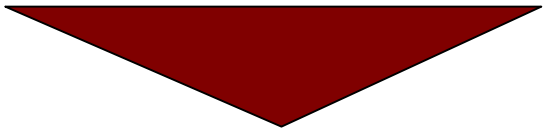
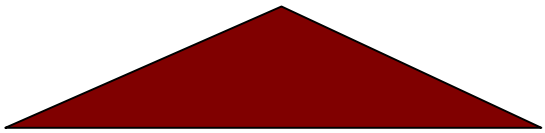
HANNES GUTZNER / GERD HÜTTERER



BASIC

mit dem Z 1013

VEB ROBOTRON-ELEKTRONIK RIESA



digitalisiert: U.Zander 2011 <zander@felix.sax.de>

Herausgeber: VEB Robotron-Elektronik Riesa
Redaktion und Gestaltung: Bernd Schalwat, VEB Robotron-Vertrieb Berlin
Satz und Druck: (52) Nationales Druckhaus, Betrieb der VOB National,
Berlin, 1055 (AG 706/229/87)

00750

Inhalt

Der Z 1013 - ein vielseitiger Mikrocomputer

Computer vervielfachen heute in nahezu allen Bereichen unserer Gesellschaft die schöpferischen Fähigkeiten der Menschen. Mit ihrer Hilfe wird berechnet, konstruiert, vermessen, verwaltet und experimentiert.

Schnell, zuverlässig, exakt und weitere Eigenschaften werden genannt, wenn man fragt, warum Mikrocomputer von vielen Arbeitsplätzen gar nicht mehr wegzudenken sind. Den Computer nutzen, um mehr leisten zu können - das ist ein vielgenanntes Motiv für das Erlernen des Umgangs mit der Rechentechnik und den Programmiersprachen. Dieser Trend wird auch an der Tatsache deutlich, daß sich viele Interessenten wünschen, einen kleinen leistungsfähigen Mikrorechner fürs Lernen und Forschen, für Arbeit und Freizeit, für Spiel und Spaß nutzen zu können.

Der Mikrorechnerbausatz Z 1013 des VEB Robotron-Elektronik Riesa wird diesen Wünschen gerecht. Auf der Basis des Mikroprozessors U 880 D erfüllt er viele Anforderungen, die vor allem jugendliche Computerfreunde an ihren Mikrocomputer stellen. Dazu gehören eine Speicherkapazität von 16 KBytes in der Grundvariante, die Erweiterungsmöglichkeit durch eine Steckverbindung, die Ein- und Ausgabe von Steuersignalen über einen PIO-Baustein, mehrere mögliche Programmiersprachen und ein Satz von 128 Pseudografikzeichen.

Die Bedienungsanleitung, die zum Lieferumfang des Mikrorechnerbausatzes Z 1013 gehört, informiert umfassend über alle wesentlichen technischen Eigenschaften, die Arbeit mit dem Maschinencode, dem Tiny-BASIC und über die Anforderungen an die peripheren Geräte.

Die vorliegende Broschüre will dem Leser vor allem bei der Nutzung der Programmiersprache BASIC auf seinem Z 1013 hilfreich zur Seite stehen. Zuvor sollen jedoch noch einige technische Arbeiten empfohlen werden.

Mit geringem Aufwand läßt sich der Bausatz in einen funktionstüchtigen Computer verwandeln. Das Verbinden der Leiterplatte mit der Flachfolientastatur durch ein anzulötendes Flachbandkabel und der Aufbau eines Netzteiles mit 12 V Wechselspannung und 20 VA abzugebender Leistung sind die Anforderungen an den polytechnisch vorgebildeten Nutzer des Z 1013.

Nach sorgfältiger Ausführung dieser Arbeiten ist der Mikrorechner bereits arbeitsfähig. Er versteht die Maschinensprache und auf dieser Grundlage aufgebaute Interpreter. Dazu gehören einer für Tiny-BASIC (etwa 3 KBytes) und einer für komfortables BASIC von etwa 10 KBytes Speicherumfang, das mit dem BASIC der anderen Kleincomputer der DDR weitgehend übereinstimmt. Unser Z 1013 dankt uns den Schutz vor Staub und mechanischer Beschädigung durch zuverlässiges Arbeiten. Der beste Schutz ist ein entsprechendes Gehäuse aus Holz, Plast oder Metall. Dem Einfalls- und Gestaltungsreichtum des Hobbybastlers sind dafür keine Grenzen gesetzt, wenn er einige Hinweise beachtet. Alle Steckverbindungen müssen gut zugänglich sein. Eine Wärmeableitungsmöglichkeit über dem Festspannungsregler ist unbedingt anzuraten. Die Flachfolientastatur sollte in einer ergonomisch (etwa 20 Grad) günstigeren Schräglage angeordnet werden. Durchsichtiges Material für die Deckplatte erhält den reizvollen Blick auf die schaltkreisbestückte Leiterplatine (Bild 1). Für die Bodenplatte ist unbedingt zu beachten, daß jegliche Gefahr von Kurzschlüssen durch metallische Schrauben oder Metallflächen, die sich eventuell auch durchbiegen können, vermieden wird.

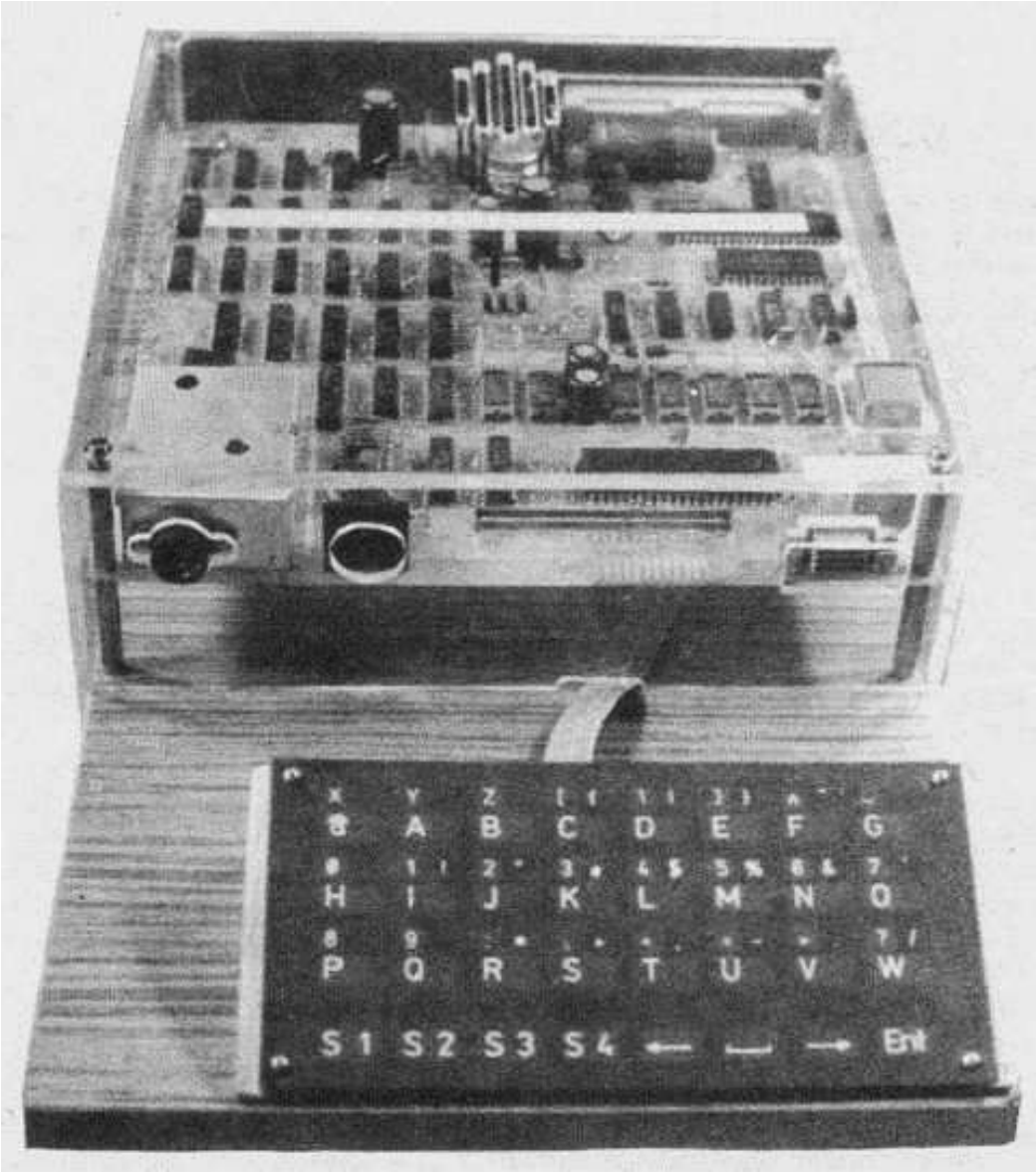


Bild 1: Z 1013 mit Verkleidung

Der sorgfältigen Vorbereitung der Verbindungskabel ist große Aufmerksamkeit zu widmen. Für die Zuleitung zum Fernsehgerät eignet sich neben dem Koaxialkabel (70 Ohm) auch eine einadrige geschirmte Leitung (Mikrofonkabel). Durch ihre Flexibilität stört sie am Arbeitsplatz kaum. Ein intaktes Diodenkabel sorgt für eine störfreie Verbindung zum Kassettenrekorder. Für den Nutzer ist es wichtig zu wissen, daß das Kassetteninterface des Z 1013 bei Verwendung des 10-K-BASIC-Interpreters mit dem der Kleincomputer KC 85/1. KC 87, KC 85/2 und KC 85/3 übereinstimmt. Dadurch sind Programme wechselseitig einlesbar. Ob sie allerdings in jedem Fall genutzt werden können, hängt von weiteren Faktoren ab, die noch besprochen werden.

Der BASIC-Interpreter des Z 1013

Eine der am häufigsten verwendeten Anweisungen in der Programmiersprache BASIC ist PRINT. Mit ihr werden Texte, Rechenergebnisse, aber auch spezielle Grafikzeichen auf dem Bildschirm angezeigt. Aber print heißt ja eigentlich drucken. Als Mitte der 60er Jahre die zwei amerikanischen Pädagogen John G. Kemeny und Thomas E. Kurtz ihren Schülern die Programmierung von Computern verständlich machen wollten, erfanden sie dazu die Programmiersprache für den Anfänger. Damals arbeiteten die meisten Computer noch mit Bedienschreibmaschinen anstelle des Bildschirmes. Der Nutzer mußte seine Eingaben über diese Schreibmaschine tätigen, ebenso gab der Computer seine Ergebnisse auf dieser Schreibmaschine aus; deshalb des PRINT für die Ausgabe.

Die heutige Arbeit am Computer mit Tastatur und Bildschirm ist natürlich angenehmer, vor allem leise und papiersparend. Sie setzt aber heute wie damals entsprechendes Wissen über die verwendete Hard- und Software voraus. Als Hardware steht uns der Z 1013 zur Verfügung, als Software ein leistungsfähiger BASIC-Interpreter, der mit dem Interpreter des KC 87 weitgehend übereinstimmt. Zum Thema Software sollten wir aber noch ein paar Worte vorausschicken.

Die Programmiersprache BASIC ist eine von rund 30 Programmiersprachen, auf der Welt. Verleiht man den Dialekten der einzelnen Sprachen (allein BASIC kann mit etwa 50 Dialekten aufwarten) eine Eigenständigkeit, dann erreicht man die Zahl 1000. Trotz dieser großen Anzahl von Programmiersprachen nimmt BASIC, insbesondere für den Klein- und Personalcomputer in der Hand des Nichtinformatikers, nach wie vor eine Favoritenrolle ein. Das hat u. a. folgende Ursachen:

1. Die Bildschirmausgabe läßt sich mit den verfügbaren Anweisungen auf einfache Weise sehr ansprechend gestalten.
2. BASIC bietet leistungsfähige Funktionen zum Umgang mit Texten.
3. BASIC enthält vielfältige Grafik-, Farb- und Tonausgabeanweisungen, die allerdings sehr dialektspezifisch oder gerätebezogen sind. Genauer gesagt ist es so, daß die Beliebtheit von BASIC die Computerhersteller dazu bewogen hat, dieser Sprache immer mehr attraktive Extras "anzuhängen". Diese Extras sind dann selbstredend auf die Computer der jeweiligen Hersteller zugeschnitten.
4. Die bei Kleincomputern fast ausschließlich verwendete interpretative Arbeitsweise mit BASIC ist außerordentlich "menschenfreundlich".

Den Unterschied zwischen der Arbeitsweise eines Interpreters und eines Compilers kann man sich am Beispiel der Arbeit von Sprachmittlern klarmachen. Wenn der Sprachmittler in seinem Büro arbeitet, dann wird ihm ein vollständiger Text mit einer bestimmten Seitenzahl vorgelegt, den er in einem Stück übersetzt und dann die Übersetzung an den Auftraggeber ausliefert. Damit ist die Sache für ihn erledigt, und er kann andere Texte übersetzen oder Urlaub machen. Das bedeutet, bezogen auf den Computer, daß die Anwesenheit des Übersetzers nicht mehr erforderlich ist, er also aus dem Speicher gelöscht werden kann. Dies ist die Arbeitsweise eines Compilers.

Der Interpreter hingegen arbeitet wie ein Dolmetscher. Der Dolmetscher muß während des gesamtem Gespräches der beiden Kommunikationspartner anwesend sein und Satz für Satz übersetzen. Dieses "scheibchenweise" Übertragen in eine andere Sprache dauert natürlich länger als die Übersetzung, dafür können Mißverständnisse sofort aus dem Weg geräumt werden, denn der Dolmetscher (Interpreter) ist ja immer anwesend. Diese sofortigen Programmkorrekturmöglichkeiten sind ein riesiger Vorteil für den praktischen Umgang mit Mikrocomputern. Deshalb wird hier auch mit Vorliebe ein BASIC-Interpreter genutzt, obwohl es, z.B. für den Personalcomputer PC 1715, auch einen BASIC-Compiler gibt.

Als Markierung einzelner oder mehrerer durch einen Doppelpunkt getrennte Anweisungen werden in BASIC Zeilennummern verwendet. Mit einer GOTO-Anweisung kann jede vorhandene Zeilennummer angesprungen werden. Dies schwört bei ungenügendem Durchdenken des Gesamtproblems einen "Programmsalat" herauf, der keinerlei Programmstruktur mehr erkennen läßt. Genauso wie ein vernünftiger Lieferwagenfahrer nicht wild drauflosfährt, sondern zuvor den Tourenplan durchdenkt, sollte auch der BASIC-Programmierer nicht sofort Tastatur und Computer "quälen", sondern sich des manchmal beschwerlichen Weges vom Problem zum Programm erinnern, der stets in folgenden Schritten zu absolvieren ist: Problemstellung, Problemanalyse, Algorithmus mit Programmablaufplan oder Struktogramm, Programmierung, Programmtest und Programmdokumentation.

Die wichtigsten Handwerkszeuge sind also zunächst nicht der Computer, sondern Bleistift, Papier und ein Radiergummi; für den Anfänger natürlich noch entsprechende Literatur. Wir können hier aus Platzgründen keinen BASIC-Lehrgang vermitteln. Dazu ist mittlerweile eine Vielzahl guter BASIC-Bücher erschienen, auf die am Schluß dieser Broschüre hingewiesen wird. Wir können hier noch nicht einmal das erwähnen, was der BASIC-Interpreter des Z1013 kann. Einige seiner Leistungen werden aber in den Programmen, die ja den Hauptteil dieser Broschüre ausmachen, Verwendung finden. Hierzu sei als Literatur das Programmierhandbuch zum KC 85/1 und KC 87 des VEB ROBOTRON-Meßelektronik "Otto Schön" Dresden empfohlen.

	Konstante	Variable	Term
Zahl	33.1234	V oder VS oder V1	VS ^ 2 / (2*A)
Zeichenkette(String)	"OTTO"	N$ oder NA$ oder N5$	NA$ + "KAR"

Bild 2: Verarbeitung von Zahlen und Zeichenketten in BASIC

Ein BASIC-Interpreter kann nicht nur mit Zahlen, sondern auch vortrefflich mit Text umgehen. Bild 2 zeigt die entsprechenden Verarbeitungsmöglichkeiten. Da es sich bei der Darstellung und Verarbeitung von Text um die Aneinanderreihung von Zeichen (Achtung, das können auch Ziffern oder Grafiksymbole sein), spricht man von Zeichenketten oder verwendet den aus dem Englischen stammenden Fachbegriff String. Zwischen Zahlen und Zeichenketten wird streng unterschieden. Den Versuch einer Vermischung ahndet der Interpreter mit der Fehlerausschrift TM ERROR, wobei das TM für "Type mismatch" steht. Allerdings bietet BASIC Sonderfunktionen, die, sofern dies vernünftig ist, eine Umwandlung von Zahlen in Zeichenketten (STR$(X)) und umgekehrt (VAL(X$)) ermöglichen.

BASIC arbeitet sowohl bei Zahlen als auch bei Strings mit Konstanten, Variablen und Termen (bei Termen mit Strings wird nur das Pluszeichen zum Aneinanderfügen von Zeichenketten akzeptiert). Zahlen werden im Interpreter des Z 1013 mit einer Genauigkeit von sieben Stellen verarbeitet, wobei sechs Stellen angezeigt werden. Der Computer benötigt intern für solch eine Zahl 4 Bytes. Damit kann er Zahlen im Bereich von etwa 10 38 bis 10 -39 verarbeiten. Wenn zur Darstellung auf dem Bildschirm die sechs Stellen nicht ausreichen, wird, wie das auch von Taschenrechnern her bekannt ist, auf die Gleitkommadarstellung mit Angabe des Exponenten umgeschaltet. Übrigens arbeitet der Interpreter des Z 1013 generell mit Gleitkommazahlen (ebenso wie die Kleincomputer der KC-Serie), eine gesonderte Vereinbarung zur Arbeit mit ganzen Zahlen ist also nicht möglich. Zur komfortablen Darstellung auf dem Bildschirm fehlt leider auch die PRINT USING-Anweisung, aber hier werden in den Programmen einige Tricks helfen.

Bild 2 weist neben Zahlenkonstanten noch Variablen und Terme aus. Variablen sind Platzhalter oder Schubfächer, deren Inhalt beliebig geändert werden kann. Jede Variable hat einen Namen, also eine Bezeichnung des Schubfaches. Dieser Name beginnt stets, wie es Bild 2 zeigt, mit einem Buchstaben. Es können ein zweiter Buchstabe oder eine Ziffer folgen. Diese zwei Zeichen interpretiert unser Z 1013 als Namen der Variablen. Eventuell vorhandene weitere Zeichen werden zwar "mitgeschleppt", aber nicht ausgewertet. So sind für den Computer die Namen AB, ABBA und ABO identisch. Für den Z 1013 raten wir von mehr als zwei Zeichen für einen Variablennamen ab, denn die Verwendung längerer Variablennamen erhöht die Wahrscheinlichkeit, mit BASIC-Schlüsselwörtern (z.B. DIM, OUT, DEF, ELSE u.a.) "ins Gehege" zu kommen. Verwendet man generell nur ein oder zwei Zeichen für Variablennamen, dann sind folgende Namen verboten, da sie zu eben diesen Schlüsselwörtern gehören: AT, FN, GO, IF, LN, ON, OR, PI und TO.

Der Umgang mit Termen ist in BASIC sehr einfach möglich. So kann z.B. problemlos ein komplizierter Term als Argument in einer Funktion stehen, ohne daß in "Hilfsvariablen" Zwischenergebnisse abgespeichert werden müssen. Allerdings kann das Vorgehen mit Hilfsvariablen zeiteffektiver sein, denn das kürzeste und eleganteste Programm ist nicht immer das schnellste. Aber das ist schon eine Sache für die Spezialisten.

Auch bei der Verwendung von Termen können wir ganz beruhigt sein. Der Interpreter geht bei der Verarbeitung von Termen mit folgender Wertigkeit vor:

(...)		
Funktion		
^ (Potenzieren mit reellem Exp.)	)	
* /	)	arithmetische Operatoren
+ -	)	
<< == < >> = >		vergleichende Operatoren
NOT	)	
AND	)	logische Operatoren
OR	)	

Stehen mehrere ranggleiche Operatoren zur Bearbeitung an, dann erfolgt diese von links nach rechts.

Bild 2 zeigt, daß Zeichenkettenkonstanten in Anführungszeichen einzuschließen sind. Zeichenkettenvariablen unterscheiden sich von Zahlenvariablen, indem an ihren Namen ein Dollarzeichen angefügt wird. Ein Aneinanderfügen von Zeichenketten ist mit einem Pluszeichen möglich. Dazu folgender Versuch mit unserem Z 1013. Mit LET NA$ = "OTTO" (mit ENTER-Taste abschließen) weisen wir einer Zeichenkettenvariablen mit dem Namen NA$ den Inhalt OTTO zu. Anschließend tasten wir PRINT NA$+ "KAR" (mit ENTER abschließen) ein. Was erscheint auf dem Bildschirm?

Wir haben soeben mit der Benutzung der LET- und PRINT-Anweisung etwas vorgegriffen. Gemäß Bild 3 sollen nun wichtige Kommandos, Anweisungen und Funktionen für die Arbeit mit dem BASIC-Interpreter unseres Z 1013 betrachtet werden. Kommandos werden in den meisten Fällen vom Nutzer über die Tastatur eingegeben und veranlassen den Computer zur sofortigen Ausführung nach Abschluß der Eingabe mit ENTER (Taste ENT). Anweisungen und Funktionen sind die Hauptbestandteile eines BASIC-Progamms, dessen einzelne Programmzeilen stets mit einer Zeilennummer beginnen. Wird diese Zeilennummer weggelassen, dann führt der Z 1013 die eingegebene Anweisung oder Funktion nach Abschluß mit ENTER sofort wie ein Kommando aus.

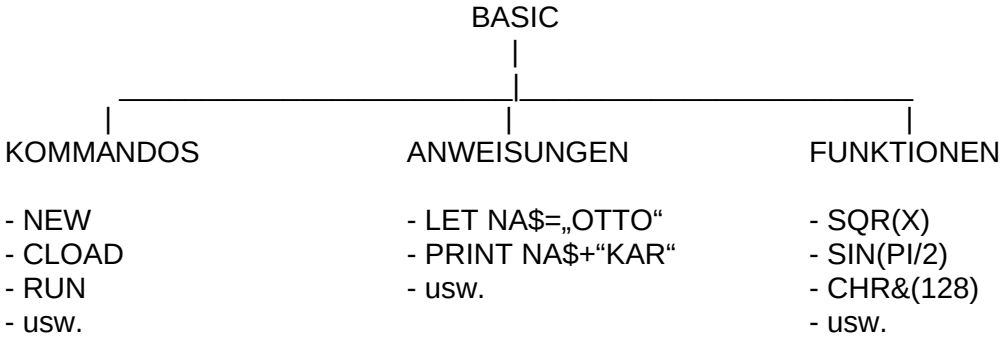


Bild 3: Kommandos, Anweisungen und Funktionen in BASIC

Dies ist natürlich nicht der Sinn des Umgangs mit einem Mikrocomputer, der ja programmgesteuert arbeiten soll. Allerdings ist unsere Trennung in Kommandos und Anweisungen nicht zu streng zu sehen. So kann z.B. das Kommando CLS zum Löschen des Bildschirmes vom Nutzer eingegeben werden, es kann aber ebenso als Anweisung in einem BASIC-Programm vorkommen. Das Löschen kann auch über die Folientastatur durch Drücken und Festhalten der Taste S4 und dabei gleichzeitiges Betätigen der Taste T ausgelöst werden.

