen:

Typ 1: Erzeugung eines Aktivators und Festlegung seiner Merkmale
Typ 2: Eintritt in die Warteschlange FLASCHEN
Typ 3: Austritt aus der Warteschlange FLASCHEN
Typ 4: Belegung der Einrichtung FLASCHEN
Typ 5: Verlassen der Einrichtung FLASCHEN
Typ 6: Eintritt in die Warteschlange FLEISCH
Typ 7: Austritt aus der Warteschlange FLEISCH
Typ 8: Belegung der Einrichtung FLEISCH
Typ 9: Verlassen der Einrichtung FLEISCH
Typ 10: Beginn der Selbstbedienung in den Regalen
Typ 11: Ende der Selbstbedienung in den Regalen
Typ 12: Auswahl der entsprechenden KASSE
Typ 13: Eintritt in die Warteschlange der KASSE

Typ 14: Austritt aus der Warteschlange KASSE
Typ 15: Belegung der Einrichtung KASSE
Typ 16: Verlassen der Einrichtung KASSE
Typ 17: Vernichten des Aktivators

Betrachtet man diese verschiedenen Ereignistypen, so ist zu bemerken, daß sich z.B. der Ereignistyp Eintritt in eine Warteschlange mehrmals wiederholt. Bei der Programmierung dieser Ereignistypen stellt man fest, daß sich viele Teilalgorithmen, wie z.B. das Einordnen in eine Liste und die Berechnung des Zeitintegrals wiederholen. Für diese Teilaufgaben kann auf die aufgeführten Unterprogramme zurückgegriffen werden. Die Planung des nächsten Ereignisses bzw. der nächsten Ereignisse muß unter Beachtung der Merkmale für jeden Ereignistyp vorgenommen werden.
Bei komplexeren Aufgabenstellungen muß für diese Arbeit ein beträchtlicher Zeitaufwand berücksichtigt werden.
Aus diesem Grund wurden Simulationssprachen entwickelt, die eine selbständige Ablaufsteuerung der Ereignisse vornehmen und damit eine wesentlich schnellere und kostengünstigere Erarbeitung von Simulationsmodellen gestatten. Eine Beschreibung dieser internen Steuerung wird in [1] gegeben.
Das Simulationsmodell wird in diesem Fall mit Hilfe von sogenannten Blöcken beschrieben, auf deren Basis dann die entsprechenden Rechenoperationen selbständig durchgeführt werden. In der DDR wird zur Simulation diskreter Modelle vielfach die Simulationssprache PS SIMDIS verwendet. Diese Sprache ist zur Zeit nur auf Rechnern der ESER-Reihe verfügbar.

Literatur

[1] Frank, M.; Lorenz, P.: Simulation diskreter Prozesse. – Leipzig: Fachbuchverlag, 1979

Autoren:

Prof. Dr. sc. nat. Peter Lorenz
Dr. Thomas Schulze

Technische Universität »Otto von Guericke« Magdeburg
Sektion Rechentechnik und Datenverarbeitung

Rechentechnische Begriffe für den Laien erklärt

ROM Read-Only-Memory, Nur-Lese-Speicher. Festwertspeicher, dessen Programmierung einmalig und endgültig beim Hersteller erfolgt. Als Programmspeicher eingesetzt. Schaltkreistypen: U 501 D (1302): 256 × 8 bit; U 505 D (2308): 1K × 8 bit; U 2316 D/2332/U 2364 D, U 2365 D/23128: 2/4/8/16K × 8 bit.

PROM Programmable ROM, programmierbarer Festwertspeicher, dessen Programmierung einmalig und endgültig beim Anwender erfolgt (z.B. durch Ausbrennen von Leitungspfaden mittels Programmierspannungen > 20 V oder durch Kurzschließen der Basis-Emitter-Dioden von Bipolartransistoren durch Stromimpulse). Schaltkreistypen: U 551 D (1602): 256 × 8 bit; U 2616 D: 2K × 8 bit.

Nutzung von Kleincomputern für den Physikunterricht in der Schule

1. Einleitung und Bedingungen

An der Notwendigkeit, Schüler, Lehrlinge und Studenten möglichst zeitig an die moderne Rechentechnik heranzuführen, bestehen wohl keine Zweifel (vgl. z. B. HAMM, 1984). Im Bereich der Volksbildung stehen dabei zunächst die Spezialschulen naturwissenschaftlich-technischer Richtung im Sinne einer Förderung begabter Schüler im Mittelpunkt des Interesses. Der Wissenschaftsbereich Methodik des Physikunterrichtes an der Sektion Physik der Friedrich-Schiller-Universität Jena beschäftigt sich im Rahmen seiner Forschung u. a. mit Problemen der Verbindung zwischen Informatikausbildung und Physikunterricht. Eine wesentliche Position ist dabei, die Einführung der Rechentechnik an den Schulen nicht allein einem vorrangig mathematisch profilierten Informatikunterricht zu überlassen. Die Rechentechnik muß vielmehr, wenn auch in unterschiedlichem Maße und mit verschiedener Spezifik, in jedes Unterrichtsfach Eingang finden, wobei naturgemäß der Physik eine besondere Bedeutung zukommt.

Mit diesem Anliegen leiten die Autoren seit September 1985 in der Spezialschule in Jena mehrere Zirkel mit je 8 Teilnehmern aus den Klassen 11 und 12. Die Schüler wurden vorrangig auf Grund ihres Studienwunsches ausgewählt. Es handelt sich um Schüler mit durchweg mindestens guten Leistungen, von denen einige schon bei der Arbeit am Bürocomputer Vorkenntnisse erworben haben. Ziel dieser Tätigkeit ist neben der Einführung in die Arbeit mit Kleincomputern und in die Programmiersprache BASIC vor allem die Bearbeitung von unterschiedlichen Problemen aus der Physik. Über erste Erfahrungen soll im folgenden berichtet werden.

Für die Arbeit stehen Kleincomputer KC 85/1 und KC 85/2 im Fachkabinett der Schule in ausreichender Zahl zur Verfügung, wobei die Schüler i. allg. einzeln an je einem Gerät arbeiten.

2. Kursinhalte und einige Erfahrungen

Der Zeitfonds für die Zirkeltätigkeit umfaßt 30 Wochen zu je zwei Stunden, wobei sich eine zusammenhängende Arbeitszeit von 120 Minuten als günstig erwiesen hat. Die an den beiden Leitlinien »Informatik« und »Physikunterricht« orientierten Kursinhalte lassen sich in folgende Schwerpunkte gliedern:

1. Einführung in die Programmierung mit BASIC und Heranführen an das Arbeiten mit Kleincomputern
   - Grundstruktur eines Kleincomputers
   - Überblick Programmiersprachen

- Algorithmen-Grundelemente
- Programmablaufplan, Struktogramm
- BASIC-Kurzlehrgang

2. Entwicklung von Programmierfertigkeiten durch die Bearbeitung geeigneter Problemkreise aus Physik und Mathematik sowie durch die Programmierung von Spielen
   - Praktische Mathematik (ausgewählte numerische Verfahren)
   - NEWTONsche Mechanik
   - Kondensator, Spule
3. Nutzung der Aufbereitung ausgewählter physikalischer Sachverhalte für eine Bearbeitung mit dem Kleincomputer zur Vertiefung und Erweiterung physikalischer Kenntnisse
   - Allgemeiner Fall des schrägen Wurfs
   - Mechanische Schwingungen
   - Passive R-C-Pässe, Spannungsteiler, Impulsformung
   - Elektrische Schwingungen, Schwingkreis

