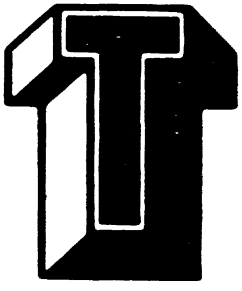
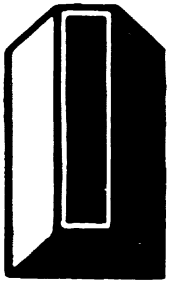
Kleinstrechner



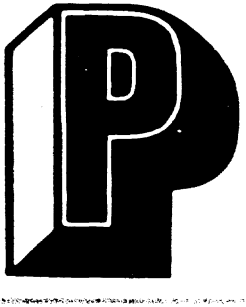
Tendenzen
und Theorien

Elektronischer
Taschenrechner



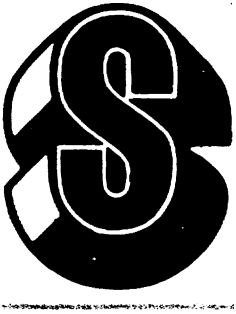
Informationen
und Ideen

Heimcomputer



Programme
und Projekte

Aufzugssteuerung
mit U 880



Spaß
und Spiel

2

Kleinstrechner-TIPS

Heft 2

Mit 21 Bildern

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

VEB Fachbuchverlag Leipzig

Inhalt

Zum Inhalt

Mit dem vorliegenden Heft 2 der Broschürenreihe »Kleinstrechner-TIPS« soll die im ersten Heft begonnene Konzeption fortgesetzt werden: Es sollen *allgemein interessierende Themen aus dem Gebiet der Kleinstrechentechnik* in einer auch dem Laien verständlichen Weise behandelt werden.

Im ersten Artikel von HORN wird über die *Programmierung einer einfachen Aufzugssteuerung* informiert. Der Leser erhält damit einen Einblick in die Anwendungsmöglichkeiten moderner hochintegrierter Schaltkreise zur Steuerung technischer Prozesse. Er erfährt dabei, welche umfangreichen Vorarbeiten geleistet werden müssen, wenn Mikroprozessoren oder -rechner bei der effektiven Lösung technischer Probleme verwendet werden sollen. Das Modell für die Aufzugssteuerung ist stark vereinfacht, so daß es für Praktikumszwecke im Hochschul- bzw. Fachschulstudium durchaus verwendet werden kann.

SCHÖNFELDER setzt mit seinem Artikel die *Anleitung für den Selbstbau eines Home-Computers* fort. Es werden die beiden Hauptanschlußeinheiten des Rechners, die Tastatur und der Bildschirm näher vorgestellt. Damit steht nunmehr bereits ein arbeitsfähiges Gerät zur Verfügung. – Die Bauanleitung wird im folgenden Heft fortgesetzt.

GUTZER stellt ein Programm für das *NIMM-Spiel gegen den K 1003* vor.

In einem Artikel wendet sich KREUL an die Besitzer von elektronischen Taschenrechnern, denen die *Rechenlogik* dieser Arbeitsmittel noch Schwierigkeiten bereitet. Es werden die internen Abläufe der wichtigsten Rechenoperationen bei Rechnern mit algebraischer Logik ohne bzw. mit Hierarchie vorgestellt, und es wird auch auf die vielen noch fremdartig erscheinende Arbeitsweise der Rechner mit Umgekehrter Polnischer Notation eingegangen.

Abschließend stellt KELLER den *Heimcomputer Z 9001* vor und bringt damit unseren Lesern etwas ganz Aktuelles.

Möge auch dieses Heft möglichst vielen Lesern weitere Anregungen zu intensiverer Beschäftigung mit den Geräten der elektronischen Kleinstrechentechnik geben.

Hans Kreul

Zur Programmierung einer einfachen Aufzugssteuerung mit dem Mikroprozessor U 880

Der Mikroprozessor als das bedeutendste Bauelement, das die Mikroelektronik hervorbrachte, führte zu einer revolutionären Weiterentwicklung in der Automatisierungstechnik, elektronischen Gerätetechnik und Konsumgüterindustrie. Er führte neben einer qualitativen Weiterentwicklung und Gebrauchswerterhöhung bekannter Erzeugnisse auch zu völlig neuen Erzeugnissen, wie Arbeitsplatz- und Heimcomputer, Schachcomputer, Fernsehspiele usw. Im vorliegenden Beitrag soll auf ein sehr breites Anwendungsgebiet der Mikroelektronik, die *Automatisierungstechnik*, näher eingegangen werden. Dabei sollen Probleme, Prinzipien und Methoden am Beispiel einer sehr vereinfachten *Aufzugsteuerung* dargestellt werden.

1. Aufgabenstellung

Die Steuerung für einen Kleinlastenaufzug, wie er etwa für Speisen und Getränke in Gaststätten und Hotels verwendet werden könnte, mit zwei Stationen (z.B. Küche und Restaurant), soll auf der Basis eines Mikrorechners realisiert werden. Dieser Aufzug ist auf Grund der vereinfachten Sicherheitsbestimmungen für Kleinlastenaufzüge und der nur notwendigen Außenbedienung in seinem Aufbau nicht kompliziert (Bild 1).

Der Fahrkorb hat zwei stabile Zustände Z1 (unten) und Z2 (oben). Der Zustand Z1 wird durch den Endlagenschalter S1 und der Zustand Z2 durch den Endlagenschalter S2 signalisiert. Jede Schachttür hat einen Türkontakt T_i, der bei geschlossener Schachttür geschlossen ist. Außerdem ist an jeder Schachttür ein Verriegelungsmagnet angebracht, der bei Erregung das Öffnen der Schachttür gestattet. Damit läßt sich realisieren, daß nur diejenige Tür geöffnet werden kann, hinter der der Fahrkorb steht.

An der Steuertafel neben jeder Schachttür werden zwei Anzeigelämpchen L1 und L2 und zwei Taster R1 und R2 vorgesehen. Das Lämpchen L1 zeigt an, daß der Fahrkorb unten ist, und L2, daß er oben ist.

Zur Unterscheidung zwischen dem ausgeschalteten Zustand und dem Zustand »Fahrkorb in Fahrt« können im letzteren beide Lämpchen eingeschaltet werden. Zur Anzeige von Fehlerzuständen, z.B. könnten beim Einschalten des Aufzugs beide Türen geöffnet sein, könnten beide Lämpchen als Blinklicht geschaltet werden. Die Taster R1 und R2 dienen zum Heranholen bzw. Fortschicken des Fahrkorbes, wobei der Taster R1 den Fahrkorb in den Zustand Z1 und der Taster R2 in den Zustand Z2 schicken soll.

Der Antrieb des Aufzugs erfolgt über

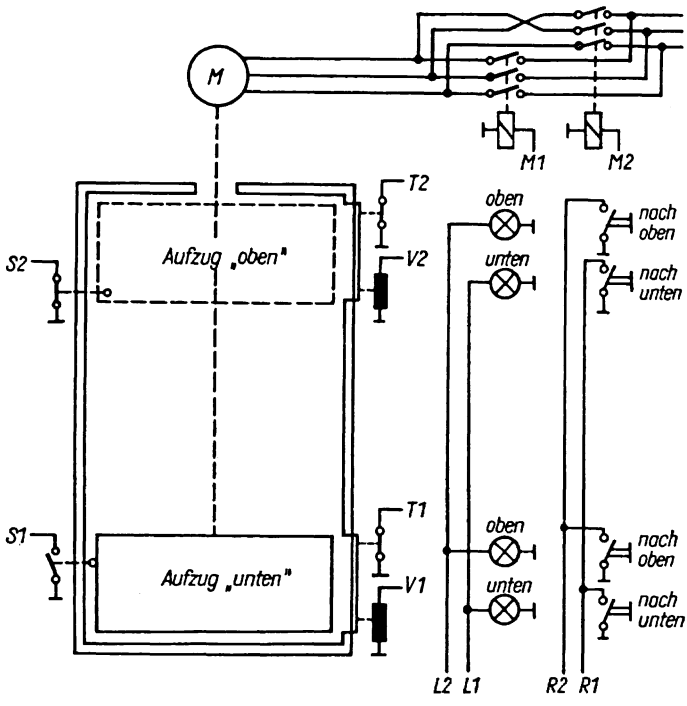


Bild 1. Struktur des Kleinlastenaufzuges mit der Anordnung der Schalt- und Bedienelemente

einen Drehstrommotor, wobei durch Vertauschen von zwei Phasen die Drehrichtung geändert wird. Über den Schütz M1 wird der Motor für die Abwärtsbewegung und über den Schütz M2 für die Aufwärtsbewegung eingeschaltet. Der Antrieb kann aber auch über Gleichstrom-Doppelschlußmotoren erfolgen, wobei die Drehrichtung durch die Polarität der Spannungsquelle festgelegt wird.

Weitere technische Ergänzungen, wie Klingeln auf den Etagen als Aufforderung zum Entladen bzw. Schließen der Tür, eine Schlaffseilvorrichtung zur Seilspannungskontrolle oder Verriegelungskontakte zur Kontrolle der Verriegelungsmagneten sind möglich. Diese technischen Erweiterungen werden aus Gründen der maximalen Vereinfachung des Beispiels nicht berücksichtigt.

2. Entwurf der Steuerelektronik

Die Steuerelektronik soll unter Verwendung eines Mikroprozessors U 880 entworfen werden. Prinzipiell ist auch jeder andere Mikroprozessortyp (U 808, U 881 u. a.) für diese Schaltung geeignet. Da aber schon in früheren Beiträgen auf den Mikroprozessor U 880 Bezug genommen wurde, soll er auch hier verwendet werden.

Neben einem *Prozessorschaltkreis* (CPU) U 880 ist ein *Speicherschaltkreis für das Steuerprogramm* und ein *Schaltkreis für die parallele Ein- bzw. Ausgabe* (PIO) U 855 erforderlich. Als Speicherschaltkreis wird zweckmäßigerweise ein löschbarer Festwertspeicher (EPROM) U 555 verwendet, da er für das Testen und für die Weiterentwicklung des Steuerprogramms auf Grund seiner mehrfachen Verwendbarkeit am günstigsten geeignet ist. Der Schaltkreis

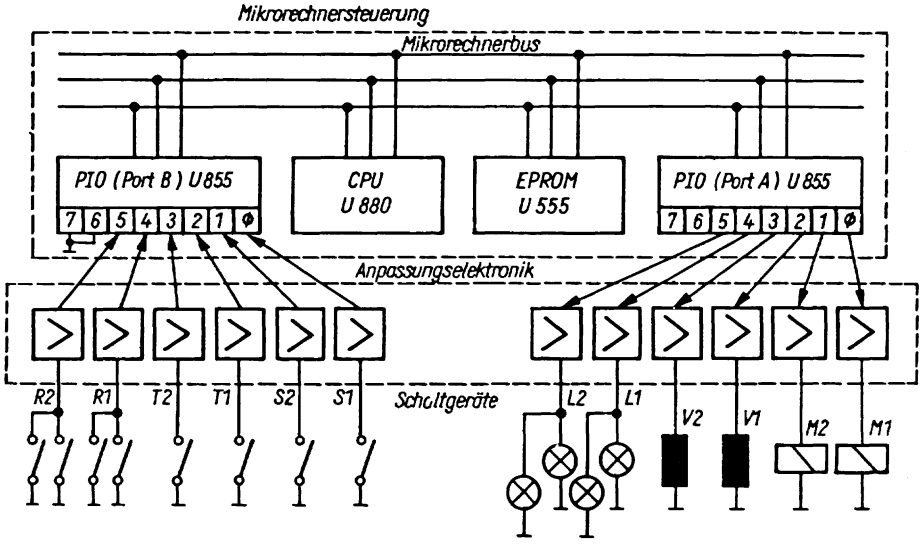


Bild 2. Struktur der Steuerelektronik mit dem Mikroprozessor U 880

PIO hat zwei Ports, A und B, die je eine 8 Bit parallele Ein- bzw. Ausgabe realisieren können. Das Port A soll für die Ausgabe der Steuersignale M1, M2, V1, V2, L1 und L2 verwendet werden. Das Port B wird entsprechend für die Eingabe der Eingangssignale S1, S2, T1, T2, R1 und R2 genutzt. Diese drei Schaltkreise bilden zusammen den Mikrorechner für die Aufzugssteuerung, indem sie über den *Mikrorechnerbus* miteinander verbunden werden. Der Mikrorechnerbus besteht aus 8 Datenleitungen, 16 Adreßleitungen (hier wären 10 Adreßleitungen ausreichend, da auf Grund der geringen Programmgröße nur 1 Schaltkreis U 555 mit einer Kapazität von 1 kByte erforderlich ist) und 6 Steuerleitungen.

Der prinzipielle Aufbau ist in Bild 2 dargestellt, wobei aus zeichnungstechnischen Gründen die beiden Ports der PIO getrennt dargestellt werden.

Neben der eigentlichen Mikrorechnersteuerung ist noch eine Anpassungselektronik erforderlich, die die Mikrorechnersignale an die äußeren Signale anpaßt. Für die Eingabesignale ist die Anpassung nicht kompliziert. Hier kann eine Widerstandsschaltung mit einem Kondensator zur Entprellung der Kontakte im einfachsten Fall verwendet werden. Wenn mit größeren induzierten Fremdspannungen zu rechnen ist, sind auch Opto-Koppler zur galvanischen Entkopplung und andere Schaltungen möglich. Für die Ausgangssignale ist unbedingt eine Verstärkung der Mikrorechnersignale über Transistorverstärkerstufen und Relais bzw. Thyristoren zur Steuerung der Leistungsschalter (Schütze), Magnete bzw. Anzeigelämpchen notwendig.

3. Entwurf des Algorithmus für die Aufzugssteuerung

In der Automatisierungstechnik werden die Steuerungsabläufe in der Regel auf der Grundlage von Modellen aus der Automatentheorie beschrieben. Das einfachste Automatenmodell, das für

die Beschreibung der meisten Steuerungsabläufe genügt, ist der endliche Automat, bei dem

$X = \{x_0, x_1, x_2, \ldots\}$
– die Menge der Eingangssignale oder das Eingabealphabet –,

$Y = \{y_0, y_1, y_2, \ldots\}$
– die Menge der Ausgangssignale oder das Ausgabealphabet – und

$Z = \{z_0, z_1, z_2, \ldots\}$
– die Menge der internen Zustände des Automaten –

jeweils **endliche** Mengen sind.
Es werden zwei Typen des endlichen Automaten unterschieden: der MOORE-Automat und der MEALY-Automat. Jeder endliche Automat läßt sich prinzipiell als MOORE- und als MEALY-Automat darstellen. Im weiteren sei hier nur der MOORE-Automat betrachtet, da er für das Beispiel der Aufzugssteuerung besonders einfach und anschaulich ist.
Der MOORE-Automat ist ein Quintupel

$$\mathfrak{M} = (X, Y, Z, \delta, \mu),$$

wobei

$\delta\colon\ Z \times X \to Z$
die Zustandsüberführungsfunktion und

$\mu\colon\ Z \to Y$
die Markierungsfunktion[1] sind.

Die Funktionen δ und μ können in der Form einer Automatentafel oder eines Automatengraphen dargestellt werden. Besonders anschaulich und für die Programmierung gut geeignet ist der Automatengraph, der aus Zuständen (Knoten) und Übergängen (Kanten) aufgebaut wird. Jedem Zustand wird beim MOORE-Automaten ein Ausgangssignal zugeordnet. Die Übergänge werden durch die Eingangssignale definiert. Graphisch wird dieser Sachverhalt wie folgt dargestellt:

Auf das Beispiel der Aufzugssteuerung angewendet, ergibt sich der in Bild 3 dargestellte Graph eines MOORE-Automaten, wobei

$X = \{S1, S2, T1, T2, R1, R2\}$

$Y = \{M1, M2, V1, V2, L1, L2\}$ und

$Z = \{Z1, Z12, Z2, Z21\}$

sind.
Die Bedeutung der Zustandskodierungen ergibt sich aus dem Zustand des Fahrkorbes:

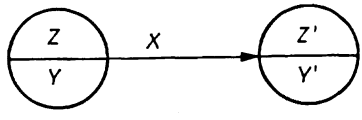
Z1 – Fahrkorb befindet sich unten,

Z12 – Fahrkorb bewegt sich von unten nach oben,

Z2 – Fahrkorb befindet sich oben,

Z21 – Fahrkorb bewegt sich von oben nach unten.

Im Zustand Z1 werden die Lämpchen L1 eingeschaltet und der Verriegelungsmagnet V1 erregt. Wenn die Tür 1 geschlossen ist (T1 = 1) und ein Taster R2 gedrückt wird (R2 = 1), geht der Automat in den Zustand Z12 über, anderenfalls verbleibt der Automat im Zustand Z1.
Im Zustand Z12 werden der Motor (M2 = 1) und die Lämpchen L1 und L2 eingeschaltet. Wenn der Kontakt S2 öffnet, geht der Automat in den Zustand Z2 über, anderenfalls verbleibt der Automat im Zustand Z12.
Analog verhält sich der Automat in den Zuständen Z2 und Z21.



[1] Beim MEALY-Automaten ist die Markierungsfunktion zusätzlich von den Eingangsvariablen abhängig: $\mu\colon Z \times X \to Y$.

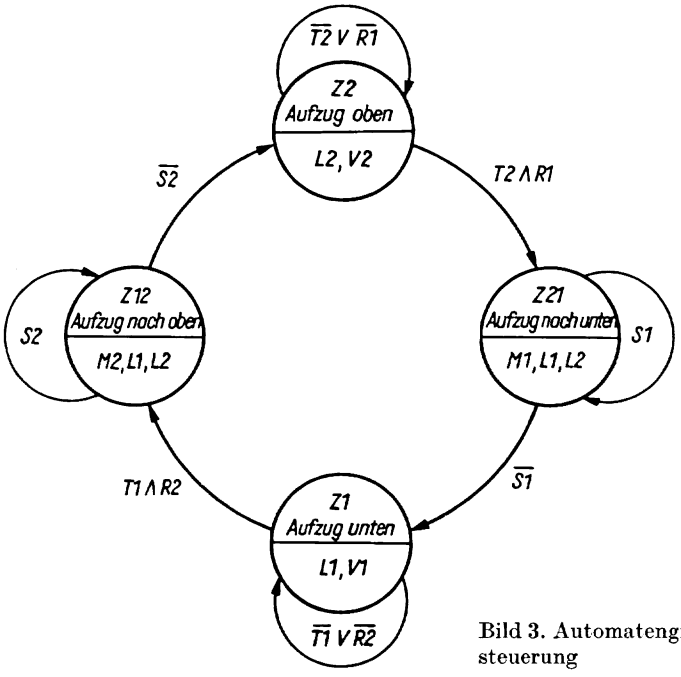


Bild 3. Automatengraph für die Aufzugssteuerung

4. Vorbereitung der programmtechnischen Umsetzung des Algorithmus für die Aufzugssteuerung

Bei der programmtechnischen Realisierung der Aufzugssteuerung ist zu beachten, daß der Automatengraph nur die normale Betriebsweise des Aufzugs beschreibt. Für die Inbetriebnahme des Aufzuges mit dem Einschalten des Mikrorechners ist ein sogenanntes Anlaufprogramm erforderlich, das den Zustand des Aufzuges »austestet« und in den entsprechenden Zustand des Automatengraphen verzweigt:

$\overline{S1} \wedge T2$: Fahrkorb befindet sich unten, und Tür 2 ist geschlossen.

Folge: Übergang in den Zustand Z1

$\overline{S2} \wedge T1$: Fahrkorb befindet sich oben, und Tür 1 ist geschlossen.

Folge: Übergang in den Zustand Z2

$T1 \wedge T2$: Beide Türen sind geschlossen, aber der Fahrkorb befindet sich weder oben noch unten.

Folge: Übergang in den Zustand Z21.

Bei allen anderen Eingangssignalen befindet sich der Aufzug in einem nichtdefinierten Anfangszustand, und eine Inbetriebnahme ist unmöglich. Die Lämpchen L1 und L2 werden durch zeitverzögertes Ein- und Ausschalten auf Blinklicht geschaltet.

Außerdem ist zu berücksichtigen, daß vor der ersten E/A-Operation der E/A-Schaltkreis PIO programmiert werden muß. In diesem Beispiel ist die Programmierung der PIO besonders einfach, da dem Port A nur die Betriebsart »Ausgabe« und dem Port B die Betriebsart »Eingabe« mitgeteilt werden muß.

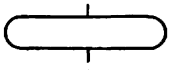
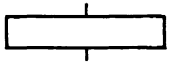
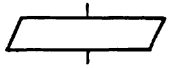
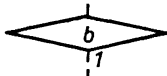
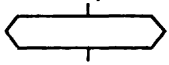
Nr.	Symbol	Bedeutung
1		Anfang, Ende, Halt, Unterbrechung
2		Operation (allgemein)
3		E/A - Operation
4		Verzweigung (wenn die Bedingung b erfüllt ist zu 1, andernfalls zu Φ)
5		Zähler, Setzen einer Bedingung

Tabelle 1. Symbolik für die Darstellung der Programmablaufpläne

Als Programmierunterstützung wird oft die Darstellung des zu programmierenden Problems in graphischer Form als *Programmablaufplan* gewählt. Der Programmablaufplan (PAP) ist eine Form eines gerichteten Graphen, die relativ gut die Dynamik der Programmabarbeitung widerspiegelt. Die Darstellung der PAPs kann nach der Linienmethode und der Kästchenmethode erfolgen. Die Kästchenmethode sei hier trotz des größeren Zeichenaufwandes wegen ihrer besseren Anschaulichkeit gewählt. Die wichtigsten Symbole sind in Tabelle 1 zusammengefaßt.
Bild 4 zeigt den kompletten PAP für die Aufzugssteuerung, wobei im linken Teil das Anlaufprogramm und im rechten Teil der Automatengraph dargestellt sind. Jeder Zustand wird durch die Ausgabe der Steuersignale, die Eingabe der Eingangsvariablen und die Analyse der Eingangsvariablen mit anschließender Verzweigung in den entsprechenden Zielzustand verkörpert.

5. Programmtechnische Realisierung der Aufzugssteuerung

Bei der programmtechnischen Realisierung sind in erster Linie *Kodierungsprobleme* zu lösen, wie die Kodierung der E/A-Port-Adressen und der E/A-Signale.
Die E/A-Port-Adressen der PIO ergeben sich aus den verwendeten Adreßleitungen für die Adressierung der PIO. Es wird die Verwendung folgender Adreßleitungen vorgeschlagen:

A_0 – Auswahl des Ports der PIO (Ø – Port A, 1 – Port B);
A_1 – Auswahl der Betriebsweise (Ø – Datentransfer, 1 – Programmierung);
A_2 – Aktivieren ($\overline{CE}$) des Schaltkreises (Ø – aktiv).

Alle anderen Adreßleitungen werden für den E/A-Transfer nicht benötigt und können bei E/A-Operationen deshalb beliebig belegt sein. Dadurch ergeben sich die im Definitionsteil des Programms (Bild 5) verwendeten Portadressen.
Die Kodierung der E/A-Signale ergibt sich aus der Bitbelegung der E/A-Ports und ist ebenfalls im Definitionsteil des Programms enthalten.
Die Kodierung des Programms soll wegen der besseren Anschaulichkeit in der *Assemblersprache SYPS-1520* und nicht im internen Maschinenkode des Mikroprozessors (Bitmuster) erfolgen.