BASIC-Kommandos

Wir werden hier nur die wichtigsten Kommandos nennen, die vor allem den Anfänger möglichst schnell an die Nutzung der hier vorgestellten Programme heranführen sollen. Nachdem der BASIC-Interpreter in den Schreib-Lese-Speicher (RAM) des Z 1013 eingelesen wurde, gelangt man mit dem Monitor-Kommando J 300 (J steht für JUMP = Sprung, dies ist aber **kein** BASIC-Kommando) zum BASIC-Interpreter. Mit dem BASIC-Kommando BYE kann der BASIC-Intertpreter verlassen und zum Betriebssystem zurückgekehrt werden. Ein wiederholter Neustart (Kaltstart) mit J 300 löscht eventuell noch vorhandene BASIC-Programme und Daten (Vorsicht!). Ein Restart (Warmstart) mit J 302 Iäßt BASIC-Programm und Daten unversehrt.

Liegen fertige Programme auf einer Magnetbandkassette vor, so ist zur Nutzung dieser Programme lediglich die Kenntnis folgender drei Kommandos erforderlich:

NEW: Dieses Kommando löscht eventuell vorhandene BASIC-Programme und Daten. Ein unüberlegtes NEW kann deshalb stundenlange Eintipparbeit zunichte machen, sofern nicht vorher die eingegebenen Daten oder das Programm auf einer Magnetbandkassette abgespeichert wurden. Vor dem Einlesen eines neuen Programms ist beim Z 1013 stets das Kommando NEW einzugeben, damit alle alten Inhalte gelöscht werden. Zu beachten ist, daß NEW nicht die Größe des Zeichenkettenspeichers auf den Standardwert von 256 Zeichen zurücksetzt. Sollte die Fehlermeldung OS ERROR plötzlich bei Programmen auftreten, die bisher tadellos liefen, dann kann der Anfänger sich mit BYE und J 300 aus der Affäre ziehen. Der Fortgeschrittene Iäßt sich mit der speziellen Funktion PRINT Fre(I$) die Größe des Zeichenkettenspeichers ausgeben und stellt dann mit dem Kommando CLEAR 256 wieder den Standardwert ein.

CLOAD"Programmname": Mit diesem Kommando wird ein BASIC-Programm in den Z 1013 eingelesen. Dazu vergleicht der Computer die letzten acht Buchstaben des vom Nutzer eingegebenen Programmnamens mit dem auf der Magnetbandkassette gespeicherten Programmnamen. Bei Gleichheit wird das Programm eingelesen, im anderen Fall erfolgt eine Fehlermeldung mit Abbruch der Leseroutine. Unser Z 1013 sucht also leider nicht die gesamte Magnetbandkassette nach dem eingegebenen Namen ab. Mit der Ausschrift FILE FOUND gibt der Z 1013 an, daß er das gesuchte Programm gefunden und eingelesen hat

(FILE = Datei). Mit der OK-Meldung und dem Promptzeichen > wartet er nun auf weitere Kommandos.

RUN: Beim Z 1013 mit Folientastatur wird das RUN-Kommando durch gleichzeitiges Drücken der Tasten S4 und E realisiert. Dieses Kommando bewirkt die Abarbeitung des Programms (run = rennen) von der ersten BASIC-Zeile an, bei der das Programm beginnt. Wird das Kommando RUN als Buchstabenfolge über die Tastatur eingegeben, dann kann auch eine Zeilennummer angefügt werden, bei der die Programmabarbeitung beginnen soll. Aus Gründen der Vereinheitlichung im Umgang mit fertiger Software sollte diese Variante nicht benutzt werden (natürlich kann Sie beim Programmtest gute Dienste leisten). Nach Start des Programms mit RUN tritt der Nutzer entsprechend dem Programm mit dem Rechner in einen Dialog, dessen "Kontaktstellen" im allgemeinen die Tastatur und der Bildschirm sind. Programmstart mit RUN löscht alle Variablen und Felder, bei Programmstart mit GOTO Zeilennummer bleiben Variablen und Felder erhalten .
Soll die Arbeit am Programm beendet werden, dann meldet sich meist nach Eingabe des gewünschten Programmabbruches der BASIC-Interpreter mit OK und dem Promptzeichen. Das Programm kann nun mit NEW gelöscht werden. Wie sind aber Programme zu beenden, die keine solche Endeabfrage enthalten? Der Anfänger neigt vielleicht zum Ziehen des Netzsteckers. Er möge dabei bedenken, daß damit auch der im Schreib-Lese-Speicher des Z 1013 stehende BASIC-Interpreter gelöscht wird und ein Wiedereinschalten damit einen völligen Neubeginn bedeutet. Besser ist es, wenn auch hierfür die entsprechenden Kommandos bekannt sind:

STOP: Dieses Kommando (Tasten S4 und K) unterbricht die Bearbeitung eines BASIC-Programms. Der Z 1013 meldet sich mit OK und dem Promptzeichen. Immer dann, wenn man den Computer nicht zum Anhalten bewegen kann (z. B. eine endlose Schleife) oder aus einem bestimmten Arbeitsmodus beim Eintippen oder Korrigieren von Programmen heraus möchte, sollte man sich der "Wunderwirkung" dieses Kommandos STOP (auch BREAK) erinnern.

CONT: Ein mit STOP unterbrochener Programmablauf kann, von wenigen Ausnahmen abgesehen, mit dem Kommando CONT (continue = fortsetzen) fortgesetzt werden (Tasten S4 und F).
Das Eintippen von Programmen, z. B. die aus der vorliegenden Broschüre, wird durch die folgenden Kommandos unterstützt:

LIST: Dieses Kommando listet ein im BASIC-Programmspeicher enthaltenes Programm auf (Tasten S4 und D), wobei mit der niedrigsten Zeilennummer begonnen wird. An das Kommando LIST kann auch die Zeilennummer angefügt werden, bei der die Auflistung beginnen soll. Mit dem Kommando LINES X kann die Anzahl X der auszugebenden Zeilen gewählt werden. Als Standard legt der Interpreter 10 Zeilen fest.

AUTO: Bei Aufruf dieses Kommandos übernimmt der Interpreter als zusätzlichen Service die Vorgabe der Zeilennummern für ein einzutippendes Programm. Er beginnt bei der Zeilennummer 10 und geht in Zehnerschritten vor. Beide Werte können durch die Eingabe von. AUTO erste Zeilennummer, Zeilennummernabstand frei gewählt werden. Der AUTO-Modus wird über das Kommando STOP verlassen.

DELETE: Mit diesem Kommando wird die gesamte BASIC-Zeile mit der auf DELETE folgenden Zeilennummer gelöscht, deshalb Vorsicht. Mit der Angabe DELETE erste Zeilennummer, letzte Zeilennummer kann auch ein ganzer Bereich gelöscht werden, deshalb erhöhte Vorsicht.

EDIT: Das Wort edit bedeutet herausgeben. Das Kommando EDIT X gibt die BASIC-Zeile mit der Zeilennummer X auf den Bildschirm aus, damit darin Veränderungen vorgenommen werden können. So kann mit den Kursortasten jede Position in der Zeile erreicht werden.

Das Löschen von Zeichen erfolgt mit dem Kommando DEL (Tasten S4 und @), Platz für Einfügungen wird mit dem Kommando INS (Tasten S4 und B) geschaffen (delete = löschen, insert = einfügen).Ist die Korrektur abgeschlossen, dann wird durch Drücken der ENTER-Taste die Übergabe der korrigierten Zeile in den BASIC-Programmspeicher bewirkt. Zugleich erscheint die nächste BASIC-Zeile auf dem Bildschirm. Jetzt gilt es aufzupassen, da der Z 1013 immer noch im EDIT-Modus ist. Soll in dieser Folgezeile keine Korrektur vorgenommen werden, dann ist dieser Modus mit dem Kommando STOP auf schnellstem Wege zu verlassen.
Es ist zu beachten, daß die Benutzung des EDIT-Modus das Löschen aller vorher definierten Felder und Variablen mit sich bringt.

RENUMBER AZ, EZ, NAZ, SW: BASIC-Programme werden ja deshalb in 5er oder 10er Schritten der Zeilennummern geschrieben, damit bei Programmkorrekturen noch einige Programmzeilen eingefügt werden können. Das ordnungsgemäße Einfügen erledigt für uns der BASIC-Interpreter. Damit entstehen aber ungleiche Schrittweiten zwischen den einzelnen Zeilennummern. Diesen Schönheitsfehler korrigiert das Kommando RENUMBER (umnumerieren). Dazu sind folgende Werte anzugeben: AZ = Anfangszeilennummer, von der ab umnumeriert werden soll, EZ = Endzeilennummer, bis zu der umnumeriert werden soll, NAZ = Neue Anfangszeilennummer, bei der dieser Block beginnen soll, SW = Schrittweite der Zeilennummern im neuen Block. Beim RENUMBER-Kommando ist Vorsicht geboten. Der Interpreter weist Eigenheiten auf, die man genau kennen muß, auch der Mensch macht hier schnell einen Fehler. Wer auf "Nummer sicher" gehen will, macht vor jedem neuen RENUMBER eine Programmkopie auf einer Magnetbandkassette.
Dieses Auslagern von Programmen auf eine Magnetbandkassette soll als letztes im Komplex der Kommandos vorgestellt werden:

CSAVE"Programmname": Nach Auslösung dieses Kommandos mit der ENTER-Taste wird das im Speicher des Z 1013 befindliche BASIC-Programm über die Diodenbuchse auf den Kassettenrecorder, der sich natürlich in Aufnahmebetrieb befinden muß, in Form von akustischen Signalen ausgegeben. Vom Programmnamen werden nur die letzten acht Zeichen mit auf das Kassettenmagnetband ausgegeben. Nach Abschluß des Auslagerns auf das Magnetband fragt der Z 1013 mit dem Kommando VERIFY (verifizieren = überprüfen), ob der Inhalt der Magnetbandkassette mit dem Speicherinhalt verglichen werden soll, denn beide enthalten ja jetzt das BASIC-Programm. Für diese Überprüfung ist das Magnetband zurückzuspulen und mit der Wiedergabetaste zu starten. Der 1013 vergleicht nun Magnetband- und Speicherinhalt. Eine OK-Meldung gibt an, daß die Überprüfung erfolgreich verlaufen ist, der Z 1013 kann also getrost ausgeschaltet werden . Einen guten Kassettenrecorder und möglichst neue Magnetbandkassetten sollte man schon haben. "Bejahrte" Kassetten genügen zwar für Hits, nicht aber für Bits.

BASIC·Anweisungen

Wer selbst Programme schreiben will, muß die Anweisungen und Funktionen kennen, die der benutzte BASIC-Interpreter bereithält. Wir können hier nicht alle Anweisungen des BASIC-Interpreters unseres Z 1013 nennen oder gar erklären und verweisen deshalb nochmals auf das Programmierhandbuch zum KC 85/1 und KC 87. Bei der Betrachtung wichtiger Anweisungen gehen wir von allgemeingültigen Grundstrukturen der Programmierung aus. Das öffnet zugleich den Blick für andere Programmiersprachen, denn sie alle arbeiten mit diesen Grundstrukturen. Lediglich die Anweisungen tragen andere Bezeichnungen. Während z. B. in BASIC die Bildschirmausgabe mit der PRINT-Anweisung erfolgt, wird in FORTH (je nach Version) z. B. ein Dezimalpunkt geschrieben. Alle Programmiersprachen realisieren, mit mehr oder weniger Variantenvielfalt, folgende Grundstrukturen:

- Eingabesequenz
- Zuweisungssequenz
- Ausgabesequenz
- Alternative
- Fallauswahl
- Schleife
- Unterprogramm

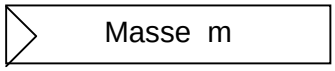
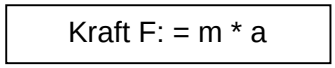
Grundstruktur	Struktogramm	BASIC - Anweisungen
Eingabesequenz		INPUT INKEY$ DATA, READ, RESTORE CLOAD* INP PEEK DEEK
Zuweisungssequenz		LET
Ausgabesequenz		PRINT CSAVE* OUT POKE DOKE

Bild 4: Eingaben, Zuweisungs- und Ausgabesequenzen in BASIC

Die ersten drei Grundstrukturen sind im Bild 4 zusammengefaßt. Das Bild enthält die entsprechende Darstellungsform in einem Struktogramm. Auf die Darstellung als Programmablaufplan wurde verzichtet, da bei der strukturierten Programmierung ohnehin dem Struktogramm "die Zukunft gehört". Für die Eingabesequenz in der Programmmiersprache BASIC gibt es folgende Anweisungen: INPUT, INKEY$, READ, CLOAD, INP, PEEK, DEEK. Der Anfänger beginnt in den meisten Fällen mit der INPUT-Anweisung zu arbeiten. Die INKEY$-Anweisung fragt die Tastatur ab und ist deshalb für Aktionsspiele von großer Bedeutung. Wir haben sie auch in unseren Spielprogrammen verwendet. Die READ-Anweisung wurde im Bibliotheksprogramm benutzt, während das Einlesen einer Datei mit

CLOAD* im Vokabelprogramm zur Anwendung kommt. Immer dann, wenn es um Steuerungsaufgaben geht, wird mit der INP-Anweisung der Inhalt eines Eingabeports eingelesen. Das Lesen des Inhalts spezieller Speicherzellen (von uns z. B. für die Kursorzeilen- und -spaltenposition genutzt) erledigen die Anweisungen PEEK und DEEK.

Die Zuweisungssequenz wird in BASIC mit der Anweisung LET realisiert. Das Wort LET kann beim Interpreter unseres Z 1013 auch weggelassen werden. In unseren Programmen sind beide Varianten zu finden. Der Anfänger sollte das LET besser schreiben, um sich den Sinn dieser Zuweisung vor Augen zu halten. So stellt in der BASIC-Anweisung LET A = 3 + W1/4.7 das Gleichheitszeichen eben kein Gleichheitszeichen im mathematischen Sinne dar (übrigens ein Schönheitsfehler von BASIC). Das Zeichen besagt vielmehr, daß der rechts vom Gleichheitszeichen stehende Ausdruck berechnet und der links vom Gleichheitszeichen stehenden Variablen zugewiesen wird. In dieser Weise funktionieren auch Durchlaufzähler, z. B. LET N = N + 1.

Die Ausgabesequenz (Bild 4) wird in BASIC durch folgende Anweisungen realisiert: PRINT, CSAVE, OUT, POKE, DOKE. Die PRINT-Anweisung bietet viele schöne Möglichkeiten der Bildschirmgestaltung, die hier nicht alle genannt werden können. Auch in unseren Programmen haben wir z. B. die PRINT AT-Anweisung, mit der Zeichen auf beliebige Zeilen- und Spaltenpositionen des Bildschirms gesetzt werden können, benutzt. Allerdings ist dabei zu beachten, daß unser Z 1013 seine ursprüngliche Kursorposition beibehält. Weitere Gestaltungsmöglichkeiten in der aktuellen Ausgabezeile bieten die Anweisungen PRINT TAB(X) und PRINT SPC(X). Übrigens kann beim Eintippen eines Programms für das Wort PRINT ein ? eingegeben werden. Das spart Tipparbeit, und der Interpreter ist so freundlich, diese Fragezeichen im Listing dann durch das Wort PRINT zu ersetzen. Die weiteren in Bild 4 angegebenen Ausgabeanweisungen sind die "Gegenstücke" zu den schon genannten Eingabeanweisungen.

```
10 REM BREMSWEGBERECHNUNG
20 INPUT "GESCHWINDIGKEIT  IN  KM/H =";V
30 INPUT "NEG. BESCHLEUNIGUNG IN  M/S^2=";A
40 LET  S=(V/3.6)^2/(2*A)
50 PRINT "BREMSWEG=";S; "METER":PRINT
60 GOTO 20
70 END
```

Bild 5: Programmierbeispiel in BASIC

Es mag unwahrscheinlich klingen, aber mit den Anweisungen INPUT, LET und PRINT können wir schon ein kleines Programm schreiben. Ein solch einfaches Programm zeigt Bild 5. Es macht deutlich, wie in die INPUT-Anweisung auf einfache Weise Text eingebaut werden kann, der dann auf dem Bildschirm erscheint. Es zeigt weiter, daß bei der PRINT-Anweisung mehrere Ausgaben hintereinander, jeweils durch ein Semikolon getrennt, ausgeführt werden können. Drei Anweisungen aus diesem Programm haben wir noch nicht erklärt, was hier schnell geschehen soll:

REM: Der auf REM (remark = Bemerkung) in derselben BASIC-Zeile folgende Text wird vom Interpreter nicht als Anweisung verarbeitet. REM-Zeilen dienen also nur zur Erklärung des Programms oder bestimmter Programmblöcke.

GOTO X: Mit dieser Steueranweisung wird der Interpreter veranlaßt, zur Programmzeile X zu springen und dort mit der Programmbearbeitung fortzusetzen. Gut durchdachte BASIC-Programme enthalten wenig GOTO-Anweisungen, ihre Struktur bleibt erkennbar.

END: Damit wird der Interpreter veranlaßt, die Programmbearbeitung zu unterbrechen. Die END-Anweisung ist im Bild 5 nicht erforderlich, wir sollten uns ihre Nutzung aber angewöhnen. Folgen im Programm auf höheren Zeilennummern z. B. Unterprogramme, dann wird mit END ein unerlaubter "Hineinsprung" verhindert. Übrigens bewegt sich unser Miniprogramm im Bild 5 in einer ewigen Schleife. Wie kommen wir aus dieser Schleife heraus?

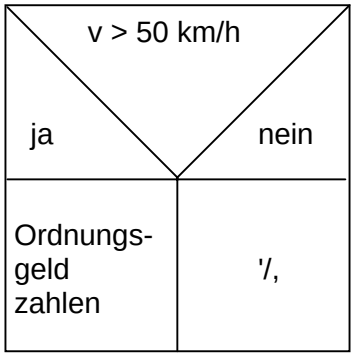
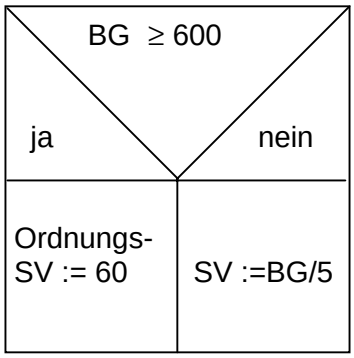
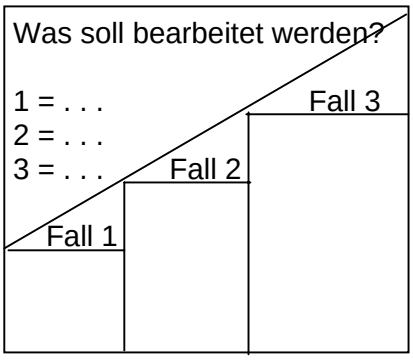
Grundstruktur	Struktogramm	BASIC -Anweisungen
unvollständige Alternative		IF . . .THEN
vollständige Alternative		IF . . . THEN . . . ELSE
Fallauswahl		ON . . . GOTO . . . , . . . , . . . ON . . . GOSUB . . . , . . . , . . .

Bild 6: Alternativen und Fallauswahl in BASIC

Bild 6 präsentiert das "Salz in der Suppe" der Programmierung. Diese "Würze" liegt in den Verzweigungen. Dazu gehören zunächst die unvollständige und vollständige Alternative, die unser Z 1013 mit den entsprechenden BASIC-Anweisungen versteht (IF = wenn, THEN = dann, ELSE = sonst). Der Computer entscheidet hier stets zwischen ja oder nein, wahr

oder falsch. Diese Anweisungen haben wir auch in unseren Programmen häufig benutzt. Bei dem Gedanken, daß auch "pfiffige" Schachprogramme im Grunde "nur" mit solchen Alternativen funktionieren, wird vielleicht der "Geistesaufwand" bei der Programmherstellung annähernd deutlich. Daran ändert auch die Tatsache nichts, daß Schachprogramme aus Gründen der Rechenzeit in Assembler geschrieben werden, da Assemblerprogramme 50 bis 100mal schneller als BASIC-Programme sind.

Soll einer aus mehr als zwei Programmpfaden ausgewählt werden, dann ist die Fallauswahl (Bild 6) anzuwenden. Vor allem zum Programmbeginn, wenn der Nutzer in einem Menü wählen kann, was er bearbeiten möchte, wird dieses Auswahlprinzip benutzt. Die zugehörige BASIC-Anweisung lautet ON K GOTO Z1 , Z2, . . ., ZN. Ist der Wert der Variablen K = 1, dann wird zur ersten angegebenen Zeilennummer Z1 gesprungen, bei K = 2 zur Zeilennummer Z2 usw. Diese Fallauswahl kann auch zum Sprung in ausgewählte Unterprogramme ON K GOSUB Z1, Z2, . . ., ZN benutzt wert den (Unterprogramm siehe Bild 7).

Eine weitere große Gruppe von Grundstrukturen sind die Schleifen oder Zyklen. Die erste Art ist die sogenannte Abweisschleife (in PASCAL mit WHILE . . . DO BEGIN . . . END), die im BASIC-Interpreter des Z 1013, ebenso wie bei den meisten anderen Kleincomputern, nicht enthalten ist. Hier läßt sich mit IF . . . THEN eine Ersatzkonstruktion schaffen. Auch die Nichtabweisschleife (in PASCAL mit REPEAT . . . UNTIL) ist nicht verfügbar, kann aber über IF . . . THEN realisiert werden. Die dritte Art von Schleifen ist die Zählschleife, über die auch der Interpreter unseres Z 1013 verfügt. Bild 7 zeigt das Struktogramm und die zugehörige BASIC-Anweisung. Beträgt die gewünschte Schrittweite 1, dann kann der Zusatz STEP 1

Grundstruktur	Struktogramm	BASIC - Anweisung
Zählschleife	Für v = 10 bis 60 mit Schrittweite 5 $s: = \frac{\left(\frac{v}{3,6}\right)^2}{2 * a}$ Geschw. v Bremsweg s Nächstes v	FOR . . . TO . . . STEP . . . NEXT
Unterprogramm	Kursorbewegung	GOSUB . . . REM UP KURSORBEWEGUNG . . . RETURN

Bild 7: Zählschleife und Unterprogrammtechnik in BASIC

auch weggelassen werden. In der Anweisung FOR V = 10 TO 60 wird V als Laufvariable bezeichnet. Die Anweisung NEXT V zeigt dem Interpreter das Schleifenende an. Sind Verwechslungen mit anderen Laufvariablen ausgeschlossen, dann kann in der NEXT-Anweisung auch die Laufvariable weggelassen werden. Die Zählschleife eignet sich hervorragend zur Herstellung von Wertetabellen (siehe Programm Bremswegtabelle).