Die kurz charakterisierten inhaltlichen Schwerpunkte durchdringen einander vielfältig, keinesfalls ist an ein chronologisches Vorgehen an Hand dieser Gliederung gedacht. So wurde von der ersten Zusammenkunft an ohne Ausnahme in jeder Veranstaltung mit den Schülern am Computer gearbeitet.
Die im Schwerpunkt 2. aufgeführten Verfahren der praktischen Mathematik sind in den Kurs insbesondere deshalb aufgenommen worden, weil die ausgewählten physikalischen Problemkreise kaum die Möglichkeit bzw. Notwendigkeit des Arbeitens mit Unterprogrammen enthielten, die UP-Technik aber Lehrgegenstand sein sollte. Gleichzeitig erwerben die Schüler damit Grundkenntnisse auf einem im Unterricht bisher unterrepräsentierten Gebiet der Mathematik. Eine besondere Funktion kommt in der skizzierten Konzeption auch der Programmierung von Spielen zu. Sie regen durch ihre spezifische, eben die Spielproblematik, zu anderen Betrachtungsweisen bei der Problemanalyse an als sie bei der Bearbeitung der physikalischen Beispiele in der Mehrzahl auftreten. Spielprogrammierung fördert erfahrungsgemäß Einfallsreichtum und variables Denken, während sich bei der Bearbeitung der unter einem Systemaspekt ausgewählten physikalischen Sachverhalte bestimmte Programmstrukturen wiederholen und damit abheben lassen, die Gefahr einer gewissen Monotonie sichtbar wird.
Im folgenden sollen einige Programmbeispiele die Kursinhalte exemplarisch konkretisieren. Die vorgestellten Programme bzw. Programmteile betrachten wir andererseits auch als Anregungen für den Leser, sich mit physikalischen Fragestellungen durch den Einsatz eines Kleincomputers intensiver zu beschäftigen.

3. Nutzung von Mikrorechnern in der Physikausbildung

3.1. Ausgewählte Probleme der Mechanik

In einer der genannten Schülergruppen (Klasse 11) wurde die Behandlung von Problemen der NEWTONschen Mechanik in Absprache mit den Schülern als Ziel gestellt, obwohl dieses Gebiet aus der Sicht der Schüler nicht das interessanteste ist. Neben der Vertiefung von Wissen und Können in BASIC und der exemplarischen Behandlung ausgewählter Einzelprobleme war es vor allem unser Anliegen, das prinzipielle Herangehen an Aufgaben der Kinematik und Dynamik (auf diese Teilgebiete haben wir uns im wesentlichen beschränkt) deutlich werden zu lassen. Damit möchten wir die Schüler befähigen, weitere Aufgaben aus der Mecha-

nik selbständig bearbeiten zu können und das Gelernte auch auf völlig neue Gebiete der Physik übertragen zu können.
Entsprechend dem gestellten Ziel wurden zunächst die bereits aus Klasse 9 bekannten Grundgleichungen der Kinematik in folgender Weise auf die Rechentechnik zugeschnitten:

$$V = V + A * D$$

$$X = X + V * D$$

$$T = T + D$$

Dabei ist D das Zeitintervall dt bzw. Δt. Nach Eingabe der gegebenen Anfangswerte wurden damit zunächst bekannte Bewegungsvorgänge iterativ behandelt: Gleichförmige Bewegung ($a = 0$, Vorgabe v), freier Fall (Vorgabe $a = g = -9{,}81\ \mathrm{m/s^2}$ und x_0) und senkrechter Wurf (Vorgabe $a = g$ und v_0). Diese Beispiele haben den Vorzug, daß die Ergebnisse auch elementar berechenbar und damit nachprüfbar sind, sie haben aber den Nachteil, daß der Schüler die Notwendigkeit der Nutzung von Kleincomputern nicht erkennt. Daher wurde anschließend die Fallbewegung aus großer Höhe (Mondentfernung) behandelt. Dabei ist die Beschleunigung jedesmal aus dem Gravitationsgesetz neu zu berechnen:

$$A = -K * M / X / X$$

Spätestens an dieser Stelle erkennt der Schüler folgendes Problem: Wahl eines relativ großen Δt (D) führt zu sehr ungenauen (u.U. sinnlosen) Ergebnissen, andererseits liefert ein kleines Δt zwar genaue, aber unbequem viele Werte (insbesondere bei numerischer Ausgabe der Ergebnisse). Als ein möglicher Ausweg wurde dazu übergegangen, beispielsweise nur jeden zehnten Wert auf dem Bildschirm auszugeben.
Das Programm für den »freien Fall aus großer Höhe« sieht dann folgendermaßen aus (Bild 1). (Wir geben in den in Abschnitt 3.1. angeführten Beispielen nur die physikalisch-mathematisch wesentlichen Programmteile an und lassen die Eingaberoutine für die Konstanten und Anfangswerte sowie die Ausgaberoutine weg.)

```
50  Z=10
60  A=-K*M/X/X
70  V=V+A*D
80  X=X+V*D
90  T=T+D
100 Z=Z-1
110 IF Z>0 THEN 60
120 (AUSGABE NUMERISCH ODER GRAFISCH)
130 GOTO 50
```

Bild 1. Freier Fall aus großer Höhe

Ein Abbruch ist hier nur durch Betätigen der STOP-Taste möglich. Die Größe Z wird hier als »Zähler« eingeführt, man kann den Wert auch anders wählen. Damit lernen die Schüler, Differentialgleichungen (2. Ordnung) zu lösen, ohne daß ihnen das zunächst bewußt wird. Weitere einfache Beispiele, wie z. B. der schräge Wurf, konnten von ihnen weitgehend selbständig bearbeitet werden. Damit konnten dann auch auf einfache Weise Probleme gelöst werden, deren geschlossene mathematische Bearbeitung sehr aufwendig ist, beispielsweise die Berechnung des Abwurfwinkels für maximale Wurfweite bei gegebener Anfangsgeschwindigkeit und von Null verschiedener Abwurfhöhe.
Für wesentlich halten wir die Position, daß eine numerische Lösung, wie wir sie hier gewinnen, der geschlossenen, analytisch gewonnenen Lösung völlig gleichwertig ist! Auf ein methodisches Problem sei an dieser Stelle hingewiesen: Die Schüler neigen häufig dazu, die Hauptarbeit beim Erarbeiten der Programme in äußere Dinge, wie Gestaltung der Überschrift, Bereitstellung eines »Bildschirmfensters« zu investieren und das mathematisch-physi-

kalisch Wesentliche etwas an den Rand der Betrachtung zu schieben. Deshalb haben wir angestrebt, als Problemlösung in den meisten Fällen lauffähige »Rohprogramme« mit vereinfachter Ein- und Ausgaberoutine anzuerkennen. Nur an der Bearbeitung einiger Beispiele (vgl. Abschnitt 3.2.) wurde veranschaulicht, wie ein nutzerfreundliches Programm mit gewissem »Komfort« daraus zu entwickeln ist. Von der Zweckmäßigkeit dieser Entscheidung überzeugten sich die Schüler an Hand des auf ein Vielfaches anwachsenden Zeitbedarfs für die Erstellung derartiger Programme, ohne daß wesentliche Beiträge zur Realisierung der Leitlinie »Physikunterricht« erbracht worden wären.

Als wesentlich neues Problem für die Schüler wurde u.a. die Bewegung von Himmelskörpern im Zentralfeld behandelt. Zwar ist aus dem Physik- und Astronomieunterricht bekannt, daß die möglichen Bahnen Kegelschnitte (Kreise, Ellipsen, Parabeln und Hyperbeln) sind, aber unter welchen Bedingungen welche Bahnform entsteht, ist nur wenig bekannt, und der Zusammenhang mit dem Gravitationsgesetz ist für sie völlig neu.

Als wesentlich wurde zunächst herausgestellt, daß das Problem analog zu den vorhergehenden zu behandeln ist, nur ist die Bewegung hier zweidimensional, d.h., obige Überlegungen und obiger Formalismus sind nun auf zwei Richtungen (x und y, entsprechende Geschwindigkeiten v und w) anzuwenden.