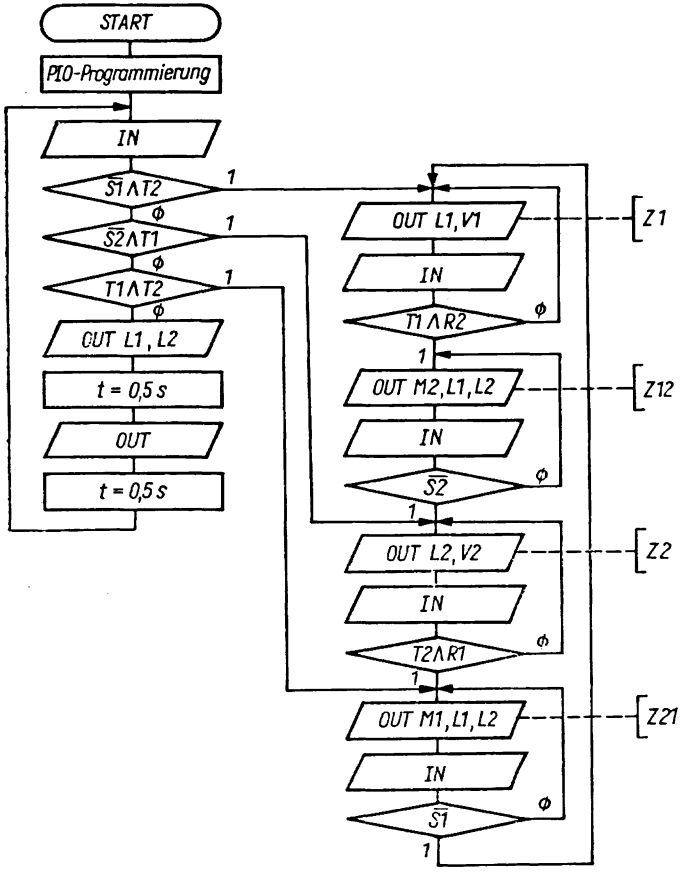


Bild 4. Programmablaufplan für die Aufzugssteuerung

Dabei sollen die Vorteile einer symbolischen Kodierung in einer Assemblersprache genutzt werden. Die Umsetzung des Assemblerprogramms erfordert aber ein Übersetzerprogramm (Assembler). Für das in Bild 5 dargestellte Programm wurde der Assembler ASSA des Mikrorechnerentwicklungssystems MRES 20 (A 5601) vom VEB Kombinat Robotron genutzt. Da das Programm aber nicht sehr groß ist, ist auch noch eine manuelle Umsetzung des Assemblerprogramms in die Maschinensprache möglich.

Für das Programm wurden folgende *Assembler-Pseudoanweisungen* verwendet:

PN name

Festlegung eines Programmnamens, erste Anweisung eines Assemblerprogramms;

symbol: EQU wert

Einem symbolischen Namen symbol wird ein Wert zugewiesen. Damit können im Programm zur Erhöhung der Lesbarkeit anstelle von kodierten Zahlen symbolische Namen verwendet werden.

```
                        ARS. ASSEMBLER  MEOS 1521 V3.2/4
    LOC        OBJ.CODE     STMT      SOURCE PROGRAM
===HEX=DEC=================================================================================19/10/83==001=====
                             00001          PN   LA -- STEUERUNG FUER EINEN KLEINLASTENAUFZUG
                             00002  ;
                             00003  ;  DEFINITIONEN FUER ANPASSEN UND BITBELEGUNGEN
                             00004  ;
                             00005  DA:    EQU  00H                    ;PIO - DATENPORT A (AUSGABE)
                             00006  DB:    EQU  01H                    ;PIO - DATENPORT B (EINGABE)
                             00007  PA:    EQU  02H                    ;PIO - PROGRAMMIERUNG DES PORT A
                             00008  PB:    EQU  03H                    ;PIO - PROGRAMMIERUNG DES PORT B
                             00009  ;
                             00010  ;  BITBELEGUNGEN FUER DAS AUSGABEPORT A
                             00011  ;
                             00012  M1:    EQU  01H                    ;MOTOR "EIN" - FAHRT NACH UNTEN
                             00013  M2:    EQU  02H                    ;MOTOR "EIN" - FAHRT NACH OBEN
                             00014  U1:    EQU  04H                    ;VERRIEGELUNGSMAGNET UNTEN
                             00015  U2:    EQU  08H                    ;VERRIEGELUNGSMAGNET OBEN
                             00016  L1:    EQU  10H                    ;ANZEIGE "FAHRKORB UNTEN"
                             00017  L2:    EQU  20H                    ;ANZEIGE "FAHRKORB OBEN"
                             00018  ;
                             00019  ;  BITBELEGUNGEN FUER DAS EINGABEPORT B
                             00020  ;
                             00021  S1:    EQU  01H                    ;ENDLAGENSCHALTER "FAHRKORB UNTEN"
                             00022  S2:    EQU  02H                    ;ENDLAGENSCHALTER "FAHRKORB OBEN"
                             00023  T1:    EQU  04H                    ;UNTERE TUER GESCHLOSSEN
                             00024  T2:    EQU  08H                    ;OBERE TUER GESCHLOSSEN
                             00025  R1:    EQU  10H                    ;FAHRTRICHTUNGSWAHL "NACH UNTEN"
                             00026  R2:    EQU  20H                    ;FAHRTRICHTUNGSWAHL "NACH OBEN"
                             00027  ;
                             00028  ;  ANLAUFPROGRAMM FUER DIE AUFZUGSSTEUERUNG
                             00029  ;
0000 00000  3E 0F            00030  START: LD   A,0FH                  ;BETRIEBSART "AUSGABE"
0002 00002  D3 02            00031         OUT  PA                     ;PROGRAMMIERUNG PORT A
0004 00004  3E 4F            00032         LD   A,4FH                  ;BETRIEBSART "EINGABE"
0006 00006  D3 03            00033         OUT  PB                     ;PROGRAMMIERUNG PORT B
0008 00008  DB 01            00034  M00:  IN   DB                     ;EINGABE DES AKTUELLEN AUFZUGSZUSTANDES
000A 00010  47               00035         LD   B,A                    ;RETTEN DER EINGABESIGNALE
000B 00011  E6 09            00036         AND  S1+T2                  ;MASKE S1+T2
000D 00013  EE 08            00037         XOR  T2                     ;VERGLEICH T2
000F 00015  CA 47 00         00038         JPZ  Z1                     ;---> Z1 (AUFZUG UNTEN)
0012 00018  78               00039         LD   A,B
```

Bild 5. Assemblerprotokoll des Steuerprogramms für die Aufzugssteuerung

```
0013 00019  E6 06        00040          AND  S2+T1              ;MASKE S2+T1
0015 00021  EE 04        00041          XOR  T1                 ;VERGLEICH T1
0017 00023  CA 61 00     00042          JPZ  Z2                 ;---> Z2 (AUFZUG OBEN)
001A 00026  78           00043          LD   A,B
001B 00027  E6 0C        00044          AND  T1+T2              ;MASKE T1+T2
001D 00029  EE 0C        00045          XOR  T1+T2              ;VERGLEICH T1+T2
001F 00031  CA 6E 00     00046          JPZ  Z21                ;---> Z21 (AUFZUG NACH UNTEN)
0022 00034  3E 30        00047          LD   A,L1+L2            ;LAMPEN L1+L2 "EIN"
0024 00036  D3 00        00048          OUT  DA                 ;AUSGABE PORT A
0026 00038  0E FF        00049          LD   C,255              ;ZEITSCHLEIFE T=0,47S
0028 00040  06 FF        00050  M01:    LD   B,255
002A 00042  78           00051  M02:    LD   A,B
002B 00043  05           00052          DEC  B
002C 00044  C2 2A 00     00053          JPNZ M02
002F 00047  0D           00054          DEC  C
0030 00048  C2 28 00     00055          JPNZ M01
0033 00051  3E 00        00056          LD   A,0                ;LAMPEN L1+L2 "AUS"
0035 00053  D3 00        00057          OUT  DA                 ;AUSGABE PORT A
0037 00055  0E FF        00058          LD   C,255              ;ZEITSCHLEIFE T=0,47S
0039 00057  06 FF        00059  M03:    LD   B,255
003B 00059  78           00060  M04:    LD   A,B
003C 00060  05           00061          DEC  B
003D 00061  C2 3B 00     00062          JPNZ M04
0040 00064  0D           00063          DEC  C
0041 00065  C2 39 00     00064          JPNZ M03
0044 00068  C3 08 00     00065          JMP  M00                ;---> PRUEFEN DES ZUSTANDES
                         00066  ;
                         00067  ;  AUTOMATENGRAPH DER AUFZUGSSTEUERUNG
                         00068  ;
0047 00071  3E 14        00069  Z1:     LD   A,L1+V1            ;LAMPE L1 UND VERRIEGELUNGSMAGNET V1
0049 00073  D3 00        00070          OUT  DA                 ;AUSGABE
004B 00075  DB 01        00071          IN   DB                 ;EINGABE DES AKTUELLEN AUFZUGSZUSTANDES
004D 00077  E6 24        00072          AND  T1+R2              ;MASKE T1+R2
004F 00079  EE 24        00073          XOR  T1+R2              ;VERGLEICH T1+R2
0051 00081  C2 47 00     00074          JPNZ Z1                 ;---> RUECKSPRUNG NACH Z1
                         00075  ;
0054 00084  3E 32        00076  Z12:    LD   A,M2+L1+L2         ;LAMPE L1+L2, AUFZUG NACH OBEN
0056 00086  D3 00        00077          OUT  DA                 ;AUSGABE
0058 00088  DB 01        00078          IN   DB                 ;EINGABE DES AKTUELLEN AUFZUGZUSTANDES
005A 00090  E6 02        00079          AND  S2                 ;MASKE S2
005C 00092  EE 00        00080          XOR  0                  ;VERGLEICH 0
005E 00094  C2 54 00     00081          JPNZ Z12                ;---> RUECKSPRUNG NACH Z12:
                         00082
```

```
0061 00097 3E 28          00083 22:   LD    A,L2+W2      ;LAMPE L2 UND VERRIEGELUNGSMAGNET V2
0063 00099 D3 00          00084       OUT   DA           ;AUSGABE
0065 00101 DB 01          00085       IN    DB           ;EINGABE DES AKTUELLEN AUFZUGSZUSTANDES
0067 00103 E6 18          00086       AND   T2+R1        ;MASKE T2+R1
0069 00105 EE 18          00087       XOR   T2+R1        ;VERGLEICH T2+R1
006B 00107 C2 61 00       00088       JPNZ  Z2           ;---> RUECKSPRUNG NACH Z2
                          00089       ;
006E 00110 3E 31          00090 Z21:  LD    A,M1+L1+L2   ;LAMPE L1+L2, AUFZUG NACH UNTEN
0070 00112 D3 00          00091       OUT   DA           ;AUSGABE
0072 00114 DB 01          00092       IN    DB           ;EINGABE DES AKTUELLEN AUFZUGSZUSTANDES
0074 00116 E6 01          00093       AND   S1           ;MASKE S1
0076 00118 EE 00          00094       XOR   0            ;VERGLEICH 0
0078 00120 C2 6E 00       00095       JPNZ  Z21          ;---> RUECKSPRUNG NACH Z21
007B 00123 C3 47 00       00096       JMP   Z1           ;---> ZUSTAND Z1
                          00097       ;
007E 00126                00098       END
                          PROGRAM CONTAINS 00000 ERRORS
```

Bild 5 (Fortsetzung)

END

Ende des Assemblerprogramms.

Zur Programmierung der Aufzugssteuerung werden in diesem Beispiel von den etwa 200 Befehlen des Mikroprozessors U 880 nur 10 Befehle verwendet. Die Kodierung dieser 10 Befehle ist in Tabelle 2 enthalten und erklärt.

Auf zwei programmtechnische Probleme sei hier noch näher eingegangen:

a) *Programmierung von Verzweigungen*, z.B.

Bei der Voraussetzung, daß die zu analysierenden Eingangssignale im Register A zur Verfügung stehen (der IN-Befehl liest die Eingangssignale in das Register A ein), werden die nicht benötigten Bitpositionen, die nicht analysiert werden, mit einem AND-Befehl (Konjunktion) und einer entsprechenden Maske gelöscht. Die Maske wird aus den Namen der beteiligten Variablen gebildet, unabhängig davon, ob sie negiert sind oder nicht. Anschließend kann das Ergebnis im Register A mit einer Maske, die aus den nichtnegierten Variablen gebildet wird, unter Verwendung des XOR-Befehls (exklusives Oder, Äquivalenz) verglichen werden. Bei Identität (1 – Bedingung erfüllt) kann mit dem folgenden JPZ-Befehl zu einer Marke verzweigt werden. Anderenfalls (Ø – Bedingung nicht erfüllt) wird der nächste Befehl nach dem JPZ-Befehl vom Prozessor abgearbeitet.

Beispiel:

AND S2 + T1:

Ausblenden nichtbenötigter Variablen

Mnem. Befehl	Flags C Z P/V S N H	Masch.kode	Zykl	Takt	Bedeutung
LD r1,r2	- - - - - -	01dddsss	1/2	4/7	Ladebefehl r1:=r2 (r1≠r2)
LD r1,n	- - - - - -	00ddd110 nnnnnnnn	2/3	7/10	Ladebefehl r1:=n
AND n	0 * P * 1 *	11100110 nnnnnnnn	2	7	Logisches UND A:=A∧n
XOR n	0 * P * 0 *	11101110 nnnnnnnn	2	7	Exklusives ODER A:=A⊕n
DEC r1	- * V * 1 *	00ddd101	1/3	4/11	Dekrementbefehl r1:=r1-1
JMP nn	- - - - - -	11000011 nnnnnnnn nnnnnnnn	3	10	Unbedingter Sprung PC:=nn
JPNZ nn	- - - - - -	11000010 nnnnnnnn nnnnnnnn	3	10	Bedingter Sprung Z=0: PC:=nn
JPZ nn	- - - - - -	11001010 nnnnnnnn nnnnnnnn	3	10	Bedingter Sprung Z=1: PC:=nn
IN n	- - - - - -	11011011 nnnnnnnn	3	10	Eingabebefehl A:=(n)
OUT n	- - - - - -	11010011 nnnnnnnn	3	11	Ausgabebefehl (n):=A

Abkuerzungen

Mnemonischer Befehl

r1 - Zieloperand (r1 = A,B,C,D,E,H,L,M)
r2 - Quelloperand (r2 = A,B,C,D,E,H,L,M)
r - Quell- oder Zieloperand (r = A,B,C,D,E,H,L)
n - Direktwert (8 Bit)
nn - Direktwert oder symbolische Adresse (16 Bit)

Standardnamen

A - Akkumulator
B,C,D,E,H,L - allgemeine Register

Bedeutung der Flags

C - Carry (Uebertrag)
Z - Zero (Ergebnis ist null)
S - Sign (Vorzeichen ist negativ)
P/V - Parity (Paritaet des Ergebnisses bei logischen Operationen)
Overflow (Ueberlauf bei arithmetischen Operationen)
H - Half-carry (Uebertrag zwischen Bit 3 und Bit 4)

Tabelle 2. Auszug aus der Befehlsliste des U 880

```
N    -  Addition/Subtraktion
        N=1, wenn vorhergegangene Operation eine Subtraktion war


Kennzeichnung der Flags
-----------------------

*   Flag wird entsprechend des Ergebnisses gesetzt
1   Flag gesetzt
0   Flag geloescht
P   Paritaet des Ergebnisses
V   Flag enthaelt Uebertrag einer Operation
X   Flag unbestimmt
-   Flag nicht beeinflusst


Maschinencode
-------------

sss  -  Kodierung des Quelloperanden r2
ddd  -  Kodierung des Zieloperanden r1
rrr  -  Kodierung des Quell- oder Zieloperanden r
nnnnnnnn  -  Kodierung des Direktwertes n
nnnnnnnn  -  Kodierung des Direktwertes nn
nnnnnnnn     (1.Byte niederw. Adressteil, 2.Byte hoeherw. Adressteil)


Kodierung der Register und Doppelregister
-----------------------------------------

sss, ddd :      B  -  000        rrr :      B  -  000
(bzw. r2,r1)    C  -  001        (bzw. r)   C  -  001
                D  -  010                   D  -  010
                E  -  011                   E  -  011
                H  -  100                   H  -  100
                L  -  101                   L  -  101
                M  -  110                   A  -  111
                A  -  111


Sonstige Abkuerzungen
---------------------

(  )   -  Inhalt eines Registers bzw. Speicherplatzes
PC     -  Programcounter (Befehlszaehler)
```

Tabelle 2 (Fortsetzung)

XOR T1:
Vergleich mit nichtnegierten Variablen
JPZ Z2
Bedingung erfüllt → Z2

b) *Programmierung von Zeitschleifen*
Zeitschleifen werden in der Regel durch Herunterzählen von Zählern gebildet. Dafür wird der DEC-Befehl (Dekrement, Erniedrigung um eins) verwendet. Im Beispiel werden zwei ineinander geschachtelte Zählschleifen verwendet, so daß die inneren Befehle (LD, DEC, JPNZ) (255×255)mal abgearbeitet werden. Die Zeitdauer der Zeitschleife hängt von der Taktzeit des Mikroprozessors t_T, der Taktanzahl der Befehle in der inneren Schleife n und der Anzahl der Schleifendurchläufe N ab:

$$t = t_T \cdot n \cdot N,$$

wobei

$t_T = 0{,}4\ \mu s$ für den 2,5 MHz-Schaltkreis

bzw.

$t_T = 1\,\mu s$ für den 1-MHz-Schaltkreis ist.

Die Taktanzahl n läßt sich aus Tabelle 2 errechnen und ist hier 18. (Der LD-Befehl wurde nur zur Erhöhung der Taktanzahl eingefügt.) Damit ergibt sich für die programmierten Zeitschleifen eine Wartezeit von etwa 0,47 s.

6. Schlußbemerkungen

Das beschriebene Programm stellt eine *einfache Grundvariante für einen Laborversuch* dar. Für den praktischen Einsatz müßte es ergänzt werden mit den Analysen der Ruhestromkreise in den einzelnen Automatenzuständen, die für die Arbeitsweise des Automaten zwar unwichtig sind, aber für die Erkennung von Fehlerzuständen bzw. Gefahrenquellen eine praktische Bedeutung haben. Außerdem sind die eingangs erwähnten und weitere technische Ergänzungen möglich und notwendig, wofür die nachfolgend aufgeführte Fachliteratur zu Rate gezogen werden sollte.

Literatur

[1] Dienhold, V.; Horn, T.: Programmierung von Mikrorechnern. 1. Lehrbrief. – Zwickau: ZLO, 1981. – Sig.-Nr. 02 1570 010

[2] Dienhold, V.; Horn, T.: Arbeitsunterlagen zur Programmierung des Mikrorechners K 1520. – Dresden: Ingenieurhochschule, 1979; 1980; 1981

[3] Horn, T.; u. a.: Arbeitsunterlagen zur Anwendung des Betriebssystems MEOS 1521 auf dem Mikrorechnerentwicklungssystem MRES 20. – Dresden: Ingenieurhochschule, 1982

[4] Roth, M.: Mikroprozessoren: Wesen, Technologie, Weiterentwicklung, Aufbau, Programmierung, Anwendung. – Ilmenau: Technische Hochschule, KdT-Hochschulsektion, 1979

[5] Schwarz, W.; Meyer, G.; Eckardt, D.: Mikrorechner: Wirkungsweise – Programmierung – Applikation. – Berlin: Verlag Technik, 1980

[6] Thiemann, H.: Aufzüge – Betrieb, Wartung, Revision. – Berlin: Verlag Technik, 1979

Autor:

Dr. sc. techn. Thomas Horn
DDR 8036 Dresden
Prohliser Straße 2b

Wissenschaftlicher Mitarbeiter
an der Sektion Informationsverarbeitung
der Ingenieurhochschule Dresden

Rechentechnische Begriffe für den Laien erklärt

Bit

Der Begriff Bit wird mit mehreren unterschiedlichen Deutungen belegt:

1. Als Abkürzung für binary digit (engl.: Binärziffer, Dualziffer)
2. Als einzelnes Zeichen einer binären Zahl (Dualzahl). Das Dualzahlsystem ist ein Zahlensystem, in dem nur die beiden Ziffern 0 und 1 (zuweilen auch mit O und L bezeichnet) benötigt werden.
3. Als kleinste Einheit der Speicherkapazität eines Speicherbausteines. Diese kleinste Speichereinheit kann nur mit zwei eindeutig voneinander unterscheidbaren Werten belegt werden, wobei normalerweise die Belegungen 0 und 1 üblich sind. Es wären jedoch auch solche Belegungen wie JA und NEIN, WAHR und FALSCH u. dergl. denkbar.

Homecomputer – ein neues Gebiet für den Amateur (Teil 2)

Das Ziel dieses Teils ist es, die *Anschlußeinheiten für die zwei Hauptperipheriegeräte, Tastatur und Bildschirm,* vorzustellen.

TASTATUR

Um mit einem Computer ordentlich arbeiten zu können und seine Leistungsfähigkeit zu nutzen, ist als Eingabegerät eine Alpha- (bzw. Schreibmaschinen-)Tastatur erforderlich. Der Zeichensatz ist dabei größer als bei einer Schreibmaschine und sollte dem ASCII-Satz entsprechen [1]. Ergänzt wird dieser Zeichenvorrat durch eine Anzahl von *Steuerzeichen,* wie z.B. *Clear* (Bildschirm löschen), die *Cursorbewegungen* ($\rightarrow \downarrow \leftarrow \uparrow$), *Insert* (Einfügen von Zeichen) und *Delete* (Streichen von Zeichen).
Es ergibt sich damit eine Tastatur mit über 64 Tastenfunktionen, was sich für einen Amateur aus ökonomischen Gründen kaum noch mit mechanischen Tasten realisieren läßt. Aus diesem Grund wurde eine Tastatur nach dem Sensorprinzip entwickelt, die sehr preiswert ist und kaum mechanische Bauteile enthält.
Um den elektrischen Aufwand klein zu halten, wurden möglichst viele Funktionen in die Software verlagert. Die Tastatur selbst besteht nur noch aus den elektrisch zu einer 8×8-Matrix zusammengeschalteten Tasten und zwei Codierschaltungen für Spalte und Zeile (Bild 1). Als Codierschaltkreise wurden die Sensor-IC's U 710 mit nachfolgenden Pegelwandlern eingesetzt. Dabei arbeiten immer 2 Schaltkreise so zusammen, daß, wenn keine Taste bedient ist, die Ausgangscodierung 8 anliegt und bei Tastenbedienung die Nummer der bedienten Spalte oder Zeile (also $0 \cdots 7$) [2]. Für den Rechner ergeben sich dann zwei Steuersignale und die in 6 Bit codierte Nummer der bedienten Taste. Erkennt der Rechner diesen Zustand, so versucht er die Coderschaltkreise wieder auf den Zustand **'8'** zu bringen, was ihm erst nach Loslassen der Taste gelingt, da die Resetschaltung bis dahin gesperrt bleibt. Auf diese Weise wird die Tastatur zusätzlich zu den U 710 nochmals entprellt.