Ubrigens kann auch rückwärts gezählt werden. Dazu ist in jedem Fall eine negative Schrittweite einzugeben, z. B. FOR I = 12 TO -10 STEP - 1. Aber auch mehrfach ineinander geschachtelte Zählschleifen sind möglich. Das kann z. B. für die Belegung oder die Abfrage der im Speicher befindlichen Spielfelder von Nutzen sein. Eine Vielzahl von Anwendungsbeispielen der ZählschIeife zeigen auch die Programme in dieser Broschüre.

Als letzte Grundstruktur soll das Unterprogramm genannt werden (Bild 7). Immer dann, wenn ein bestimmter Programmabschnitt mehrfach benötigt wird, leistet die Unterprogrammtechnik gute Dienste. Man kann ein solches Unterprogramm auch als eigenständigen Modul auffassen, der z. B. bestimmte Kursorbewegungen realisiert, die wiederholte Erzeugung von BiIdschirminhaIten organisiert, die Anzahl der Tage zwischen zwei Kalenderdaten berechnet oder bestimmte Umrechnungen vornimmt. Ein solches Unterprogramm (engl. subroutine) wird mit der BASIC-Anweisung GOSUB X aufgerufen, wobei zur Zeilennummer X gesprungen wird. Allerdings "merkt" sich in diesem Fall der Interpreter (bei GOTO X tut er das nicht) die Stellen, an der er das Hauptprogramm verlassen hat, damit er nach Beendigung seiner Arbeit im Unterprogramm auch mit der nach der GOSUB-Anweisung stehenden Anweisung im Hauptprogramm fortsetzen kann.

Der BASIC-Interpreter unterscheidet, im Gegensatz zu anderen Programmiersprachen, nicht zwischen Variablennamen im Haupt- und im Unterprogramm. Einmal verwendete Variablennamen bleiben also alle im Haupt- und im Unterprogramm gleichermaßen aktiv. Hier gilt es aufzupassen, damit kein "Variablensalat" entsteht. Des weiteren sieht der BASIC-Interpreter (neben der sowieso vorhandenen Zeilennummer) keine besondere Kennzeichnung der Stelle vor, an der ein Unterprogramm beginnt. Wir empfehlen deshalb, ein Unterprogramm mit REM UP Name des Unterprogramms beginnen zu lassen. Damit findet sich der Programmhersteller auch noch nach Monaten in seinem eigenen Programm zurecht. Außerdem ist es sinnvoll, Unterprogramme auf runden Zeilennummern beginnen zu lassen (z. B. 1000, 2000 oder 10000).

Jedes Unterprogramm ist mit der Anweisung RETURN abzuschließen. Dies ist für den Interpreter die Aufforderung, in das Hauptprogramm zurückzukehren, und zwar zu der Anweisung, die auf die zuletzt ausgeführte GOSUB-Anweisung folgt. Damit ist alles wieder im Lot, und das Unterprogramm kann erneut von beliebiger Stelle vom Hauptprogramm aus aufgerufen werden. Ubrigens sind auch mehrfach ineinandergeschachtelte Unterprogrammaufrufe möglich, aber das ist schon wieder etwas für Könner. Wir schließen damit die unvollständige Vorstellung der BASIC-Anweisungen unseres Z 1013 ab. Hier wurde u. a. die Arbeit mit Zahlen- und Zeichenkettenfeldern nicht behandelt. Diese und weitere Anweisungen werden aber in einigen Programmen dieser Broschüre genutzt. Deren Erklärung liefern die im Anhang genannten BASIC-Bücher.

BASIC-Funktionen

Hier sollen die numerischen und die Zeichenkettenfunktionen, die der BASIC-Interpreter unseres Z 1013 bietet, kurz genannt werden. Vor allem die leistungsfähigen Zeichenkettenfunktionen sind es, die BASIC zu solch einer Beliebtheit verholfen haben. Doch zunächst die folgenden numerischen Funktionen:

ABS(X) - liefert den absoluten Betrag des numerischen Ausdruckes X.

INT(X) - liefert die nächstkleinere ganze Zahl von X (integer = ganz). So ergibt z. B. INT (3.14) die Zahl 3, aber INT (-3.14) ergibt die Zahl -4.

SGN(X) - entspricht der Signumfunktion. Sie liefert 1 für X > 0, 0 für X = 0 und - 1 für X<0.

SQR(X) - entspricht . / (square = Quadrat, root = Wurzel).

LN(X) - liefert den natürlichen Logarithmus von X. Die meisten BASIC-Interpreter "verstehen" nur den natürlichen Logarithmus. Wird das Ergebnis durch LN(10) geteilt, so ergibt sich der dekadische Logarithmus.

EXP(X) - entspricht der Exponentialfunktion e^x. Damit kann man sehr leicht die Stellenzahl des Interpreters "auskitzeln". Bis zu welchem X-Wert wird der Interpreter unseres Z 1013 wohl mitspielen?

An trigonometrischen Funktionen bietet unser BASIC-Interpreter SIN(X), COS(X) und TAN(X). Hierbei ist zu beachten, daß der Winkel X im Bogenmaß einzugeben ist. Das gilt übrigens für die Mehrzahl der BASIC-Interpreter und ist nicht tragisch. Wir können getrost den Winkel im Gradmaß eingeben und ihn z. B. der Variablen AL zuweisen. Für die Berechnung eines Sinuswertes schreiben wir dann PRINT SIN (AL*PI/180). Die Zahl π ist im BASIC-Interpreter des Z 1013 fest gespeichert und kann mit PI aufgerufen werden. Als Umkehrfunktion bietet der Interpreter nur die Funktion arc tan, in BASIC-Schreibweise ATN(X). Das Ergebnis ist ein Winkel, aber wieder im Bogenmaß.

Eine seltsame, aber wichtige Funktion für Simulationen und Spiele ist die Funktion RND(X). Für X wird im allgemeinen der Wert 1 gewählt. Solange X größer als 0 ist, fungiert X als Scheinargument, das keinen Einfluß auf das von der Funktion gelieferte Ergebnis hat. RND(1) liefert eine Zufallszahl Z, für die gilt $0 \leq Z < 1$. Allerdings liefert der Interpreter unseres Z 1013 nach jedem Neustart die gleiche Reihe von Zufallszahlen (exakter spricht man von Pseudozufallszahlen), was manchen Spieler oder Vokabellerner schon enttäuscht haben mag. Leider fehlt unserem BASIC-Interpreter die Anweisung RANDOMIZE, die für einen zufälligen Startwert und damit für eine neue Zufallszahlenreihe sorgt. Gibt man für X bei RND(X) eine negative Zahl ein, dann wird ebenfalls ein neuer Startwert und damit eine neue Zufallszahlenreihe erzeugt. Aber diese negative Zahl ist halt auch nicht zufällig, sondern muß von uns vorgegeben werden.

Für die Erzeugung von Zufallszahlen in einem beliebigen Bereich (z. B. Würfel von 1 bis 6, Tele-Lotto von 1 bis 35 usw.) kann folgende Normierungsgleichung Verwendung finden:

LET Z = INT (N*RND(1) + Nl) Nl = Niedrigste ganze Zahl des Bereiches
N = Anzahl der möglichen ganzen Zahlen im Bereich.

Abschließend noch die Vorstellung der Zeichenkettenfunktionen:

ASC(X$) - liefert den Zahlenwert des ASCII-Kodes des ersten Zeichens der Zeichenkette X$. Der ASCII-Kode ist in jedem BASIC-Handbuch abgedruckt, so hat z. B. der Buchstabe A den Kode 65.

CHR$(X) - (Umkehrung zu ASC(X$)) liefert das Zeichen, das zur Kodezahl X im ASCII-Kode gehört.

VAL(X$) - wandelt den als Text vorliegenden Zahlenwert in eine Zahl um, mit der dann auch gerechnet werden kann.

STR$(X) - (Umkehrung zu VAL(X$)) wandelt die Zahl X in eine Zeichenkette um.

LEN(X$) - liefert die Anzahl der Zeichen in der Zeichenkette X$.
Damit kann man z. B. auf einfache Weise überprüfen, ob eine Personenkennzahl tatsächlich mit 12 Stellen eingegeben wurde.

LEFT$(X$,K) - liefert die ersten K Zeichen der Zeichenkette X$. Mit LEFT$(PK$,6) kann z. B. das Geburtsdatum aus der Personenkennzahl "heraus gefiltert" werden.

RIGHT$(X$, K) - liefert die letzten K Zeichen der Zeichenkette X$.

MID$(X$, K, I) - liefert I Zeichen der Zeichenkette X$, wobei mit dem K-ten Zeichen begonnen wird. Die Funktion MlD$(PK$, 7, 1) liefert z. B. das 7. Zeichen der Personenkennzahl und damit das Geschlecht der Person.

STRING$(K, X$)- liefert die K-malige Wiederholung der Zeichenkette X$. So wird z. B. die Unterstreichung eines unter der Variablen UE$ abgespeicherten Textes mit beliebiger Länge durch folgende Anweisung und Funktion korrekt erledigt: PRINT STRING$(LEN(UE$),"-").

INSTR(X$, Y$) - X$ ist hier eine Teilzeichenkette. Die Funktion untersucht, ob diese Teilzeichenkette X$ vollständig in der Zeichenkette Y$ enthalten ist. Wenn das der Fall ist, dann liefert die Funktion die Position des ersten Auftretens von X$ in Y$. Wird die Zeichenkette nicht gefunden, liefert die Funktion den Wert 0. Die Funktion ist bei Suchprogrammen (z. B. in unserem Bibliotheksprogramm bei der Suche nach dem Titel) wertvoll, da sie den Computer "großzügiger" macht. Steht z. B. in der Variablen T$ der Buchtitel "DER SCHATZ IM SILBERSEE", dann liefert INSTR("SILBERSEE", T$) den Wert 15. Da dieser Wert ungleich Null ist, gilt der Buchtitel als gefunden. Vor der Behandlung von Zeichenketten muß gesichert sein, daß der String nicht leer ist (z. B. nach Programmstart noch nicht geladen), sonst führen diese Befehle zu Fehlerbildern, deren Ursache schwer zu deuten ist.

Wir schließen jetzt die unvollständige Vorstellung von Kommandos, Anweisungen und Funktionen für den BASIC-Interpreter unseres Z 1013 ab und haben damit einen wichtigen Grundstein für die nun beginnende praktische Tätigkeit gelegt.

Die Arbeit mit dem BASIC-Interpreter des Z 1013

Von dem, was der BASIC-Interpreter unseres Computers zu leisten vermag, besitzen wir jetzt eine Vorstellung. Wenden wir uns der Arbeit damit zu.

Starten des BASIC-Interpreters

Der BASIC-Interpreter befindet sich auf einer Magnetbandkassette, von der er in den Arbeitsspeicher des Z 1013 geladen werden muß. Nach der Inbetriebnahme von Computer und Fernsehgerät meldet sich der Z 1013 im Grundzustand. Auf dem Bildschirm wird sichtbar:

```
robotron Z 1013 / 2.02
```

Das ist die Aufforderung zur Eingabe eines Maschinenprogrammes. Der BASIC-Interpreter stellt ein solches Maschinenprogramm dar. Z 1013 und Recorder werden mit einem Diodenkabel verbunden. Dann ist die Interpreterkassette auf die Anfangsmarkierung zu positionieren. Auf der Flachfolientastatur muß die Anweisung zum Einlesen des Interpreters in Monitorkommandos geschrieben werden. Sie lautet:

```
L 100 2AFF
```

Der Recorder wird durch Drücken der Wiedergabetaste gestartet. Bei Ertönen des Pilottones ist die ENTER-Taste (bezeichnet mit ENT) des Z 1013 zu betätigen. Damit beginnt der Ladevorgang, der etwa 40 Sekunden andauert. Der Kursor bewegt sich dabei nicht. Erst nach fehlerfreiem Einlesen des Interpreters springt der Kursor eine Zeile weiter. Eine neue Eingabe wird erwartet. Mit der Eingabe

```
J 300
```

und Betätigung der ENTER-Taste ist der Interpreter zu starten. Er gibt folgende Meldung aus:

```
HC-BASIC
MEMORY SIZE ?:
```

Soll kein Speicherplatz für ein weiteres Maschinenprogramm reserviert werden (anderenfalls müßte die obere Grenze des BASIC-Programmspeichers als Dezimalzahl eingegeben werden), ist einfach die ENTER-Taste zu betätigen. Der Z 1013 teilt uns daraufhin mit, daß er 4846 Bytes für die weitere Arbeit bereitstellt, die für BASIC-Programme und Daten genutzt werden können. Bei Anschluß einer externen Speichererweiterung erhöht sich dieser Wert entsprechend der Zahl der RAM-Module.

Erarbeiten und Nutzen von Programmen

Nach dem erfolgreichen Start des Interpreters wird das "BASIC mit dem Z 1013" Realität. Unter Berücksichtigung des Grades der Beherrschung der Programmiersprache können erste kleine Programme selbst erprobt werden. Wer sich da noch nicht herantraut, sollte vielleicht einfache Programme auswählen, die auf den folgenden Seiten zur Nutzung angeboten werden und sich in der Eingabe üben. Es ist aber auch möglich, BASIC-Programme für den Z 1013, die auf einer Kassette gespeichert wurden, mit dem Befehl CLOAD "Programmname" in den Computer einzuladen und abzuarbeiten.

An dieser Stelle soll darauf hingewiesen werden, daß ein fehlerfreies Einlesen eines Programms ein sicher funktionierendes Tonbandgerät mit konstanter Laufgeschwindigkeit sowie mit sauberem und exakt justiertem Tonkopf erfordert. Ein starker Signalpegel bei der Aufnahme eines Programms (Handaussteuerung mit relativ weit aufgedrehtem Regler) erspart Sorgen beim späteren Einlesen.

Verlassen und Neustarten des BASIC-Interpreters

Der BASIC-Interpreter kann auf zwei verschiedenen Wegen verlassen werden. Den Befehl BYE und anschließende Betätigung der ENTER-Taste nutzt man, wenn planmäßig im Maschinencode weitergearbeitet werden soll. Möchte man das gerade bearbeitete BASIC-Programm retten, ist unverzüglich

```
J 302
```

einzugeben und die ENTER-Taste zu drücken (Restart). Genügt es, den Interpreter zu aktivieren, wird

```
J 300
```

geschrieben und die ENTER-Taste betätigt.

X **@**	**Y** **A**	**Z** **B**	**[{** **C**	**\ |** **D**	**] }** **E**	**^** **F**	**_** **G**
0 **H**	**1 !** **I**	**2 "** **J**	**3 #** **K**	**4 $** **L**	**5 %** **M**	**6 &** **N**	**7 '** **O**
8 (**P**	**9)** **Q**	**: *** **R**	**; +** **S**	**< ,** **T**	**= -** **U**	**> .** **V**	**? /** **W**
S 1	**S 2**	**S 3**	**S 4**	←	└-------┘	→	**Ent**

Bild 8: Flachfolientastatur des Z 1013

Arbeit mit der Tastatur

Die Original-Flachfolientastatur des Z 1013 besitzt 8 x 4 Tastenfelder (Bild 8). Die Tastenanordnung erscheint dem sachkundigen Nutzer anderer Computertypen ungewohnt. Zwar sind Flachfolientastaturen auf dem Computermarkt nicht so selten, aber sie besitzen in der Regel eine andere Anordnung. Die übliche Anordnung greift auf eine schreibmaschinenähnliche Lage der alphanumerischen Zeichen zurück. Die Tastatur des Z 1013 dagegen ist völlig nach dem Alphabet geordnet. Dies erleichtert vor allem schreibmaschinenungewohnten Nutzern für den Anfang das Auffinden der einzelnen Buchstaben. Die untere Reihe der Tastatur besitzt von links aus die Umschalttasten S1 bis S4. Deren Aufgaben sind der Bedienungsanleitung zu entnehmen. Im Zusammenhang mit dem BASIC-Interpreter ist es wichtig zu wissen, daß mit der Taste S4 Umschaltungen auf Steuerfunktionen bewirkt werden, die zur Erhöhung des Bedienkomforts des Z 1013 beitragen.

Tastenkombination	Bezeichnung	Codierung	Bedeutung
S4 / @	DEL	31	Zeichen löschen
S4 / A	Alpha	17	Alphaumschaltung
S4 / B	INS	26	Einfügen eines Leerzeichens auf die Stelle vor der aktuellen Kursorposition
S4 / C	ESC	27	Escape
S4 / D	LIST	28	listet BASIC-Programm
S4 / E	RUN	29	startet BASIC-Programm
S4 / F	CONT	30	Programmfortsetzung
S4 / G	Grafik	25 ?	Umschaltung auf Grafik
S4 / J	CL LN	2	Eingabezeile löschen
S4 / K	STOP	3	Programmabbruch
S4 / P	←	8	Kursor nach links
S4 / Q	→	9	Kursor nach rechts
S4 / R	↓	10	Kursor nach unten
S4 / S	↑	11	Kursor nach oben
S4 / T	CLS	12	Bildschirm löschen
S4 / U	ENT	13	Enter
S4 / V	|←	25 ?	Kursor an den Zeilenanfang
S4 / W	→|	24	Kursor an das Zeilenende

Nutzung der Grafiksymbole

Die Umschaltung S4 / G macht es möglich, 119 verschiedene Grafiksymbole (vgl. Anlage 7 des Handbuches Teil II B zum Z 1013) direkt über die Betätigung der zugeordneten Tasten auf den Bildschirm bzw. in ein BASIC-Programm zu bringen. Mit Hilfe dieser Symbole, die sich natürlich auch über die CHR$-Funktion mit Angabe des entsprechenden Zahlenwertes der ASCII-Tabelle aufrufen lassen, ist es möglich, Zeichnungen, Bilder und Spielfiguren in großer Vielfalt zu konstruieren.

Der Z 1013 konkret

Die Fähigkeiten eines Computers

Für den optimistischen Zeitungs- und Zeitschriftenleser kann ein Computer fast alles. Da wird von Computern berichtet, die das Wetter vorhersagen, Musikstücke analysieren, Autos, Flugzeuge, Triebfahrzeuge und Schiffe rentabel steuern, Weinfälschungen nachweisen, 80 Prozent des Expertenwissens eines Facharztes parat haben, Vorgänge aus Natur und Technik simulieren, chinesische Schriftzeichen verarbeiten, Sprachen übersetzen und sogar gesprochene Sätze in geschriebene Texte umwandeln. Darüber hinaus berechnen Computer Konstruktionen, erstellen Zeichnungen, liefern die erforderlichen Unterlagen für die Fertigung, steuern Produktionsprozesse, geben Fahrplan und Stadtinformationen, verkaufen Fahrkarten, helfen in Sparkassen und Versicherungen, sortieren Postsendungen und ermöglichen trickreiche Darstellungen im Fernsehen.

Aus dieser Aufzählung wird klar, daß der Computer ein Werkzeug (Arbeitsmittel) von außerordentlich großer Komplexität und volkswirtschaftlicher Bedeutung ist. Zugleich ist er aber für den Berufs- und Hobbyelektroniker ein Arbeitsgegenstand. Diese Doppelfunktion, gepaart mit seiner Vielfältigkeit, führt nicht selten in Industrie, Wissenschaft und Technik, aber auch im privaten Bereich, zu Unklarheiten bezüglich der Möglichkeiten und Grenzen eines bestimmten Computertyps. So zählt ein Mensch in 30 s etwa bis 100, der BASIC-Interpreter unseres Z 1013 bis rund 5000, ein Superrechner hingegen zählt bis 500 Milliarden. Die Berücksichtigung der Leistungsklasse eines Computers ist demnach für eine sachgetreue Nutzung, die Enttäuschungen weitgehend ausschließt, unumgänglich. Auf die Klasse der Mikrorechnerbausätze (hierzu gehört unser Z 1013) und Kleincomputer folgt die Klasse der Büros und Personalcomputer (A 5110/20/30, PC 1715, A 7100). Ihr folgt die Klasse der Kleinrechnersysteme, die sich zu CAD/CAM-Arbeitsstationen ausbauen lassen (z. B. AKT 6454). Dabei ist zu beachten, daß beim Übergang zur nächsthöheren Leistungsklasse die Preise um den Faktor 10 bis 20 und mehr steigen können. Eben diese Leistungsklasse bestimmt die Genauigkeit, Schnelligkeit, Zuverlässigkeit und Eleganz der Verarbeitung von Informationen. Dennoch bieten Computer prinzipiell folgende spezifische Möglichkeiten. Ein Computer kann:

- Prozesse steuern (Anschluß über Sensoren und Stelleinrichtungen),
- schnell und genau rechnen (Umgang mit Zahlen),
- Buchstaben und Buchstabenketten speichern, sortieren und darin suchen (Umgang mit Text),
- Pseudografikzeichen und Bildpunkte (Pixel) auf den Bildschirm setzen (Punkte, Linien, Flächen erzeugen),
- Farben auf den Bildschirm bringen,
- Zufallszahlen erzeugen (würfeln) und damit Zahlen, Texte, Pseudografikzeichen, Pixel und Farben zufällig manipulieren.

Die kombinierte Nutzung dieser Möglichkeiten ergibt das komplexe Werkzeug Computer. Eine kleine Auswahl von Einsatzfällen sind die Überwachung oder Steuerung bestimmter Abläufe, die Durchführung zeitaufwendiger Berechnungen, die Herstellung und das Redigieren von Texten, das Suchen in Dateien, die Nutzung als "Wissensspeicher" beim Problemlösen, die Förderung von Lehren, Lernen und Spiel und die Nutzung als kreatives Werkzeug für den Konstrukteur und den Künstler.

Nun kann natürlich nicht jeder Computer alles. Können sie aber das gleiche tun, dann meist nicht in gleicher Qualität. So haben z. 8. die Hersteller von Mikrorechnerbausätzen und von Kleincomputern, vor allem aus ökonomischen Erwägungen heraus, an ausgewählte

Einsatzfälle vorrangig gedacht. Folglich fehlt unserem Z 1013 der Umgang mit Pixeln (Vollgrafik) und mit Farbe, aber alle anderen genannten Fähigkeiten lassen sich nutzen. Daß dies schon mit wenig Aufwand eine Masse Computerwissen und Computerspaß zu vermitteln vermag, sollen die folgenden Programme zeigen.