Notwendige Vorbetrachtungen: Für die Beschleunigung gilt $a = k \cdot m/r^2$ (k Gravitationskonstante), für die x-Komponente $a_x = -\frac{a \cdot x}{r} = -\frac{k \cdot m \cdot x}{(x^2+y^2)^{1,5}}$ (wegen $r^2 = x^2 + y^2$) und entsprechendes für die y-Komponente. Als Beispiel wurde für m die Masse der Erde eingegeben; dann ist $k \cdot m = 4 \cdot 10^{14}\ \mathrm{m^3/s^2}$, weiter wurde $x_0 = 0$ und $y_0 = 3{,}84 \cdot 10^8\ \mathrm{m}$ (Mondentfernung) eingegeben. Die Kreisbahngeschwindigkeit beträgt für diese Werte $v_0 = 1020{,}62\ \mathrm{m/s}$. Beim Lauf eines entsprechenden Programmes mit diesen Werten und Ausgabe von Winkel φ und Abstand r erhält man ($\Delta t = 1000\ \mathrm{s}$, $N = 20$, also Ausgabe nach jeweils 20000 s) für $\varphi = 90°, 180°, 270°$ und $360°$ Werte für $r = 383491$ km, 384001 km, 384512 km und 384001 km. Das ist unbefriedigend, soll doch die Bahn mit den gewählten Werten kreisförmig sein! Daran ändert auch der »gute« Wert für $\varphi = 360°$ nichts. Eine Verkleinerung von Δt ändert das nicht wesentlich, erhöht aber bedeutend die Rechenzeit. An dieser Stelle wurde mit den Schülern (die inzwischen im Mathematikunterricht in die Differentialrechnung eingeführt worden sind) in sehr einfacher Form mit gutem Erfolg das EULERsche Verfahren zur Verbesserung der Rechnung eingeführt. Bezüglich dieser Methode muß hier auf die entsprechende Literatur (z.B.[1]) verwiesen werden. Es soll aber der entsprechende Programmteil angegeben und erläutert werden (Bild 2).

```
60  A=-K*M/((X*X+Y*Y)^1.5)
70  V=V+A*X*D/2
80  W=W+A*Y*D/2
90  Z=20
100 X=X+V*D
110 Y=Y+W*D
120 A=-K*M/((X*X+Y*Y)^1.5)
130 V=V+A*X*D
140 W=W+A*Y*D
150 T=T+D
160 Z=Z-1
170 IF Z>0 THEN 100
180 (AUSGABE T,R,FI ODER T,X,Y ODER GRAFISCH)
190 GOTO 90
```

Bild 2. EULERsches Verfahren

In (60) wird im wesentlichen die Beschleunigung in Abhängigkeit von x und y berechnet, der berechnete Wert A ist gleich a/r, er ergibt, mit x bzw. y multipliziert, die Beschleunigungskomponenten. In (70) und (80)

geschieht das Entscheidende! Hier werden die Geschwindigkeitskomponenten neu berechnet, aber nicht wie in früheren Beispielen zu einem um Δt späteren Zeitpunkt, sondern nur zu einem um $\Delta t/2$ später gelegenen Zeitpunkt. Damit wird versucht, für die Neuberechnung der Ortskoordinaten in (100) und (110) solche Geschwindigkeitswerte zu nutzen, die in der Mitte **zwischen** t und $t + \Delta t$ liegen (natürlich sind diese wegen der Unkenntnis über die Bahnkurve weder genau angebbar, noch sind genau **diese** Werte einzusetzen). In (120) wird wiederum die Beschleunigung berechnet und in (130) und (140) wieder **Zwischenwerte** für die Geschwindigkeitskomponenten. In (150) wird dann die Zeit inkrementiert, in (160) der »Zähler« dekrementiert. Zeile (170) führt nach 20 Schritten zur Ausgabe (die übrigens auch an anderer Stelle, z.B. nach (110) stehen könnte), anderenfalls zum nächsten Zyklus. Nach (190) wird ebenfalls der nächste Rechenzyklus begonnen, nachdem der »Zähler« neu gesetzt wurde.

So einfach dieses Verfahren ist (weitere Verbesserungen sind möglich, wurden aber mit den Schülern nicht behandelt), so liefert es doch wesentlich bessere Ergebnisse. Die maximale Abweichung von der Kreisbahn liegt hier ($\Delta t = 1000$ s) für einen Umlauf bei 2 km (Genauigkeit $5 \cdot 10^{-6}$!) und erhöht sich für $\Delta t = 5000$ s lediglich auf 34 km. Bei Veränderung der Anfangsgeschwindigkeit lassen sich nun die unterschiedlichsten Bahnen (Ellipsen, Hyperbeln) »erzeugen«, was insbesondere bei grafischer Darstellung mit dem KC 85/2 sehr reizvoll ist. Die sich hier abzeichnende Problematik, in welchem Maße der Einsatz moderner Rechentechnik in der Schule Veränderungen im Mathematikunterricht (Probleme der bisher kaum behandelten Numerik) erfordert, sei hier nur angedeutet.

Als ein sehr interessantes Beispiel wird gegenwärtig mit der Schülergruppe der sehr allgemeine Fall des schrägen Wurfes behandelt. Dieses Problem ist deshalb so interessant, weil eine geschlossene Lösung hier **prinzipiell** nicht möglich und weil es sehr vielseitig ist. Es wird der Luftwiderstand berücksichtigt, wobei die Luftdichte in Abhängigkeit von der Höhe einbezogen wird. Damit sind neben ballistischen Aufgabenstellungen z.B. auch Probleme des Eintritts von Erdsatelliten in die Erdatmosphäre oder die Bewegung fallender Regentropfen berechenbar.

Das grundsätzliche Herangehen an die Problematik entspricht obigen Beispielen, aber einige Besonderheiten sind zu beachten. Wenn der Luftdruck bzw. die Luftdichte bis in große Höhen (110 km) berechnet werden soll, ist die barometrische Höhenformel nicht mehr ausreichend. Im Programmausschnitt von Bild 3 werden in (10) bis (60) die Zehner-Logarithmen des relativen Luftdruckes (für $h = 0$ ist $p_r = 1$) in Abständen von 10 km als Feld eingelesen. Mit 12 Werten wird damit eine Höhe bis 110 km erfaßt. Vor jeder Berechnung der Beschleunigung wird das

```
10  DIM F(12)
20  DATA 0,-.4656,-1.1391,-1.8229,...(12WERTE)
30  FOR N=0 TO 11
40  READ B
50  F(N)=B
60  NEXT N
70  GOSUB 400
400 YR=Y/10000
410 N=INT(YR)
420 S=F(N)+(F(N+1)-F(N))*(YR-N)
430 Q=10^S
440 R=C*Q*SQR(U*U+V*V)
450 IF R<1 THEN E=24/R:GOTO 480
460 IF R<400 THEN E=24/(R^.646):GOTO 480
470 E=0.5
480 E=-E*R*B
490 RETURN
```

Bild 3. Berechnung des Luftdruckes bis in große Höhen

Unterprogramm (400) zwecks Berechnung des Luftwiderstandes aufgerufen. Dort erfolgt in (400) bis (420) eine lineare Interpolation zwischen den Logarithmen des Luftdruckes, die sehr genau ist, weil der Logarithmus des Luftdruckes näherungsweise proportional zur Höhe ist (in der barometrischen Höhenformel ist das genau der Fall). In (430) wird dann der relative Luftdruck in der entsprechenden Höhe berechnet.