Anschlußsteuerung der Tastatur

Die Anschlußsteuerung (Bild 2) ist ebenfalls sehr einfach gehalten. Sie besteht aus einem Eingabetor und drei Ausgabetoren, wovon allerdings nur ein Tor Daten bringt. Bei den zwei anderen Toren wird nur die Toradresse als Steuersignal benutzt. Das betrifft das Resetsignal und die Ansteuerung eines akustischen Signales zur Bedienkontrolle (Bild 3).
Der Tastencode wird über ein normales

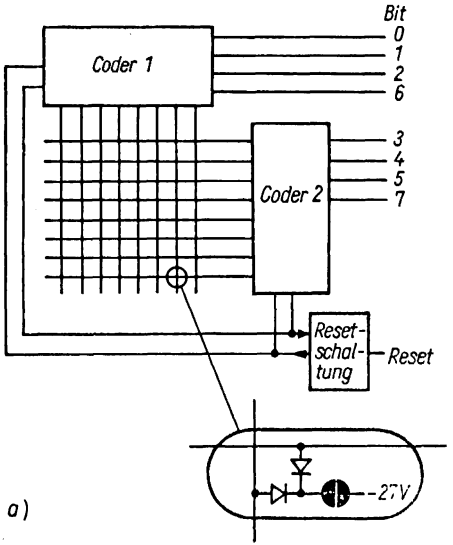
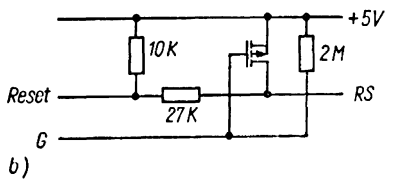
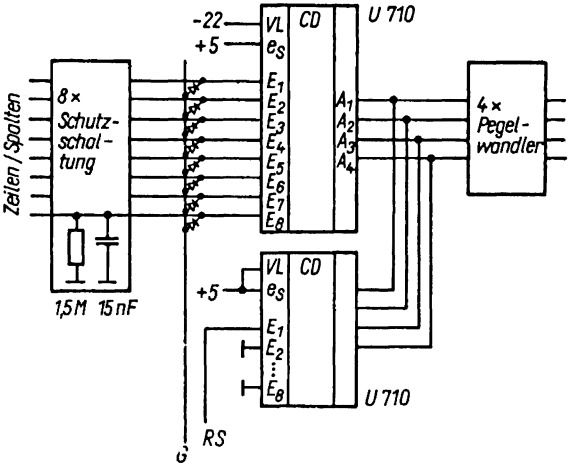


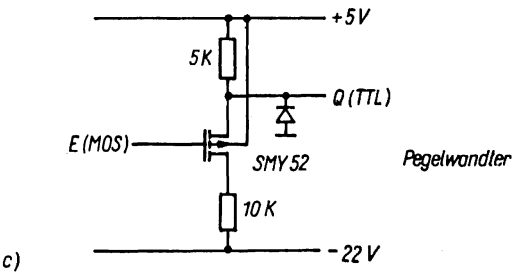
Bild 1a
Blockschaltbild der Tastatur

Bild 1b
Tastatur – Resetsperre

Bild 1c
Tastatur – Schaltung eines Coders







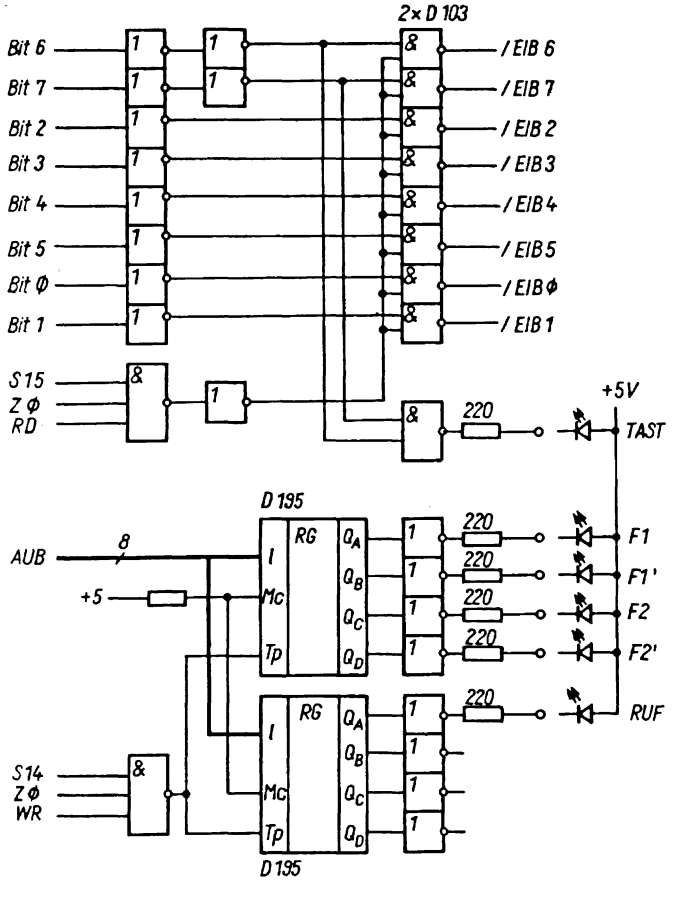


Bild 2
Anschlußsteuerung Tastatur

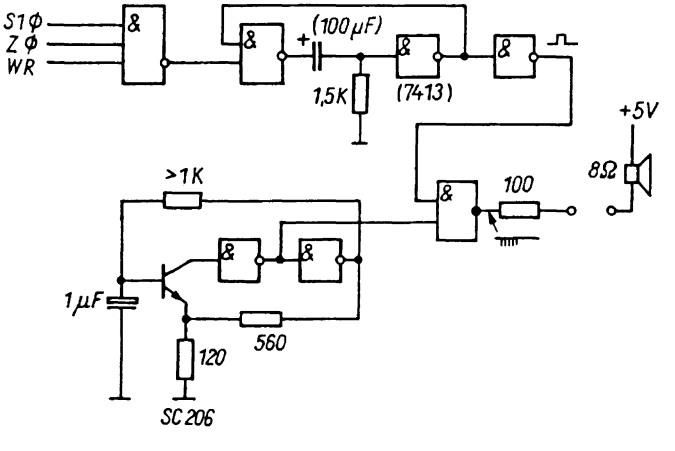


Bild 3
Akustisches Signal

Tor eingelesen. Die Negation ist durch die Pegelwandler in der Tastatur erforderlich. Durch die doppelte Negation der Steuersignale sind diese ebenfalls high-aktiv, wodurch der Rechner bei ungenutzter Tastatur eine NULL liest. Durch Verknüpfung beider Steuersignale läßt sich eine Kontrolle der Tasten erreichen. Das ist notwendig, da die Schaltung die Tastenwerte speichert, wenn die Tastatur nicht vom Rechner abgefragt wird. Aus diesem Kontrollsignal ließe sich auch ein Interrupt ableiten. Das Ausgabetor mit dem Register dient der Statusanzeige der Tastatur. Die Anzeige umfaßt das RUF-Signal (wenn der Rechner ein Zeichen erwartet) und 4 LED zur Kontrolle des Zustandes der Funktionstasten.

Die Resetschaltung verlängert den Ausgabeimpuls zum Löschen in die Größe von 100 µs, da dies die Mindestimpulsdauer des Sensorschaltkreises ist. Eventuell muß das Gatter X (Bild 4) gegen ein Monoflop getauscht werden. Die Arbeitspunkte der Transistoren sind so einzustellen, daß sich minimale Schaltzeiten ergeben (keine Übersteuerung). Die angegebenen Werte sind als Richtwerte zu betrachten.

Tastaturtreiber

Da die 64 Tasten nicht ausreichen und die Tastatur nicht selbständig arbeitet, ist ein umfangreiches Treiberprogramm erforderlich.

Die Tasten liefern die dualen Codierungen von 00H···3FH. Von diesen Tasten wurden die unteren 48 mit Zeichen belegt (Code 00···2FH). Von den 16 verbleibenden dienen die Tasten mit dem Code 3EH als Taste F1 und die mit der Codierung 3FH als F2. Der Rest wurde mit Steuerzeichen belegt, wie Leerzeichen, Wagenrücklauf-Zeilenschaltung (NL), Cursorbewegungen usw. Diese Tasten werden von der Funktionsumschaltung nicht berührt. Damit ergeben sich am Ende des Tastaturprogrammes 128 darstellbare Zeichen und 14 Steuerzeichen. Die darstellbaren Zeichen besitzen den Code 00H···7FH, die Steuerzeichen den Code 80H···8DH, und die in Funktionsebene 2 übrigbleibende Zeile (Eingangscode 20H···2FH) erhält die Codierungen von 90H bis 9FH.

Der Tastatur liegt insgesamt folgende Arbeitsweise zugrunde:

– Taste bedienen ergibt ein Zeichen,

– Taste F1 einmal bedient schaltet für die folgende Bedienung auf Funktionsebene 1 um (z.B. Großbuchstaben),

– Taste F1 zweimal hintereinander bedient schaltet bis auf Widerruf auf Funktionsebene 1 um,

– Gleiches gilt für Taste F2 analog,

– Bedienung von F1–F2 bzw. F2–F1 schaltet auf Grundebene zurück.

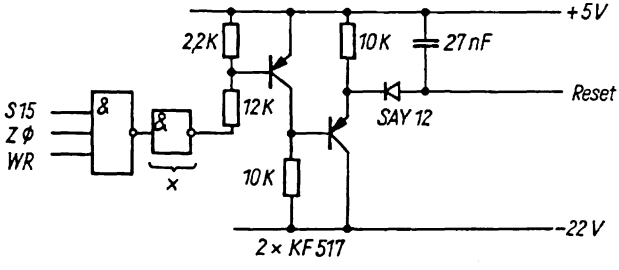


Bild 4
Resetschaltung (Treiber)

– Wird eine Taste länger als 1,5 s gedrückt, so beginnt das Programm automatisch mit der Wiederholung des Zeichens (Echofunktion).

Das Tastaturprogramm zerstört keine Register des Rechners und liefert nach jedem Aufruf einen Zeichencode im A-Register. Das Programm entprellt die Tastatur ein drittes Mal. Es können also anstelle der Sensortasten auch mechanische Tasten verwendet werden. Außerdem erfolgt eine Codierungssicherung, indem der Tastencode, der zuerst erkannt wurde, mit dem letzten vor dem Löschen eingelesenen Code übereinstimmen muß. Ist das nicht der Fall, so wird die Eingabe ignoriert. Dadurch ist eine Bedienung von zwei Tasten zugleich erfolglos.
Die vom Treiber realisierte Echofunktion ist in ihrer Geschwindigkeit veränderbar. Der Wert liegt dabei zwischen 1 (10 Zeichen je Sekunde) und 0FH (etwa 1,5 s je Zeichen) und ist auf der Speicherzelle F7 abzulegen.
Die vorgestellte Tastatur ist nicht voll kompatibel zu dem anschließend beschriebenen Bildschirmsystem. Dieses ist nur für 64 darstellbare Zeichen ausgelegt. Bei Darstellung von mehr als 64 Zeichen würde ein größerer Zeichengenerator und ein Bildwiederholspeicher mit 9 Bit erforderlich sein. Die 64 dadurch freien Zeichen können entweder mit den Inverszeichen (schwarz auf weiß) belegt werden oder bleiben für spätere Vorhaben frei.

BILDSCHIRMSTEUERUNG

Die Bildschirmsteuerung teilt sich in drei grundsätzliche Baugruppen. Das sind die *Zeitsteuerung*, die *Zeichenerzeugung* (Zeichengenerator) und die *Bildinhaltssteuerung* (Bildwiederholspeicher). In den hier dargestellten Schaltungen werden einige Teile vereinfacht. Das hat den Grund, daß es in diesen Steuerbaugruppen eine große Anzahl von Variationsmöglichkeiten hinsichtlich Leistungsfähigkeit und Aufwand gibt. Die hier vorgestellte Schaltung ist das Konzentrat einer 6 Leiterplatten umfassenden rastergraphischen Bildschirmsteuerung mit 8 Bildwiederholspeichern und flexiblem Zeichengenerator. Der hier angegebene Aufbau stellt zwar nicht das technische Aufwandsminimum dar, hat aber den Vorteil, daß er einfach den eigenen Vorstellungen und Bedürfnissen angepaßt werden kann.

Grundlagen

Um einen industriell hergestellten Fernseher ansteuern zu können, ist es in jedem Fall erforderlich, ein einigermaßen normgerechtes Bild-Synchron-Signal herzustellen. Aus diesem Grund einige Parameter dieses Signals:
Das Fernsehbild besteht bekanntlich aus 625 Zeilen (in 2 Halbbildern) mit einer Dauer von je 64 µs. Der Einfachheit halber werden für Displays zwei identische Halbbilder gezeichnet, so daß pro Bild 312 Zeilen anzusetzen sind. Aus diesen beiden Parametern, Zeilenlänge und Zeilenanzahl, ergibt sich der Grundaufbau der Zeitsteuerung.
Für das eigentlich darzustellende Bild ist entscheidend, daß von den 312 Zeilen etwa 256 auf dem Bildschirm sichtbar sind. Der Rest wird durch Synchron- und Austastsignale dunkelgetastet. Ähnliches trifft für die Zeile zu, die nur für etwa 43 µs sichtbar ist. Aus diesen Werten resultieren die erforderlichen Teilerverhältnisse und Taktfrequenzen.
Es ist weiter zu beachten, daß die höchste Ausgangsfrequenz (Grundwelle) des BAS-Signales der halben Taktfrequenz des Systems entspricht. Diese tritt auf,

wenn abwechselnd ein heller und ein dunkler Punkt dargestellt werden. Da die Videoendstufen im Fernsehgerät eine Bandbreite von etwa 5 MHz, haben und der ZF-Verstärker zumeist um 4 MHz, liegt, kann nur bei Systemfrequenzen bis etwa 7 MHz, über den Antenneneingang eingespeist werden. Bei Takten über 10 MHz, ist ein Umbau der Videoendstufe des Fernsehgerätes nicht zu umgehen.

Bei einer indirekten Einspeisung wird das Bildsignal über einen Kondensator zwischen ZF-Verstärker und Videoendstufe eingespeist. Der ZF-Verstärkerausgang ist dabei elektrisch abzutrennen (Schalter). Eventuell müssen die Arbeitspunkte über Widerstände wieder hergestellt werden.

Aus den oben aufgeführten Gründen wird das Steuersystem für 32 Zeichen je Zeile ausgelegt. Eine Umgestaltung ist jedoch leicht möglich.

Zeichenerzeugung (Bild 5)

Die Darstellung der Zeichen erfolgt in Form einer Punktkette. Dazu muß das Zeichen in einem Speicherschaltkreis als Punktbild (d. h. Bitanordnung) zur Verfügung stehen. Dabei wird der Einfachheit halber der Abstand zwischen den einzelnen Zeichen und Teilen mit als Bestandteil des Zeichens abgelegt. Unter einer Speicheradresse stehen dann die auf einer Zeile nebeneinander anzuordnenden Punkte, und jede zu beschreibende Zeile belegt eine Adresse im Speicher. An diesen Speicherschaltkreis muß man demzufolge die Adresse des darzustellenden Zeichens und der davon gerade benötigten Zeile anlegen. Das am Ausgang erscheinende Bitmuster muß dann seriell an die Videoendstufe gegeben werden. Über die Äquivalenz ist eine Negation des dargestellten Zeichens möglich (Inversdarstellung).

Die Pufferung der Information ist erforderlich, da nach dem Auslesen der

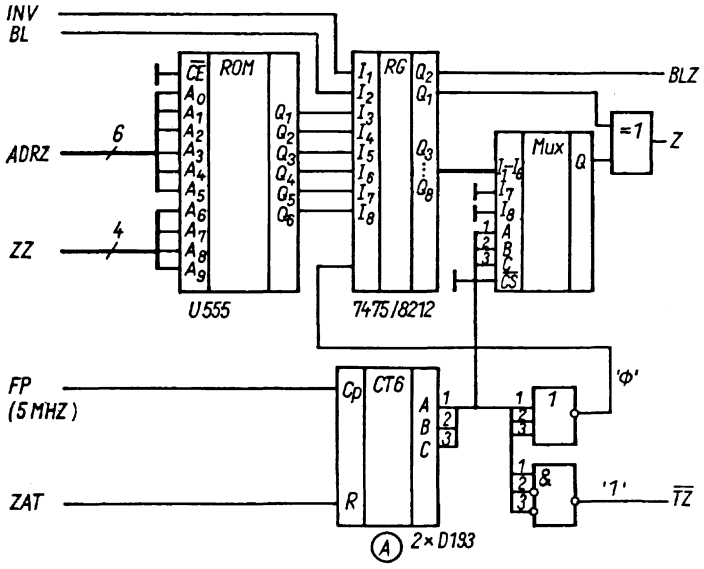


Bild 5. Zeichenerzeugung

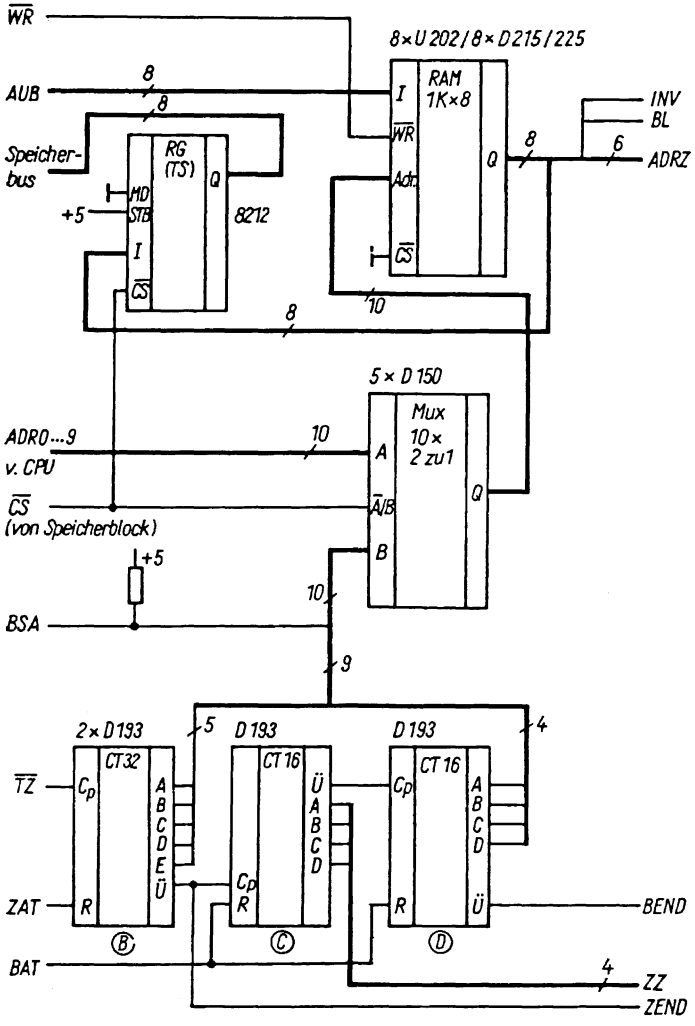


Bild 6
Bildinhaltsteuerung

Information sofort mit dem Ruf des nächsten Zeichens begonnen werden muß, da die Speicherzugriffszeiten etwa so groß wie die Darstellungszeit sind. Der Zähler A legt fest, wieviel Punkte ein Zeichen einschließlich Zwischenraum breit ist. Er wurde auf 6 Punkte (5 Zeichen, 1 Lücke) festgelegt.

Bildinhaltsteuerung (Bild 6)

Kernstück dieses Teils ist der Bildwiederholspeicher. In ihm werden die Zeichen in ihrer darzustellenden Reihenfolge abgelegt. Er muß zum einen von der Steuerung zu erreichen sein und zum anderen dem Rechner Zugriffsmöglichkeiten einräumen. Der relativ kleine Bildwiederholspeicher ist hier als Bestandteil des Adreßraumes der CPU

eingesetzt. Bei größeren Kapazitäten ist auch eine Ansteuerung über E/A-Tore möglich.
Die CPU kann über den Ausgabebus den Speicher schreiben und über den Puffer lesen. Zu beiden Aktionen wird der Multiplexer von den Adressen der Bildschirmsteuerung auf die Adressen der CPU umgeschaltet. Die gelesenen Daten müssen durch den Pufferschaltkreis vom Speicherbus getrennt werden, da der Bildwiederholspeicher selbst ständig im aktiven Zustand steht, die Daten den Bus aber nicht blockieren dürfen.
Die vorhandene Zählerkette bestimmt das Bildformat. Mit dem Zähler B wird die Zahl der Zeichen je Zeile festgelegt. Er wird mit dem Signal ZAT (Zeilenaustastung) nach jeder Zeile für die 'Dunkelzeit' gestoppt. Die gleiche Funktion realisiert das Signal BAT für die Bildaustastung. Der Zähler C legt fest, wieviel Fernsehzeilen zu einer Alphazeile gehören. Von diesem Zähler wären solche Funktionen wie Unterstreichungsposition abzuleiten. Dieser Zähler steuert zur Adressierung der Zeile innerhalb des Zeichens den Zeichengenerator an. Der dritte Zähler (D) bestimmt die Anzahl der Alphazeilen auf dem Bildschirm. Sie wurden hier auf 16 eingestellt. Sollen mehr verwendet werden, müßte der 5. Ausgang des Zählers mit BSA verbunden werden. Es können bei allen drei Zählern andere Teilverhältnisse eingesetzt werden. Es ergeben sich dann lediglich andere Zuordnungen im Speicher. Es ist aber darauf zu achten, daß das Produkt der Teilerfaktoren von Zähler C und D etwa 265 ergibt, da sonst die Bildschirmfläche überschritten oder nicht ausgenutzt wird. Aus gleichem Grund darf das Produkt von (Teilerv. A × Teilerv. B × 1/Punktfrequenz) 43 µs nicht überschreiten.
Das Signal BSA kann zur Bildsatzauswahl (z.B. Systembild/Nutzerbild) genutzt oder an die Zähler mit angeschlossen werden.

Zeitsteuerung

Alle Synchron- und Austastsignale werden von einer Teilerkette abgeleitet (Bild 7). Die Eingangsfrequenz beträgt 1 MHz. Die Punktfrequenz (FP in Bild 5) muß ein ganzzahliges Vielfaches dieser Eingangsfrequenz sein, da sonst Phasenfehler zwischen den Zeilen auftreten.
Das Prinzip wurde aus [3] entnommen. Dort wird auch ein Modulator für die Einspeisung über den Antenneneingang vorgestellt.
Der Zeilensynchronimpuls wird von einem Monoflop gebildet und beträgt etwa 5 µs. Er liegt am Zeilenanfang (fallende Flanke von F). Der Bildsynchronimpuls beginnt am Bildende und endet nach der 3. Zeile, was etwa 190 µs ergibt. Beide werden zusammengefaßt und dem Mischer zugeführt.
Das Flip-Flop zur Zeilenaustastung wird 15 µs nach Zeilenbeginn auf »hell« geschaltet. Damit entsteht auf dem Fernsehbildschirm ein linker Rand von etwa 1 cm. Dunkelgetastet wird die Zeile durch das Signal 'ZEND'. Das gleiche Verfahren wird für die Bildaustastung angewendet, welche bei Zeile 32 helltastet und mit dem Signal 'BEND' diesen Zustand wieder aufhebt.
Vor dem Mischer (Bild 8) werden die Austastsignale, das Zeichen und das Bit zur Blinkerlaubnis zusammengefaßt. Die Diode am Gatterausgang bewirkt die notwendige Pegelabsenkung des Bildsignals gegenüber dem Synchronsignal.