Jetzt nutzen wir den Z 1013

Werkzeugkiste für Software des Z 1013

Das Vertrautsein mit der technischen Bedienung des Z 1013 und den Grundlagen der höheren Programmiersprache BASIC sind unerläßliche Voraussetzungen, um den Computer erfolgreich nutzen zu können. Bei der praktischen Arbeit merkt man allerdings schnell, daß dies allein noch nicht ausreicht. Die Erfahrung im Umgang mit dem Mikrorechner bringt jedem Nutzer täglich neue Erkenntnisse, wie etwas "so" oder "besser" zu machen ist. Probleme einer optimalen und dennoch gut überschaubaren Bildschirmgestaltung, Arbeit mit POKE und PEEK, immer wieder benötigte Progammroutinen und vieles andere möchte jeder Anwender für das eigene Programmieren effektiv beherrschen. Einige dieser Kniffe packen wir in unsere "Werkzeugkiste".

Bildschirmgestaltung

Der Bildschirm, den unser Z 1013 anspricht, teilt sich in ein Quadrat von 32 Zeilen und 32 Spalten auf. Dies bedeutet, daß wir beim Programmieren maximal 1024 Bildspeicherplätze gleichzeitig belegen können. Der BASIC-Interpreter bietet dafür zahlreiche Möglichkeiten an, die es sinnvoll zu nutzen gilt. Jeder Bildspeicherplatz besitzt seine eigene Adresse, die wir kennen und ansprechen müssen, wenn etwas auf dem Bildschirm an einer ganz bestimmten Stelle sichtbar werden soll. Für die praktische Arbeit schaffen wir uns zunächst ein Hilfsmittel. Aus kleinkariertem Papier schneiden wir ein Raster von 32 X 32 Karos aus und kleben es auf starken Kanon. Darauf werden die Speicherplätze so aufgetragen, wie es die Zeichnung (Bild 9) darstellt. Über dem Karton befestigen wir eine stabile, farblose, abwischbare Folie, die an den Rändern verklebt wird. Mit Foliestiften oder wasserlöslicher Auszieh-tusche kann letzt jede gewünschte Bildschirmaufteilung vor dem Programmieren ausprobiert werden. Die dann anzusprechenden Speicherplätze sind aus dem Raster ablesbar. Dafür gibt es immer zwei mögliche Werte. Jeder Speicherplatz läßt sich als Punktepaar (x, y) oder als Einzelwert mit. negativem Vorzeichen darstellen. Der x-Wert gibt die Zeilenposition, der y-Wert die Spaltenposition an. Den EinzeIwert ermittelt man, indem zum negativen Wert, der auf der Zeichnung am jeweiligen Zeilenanfang links sichtbar ist, der Spaltenwert addiert wird. So entspricht zum Beispiel dem Bildschirmplatz P(5,9) der Speicherplatz -4951. Die negativen Zahlen rühren daher, daß der Bildwiederholspeicher im hexadezimalen Adressenbereich E000 bis EFFF oder im dezimalen Bereich 60416 bis 61439 liegt. Dieser Adressenbereich kann durch negative ganze Zahlen adressiert werden, die die Differenz zwischen der Speicherplatzadresse und 2^{16}=65536 darstellen.

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	. . . . .	29	30	31
-5120	0																		
-5088	1																		
-5056	2																		
-5024	3																		
-4992	4																		
-4960	5																		
-4928	6																		
-4896	7																		
-4864	8																		
-4832	9																		
-4800	10																		
-4768	11																		
-4736	12																		
-4704	13																		
-4672	14																		
-4640	15																		
-4608	16																		
-4576	17																		
-4544	18																		
-4512	19																		
-4480	20																		
-4448	21																		
-4416	22																		
-4384	23																		
-4352	24																		
-4320	25																		
-4288	26																		
-4256	27																		
-4224	28																		
-4192	29																		
-4160	30																		
-4128	31																		

Bild 9: Raster zum Bildwiederholspeicher des Z 1013

Möglichkeiten mit PRINT

Mit den unterschiedlichen Formen der PRINT-Anweisung wird immer ein Bildschirmplatz angesprochen.

PRINT"Ausdruck" beginnt die Darstellung auf dem Bildschirm am linken Zeilenrand auf der Spaltenposition 0. Die Anweisung nutzt maximal alle 32 Plätze der Zeile zum Schreiben. Dann erfolgt automatisch der Sprung an den linken Rand der nächsten Zeile. Auch eine neue PRINT-Anweisung bewirkt immer den Sprung an den Anfang der folgenden Zeile.

PRINT TAB(Y)"Ausdruck" schafft die Möglichkeit zum tabellarischen Darstellen. In der Zeile wird das erste Zeichen des Ausdruckes in die Spaltenposition Y geschrieben. Der Zeilenwechsel erfolgt genauso automatisch wie bei PRINT"Ausdruck". Durch die fortgesetzte Verwendung von PRINT TAB entsteht die Tabellenform.

PRINT AT(X,Y);"Ausdruck" verwirklicht den Beginn der Darstellung des Ausdruckes genau auf Bildschirmplatz der X-ten Zeile und der Y-ten Spalte.

PRINT CHR$(N) ist in allen vorgenannten Versionen einsetzbar. Statt eines konstanten Ausdruckes bewirkt die CHR$-Funktion die Darstellung des Zeichens, das im ASCII-Code den Wert N besitzt. Die Anwendung der CHR$-Funktion ist besonders für die Verwendung der Pseudografiksymbole zu empfehlen.

PRINT SPC(Z)"Ausdruck" ermöglicht es, vor den Ausdruck oder auch zwischen zwei Ausdrücke genau Z Leerzeichen zu setzen.

Möglichkeiten mit WINDOW

WINDOW X1,X2,Y1,Y2 legt innerhalb des 32 X 32 Rasters ein aktuelles Ausgabefenster fest. Das Fenster wird durch die Bildschirmplätze der Punktpaare P1(x1,y1), P2(x1,y2), P3(x2,y1) und P4(x2,y2) begrenzt. Nachfolgende PRINT- und INPUT-Anweisungen beziehen sich ausschließlich auf das aktuelle Fenster. Lediglich die PRINT-AT-Anweisung machte eine Ausnahme. Mit ihrer Hilfe kann man auch nach der Festlegung des Fensters den gesamten Bildschirmbereich ansprechen. Die Anweisung WINDOW ohne Angabe von Zeilen- und Spaltenpositionen erklärt den gesamten Bildschirm zum aktuellen Fenster und hebt damit die Einschränkung auf, die von WINDOW Xl,X2,Y1,Y2 vorgenommen wurde.

Möglichkeiten mit POKE

Durch die POKE-Anweisung wird der einzelne Bildspeicherplatz direkt und sehr schnell angesprochen.

POKE -4951,N setzt auf den Speicherplatz mit der Adresse -4951 das Zeichen des ASCII-Codes mit dem Wert N. POKE eignet sich gut für die Darstellung von Grafikzeichen und für Spielabläufe. Der Programmieraufwand ist geringer als mit PRINT-Anweisungen. Allerdings muß man bei der Bestimmung der Speicherplätze gut aufpassen.

POKE-Anweisungen werden nicht nur zum Ansprechen der Bildspeicherplätze verwendet, die beim Z 1013 im Bereich zwischen -5120 und -4097 liegen. Mit POKE erreicht man auch solche Speicherplätze, die auf Betriebsabläufe des Computers Einfluß nehmen. Das zu wissen hilft uns, bestimmte Zusatzfunktionen zu nutzen, die der BASIC-Interpreter auf den ersten Blick nicht bietet.

Bei vielen Programmen soll aus Gestaltungsgründen eine INPUT-Anweisung auf einen beliebigen Platz P(x,y) des Bildschirmes gesetzt werden. Die Lösung mit Hilfe von WINDOW ist oft zu umständlich, weil das aktuelle Fenster nach der Eingabe aufgehoben werden muß. Eine bequemere Lösung funktioniert so:

```
Zeilennummer POKE 112,Y : POKE 113,X : INPUT "Eingabe";E
```

Zu beachten ist lediglich, daß aus Gründen der internen Organisation nach INPUT und Anführungszeichen ein Leerzeichen vor das erste Zeichen des Ausdruckes zu setzen ist.

Mit der Funktion PEEK (Speicherplatzadresse) kann erfragt werden, ob und wie der angesprochene Speicherplatz belegt ist. Die Anwendung dieser Möglichkeit wird im Unterprogramm Begrenzungsrahmen und im Programm GALTON-Brett demonstriert.

Verkettung von zwei BASIC-Programmen

Die Notwendigkeit, zwei verschiedene BASIC-Programme miteinander zu verknüpfen, wird sich öfter ergeben. Es ist jedoch nicht ohne weiteres möglich, zwei Programme gleichzeitig von der Kassette in den Arbeitsspeicher des Computers zu laden. Auch hier hilft ein Griff in die "Werkzeugkiste".

Vorbereitung des ersten Programmes:

Das Prograrnm soll aus den Zeilen 10 bis 100 bestehen. Wir fügen zwei weitere, zunächst scheinbar bedeutungslose Zeilen hinzu

```
  10
   :   normales Programm
 100
 110   GOSUB 1000
 120   END
```

Dieses Programm wird mit CSAVE "Name 1" auf Kassette gespeichert.

Vorbereitung des zweiten Programmes:

Das Programm soll aus den Zeilen 1000 bis 2000 bestehen. Ein Hilfsprogramm aus 4 Zeilen wird hinzugefügt.

```
   10   PRINT "Verkettungshilfe"
   20   GOSUB 1000
   30   END
 1000
    :   normales Programm
 2000
 2010   Return
```

Dieses Programm wird mit **LIST # 1 "Name 2"** 1000 extern auf der Kassette ausgelagert.

Verbinden der Programme 1 und 2:

Das erste Programm wird mit **CLOAD "Name 1"** in den Computer eingelesen.

Das zweite Programm wird mit **LOAD # 1 "Name 2"** von der Kassette in den Arbeitsspeicher des Computers gerufen.

Die nicht mehr benötigten Hilfszeilen 110, 120 und 2010 werden gelöscht.

Die Identität der Anweisung GOSUB 1000 in beiden Programmen wurde genutzt, um das zweite Programm als Quellprogramm zu speichern und dann in das Arbeitsprogramm einzuordnen.

Schutz eines Programms

Manchmal wünscht sich der Programmierer, daß seine erarbeitete Software gegen unerlaubtes Kopieren geschützt sein soll. Das ist möglich, wenn man die Adresse von Speicherplätzen kennt, die solchen Schutz organisieren können. Ein Programm für den Z 1013 Iäßt sich gegen unberechtigte Eingriffe schützen, indem die Anweisung zum Abspeichern auf Kassette in folgende Form gekleidet wird:

```
POKE 11101,1 : CSAVE "Name"
```

Liest man das Programm von der Kassette erneut in den Speicher des Z 1013 ein, Iäßt es sich normal abarbeiten. Versucht man jedoch einen Eingriff vorzunehmen, passiert folgendes:

LIST	EDIT	CSAVE "NAME"
oder	oder	
ERROR	ERROR	ERROR

Der Schutz Iäßt sich durch Eingabe der Anweisung

POKE 11102,0 aufheben.

Nutzung der Programme anderer Computertypen

Das Kassetteninterface des Z 1013 ist bei Benutzung des 10 KByte-BASIC-Interpreters identisch mit den Kassetteninterfaces der Kleincomputer KC 85/1 und KC 85/2/3. Dies bedeutet, daß man Programme von der Kassette auch in jeden anderen Computer der oben genannten Typen einladen kann. Eine uneingeschränkte Nutzung dieser Programme ist jedoch nicht ohne weiteres möglich. Dazu muß erst geprüft werden, ob Interpreter und Programmgestaltung auch für beide Computer übereinstimmen. Oft lassen sich Programme des anderen Computertyps durch wenige Eingriffe so verändern, daß sie für den Z 1013 voll genutzt werden können. Der folgende Prüfvorgang soll helfen, diese Veränderungen systematisch vorzunehmen.

	Problem	Prüfung und Abänderung
1.	Speicherplatzbedarf	Ermitteln der Speicherkapazität des Programms durch PRINT FRE(X). Ist die erforderliche Speicherkapazität kleiner als 4846 Bytes, kann das Programm in der Grundvariante des Z 1013 weiter bearbeitet werden.
2.	Bildspeicher	Folgende Anweisungen im Listing sind zu überprüfen und gegebenenfalls zu verändern:
	Z1013 32 Zeilen, 32 Spalten	- WINDOW - PRINTAT
	KC 85/1 24 Zeilen, 40 Spalten	- PRINT TAB - Zeilenlänge der PRINT-Anweisung - Silbentrennung in PRINT-Anweisungen
	KC 85/2/3 32 Zeilen, 40 Spalten	- POKE für Bildspeicher
3.	Farbanweisungen	Der Z 1013 ignoriert Farbanweisungen. Um Speicherplatz zu sparen, können sie gelöscht werden.
4.	POKE-Anweisungen außerhalb des Bildspeichers	Hier muß probiert werden. Da die Betriebssysteme unterschiedlich organisiert sind, ist meist keine Abhilfe möglich.
5.	BEEP und SOUND	Die Anweisungen zur Tonerzeugung sind grundsätzlich anders aufgebaut. Eine entsprechende Änderung des Programms ist nicht ohne weiteres möglich.
6.	Vollgrafik des KC 85/2/3	Der Z 1013 verfügt nicht über Vollgrafik. Prinzipiell läßt sich natürlich auch aus dem Zeichensatz der Pseudografik fast alles zusammenfügen. Das ist für einen Programmumbau jedoch zu aufwendig. Hier sollte das Programm lieber gleich für den Z 1013 neu geschrieben werden.

Vielseitig verwendbare Unterprogramme

Beim Programmieren stellt man fest, daß viele Bestandteile eines Programms auch in anderen Programmen erneut vorkommen oder genutzt werden könnten. Das sind meistens recht kurze Unterprogramme oder Programmroutinen. Einige davon sollten den Inhalt unserer "Werkzeugkiste" bereichern.

Unterprogramm:
Steuern bzw. Zeichnen eines definieren Zeichens auf dem Bildschirm (636 Bytes)

Dieses Unterprogramm ist notwendiger Bestandteil vieler Spielprogramme, bei denen Bewegungsabläufe durch den Nutzer selbst gesteuert oder Figuren auf dem Bildschirm gezeichnet werden sollen. Die neun Tasten des rechten oberen Teils der Folientastatur stellen das "Steuerpult" dar (Bild 10). Die Lage eines Buchstabens zum Buchstaben N bestimmt die Bewegungsrichtung.

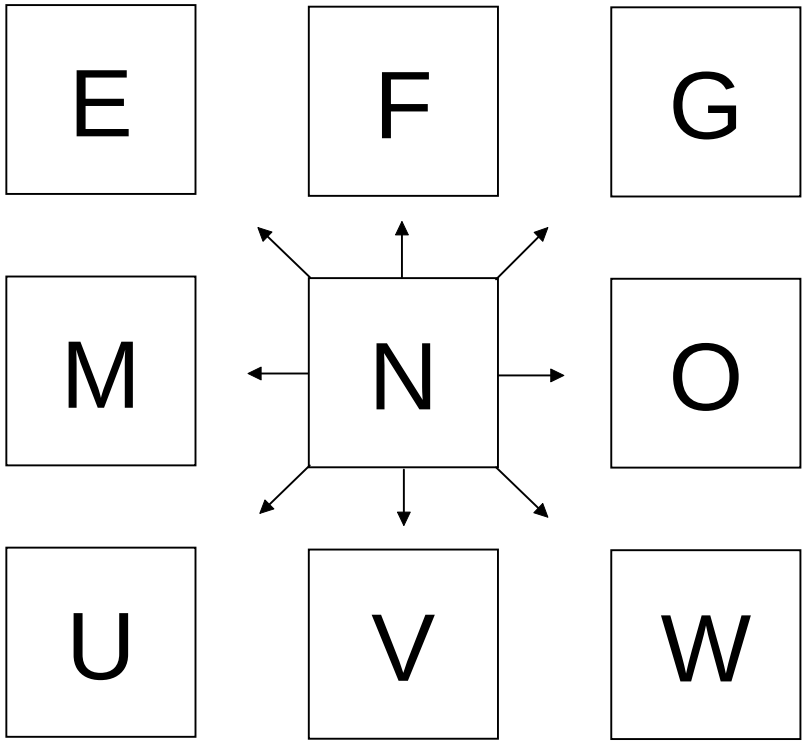


Bild 10: Steuerpult für Spiel- und Zeichenprogramme

Unter Ausnutzung der Aufteilung des Bildspeichers in Speicherplätze ist jede Veränderung eines Bildpunktes durch entsprechende Addition bzw. Subtraktion zum ursprünglichen Speicherplatz so zu organisieren, daß acht verschiedene Wechsel möglich werden. Das Setzen von Z = 0 in Zeile 1310 gewährleistet, daß zufällige andere Tastaturberührungen im Steuerablauf unberücksichtigt bleiben. Das Programm startet grundsätzlich mit der Funktion "Steuern", d. h. nach vollzogenem Platzwechsel des Zeichens wird der vorhergehende Bildschirmplatz gelöscht. Die Umschaltung auf die Funktion "Zeichnen" erfolgt durch

Tastatureingabe "P". Nach dieser Anweisung wird ein Löschen des früheren Bildschirmplatzes nicht wirksam, die früheren Bildschirmpositionen des Zeichens verbleiben auf dem Terminal, und die fortgesetzte Zeichenposition stellt sich als Zeichnung dar. Eine Tastatureingabe "H" schaltet auf "Steuern" zurück. Mit der Tastatureingabe "@" wird das Programm beendet (Listing 1).

```
1000 REM  UP PUNKT STEUERN/ZEICHNEN
1010 PRINT "EINGABE DER STARTPOSITION AUF":PRINT
1020 PRINT "DEM BILDSCHIRM":PRINT
1030 PRINT "(WERT ZWISCHEN -5120 UND -4097)" :PRINT
1040 INPUT X:CLS:S=1:PRINTAT(0,0);CHR$(32)
1120 P=42 : POKE X,P:REM P IST WERT DES ASCII
1130 T$=INKEY$
1140 IF T$="U" THEN Z= 31
1150 IF T$="V" THEN Z= 32
1160 IF T$="1" Z= 33
1170 IF T$="M" THEN Z= -1
1180 IF T$="N" THEN Z=  0
1190 IF T$="O" THEN Z=  1
1200 IF T$="E" THEN Z=-33
1210 IF T$="F" THEN Z=-32
1220 IF T$="G" THEN Z=-31
1230 Y=X+Z
1260 IF T$="5" THEN 1350:REM ENDE
1270 IF T$="H" THEN S=1:REM STEUERN
1280 IF T$="P" THEN S=2:REM ZEICHNEN
1290 IF S=1 THEN 1310
1300 IF S=2 THEN 1320
1310 Z=0:POKE Y,P :POKE X,32:GOTO 1330
1320 POKE Y,P
1330 X=Y
1340 GOTO 1120
1350 END
```

Listing 1: Zeichnen auf dem Bildschirm

Zeilen 1000 bis 1040

Die Ausgangsposition des Punktes ist als Zahlenwert eines Bildspeicherplatzes einzugeben.

Zeile 1100

Mit Hilfe der POKE-Anweisung wird das Zeichen gesetzt. Der vorgegebene Wert P entspricht im Beispiel dem Sternchen-Zeichen. Jeder beliebige andere Wert des ASCII-Zeichensatzes kann ebenfalls dafür gewählt werden.

Zeilen 1130 bis 1230

Die INKEY$-Abfrage der Tasten des "Steuerpultes" bewirkt die Veränderung der Bildschirmposition durch die Zuweisung eines konkreten Wertes Z, der die Bildschirmposition zum augenblicklichen Standort neu bestimmt.

Zeilen 1260 bis 1300

Die Organisation der Steuern/Zeichen-Umschaltung erfolgt durch Tastaturabfrage, Zuordnung von Werten S = 1 bzw. S = 2 und entsprechende Sprunganweisungen.

Zeilen 1310 bis 1350

Mit den POKE-Anweisungen wird das Neusetzen bzw. Löschen des Zeichens bewirkt. Durch GOTO-Befehl nach 1120 wird die Wiederholung des Steuer/Zeichen-Vorganges eingeleitet.

Unterprogramm: Begrenzungsrahmen (203 Bytes)

Dieses Unterprogramm stellt eine sinnvolle Ergänzung zum vorangegangenen Unterprogramm (Steuern eines Zeichens) dar. Sein Einsatz ist aber auch für andere Anwendungsfälle denkbar. Wir beziehen uns bei der Erläuterung auf das vorhergehende Programmbeispiel .

Das Zeichen läßt sich nach jeder beliebigen Seite aus dem Bildschirm heraussteuern und wird dabei unsichtbar. Dieser Effekt ist für Spiele störend. Wir wollen deshalb einen Rahmen "bauen", der den Bildschirmrand begrenzt und den gesteuerten Punkt beim Anprall stoppt. Dazu muß allerdings ein Zeichen gewählt werden, das sich vom gesteuerten Zeichen eindeutig unterscheidet. Wir zwingen den gesteuerten Punkt vor jedem Positionswechsel über die PEEK-Anweisung zu der Frage, ob der neue Platz noch frei ist. Liegt keine Besetztantwort vor, wird die gewünschte neue Position eingenommen. Wird über PEEK eine Besetztmeldung gegeben, verbleibt das gesteuerte Zeichen auf seinem alten Platz. Damit wird das Verlassen des Bildschirmes unmöglich. Die Zeilennumerierung des Programms wurde so vorgenommen, daß es sich nahtlos in das vorangegangene Programm einkoppeln läßt (Listing 2).

```
1050 REM RAHMEN DES BILDSCHIRMES
1060 REM IN UP PUNKTSTEUERUNG EINFUEGEN
1070 FOR I=0 TO 31:POKE(-5120+I),43:POKE(-4128+I),43
1080 POKE(-5120+(32*I)),43:POKE(-5089+(32*I)),43:NEXT I
1090 GOTO1120
1100 IF PEEK(Y)=43 THEN Y=X
1110 RETURN
1250 GOSUB1100
```

Listing 2: Herstellung eines Begrenzungsrahmens

Zeilen 1070 bis 1080

Der Rahmen wird gesetzt. Dazu werden die Bildspeicherplätze der Randpositionen durch Schleifenanweisungen über POKE belegt.

Zeile 1100

Diese Zeile stellt ein eigenes Unterprogramm dar. Sie erforscht, ob die gewünschte neue

Steuerposition frei oder besetzt ist. Im Besetztfall weist sie die alte Position als neue Position zu.