Zur Berechnung des Luftwiderstandes wird berücksichtigt, daß dieser nicht nur von der Luftdichte und der Geschwindigkeit abhängt, sondern daß in verschiedenen Bereichen dafür auch unterschiedliche Gleichungen zu benutzen sind. Welche Gleichung zu benutzen ist, hängt von der REYNOLDschen Zahl R ab ($R = l \cdot v \cdot \varrho/\eta$, dabei ist l eine charakteristische Länge, v die Geschwindigkeit, ϱ die Dichte und η die Zähigkeit der Luft, Näheres siehe z.B. [2]). Diese Zahl wird in (440) berechnet. In Abhängigkeit von deren Wert wird dann nach unterschiedlichen Gleichungen in (450) bis (480) der wesentliche Teil des Luftwiderstandes berechnet. B und C sind dabei früher eingegebene Konstanten, in die Dichte und Zähigkeit der Luft sowie Masse und Radius des bewegten Körpers eingehen.

Das gesamte Programm ist so angelegt, daß wahlweise das Unterprogramm (400 bis 490) umgangen werden kann. In diesem Fall erfolgt die Berechnung der Bahn ohne Luftwiderstand, und insbesondere bei grafischer Darstellung mit dem KC 85/2 sind interessante Vergleiche möglich. Es wird u.a. deutlich, daß der Luftwiderstand die Bahnkurve ganz erheblich beeinflußt. An diesem Beispiel wird die mögliche Komplexität der behandelbaren Probleme deutlich.

3.2. Ausgewählte Probleme der Elektrik

Für einen aus Schülern der Klasse 12 gebildeten Zirkel wurde das Ziel formuliert, ausgehend von den Lade- und Entladevorgängen des Kondensators, den Ein- und Ausschaltvorgängen der Spule über die Betrachtung von passiven R-C-Pässen zum Schwingkreis zu gelangen. Die rechentechnische Bearbeitung sollte grundsätzlich *ohne* Lösung der betreffenden Differentialgleichungen, also ausschließlich durch iterative Ansätze erfolgen. Das entspricht der in Abschnitt 3.1. an Beispielen aus der Mechanik dargestellten gleichen Grundkonzeption.
Für das Teilziel Kondensator stellen wir zwei unterschiedlich gestaltete Programme vor, die im Zirkel während des ersten Schulhalbjahres entstanden sind.

Beispiel 1: Laden eines Kondensators

(Ausgabe auf dem KC 85/1, vgl. Bild 4)
Ein Kondensator mit der Kapazität C wird über einen Widerstand R an eine Quelle mit der Spannung U_0 angeschlossen. In einer Tabelle ist die Kondensatorspannung u_C in Abhängigkeit von der Zeit darzustellen.
Für den gegebenen Sachverhalt gilt $U_0 = u_C + i \cdot R$, wobei mit i der momentane Ladestrom erfaßt wird. Innerhalb hinreichend kleiner Zeitintervalle Δt kann dieser Ladestrom $i = \frac{U_0 - u_C}{R}$ als konstant betrachtet werden.
Im Sinne von Zuweisungen gelten damit die folgenden Relationen:

$$\Delta u = \frac{U_0 - u_C}{R \cdot C} \cdot \Delta t \qquad (130)$$

$$u_C = u_C + \Delta u \qquad (140)$$

$$t = t + \Delta t \qquad (150)$$

Wie ist die Forderung nach sehr kleinen Zeitintervallen Δt zu erfüllen? Eine

```
10 WINDOW:CLS:PRINT
20 INPUT"U0,R,C:";U0,R,C
30 PRINT:PRINT
40 PRINT"t in ms","U in V","DU in mV"
50 PRINTSTRING$(36,CHR$(95))
60 WINDOW 7,21,0,39
70 PRINT SPC(2)"0",SPC(2)"0",U0
80 TAU=R*C:DT=TAU/1E3:IF TAU>=1E-3 THEN120
90 PRINTAT(3,0);"t in MIKRO-":PRINTAT(4,0);STRING$(4,CHR$(32))
100 PRINTAT(4,1);"SEKUNDEN"
110 PRINTAT(1,24);"TAU=",TAU*1E6,"MIKRO-":PRINTAT(2,31);"SEKUNDEN":GOTO130
120 PRINTAT(1,26);"TAU=",TAU*1E3,"ms"
130 DU=(U0-U)*DT/TAU
140 U=U+DU:N=N+1
150 T=T+DT
160 IF N<200 THEN130
170 IF TAU>=1E-3 THEN190
180 PRINT INT(T*1E6*1E2+.5)/1E2;:GOTO200
190 PRINT INT(T*1E3*1E2+.5)/1E2;
200 PRINT,INT(U*1E2+.5)/1E2,INT(DU*1E3* E2+.5)/1E2:N=0
210 PRINTAT(23,11);STRING$(12,CHR$(32))
220 PRINTAT(23,11);"U=",INT(U0*(1-EXP(-T/TAU))*1E2+.5)/1E2,"V"
230 GOTO130
```

Bild 4. Laden eines Kondensators

beliebige Festlegung von Δt erweist sich als wenig sinnvoll, da auf diese Weise die mit $\tau = R \cdot C$ erfaßte zeitliche Gesetzmäßigkeit des Ladevorganges unberücksichtigt, aber von wesentlichem Einfluß auf die Aussagekraft der Tabelle bliebe. Als zweckmäßig erscheint deshalb die Berechnung von Δt innerhalb des Programmes (80) in Abhängigkeit von der Zeitkonstanten τ.

Nachteilig für Umfang und Aussagekraft der angestrebten Tabelle wirkt sich die mit der Festlegung von Δt ausgelöste »Meßwertflut« aus. Zur Verbesserung der Übersichtlichkeit wird im Programm vereinbart (160), nur beispielsweise jeden 200sten der berechneten u_C-Werte in der Tabelle auszugeben. Damit wird den Forderungen nach Genauigkeit und Übersichtlichkeit gleichermaßen entsprochen.

Zur Beurteilung der mit $\Delta t = \tau \cdot 10^{-3}$ erreichten Genauigkeit erfolgt parallel zum entwickelten Lösungsverfahren zu jedem ausgegebenen Tabellenwert von u_C die Berechnung und Ausgabe (220) der momentanen Kondensatorspannung nach der bekannten Beziehung $u_C = U_0 (1 - \exp(-t/\tau))$. Als Vorbereitung auf das Verständnis der sich anschließenden grafischen Veranschaulichung des betrachteten Vorganges ist die Ausgabe der Spannungsänderung am Kondensator während des Ladens in der 3. Tabellenspalte zu sehen (200).

In (80) wird eine Alternative für $\tau < 10^{-3}$ s organisiert, die in (90), (100), (110), (180) realisiert ist. Hingewiesen sei ferner auf die Festlegung der Stellenzahl N nach dem Komma, mit der ein Wert Z ausgegeben werden soll, durch die Operation

$$\text{INT}\,(Z * 10^N)/10^N,$$

und auf die »Rundungsautomatik« mit + 0.5 in den Zeilen (180), (190), (200), (220).