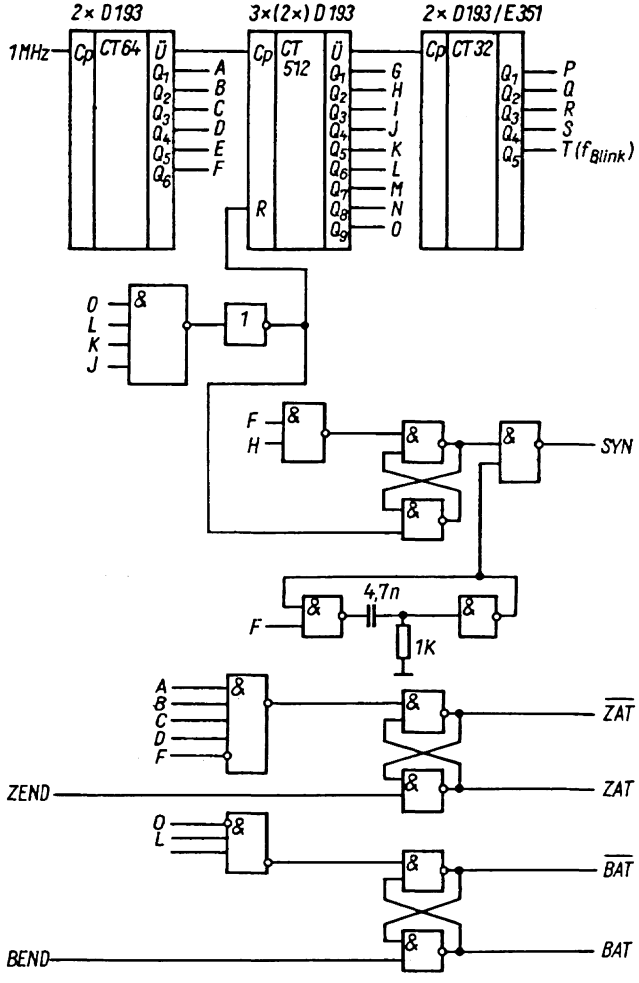


Bild 7. Zeitsteuerung

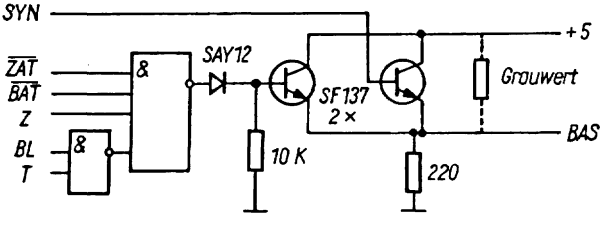


Bild 8. Mischer

```
0001                            ;
0002                            ;
0003                            ;  ZEITSCHLEIFE 1
0004    0000  DDE5      LOOP1:  PUSH  IX
0005    0002  DD21C800          LD    IX,ZEIT1
0006    0006  DD7701            LD    (IX+1),A
0007    0009  DD360000  LOOPA:  LD    (IX+0),0H
0008    000D  DD3400    LOOPB:  INC   (IX+0)
0009    0010  20FB              JRNZ  LOOPB-#
0010    0012  DD3501            DEC   (IX+1)
0011    0015  20F2              JRNZ  LOOPA-#
0012    0017  DDE1              POP   IX
0013    0019  C9                RET
0014                            ; A=1     4MS
0015                            ; A=255   OA 1 S
0016                            ;
0017                            ;
0018                            ; TASTATUR
0019                            ; ========
0020                            ;
0021                            ; REGISTERBELEGUNG
0022                            ; NACH AUSSEN :  AF   GENUTZT
0023                            ;                WERT AM ENDE IN <A>
0024                            ;
0025                            ; INTERN      :  <B>   STATUS
0026                            ;                <C>   EINGANGSPUFFER ZEICHEN
0027                            ;                      VON HARDWARE
0028                            ;                <E>   ECHOZEITFAKTOR
0029                            ;                <D>   ZEICHEN BEI UMCODIERUNG
0030                            ;                <H>   MASKE
0031                            ;
0032                            ;
0033    001A  C5        TAST:   PUSH  BC
0034    001B  D5                PUSH  DE
0035    001C  E5                PUSH  HL
0036    001D  26C0              LD    H,0C0H
0037    001F  3AC700            LD    A,(F4)
0038    0022  CBE7              SET   4,A
0039    0024  47                LD    B,A
0040    0025  D30E              OUT   0EH
0041    0027  3AC600            LD    A,(F7)
0042    002A  E60F              AND   0FH
0043    002C  5F                LD    E,A
0044                            ;
0045                            ; SCHLEIFE :  WARTEN AUF DRUECKEN DER TASTE
0046    002D  DB0F      TASTA:  IN    0FH
0047    002F  4F                LD    C,A
0048    0030  A4                AND   H
0049    0031  BC                CMP   H
0050    0032  2809              JRZ   TASTS-#
0051    0034  3E14              LD    A,20
0052    0036  CD0000            CALL  LOOP1
0053    0039  1E1E              LD    E,30
0054    003B  18F0              JR    TASTA-#
0055                            ;
0056                            ; AUSGABE  'PEEP'
0057    003D  D30A      TASTS:  OUT   0AH
0058                            ;
0059                            ; SCHLEIFE :  WARTEN AUF LOSLASSEN DER TASTE
0060    003F  3E06      TASTB:  LD    A,6
0061    0041  CD0000            CALL  LOOP1
0062    0044  DB0F              IN    0FH
0063    0046  B9                CMP   C
0064    0047  20E4              JRNZ  TASTA-#
0065
0066    0049  D30F              OUT   0FH
0067    004B  1D                DEC   E
0068    004C  2804              JRZ   TASTP-#
0069    004E  3E06              LD    A,6
0070    0050  CD0000            CALL  LOOP1
0071    0053  DB0F              IN    0FH
0072    0055  A4                AND   H
0073    0056  20E7              JRNZ  TASTB-#
0074                            ;
0075                            ; ZEICHEN IST UEBERNOMMEN ==> AUSWERTUNG
```

```
0076    0058  79          TASTP:  LD    A,C
0077    0059  E63F                AND   3FH
0078    005B  57                  LD    D,A
0079    005C  FE3E                CMP   3EH
0080    005E  281D                JRZ   TAST1-#
0081    0060  FE3F                CMP   3FH
0082    0062  2830                JRZ   TAST2-#
0083    0064  E630                AND   30H
0084    0066  FE20                CMP   30H
0085    0068  203A                JRNZ  TAST3-#
0086                      ;
0087                      ;  STEUERZEICHEN  CODE 00...0FH
0088    006A  7A                  LD    A,D
0089    006B  E60F                AND   0FH
0090    006D  F620                OR    20H
0091                      ;
0092                      ; ENDEBEHANDLUNG  /  ABGANG
0093    006F  57          TASTC:  LD    D,A
0094    0070  78                  LD    A,B
0095    0071  E50A                AND   0AH
0096    0073  32C700              LD    (F4),A
0097    0076  D30E                OUT   0EH
0098    0078  7A                  LD    A,D
0099    0079  E1                  POP   HL
0100    007A  D1                  POP   DE
0101    007B  C1                  POP   BC
0102    007C  C9                  RET
0103                      ;
0104                      ;
0105                      ; FUNKTION 1  SETZEN/RUECKSETZEN
0106    007D  CB40        TAST1:  BIT   0,B
0107    007F  2004                JRNZ  TASTH-#
0108    0081  CBC0                SET   0,B
0109    0083  1802                JR    TASTI-#
0110    0085  CBC8        TASTH:  SET   1,B
0111    0087  CB50        TASTI:  BIT   2,B
0112    0089  2804                JRZ   TASTJ-#
0113    008B  78          TASTK:  LD    A,B
0114    008C  E610                AND   10H
0115    008E  47                  LD    B,A
0116    008F  78          TASTJ:  LD    A,B
0117    0090  D30E                OUT   0EH
0118    0092  1899                JR    TASTA-#
0119                      ; ==> ZURUECK ZUR TASTENABFRAGE
0120                      ;
0121                      ; FUNKTION 2  SETZEN/RUECKSETZEN
0122    0094  CB50        TAST2:  BIT   2,B
0123    0096  2004                JRNZ  TASTL-#
0124    0098  CBD0                SET   2,B
0125    009A  1802                JR    TASTM-#
0126    009C  CBD8        TASTL:  SET   3,B
0127    009E  CB40        TASTM:  BIT   0,B
0128    00A0  28ED                JRZ   TASTJ-#
0129    00A2  18E7                JR    TASTK-#
0130                      ;
0131                      ;
0132                      ;
0133                      ;  UEBERSCHLUESSELLUNG DER CODIERUNG , WENN
0134                      ;  UNGLEICH CODE 8XH
0135    00A4  78          TAST3:  LD    A,B
0136    00A5  E603                AND   3H
0137    00A7  2813                JRZ   TASTH-#
0138                      ;
0139                      ; NOCH F1 MOEGLICH
0140    00A9  7A                  LD    A,D
0141    00AA  E630                AND   30H
0142    00AC  FE20                CMP   20H
0143    00AE  2007                JRNZ  TAST4-#
0144                      ;
0145                      ; SYSTEMSTEUERFUNKTIONEN AUF F1 ==> CODE 90...9FH
0146    00B0  7A                  LD    A,D
0147    00B1  E60F                AND   0FH
0148    00B3  F690                OR    90H
0149    00B5  18B8                JR    TASTC-#
0150
```

```
0151                    ; F1 WAR GESETZT ==> CODE 30...4FH
0152    00B7  7A        TAST4:  LD    A,D
0153    00B8  C630              ADD   30H
0154    00BA  18B3              JR    TASTC-#
0155                    ;
0156                    ; KEINE ODER F2 WAR GESETZT
0157    00BC  78        TASTN:  LD    A,B
0158    00BD  E60C              AND   0CH
0159    00BF  7A                LD    A,D
0160    00C0  28AB              JRZ   TASTC-#
0161                    ; KEINE FUNKTION GESETZT ==> CODE  00...2FH
0162    00C2  C650              ADD   50H
0163    00C4  18A9              JR    TASTC-#
0164                    ; F2 WAR GESETZT ==> CODE 50...7FH
0165                    ;
0166                    ;
0167                    ;
0168                    ;  R A M
0169                    ;
0170    00C6            F7:     BER   1                 ; ECHOZEITFAKTOR (0...F)
0171    00C7            F4:     BER   1                 ; STATUSREGISTER TASTATUR
0172    00C8            ZEIT1:  BER   2
0173                    ;
0174                    ;
0175                    ;
0176                            END
KEINE SYNTAX-FEHLER CRAS 4200-K1520
```

Videotreiber

Das Videotreiberprogramm (vgl. S. 29 bis 31) verwaltet den Bildschirm in Form einer Schreibmaschinensteuerung, d. h., der Bildschirm bekommt ein Zeichen übergeben, das er an die derzeitige 'Wagenposition' (gleich Cursorposition) setzt.
Der Treiber ignoriert alle Zeichen, die kleiner als 20H und größer gleich 60H sind. Eine Ausnahme bilden dabei die im Programmkopf angegebenen Steuerzeichen.
Durch die Maskierung der Zeichen auf 64 muß die Anordnung der Zeichen im Zeichengenerator anders als in [1] erfolgen. Der Z-Generator beginnt mit dem Buchstabenblock (32 Zeichen) und wird vom Block der Interpunktionszeichen und Ziffern gefolgt. Diese Anordnung entspricht auch den industriell gefertigten Zeichengeneratoren [4].
Vom Treiberprogramm wird kein Rollbild realisiert. Es wurde zugunsten der Einfachheit darauf verzichtet. Das Programm dazu könnte sinnvollerweise in das Unterprogramm zur Cursorbegrenzung (CUGR) mit eingebunden werden.
Durch den Ruf des Programms mit dem Zeichen 0EH im Akku wird der Bildschirm gelöscht und der Treiber initialisiert. Dieser Ruf sollte deshalb sofort nach dem Einschalten des Rechners erfolgen.

(Fortsetzung in Heft 3)

```
0001                            ;       PN    VIDEO
0002                            ;
0003                            ; K L E I N E R  V I D E O T R E I B E R
0004                            ;
0005                            ;
0006                            ;
0007                            ;
0008                            ; INPUT :  ASCII-ZEICHEN IN <A>
0009                            ;
0010                            ; ZUGELASSENE ZEICHEN (CODIERUNG IN HEX) :
0011                            ;     NL = 1E       HT =  9       LF = 0A
0012                            ;     CR = 0D       BS =  8       CLR= 0E
0013                            ;     BLINK E/A     = 13
0014                            ;     INVERS E/A    = 14
0015                            ;     CURSOR HOCH   = 11
0016                            ;       "=  RUNTER= 0A
0017                            ;       "=  RECHTS= 12
0018                            ;       "=  LINKS = 08
0019                            ;
0020                            ;
0021                            ;
0022                            ; PROGRAMM VERAENDERT KEINE REGISTER
0023                            ;
0024  0000  E5          VIDEO:  PUSH  HL
0025  0001  D5                  PUSH  DE
0026  0002  C5                  PUSH  BC
0027  0003  F5                  PUSH  AF
0028                            ;
0029  0004  CD8900              CALL  CURE                ; CORSOR LOESCHEN
0030  0007  E67F                AND   7FH                 ; NO PARITY
0031  0009  2A0001              LD    HL,(CUSO)
0032                            ; AUFSPALTUNG STEUERZEICHEN
0033  000C  FE1E                CMP   1EH
0034  000E  2876                JRZ   VA-#                ; NL
0035  0010  FE09                CMP   9
0036  0012  2834                JRZ   VB-#                ; HT
0037  0014  FE0A                CMP   10
0038  0016  2840                JRZ   VC-#                ; LF
0039  0018  FE0D                CMP   0DH
0040  001A  2842                JRZ   VD-#                ; CR
0041  001C  FE08                CMP   8
0042  001E  2844                JRZ   VE-#                ; BS
0043  0020  FE0E                CMP   0EH
0044  0022  CAA100              JPZ   VL                  ; CLEAR
0045  0025  FE11                CMP   11H
0046  0027  283E                JRZ   VF-#                ; CU HOCH
0047  0029  FE12                CMP   12H
0048  002B  2842                JRZ   VG-#                ; CU RECHTS
0049  002D  FE13                CMP   13H
0050  002F  2841                JRZ   VH-#                ; BLINK
0051  0031  FE14                CMP   14H
0052  0033  2847                JRZ   VJ-#                ; INVERS
0053                            ;
0054  0035  FE20                CMP   20H
0055  0037  3807                JRC   VZ-#                ; <20H  ==> NICHT DARSTELLBAR
0056  0039  FE60                CMP   60H
0057  003B  3003                JRNC  VZ-#                ; >=60H          =="==
0058                            ;
0059  003D  CDD200              CALL  ZEISE               ; ZEICHEN SETZEN
0060                            ;
0061  0040  CDE900    VZ:       CALL  CUSE                ; CURSOR-SETZEN
0062  0043  F1                  POP   AF
0063  0044  C1                  POP   BC
0064  0045  D1                  POP   DE
0065  0046  E1                  POP   HL
0066  0047  C9                  RET
0067                            ;
0068                            ;
0069                            ; HT
0070  0048  110800    VB:       LD    DE,8
0071  004B  19                  ADD   HL,DE
0072  004C  7D                  LD    A,L
0073  004D  E6F8                AND   0F8H
0074  004F  6F                  LD    L,A
0075  0050  220001    VBA:      LD    (CUSO),HL
0076  0053  CDF500              CALL  CUGR
0077  0056  18E8      VBB:      JR    VZ-#
0078                            ; LF
0079  0058  112000    VC:       LD    DE,ZL
0080  005B  19                  ADD   HL,DE
```

```
0081    005C  18F2            JR    VBA-#
0082
0083    005E  7D        VD:   LD    A,L
0084    005F  E6E0            AND   0E0H                ; MASKE=100H-ZL !!!
0085    0061  6F              LD    L,A
0086    0062  18EC            JR    VBA-#
0087                  ; BS
0088    0064  2B        VE:   DEC   HL
0089    0065  18E9            JR    VBA-#
0090                  ; CU HOCH
0091    0067  113000    VF:   LD    DE,2L
0092    006A  AF              XOR   A
0093    006B  ED52            SBC   HL,DE
0094    006D  18E1            JR    VBA-#
0095                  ; CU RECHTS
0096    006F  23        VG:   INC   HL
0097    0070  18DE            JR    VBA-#
0098                  ; BLINK
0099    0072  3A0201    VH:   LD    A,(BLI)
0100    0075  EE40            XOR   40H
0101    0077  320201          LD    (BLI),A
0102    007A  18D9            JR    VBB-#
0103                  ; INVERS
0104    007C  3A0201    VJ:   LD    A,(BLI)
0105    007F  EE80            XOR   80H
0106    0081  320201          LD    (BLI),A
0107    0084  18D3            JP    VBB-#
0108                  ; NL
0109    0086  112000    VA:   LD    DE,2L
0110    0089  19              ADD   HL,DE
0111    008A  7D              LD    A,L
0112    008B  E6E0            AND   0E0H
0113    008D  6F              LD    L,A
0114    008E  224001          LD    (CUSO),HL
0115    0091  CDF500          CALL  CUSR
0116    0094  113201          LD    DE,BILD
0117    0097  19              ADD   HL,DE
0118    0098  0620            LD    B,2L
0119    009A  3620      VAA:  LD    M,20H
0120    009C  23              INC   HL
0121    009D  10FE            DJNZ  VAA-#               ; NEUE ZEILE LOESCHEN
0122    009F  18B5            JR    VBB-#
0123                  ;
0124                  ; CLEAR
0125    00A1  213201    VL:   LD    HL,BILD
0126    00A4  010B02          LD    BC,BEND-BILD
0127    00A7  3620      VLA:  LD    M,20H               ; SPACE LADEN
0128    00A9  0B              DEC   BC
0129    00AA  78              LD    A,B
0130    00AB  B1              OR    C
0131    00AC  20F9            JRNZ  VLA-#
0132    00AE  210201          LD    HL,BLI
0133    00B1  ED430001        LD    (CUSO),BC
0134    00B5  79              LD    M,2
0135    00B6  C34000          JMP   V2
0136                  ;
0137                  ;
0138                  ; CURSOR LOESCHEN
0139                  ;
0140    00B9  4F        CURE: LD    C,A                 ; A RETTEN
0141    00BA  3A0201          LD    A,(BLI)
0142    00BD  47              LD    B,A
0143    00BE  113201          LD    DE,BILD
0144    00C1  2A4001          LD    HL,(CUSO)
0145    00C4  F5              PUSH  AF
0146    00C5  7C              LD    A,H
0147    00C6  E631            AND   31
0148    00C8  67              LD    H,A
0149    00C9  F1              POP   AF
0150    00CA  19              ADD   HL,DE
0151    00CB  7E              LD    A,M                 ; ALTES ZEICHEN HOLEN
0152    00CC  E63F            AND   3FH                 ; NO BLINK ; NO INVERS
0153    00CE  B0              OR    B                   ; BLINK/INVERS NEU
0154    00CF  77              LD    M,A
0155    00D0  79              LD    A,C
0156    00D1  C9              RET
0157                  ;
0158                  ;
0159                  ; ZEICHEN SETZEN
0160                  ;
```

```
0161    00D2  E63F      ZEISE:  AND   3FH                ; MASK AUF ZL ZEICHEN
0162    00D4  B0                OR    B                  ; BLINK/INVERS NEU
0163    00D5  4F                LD    C,A
0164    00D6  110301            LD    DE,BILD
0165    00D9  2A0001            LD    HL,(CUSO)
0166    00DC  19                ADD   HL,DE
0167    00DD  71                LD    M,C                ; ZEICHEN RAUS
0168    00DE  2A0001            LD    HL,(CUSO)
0169    00E1  23                INC   HL
0170    00E2  220001            LD    (CUSO),HL
0171    00E5  CDF500            CALL  CUGR
0172    00E8  C9                RET
0173                    ;
0174                    ;
0175                    ; CURSOR SETZEN
0176                    ;
0177    00E9  2A0001    CUSE:   LD    HL,(CUSO)
0178    00EC  110301            LD    DE,BILD
0179    00EF  19                ADD   HL,DE
0180    00F0  7E                LD    A,M
0181    00F1  F6C0              OR    0C0H               ; BLINK+INVERS EIN
0182    00F3  77                LD    M,A
0183    00F4  C9                RET
0184                    ;
0185                    ;
0186                    ; CURSOR BEGRENZEN AUF 0...1FF
0187                    ;
0188    00F5  2A0001    CUGR:   LD    HL,(CUSO)
0189    00F8  7C                LD    A,H
0190    00F9  E601              AND   01
0191    00FB  67                LD    H,A
0192    00FC  220001            LD    (CUSO),HL
0193    00FF  C9                RET
0194                    ;
0195                    ;
0196                    ;  R A M - ZELLEN
0197                    ;
0198                    ZL:     EQU   32                 ; ZEILENLAENGE
0199                    BG:     EQU   1                  ; BILDGROESSE (= H(BEND-BILD-1)
0200    0100            CUSO:   DEF   2                  ; POSITION DES CURSOR (RELATIV)
0201    0102            BLI:    DEF   1                  ; FLAG FUER BLINK + INVERS
0202                    ;
0203                    ;
0204                    ;
0205    0103            BILD:   DEF   512                ; BILDWIEDERHOLSPEICHER
0206    0303            BEND:   DEF   0
0207                    ;
0208                    ; BILDWHSP. SOLLTE AUF EINER CHIPGRENZE BEGINNEN
0209                    ;           ( MUSS ABER NICHT )
0210                    ; BIT 0...5 IM BILDWHSP. IST DER CODE DES ZEICHENS
0211                    ; BIT 6           ="=     IST DAS BLINKBIT
0212                    ; BIT 7           ="=     KENNZEICHNET DIE INVERSDARSTELLUNG
0213                    ;
0214                    ;
0215                    ;
0216                            END
KEINE SYNTAX-FEHLER CRAS 4200-K1529
```

Literatur

[1] Taschenbuch Elektrotechnik, Bd. 2. PHILIPPOW, E. – Berlin: Verlag Technik, 1977. – S. 34

[2] MOS-Schaltkreise: Datenbuch des VEB Funkwerk Erfurt. – Erfurt, 1978. – S. 117–121
oder
Halbleiterinformation 174. – In: Radio – Fernsehen – Elektronik. – Berlin 30 (1981) 5. – S. 307

[3] Bildmustergenerator mit elektronisch erzeugtem Kreis/JUNGNICKEL, H. – In: Radio – Fernsehen – Elektronik. – Berlin 25 (1976) 21. – S. 681–684

[4] Halbleiterinformation 155. – In: Radio – Fernsehen – Elektronik. – Berlin 28 (1979) 3. – S. 174, Bild 5b

Autor:

Dr.-Ing. Gert Schönfelder
DDR 8029 Dresden
Auf dem Eigen 39

Problemanalytiker im Rechenzentrum der Sektion Informationsverarbeitung der Ingenieurhochschule Dresden

Nimm-Spiel gegen den K 1003

(Der Anfang dieses Artikels erschien schon in Heft 1. Da aber nicht alle Leser dieses Heft haben, veröffentlichen wir den vollständigen Artikel.)

Schon vor dem Anbruch der Computerära wurde das *Nimm-Spiel* gespielt. Ohne die leiseste Ahnung von einem Taschen- oder Tischrechner spielten zwei Personen gegeneinander. Das war und ist auch heute noch mit ganz einfachen Spielregeln möglich: Auf den Tisch wird eine bestimmte Anzahl von Gegenständen gelegt. Hier bieten sich Streichhölzer an, weshalb das Spiel auch als *Streichholzspiel* bekannt ist. Weiterhin wird festgelegt, welche maximale Anzahl von Streichhölzern jeder Spieler bei jedem Zug wegnehmen darf, wobei mindestens ein Holz wegzunehmen ist. Verloren hat diejenige Person, die das letzte Streichholz wegnehmen muß.
Im praktischen Spielverlauf wurde oft deutlich, daß, von wenigen Ausnahmen abgesehen, stets die gleiche Person gewann. Sie war es meistens auch, die das Streichholzspiel vorschlug, weil sie offenbar wußte, »wie es geht«, und auf die Unwissenheit des Gegners hoffte.
Wir wollen bei diesem Spiel die eine der beiden Personen durch den programmierbaren Tischrechner K 1003 ersetzen, also gegen den Rechner spielen. Es eignen sich dafür natürlich auch andere programmierbare Rechner. Ein tastenprogrammierbarer Taschenrechner sollte aber mindestens 150 Befehlsplätze und 5 Datenplätze haben. Besonders geeignet sind Rechner mit Drucker (K 1003, TI 58/59, HP 41, PC-Serie von Sharp, PB-Serie und FX-Serie von Casio u. a.). Sie drucken den Spielverlauf aus und ermöglichen über Textausdrucke einen Minidialog mit dem Rechner. Das wiederum ist bedienerfreundlich, weil umfangreiche Beschreibungen des Programmablaufes entfallen. Die Textprogrammierung »frißt« allerdings Befehlsschritte, wie wir noch sehen werden.
Wir haben aus einer Vielzahl von Spielen hier bewußt das Steichholzspiel gewählt, da es, vor allem für den Anfänger, einen zweifachen Erkenntnisgewinn bringt (vom Spaß einmal ganz abgesehen).