Zeile 1250

Diese Zeile gehört in das Steuerprogramm. Sie wird hier nur der Vollständigkeit halber ausgewiesen. Diese Zeile bewirkt nach der INKEY$-Abfrage die Einleitung des Prüfvorganges im Unterprogramm der Zeile 1100.

Unterprogramm: Alphabetisches Ordnen von Begriffen (390 Bytes)

Dieses Programm kann durch eine entsprechende Gestaltung zum selbständigen Programm ausgebaut oder als Hilfsprogramm in eine entsprechende Datei eingebaut werden. In der vorliegenden Variante können bis zu 300 eingegebene Begriffe alphabetisch sortiert werden. Dies läßt sich durch eine entsprechende Änderung der DIM-Anweisung auch noch erweitern, wenn das der verfügbare Speicherbedarf des vorgesehenen Gesamtprogramms erlaubt. Diese Entscheidung bleibt dem Nutzer überlassen. Die interne Effektivität des Sortiervorganges ermöglicht ein sehr schnelles Arbeiten. Dies wird möglich, weil das Wortfeld W(Z) einem Indexfeld A(Z) auf der Grundlage des ASCII-Codes zugeordnet und nur nach dem numerischen Wert der Zahlen A(Z) intern sortiert wird (Listing 3).

```
1000 REM  UP ALPHABETISCHES ORDNEN
1010 DIM W$(300),A(300)
1020 CLS:PRINT"STICHWORTE ZUM ORDNEN EINGEBEN":PRINT
1030 PRINT"EINGABE MIT  *  BEENDEN":PRINT
1040 INPUT W$
1050 IF W$="*" THEN1160
1060 Z=Z+1
1070 B$=LEFT$(W$,1)
1080 W=ASC(B$)
1090 W$=CHR$(W) + RIGHT$(W$,LEN(W$)-1)
1100 W$(Z)=W$:A(Z)=Z
1110 FOR K=Z TO 1 STEP-1
1120 IF W$(A(K))>W$(A(K-1)) THEN1040
1130 Y=A(K):A(K)=A(K-1):A(K-1)=Y
1140 NEXT K
1150 GOTO1040
1160 FOR K=0 TO Z
1170 PRINT W$(A(K))" ";
1180 NEXT K
```

Listing 3: Alphabetisches Ordnen von Begriffen

Zeilen 1000 bis 1060

Die Eingabe der Stichworte wird organisiert. Das Ende des Eingabevorgangs wird durch ein Sternchen (CHR$ (42)) angezeigt.

Zeilen 1070 bis 1100

Hier werden das erste Zeichen der Zeichenkette des eingegebenen Wortes abgetrennt, der

dazugehörige ASCII-Wert ermittelt und die in der Zeichenkette nachfolgenden weiteren Buchstaben analog untersucht.

Zeilen 1110 bis 1140

Die vorliegenden Felder werden einer Prüfung unterzogen und die Elemente des Indexfeldes in der Größe verglichen. Falls erforderlich, werden die Feldelemente umgestellt. Dieses Sortierverfahren ermöglicht eine erhebliche zeitliche Straffung.

Zeilen 1150 bis 1180

Die alphabetische Ausgabe erfolgt auf der Grundlage der festgelegten Reihenfolge der Variablen des Indexfeldes A(Z), für die jetzt die zugehörige Zeichenkette des Wortfeldes W(Z) abgerufen wird.

Unterprogramm: Runden einer Zahl (274 Bytes)

Der BASIC-Interpreter des Z 1013 ist vorzüglich zum Berechnen von einfachen und komplizierten mathematischen Ausdrücken geeignet. Ein Nachteil besteht darin, daß die Rechengenauigkeit auf sechs Stellen festgelegt wird. Die siebente oder jede weitere Stelle einer mathematischen Konstante werden gerundet. Damit müssen Ungenauigkeiten in Kauf genommen werden. Diese "Eigenheit" weisen auch andere Computertypen auf. Für uns ist es jedoch oftmals wichtiger, bei einem errechneten Ergebnis auf viel weniger Stellen hinter dem Komma runden zu können. Es genügt für die meisten Anwendungen im Alltagsgebrauch festzustellen, daß eine Wegstrecke 23,4 km lang ist. Exakt würde der Z 1013 vielleicht den Wert 23,4325 km ausweisen. Unser Unterprogramm hilft, in Programme zur Berechnung mathematischer Aufgaben eine Routine zum Runden des Ergebnisses einzubauen (Listing 4).

```
1000 REM UNTERPROGRAMM ZUM RUNDEN
1010 CLS:PRINT"AUF WIEVIEL STELLEN NACH DEM"
1020 PRINT:PRINT"KOMMA SOLL GERUNDET WERDEN ?":PRINT
1030 INPUT KS
1040 PRINT:PRINT:INPUT"EINGABE DER ZAHL:";Z:PRINT:PRINT
1050 Z$=STR$(INT(Z*10^KS+0.5))
1060 L=LEN(Z$)
1070 E$=LEFT$(Z$,L-KS)+"."+RIGHT$(Z$,KS)
1080 E=VAL(E$)
1090 PRINT:PRINT:PRINT"ERGEBNIS:";E
```

Listing 4: Runden einer Zahl

Zeilen 1010 bis 1030

Zunächst legen wir den Faktor KS selbst fest. Dieser Faktor entspricht der Anzahl von Stellen nach dem Komma.

Zeile 1040

Die zu bearbeitende Zahl ist einzugeben. Es versteht sich, daß bei der Nutzung des Unterprogramms in einem Hauptprogramm der Zahlenwert nicht gesondert eingegeben zu

werden braucht, weil er z. B. durch eine LET-Anweisung bereitgestellt werden kann. Diese Zeile wäre entsprechend zu ändern.

Zeile 1050

Die eingegebene Zahl Z wird in eine Zeichenkette Z$ verwandelt. Zuvor erfolgt jedoch bereits ein Rundungsprozeß. Die Zahl wird mit einem Faktor 10^{KS} multipliziert. Die Addition von 0,5 organisiert das Auf- bzw. Abrunden nach den üblichen Regeln. Durch die Anwendung der INTEGER-Funktion wird dann alles abgetrennt, was hinter den gewünschten Kommastellen noch an Ziffern steht.

Zeile 1060

Die Anzahl der Elemente der Zeichenkette Z$ wird ermittelt.

Zeile 1070

Mit Hilfe von Stringfunktionen bilden wir aus der Zeichenkette Z$ die Zeichenkette des Ergebnisses E$. Wir trennen die Zeichenkette, indem wir den Wert KS rechts von der Länge L der Zeichenkette abtrennen und ein Komma dazwischen setzen.

Zeile 1080

Mit der Zeichenkette E$ haben wir scheinbar schon das gewünschte Ergebnis. Die Zeichenkette könnten wir jedoch nicht verwenden, wenn weiter damit gerechnet werden soll. Deshalb wird die Zeichenkette E$ zum Schluß wieder in eine numerische Konstante E verwandelt. Dazu wird die VAL-Funktion genutzt.

Unterprogramm: Formatierte Ausgabe von Zahlen (238 Bytes)

Der BASIC-Interpreter bewirkt mit der PRINT-Anweisung immer eine linksbündige Ausgabe auf dem Bildschirm. Der Computer kann nicht so ohne weiteres erkennen, daß wir Zahlen lieber nach Dezimalstellen geordnet untereinander stehen haben wollen. Bei der Wiedergabe von drei Zahlen entsteht z. B. folgendes Bild:

```
1.23
125
41.1
```

Für eine bessere Überschaubarkeit und Weiterverarbeitung wünschten wir jedoch die Zahlen so:

```
  1.23
125
 41.1
```

Dazu verhilft uns das folgende Unterprogramm, das sich einzeln, aber auch kombiniert mit dem UP "Runden einer Zahl" gebrauchen läßt (Listing 5).

```
1000 REM FORMATIERTE AUSGABE UND SUMMENBILDUNG
1010 DIM A$(100):Z=0
1020 Z=Z+1
1030 INPUT A$(Z)
1040 IF A$(Z)="*" THEN 1050:ELSE 1020
1050 CLS:FOR N=1 TO Z-1
1060 L=LEN(A$(N)):A(N)=VAL(A$(N))
1070 PRINTTAB(15-L)A(N)
1080 A=A+A(N):NEXT N
1090 A$=STR$(A)=LEN(A$)-1
1100 PRINT"SUMME :";TAB(15-L)A
```

Listing 5: Formatierte Ausgabe von Zahlen

Zeilen 1010 bis 1040

Ein Speicherbereich für 100 Zeichenketten wird reserviert. Dieser Wert läßt sich beliebig verändern und wird nur durch die Anzahl der formatiert auszugebenden Zahlen bestimmt. Den Zähler Z für die Anzahl der einzugebenden Zahlen setzt das Programm auf Null. Die Eingabe der jeweiligen Zahl wird gleich als String aufgefaßt. Das spart nachträgliches Umwandeln. Mit dem Sternchenzeichen ist die Eingabe abzuschließen.

Zeilen 1050 bis 1100

Für jede eingegebene Zeichenkette wird die Länge L bestimmt. Danach erfolgt die Umwandlung in einen numerischen Wert. Die PRINT TAB-Anweisung organisiert die rechtsbündige Wiedergabe jeder Zahl, indem vom Tabulatorstellenwert die Länge der Zeichenkette subtrahiert wird. Um das Unterprogramm zu komplettieren, werden die eingegebenen Werte addiert und das Ergebnis ebenfalls formatiert ausgedruckt. Statt der Addition kann man nach Bedarf auch andere mathematische Operationen vornehmen.

Unterprogramm: Umwandlung einer Dezimalzahl in eine Hexadezimalzahl (186 Bytes)

Für viele Anwendungen interner Funktionen eines Computers werden Zahlenwerte nicht im Dezimalsystem, sondern im Hexadezimalsystem benötigt. Das versetzt den Nutzer in die Verlegenheit, Zahlen in ein anderes System umwandeln zu müssen. Beispielsweise sind die Startadressen eines Maschinenprogramms im Hexadezimalsystem angegeben. Zum Einladen des BASIC-Interpreters schreiben wir stets L 100 2AFF. Im Alltagsgebrauch nutzen wir immer das Dezimalsystem. Die Basiszahl ist 10, und darauf bezieht sich alles. Die Basiszahl des Hexadezimalsystems ist 16. Beim Umwandeln müssen wir also genau berechnen, welche Zahl in einem Zahlensystem der im anderen entspricht. Die folgende Routine hilft dabei. Zu beachten ist, daß im Hexadezimalzahlbereich die Ziffern 0 bis 9 nicht ausreichen und deshalb die Buchstaben A bis F hinzugezogen werden müssen (Listing 6).

```
1000 REM UP DEZIMAL-HEXADEZIMAL
1010 HX$=""
1020 INPUT"DEZIMALZAHL:";D
1030 R=D-INT(D/16)*16
1040 IF R<10 THEN HX$=CHR$(48+R)+HX$:ELSE HX$=CHR$
     (55+R)+HX$
1050 IF D>15 THEN D=INT(D/16):GOTO 1030
1060 PRINT"HEXADEZIMALZAHL:";HX$
```

Listing 6: Umwandlung Dezimalzahl in Hexadezimalzahl

Zeilen 1000 bis 1030

Die Hexadezimalkette wird für den Anfang auf eine inhaltslose Kette gesetzt. Es erfolgt die Eingabe der zu berechnenden Zahl. Die Bildung einer Hilfszahl R ist nötig, um nachfolgend mit dem ASCII-Code die Bildung einer Zeichenkette aus Ziffern und Buchstaben bewirken zu können.

Zeilen 1040 bis 1060

Durch die Kombination von Bedingungen in bezug auf die Hilfszahl R wird über die CHR$-Funktion aus dem ASCII-Code der entsprechende Hexadezimalwert zwischen 0 und 9 bzw. zwischen A und F ausgewählt. Falls die eingegebene Dezimalzahl größer als 15 ist, wiederholt sich dieser Vorgang unter Verringerung des Wertes von D solange, bis D einen Wert angenommen hat, der kleiner als 15 ist. Damit ist der Berechnungsvorgang beendet, und die Hexadezimalzahl wird gedruckt.

Unterprogramm: Umwandlung einer Hexadezimalzahl in eine Dezimalzahl (204 Bytes)

Dieses Programm liefert die Umkehrung der vorangegangenen Beschreibung. Jetzt wird aus der Zeichenkette einer Hexadezimalzahl über die Werte des ASCII-Codes für Ziffern und die Buchstaben A bis F der numerische Wert der Dezimalzahl ermittelt (Listing 7).

```
1000 REM UP HEX-DEZ
1010 D=0
1020 INPUT"HEXADEZIMALZAHL:";HX$
1030 L=LEN(HX$)
1040 FOR N=1 TO L
1050 W=ASC(MID$(HX$,N,1))
1060 IF W<58 THEN W=W-48:ELSE W=W-55
1070 D=W + D*16
1080 NEXT N
1090 PRINT"DEZIMALZAHL:";D
```

Listing 7: Umwandlung Hexadezimalzahl in Dezimalzahl

Der Rechenmeister Z 1013

Programm: Zinseszinsberechnung (371 Bytes)

Aller Anfang ist schwer - wir fangen aber mit etwas Leichtem an. Bei unserer einfachen Zinseszinsberechnung wird von einer einmaligen Einzahlung ausgegangen. Im Programm wird das Guthaben bei jährlicher Verzinsung mit einem Zinssatz von 3,25 % berechnet. Der BASIC-Interpreter des Z 1013 rechnet ebenso wie die Interpreter der Computer KC 85/3 und KC 87 mit einer Genauigkeit von 7 Stellen, wovon 6 Stellen auf dem Bildschirm angezeigt werden. Die 7. Stelle fungiert als Schutz- und Rundungsstelle. Wer die Absicht hat, Millionär zu werden, der bekommt bei dieser Stellengenauigkeit schon Schwierigkeiten mit den Pfennigen, denn bei mehr als sechs Stellen wird auf das Gleitkommaformat mit Angabe eines Exponenten umgeschaltet. Hier arbeiten wissenschaftliche Taschenrechner mit der Anzeige von zehn Stellen oder der Schultaschenrechner SR 1 mit acht Stellen genauer. Das muß vor allem bei ökonomischen Berechnungen beachtet werden, denn dort muß ja alles "bis auf den letzten Pfennig" stimmen. Komfortablere BASIC-Interpreter bieten deshalb die Möglichkeit, auch mit einer 14stelligen Genauigkeit zu rechnen.

Das vorliegende Programm berücksichtigt noch nicht den Fall, daß jährlich eine weitere Einzahlung hinzukommt. Dies wäre z. B. für einen Vergleich zwischen regelmäßiger Einzahlung auf ein Sparkonto und für eine Versicherung von Interesse. Nach dieser "Umbauidee" nun zum vorliegenden Programm (Listing 8).

```
10 REM ZINSESZINSBERECHNUNG
20 WINDOW:CLS
30 LET ZS=3.25:REM ZINSSATZ
40 INPUT"EINZAHLUNG IN M=";EZ
50 INPUT"EINZAHLUNGSJAHR=";EJ
60 INPUT"AUSZAHLUNGSJAHR=";AJ
70 IF AJ<=EJ THEN PRINT"EINGABEFEHLER":PAUSE 10:GOTO 50
80 LET GH=EZ*(1+ZS/100)^(AJ-EJ)
90 LET GH=INT(GH*100+.5)/100
100 PRINT"DAS GUTHABEN"
110 PRINT"BETRAEGT        =";GH;"M":PRINT
120 INPUT"WEITERE BERECHNUNGEN (J/N)?";A$
130 IF A$="J" THEN PRINT:GOTO 40
140 CLS:END
```

Listing 8: Zinseszinsberechnung

Zeilen 10 bis 30

Nach der Bezeichnung des Programms wird das Standardfenster eingestellt und der Bildschirm gelöscht. Diese "Vorarbeiten" werden wir zukünftig nicht gesondert erwähnen. In Zeile 30 wird der Zinssatz (Variable ZS) zu 3,25 % festgelegt. Durch die interpretative Arbeitsweise ist es leicht möglich, diesen Zinssatz auf einen "Traumwert" zu erhöhen. Natürlich kann der Zinssatz auch über eine INPUT-Anweisung eingegeben werden.

Zeilen 40 bis 70

Hier werden die erforderlichen Eingaben realisiert. Wenn das Auszahlungsjahr nicht größer als das Einzahlungsjahr ist, wird durch Zeile 70 eine Berechnung verhindert.

Zeilen 10 bis 30

In Zeile 80 steht die Gleichung zur Berechnung der Zinseszinsen. In der Folgezeile wird das Ergebnis auf zwei Stellen nach dem Komma gerundet. Diesen Rundungsmechanismus werden wir auch in einigen anderen Programmen verwenden. Anschließend wird das gerundete Ergebnis auf dem Bildschirm angezeigt.

Zeilen 120 bis 140

Auch dieses Abfrageprinzip nach weiteren Berechnungen wird in einigen anderen Programmen Verwendung finden. In Zeile 120 erwartet der Computer die Eingabe des Buchstaben J oder N. Wird mit J geantwortet, dann erfolgt nach Ausdruck einer Leerzeile auf dem Bildschirm der Rücksprung in die Programmzeile 40, wo neue Eingaben werte erwartet werden. In allen anderen Fällen wird mit Zeile 140 der Bildschirm gelöscht und die Programmbearbeitung beendet. Diese Anweisungen hätten auch als ELSE-Zweig in Zeile 130 eingefügt werden können.
Unser einfaches Programm kann noch verschiedenartig erweitert werden. Wie wäre es z. B. mit einer tabellarischen Darstellung, bei der sich das Auszahlungsjahr schrittweise erhöht? Möglich wäre auch die Berechnung der Zinseszinsen pro Tag, so wie es beim Spar-Giro-Konto auf der Sparkasse geschieht. Aber das wird dann schon ein eigenständiges Programm.

Programm: Körner auf dem Schachbrett (318 Bytes)

Der Historiker al-ja'qubi berichtet, daß sich der Erfinder des Schachspieles von der Tochter seines Königs als Geschenk die Menge aller Weizenkörner erbat, die sich ergibt, wenn man auf das erste Feld ein Korn, auf das zweite zwei, auf das dritte vier und so fort legt.

```
Feld-  Anzahl der      Gesamtsumme
Nr.    Koerner/Feld    der Koerner
     1               1               1
     2               2               3
     3               4               7
     4               8              15
     5              16              31
     6              32              63
     7              64             127
     8             128             255
     9             256             511
    10             512            1023
    11            1024            2047
    12            2048            4095
    13            4096            8191
    14         8192.01           16383
    15           16384           32767
    16           32768           65535
    17           65536          131071
    18          131072          262143
    19          262144          524287
    20          524288        1.04858E+06
    21      1.04858E+06       2.09715E+06
    22      2.09715E+06        4.1943E+06
    23       4.1943E+06       8.38861E+06
    24      8.38861E+06       1.67772E+07
    25      1.67772E+07       3.35544E+07
    26      3.35544E+07        6.7109E+07
    27       6.7109E+07       1.34218E+08
                             <ENTER>
```

Bild 11: Anzahl der Körner auf dem Schachbrett (Ausschnitt)

Aber die Getreidevorräte des Landes reichten nicht aus, um diesen Wunsch zu erfüllen. Auch die Umrechnung des Getreides in Gold führte zur Erschöpfung aller Schätze des Landes. Der Erfinder verzichtete schließlich auf diesen Wunsch. Die Königstochter fragte ihn nach der Zahl der Körner, die er insgesamt mit den 64 Feldern des Schachbrettes verlangt hatte. Wir wollen diese Frage mit dem folgenden kleinen Programm beantworten .

Das Programm zeigt, daß der Z 1013 auch mit Kleinbuchstaben umgehen kann. Es zeigt allerdings auch, daß bei der Potenzfunktion (Zeichen ^ auf der Tastatur) unerwartete Rechenungenauigkeiten auftreten können. Bild 11 zeigt das Rechenergebnis für die ersten 27 Felder, wobei im Feld Nr. 14 eine Ungenauigkeit von 1/100 auftritt, denn 2^{13} = 8192. Das Bild macht auch das automatische Umschalten des Interpreters auf das Gleitkommaformat mit Angabe eines Exponenten deutlich. Weiterhin zeigt das Bild, daß die Gesamtsumme der Körner bei der Betrachtung von n Feldern gleich der Summe aus der Anzahl der Körner auf Feld n und der Gesamtsumme der Körner bei n-1 Feldern ist. Unter diesem Gesichtspunkt hätte die Spalte mit der Anzahl der Körner pro Feld weggelassen werden können. Aber beim Vergleich beider Spalten kann man sehen, wie dem Interpreter die großen Zahlen "zu schaffen" machen. Bei 64 Feldern ergibt sich eine Gesamtsumme von rund 10^{19} Körnern. Wieviel LKW-Ladungen mögen das wohl sein? Wir wollen das hier nicht ausrechnen, sondern das Programm in Listing 9 betrachten.

```
10 REM KOERNER AUF SCHACHBRETT
20 WINDOW:CLS
30 LET S=0:DEF FN P(X)=LEN(STR$(X))
40 PRINT"Feld- Anzahl der   Gesamtsumme"
50 PRINT" Nr.  Koerner/Feld der Koerner"
60 WINDOW 3,31,0,31
70 FOR I=1 TO 64
80 LET KF=2^(I-1) :LET S=S+KF
90 IF PEEK(113)>30 THEN PRINT:PRINT TAB(22);"<ENTER>";:
   INPUT"";A$:CLS
100 PRINT TAB(3-FN P(I));I;TAB(17-FN P(KF));KF;TAB
    (30-FN P(S));S
110 NEXT I
120 END
```

Listing 9: Körner auf dem Schachbrett

Zeilen 10 bis 50

Der Summenzähler S wird zur Sicherheit in Zeile 30 auf Null gesetzt. Der Programmstart mit dem Kommando RUN löscht zwar alle Variablen, aber sicher ist sicher, denn bei einem unvorhergesehenen Start mit GOTO bleibt der Inhalt der Variablen erhalten. Die selbst definierte Funktion P(X) organisiert die Rechtsbündigkeit der Ausgabe. Eine ähnliche Funktion zur kommabündigen Darstellung werden wir auch in anderen Programmen verwenden. Das geschieht in Ermangelung der PRINT USING-Anweisung, über die auch die Computer der KC-Serie nicht verfügen. Mit den Zeilen 40 bis 50 wird der Tabellenkopf auf den Bildschirm gebracht.