Der Programmlauf kann durch ein STOP-Kommando ab- bzw. unterbrochen werden.

```
10  WINDOW:CLS
20  PRINT"Spannungsverlauf beim Laden und Entladen"
30  PRINTSPC(11)"eines Kondensators":PRINT
40  PRINTSTRING$(40,CHR$(238))
45  REM KOORDINATENSYSTEM
50  PRINTAT(22,1);STRING$(38,CHR$(248))
60  FOR I=6 TO 22
70  PRINTAT(I,1);CHR$(159)
80  NEXT I
90  PRINTAT(22,1);CHR$(136)
100 PRINTAT(23,36);"t"
110 PRINTAT(23,18);"4*TAU"
120 PRINTAT(7,1);"u"
130 PRINTAT(9,0);"U0",STRING$(38,CHR$(46))
135 REM NORMIERUNG UND RECHNUNG -LADEN-
140 R=1:C=1:U0=1
150 TAU=R*C:T0=4*TAU
160 DT=TAU/1E2
170 DU=(U0-U)*DT/TAU
171 REM BLINKENDE ANZEIGE  >LADEN(
172 IF K)=10 THEN PRINTAT(9,8);">LADEN("
173 K=K+1
174 IF K(10 THEN PRINTAT(8,8);STRING$(7,CHR$(32))
175 IF K=20 THEN K=0
180 U=U+DU:N=N+1
190 T=T+DT
195 REM GRAPHIK  -LADEN-
200 IF N(23 THEN 170:ELSE X=INT(T/T0*18+.5)+1:Y=23-INT(U/U0*13+.5)
210 PRINTAT(Y,X);CHR$(213):N=0
220 IF T(=T0 THEN 170
225 REM RECHNUNG  -ENTLADEN-
230 DU=-U*DT/TAU
231 REM BLINKENDE ANZEIGE  >ENTLADEN(
232 IF K(10 THEN PRINTAT(8,25);">ENTLADEN("
233 K=K+1
234 IF K)=10 THEN PRINTAT(8,25);STRING$(10,CHR$(32))
235 IF K=20 THEN K=0
240 U=U+DU:N=N+1
250 T=T+DT
260 IF N(23 THEN 230
270 X=INT(T/T0*18+.5)+1:Y=22-INT(U/U0*13+.5)
280 PRINTAT(Y,X);CHR$(213):N=0
290 IF T(=2*T0 THEN 230
295 REM KURSOR VERBANNEN
300 WINDOW 5,5,39,39:STOP
```

Bild 5. Spannungsverlauf beim Laden und Entladen eines Kondensators

Beispiel 2: Laden und Entladen von Kondensatoren (Ausgabe auf dem KC 85/1). Der typische Verlauf der Spannung beim Laden und Entladen von Kondensatoren ist darzustellen.

Die Überlegungen zum Ladevorgang sind vom Beispiel 1 übertragbar; für den Entladevorgang gilt entsprechend $u_C = -i \cdot R$. Die Spannungsänderung berechnet sich zu $\Delta u = -\frac{u_C}{R \cdot C} \cdot \Delta t$.

Damit stehen für den Programmteil »Entladen« folgende Beziehungen zur Verfügung:

$$\Delta u = -\frac{u_C}{R \cdot C} \cdot \Delta t \qquad (230)$$

$$u_C = u_C + \Delta u \qquad (240)$$

$$t = t + \Delta t \qquad (250)$$

Das Programm wird in Bild 5 für den KC 85/1 angegeben.

4. Abschließende Bemerkungen

Weitere sehr interessante Beispiele, die mit den Schülern bearbeitet wurden bzw. werden, sind freie, gedämpfte und erzwungene mechanische und elektrische Schwingungen. Die numerische Behandlung erzwungener Schwingungen führt zu überraschenden Einsichten über das Einschwingverhalten der Systeme, die auch die geschlossene Behandlung in der Grundausbildung an Hochschulen nicht liefert.

Hervorzuheben ist bei allen behandelten Problemen das große Interesse aller beteiligten Schüler an der Mechanik, die sonst auch an Spezialschulen wegen ihrer geringen Anziehungskraft für Schüler nicht leicht zu unterrichten ist! Zusammenfassend lassen die Erfahrungen bei der Arbeit mit den Schülern die Aussage zu, daß der konzeptionelle Ansatz einer Verbindung von Informatik-Grundausbildung und vertiefender Behandlung ausgewählter, auch anspruchsvollerer Probleme der Schulphysik unter den eingangs skizzierten Bedingungen aus der Sicht aller Beteiligten erfolgreich umgesetzt werden konnte.

Literatur

[1] Zurmühl, R.: Praktische Mathematik für Ingenieure und Physiker. – Berlin, Göttingen, Heidelberg, 1963

[2] Mende, D.; Simon, G.: Physik – Gleichungen und Tabellen. – Leipzig, 1986

[3] Hamm, G.: Erfahrungen beim Einsatz programmierbarer Kleinstrechner in der Schule. – In: Kleinstrechner-TIPS, Heft 1. – Leipzig, 1984

Autoren:

Prof. Dr. sc. paed. Dr. rer. nat. Klaus Jupe

Pädagogische Hochschule
Dr. »Theodor Neubauer« Erfurt-Mühlhausen
Sektion Mathematik/Physik

Dr. Hans-Joachim Löhr

Friedrich-Schiller-Universität Jena
Sektion Physik
WB Methodik des Physikunterrichts

Rechentechnische Begriffe für den Laien erklärt

EPROM Erasable PROM, lösch- und reprogrammierbarer PROM, dessen Löschen z. B. durch UV-Bestrahlung erfolgen kann (Halbleiterchip deshalb unter Quarzfenster montiert); Programmierspannungen > 10 V. Schaltkreistypen: U 552 C (1702): 256 × 8 bit; U 555 C (2708): 1K × 8 bit; U 2716 C: 2K × 8 bit; U 2732 C: 4K × 8 bit; U 2764 C: 8K × 8 bit.

RMM Read-Mostly-Memory, Meist-Lese-Speicher, Festwertspeicher für seltenes (und damit zulässigerweise langsames) Einschreiben von Informationen, z. B. der derzeitige EAROM.

EAROM Electrically alterable ROM, elektrisch veränderbarer Festwertspeicher, dessen Programmierung im Speicherblock des Rechners mittels einer der normalen Betriebsspannungen erfolgen kann; z. Z. international noch in der Entwicklung und noch mit zu großen Zeiten für die Umprogrammierung behaftet.

Gleichungen – vom Computer gelöst

Eine lineare oder eine quadratische Gleichung lösen zu müssen, das ist für niemanden heute mehr ein Problem, denn für diese Gleichungstypen gibt es einfache Lösungsverfahren (lineare Gleichungen) oder Lösungsformeln (quadratische Gleichungen), mit deren Hilfe man die gesuchten Lösungen schnell und mit wenig Aufwand ermitteln kann. Zu ihrer Lösung einen Computer heranzuziehen ist wenig sinnvoll, denn bevor man das zugehörige Programm in den Rechner eingegeben hat, hat man die Gleichung schon längst mit den genannten Mitteln gelöst.

Problematischer wird die Sache allerdings, wenn man Gleichungen höheren als zweiten Grades lösen soll oder wenn in den zu lösenden Gleichungen kompliziertere mathematische Funktionen auftreten, wie das bei der Lösung technischer oder ökonomischer Probleme ja häufig der Fall ist. Für diese Sorte von Gleichungen gibt es nur in den seltensten Fällen Lösungsverfahren, die zu einer »geschlossenen Lösung« führen, so daß man zu *Näherungsverfahren* greifen muß, die erfahrungsgemäß sehr *rechenintensiv* und damit sehr *zeitaufwendig* sind. Hier könnte einem der Computer sehr wohl die lästige und umfangreiche Rechenarbeit abnehmen, wenn man ein geeignetes Programm hätte, mit dessen Hilfe *beliebige Gleichungen mit einer Variablen* gelöst werden können.

Aus diesem Grunde soll im folgenden ein Lösungsverfahren vorgestellt werden, dessen mathematische Grundlagen recht einfach einzusehen sind und das es gestattet, alle Lösungen einer Gleichung innerhalb eines vorgegebenen Bereiches mit einer frei wählbaren Genauigkeit zu ermitteln. Es ist ein sogenanntes *Einschachtelungsverfahren,* bei dem die Lösung der Gleichung anfangs nur in einer groben Annäherung ermittelt wird, diese Annäherung an den wahren Lösungswert dann aber so lange verbessert wird, bis er den geforderten Genauigkeitsansprüchen genügt.