1. Das Streichholzspiel ist gut geeignet, um die Schritte von P (wie Problem) nach P (wie Programm) zu zeigen, nämlich: *Problemanalyse, Lösungsalgorithmus, Programmablaufplan, Rechenprogramm.*

2. Das Streichholzspiel macht deutlich, daß der Rechner mit seinem Programm nie »schlauer« sein kann als der Mensch. Damit ist natürlich nur der Mensch gemeint, der um die Dinge weiß.

Dieses Spiel bietet damit auch für Arbeitsgemeinschaften, die u. U. noch keinen programmierbaren Rechner zur Verfügung haben, Denkvergnügen, Diskussionsstoff und Spaß.

Beginnen wir nun mit der **Problemanalyse.** Der Rechner und wir, also die beiden Gegner, versuchen, die *Gewinnstrategie* anzuwenden. Kennen wir sie nicht, so haben wir keine Siegeschancen. Um diese Gewinnstrategie zu erfahren, werden wir »das Pferd am Schwanz aufzäumen«. Dazu betrachten wir folgendes Beispiel: Die Gesamtzahl G der Streichhölzer sei $G = 15$. Die maximale Anzahl pro Zug, kurz als Maximalzug M bezeichnet, sei $M = 3$. Damit liegen 15 Streichhölzer auf dem Tisch. Der Rechner und wir dürfen also pro Zug 1 oder 2 oder 3 Hölzer wegnehmen. Damit der Rechner verliert, muß er in der letzten Runde genau 1 Streichholz vorfinden, das er notgedrungen ziehen muß. In der vorletzten Runde aber muß der Rechner 5 Hölzer vorfinden. Wenn er nämlich 3 Hölzer wegnimmt (Rest 2), dann nehmen wir in der letzten Runde nur 1 Holz weg. Auch das Wegnehmen von einem Holz oder zwei Hölzern führt in der letzten Runde stets zu seiner Niederlage. Führen wir diesen Gedanken fort, dann muß der Rechner in der drittletzten Runde 9 und in der viertletzten Runde 13 Hölzer vorfinden, damit wir zum Sieg gelangen.

Damit lautet für $M = 3$ die Gewinnreihe: 1, 5, 9, 13 ....

Da die Gesamtzahl $G = 15$ ist, interessieren uns die weiteren Glieder der Gewinnreihe nicht mehr. Da wir aber das Problem *für beliebige* G, M (wobei $G \gg M$) analysieren wollen, stellen wir folgende allgemeine Gewinnreihe auf:

$1, M + 2, 2M + 3, 3M + 4,$
$4M + 5, \ldots .$

Anders ausgedrückt ergibt sich ein beliebiges Glied der Gewinnreihe aus der Summe der vorherigen Glieder plus $(M + 1)$.

Wir erkennen daraus, daß der Gewinner schon nach dem ersten Zug feststeht. Wollen *wir* gewinnen, so müssen wir sofort nach Spielbeginn auf die Gewinnreihe »springen«. NIMM-Spiel-Programme in Rechenzentren gaukeln dem ahnungslosen Nutzer oft Großzügigkeit vor, indem sie die Wahl lassen, ob Rechner oder Person beginnen soll. Auch ich erinnere mich an die Zeit, in der den Besuchern des Zentralinstitutes für Schweißtechnik in Halle nicht nur schweißtechnische Forschungsergebnisse, sondern auch so ein »großzügiges« NIMM-Spiel vorgeführt wurde. Lassen wir aber dem Rechner den ersten Zug, dann »springt« er sofort auf die Gewinnreihe und hat den Sieg für sich schon in der Tasche (vielleicht besser: im Speicher).

Es gibt allerdings für das eben Gesagte eine einzige *Ausnahme.* Gehört nämlich die Gesamtmenge G selbst zur Gewinnreihe, dann gelingt es dem Beginner im 1. Zug nicht, auf die nächstkleinere Zahl der Gewinnreihe zu springen. Er darf aber auch nicht 0 Streichhölzer ziehen, um auf der Gewinnreihe zu bleiben. Sollte Ihnen also bei Ihrem nächsten Besuch in einem Rechenzentrum beim NIMM-Spiel der Rechner die Wahl des Beginnens lassen, dann wählen sie M und G so, daß G auf der Gewinnreihe liegt, und lassen »großzügig« den Rechner beginnen. Anschließend »springen« Sie dann auf die Gewinnreihe und besiegen ihn trotzdem.

Da der eben geschilderte Fall eine Ausnahme ist, wollen wir in unserem Spielprogramm von vornherein festlegen, daß *wir* beginnen. Nach dieser Problemanalyse können Sie nun schon gegen einen unwissenden Partner spielen und, sofern Sie sich im Spiel nicht verrech-

nen, gewinnen. Hier noch zum Abschluß gebräuchliche Gewinnreihen:

Für $M = 3$ gilt:

1, 5, 9, 13, 17, 21, 25, 29, . . .

Für $M = 4$ gilt:
1, 6, 11, 16, 21, 26, 31, 36, . . .

Für $M = 5$ gilt:
1, 7, 13, 19, 25, 31, 37, 43, . . .

Der **Lösungsalgorithmus,** wir bezeichneten den Kern der Sache als Gewinnstrategie, trat ja schon bei der eben angestellten Problemanalyse zutage. Für das angestrebte Programm werden wir folgende Schritte zu bearbeiten haben:

1. Eingabe der Gesamtzahl G und des Maximalzuges M,

2. Eingabe der Anzahl P der Hölzer, die wir zu Beginn ziehen,

3. Prüfung, ob $M \geqq P > 0$ ist. Falls dies nicht der Fall ist, wird die falsche Eingabe als Betrugsversuch gewertet, und der Ablauf wird abgebrochen.

4. Aufbau der Gewinnreihe, deren aktuellen Wert wir mit W bezeichnen wollen.

5. Sollten wir uns auf der Gewinnreihe befinden, dann tritt für die »Verlegenheitszüge« des Rechners ein Pseudozufallszahlengenerator in Aktion, der aus einer Zufallszahl Z einen zufälligen Rechnerzug P ($M \geqq P > 0$) erzeugt und damit für Abwechslung sorgt. Aus diesem Grunde wird auch beim Programmstart die Anfangszahl $Z = 0{,}5$ vorgegeben.

6. Im Programmablauf muß vor jedem Zug (von uns und vom Rechner) geprüft werden, ob nur noch genau 1 Streichholz übrig ist. Sobald diese Situation eintritt, wird das Programm beendet, und der Drucker weist den Sieg des Rechners oder unseren Sieg (das ist natürlich erfreulicher) aus.

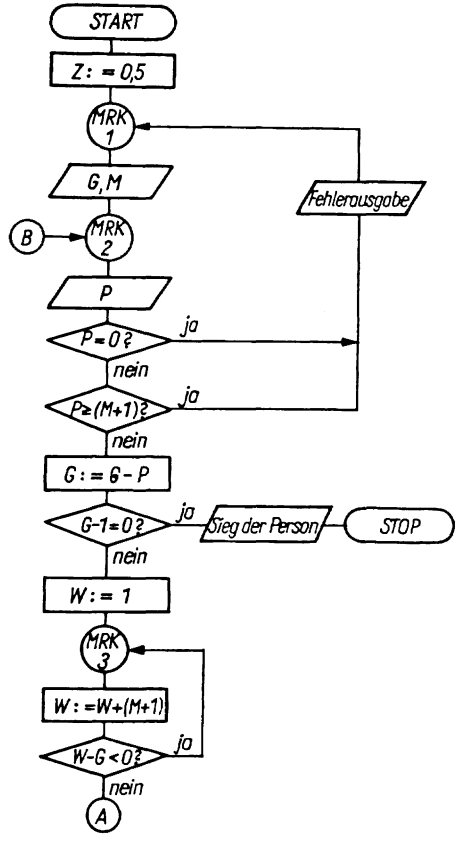


Bild 1. Programmablaufplan

Diese groben Darstellungen des Lösungsalgorithmus reichen für uns als Programmierer natürlich noch nicht aus, um daraus ein Rechenprogramm zu erstellen. Wir werden deshalb in einem **Programmablaufplan,** kurz als PAP bezeichnet, den gesamten Ablauf mit den entsprechenden Formelzeichen genau darstellen. Dazu noch einmal die verwendeten Formelzeichen:

G Gesamtzahl der Streichhölzer
M Maximalzug (maximale Anzahl pro Zug)
P Anzahl eines Zuges von Mensch und Rechner
W Aktueller Wert der Gewinnreihe

Z Zufallszahl für die Erzeugung eines »Verlegenheitszuges« P durch den Rechner

Den Programmablaufplan in der gebräuchlichen Symbolik zeigt Bild 1. Nach dem Programmstart wird die Anfangszahl $Z = 0{,}5$ für den Pseudozufallsgenerator bereitgestellt. Die *Marken* 1 bis 3 (MRK 1, MRK 2, MRK 3) werden uns die Programmierarbeit erleichtern, da sie *Einsprungstellen in das Programm* festlegen. Zunächst werden G und M eingegeben. Nach Eingabe von P wird dessen Richtigkeit geprüft. Im weiteren wird mit $G - 1 = 0$? die *Endebedingung* geprüft und dann, bei Verneinung dieser Frage, der aktuelle Wert W der Gewinnreihe gebildet. Je nachdem, ob wir auf der Gewinnreihe sind oder ob sich der Rechner darauf befindet, wird das Programm verzweigt. Sind wir auf der Gewinnreihe, dann bildet der Rechner mit Hilfe des Pseudozufallszahlengenerators einen »Verlegenheitszug« P (rechter Teil des PAP). Es gibt viele Möglichkeiten, solche Pseudozufallszahlen zu erzeugen. Im PAP ist eine Variante angegeben, die $\pi + Z$ im Exponenten einer e-Funktion benutzt. Manche Rechner haben in austauschbaren Moduln (z.B. TI 58/59) oder Blöcken (z.B. K 1002/1003) solche Pseudozufallszahlengeneratoren fest programmiert. Allerdings ist der Generator im Statistik-Block des K 1002/1003 sehr bescheiden, was die Güte der Zufälligkeit anbelangt. Die als zehnstelliger Dezimalbruch einzugebende Anfangszahl wird mit der Zahl 29 multipliziert. Der entstehende Nachkommateil liefert dann eine Zufallszahl zwischen 0 und 1. Die Normierung für P mit $M \geqq P > 0$ kann dann wie in unserem PAP über $P = \text{int}\,(M \cdot Z) + 1$ erfolgen [int $(M \cdot Z)$ heißt ganzzahliger Anteil von $(M \cdot Z)$].

Ist der Rechner aber selbst siegesgewiß

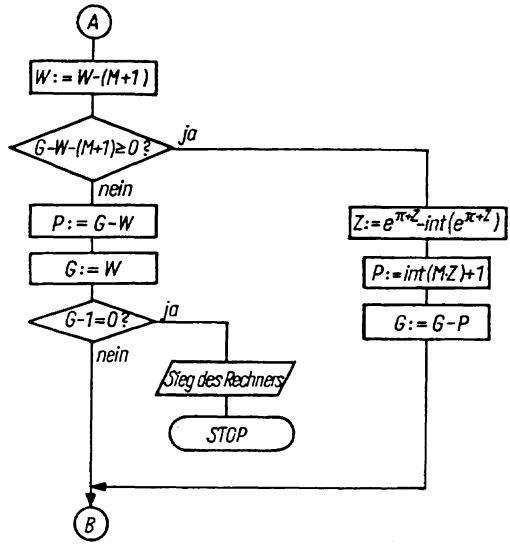


Bild 1 (Fortsetzung)

auf der Gewinnreihe, dann wird der linke Programmzweig des PAP durchlaufen, bei dem auch die Endebedingung für den Sieg des Rechners geprüft wird.

Der Anfänger kann mit Papier, Bleistift und einem Beispiel versuchen, diesen PAP im »Trockenkurs« abzuarbeiten. Dabei ist besonders auf solche Ergibt-Anweisungen wie $G := G - P$ oder $W := W + (M + 1)$ zu achten.

Die Krönung unserer Bemühungen soll aber nun das **Rechenprogramm** sein. Allerdings wird an dieser Stelle deutlich, daß der *entscheidende geistige Aufwand in den Stufen Problemanalyse und Lösungsalgorithmus* steckt. Mit einem guten Programmablaufplan und soliden Programmierkenntnissen ist dieser letzte Schritt bei weitem nicht so »geistintensiv« wie die vorangegangenen. Wir wollen, wie schon eingangs erwähnt, das Programm für den tastenprogrammierbaren Tischrechner K 1003 unter effektiver Nutzung des Thermo-

druckwerkes schreiben. Besitzer anderer programmierbarer Rechner können den PAP aus Bild 1 natürlich auch für ihren Rechner in ein Rechenprogramm umsetzen. Ebensogut wäre die Programmierung in einer problemorientierten Sprache wie FORTRAN, BASIC oder PASCAL möglich. Für den K 1003 werden wir die Datenspeicher folgendermaßen belegen:

Datenspeicherplatz

000	enthält	$M + 1$
001	enthält	G
002	enthält	W
003	enthält	P
004	enthält	Z

Es werden also insgesamt 5 Datenspeicherplätze benötigt. Damit kann die beim K 1003 beim Einschalten vorhandene Speicherplatzverteilung mit 7 Datenspeicherplätzen beibehalten werden.

Der *Numerateur* beim K 1002/1003 hilft beim Betrieb ohne Drucker bei der *Kennzeichnung der Ein- und Ausgaben.* Obwohl er im Druckerbetrieb mit Dialog (so wollen wir unser Programm erstellen) nicht benötigt wird, wollen wir ihn doch für die K-1002-Besitzer ins Programm einbauen. Allerdings müßten für den nichtdruckenden Betrieb noch einige Programmänderungen, insbesondere für die Ausgabe der Niederlage- oder Siegesnachricht, vorgenommen werden.[1] Für den Numerateur (NUM) gilt folgendes:

bei NUM = 01	Eingabe von G
bei NUM = 02	Eingabe von M
bei NUM = 03	Eingabe von P

Das vollständige Rechenprogramm für den K 1003 ist in Tabelle 1 dargestellt. Es umfaßt 467 Befehlsschritte. Dabei ist unbedingt zu beachten, daß rund 300 Befehle für die Arbeit mit Drucker (Text-, Tabellen- und Ergebnisdruck) benötigt wurden.

Solange wir genügend Speicherplatz haben, sollten wir aber darüber nicht traurig sein. Denn *je höher der Druckkomfort, desto bedienerfreundlicher ist das Programm.* Das soll an den folgenden Rechenbeispielen gezeigt werden.

Tabelle 2 zeigt zunächst, wie der Rechner mit seinem Programm auf falsche Eingaben von P reagiert. In Tabelle 3 wurde der relativ seltene, für uns nicht gewinnbare Fall dargestellt. Für $M = 4$ lautet die Gewinnreihe 1, 6, 11, 16, 21. Da die Gesamtzahl $G = 21$ auf der Gewinnreihe liegt, können wir uns im 1. Zug drehen und wenden, wie wir wollen, der Rechner wird dieses Spiel gewinnen.

Tabelle 4 zeigt für die Gesamtzahl $G = 40$ und den Maximalzug $M = 5$ zunächst eine Verlierer- und dann eine Gewinnerrechnung. Bei der Verliererrechnung steht unter »Rest« die Gewinnreihe von 31, 25, 19, 13, 7, die dem Rechner zum Sieg verhilft. Nach dem ersten unbedachten Zug von uns (im Bsp. unser 1. Zug mit $P = 5$) zieht der Rechner 4 Hölzer. Damit »springt« er auf die Gewinnreihe und bleibt bis zum siegreichen Ende darauf.

Bei der Gewinnerrechnung ziehen wir im 1. Zug 3 Streichhölzer. Damit kommen wir durch $40 - 3 = 37$ auf die Gewinnreihe. Obwohl uns der Rechner mit seinen »Verlegenheitszügen« 1, 5, 1, 3, 2, 1 verwirren will, lassen wir uns nicht beirren (verrechnen uns aber auch nicht beim Kopfrechnen) und verharren auf der Gewinnreihe.

Wir wollen mit diesen Beispielen unsere Betrachtungen zum Streichholzspiel abschließen. Zu Beginn wurde ein zweifacher Erkenntnisgewinn versprochen. Vielleicht mag der eine oder andere Leser über den bescheidenen Wis-

[1] Die ursprüngliche PAP- und Programmfassung für den K 1002 ohne Drucker hat Herr Karl-Heinz Tirsch, Dessau, erarbeitet.

0000	MRK	0039	H	0078	S	0117	2
0001	STM	0040	L	0079	T	0118	0
0002	DP	0041	=	0080		0119	
0003	5	0042	TEX	0081		0120	
0004	TXR	0043	GL	0082	S	0121	
0005	4	0044	STP	0083	I	0122	
0006	TEX	0045	TXR	0084	E	0123	
0007	S	0046	1	0085		0124	
0008	T	0047	TEX	0086	!	0125	
0009	R	0048	DRU	0087		0126	!
0010	E	0049	2	0088	I	0127	TEX
0011	I	0050	0	0089	C	0128	GL
0012	C	0051	ZS	0090	H	0129	STP
0013	H	0052	M	0091	ZS	0130	TEX
0014	H	0053	A	0092	-	0131	
0015	O	0054	X	0093	-	0132	
0016	L	0055	I	0094	-	0133	
0017	Z	0056	M	0095	-	0134	
0018	S	0057	A	0096	-	0135	
0019	P	0058	L	0097	-	0136	
0020	I	0059	Z	0098	-	0137	DRU
0021	E	0060	U	0099	-	0138	1
0022	L	0061	G	0100	-	0139	0
0023	ZS	0062	=	0101	-	0140	
0024	TEX	0063	TEX	0102	!	0141	
0025	MRK	0064	GL	0103	-	0142	!
0026	1	0065	STP	0104	-	0143	TEX
0027	NUM	0066	TEX	0105	-	0144	=0
0028	KOM	0067	DRU	0106	-	0145	STM
0029	0	0068	2	0107	ZS	0146	6
0030	TEX	0069	0	0108	TEX	0147	ST
0031	G	0070	ZS	0109	MRK	0148	ST
0032	E	0071	TEX	0110	2	0149	TXR
0033	S	0072	1	0111	NUM	0150	3
0034	A	0073	ADD	0112	2	0151	TRX
0035	M	0074	TXR	0113	TRX	0152	MOD
0036	T	0075	TEX	0114	1	0153	SUB
0037	Z	0076	R	0115	TEX	0154	>=0
0038	A	0077	E	0116	DRU	0155	STM

Tabelle 1

```
0156 6
0157 ST
0158 ST
0159 TRX
0160 1
0161 TRX
0162 SUB
0163 3
0164 TXR
0165 1
0166 MOD
0167 1
0168 SUB
0169 =0
0170 STM
0171 7
0172 ST
0173 ST
0174 1
0175 TXR
0176 2
0177 MRK
0178 3
0179 TRX
0180 TRX
0181 ADD
0182 2
0183 TXR
0184 2
0185 TRX
0186 SUB
0187 1
0188 <0
0189 STM
0190 3
0191 ST
0192 ST
0193 TRX
0194 2
0195 TRX
0196 SUB
0197 TXR
0198 2
0199 TRX
0200 1
0201 TRX
0202 SUB
0203 2
0204 TXR
0205 3
0206 TRX
0207 SUB
0208 >=0
0209 STM
0210 5
0211 ST
0212 ST
0213 TRX
0214 3
0215 STM
0216 UP
0217 9
0218 TRX
0219 2
0220 TXR
0221 1
0222 MOD
0223 1
0224 SUB
0225 =0
0226 STM
0227 8
0228 ST
0229 ST
0230 STM
0231 2
0232 MRK
0233 5
0234 TRX
0235 4
0236 PI
0237 ADD
0238 E↑X
0239 KNO
0240 INT
0241 SUB
0242 TXR
0243 4
0244 MOD
0245 1
0246 TRX
0247 VXV
0248 SUB
0249 MUL
0250 INT
0251 1
0252 ADD
0253 STM
0254 UP
0255 9
0256 TRX
0257 1
0258 VXV
0259 SUB
0260 TXR
0261 1
0262 STM
0263 2
0264 MRK
0265 6
0266 TEX
0267   B
0268   E
0269   T
0270   R
0271   U
0272   G
0273   S
0274   V
0275   E
0276   R
0277   S
0278   U
0279   C
0280   H
0281   !
0282   !
0283   B
0284   E
0285   G
0286   I
0287   N
0288   N
0289   E
0290   N
0291
0292   S
0293   I
0294   E
0295
0296   N
0297   E
0298   U
0299 TEX
0300 ZS
0301 STM
0302 1
0303 MRK
0304 7
0305 TEX
0306
0307
0308
0309
0310
0311
```

Tabelle 1 (Fortsetzung)

```
0312
0313
0314
0315
0316    !
0317
0318
0319    1
0320 ZS
0321    S
0322    I
0323    E
0324
0325    H
0326    A
0327    B
0328    E
0329    N
0330
0331    G
0332    E
0333    W
0334    O
0335    N
0336    -
0337    N
0338    E
0339    N
0340    !
0341 ZS
0342    I
0343    C
0344    H
0345
0346    G
0347    R
0348    A
0349    T
0350    U
0351    L
0352    I
0353    E
0354    R
0355    E
0356 ZS
0357    I
0358    H
0359    N
0360    E
0361    N
0362
0363    Z
0364    U
0365    M
0366
0367    S
0368    I
0369    E
0370    G
0371 ZS
0372 TEX
0373 ZS
0374 ZS
0375 STP
0376 MRK
0377 8
0378 TEX
0379
0380
0381
0382
0383
0384
0385
0386    1
0387
0388
0389    !
0390 ZS
0391    S
0392    I
0393    E
0394
0395    H
0396    A
0397    B
0398    E
0399    N
0400
0401    L
0402    E
0403    I
0404    D
0405    E
0406    R
0407    V
0408    E
0409    R
0410    L
0411    O
0412    R
0413    E
0414    N
0415    !
0416 ZS
0417    M
0418    A
0419    N
0420
0421    M
0422    U
0423    S
0424    S
0425
0426    E
0427    S
0428
0429    E
0430    B
0431    E
0432    N
0433    K
0434    O
0435    E
0436    N
0437    N
0438    E
0439    N
0440 ZS
0441 TEX
0442 ZS
0443 ZS
0444 STP
0445 MRK
0446 9
0447 TEX
0448
0449
0450
0451
0452
0453
0454
0455
0456
0457
0458    !
0459
0460 DRU
0461    1
0462    0
0463 ZS
0464 TEX
0465 UP
0466 END
```

Tabelle 1 (Fortsetzung)

Tabelle 2

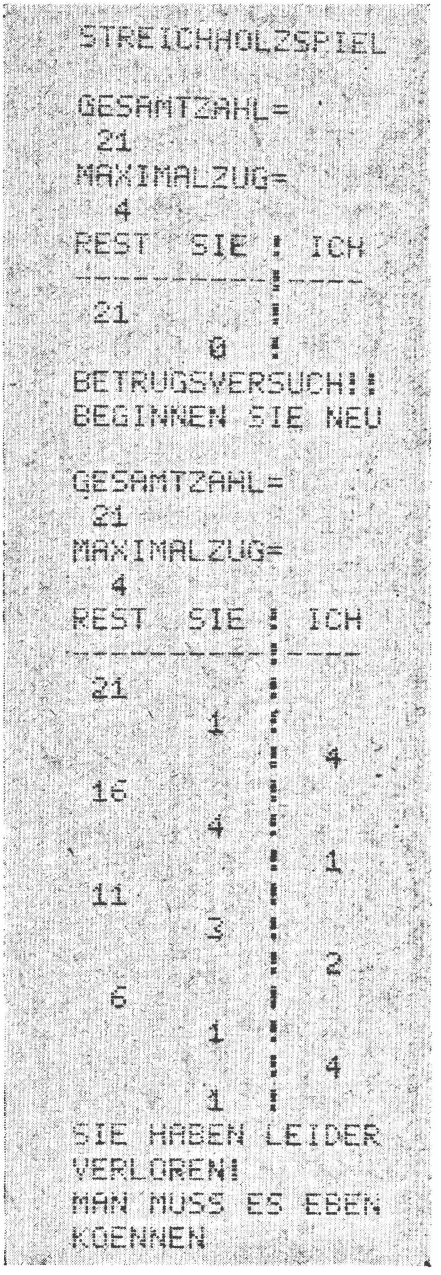


Tabelle 3

senszuwachs enttäuscht sein. Er mag sich mit den Worten von IMMANUEL KANT (1724–1804) trösten:
»Ein kleiner Anfang, der aber Epoche macht, indem er der Denkungsart eine

```
STREICHHOLZSPIEL

GESAMTZAHL=
 40
MAXIMALZUG=
  5
REST   SIE  !  ICH
------------!-----
 40         !
       5    !
            !   4
 31         !
       3    !
            !   3
 25         !
       3    !
            !   3
 19         !
       5    !
            !   1
 13         !
       1    !
            !   5
  7         !
       2    !
            !   4
       1    !
SIE HABEN LEIDER
VERLOREN!
MAN MUSS ES EBEN
KOENNEN
```

```
STREICHHOLZSPIEL

GESAMTZAHL=
 40
MAXIMALZUG=
  5
REST   SIE  !  ICH
------------!-----
 40         !
       3    !
            !   1
 36         !
       5    !
            !   5
 26         !
       1    !
            !   1
 24         !
       5    !
            !   3
 16         !
       3    !
            !   2
 11         !
       4    !
            !   1
  6         !
       5    !
            !   1
SIE HABEN GEWON-
NEN!
ICH GRATULIERE
IHNEN ZUM SIEG.
```

Tabelle 4

ganz neue Richtung gibt, ist wichtiger als die ganze unabsehliche Reihe von darauffolgenden Weiterungen der Kultur«. Gibt es Treffenderes, unsere Computergegenwart zu beschreiben?