Zeilen 10 bis 30

Mit der WINDOW-Anweisung in Zeile 60 wird dafür gesorgt, daß beim Weiterrollen des Bildschirmes der Tabellenkopf unversehrt bleibt. In der Laufanweisung von Zeile 70 bis 110 erfolgt die Berechnung der Anzahl der Körner pro Feld (Variable KF) und der aktuellen

Gesamtzahl der Körner (Variable S). Da der Z 1013, ebenso wie die Computer der KC-Serie, bei vollgeschriebenem Bildschirm nicht anhält, sondern den Bildschirm nach oben abrollt, müssen wir bei vollem Bildschirm für einen Computerstop sorgen. Das geschieht durch Abfrage der aktuellen Kursorposition, genauer der Zeile, in der der Kursor sich befindet. Hierzu bietet z. B. der KC 85/3 die Anweisung CSRLIN(n). Der KC 87 und der Z 1013 haben diese Anweisung nicht. Beim Z 1013 ist die aktuelle Kursorzeile in der Adresse 113 gespeichert, die mit der PEEK-Anweisung abgefragt wird. Wenn diese Zahl größer als 30 ist, dann erscheint auf dem Bildschirm das Wort ENTER, und der Computer hält an (Programmzeile 90). Dieses Anhalten wird durch die INPUT-Anweisung "erzwungen". Nach Drücken der ENTER-Taste wird der Bildschirm gelöscht und anschließend neu beschrieben. Das erfolgt in der Zeile 100, wobei durch den Aufruf der selbst definierten Funktion P(X) für eine rechtsbündige Darstellung gesorgt wird. Ein Schönheitsfehler des Programms besteht darin, daß nach Beendigung der Arbeit das Bildschirmfenster nicht wieder "voll aufgemacht" wird. Wie wäre dies wohl zu ändern?

Programm: Bremswegberechnung (584 Bytes)

Das folgende Programm könnte die Grundlage einer Bremswegtabelle für die Verkehrsunfallbereitschaft bilden. Hier werden nämlich für bestimmte Geschwindigkeitsbereiche die Bremswege bei trockener, nasser und vereister Straße ermittelt. Dabei wird mit folgenden Bremsverzögerungen gerechnet: Trockene Straße a = 6 m/s^2; nasse Straße a = 4 m/s^2; vereiste Straße a = 2 m/s^2. Diese Werte können in den entsprechenden Gleichungen natürlich geändert werden (Listing 10).

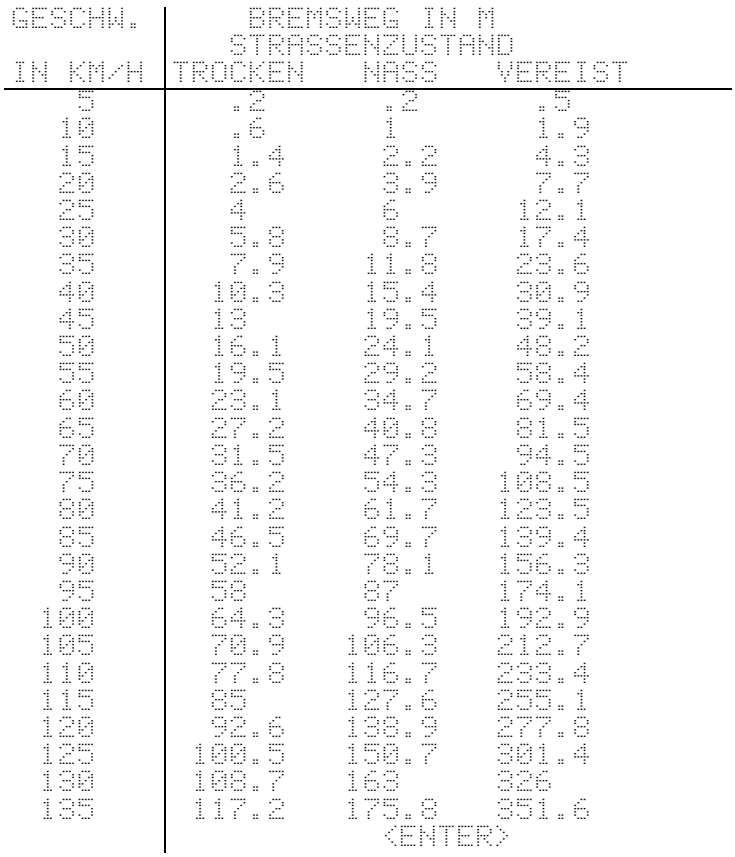
```
10 REM BREMSWEGTABELLE
20 WINDOW:CLS
30 DEF FN R(X)=INT(X*10+.5)/10
40 DEF FN P(X)=LEN(STR$(INT(X)))
50 REM TABELLENKOPF
60 PRINT"GESCHW.";CHR$(161);"    BREMSWEG IN M"
70 PRINT TAB(7);CHR$(161);"   STRASSENZUSTAND"
80 PRINT"IN KM/H";CHR$(161);"TROCKEN   NASS   VEREIST";
90 PRINT STRING$(32,CHR$(160))
100 REM BERECHNUNG UND DARSTELLUNG
110 WINDOW 4,31,0,31
120 FOR V=5 TO 180 STEP 5
130 LET FA=(V/3.6)^2
140 LET ST=FA/(2*6):LET ST=FN R(ST)
150 LET SN=FA/(2*4):LET SN=FN R(SN)
160 LET SV=FA/(2*2):LET SV=FN R(SV)
170 IF PEEK(113)>31 THEN PRINT TAB(22);"<ENTER>";:
    INPUT"";A$:CLS
180 PRINT TAB(4-FN P(V));V;TAB(7);CHR$(161);TAB(12-FN P
    (ST));ST;
190 PRINT TAB(20-FN P(SN));SN;TAB(28-FN P(SV));SV
200 NEXT V
210 END
```

Listing 10: Bremswegberechnung

Zeilen 10 bis 40

Die in der Zeile 30 selbstdefinierte Funktion R(X) realisiert das Runden einer Zahl auf eine Stelle nach dem Komma. Da der Bremsweg in Metern angegeben wird, ist dies ein für die

Praxis vernünftiger Wert. Die in der Folgezeile definierte Funktion P(X) realisiert im Zusammenhang mit der TAB-Anweisung eine kommabündige Darstellung der Ergebnisse. Schwierigkeiten ergeben sich bei Zahlen wie 0.2, da der Computer die vorlaufende Null nicht anzeigt. Diesen Schönheitsfehler zeigt Bild 12 beim Ausdruck der Ergebnisse für 5 und 10 km/h.



GESCHW.	BREMSWEG IN M		
	STRASSENZUSTAND		
IN KM/H	TROCKEN	NASS	VEREIST
5	.2	.2	.5
10	.6	1	1.9
15	1.4	2.2	4.3
20	2.6	3.9	7.7
25	4	6	12.1
30	5.8	8.7	17.4
35	7.9	11.8	23.6
40	10.3	15.4	30.9
45	13	19.5	39.1
50	16.1	24.1	48.2
55	19.5	29.2	58.4
60	23.1	34.7	69.4
65	27.2	40.8	81.5
70	31.5	47.3	94.5
75	36.2	54.3	108.5
80	41.2	61.7	123.5
85	46.5	69.7	139.4
90	52.1	78.1	156.3
95	58	87	174.1
100	64.3	96.5	192.9
105	70.9	106.3	212.7
110	77.8	116.7	233.4
115	85	127.6	255.1
120	92.6	138.9	277.8
125	100.5	150.7	301.4
130	108.7	163	326
135	117.2	175.8	351.6

<ENTER>

Bild 12: Bremswegtabelle (Ausschnitt)

Zeilen 10 bis 40

Für die Gestaltung des Tabellenkopfes werden die Pseudografikzeichen mit den Codezahlen 161 (senkrechter Strich) und 160 (waagerechter Strich) genutzt. Sie lassen sich durch Setzen eines Semikolons leicht in den auszugebenden Text einfügen. Allerdings ergeben sich bei der Benutzung der PRINT AT-Anweisung beim Z 1013 und beim KC 87 hierbei Schwierigkeiten. Sofern es sich nur um Zeichenketten handelt, kann in diesem Fall das Pluszeichen benutzt werden. Das Semikolon am Ende der Zeile 80 verhindert, daß der Kursor von der letzten Bildschirmspalte auf die nächste Zeile geht und damit eine unerwünschte Leerzeile erzeugt.

Zeilen 10 bis 40

Die WINDOW-Anweisung in Zeile 110 verhindert bei einem Rollen des Bildschirmes die Zerstörung des Tabellenkopfes. Mit der Laufanweisung in Zeile 120 kann nach Belieben experimentiert werden, indem -- und Endwert und die Schrittweite verändert werden. Der Faktor FA wird in jeder der drei folgenden Gleichungen benötigt. Hinter den Zahlen 6, 4 und 2 in den Programmzeilen 140 bis 160 verbergen sich die Bremsverzögerungen. In jeder Zeile wird der Wert nach seiner Berechnung auf eine Stelle nach dem Komma gerundet. Wie bei dem Körnerprogramm sorgt die Zeile 170 für ein vom Nutzer steuerbares Beschreiben des Bildschirmes. Die Zeilen 180 und 190 organisieren die Ausgabe auf dem

Bildschirm. Dabei wird auch der senkrechte Strich (CHR$(161)) "mitgeschleppt". Das Semikolon am Ende der Zeile 180 verhindert den Sprung des Kursors in die nächste Zeile. Es ist schon erstaunlich, was man bei angemessenem Runden der Ergebnisse alles auf den 32-Spalten-Bildschirm des Z 1013 bringen kann.

Programm: Kalenderdaten (2695 Bytes)

Das Programm verarbeitet Kalenderdaten vom 1. 1. 1583 an. Es basiert auf dem heute noch gültigen Gregorianischen Kalender, den Papst Gregor im Oktober 1582 eingeführt hat. Das Programm berechnet den Wochentag zu beliebigen Daten, womit sich alle Sonntagskinder dies nun auch vom Z 1013 bestätigen lassen können. Das Programm berechnet aber auch die Anzahl der Tage zwischen zwei Daten. Die Berechnung unter Einbeziehung der Schaltjahrregeln ist gar nicht so einfach. Schließlich wollen wir das Programm noch gegen Fehleingaben sicher machen. Es soll also z. B. keinen 29. 2. "annehmen", den es nicht gegeben hat oder nicht geben wird. Das macht unser BASIC-Programm schon recht umfangreich, wie Listing 11 zeigt.

```
10 REM WOCHENTAGE UND TAGE ZWISCHEN DATEN
20 DIM T(2):DIM M(2):DIM J(2):DIM FA(2):DIM WO$(6)
30 GOSUB 1000
40 WINDOW 0,31,0,31:CLS
50 PRINT"WAS MOECHTEN SIE BEARBEITEN?":PRINT:PRINT
60 PRINT"WOCHENTAG ZU EINEM DATUM=1":PRINT
70 PRINT"TAGE ZWISCHEN ZWEI DATEN=2":PRINT:PRINT
80 INPUT"BITTE KENNZAHL EINGEBEN:";KZ
90 IF KZ<>1 AND KZ<>2 THEN PRINT"EINGABEFEHLER":
   PAUSE 10:GOTO 40:ELSE CLS
100 IF KZ=1 THEN GOTO 300
110 REM BERECHNUNG DER TAGE ZWISCHEN ZWEI DATEN
120 GOSUB 2000
130 WINDOW 27,31,0,31:CLS
140 FOR I=1 TO 2
150 POKE 113,28:POKE 112,I*15-15:PRINT" ";I;". DATUM"
160 POKE 113,29:POKE 112,I*15-15:INPUT"  TAG  =";T(I):LET
    T(I)=INT(T(I))
170 POKE 113,30:POKE 112,I*15-15:INPUT"  MONAT=";M(I):LET
    M(I)=INT(M(I))
180 POKE 113,31:POKE 112,I*15-15:INPUT"  JAHR =";J(I):LET
    J(I)=INT(J(I))
190 GOSUB3000
200 IF FL=1 THEN PRINT"EINGABEFEHLER":PAUSE 10:GOTO 130
210 NEXT I
220 FOR I=1 TO 2:GOSUB4000:NEXT I
230 LET TA=ABS(FA(2)-FA(1))
240 IF Z>=24 THEN GOSUB 2000
250 WINDOW 2,26,0,31:CLS
260 PRINT"ZWISCHEN DEM";STR$(T(1));".";STR$(M(1));".";
    J(1);"UND"
270 PRINT STR$(T(2));".";STR$(  M(2));".";J(2);"LIEGEN";
    TA;"TAGE":PRINT
280 LET Z=Z+3:GOTO 900
300 REM ANGABE DES WOCHENTAGES
310 GOSUB 2000
320 WINDOW 27,31,0,31:CLS
330 LET I=1:INPUT"TAG  =";T(I):LET T(I)=INT(T(I))
340 INPUT"MONAT=";M(I):LET M(I)=INT(M(I))
350 INPUT"JAHR =";J(I):LET J(I)=INT(J(I))
360 GOSUB3000
370 IF FL=1 THEN PRINT"EINGABEFEHLER":PAUSE 10:GOTO 320
390 GOSUB4000
```

```
390 LET WO=FA(I)-INT(FA(I)/7)*7
400 IF Z=>26 THEN GOSUB 2000
410 WINDOW Z,26,0,31
420 PRINT STR$(T(I));".";STR$(M(I));".";J(I);"IST EIN ";WO$(WO)
    :PRINT
430 LET Z=Z+2 :GOTO 900
900 REM PROGRAMMFORTSETZUNG
910 WINDOW 27,31,0,31:CLS
920 PRINT"FORTSETZUNG DER BERECHNUNGEN=F"
930 PRINT"ZURUECK ZUM MENUE           =M"
940 PRINT"ABSCHLUSS DER BERECHNUNGEN  =E"
950 LET A$=INKEY$
960 IF A$="F" AND KZ=2 THEN GOTO 130
970 IF A$="F" AND KZ=1 THEN GOTO 320
980 IF A$="M" THEN GOTO 40
990 IF A$="E" THEN WINDOW 0,31,0,31:CLS:END:ELSE GOTO 950
1000 REM UP WOCHENTAGE
1010 FOR I=0 TO 6
1020 READ WO$(I)
1030 NEXT I:RETURN
1040 DATA SAMSTAG,SONNTAG,MONTAG,DIENSTAG
1050 DATA MITTWOCH,DONNERSTAG,FREITAG
2000 REM UP ANZEIGEFENSTER
2010 LET Z=0:WINDOW 0,26,0,31:CLS
2020 RETURN
3000 REM UP EINGABEKONTROLLE (FL=1 BEDEUTET FEHLER)
3010 IF J(I)<=1582 THEN GOTO3150
3020 IF M(I)<1 OR M(I)>12 THEN GOTO3150
3030 IF (M(I)=1 OR M(I)=3 OR M(I)=5) AND (T(I)<1 OR T(I)>31)
    THEN3150
3040 IF (M(I)=7 OR M(I)=8 OR M(I)=10) AND (T(I)<1 OR T(I)>31)
    THEN3150
3050 IF M(I)=12 AND (T(I)<1 OR T(I)>31) THEN GOTO3150
3060 IF (M(I)=4 OR M(I)=6 OR M(I)=9) AND (T(I)<1 OR T(I) >30)
    THEN3150
3070 IF M(I)=11 AND (T(I)<1 OR T(I)>30) THEN GOTO3150
3080 IF J(I)/100=INT(J(I)/100) THEN GOTO3120
3090 REM KEIN VOLLES JAHRHUNDERT
3100 IF M(I)=2 AND J(I)/4=INT(J(I)/4) AND T(I)>=1 AND T(I)<=29
    THEN3160
3110 IF M(I)=2 AND (T(I)<1 OR T(I)>28) THEN GOTO3150:ELSE GOTO
    3160
3120 REM VOLLES JAHRHUNDERT
3130 IF M(I)=2 AND J(I)/400=INT(J(I)/400)AND T(I))=1AND T(I)
    <=29 THEN3160
3140 IF M(I)=2 AND (T(I)<1 OR T(I)>28) THEN GOTO3150:ELSE GOTO
    3160
3150 LET FL=1:RETURN
3160 LET FL=0:RETURN
4000 REM UP FAKTORBERECHNUNG
4010 LET FA(I)=365*J(I)+T(I)+31*(M(I)-1))
4020 IF M(I)>2 THEN GOTO4050
4030 LET FA(I)=FA(I)+INT((J(I)-1)/4)-INT(.75*(INT((J(I)-1)/100)
    +1))
4040 RETURN
4050 FA(I)=FA(I)-INT(.4*M(I)+2.3)+INT(J(I)/4)-INT(.75*(INT(J(I)
    /100)+1))
4060 RETURN
```

Listing 11: Kalenderdaten

Zeilen 10 bis 100

Nach der Dimensionierung erforderlicher Felder wird im Unterprogramm ab Zeile 1000 das

Feld WO$ mit den Wochentagen "gefüllt". Die folgenden Zeilen bringen das Menü auf den Bildschirm, und die Auswahl wird über eine INPUT-Anweisung realisieren. Dabei werden falsche Eingaben (weder 1 noch 2) zurückgewiesen.

Zeilen 110 bis 280

Das Programm arbeitet mit einem Eingabe- und einem Anzeigefenster. Das erhöht den Programmieraufwand, liefert aber eine gute Bildschirmgestaltung. Da in diesem Programmteil zwei Daten zur Berechnung der Tage zwischen ihnen eingegeben werden müssen, ist die Eingabe in den Zeilen 140 bis 210 als Laufanweisung programmieren. AIIerdings muß hier mit POKE 113 die Zeilen- und mit POKE 112 die Spaltenposition des Kursors gesetzt werden. Eleganter geht das mit der den Wochentagen, über die aber der Z 1013 und der KC 87 nicht verfügen. Bei Eingabe eines falschen Datums wird im Unterprogramm ab Zeile 3000 das Fehlerflag FL = 1 gesetzt, das in Zeile 200 ausgewertet wird. In Zeile 220 werden durch Sprung in, das Unterprogramm ab Zeile 4000 die beiden Datenfaktoren ermittelt, deren Differenz die Anzahl der Tage TA bildet (Zeile 230). In Zeile 240 wird gefragt, ob das Anzeigefenster auf dem Bildschirm schon vollgeschrieben ist (Zeilenzähler Z). Falls nicht, wird durch schrittweise Verkleinerung dieses Anzeigefensters (Zeile 250) das Ergebnis auf dem Bildschirm ausgegeben.

Zeilen 300 bis 430

Dieser Programmteil ist etwas einfacher aufgebaut, da zu einem eingegebenen Datum nur der Wochentag zu berechnen ist. Der Index des Wochentages WO wird in Zeile 390 ermittelt, so daß sich in Zeile 420 mit WO$(WO) der korrekte Wochentag ergibt.

Zeilen 900 bis 990

In diesen Zeilen werden die möglichen Programmfortsetzungen über ein Menü organisiert. Die drei Fortsetzungsmöglichkeiten erscheinen im Eingabefenster. Die Abfrage erfolgt mit der INKEY$-Anweisung. Daraus ergeben sich gleich zwei Vorteile; das Drücken der ENTER-Taste wird gespart, und auf Fehleingaben erfolgt keine Reaktion. Soll die Programmbearbeitung abgeschlossen werden (Taste E), dann werden in Zeile 990 das Fenster wieder "voll aufgemacht" und der Bildschirm gelöscht.

Programm: 390 (2700 Bytes)

Von zwei parallelen Schulklassen sollen Reisewünsche ermittelt werden. Meer, Wald, Gebirge, Städtereise, Winterurlaub, Camping u. a. werden in Betracht gezogen. Jeder Schüler hat seine eigenen Vorstellungen. Und dennoch wiederholt sich vieles. Stimmen die Urlaubswünsche in beiden Klassen ungefähr überein? Das kann statistisch berechnet werden. Zu den Arbeitsmethoden der natura und gesellschaftswissenschaftlichen Forschung gehört die mathematisch begründete statistische Auswertung von Experimenten, Analysen und Befragungen. Häufig kann erst mit eindeutigen statistischen Berechnungen und Prüfverfahren der Beweis erbracht werden, daß gewonnene Erkenntnisse mit einer gewissen Wahrscheinlichkeit verallgemeinert werden können. Eine Vielzahl statistischer Methoden und Verfahren stehen zur Verfügung, von denen die jeweils zweckmäßigsten vom Bearbeiter ausgewählt werden müssen.
Solche Berechnungen sind mit einem hohen Rechenaufwand verbunden. Der Z 1013 löst

komplizierte Aufgaben sekundenschnell. Selbst mit dem Taschenrechner brauchte man dazu manchmal Stunden. Zu den bekanntesten statistischen Verfahren gehört das CHI-Quadrat-Verfahren (Listing 12). Auf dessen theoretische Grundlagen kann hier nicht eingegangen werden. Darüber sollte sich der interessierte Leser in der Fachliteratur informieren.

```
10 CLS:PRINT"X-QUADRATVERFAHREN":PRINT
20 PRINT"FUER ZWEI UNABHAENGIGE":PRINT"STICHPROBEN"
30 PRINT"(vgl.:CLAUSS/EBNER,1974,S.255)"
40 PRINT:PRINT:PRINT
50 PRINT"DAS PROGRAMM BERECHNET DEN":PRINT"X0-WERT"
60 PRINT"UND PRUEFT DIE SIGNIFIKANZ FUER"
70 PRINT"EINE STATISTISCHE SICHERHEIT":PRINT"VON 95%"
80 PRINT:PRINT:PRINT
90 PRINT"ES SIND BIS K = 30 MOTIVGRUPPEN"
100 PRINT"(KLASSEN) MOEGLICH"
110 PRINT AT(29,23);"> ENTER <"
120 IF INKEY$<>CHR$(13)THEN 120
200 REM ORDNUNGSSCHEMA
210 CLS:PRINT"ORDNE DIE AUSGANGSDATEN NACH"
220 PRINT"FOLGENDEM SCHEMA:"
230 PRINT:PRINT:PRINT
240 PRINT TAB(18) "HAEUFIGKEIT"
250 PRINT"MOTIVGRUPPE"
260 PRINT"(KLASSE)","PROBE 1","PROBE2"
270 PRINT"═══════════════════════════════"
280 PRINT"1","F11","F21"
290 PRINT"2","F12","F22"
300 PRINT"3","F13","F23"
310 PRINT:PRINT":",":",":":PRINT
320 PRINT"K","F1K","F2K"
330 PRINT"———————————————————————————"
340 PRINT"SUMME","N1","N2"
350 PRINT"———————————————————————————"
360 PRINT AT (29,23);"> ENTER <"
370 IF INKEY$(>CHR$(13) THEN 370
400 REM EINGABEANWEISUNGEN
410 CLS
420 PRINT"EINGABEANWEISUNG:"
440 PRINT:PRINT:PRINT
450 PRINT"S C H R E I B E":PRINT
460 PRINT"510 DATA F11,F21,F12,F22,F13,"
470 PRINT"          F23, ... ,F1K,F2K,-1,-1"
480 PRINT:PRINT:PRINT
490 PRINT AT(29,11);"STARTE MIT >RUN 500<"
495 END
500 REM BERECHNUNG X0
510 DATA 90,81,20,39,16,21,18,5,-1,-1
520 N1=0:N2=0:F1=0
530 READ A,B
540 IF ABO THEN 590
550 N1 = N1 + A
560 N2 = N2 + B
570 F1 = F1 + (A*A)/(A+B)
580 GOTO 530
590 N = N1 + N2
600 X0 =((N*N)/(N1*N2))* (F1-((N1*N1)/N))
605 CLS
```

```
610 PRINT"ANZAHL":PRINT"DER MOTIVGRUPPEN (KLASSEN)":PRINT:
    PRINT
620 INPUT K
700 REM VERGLEICH MIT TABELLE
710 DIM Z(30),T(30)
720 FOR F=1 TO 30
730 READ Z(F):NEXT F
740 FOR F =1 TO 30
750 READ T(F) :NEXT F
760 FOR F=1 TO 30
770 IF Z(F) =K-1 THEN 900
780 NEXT F
800 REM X0-TABELLE
810 DATA 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
820 DATA 16,17,18,19,20,21,22,23,24,25,26,27,28,29,30
830 DATA 3.84,5.99,7.81,9.49,11.1,12.6,14.1,15.5,16.9,
    18.3,19.7,21.0,22.4
840 DATA 23.7,25.0,26.3,27.6,28.9,30.1,31.4,32.7,33.9,
    35.2,36.4,37.7
850 DATA 38.9,40.1,41 .3,42.6,43.8
900 REM ERGEBNISAUSDRUCK
910 CLS:PRINT"E R G E B N I S A U S D R U C K"
920 PRINT:PRINT:PRINT
930 PRINT"UMFANG STICHPROBE 1: ";N1
940 PRINT"UMFANG STICHPROBE 2: ";N2
950 PRINT"GESAMTPOPULATION:    ";:
960 PRINT"MOTIVGRUPPEN:        ";K
970 PRINT:PRINT:PRINT
980 PRINT"BERECHNETER X0-WERT: ";    ;"
990 PRINT"X0-WERT NACH TABELLE:";T(F)
1000 PRINT:PRINT:PRINT
1010 IF X0>=T(F) THEN 1020 ELSE 1050
1020 PRINT"DA X0 >= XG-TABELLE:" :PRINT
1030 PRINT :PRINT"SIGNIFIKANNTER UNTERSCHIED"
1040 GOTO 1070
1050 PRINT"DA X0 < X0-TABELLE:":PRINT
1060 PRINT:PRINT"KEIN SIGNIFIKANNTER UNTERSCHIED"
1070 PRINT:PRINT:PRINT
1080 END
```

Listing 12: CHI-Quadrat-Verfahren

Zeilen 10 bis 120

Die Leistungen des statistischen Verfahrens mit dem vorliegenden Programm werden erläutert. Es ermittelt den CHI-Quadrat-Wert für zwei unabhängige Stichproben mit maximal 30 Merkmalsausprägungen (Motivgruppen), addiert die Stichprobenumfänge und bestimmt das Signifikanzniveau unter Zugrundelegung von 95 % statistischer Sicherheit.