Betrachten wir die Vorgehensweise, die diesem Näherungsverfahren zugrunde liegt:

Es soll vorausgesetzt werden, daß sich die zu lösende Gleichung in der Form

$$F(X) = 0 \qquad (1)$$

darstellen läßt. Diese Forderung ist im allgemeinen ohne Schwierigkeiten erfüllbar. So lassen sich beispielsweise die Gleichungen

$3x + 5 = 7x - 9$ oder
$0{,}27x^2 - 1{,}54x = 2{,}97$ oder
$5 \cdot e^{\sin x} = \ln(\cos x) + x^2/4 - 3$

ohne weiteres umformen in

$4x - 14 = 0,$
$0{,}27x^2 - 1{,}54x - 2{,}97 = 0$

bzw.

$$5 \cdot e^{\sin x} - \ln(\cos x) - x^2/4 + 3 = 0.$$

Nun betrachtet man *anstelle der Gleichung* (1) *die Funktion*

$$Y = F(X), \qquad (2)$$

die sich ja im X, Y-Koordinatensystem als *Kurve* darstellen läßt (vgl. Bild 1).

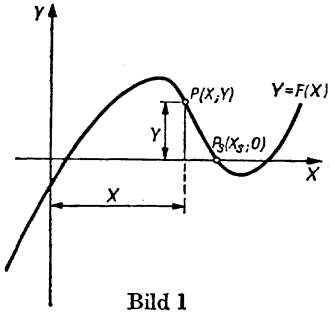


Bild 1

In einem Punkt P_s mit der Abszisse X_s, in dem diese Kurve die x-Achse schneidet, gilt für die zugehörige Ordinate $Y_s = 0$, d.h., für diesen Punkt gilt

$$F(X_s) = 0.$$

Dies wiederum bedeutet aber, daß X_s eine *Lösung der gegebenen Gleichung* (1) sein muß.

Die Gleichung (1) zu lösen heißt also, diejenigen Punkte aufzusuchen, in denen die Kurve, die zur Funktion (2) gehört, die X-Achse schneidet.

Um die Nullstellen der Funktion $Y = F(X)$ zu finden, kann man am linken Ende des gegebenen Intervalls jeweils zwei um die Anfangsschrittweite DX voneinander entfernte X-Werte

$$X_1 \text{ und } X_2 = X_1 + DX$$

betrachten und die zugehörigen Y-Werte Y_1 und Y_2 berechnen. Besitzen diese beiden *Y-Werte das gleiche Vorzeichen*, dann ist dies ein Zeichen dafür, daß die beiden Kurvenpunkte $P_1(X_1; Y_1)$ und $P_2(X_2; Y_2)$ auf derselben Seite der

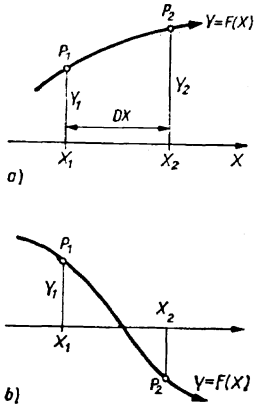


Bild 2

X-Achse liegen (vgl. Bild 2, Fall a) und daß – zumindest im Normalfalle – zwischen X_1 und X_2 keine Nullstelle der Funktion $Y = F(x)$ und damit auch keine Lösung der Gleichung $F(X) = 0$ liegt. Geht man nun entlang der X-Achse mit Schritten der Schrittweite DX weiter, so wird man – sofern die Gleichung innerhalb des zu untersuchenden Intervalles eine Lösung besitzt – an die Stelle kommen, wo sich die beiden zu den Teilintervallenden X_1 und X_2 gehörenden Y-Werte Y_1 und Y_2 *in ihrem Vorzeichen voneinander unterscheiden* (vgl. Bild 2, Fall b). In diesem Falle ist dann

$$Y_1 * Y_2 < 0, \qquad (3)$$

und die Werte X_1 und X_2 können als »grobe Näherungswerte« für die Lösung der Gleichung (1) angesehen werden.

Sollte die Gleichung innerhalb des zu untersuchenden Intervalles keine Lösung haben, so erreicht man das Intervallende, ohne zwei aufeinander folgende Y-Werte gefunden zu haben, für die die Bedingung (3) gilt. In diesem Falle kann die Rechnung abgebrochen werden, und man kann ggf. innerhalb eines anderen Intervalles darangehen, nach Lösungen der Gleichung zu suchen.

Hat man jedoch beim »Abtasten« der X-Achse ein Teilintervall $[X_1; X_2]$ gefunden, für das sich die Y-Werte Y_1 und Y_2 des Teilintervalls in ihrem Vorzeichen voneinander unterscheiden, so muß – eine stetige Funktion $Y = F(X)$ vorausgesetzt – innerhalb dieses Teilintervalles eine Lösung der Gleichung $F(X) = 0$ liegen.

Man geht dann zum linken Randpunkt dieses Teilintervalles zurück, verringert die bisherige Schrittweite auf 1/10 ihres Wertes und sucht nun in diesem kleineren Teilbereich in der gleichen Weise wie bisher nach einem neuen Teilintervall, innerhalb dessen der Übergang von positiven zu negativen bzw. von negativen zu positiven Y-Werten erfolgt.

Dieses Zurückgehen auf den Anfangspunkt des Teilintervalles, in dem der Vorzeichenwechsel bei den Y-Werten erfolgt mit gleichzeitigem Verringern der Schrittweite wird so lange fortgesetzt, bis die Schrittweite kleiner wird als die Genauigkeitsschranke, die an die zu ermittelnden Lösungen gestellt worden ist.

Auf diese Weise lassen sich die Lösungen der Gleichungen, die innerhalb des zu untersuchenden Intervalles liegen, so genau bestimmen, wie man es wünscht.

Der Arbeitsablauf für die Durchführung des Einschließungsverfahrens ist in Bild 3 in Form eines Struktogramms dargestellt (vgl. HORN: Wie kann ein Programm systematisch entworfen werden?, Heft 3).

Für die Berechnung der Lösungen einer Gleichung innerhalb eines bestimmten Intervalles ist – wie aus den bisherigen Darlegungen hervorgeht – naturgemäß ein erheblicher Rechenaufwand erforderlich. Da es sich jedoch bei den durchzuführenden Operationsfolgen immer wieder um gleichartige Rechnungen handelt, die jeweils nur mit immer neuen Zahlenwerten durchzuführen sind, ist es sinnvoll, ein Programm zu schreiben, durch das der Computer in die Lage versetzt wird, diese umfangreichen Rechenarbeiten durchzuführen und damit den Menschen von der monotonen Zahlenrechnung zu entlasten. Am Ende dieses Artikels (s. S. 62) ist ein solches Rechenprogramm in der Programmiersprache BASIC formuliert. das mit Hilfe der Kleincomputer KC 85/1 bzw. KC 85/2 (und nach geringfügigen Abänderungen zur Anpassung an den jeweiligen BASIC-Dialekt) auch mit Hilfe anderer Kleinrechner abgearbeitet werden kann. Dieses Programm ist dialogfähig gestaltet, so daß jeder Nutzer, der über die Prinzipien des Einschachtelungsverfahrens Bescheid weiß, seine Gleichungen mit Hilfe des vorliegenden Programmes lösen kann. Der Nutzer wird dabei vom Rechner geführt, so daß er auch keine speziellen Programmierkenntnisse besitzen muß, um das Programm abarbeiten zu können. Er muß lediglich in der Lage sein, das Programm in den Rechner zu laden, die linke Seite seiner Gleichung in einer ordnungsgemäßen BASIC-Notation einzugeben und das Programm zu starten. Innerhalb des Programmes wurde versucht, den eingeschlagenen Lösungsweg in Kurzform durch Kommentare zu erläutern.