Autor:

Dr. Hannes Gutzer
DDR 4090 Halle-Neustadt
Am Südpark 581/6
Zentralinstitut für Schweißtechnik Halle

Wie rechnet ein elektronischer Taschenrechner?

Wer schon mit unterschiedlichen Typen elektronischer Taschenrechner gearbeitet hat, der wird zu seinem Leidwesen festgestellt haben, daß man eine bestimmte Tastenfolge, die eine Aufgabe auf dem einen Rechner richtig löst, nicht einfach auf einen anderen Rechner übertragen darf. Die Tastenfolge, die auf dem einen Taschenrechner zu einem ganz bestimmten Ergebnis führt, kann – auf einen anderen Rechnertyp übertragen – ein ganz anderes Resultat nach sich ziehen.
Das beginnt schon bei ganz einfachen Aufgabenstellungen. So liefert beispielsweise die Tastenfolge

[2] [×] [3] [+] [4] [×] [5] [=]

bei vielen der gebräuchlichen Rechnertypen (so u. a. auch beim MR 411 und beim MR 610) *nicht* das vermutlich erwartete Ergebnis 26, sondern diese Rechner zeigen als Resultat die Zahl 50 an. Bei anderen Rechnertypen erhält man dagegen den Wert 26; und schließlich gibt es Rechner, auf denen man vergeblich versuchen wird, diese Tastenfolge einzutippen, weil diese Rechnertypen keine Taste mit einem Gleichheitszeichen besitzen.
Wie soll man sich da noch zurechtfinden, wenn man schon bei einer solch einfachen Aufgabe unterschiedliche Lösungswerte erhält? Kann man sich denn überhaupt noch darauf verlassen, daß der Taschenrechner auch das ausrechnet, was man von ihm erwartet?
Die unterschiedlichen Reaktionsweisen der Taschenrechner auf bestimmte Tastenfolgen hängen davon ab, welche *Rechnerlogik* dem jeweiligen Taschenrechnertyp zugrunde gelegt worden ist, und diese Rechnerlogik wiederum steht in engem Zusammenhang mit den beim Bau der Rechner verwendeten mikroelektronischen Bausteinen. Um richtig und sicher mit seinem Taschenrechner arbeiten zu können, ist es unbedingt erforderlich, daß man genau darüber Bescheid weiß, welche Rechenlogik der verwendete Rechner besitzt und wie die einzelnen Rechenoperationen innerhalb des Rechners in Abhängigkeit von der Rechnerlogik ablaufen. Aus diesem Grunde soll im folgenden versucht werden, einen Überblick über den *Ablauf der wichtigsten Rechenoperationen* bei den derzeit gebräuchlichen Rechnertypen zu geben.
Man unterscheidet

Rechner mit algebraischer Logik ohne Hierarchie und ohne Klammerstruktur,

Rechner mit algebraischer Logik ohne Hierarchie und mit Klammerstruktur,

Rechner mit algebraischer Logik mit Hierarchie sowie

Rechner mit Umgekehrter Polnischer Notation.

Die ersten beiden Rechnertypen werden häufig auch als ALO-Rechner bezeichnet, und die zuletzt genannten Rechner bezeichnet man auch als UPN-Rechner. Die bekannten Rechnertypen MR 411 und MR 610 und deren Abkömmlinge gehören zu der Gruppe der ALO-Rechner. UPN-Rechner werden insbesondere von der amerikanischen Firma HEWLETT-PACKARD produziert.

Am unproblematischsten ist die Handhabung der *Taschenrechner mit algebraischer Logik und mit Hierarchie.*

Bei diesen Rechnern werden die sogenannten *Vorrangregeln* der Arithmetik anerkannt, nach denen bei mehreren aufeinanderfolgenden Rechenoperationen festgelegt ist, daß Multiplikationen und Divisionen vorrangig vor den Additionen und Subtraktionen auszuführen sind (im Voksmund sagt man: »Punktrechnung geht vor Strichrechnung«) und daß, wenn von dieser Vereinbarung abgewichen werden soll, Klammern zu setzen sind. Die eingangs erwähnte Tastenfolge

[2] [×] [3] [+] [4] [×] [5] [=]

entspricht also bei diesem Rechnertyp der Berechnung von x aus der Aufgabe

$$2 \cdot 3 + 4 \cdot 5 = x$$

mit dem Ergebnis $x = 26$, was man bei unvoreingenommener Betrachtungsweise auch erwartet hätte.

Die Rechner mit algebraischer Logik und mit Hierarchie bereiten also bei ihrer Anwendung keine besonderen Schwierigkeiten, so daß auf sie nicht ausführlicher eingegangen zu werden braucht. Es soll jedoch darauf hingewiesen werden, daß es bedauerlicherweise nur sehr wenige Taschenrechner-Fabrikate gibt, denen die algebraische Logik mit Hierarchie zugrunde gelegt ist.

Die Mehrzahl der derzeit gebräuchlichen Typen von Taschenrechnern besitzt eine *algebraische Logik ohne Hierarchie*, und davon wiederum hat ein Teil auch Klammertasten.

Damit diese Rechner die durch Betätigen der verschiedenen Tasten des Tastenfeldes übermittelten Informationen (zu verarbeitende Daten und auszuführende Operationen) ordnungsgemäß ausführen können, enthält das Rechenwerk dieser Typen stets mindestens *zwei Register*, in denen die zu einer Verknüpfung durch eine Rechenoperation vorgesehenen Operanden zwischengespeichert werden und bei denen dann in einem dieser beiden Register das Ergebnis der durchgeführten Rechenoperation untergebracht wird. Diese beiden Register werden als X-Register und als Y-Register bezeichnet.

Zweistellige Rechenoperationen, das sind solche, bei denen aus zwei Operanden mit Hilfe einer Rechenoperation eine neue Größe, das Ergebnis, gebildet wird (also Addition, Subtraktion, Multiplikation, Division und das Potenzieren Y^X), laufen bei diesen Rechnern in den folgenden Teilschritten ab:

1. Schritt: Eingeben des ersten Operanden über die Zahlentastatur. Der Operand wird im X-Register gespeichert und kann zur Kontrolle im Display abgelesen werden.

2. Schritt: Eintasten der auszuführenden Rechenoperation

[+], [−], [×], [÷] oder [Y^X].

Der erste Operand wird dabei automatisch in das Y-Register übertragen, verbleibt dabei aber auch gleichzeitig noch im X-Register.

3. Schritt: Eingeben des zweiten Operanden über die Zahlentastatur (oder durch Rückruf aus einem Speicher). Dabei wird der im X-Register verbliebene erste Operand »überschrieben«, der zweite Operand befindet sich nun

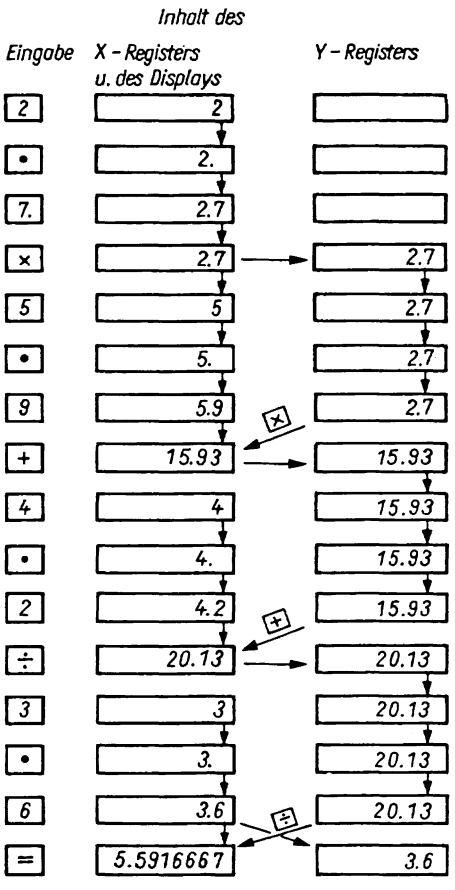


Bild 1

an seiner Stelle. Er kann auch im Display kontrolliert werden.
Die Verteilung der beiden Operanden auf die beiden Register ist demzufolge nunmehr wie folgt: erster Operand im Y-Register, zweiter Operand im X-Register. Stimmt der zweite Operand mit dem ersten überein, so kann dessen erneute Eingabe entfallen.[1]

[1] Die letzte Bemerkung trifft nicht bei allen Rechnertypen zu.

4. Schritt: Eintasten einer neuen Rechenoperation

[+], [−], [×], [÷] oder [Y^X]

bzw. Eintasten des Gleichheitszeichens [=]. Dabei wird vom Rechenwerk das Ergebnis der im zweiten Schritt vorbereiteten Rechenoperation ermittelt und ggf. gleichzeitig eine neue Rechenoperation vorbereitet. Wurde dabei eine der Tasten

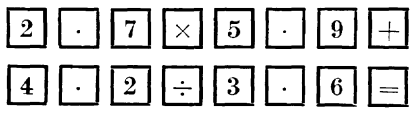
[+], [−], [×], [÷] oder [Y^X]

betätigt, so wird das Ergebnis der im zweiten Schritt vorbereiteten Rechenoperation sowohl im X- als auch im Y-Register gespeichert. Die in diesen beiden Registern vorher vorhandenen Zahlen gehen also verloren. Wurde hingegen zuletzt die [=]-Taste gedrückt, so wird das Ergebnis der Rechnung im X-Register gespeichert, während der bisherige Inhalt des X-Registers in das Y-Register übertragen wird. (Beim Betätigen der Gleichheitstaste bleiben demnach der zweite Operand sowie das Ergebnis der letzten Rechnung für den weiteren Rechenablauf erhalten.) – Das Ergebnis der Rechnung kann in jedem Falle im Display abgelesen werden.

Anmerkung: Es gibt Rechnermodelle, bei denen die hier beschriebenen Zahlentransporte zwischen den beiden Rechenregistern in einer etwas anderen Weise verlaufen. So bleibt beispielsweise bei einigen Rechnertypen nach dem Betätigen der Gleichheitstaste der bisherige Inhalt des Y-Registers erhalten. *Es wird daher dringend empfohlen, sich beim Erwerb eines neuen Taschenrechners genau Klarheit darüber zu verschaffen, welche Vorgänge sich bei den einzelnen Rechenoperationen zwischen den beiden Registern abspielen.* An Hand einfacher Beispiele ist dies ohne Schwierigkeiten möglich.

In Bild 1 werden die jeweiligen Zahlen-

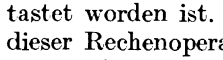
verschiebungen zwischen den beiden Rechenregistern für die Tastenfolge



ausführlich dargestellt.
Verfolgt man den Ablauf der einzelnen Rechenoperationen, so ist zu erkennen, daß mit dieser Tastenfolge der Ausdruck

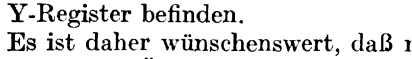
$$\frac{2{,}7 \cdot 5{,}9 + 4{,}2}{3{,}6}$$

berechnet wurde.
Man beachte, daß bei jeder Betätigung einer Operationstaste für zweistellige Rechenoperationen diejenige Operation ausgeführt wird, die *vorher* eingetastet worden ist. An der Ausführung dieser Rechenoperation sind stets diejenigen Operanden beteiligt, die sich zu diesem Zeitpunkt gerade im X- und im Y-Register befinden.
Es ist daher wünschenswert, daß man stets den Überblick darüber behält, welche Operanden sich in den beiden Rechenregistern befinden.
Bei *einstelligen Rechenoperationen* wird aus einem Operanden aufgrund einer bestimmten Rechenvorschrift ein neuer Zahlenwert, das Ergebnis der Rechenoperation, bestimmt. Beispiele für einstellige Rechenoperationen sind demzufolge das Quadrieren, das Quadratwurzelziehen, die Bestimmung von Logarithmen, von trigonometrischen Funktionswerten usw.
Für die Ausführung einer einstelligen Rechenoperation mit Hilfe eines Taschenrechners gilt die folgende Regel:
Wird eine zu einer einstelligen Rechenoperation gehörende Taste betätigt, so ermittelt der Rechner den zugehörigen Funktionswert aus dem im *X-Register* befindlichen Zahlenwert. Das Ergebnis erscheint im Display und wird im X-Register gespeichert. Der Inhalt des Y-Registers bleibt dabei unverändert.
Daraus folgt:
Will man das Ergebnis einer einstelligen Rechenoperation mit Hilfe eines Taschenrechners ermitteln, so muß man entgegen der in der Mathematik gebräuchlichen Schreibweise und Sprechweise zuerst das Argument der Funktion eingeben und darf erst danach auf die entsprechende Funktionstaste drücken. Ein Betätigen der Gleichheitstaste ist nicht erforderlich, da die Berechnung des Funktionswertes bereits durch den Druck auf die zugehörige Funktionstaste ausgelöst wird.
Den Funktionswert ln 37 erhält man also, wenn man die Tastenfolge

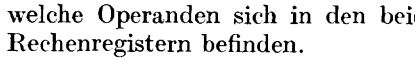


drückt.

Aus den Bildern 2 und 3 geht hervor, daß die beiden einander sehr ähnlichen Tastenfolgen



und



unterschiedliche Ergebnisse liefern. Im ersten Fall ergibt sich der Wert von

$$4{,}7 + \sqrt{3{,}2} = 6{,}4888544,$$

hingegen erhält man im zweiten Fall

$$\sqrt{4{,}7 + 3{,}2} = 2{,}8106939.$$

Taschenrechner mit *algebraischer Logik ohne Hierarchie*, die *Klammertasten* besitzen, müssen neben den beiden bereits erwähnten Rechenregistern X und Y noch *weitere Hilfsregister*, sogenannte Kellerspeicher besitzen, die mit K1, K2, ... bezeichnet werden sollen. Von der Anzahl der vorhandenen Kellerspeicher eines Rechners hängt es ab, wieviel Klammerebenen ineinander geschachtelt werden dürfen. Besitzt ein

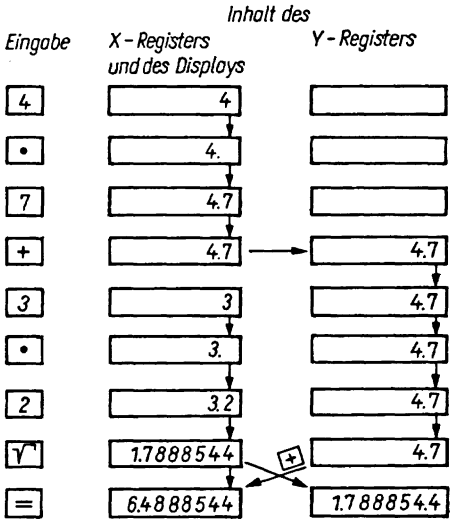


Bild 2

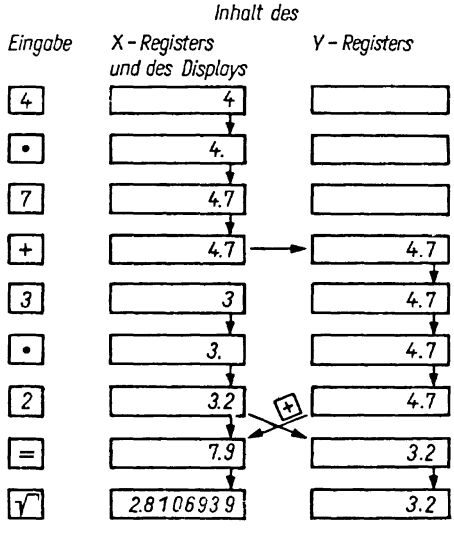


Bild 3

Rechner n Kellerspeicher, so können insgesamt n Klammern ineinander geschachtelt werden.

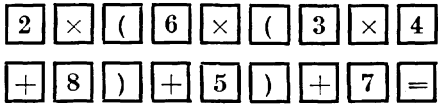
Im allgemeinen laufen die Rechenoperationen bei derartigen Rechnern genau so ab, wie es bisher beschrieben wurde. Lediglich in dem Fall, daß die Taste mit der öffnenden Klammer gedrückt wird, wird der Inhalt des Y-Registers in den Kellerspeicher mit der niedrigsten Nummer transportiert, und die bisherigen Inhalte des X- und des Y-Registers werden gelöscht. Sollte in dem Kellerspeicher mit der niedrigsten Nummer bereits ein Zahlenwert vorhanden gewesen sein, so wird dieser in den Kellerspeicher mit der nächsten Nummer gebracht, und falls dort ebenfalls bereits ein Wert gespeichert gewesen sein sollte, wird dieser wiederum in den Kellerspeicher der nächst höheren Ebene transportiert. – Genau umgekehrt verlaufen die Datentransporte, wenn die Taste mit der schließenden Klammer gedrückt wird. In diesem Fall wird der Wert aus dem Kellerspeicher mit der niedrigsten Ordnungsnummer in das Y-Register übertragen, während die Inhalte der übrigen Kellerspeicher jeweils um eine Ebene nach unten rücken. Es gilt also das Prinzip

last in — first out (LIFO) :

was zuletzt in den Kellerspeicher gebracht wurde, wird als erstes wieder aus ihm entlassen.

Auf diese Weise können die Inhalte der einzelnen Klammerebenen für sich berechnet und auch systematisch gespeichert werden, damit sie dann, wenn die jeweilige Klammerebene wieder geschlossen wird, für die weitere Berechnung zur Verfügung stehen.

In Bild 4 ist ein solcher Datentransport für einen Rechner dargestellt, der zwei Klammerebenen (und demzufolge zwei zusätzliche Kellerspeicher K1 und K2) besitzt, für die Tastenfolge,



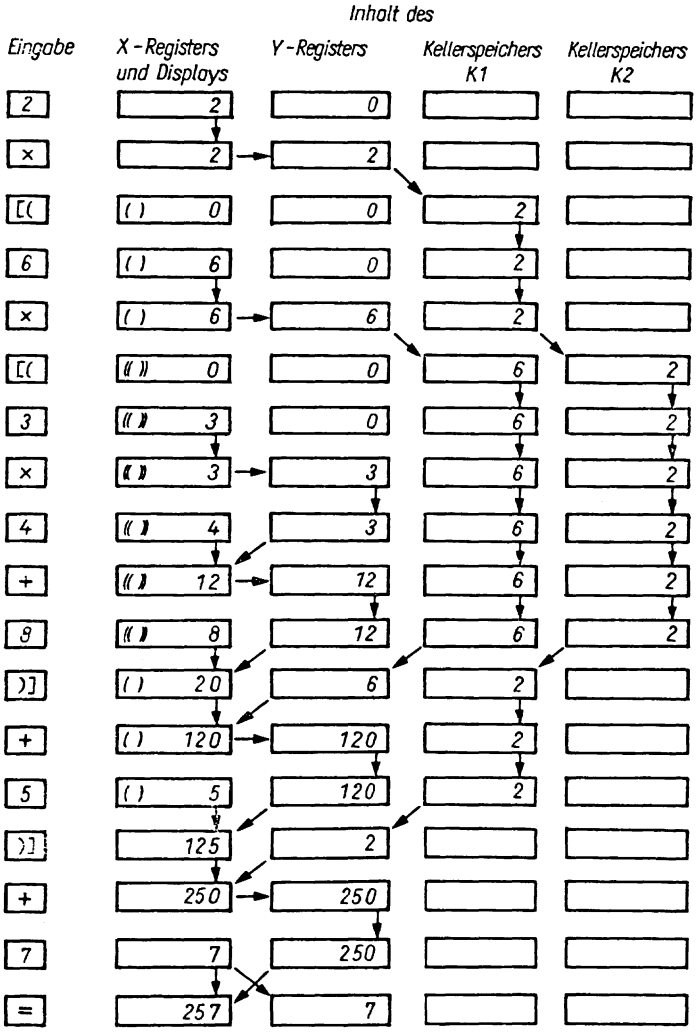


Bild 4

dargestellt. Es ist ersichtlich, daß mit Hilfe dieser Tastenfolge die Aufgabe

$$2 \cdot [6 \cdot (3 \cdot 4 + 8) + 5] + 7 = 257$$

gelöst wurde.

Das Prinzip der Kellerspeicher liegt auch den *Taschenrechnern mit Umgekehrter Polnischer Notation* zugrunde.

Die vom polnischen Mathematiker Lukasiewicz eingeführte Umgekehrte Polnische Notation ist eine neuartige Schreibweise für mathematische Ausdrücke, bei der das Setzen von Klammern vermieden werden kann. Sie legt fest, daß *zuerst* die an einer Rechenoperation beteiligten *Operanden* notiert

werden und daß *danach* erst aufgeschrieben wird, *was* mit diesen Operanden *geschehen* soll.
So würde beispielsweise die Notation

$a\ b +$

bedeuten, daß die beiden Operanden a und b durch die mit + gekennzeichnete Rechenoperation miteinander zu verknüpfen sind, daß also in der herkömmlichen Schreibweise ausgedrückt die Summe

$a + b$

ermittelt werden soll. Entsprechend müßte die Notation

$a\ b\ +\ c\ d\ +\ \times$

wie folgt gedeutet werden: Zuerst ist die Summe von a und b zu bilden, dann die Summe von c und d, und schließlich sollen diese beiden Summen miteinander multipliziert werden. Die obige Notation entspricht also der üblichen Schreibweise von

$(a + b) \cdot (c + d)$.