Zeilen 200 bis 370

Der Z 1013 kann die gestellte Aufgabe nur dann zuverlässig lösen, wenn die erfaßten Zahlenwerte zuvor nach einem vorgegebenen Schema geordnet werden. Dieses Schema wird dargestellt.

Zeilen 400 bis 495

Die ermittelten Werte müssen zur statistischen Bearbeitung in einer exakten Reihenfolge in einer DATA-Anweisung abgelegt werden. Diese Eingabeanweisung wird erläutert, die Aufforderung zum Schreiben der Zeile 510 DATA . . . . stellt also die Aufforderung zu einem Eingriff in das Programm selbst dar. Deshalb wird der Programmdurchlauf in Zeile 495 zunächst beendet und später durch die Eingabe von RUN 500 fortgesetzt. Den Abschlußeintrag in die DATA-Anweisung bilden zweimal die Werte -1.

Zeilen 500 bis 600

Das CHI-Quadrat-Verfahren wird nach der Formel von BRANDT-SNEDECOR durchgeführt. Wenn die READ-Anweisung in Zeile 530 einen Wert liest, der kleiner als Null ist (-1), beendet Zeile 540 das Einlesen und erteilt den Auftrag zum Beginn des Rechenvorganges.

Zeilen 610 bis 620

Die Anzahl der Merkmalsausprägungen (Motivgruppen) des auszuwertenden Experiments ist mittels einer INPUT-Anweisung einzugeben.

Zeilen 700 bis 850

Aus einer Standardtabelle für CHI-Quadrat-Werte sucht der Computer den Vergleichswert für die entsprechende Anzahl von Motivgruppen und die geforderte statistische Sicherheit von 95 % heraus. Die Tabelle ist in den DATA-Anweisungen der Zeilen 800 bis 850 untergebracht.

Zeilen 910 bis 1080

Die Ergebnisse der Berechnungen werden ausgedruckt. Zur Kontrolle werden einige Zwischenwerte mit auf den Bildschirm gebracht. In den Zeilen 1000 bis 1080 wird ermittelt, ob sich die beiden Stichproben signifikant unterscheiden.

Der wortgewaltige Z 1013

Programm: Bibliothek (4722 Bytes)

So richtig "wortgewaltig" wird unser Z 1013 erst, wenn durch zusätzliche RAM-Module eine Erhöhung seiner Speicherkapazität erfolgt. Für die Grundversion mit einem frei verfügbaren Speicher von 4846 Bytes lassen sich aber die Grundprinzipien von

Programmen, die mit Wort und Text umgehen, zeigen. Das Bibliotheksprogramm (Listing 13) "schafft" in der Grundversion des Z 1013 die Verwaltung von maximal 60 Büchern. Im Programm kann nach Verfasser, Titel oder Sachgebiet gesucht werden. Bei der Suche nach dem Sachgebiet erfolgt als zusätzlicher Service für die gefundenen Bücher eine alphabetische Sortierung nach dem Verfasser. Diese Sortierung benötigt allerdings Speicherplatz für den entsprechenden Programmteil und zwei zusätzliche Datenfelder. Bei Streichung dieses Programmteiles kann somit eine größere Anzahl von Büchern untergebracht werden. Wir wollten aber im Hinblick auf die zu erwartend den Erweiterungsbaugruppen für den Z 1013 dem Leser diesen Sortierzusatz nicht vorenthalten.

Bei näherer Programmbetrachtung wird klar, daß die Bücher in DATA-Zeilen abgelegt sind. Dies ist eine Möglichkeit zur Arbeit mit Datenbeständen. Hier ist zu beachten, daß die in DATA-Zeilen abgespeicherten Daten Bestandteil des BASIC-Programms sind und nicht ohne weiteres geändert werden können. Aus diesem Grund ist

```
5 REM BIBLIOTHEK
10 WINDOW:CLS:CLEAR 26
15 REM SACHGEBIETE FESTLEGEN
20 DIM A$(11):RESTORE
25 FOR I=1 TO 10
30 READ A$(I)
35 NEXT I
40 DATA NATURWISSENSCHAFTEN,GESELLSCHAFTSWISSENSCHAFTEN,
   KLASSIK
45 DATA BUERGERLICH-KRITISCHER REALISMUS,SOZIALISTISCHER R
   EALISMUS
50 DATA ROMAN,ABENTEUER,SCIENCE-FICTION,SPORT,MAERCHEN
55 REM ANZAHL DER BUECHER LADEN
60 RESTORE 1000:READ N
65 CLS:RESTORE1001
70 PRINT:PRINT "BIBLIOTHEK":PRINT STRING$(10,"="):PRINT
75 PRINT "WAS MOECHTEN SIE BEARBEITEN?":PRINT:PRINT
80 PRINT "1=SUCHE NACH VERFASSER":PRINT
85 PRINT "2=SUCHE NACH TITEL":PRINT
90 PRINT "3=SUCHE NACH SACHGEBIET":PRINT
95 PRINT:PRINT "BITTE KENNZAHL EINGEBEN!";:INPUT"";KE
100 ON KE GOTO150,215,280
105 PRINT"EINGABEFEHLER":PAUSE 20:GOTO65
110 REM SUCHE OHNE ERFOLG
115 PRINT AT(5,5);"SUCHE VERGEBLICH!"
120 POKE 113,31:POKE 112,22:INPUT" <ENTER>";E$:RUN
125 REM UP ZUORDNUNG DER SACHGEBIETE
130 FOR J=1 TO 10
135 IF Z$=LEFT$(A$(J),2) THEN RETURN
140 NEXT J
145 LET A$(J)="DATENFEHLER BEI SACHGEBIET":RETURN
150 REM SUCHE NACH VERFASSER
155 CLS:PRINT"GEBEN SIE DEN FAMILIENNAMEN":PRINT
160 PRINT"DES VERFASSERS EIN ! ":PRINT:INPUT"";V$:CLS:LET
    Z=0
165 FOR I=1 TO N
170 READ X$,Y$,Z$
175 IF V$<>X$ THEN GOTO200
180 GOSUB130
185 IF PEEK(113)>25 THEN GOSUB 900                    ~
190 PRINT X$;":":PRINT Y$;"/":PRINT A$(J):PRINT
195 LET Z=Z+1
200 NEXT I
205 IF Z=0 THEN GOTO115
210 POKE 113,31:POKE 112,22:INPUT" <ENTER>";E$:RUN
```

```
215 REM SUCHE NACH TITEL
220 CLS:PRINT"GEBEN SIE DEN HAUPTTITEL EIN !"
225 PRINT:INPUT"";T$:CLS:LET Z=0
230 FOR I=1 TO N
235 READ X$,Y$,Z$
240 IF INSTR(T$,Y$)=0 THEN GOTO265
245 GOSUB130
250 IF PEEK(113)>25 THEN GOSUB 900
255 PRINT X$;":":PRINT Y$;"/":PRINT A$(J):PRINT
260 LET Z=Z+1
265 NEXT I
270 IF Z=0 THEN GOTO115
275 POKE 113,31:POKE 112,22:INPUT" <ENTER>";E$:RUN
280 REM SUCHE NACH SACHGEBIET
285 REM VERZEICHNIS DER SACHGEBIETE
290 CLS:PRINT"FOLGENDE SACHGEBIETE SIND":PRINT"ERFASST:":
    PRINT:PRINT
295 PRINT"NA=";A$(1):PRINT:PRINT"GE=";A$(2):PRINT:
    PRINT"KL=";A$(3):PRINT
300 PRINT"BU=";A$(4):PRINT:PRINT"SO=";A$(5):PRINT:
    PRINT"RO=";A$(6):PRINT
305 PRINT"AB=";A$(7):PRINT:PRINT"SC=";A$(8):PRINT:
    PRINT"SP=";A$(9):PRINT
310 PRINT"MA=";A$(10):PRINT:PRINT
315 PRINT"GEBEN SIE DIE ERSTEN BEIDEN":PRINT"BUCHSTABEN D
    ES SACHGEBIETES"
320 PRINT"EIN !":PRINT:INPUT"";S$:CLS:LET Z=0
325 FOR I=1 TO N
330 READ X$,Y$, Z$
335 IF Z$=LEFT$(S$,2) THEN LET Z=Z+1
340 NEXT I
345 IF Z=0 THEN GOTO115
350 RESTORE1001:LET K=0
355 DIM U$(Z):DIMV$(Z)
360 FOR I=1 TO N
365 READ X$,Y$,Z$
370 IF Z$=LEFT$(S$,2) THEN GOSUB390
375 NEXT I
380 IF Z=1 THEN GOTO460:ELSE GOTO410
385 REM UP UMSPEICHERN
390 LET K=K+1
395 LET U$(K)=X$:LET V$(K)=Y$
400 RETURN
405 REM ALPHABETISCHE SORTIERUNG
410 FOR L=2 TO Z
415 IF U$(L-1)<=U$(L) THEN GOTO450
420 FOR M=1 TO L-1
425 IF U$(L-M)<=U$(L+1-M) THEN GOTO450
430 LET B$=U$(L+1-M):LET D$=V$(L+1-M)
435 LET U$(L+1-M)=U$(L-M):LET V$(L+1-M)=V$(L-M)
440 LET U$(L-M)=B$:LET V$(L-M)=D$
445 NEXT M
450 NEXT L
455 REM AUSDRUCK
460 FOR I=1 TO Z
465 IF PEEK(113)>25 THEN GOSUB 900
470 PRINT U$(I);":":PRINT V$(I):PRINT
475 NEXT I
480 POKE 113,31:POKE 112,22:INPUT" <ENTER)";E$:RUN
900 REM UP BILDSCHIRM LOESCHEN
910 POKE 113,31:POKE 112,0:PRINT" FORTSETZUNG DER ANZEI
    GE MIT"
```

```
915 INPUT"<ENTER>";E$
920 CLS:RETURN
1000 DATA 60:REM ANZAHL DER BUECHER
1001 DATA VERNE ,FUENF WOCHEN IM BALKAN,AB
1002 DATA ...,WISSENSSPEICHER CHEMIE,NA
1003 DATA ...,PHYSIK IN UEBERSICHTEN,NA
1004 DATA ...,MATHEMATIK IN UEBERSICHTEN,NA
1005 DATA LENIN,WAS TUN?,GE
1006 DATA MARX,KRITIK DES GOTHAER PROGRAMMS,GE
1007 DATA ENGELS,DER ANTEIL DER ARBEIT AN DER MENSCHWER
     DUNG DES AFFEN,GE
1008 DATA ...,PHILOSOPHISCHES WOERTERBUCH,GE
1009 DATA GOETHE,FAUST,KL
1010 DATA GOETHE,EGMONT,KL
1011 REM IN DEN ZEILEN 1011 BIS 1060 STEHEN WEITERE
     50 BUECHER
```

Listing 13: Hausbibliothek

solch eine Vorgehensweise für die Führung eines sich ständig ändernden Datenbestandes (z. B. Lagerwirtschaft) nicht sinnvoll. In diesen Fällen wird mit eigenständigen Dateien gearbeitet, die nicht zum BASIC-Programm gehören, aber über das Programm eingelesen werden können. Diesen Fall untersuchen wir beim folgenden Programm, bei dem es um Vokabeln geht.

Zeilen 5 bis 65

Hier werden einige Vorarbeiten getroffen. In Zeile 10 vermindern wir den Textspeicherbereich auf 26 Byte und gewinnen damit noch "kostbare" 230 Byte für das Programm. Hier gilt es zu beachten, daß der Textspeicherbereich im Programm nicht wieder auf 256 Byte (entspricht dem Standard nach Einschalten des Interpreters) heraufgesetzt wird. Auch das NEW-Kommando stellt den Bereich nicht zurück.

In den Programmzeilen 20 bis 50 werden die Sachgebiete der Textvariablen A$(1) über eine READ-DATA-Anweisung zugeordnet. Das ist allerdings nicht speicherplatzsparend, denn die Sachgebiete stehen nun sowohl im Programm als auch im Datenfeld A$(1).

In Zeile 60 wird von DATA-Zeile 1000 die Anzahl der Bücher, im vorliegenden Fall 60 Stück, gelesen. Mit RESTORE 1001 wird stets der DATA-Zeiger auf das erste abgespeicherte Buch (siehe Programmzeile 1001) gestellt.

Zeilen 70 bis 105

In diesen Programmzeilen erfolgt die Darstellung des Menüs auf dem Bildschirm, bei in Zeile 105 Fehleingaben "abgewehrt" werden. Die Fallauswahl in Zeile 100 gibt die Startadressen für die Suche nach Verfasser, nach Titel und nach Sachgebiet an.

Zeilen 110 bis 120

Wenn bei der Suche im Datenbestand kein Buch gefunden wurde, dann erfolgt die Abarbeitung dieses Programmteils. Der Neustart mit RUN ist vielleicht nicht ganz einsichtig, denn das wiederholte Einlesen der Sachgebiete und der Anzahl der Bücher könnte ja entfallen. Schwierigkeiten bereitet aber z. 8. eine mehrfache Vereinbarung der Felder U$(Z) und V$(Z) (Zeile 355), die bei Wiederstart mit der GOTO-Anweisung auftreten würden. Um solchen Problemen aus dem Wege zu gehen (weitere lauern im sehr knapp bemessenen Textspeicherbereich), erfolgt grundsätzlich jeder Wiederstart des Programms mit RUN. Sicherlich gibt es einen eleganteren Weg, ihn möge der Leser finden.

Zeilen 125 bis 145

Dieses kleine Unterprogramm unterstützt das Sparen von der Speicherplatz auf folgende Weise. In den DATA-Zeilen sind die Bücher mit Verfasser, Titel und Sachgebiet abgespeichert. Allerdings werden vom Sachgebiet nur die ersten beiden Buchstaben in den Speicher gebracht (siehe ab Zeile 1001). Immer dann, wenn eine Zuordnung der ersten beiden Buchstaben des Sachgebietes zum vollständigen Wort erforderlich ist (z. B. bei der Bildschirmausgabe), erfolgt eine Aufruf dieses Unterprogramms.

Zeilen 150 bis 210

Zur Suche nach dem Verfasser ist dessen Familienname einzugeben. Der Buchbestand wird mit einer FOR-NEXT-Anweisung "durchforstet". Dabei werden mit einer READ-Anweisung der Verfasser der Variablen X$, der Titel der Variablen Y$ und die ersten beiden Buchstaben des Sachgebiets der Variablen Z$ zugewiesen. Falls in Zeile 205 die Zählervariable Z noch den Wen 0 enthält, dann war die Suche vergeblich.

Zeilen 215 bis 275

Die Suche nach dem Titel erfolgt analog. Allerdings unterscheidet sich das Suchprinzip. Bei der Suche nach dem Verfasser wurde in Programmzeile 175 eine absolute Gleichheit gefordert. Bei der Titelsuche wird die INSTR-Anweisung verwendet (Zeile 240). Je nach Vollständigkeit der Eingabe des Titels wird hier mit entsprechender Weitschweifigkeit gesucht.

Zeilen 280 bis 480

Zunächst werden in einer Übersicht alle Sachgebiete auf dem Bildschirm angezeigt. In Zeile 320 erfolgt die Belegung der Variablen S$ mit den ersten beiden Buchstaben des gewünschten Sachgebietes. Der Suchlauf in den Zeilen 325 bis 340 dient nur der Bestimmung der Anzahl Z der zum Sachgebiet gehörenden Bücher. Mit dieser Variablen Z werden dann die Arbeitsfelder U$(Z) und V$(Z) dimensioniert. in der Laufanweisung von Zeile 360 bis 375 werden diese Felder dann mit Hilfe des Unterprogramms ab Zeile 385 gefüllt. Das alles kostet, ebenso wie das folgende alphabetische Sortieren, Zeit (vor allem bei Beständen von 500 und mehr Büchern). Nach der alphabetischen Fall nach dem Verfasser (Variable U$(Z)) erfolgt ab Zeile 460 die Ausgabe auf dem Bildschirm. Um ein unerwünschtes Abrollen des Bildschirminhalts zu verhindern, wird in Zeile 465 die Zeilenposition des Kursors abgefragt

und im Bedarfsfall das kleine Unterprogramm ab Zeile 900 aufgerufen. Auf diese Weise können in aller Ruhe die bibliographischen Angaben gelesen werden.

ab Zeile 1000

Die Zeile 1000 enthält die Anzahl der Bücher. Die Zahl ist also bei Erhöhung oder Verringerung des Bestandes zu aktualisieren. Der in DATA-Zeilen abgelegte Buchbestand beginnt ab Zeile 1001. Eine Fortführung in Einerschritten erleichtert die Übersicht über die Anzahl der aufgenommenen Bücher, Verfasser, Titel und die ersten beiden Buchstaben des Sachgebietes werden durch Kommas getrennt. Der Z. 1013 verlangt bei DATA-Texten keine Anführungszeichen. Das gilt auch für die Kleincomputer der KC-Serie, jedoch nicht für alle BASIC-Interpreter anderer Computer. Schwierigkeiten gibt es nur, wenn im Buchtitel selbst ein Komma vorkommt. Hier müssen dann Anführungsstriche gesetzt werden.

Richtigen Spaß macht die ganze Sache natürlich erst mit dem eigenen Buchbestand und mindestens 500 Büchern. Hier kann der erweiterte Z 1013 zeigen, was in ihm steckt.