Abschließend seien noch einige Bemerkungen zur Nutzung des Programmes angefügt.

Bei den ersten Eingabeanforderungen möchte der Rechner wissen, mit welcher Anfangsschrittweite das Intervall, innerhalb dessen nach Lösungen gesucht werden soll, abzutasten ist. Es ist sinnvoll, diese *Anfangsschrittweite nicht zu groß* zu wählen, denn dann kann es vorkommen, daß der Rechner Lösungswerte nicht findet. In Bild 4 ist ein solcher Fall grafisch dargestellt. Bei der dort gewählten Anfangsschrittweite DX

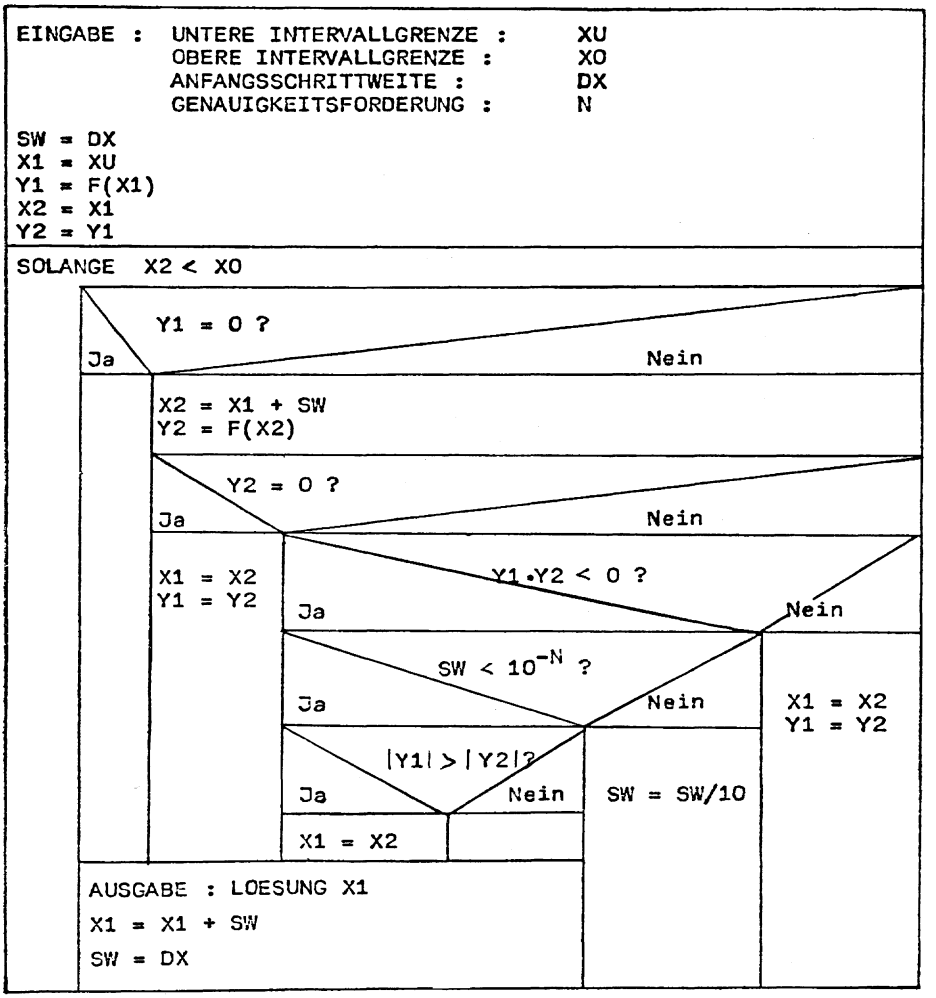


Bild 3. Struktogramm für die Lösung einer Gleichung mit Hilfe des Einschachtelungsverfahrens

erhält man sowohl bei X_1 als auch bei X_2 einen positiven Y-Wert. Der Rechner findet also keine Veranlassung, innerhalb des Intervalles $[X_1; X_2]$ nach einer Lösung der Gleichung zu suchen. – Hätte man von Anfang an die halbe Schrittweite gewählt, so würde der Rechner sowohl zwischen X_1 und X_2 als auch zwischen X_3 und X_2 nach einer Lösung der Gleichung suchen.

Mehrfache Lösungen der Gleichung von einer geradzahligen Ordnung (doppelte, vierfache, ... Lösungen) *werden* durch dieses Verfahren im allgemeinen *nicht aufgefunden*, es sei denn, die Anfangsschrittweite ist zufälligerweise so groß

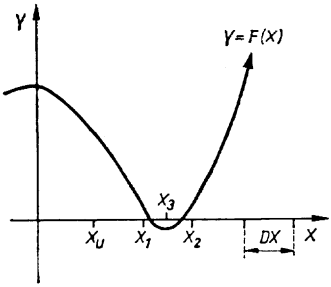


Bild 4

gewählt, daß der Endpunkt eines Suchintervalles genau auf den Punkt fällt, in dem die Kurve $Y = F(X)$ die X-Achse berührt. Da zwei-, vier-, sechsfache ... Lösungen jedoch nicht gerade sehr häufig auftreten, ist es wohl erlaubt, diesen Mangel des Programmes in Kauf zu nehmen.

Man kann das Verhalten des Programmes in einem solchen Fall überprüfen, indem man beispielsweise die Gleichung

$\sin x + 1 = 0$

in der Umgebung von $x = \frac{3}{2}\pi$ für verschiedene Genauigkeitsanforderungen untersucht. Wählt man $\Delta x = 0{,}001$, so erhält man mit dem KC 85/1 keine Lösung, für $\Delta x = 0{,}0005$ erhält man eine, für $\Delta x = 0{,}0002$ drei und für $\Delta x = 0{,}0001$ gar fünf, allerdings unmittelbar nebeneinander liegende Lösungen.

Auch die Genauigkeitsforderungen dürfen bei der Nutzung eines KC 85/1 bzw. KC 85/2 nicht zu hoch getrieben werden. Diese Rechner verarbeiten die reellen Zahlen in Form von Gleitkommazahlen mit einer fünf-, maximal sechsstelligen Mantisse. Das heißt, daß die von diesen Rechnern erreichbare Genauigkeit bei maximal 5 bzw. 6 geltenden Ziffern liegt. Liegt also das Lösungsintervall zwischen -1 und $+1$, so kann man ohne weiteres eine Genauigkeit von 10^{-5} fordern. Bei Intervallen, die darüber hinaus gehen, jedoch zwischen -10 und $+10$ liegen, ist die höchste zu fordernde Genauigkeit 10^{-4} usw.