Auf der Basis dieser neuartigen Notation mathematischer Ausdrücke arbeiten die Taschenrechner mit Umgekehrter Polnischer Notation. Sie besitzen mehrere Kellerspeicher (hier *Stack-Register* genannt), zwischen denen die eingegebenen Operanden je nach der gewählten Tastenfolge hin- und hertransportiert werden.
Der Ablauf von Rechnungen einschließlich der dabei auftretenden Datentransporte zwischen den einzelnen Stack-Registern läßt sich wie folgt kurz beschreiben:

1. Jeder Taschenrechner mit Umgekehrter Polnischer Notation besitzt *mehrere* (meist 3 oder 4) sogenannte *Stack-Register*, die der Reihe nach als X-, Y-, Z- und T-Register bezeichnet werden sollen.

2. Die *Eingabe* einer Zahl erfolgt stets in das *X-Register*.

3. Werden *mehrere Zahlen* unmittelbar nacheinander eingegeben, so rücken die bisherigen Registerinhalte jeweils um *eine Stufe nach oben*, d. h., der bisherige Inhalt des X-Registers wandert in das Y-Register, dessen bisheriger Inhalt geht in das Z-Register über usw. Der bisherige Inhalt des letzten Stack-Registers (in unserem Fall des T-Registers) geht bei einer weiteren Zahleneingabe verloren.

4. Bei *einstelligen Rechenoperationen* wird der auf Grund der gedrückten Funktionstaste zu berechnende Funktionswert aus der Zahl ermittelt, die sich im *X-Register* befindet. Das Ergebnis dieser einstelligen Rechenoperation erscheint dann im X-Register und im Display.

5. Bei *zweistelligen Rechenoperationen* werden die jeweiligen Operanden *dem Y- und dem X-Register* entnommen. Das Ergebnis der Rechnung wird im X-Register gespeichert. Die Inhalte der höheren Register rücken dabei jeweils um eine Stufe *nach unten* (d. h., der Inhalt des Z-Registers geht in das Y-Register über, der des T-Registers in das Z-Register).

Das Verhalten des T-Registers ist dabei von Rechnertyp zu Rechnertyp verschieden. Bei manchen Rechnern bleibt der bisherige Inhalt des T-Registers erhalten, bei anderen dagegen entsteht bei derartigen »Abwärtstransporten« im T-Register der Wert Null.

Da sich alle Rechenoperationen entweder nur im X-Register (einstellige Operationen) oder nur zwischen dem X- und dem Y-Register (zweistellige Operationen) abspielen und die Ergebnisse aller Operationen im X-Register gespeichert werden und damit auch sofort im Display ablesbar sind, sobald eine Operationstaste gedrückt wurde, benötigen Rechner mit Umgekehrter

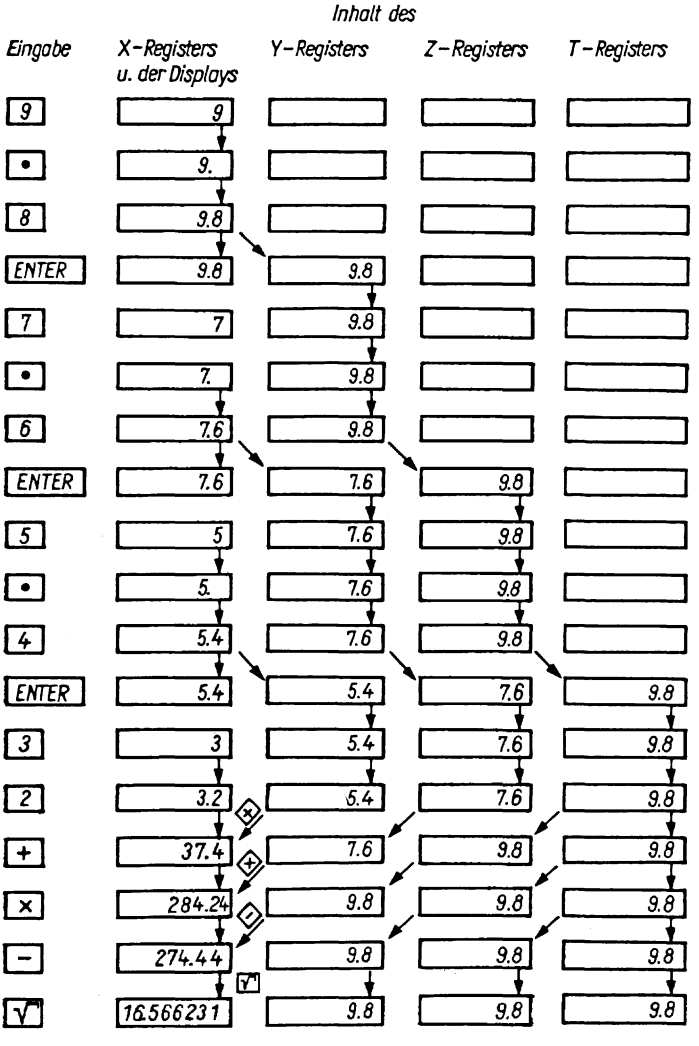

Inhalt des
Eingabe
X-Registers u. der Displays
Y-Registers
Z-Registers
T-Registers
9
•
8
ENTER
7
•
6
ENTER
5
•
4
ENTER
3
2
+
×
−
√
9
9.
9.8
9.8
7
7.
7.6
7.6
5
5.
5.4
5.4
3
3.2
37.4
284.24
274.44
16.566231
9.8
9.8
9.8
9.8
7.6
9.8
7.6
9.8
5.4
7.6
9.8
5.4
7.6
9.8
3.2
7.6
9.8
37.4
9.8
9.8
284.24
9.8
9.8
274.44
9.8
9.8
16.566231
9.8
9.8

Bild 5

Polnischer Notation *keine Ergebnistaste* [=].

Dafür müssen jedoch – zumindest bei den zweistelligen Operationen – häufig mehrere Zahlen unmittelbar nacheinander eingegeben werden. Da nun jede Zahl ziffernweise eingetastet werden muß, muß eine Möglichkeit dafür geschaffen werden, daß der Rechner erkennen kann, wann die eine Zahleneingabe abgeschlossen ist und die nächste beginnt. Dies geschieht durch die sogenannte ENTER-Taste

[ENTER] oder [ENTER ↑] oder [↑].

(Das Fehlen einer [=]-Taste und das Vorhandensein einer ENTER-Taste ist ein untrügliches Kennzeichen für einen Rechner mit Umgekehrter Polnischer Notation.)

Für die Zahleneingabe bei Taschenrechnern mit Umgekehrter Polnischer Notation gilt die folgende Regel:

Die *Eingabe einer Zahl* in einen Rechner mit Umgekehrter Polnischer Notation wird abgeschlossen, wenn nach dem Eintasten der letzten Ziffer *eine* der folgenden Tasten betätigt wird:

- die ENTER-Taste,
- eine beliebige Operations- oder Funktionstaste,
- eine Speichertaste oder
- eine Löschtaste

In Bild 5 sind die Registerverschiebungen veranschaulicht, die sich bei der Tastenfolge

[9] [·] [8] [ENTER] [7] [·] [6]
[ENTER] [5] [·] [4] [ENTER]
[3] [2] [+] [×] [−] [|⁻]

ergeben. Es ist nach dem bisher Gesagten leicht nachzuprüfen, daß es sich bei dieser Tastenfolge um die Lösung der Aufgabe

$$\sqrt{(32 + 5{,}4) \cdot 7{,}6 - 9{,}8}$$

handelt.

Autor:
Prof. Dr.-Ing. Hans Kreul
DDR 8800 Zittau
Bruno-Schröter-Str. 1
a. o. Professor an der Ingenieurhochschule Zittau
Abteilung EDV und Rechentechnik

Rechentechnische Begriffe für den Laien erklärt

Byte

Unter einem Byte versteht man die Zusammenfassung von acht Binärstellen zu einer Einheit. Im Speicher eines Digitalrechners stellt ein Byte die kleinste addressierbare (direkt aufrufbare) Einheit dar. Aus diesem Grunde wird die Speicherkapazität einer EDVA meist in Byte (B), Kilobyte (kB) oder Megabyte (MB) angegeben. Deutet man jede mögliche Belegung der acht Binärstellen eines Bytes als Dualziffer (0 oder 1), so ergeben sich die Dualzahlen

$0000\ 0000_{(2)}$ bis $1111\ 1111_{(2)}$,

die den Dezimalzahlen 0 bis 255 entsprechen. In einem Byte können mithin insgesamt 256 unterschiedliche Informationen gespeichert werden.

Der Heimcomputer robotron Z 9001

1. Was ist ein Heimcomputer?

Als Heimcomputer werden kleine, im allgemeinen sehr preiswerte Rechner bezeichnet, die in jüngster Zeit auch als Konsumgut in den Heimbereich Einzug halten. Sie können über ein Antennenkabel mit einem Fernsehgerät verbunden werden und sind dann sehr einfach als Computer oder auch zur Durchführung von Fernsehspielen nutzbar.

Heimcomputer besitzen eine Tastatur, über die Zahlen, Buchstaben und Zeichen eingegeben werden können, die der Rechner entsprechend ihrer Bedeutung verarbeitet. Auf dem Fernsehschirm werden alle dem Computer eingegebenen Befehle und Daten ebenso angezeigt wie die berechneten Ergebnisse und sonstige Informationen.

Für einfache Nutzungsfälle wird ein Heimcomputer in der Programmiersprache BASIC programmiert. Diese Sprache existiert in sehr vielen Varianten, die auf den verschiedenen Computer-Typen jedoch nur geringfügig voneinander abweichen. In den meisten Fällen wurde eine Anpassung der Sprache an die Möglichkeiten des konkreten Rechners vorgenommen.

Mit BASIC können, ähnlich wie beim Taschenrechner, einfache Rechenoperationen ausgeführt werden, wobei Aufgabe und Ergebnis am Bildschirm sichtbar sind. Viel wichtiger ist aber die Tatsache, daß auch Programme eingegeben und anschließend beliebig oft abgearbeitet werden können.

Zur Speicherung dieser Programme über einen längeren Zeitraum dient dem Heimcomputer ein gewöhnliches (Audio-) Kassettenmagnetbandgerät, welches über ein Diodenkabel an den Rechner anschließbar ist. Durch spezielle Kommandos lassen sich die in den Rechner eingegebenen Programme auf Magnetband ausgeben, dort speichern und später wieder einlesen.

An viele Geräte können außerdem noch einfache Drucker angeschlossen werden, so daß auch ein Ausdrucken der Programme und Berechnungsergebnisse möglich wird. Häufig sind Spielhebel zur Steuerung von Bewegungsabläufen anschließbar.

BASIC-Programme für den Heimcomputer kann man auf Magnetbandkassetten kaufen oder aber auch selbst erstellen. Diese Programme können sehr unterschiedlichen Inhalts sein. Neben einfachen Rechenprogrammen für Aufgaben aus dem Schulbereich sind wissenschaftlich-technische Berechnungen für ingenieurtechnische Anwendungen ebenso denkbar wie Spielprogramme, z.B. in Form von Logik- und Reaktionsspielen, oder Programme zur Speicherung von Informationen und Statistiken.

Bild 1. Heimcomputer mit Fernsehgerät und Kassettengerät

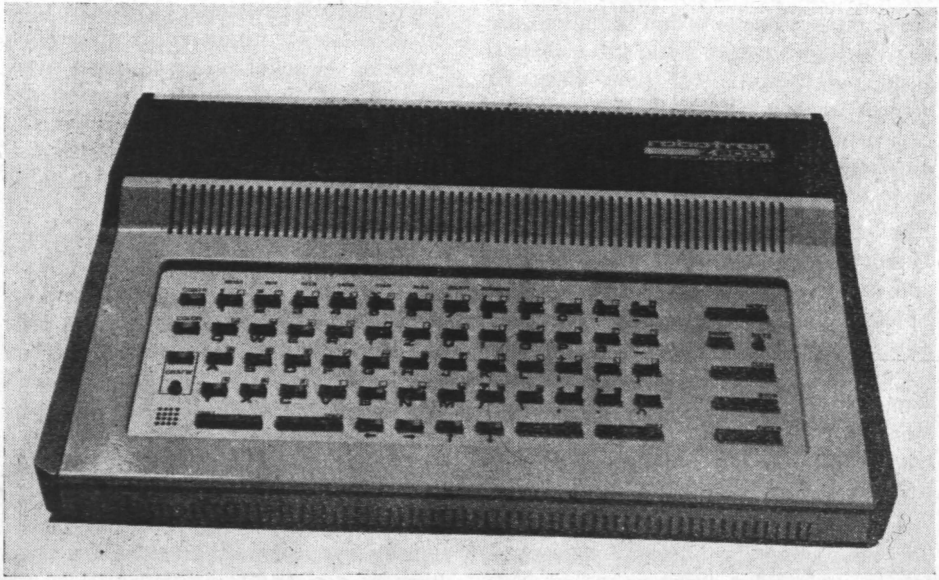


Bild 2. Tastatur

Besondere Bedeutung haben Heimcomputer für die Unterstützung der Ausbildung, zum Vertrautwerden mit der Rechentechnik. So lassen sich viele Unterrichtsaufgaben mit einem Heimcomputer anschaulich lösen und Gesetzmäßigkeiten aus vielen Fachgebieten demonstrieren.

Nicht zuletzt stehen dem technisch interessierten Amateur bei einigen Heimcomputern viele Möglichkeiten offen, selbstgebaute elektronische Baugruppen an den Computer anzuschließen und durch ein entsprechendes Programm zu steuern.

Der Heimcomputer robotron Z 9001 entspricht in seiner Grundgestaltung den oben geschilderten Prinzipien.

Seine speziellen technischen Daten und Eigenschaften sind nachfolgend erläutert:

- 49 Tasten für die Eingabe von Groß- und Kleinbuchstaben, Ziffern und Zeichen
  16 Tasten zur Kommandoausführung bzw. zur Kursorsteuerung (vgl. Bild 2)
- 128 alphanumerische bzw. Steuerzeichen
  128 Grafikzeichen (teilweise von der Tastatur aus ansprechbar)
- Darstellung von 24 Zeilen und 40 Spalten auf dem Bildschirm
- 16-KByte-RAM-Speicher (für den Anwender verfügbar). Damit sind etwa 800 bis 1000 BASIC-Programmbefehle speicherbar.

Das Grundgerät bietet die Möglichkeit, 4 Zusatzmodule zu stecken, die funktionelle bzw. Speicher-Erweiterungen beinhalten können (vgl. Bild 3). Damit ist es möglich, den BASIC-Interpreter sowohl von einer Magnetbandkassette zu laden, als auch als Modul zusätzlich zu stecken.

Neben dem BASIC-Modul gibt es unter anderem folgende Erweiterungsmöglichkeiten, die im Handel als Zubehör angeboten werden:

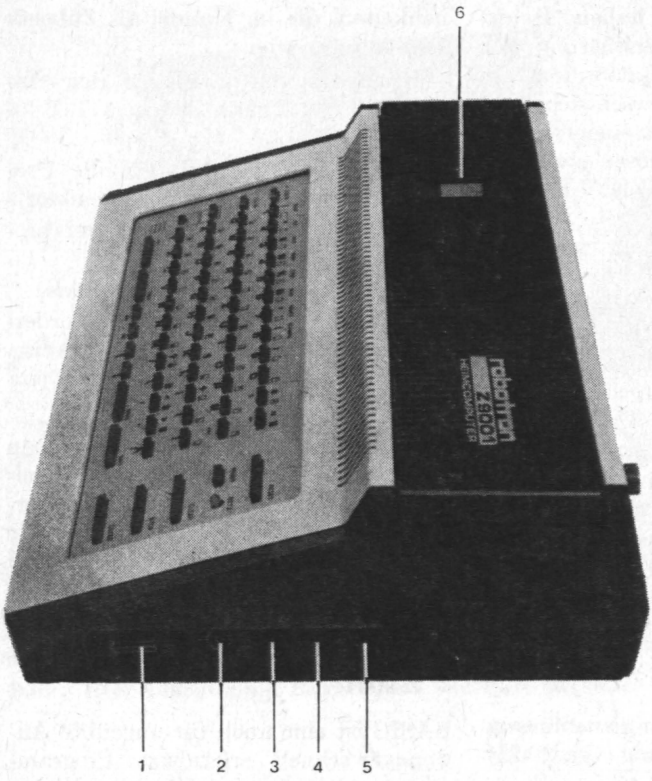
- Drucker-Modul (z.B. für den Anschluß des Thermodruckers TD 40/K 6303)
- ASSEMBLER-Modul (für die Programmierung in K1520-Assembler)
- Musik-Modul (3 Oktaven, frei programmierbar)
- Speicher-Modul (16-KByte-RAM)
- Farb-Modul (Je 8 Farben für Vordergrund und Hintergrund wahlweise. Dieser Modul wird in das Gerät eingebaut).

Über weitere Steckmöglichkeiten an der rechten Geräteseite können Spielhebel (für Bildschirmspiele) und verschiedene andere elektronische Geräte oder Baugruppen angeschlossen und gegebenenfalls gesteuert werden.

2. Die Programmiersprache BASIC des Z 9001

BASIC ist eine auch für ungeübte Anwender schnell erlernbare Programmiersprache, die vor allem den Dialog mit dem Rechner, z.B. die Einflußnahme auf die Programmabarbeitung, sehr gut unterstützt.

Mit Hilfe des BASIC des Z 9001 können sowohl einfache mathematische Operationen (im sogenannten „Taschenrechnermodus") sofort ausgeführt, als auch Programme eingegeben, abgearbeitet und auf Magnetbandkassette gespeispeichert werden. Auch die Ausgabe von Programmen, Rechenergebnissen und anderen Informationen auf einen Drucker ist möglich. Ebenso können die Heimcomputer-spezifischen Ergänzungsmodule (Farbmodul, Musikmodul, Spielhebel u.a.) sowie der für die Gestaltung von Spielen bedeutsame Grafik-Zeichensatz (128 spezielle Grafiksymbole) vom BASIC einfach angesprochen werden. Weiterhin lassen



1 – Anschluß für 8 E/A-Kanäle
2 – H/F-Anschluß
3 – Anschluß für Tonband
4 – Spielhebel 1
5 – Spielhebel 2
6 – 4 Modulsteckplätze über K 1520-BUS (abgedeckt)

Bild 3. Ansicht des Z 9001 mit Modulsteckplätzen und E/A-Anschlußbuchsen (Modellfoto)

sich digitale Informationen von angeschlossenen Meßgeräten abrufen und weiterverarbeiten und auch Steuersignale an entsprechende Geräte übertragen.
Das BASIC des Z 9001 ist in seinem Aufbau (Befehls- und Kommando-Vorrat, syntaktische Struktur) dem international weitverbreiteten Standard angeglichen, so daß eine Übertragung von Programmen vieler anderer Rechner auf den Z 9001 in der Regel leicht fällt. Wir wollen uns nun einen Überblick über die wichtigsten Anweisungen und Kommandos und ihre Wirkungsweise verschaffen.

Einfache Rechenoperationen

Wie bereits erwähnt, können im BASIC ähnlich wie mit einem Taschenrechner einfache mathematische Rechenoperationen ausgeführt werden.
Dazu stehen die Grundoperationen +, −, ×, / und ∧ (Potenzsymbol), sowie die Standardfunktionen

ABS (X) – Absoluter Betrag
ATN (X) – Arcustangens
COS (X) – Cosinus (X in Bogenmaß)
EXP (X) – Exponentialfunktion e^x
INT (X) – Ganzer Teil
LN (X) – Natürlicher Logarithmus
RND (X) – Zufallszahl
SGN (X) – Signumfunktion

SIN (X) – Sinus (X in Bogenmaß)
SQR (X) – Quadratwurzel
TAN (X) – Tangens (X in Bogenmaß)
PI – Konstante π

zur Verfügung. Diese Funktionen können beliebig auf numerische Größen (Zahlen, Variablen oder Ausdrücke) angewendet und miteinander verknüpft werden.
Dabei entsteht jeweils ein »numerischer Ausdruck«, zum Beispiel

5 * 7 + SIN (PI * 45/180)

oder

−7 * (4 + EXP (2)).

Will man das Ergebnis einer solchen Rechenoperation am Bildschirm anzeigen, so muß dem Ausdruck das Schlüsselwort PRINT (»Drucke«) vorangestellt werden.
Die Eingabe des gesamten Befehls erfolgt zeichenweise über die Tastatur, wobei jede Befehlszeile durch Betätigung der ENTER-Taste abgeschlossen wird.
Der Rechner arbeitet die Aufgabe sofort nach ENTER ab und zeigt das Ergebnis an.

Beispiel: Es wird eingegeben

PRINT 5 * 7 + 8 * 6

ENTER

Auf dem Bildschirm sind dann Aufgabe und Ergebnis in folgender Weise dargestellt

```
PRINT 5*7+8*6
83
OK
>■
```

Ab der Position des Kursors (■) kann nun die Eingabe einer weiteren Aufgabe beginnen.

Zur Verkürzung der Eingabe kann das Schlüsselwort PRINT auch durch ein Fragezeichen (»?«) ersetzt werden. Eine Folge von Aufgaben und Ergebnissen ergibt am Bildschirm dann

```
?SIN(PI*45/180)
 .707107
OK
?-5*7
-35
OK
?0.5*LN(10)/(5-3*5)
-.115129
OK
>■
```

wobei die Zeilen 1, 4 und 7 jeweils über die Tastatur eingegeben und mit ENTER abgeschlossen werden.

Variablen und Bezeichnungen

Bevor wir zur Erläuterung einiger einfacher BASIC-Befehle übergehen können, muß noch vorangestellt werden, daß im BASIC des Z 9001 die Verwendung von zwei Variablen-Typen möglich ist.
Zunächst gibt es die numerischen Variablen, die jeweils einen konkreten Zahlenwert (4-Byte-Gleitkommazahl) beinhalten, der der Variablen durch eine Eingabe- oder eine Ergibtanweisung zugewiesen wird. Außerdem können solche Variablen auch mit den im voranstehenden Abschnitt erläuterten Rechenoperationen zu numerischen Ausdrucken verknüpft werden. Im Gegensatz zur üblichen Schreibweise in der Mathematik, in der Variablen jeweils durch einen Buchstaben (z.B.: x, y, a, b, i, j, ...) bezeichnet werden, kann man im BASIC des Z 9001 bis zu zwei Zeichen (Großbuchstabe und Großbuchstabe bzw. Ziffer) zur Bezeichnung nutzen. Dadurch ist es möglich, aus-

reichend viele Variablen zu unterscheiden.

Beispiel: A, A1, AS, XX, R7, K9

Neben den numerischen Variablen gibt es beim Z 9001 noch den Typ der Zeichenkettenvariablen (auch String-Variablen), die beliebigen Text bzw. Zeichenfolgen (Groß- und Kleinbuchstaben, Sonderzeichen, Grafikzeichen) bis zu 255 Zeichen beinhalten können. Die Variablenbezeichnung erfolgt wie für numerische Variablen, nur wird noch das Zusatzzeichen »$« (Dollarzeichen) angefügt.

Beispiele: B$, A1$, Z$, T5$, KK$

Eine Wertzuweisung für alle Variablen erfolgt wie üblich durch das Gleichheitszeichen, wobei auf der rechten Seite auch Ausdrücke stehen dürfen.

Beispiele: A=5.7
C1=−12.6*4
XX=3*C1
A$=»WORT1«
ZK$=»Zeichenkette«

Außerdem können mittels der DIM-Anweisung Variablen-Felder (Matrizen) für numerische und Zeichenkettenvariable vereinbart werden:

DIM X (3, 4) definiert die Matrix

X (0, 0)..... X(0, 4)
⋮ ⋮
X (3, 0)..... X(3, 4)

und stellt Speicherplatz für sie bereit. Die Elemente X(I, J), I=0, ..., 3, J=0, ..., 4 können dann im Programm wie einfache numerische Variable benutzt werden. Ebenso kann durch

DIM T$ (2, 4)

eine Matrix vereinbart werden, deren Elemente Zeichenketten sind.