Stellt man zu hohe Genauigkeitsanforderungen, so kann es vorkommen, daß der Rechner im Verlaufe der Verfeinerung der Intervallgrenzen zu keinem Ende kommt und die Rechnung durch Betätigung der ⟨STOP⟩- bzw. der ⟨HALT⟩-Taste gestoppt werden muß.

```
10 REM Programm zur Ermittlung der
20 REM reellen Loesungen einer be-
30 REM liebigen Gleichung der Form
40 REM
50 REM          F(X) = 0
60 REM
70 REM innerhalb eines vorgegebenen
80 REM Intervalls ⟨XU; XO⟩ mit einer
90 REM frei waehlbaren Genauigkeit
100 REM von N Stellen nach dem Komma.
110 REM
120 REM
300 CLS : PRINT : PRINT
310 PRINT "Geben Sie die linke Seite" ;
          " der zu loe-"
320 PRINT "senden Gleichung in der" ;
          " nachfolgenden"
330 PRINT "Anweisungszeile nachdem";
          " Gleichheits-"
340 PRINT "zeichen ein !"
350 PRINT : PRINT
360 PRINT "Druecken Sieanschlieszend";
          " in der an-"
370 PRINT "gegebenen Reihenfolge :"
380 PRINT
390 PRINT "     die ⟨ENTER⟩-Taste ,"
400 PRINT "     die ⟨STOP⟩-Taste ,"
410 PRINT "     die Zeichenfolge R⟩
                GOTO 450"
420 PRINT "     und die ⟨ENTER⟩
                -Taste !"
430 PRINT : PRINT
440 EDIT 450
450 DEF FNY(X) =
460 CLS
470 PRINT "  In welchem Intervall soll";
          " nach Loesun-"
480 PRINT "  gen gesucht werden ?"
490 PRINT
```

```
500 INPUT "    UNTERE INTERVALLGRENZE :" ;
          XU
510 INPUT "    OBERE INTERVALLGRENZE  :" ;
          XO
520 PRINT : PRINT
530 PRINT " Mit welcher Anfangsschritt-" ;
          " weite soll"
540 PRINT " das Intervall nach Loesun-"
          " gen abgesucht"
550 PRINT " werden ?"
560 PRINT
570 INPUT "    ANFANGSSCHRITTWEITE    :" ;
          DX
580 PRINT : PRINT
590 PRINT " Mit welcher Genauigkeit " ;
          " soll die Loe-"
600 PRINT " sung ermittelt werden ?"
610 PRINT
620 PRINT "    GEWUENSCHTE STELLENZAHL"
630 PRINT "    DES ERGEBNISSES NACH DEM"
640 INPUT "    KOMMA                  " ; N
650 PRINT
660 PRINT " HABEN SIE ALLES RICHTIG EIN" ;
          "GEGEBEN ?"
670 PRINT " WOLLEN SIE DIE EINGABEN " ;
          " NOCH EINMAL"
680 INPUT " KORRIGIEREN ?    (J/N)     " ; A$
690 IF A$ = "J" THEN 460
700 CLS : K = 0
710 PRINT : PRINT
720 PRINT "    ICH DENKE ANGESTRENGT  " ;
          "NACH ."
730 REM
740 REM In den Anweisungen 780 bis 940
750 REM wird das Intervall schrittwei-
760 REM se nach Loesungen abgetastet.
770 REM
780 SW = DX
790 X1 = XU
800 Y1 = FNY(X1)
810 Y2 = Y1
820 IF Y1 = 0 THEN 1070
830 X2 = X1 + SW
840 IF X2 > XO THEN 1250
850 Y2 = FNY (X2)
860 IF Y2 = 0 THEN 1070
870 IF Y1 * Y2 < 0 THEN 990
880 REM Im Falle Y1 * Y2 < 0 liegt zwi-
890 REM schen X1 und X2 eine Loesung.
900 REM
910 X1 = X2
920 Y1 = Y2
930 GOTO 800
```

```
940 REM
950 REM  Es folgt der Teil, in dem die
960 REM  Schrittweite auf ein Zehntel
970 REM  verkleinert wird.
980 REM
990 IF ABS (X2 - X1) < 10 ∧ (-N) THEN 1070
1000 SW = SW/10
1010 GOTO 800
1020 REM
1030 REM  Die Anweisungen 1080 bis 1180
1040 REM  enthalten die Ausgabe einer
1050 REM  aufgefundenen Loesung der ge-
1060 REM  gebenen Gleichung .
1070 REM
1080 PRINT : PRINT
1090 IF ABS (Y2) < ABS(Y1) THEN X1 = X2
1100 PRINT "  Die gegebene Gleichung " ;
           "hat bei"
1110 K = K + 1 : PRINT
1120 PRINT "          X = " ;
     INT (X1 * 10 ∧ N + 0.5)/10 ∧ N
1130 PRINT
1140 PRINT "  eine  " ;K; ". Loesung  . '
1150 PRINT :  PRINT
1160 SW = DX
1170 X1 = X1 + SW
1180 GOTO 800
1190 REM
1200 REM  Durch die Anweisungen 1250 bis
1210 REM  1470 wird die Rechnung abge-
1220 REM  schlossen, da die obere Inter-
1230 REM  vallgrenze erreicht ist .
1240 REM
1250 PRINT :  PRINT
1260 PRINT "  Mit X2 =  "; X2;" ist die" ;
           "  obere Inter-"
1270 PRINT "  vallgrenze erreicht bzw." ;
           "  ueberschritten  ."
1280 IF K = 0 THEN 1410 : PRINT
1290 PRINT "  DIE AUFGABE IST DAMIT" ;
           "  BEENDET  ."
1300 PRINT :  PRINT
1310 PRINT "  Soll dieselbe Gleichung" ;
           "  noch einmal"
1320 PRINT "  unter neuen Bedingungen" ;
           "  (Intervall, "
1330 PRINT "  Schrittweite, Genauigkeit) " ;
           "  geloest"
1340 INPUT "  werden ?    (J/N)      " ;A$
1350 IF A$  = "J" THEN 460
1360 CLS : PRINT : PRINT
1370 PRINT "  Wollen Sie mit diesem" ;
           "  Programm noch"
```

```
1380 PRINT  "   eine weitere Gleichung" ;
            "   loesen ?"
1390 INPUT  "   (J/N)                  " ;A$
1400 IF A$  = "J" THEN 300 : ELSE 1450
1410 PRINT  "   Die Gleichung besitzt in" ;
            "   dem gegebe-"
1420 PRINT  "   nen Intervall keine" ;
            "   Loesung  ."
1430 PRINT  :   PRINT
1440 GOTO 1290
1450 CLS
1460 PRINT  AT (10,5) ; "A U F   W I E" ;
            "   D E R S E H E N   !"
1470 END
```

Autor:

Prof. Dr.-Ing. Hans Kreul

a. o. Prof. an der Ingenieurhochschule Zittau
Abteilung EDV und Rechentechnik

ISBN 3-343-00227-5

1. Auflage
Lizenznummer 114-210/4/87
LSV 1083
Verlagslektor: Helga Fago
Printed in GDR
Satz und Druck:
Messedruck Leipzig, Bereich Borsdorf
III-18-328
Redaktionsschluß: 15. 8. 1987
Bestellnummer: 547 2263
00780

Kleinstrechner-TIPS/Hrsg. von
Hans Kreul u.a. – Leipzig: Fachbuchverl.,
H. 7. – 1. Aufl. – 1987. – 64 S.: 33 Bild.

Herausgeber:

Prof. Dr.-Ing. Hans Kreul, Ingenieurhochschule Zittau, Abt. EDV und Rechentechnik, Theodor-Körner-Allee 16, Zittau, 8800;

Doz. Dr. sc. techn. Thomas Horn oder Doz. Dr.-Ing. Wilhelm Leupold, Informatik-Zentrum des Hochschulwesens an der Technischen Universität Dresden, Mommsenstr. 13, Dresden, 8027

Anschrift des Verlages:
VEB Fachbuchverlag Leipzig
PSF 67
Leipzig – 7031
DDR

7,80

Die Broschürenreihe

KLEINSTRECHNER-TIPS

behandelt

- Tendenzen und Theorien
- Informationen und Ideen
- Programme und Projekte
- Spaß und Spiel

und stellt sich das Ziel

- den Nutzer der Mikrorechentechnik aus allen Bereichen der Volkswirtschaft und dem Bildungswesen bei der Einarbeitung in die Informatik und Computertechnik zu unterstützen
- Entwicklungstendenzen der Informatik und Computertechnik vorzustellen und zur Erweiterung des Grundwissens beizutragen
- Anregungen für den Computereinsatz zu geben und Beispielprogramme für Kleincomputer zu veröffentlichen

um somit einem großen Kreis von Freunden der Informatik und Computertechnik zu helfen, sich moderner Hilfsmittel und Methoden zu bedienen.

ISBN 3-343-00227-5