Einfache Programme

Nach den Vorbemerkungen wollen wir jetzt einige einfache Programme in BASIC formulieren und ihre Abarbeitung am Z 9001 beschreiben.
Dazu muß man wissen, daß jede Programmzeile mit einer Zeilennummer beginnt und eine oder mehrere Anweisungen (durch Doppelpunkt getrennt) enthält. Bei der Neuerstellung eines Programms wird jede Programmzeile über die Tastatur eingegeben und mit ENTER abgeschlosssen. Danach wird sie vom Rechner unmittelbar in den Programmspeicher übernommen.
Nach beendeter Eingabe können die Programme durch das Kommando »RUN« gestartet werden. Hierzu steht die Taste RUN zur Verfügung, oder es wird

RUN ENTER

eingetippt. Die Abarbeitung der Programme erfolgt, wenn nicht anders angewiesen, in der Reihenfolge der Numerierung.
Einfache Programme bestehen nur aus Eingabe-, Berechnungs- und Ausgabe-Anweisungen, die durch die Schlüsselworte

INPUT – Eingabeanweisung
LET – Berechnungs- oder Ergibtanweisung (dieses Schlüsselwort kann auch entfallen)
PRINT – Ausgabeanweisung

charakterisiert werden.

Beispiel: Soll der Mittelwert von drei Zahlen berechnet werden, so genügt folgendes Programm:

```
10 INPUT »3 Zahlen eingeben :« ;
   A,B,C
20 LET M=(A+B+C)/3
30 PRINT »Mittelwert :« ; M
```

Die Abarbeitung dieses einfachen Beispiels erfolgt in folgender Weise:

Das Programm wird durch [RUN] gestartet. Durch die Anweisung 10 wird dann der Text

```
3 Zahlen eingeben: ■
```

am Bildschirm angezeigt.
Der Rechner wartet jetzt auf die Eingabe der 3 Zahlen A, B, C, die durch Komma getrennt werden müssen, z. B.:

5, 7, 12 [ENTER]

Danach wird gemäß Zeile 20 der Mittelwert M berechnet und mit Hilfe der Anweisung 30 am Bildschirm ausgegeben. Insgesamt entsteht folgendes Bild:

```
3 Zahlen eingeben: 5, 7, 12
Mittelwert       : 8
OK
>■
```

Durch »OK« wird das Ende der Programmabarbeitung angezeigt.

Diese kleine Rechnung läßt sich sicher auch auf einem Taschenrechner sehr schnell ausführen. Wollen wir aber zum arithmetischen Mittelwert gleichzeitig noch das geometrische und quadratische Mittel bestimmen, so ist das im BASIC wesentlich einfacher möglich.
Zur Realisierung dieser Programmerweiterung wird zusätzlich zum bereits gespeicherten Programm folgendes eingegeben:

a) Zur Berechnung der gesuchten Werte:

```
21 MG=(A*B*C)^(1/3)
```
[ENTER]

```
22 MQ = SQR ((A * A + B * B +
   C * C)/3)
```
[ENTER]

b) Zur Ausgabe der Werte:

```
40 PRINT »Geometr. Mittel: « ; MG
```
[ENTER]

```
50 PRINT »Quadrat.Mittel: « ; MQ
```
[ENTER]

Die Ausführung des jetzt im Rechner gespeicherten Programms nach [RUN] liefert am Bildschirm

```
3 Zahlen eingeben: 5, 7, 12
Mittelwert:        8
Geometr.Mittel: 7.48887
Quadrat.Mittel: 8.52448
OK
>■
```

Selbstverständlich kann dieses Programm beliebig oft abgearbeitet werden, wobei nach [RUN] stets nur die 3 Zahlenwerte einzugeben sind.

BASIC-Kommandos

Zur Eingabe, Abarbeitung und Änderung der BASIC-Programme stehen Kommandos zur Verfügung, die ebenfalls mit [ENTER] abgeschlossen und anschließend vom Rechner sofort ausgeführt werden. Für einige Kommandos stehen Tasten zur Verfügung, die eine sofortige Ausführung dieser Kommandos bewirken (z. B. für [RUN], [STOP], [LIST], [CONT], [PAUSE] – vgl. Tastatur). Die Handhabung der wichtigsten Kommandos wollen wir kurz beschreiben.

LIST

Mit dem LIST-Kommando kann ein Programm ausgegeben (»gelistet«) wer-

den. Dies erfolgt beim Heimcomputer standardmäßig auf dem Bildschirm. Wird das Programm zur Mittelwertberechnung wie S.56 eingegeben und ergänzt, so wird es nach

LIST [ENTER]

am Bildschirm wie folgt angezeigt:

```
10 INPUT »3 Zahlen eingegeben:«,
   A, B, C
20 LET M=(A+B+C)/3
21 MG=(A*B*C) (1/3)
22 MQ=SQR
   ((A*A+B*B+C*C)/3)
30 PRINT »Mittelwert      :«;M
40 PRINT »Geometr. Mittel:«;MG
50 PRINT »Quadrat. Mittel :«;MQ
OK
>■
```

Zu Auslösung des Kommandos kann ebenso die [LIST]-Taste gedrückt werden.

EDIT

Durch das EDIT-Kommando wird die Änderung von Programmen einfach möglich. Nach

EDIT 20 [ENTER]

wird vom beschriebenen Mittelwertprogramm die Zeile mit der Nummer 20 angezeigt, also

```
20 LET M=(A+B+C)/3 ■
```

Nun kann diese Zeile mit Hilfe der Kursortasten [→], [←] sowie der Taste zum Steichen und Einfügen von Zeichen (Taste DEL/INS) korrigiert werden, z.B. kann also das überflüssige Schlüsselwort LET herausgestrichen werden. Nach [ENTER] wird die korrigierte Zeile wieder in das Programm übernommen und anschließend Zeile 21 zur Änderung bereitgestellt.
Der EDIT-Modus kann durch [STOP] verlassen werden.

CSAVE/CLOAD

Da man vor allem längere Programme nicht mehrmals in den Rechner eintippen möchte, sollte man diese auf einer Magnetbandkassette speichern. Mit dem Kommando

CSAVE »PROGR1« [ENTER]

erfolgt die Speicherung des gesamten BASIC-Programmes auf der Kassette unter dem Namen »PROGR1«. Als Name kann hierbei jede Zeichenkette mit maximal 8 Zeichen benutzt werden.
Zu einem späteren Zeitpunkt läßt sich das gespeicherte Programm nach Positionierung des Magnetbandes vor den Programmanfang durch

CLOAD »PROGR1« [ENTER]

wieder in den Rechner laden und dann abarbeiten.
Der Vollständigkeit halber sei abschließend noch der komplette Kommandosatz des Z 9001 angeführt:

AUTO	– Automatische Zeilennumerierung bei Eingabe
BYE	– Verlassen des BASIC-Interpreters
CLEAR	– Löschen aller Variablen
CONT	– Fortsetzung unterbrochener Programme
CSAVE	– Abspeichern von Programmen auf Magnetband

CLOAD	– Laden von Programmen vom Magnetband
DELETE	– Löschen von Programmzeilen
EDIT	– Programmänderungsmodus
LINES	– Anzahl der bei LIST ausgegebenen Programmzeilen
LIST	– Programmausgabe (abschnittsweise, Fortsetzung jeweils nach ENTER)
LIST#n	– Programmausgabe auf Drucker oder Magnetband
LOAD	– Laden von LIST-gespeicherten Programmen vom Magnetband
NEW	– Löschen des gesamten BASIC-Programmes
NULL	– Legt Anzahl auszugebender Nullen bei LIST# fest
RENUMBER	– Automatische Neunumerierung des BASIC-Programmes
RUN	– Programmstart
TROFF	– Ausschalten TRACE-Modus
TRON	– Einschalten TRACE-Modus
WIDTH	– Länge der Ausgabezeilen festlegen

Weitere BASIC-Anweisungen

Neben den bereits genannten fundamentalen Anweisungen INPUT, LET und PRINT stehen eine Reihe weiterer Anweisungen zur Verfügung, die zur rationellen Gestaltung vieler Programme nützlich sind. Einige sollen kurz erläutert werden:

REM – Anweisung

Zur besseren Übersichtlichkeit der Programme können nach dem Schlüsselwort »REM« beliebige Kommentare in die Programme eingefügt werden, die keinen Einfluß auf deren Abarbeitung haben. »REM« kann auch durch »!« ersetzt werden.

FOR ... NEXT – Anweisung

Zur mehrfachen Abarbeitung von Anweisungsgruppen (z.B. bei der Berechnung von Tabellen, Folgen, Reihen oder bei Iterationszyklen) ist eine sogenannte »Laufanweisung« erforderlich, die im BASIC durch die FOR... NEXT-Anweisung gebildet wird. Sollen z.B. die Funktionen

$$f(x) = x \cdot e^x, \quad g(x) = \sqrt{3x + 10}$$

im Intervall [–2,2] im Abstand von 0,5 tabelliert werden, so wird dies durch folgendes Programm erreicht:

```
40 REM Tabellenberechnung
50 FOR X=-2 TO 2 STEP .5
60 F=X*EXP(X)
70 G=SQR(3*X+10)
80 PRINT X, F, G
90 NEXT X
```

Dabei bedeutet die Anweisung 50, daß die folgenden Anweisungen mehrfach ausgeführt werden sollen, und zwar für X=–2 bis X=2 in Abständen von 0.5, d.h. für

X = –2, –1.5, –1, –0.5, 0, 0.5, 1, 1.5, 2.

Die Anweisung 90 beendet diese Anweisungsgruppe.
Zur Zeile 80 muß noch erläutert werden, daß die durch PRINT auszugebenden Elemente durch Komma oder Semikolon getrennt werden können. Das Komma bewirkt die Ausgabe in einem festen Raster (Tabelle), während

durch ein Semikolon die Werte fortlaufend ausgegeben werden.
Die Ausführung dieses Programmes liefert am Bildschirm:

```
-2          -.270671        2
-1.5        -.334695        2.34521
-1          -.367879        2.64575
-.5         -.303265        2.91548
 0           0              3.16228
 .5          .824361        3.39117
 1          2.71828         3.60555
 1.5        6.72253         3.80789
 2         14.7781          4
OK
>■
```

IF ... THEN – Anweisung

Häufig ist es erforderlich, einige Anweisungen nur unter gewissen Bedingungen auszuführen (z.B. Berechnung von SQR(X) nur für $X \geqq 0$). Dazu dient die Anweisung

IF bedingung THEN anweisung.

Hierbei steht für »bedingung« ein logischer Ausdruck (z.B. $A < B$, $X > 5$, $A\$ = $»A«), der wahr oder falsch sein kann.
Nur wenn er wahr ist, wird die auf THEN folgende »anweisung« ausgeführt. Wesentlich für die Nutzung ist, daß die »bedingung« auch Vergleiche von Zeichenketten beinhalten darf.

Beispiel:
```
 90 INPUT »Gib Wert für A ein: »; A
100 IF A >0 THEN PRINT »A ist positiv«
110 IF A<0 THEN PRINT »A ist negativ«
```

GOTO – Anweisung

Durch die Anweisung

GOTO zeilennummer

kann ein Programm, abweichend von der normalen Reihenfolge, in einer anderen Programmzeile entsprechend der »zeilennummer« fortgesetzt werden. Diese »zeilennummer« muß als Zahlenwert konkret angegeben werden, beispielsweise

210 GOTO 40.

Ein Beispiel für die Anwendung der beschriebenen Befehle ist das folgende Ratespiel, in dem im Befehl 30 durch einen Spieler (A) eine Zahl eingegeben werden muß, die ein anderer (B) dann erraten soll.

```
 10 REM Zahlenratespiel
 20 REM Spieler A gibt vor (Zeile 30)
 30 INPUT »Vorgabezahl (Spieler A):«; A
 35 REM Loeschen des Bildschirmes in 36
 36 CLS
 40 REM Spieler B darf raten (Zeile 50)
 50 INPUT »Gib einen Tip ein: «; B
 60 IF A< B THEN PRINT B, »ist zu groß«
 70 IF A>B THEN PRINT B, »ist zu klein«
 80 IF A=B THEN 100
 90 GOTO 40
100 PRINT B, »ist die gesuchte Zahl«
110 INPUT »Neues Spiel (J/N)?«; A$
120 IF A$=»J« THEN GOTO 30
130 PRINT »ENDE«
```

Unter Nutzung der beschriebenen Standardfunktionen kann sehr leicht auch der Rechner als »Spieler A« eingesetzt werden. Ersetzt man Zeile 30 durch

30 A = INT(100 * RND(1)),

so wird automatisch eine ganzzahlige zweistellige Zufallszahl erzeugt.

Der Ablauf des Programms am Bildschirm vollzieht sich in folgender Weise:

```
Gib einen Tip ein: 50
  50 ist zu groß
Gib einen Tip ein: 10
  10 ist zu klein
Gib einen Tip ein: 25
  25 ist zu klein
Gib einen Tip ein: ■
```

Neben den mathematischen Standardfunktionen stehen im BASIC des Z 9001 einige Funktionen zur Zeichenkettenverarbeitung zur Verfügung.
Von diesen sollen hier CHR$, LEN und STRING$ erläutert werden.

CHR$(N) – Liefert das Zeichen mit der Nummer N in der ASCII-Code-Tabelle.

LEN(A$) – Liefert die Anzahl der Zeichen, die die Zeichenkette A$ enthält.

STRING$(N, A$) – Setzt die Zeichenkette A$ genau N-mal aneinander.

Zusätzlich sei erwähnt, daß man Zeichenketten durch »+« aneinanderfügen kann und in einer IF ... THEN-Anweisung wie bei Zahlen die Vergleichsoperatoren <, >, =, <=, >= anwendbar sind. Damit kann man mit Zeichenketten praktisch »rechnen«, wobei die Vergleichsoperatoren die Zeichenketten bezüglich ihrer alphabetischen Ordnung (allgemeiner: entsprechend der ASCII-Code-Numerierung) sortieren, wie das folgende Programmbeispiel zeigt:

```
 5 ! Sortierung zweier Namen
10 INPUT »Gib 2 Namen ein: «;
   N1$, N2$
20 IF N1$<N2$ THEN GOTO 60
30 H$=N2$
40 N2$=N1$
50 N1$=H$
60 PRINT »Alphabetische Reihen-
   folge: «
70 PRINT N1$; CHR$(44); N2$
80 GOTO 10
```

Werden nach RUN in Anweisung 10 die Namen KARL, ERICH eingegeben so ist am Bildschirm folgendes zu sehen:

```
Gib 2 Namen ein: KARL, ERICH
Alphabetische Reihenfolge:
ERICH, KARL
Gib 2 Namen ein: ■
```

Zur Erläuterung der Zeile 70 sei vermerkt, daß der Rechner alle Buchstaben, Ziffern, Sonder- und Grafikzeichen in codierter Form speichert. Jedem dieser Zeichen entspricht eine Zahl (0 bis 255). Durch die Funktion CHR$(N) kann umgekehrt einer Zahl auch das zugehörige Zeichen zugeordnet werden.
In Zeile 70 liefert der Wert CHR$(44) das Kommazeichen, welches zwischen ERICH und KARL ausgegeben wird.
Die Anwendungsmöglichkeiten der beschriebenen Zeichenkettenfunktionen sind sehr vielfältig. Neben der Analyse von Zeichenketten können besonders Probleme der Bildgestaltung zweckmäßig gelöst werden.
Soll eine Überschrift vorher unbekannter Länge unterstrichen werden, so läßt sich dies mit den obigen Funktionen leicht realisieren. Im folgenden Programmteil soll die Textzeile

ABRECHNUNG FUER monatsname

für beliebigen »monatsnamen« entsprechend der Textlänge mit dem Zeichen Nr. 160 (■) unterstrichen werden.

```
10 ! Anwendung der ZK-Funktionen
20 A$=»ABRECHNUNG FUER«
```

```
30 INPUT »Gib Monatsnamen ein: «;
   MO$
35 CLS: REM Bildschirm loeschen
40 N=LEN(MO$)+16
50 PRINT A$+MO$
60 PRINT STRING$(N, CHR$(160))
70 ! Datenauswertung fuer lfd. Monat
 :
```

Zusammenstellung der BASIC-Anweisungen

Da eine vollständige Beschreibung aller Anweisungen des Z 9001 den Rahmen dieses Abschnittes sprengen würde, soll wenigstens eine Übersicht der Schlüsselwörter aller Funktionen und Anweisungen aufgeführt werden:

a) Logische Operatoren
AND, NOT, OR

b) Mathematische Funktionen
vgl. S. 54

c) Zeichenketten-Funktionen

ASC	– ASCII-Code eines Zeichens
CHR$	– Zeichen entsprechend ASCII-Nummer
INSTR	– Position einer ZK in einer anderen
LEN	– Länge einer Zeichenkette
LEFT$	– Linker Teil einer Zeichenkette
MID$	– Mittlerer Teil einer Zeichenkette
RIGHT$	– Rechter Teil einer Zeichenkette
STR$	– Wandelt Zahl in Zeichenkette um
STRING$	– Vervielfacht eine Zeichenkette
VAL	– Wandelt Zeichenkette in Zahl um

d) Sonderfunktionen

FRE	– Gibt noch freien Speicherplatz an
POS	– Liefert Kursorposition
SPC	– Ausgabe von Leerzeichen
USR	– Aufruf nutzereigener Maschinenprogramme
TAB	– Setzt Kursor auf TAB-Position

e) Anweisungen

BEEP	– Tonerzeugung
BORDER	– Farbzuweisung für Randbereich
CALL	– Aufruf eines Maschinenprogramms
CLS	– Bildschirm löschen
DATA	– Datenbereitstellung
DEEK	– 2-Byte-Eingabe
DEF FN	– Definition einzeiliger Funktionen
DIM	– Feldvereinbarung
DOKE	– 2-Byte-Ausgabe
END	– Phys. Programmende
FOR...NEXT	– Laufanweisung
GOSUB	– Sprung zu einem Unterprogramm
GOTO	– Sprung zu einer Zeile
IF...THEN...ELSE	– Bedingte Anweisungen
INK	– Festlegung Vordergrundfarbe
INKEY$	– Zeicheneingabe von Tastatur
INPUT	– Eingabe Zahlen oder Zeichenkette
INP	– Lesen einer Port-Adresse
JOYST	– Spielhebelabfrage
LET	– Ergibtanweisung
ON...GOTO	– Verzweigungsanweisung
ON...GOSUB	– Verzweigung zu Unterprogrammen
OUT	– Ausgabe auf Port-Adresse

PAPER	– Festlegung Hintergrundfarbe
PAUSE	– Programmunterbrechung
PEEK	– Lesen eines Bytes von einer Speicheradresse
POKE	– Schreiben eines Bytes in eine Speicheradresse
PRINT	– Standardausgabe von Zahlen und Zeichen
PRINT AT	– Ausgabe ab spezieller Bildschirmposition
READ	– Lesen der DATA-Anweisungen
REM	– Bemerkung (Kurzzeichen: !)
RESTORE	– Setzen des Zeigers für READ-Anweisung
RETURN	– Abschluß eines Unterprogrammes
STOP	– Programmabbruch
WAIT	– Zeitweilige Programmunterbrechung
WINDOW	– Festlegung des Rollbereiches am Bildschirm

3. Erweiterungsmöglichkeiten des Heimcomputers Z 9001

Wie bereits angedeutet wurde, besitzt der Heimcomputer Z 9001 eine Reihe von Möglichkeiten, zusätzliche Peripherie anzuschließen.
Das wichtigste externe Gerät außer Fernsehgerät und Kassettentonband ist sicher ein Drucker, der beim Z 9001 über einen V24-Druckermodul angeschlossen werden kann. In der DDR steht z.B. der Thermodrucker TD 40 (robotron K 6303) zur Verfügung.

Da der Drucker über den Ausgabekanal 2 angesprochen wird, ist es möglich, mit

LIST#2 ENTER

Programmausdrucke auf den Drucker auszugeben oder innerhalb der Programme mit PRINT#2;... Daten zu drucken.

Beispiel: Ersetzt man im Beispiel zur FOR...NEXT-Anweisung (Abschnitt 2) Zeile 80 durch

80 PRINT#2; X, F, G

so wird die gesamte Tabelle nicht auf dem Bildschirm, sondern auf dem Drucker ausgegeben.

Typisch für Heimcomputer ist die Möglichkeit des Anschlusses von 2 Spielhebeln. Die Stellung dieser Spielhebel (9 Positionen) kann durch die Funktion JOYST(X) abgefragt und in einem Spielprogramm z.B. zur Erzeugung von Bewegungen (Autorennen, Katze und Maus u.ä.) genutzt werden.

Beispiel:
```
100 ! Spielhebel 1 abfragen
110 A=JOYST(1)
120 IF A=4 THEN...!
    Reaktion
```

Falls der Farbmodul im Heimcomputer eingebaut ist, können alle Zeichen farbig dargestellt werden.
Dafür stehen 8 Farben zur Verfügung, die unabhängig voneinander für die Farbe der Schrift (Befehl INK) bzw. des Schriftgrundes (Befehl PAPER) ausgewählt werden können. Den Farben sind die Zahlen 1, ..., 8 in der Reihenfolge schwarz, rot, grün, gelb, blau, purpur, hellblau, weiß zugeordnet, so daß die Anweisungen

```
200 N=2
210 INK N
220 PAPER 1
230 PRINT »rot auf schwarz«
```

eine rote Schrift auf schwarzem Hintergrund hervorrufen.
Weiterhin steht ein Musik-Modul zur Verfügung, mit dem über den Befehl

BEEP N, M, L

auch Tonfolgen programmiert werden können.

Dabei codieren N – die Tonhöhe
M – die Tonlänge
L – die Lautstärke.

Interessante Anschlußmöglichkeiten bietet der Heimcomputer Z 9001 durch die 8 parallelen E/A-Kanäle, über die verschiedene elektronische Geräte und Hobby-Anlagen anschließbar sind.
Weiterhin können z.B. mit dem zum Computer erhältlichen Analog-Digital-Wandler Meßwerte (Spannungen, Temperaturen, Kräfte, Zeiten usw.) in den Rechner übernommen und dort verarbeitet werden. Ebenso kann der Rechner Signale ausgeben, mit denen andere Geräte und Anlagen (elektrische Eisenbahnen usw.) gesteuert werden können.
Neben den als Zubehör angebotenen Ergänzungsteilen bleibt es dem Bastler überlassen, weitere Geräte unter Beachtung der Anschlußbedingungen für den Heimcomputer ansprechbar zu machen.

Autor:

Dr. rer. nat. Gert Keller

DDR 8023 Dresden
Eisenberger Straße 9

VEB Robotron Meßelektronik
„Otto Schön" Dresden

1. Auflage
Lizenznummer 114-210/50/84
LSV 1083
Verlagslektor: Helga Fago
Gestaltung: Lothar Gabler
Printed in GDR

Satz und Druck:
Messedruck Leipzig, Bereich Borsdorf
III-18-328

Redaktionsschluß: 15. 3. 1984
Bestellnummer: 546 921 6
00780

Vorschau

Heft 3

Ehrsam: Die Taschenrechnerproduktion in der DDR

Schönfelder: HOMECOMPUTER – ein neues Gebiet für den Amateur (Teil 3)

Hübner: Polycomputer 880

Schönfelder: Computerspiele – mehr als eine Spielerei

Horn: Wie kann ein Programm systematisch entworfen werden ?

Die Broschürenreihe

KLEINSTRECHNER-TIPS

behandelt

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

und will dem Laien auf dem Gebiet der Informationsverarbeitung Anregungen geben für seine Arbeit mit

- einfachen oder programmierbaren Tisch- und Taschenrechnern,
- Heim- und Videocomputern,
- Mikrorechnern

und anderen modernen Rechenhilfsmitteln.