Programm: Vokabeln (1874 Bytes)

Auch bei diesem Programm kann der Z 1013 seine Stärken so richtig zeigen, wenn zusätzliche RAM-Module die Kapazität des Schreib-Lese-Speichers erhöhen. Das Programm dient zum Anlegen eines individuellen Vokabelheftes, es ist also kein Lehrprogramm, sondern soll bei der Suche nach Vokabeln helfen. Leider hat unser Z 1013 keinen kyrillischen Zeichensatz. Es ist auch nicht möglich, eigene Grafikzeichen zu definieren, mit denen wir dann einen kyrillischen Zeichensatz hätten "bauen" können. Deshalb ist es leider auch nicht möglich, Sonderbuchstaben aus anderen Sprachen selbst zu generieren. Bei englischen Vokabeln gibt es keine Schwierigkeiten. Das Programm wurde so gestaltet, daß es bei der Suche gleichgültig ist, ob das deutsche oder das englische Wort eingegeben wird. Der Z 1013 sucht mit Hilfe der Funktion INSTR den gesamten Datenbestand durch und gibt bei erfolgreicher Suche das entsprechende englisch-deutsche Wortpaar aus. Wer "schärfer" fragen möchte (genau ein Wort in einer Sprache), müßte die Suchroutine entsprechend ändern. Listing 14 zeigt das vollständige Programm. Mit Hilfe dieses Programms ist zunächst die individuelle Vokabeldatei zu erstellen, die dann auf einer Magnetbandkassette ausgelagert und gesondert aufbewahrt werden kann.

```
10 REM VOKABELN ENGLISCH-DEUTSCH
20 WINDOW:CLS
30 CLEAR 1800:DIM A$(150,1)
40 PRINT "VOKABELN ENGLISCH-DEUTSCH":PRINT:PRINT
50 PRINT"1=VOKABELBESTAND EINLESEN":PRINT
60 PRINT"2=IM VOKABELBESTAND SUCHEN":PRINT
70 PRINT"3=VOKABELBESTAND ERGAENZEN":PRINT
80 PRINT"4=VOKABELN STREICHEN":PRINT
90 PRINT"5=VOKABELBESTAND ABSPEICHERN":PRINT
100 IN$=INKEY$
110 IF IN$=""THEN GOTO100
120 LET VZ=VAL(IN$)
130 LET IN$=""
140 LET I=0:CLS
150 ON VZ GOSUB 210,310,510,710,910
160 CLS:GOTO40
```

```
200 REM EINLESEN VON KASSETTE
210 PRINT"VOKABELDATEI  IN"
220 PRINT"KASSETTENGERAET EINLEGEN!";PRINT
230 INPUT"BEI PFEIFTON <ENTER>-TASTE";IN$
240 CLOAD*"VOKDAT";A$
250 PRINT:PRINT"KASSETTENGERAET AUSSCHALTEN!"
260 POKE 113,32:POKE 112,22:INPUT" <ENTER>";IN$
270 RETURN
300 REM SUCHEN IM BESTAND
310 LET IN$="":INPUT "SUCHWORT:";SW$:PRINT
320 LET ZE=0
330 IF LEN(A$(1,0))=0THEN INPUT"WEITERE SUCHE (J/N)?";IN$:
    ELSE GOTO350
340 IF IN$="J" THEN CLS:LET C=0:GOTO310:ELSE RETURN
350 IF LEN(SW$)=0 THEN GOTO370
360 IF INSTR(SW$,A$(I,0))+INSTR(SW$,A$(I,1))=0THEN GOTO380
370 PRINT A$(I,0);TAB(15);A$(I,1):PRINT:LET ZE=ZE+1
380 LET I=I+1
390 IF ZE<25 THEN GOTO330
400 POKE 113,31:POKE 112,22:INPUT" <ENTER>";IN$
410 GOTO320
500 REM BESTAND ERGAENZEN
510 IF LEN(A$(I,0))=0 THEN GOTO550
520 LET I=I+1
530 IF I>150 THEN RETURN
540 GOTO510
550 POKE 113,5:POKE 112,0:INPUT" ENGL.:";A$(I,0)
560 POKE 113,5:POKE 112,15:INPUT" DEUTSCH:";A$(I,1)
570 PRINT:PRINT:PRINT TAB(10);"KONTROLLE"
580 PRINT:PRINT A$(I,0);TAB(15);A$(I,1):PRINT
590 INPUT"WEITERE ERGAENZUNGEN (J/N)?";IN$
600 IF IN$="J" THEN CLS:GOTO510
610 RETURN
700 REM VOKABELN STREICHEN
710 INPUT"ZU STREICHENDE VOKABEL:";ST$:PRINT
720 IF LEN(A$(I,0))=0 THEN RETURN
730 IF LEN(ST$)=0 THEN GOTO770
740 IF INSTR(ST$,A$(I,0))<>0 OR INSTR(ST$,A$(I,1))<>0 THEN
    GOTO 770
750 LET I=I+1
760 GOTO720
770 PRINT A$(I,0);TAB(15);A$(I,1):PRINT
780 LET FL=0
790 INPUT"STREICHEN=1:";FL
800 IF FL<>1 THEN GOTO750
810 FOR J=I TO 149
820 LET A$(J,0)=A$(J+1,0)
830 LET A$(J,1)=A$(J+1,1)
840 NEXT J
850 LET A$(150,0)=""
860 LET A$(150,1)=""
870 GOTO720
900 REM ABSPEICHERN AUF KASSETTE
910 PRINT"BAND IN KASSETTENGERAET"
920 PRINT"EINLEGEN!"
930 PRINT"GERAET AUF AUFNAHME STELLEN"
940 INPUT"<ENTER>-TASTE";IN$
950 CSAVE*"VOKDAT";A$
960 PRINT:PRINT"KASSETTENGERAET AUSSCHALTEN!"
970 POKE 113,32:POKE 112,22:INPUT" <ENTER>";IN$
980 RETURN
```

Listing 14: Vokabelheft

Zeilen 10 bis 160

In Zeile 30 wird der Textspeicherbereich, der ja die Vokabeldatei aufnimmt, auf 1800 Bytes erhöht. Das ist das Maximum, was unser Z 1013 in seiner Grundversion ohne Speichererweiterung verkraften kann. Auch hier ist, wie im Programm Bibliothek, zu beachten, daß der Textspeicherbereich nicht auf den Standardwert von 256 Bytes zurückgesetzt wird. Bei Arbeitsabschluß ist nach Ausführung des Kommandos NEW deshalb CLEAR 256 einzugeben. Ist die Anwendung der CLEAR-Anweisung im Programm notwendig, so gehen damit auch alle vorhergehenden GOSUB-RETURN-Beziehungen verloren. Mit der Anweisung DIM A$(150,1) wird Platz für 150 Vokabeln geschaffen. Das englische Wort steht dann unter A$(I,0) und das deutsche Wort unter A$(I,1).

Die fünf Auswahlmöglichkeiten im Menü zeigen den Umgang mit dem Programm. Alle fünf Programmteile sind als Module in Unterprogrammtechnik ausgeführt und werden von Zeile 150 aus über ON VZ GOSUB . . . aufgerufen.

Zeilen 200 bis 270

Dieses Unterprogramm realisiert das Einlesen des Vokabelbestandes von einer Magnetbandkassette. Wenn der Bestand vorliegt, ist mit dem Einlesen der Datei *VOKDAT" zu beginnen. Bei der Arbeit mit Dateien müssen folgende Bedingungen beachtet werden:

- Der Textspeicherbereich muß gleich oder größer dem Textspeicherbereich bei der Herstellung der Datei sein
- Die Dimensionierungsanweisung DlM A$(150,1) muß stets, z. B. bei Nutzung der Datei in einem anderen Programm, beibehalten werden.
- Der Name der Datei und der Variablenname (also "VOKDAT"; A$) müssen ebenfalls beibehalten werden.

Zeilen 300 bis 410

Das Suchen im Bestand ist natürlich nur dann sinnvoll, wenn ein Datenbestand im Speicher vorhanden ist. In Zeile 310 wird das Suchwort eingegeben und der Variablen SW$ zugewiesen. Der Datenbestand wird solange durchsucht, bis in Zeile 330 bei LEN(A$(I,0)) = 0 das Ende des Bestandes signalisiert wird. Die Entscheidung über eine erfolgreiche Suche wird in Zeile 360 getroffen. Hier kann der Nutzer das Suchprinzip ändern. War die Suche erfolgreich, dann wird das Wortpaar in Zeile 370 auf dem Bildschirm ausgegeben. Was wird bei diesem Suchprinzip wohl geschehen, wenn als Suchwort z. B. nur ein E oder ein Leerzeichen eingegeben oder nur die ENTER-Taste gedrückt wird? Bleibt im letzten Fall der ursprüngliche Inhalt von SW$ erhalten?

Zeilen 500 bis 610

Zur Ergänzung des Datenbestandes wird in Zeile 510 zunächst das letzte belegte Datenfeld bestimmt, damit die Anfügung auch ordentlich funktioniert. Nach Eingabe des englisch-deutschen Wortpaares wird es zur Kontrolle noch einmal ausgegeben, um Schreibfehler

möglichst gering zu halten. Sollten keine weiteren Ergänzungen angefügt werden, wird zum Menü zurückgekehrt. Die vollständige "Eigensteuerung" des Programms ist für den Erhalt des Datenbestandes wichtig. Sollte. z. B. auf Grund unvorhergesehener Dinge das Programm mit STOP unterbrochen werden, darf nur mit GOTO (z. B. GOTO 40), nicht aber mit RUN wieder gestartet werden, denn RUN löscht den gesamten Datenspeicher und kann damit die zeitaufwendige Eintipparbeit zunichte machen.

Zeilen 700 bis 870

Die zu streichende Vokabel wird der Variablen ST$ zugewiesen, im Bestand gesucht und schließlich das zusammengehörende möglichst angezeigt (Zeile 770). Zur Sicherheit wird in Zeile 790 noch einmal gefragt, ob die Streichung tatsächlich durchgeführt werden soll. Falls ja, wird das "Streichungsflag" FL mit dem Wert 1 belegt. Die folgende Umsortierung in der Zählschleife von Zeile 810 bis 840 überschreibt das zu streichende Wortpaar durch ein "Vorrücken" des folgenden Datenbestandes. Allerdings braucht unser Z 1013 dafür schon ein wenig Zeit.

Zeilen 900 bis 980

Wir haben das "Gegenstück" zum Einlesen, nämlich das Abspeichern eines Vokabelbestandes auf einer Magnetbandkassette, vor uns. Auch hier müssen die schon genannten Bedingungen zur Arbeit mit Dateien beachtet werden. Mit diesem Unterprogramm zum Abspeichern kann nach Abschluß von Ergänzungs- oder Streichungsarbeiten im Vokabelbestand die aktuelle Datei auf den externen Datenträger Magnetbandkassette ausgelagert werden.

Programm: Stichwortverzeichnis (1980 Bytes)

In Bibliotheken gibt es neben dem Autorenverzeichnis und dem systematischen Katalog meist auch noch ein Stichwortverzeichnis (auch Schlagwortverzeichnis). Hat man keine konkreten Vorstellungen von bestimmten Buchtiteln oder Zeitschriftenartikeln, sondern nur vom inhaltlichen Begriff, um den es geht, hilft dieses Verzeichnis. Sucht man das Stichwort "BASIC", findet man einen ganz beachtlichen Nachweis zahlreicher Quellen. Man notiert die Signatur, den Titel und den Namen des Autoren. Damit läßt sich das Gewünschte in der Ausleihe bereitstellen. Für die private Aufarbeitung eines Stichwortverzeichnisses kann der Z 1013 ausgezeichnete Dienste leisten. Bei den Vorarbeiten muß man sich darüber klarwerden, unter welchen Stichworten die Quellen aufgeteilt werden sollen. Das vorliegende Beispiel ist nach eigenem Bedarf abänderbar und erweiterungsfähig. Die Stichworte beziehen sich auf Computerliteratur. Ihnen wurden Bücher und Zeitschriften zugeordnet, die sich in den letzten Jahren zum Thema geäußert haben (Listing 15).

```
10 CLS:REM STICHWORTVERZEICHNIS
20 PRINTAT(12,6);"STICHWORTVERZEICHNIS"
30 FOR N=4 TO 27:PRINTAT(10,N);CHR$(42):PRINTAT(14,N);
   CHR$(42):NEXT N
40 FOR N=11 TO 13:PRINTAT(N,4);CHR$(42):PRINTAT(N,27);
   CHR$(42):NEXT N
50 PRINTAT(0,0);CHR$(32)
60 PRINTAT(30,23);">ENTER<"
70 IF INKEY$<>CHR$(13) THEN 70
100 CLS:DIM S$(20)
110 REM LEISTUNGSANGEBOT
120 PRINT:PRINT:PRINT
130 PRINT"WELCHE LEISTUNG WIRD GEWUENSCHT?"
```

```
140 PRINT:PRINT:PRINT
150 PRINT"   NEUEINTRAGUNG              = 1"
160 PRINT:PRINT
170 PRINT"   STICHWORTE ANZEIGEN        = 2"
171 PRINT:PRINT
176 PRINT"   PROGRAMM BEENDEN           = 3"
180 PRINT:PRINT:PRINT
190 INPUT"WAS WIRD GEWAEHLT ?";Z
200 ON Z GOTO 300,400,650
300 CLS: LIST 1000
400 CLS:PRINT:PRINT"STICHWORTANGEBOT":PRINT"============
    ===":PRINT
410 RESTORE710:FOR N=1 TO 6:READ S$(N):PRINTS$(N);TAB(30)N
420 PRINT:NEXT N
430 PRINT:INPUT"ZAHL EINGEBEN";N
440 CLS:PRINT"STICHWORT: ";S~(N)
450 PRINT"————————————————————————————————"
500 REM SUCHVORGANG
510 RESTORE 1000:FOR K=1 TO 200:READ A$
520 B$=LEFT$(A$,1):C$=MID$(A$,3,1)
530 B=VAL(B$):C=VAL(C$)
540 IF B=N OR C=N THEN PRINTA$:PRINT
550 IF A$="XXX" THEN PRINTAT(30,0);"ENDE DER EINTRAGUNG":
    GOTO 600
560 NEXT K
600 PRINTAT(30,24);">ENTER<"
610 IF INKEY$<>CHR$(13) THEN 610
620 CLS:GOTO 110
650 CLS:END
700 REM STICHWORTANGEBOT
710 DATA PROGRAMMIERSPRACHEN
720 DATA BASIC
730 DATA COMPUTER/ALLGEMEIN
740 DATA SOFTWARE/ANREGUNGEN
750 DATA DDR-KLEINCOMPUTER
760 DATA Z 1013
1000 DATA 5/6  TECHNIKUS 7/86
1010 DATA 5  Z9001-BERUFSBILDUNG-7/8/84
1020 DATA 3  GUTZER-MIKROCOMPUTER 1985
1030 DATA 2/5 BUECKNER-KLEINCOMPUTER 1986
1040 DATA 3 FRANKE-MIKRORECHENTECHNIK 1986
1050 DATA 2/5 Z9001-HANDBUCH  1985
1060 DATA 5 Z9001-KLEINSTRECHNER-TIPS NR. 2
1070 DATA 1 PASCAL-NACHRICHTENTECHNIK 7/83
1080 DATA 1 PAULIN-PASCAL  1986
1090 DATA 1/3 ADLER-RECHENANLAGEN 1985
1100 DATA 1 KLEINSTRECHNER-TIPS NR.4
1110 DATA 1 WISSENSSPEICHER PROGR. 1986
1120 DATA 4 JUGEND+TECHNIK
1130 DATA 1/2 BOON-BASIC  1983
1140 DATA 4 TECHNIKUS
1150 DATA 1/2 MUELLER-BASIC 1986
1160 DATA 4 IMPULS 68
1170 DATA 1/2 STRELOCKE-BASIC 1981
1180 DATA 4 ALPHA
1190 DATA 1/2 MEGA-BIT (JU+TE) 1986
1200 DATA 4 RADIO/FERNSEHEN/ELEKTRONIK
1210 DATA 1/2 IMPULS 68 HEFT 1/86
1220 DATA 4 MIKROPROZESSORTECHNIK
```

```
1230 DATA 1/2 K85/2-HANDBUCH  1986
1240 DATA 4 RECHENTECHNIK/DATENVERARB.
1250 DATA 1/2 WISSENSSPEICHER BASIC 1986
1260 DATA 4 KLEINSTRECHNER-TIPS
1270 DATA 4 FUNKAMATEUR
2000 DATA XXX
```

Listing 15: Stichwortverzeichnis

Zeilen 100 bis 200

Zunächst dimensionieren wir ein Zeichenkettenfeld für die Stichworte. Wir haben für 20 mögliche Stichworte Speicherplatz reserviert, obwohl nur 6 Begriffe verwendet werden. Damit verfügen wir über eine Reserve für eine spätere eventuelle Erweiterung. Das Leistungsangebot unterbreitet die Möglichkeit des Eintragens neuer Quellen, das Suchen nach vorgegebenem Stichwort und die Beendigung des Programmes. ON . . .GOTO . . . bewirkt den Sprung in die richtige Zeile.

Zeile 300

Diese Zeile bewerkstelligt, daß die bereits eingetragenen Quellen in den DATA-Zeilen ab Zeilennummer 1000 aufgelistet werden. Für den Vergleich kann man in aufgelisteten Zehnerschritten mitlesen und in der ersten freien DATA-Zeile mit dem Neueintrag beginnen.

Zeilen 400 bis 450

Wir geben die DATA-Anweisung in Zeile 710 mit RESTORE frei. In der Schleife werden durch die READ-Anweisung die Zeichenketten der Stichworte gelesen und auf den Bildschirm gebracht. Jedem Stichwort wird darüber hinaus eine Ordnungszahl zu gewiesen, die für den weiteren Arbeitsablauf eine wichtige Rolle spielt. Mit der Auswahl des Stichwortes durch die Eingabe der zugehörigen Ordnungszahl N leitet das Programm den Suchvorgang ein.

Zeilen 500 bis 560

Eine neue Schleifenanweisung kann maximal 200 Durchläufe aufweisen. Diese Größe ist willkürlich gewählt. Spätestens nach dem 200. Durchlauf wird die Abbruchbedingung in Kraft gesetzt. in der Schleife liest die READ-Anweisung alle gespeicherten Quellen. Die Quellen werden ab Zeile 1000 abgelegt. Vor jeder Quelle sind ein oder zwei Ordnungsnummern als Bestandteil des Strings notiert. Diese Ordnungsnummern beziehen sich auf die Ordnungsnummer des Stichwortes. Jeder Quelle können (aber müssen nicht) zwei Stichworte zugeordnet sein. Die Stringfunktionen trennen die Suchnummern ab und bilden ihre numerischen Werte. Stimmt der gebildete numerische Wert mit dem Ordnungswert N des Stichwortes überein, wird die Quelle gedruckt. In der letzten DATA-Zeile 2000 ist ein bedeutungsloser String XXX untergebracht. Wenn er gelesen wird, ist der Suchvorgang beendet, weil keine weiteren Quelleneintragungen vorliegen.

Zeilen 600 bis 650

Nach Abschluß des Suchvorganges wird der Rücksprung zum Menü bzw. die Beendigung des Programms bewirkt.

Zeilen 700 bis 760

Hier werden die Stichworte gespeichert.

Zeilen 1000 bis 2000

Die Quellen sollten so aufbereitet werden, daß sie möglichst nicht mehr als 32 Zeichen aufweisen (Zeilenlänge!). Ab Zeile 1280 können beliebige weitere Eintragungen vorgenommen werden.

Der Steuermann Z 1013

Programm: Digitaluhr (2200 Bytes)

Versuche, die Zeit zu bestimmen und zu messen, gehören mit Sicherheit zu den ältesten Bestrebungen menschlicher Kultur. Sonnen- und Sanduhren im Altertum, immer kleiner und genauer werdende mechanische Zeitmesser und in den letzten Jahren zunehmend elektronische Uhren zeugen vom schöpferischen Erfindergeist der Menschen. Mikroelektronik und Computertechnik ermöglichen heute digital anzeigende exakte Uhren. Allabendlich kann man sich davon im Fernsehen überzeugen. Mit Hilfe unseres Z 1013 und der Programmiersprache BASIC wollen wir uns eine Uhr aufbauen, die einem normalen Anspruch an Genauigkeit gerecht wird. Unsere Uhr soll als Normaluhr, als Stoppuhr und als Wecker einsetzbar sein (Listing 16).

```
10 CLS:REM DIGITALUHR
20 PRINTAT(12,6);"D I G I T A L U H R"
30 FOR N=4 TO 26:PRINTAT(10,N);CHR$(42):PRINTAT(14,N);
   CHR$(42):NEXT N
40 FOR N=11 TO 13:PRINTAT(N,4);CHR$(42):PRINTAT(N,26);
   CHR$(42):NEXT N
50 PRINTAT(0,0);CHR$(32)
60 PRINTAT(30,23);">ENTER<"
70 IF INKEY$<>CHR$(13) THEN 70
100 CLS:REM LEISTUNGSANGEBOT
110 PRINT:PRINT:PRINT
120 PRINT"DIE UHR ARBEITET MIT FOLGENDEN":PRINT:PRINT
    "VARIANTEN :"
130 PRINT:PRINT:PRINT
140 PRINTSPC(5);"WECKUHR    = 1":PRINT:PRINT
150 PRINTSBC(5);"STOPPUHR   = 2":PRINT:PRINT
160 PRINTSPC(5);"NORMALUHR  = 3":PRINT:PRINT
170 INPUT"WELCHE UHR WIRD GEWUENSCHT ?";U
180 CLS:ON U GOTO 200,400,600
```

```
200 REM EINGABEN FUER WECKUHR
210 LET WU=1:PRINT:PRINT:PRINT
220 PRINT"WANN SOLL GEWECKT WERDEN ?" :PRINT:PRINT
230 INPUT"EINGABE STUNDE:     ";S$:PRINTAT(6,23;"UHR"
240 PRINT:INPUT"EINGABE MINUTEN:    ";M*:PRINTAT(8,23);
    "MINUTEN"
250 PRINT:PRINT:PRINT:PRINT
260 PRINT"   - REKORDER UND COMPUTER"
270 PRINT"     VERBINDEN !":PRINT
280 PRINT"   - SCHNELLSTOPTASTE DES"
290 PRINT"     REKORDERS DRUECKEN !":PRINT
300 PRINT"   - REKORDER AUF AUFNAHME"
310 PRINT"     STELLEN !":PRINT
320 PRINT"   - LAUTSTAERKE EINREGELN !"
330 PRINTAT(30,23);">ENTER<"
340 IF INKEY$<>CHR$(13) THEN 340
350 S=LEN(S$):M=LEN(S$)
360 SZ$=LEFT$(S$,S-1):SZ=VAL(SZ$)
370 MZ$=LEFT$(M$,M-1):MZ=VAL(M2$)
380 SE$=RIGHT$(S$,S-1):SE=VAL(SE$)
390 ME$=RIGHT$(M$,M-1):ME=VAL(ME$):GOTO 600
400 REM STOPPUHR
410 A=0:REM ZEHNERSTELLE STUNDE
420 B=0:REM EINERSTELLE STUNDE
430 C=0:REM ZEHNERSTELLE MINUTE
440 D=0:REM EINERSTELLE MINUTE
450 E=0:F=0:REM SEKUNDEN
460 PRINT:PRINT"STARTE MIT >ENTER< !"
470 PRINT:PRINT"STOPPE MIT > S4 / K < !"
480 PRINTAT(4,0);CHR$(32)
490 IF INKEY$<>CHR$(13) THEN490
500 N=340:GOTO 1000
600 CLS: REM NORMALUHR
610 PRINT:PRINT:PRINT
620 PRINT"STELLE DIE UHR  !"
630 PRINT:PRINT:PRINT
640 INPUT"ZEHNERSTELLE STUNDE:";A:PRINT
650 INPUT"EINERSTELLE STUNDE :";B:PRINT
660 INPUT"ZEHNERSTELLE MINUTE:";C:PRINT
670 INPUT"EINERSTELLE MINUTE :";D
680 E=0:F=0
690 PRINT:PRINT:PRINT:PRINT:PRINT
700 PRINT"STARTE BEIM ZEITZEICHEN !"
710 PRINTAT(30,23);">ENTER<"
720 IF INKEY$<>CHR$(13) THEN 720
730 CLS:N=260:GOTO 1000
1000 REM ARBEIT DER UHR
1010 IF WU=1 THEN PRINT"WECKZEIT ";S$;".";M$;" UHR"
1020 PRINTAT(20,8);A
1030 PRINTAT(20,10);B
1040 PRINTAT(20,13);":"
1050 PRINTAT(20,14);C
1060 PRINTAT(20,16);D
1070 PRINTAT(20,19);":"
1080 PRINTAT(20,20);E
1090 PRINTAT(20,22);F
1100 IF WU=1 AND A=SZ AND B=SE AND C=MZ AND D=ME AND E=0
     AND F=0 THEN 1300
```

```
1110 FOR Z=0 TO N:NEXT Z
1120 F=F+1
1130 IF F=10 THEN 1140:ELSE 1090
1140 F=0:E=E+1
1150 IF E=6 THEN 1160:ELSE 1080
1160 E=0:D=D+1
1170 IF D=10 THEN 1180:ELSE 1060
1180 D=0:C=C+1
1190 IF C=6 THEN 1200:ELSE 1050
1200 C=0:B=B+1
1210 IF B=10 THEN 1220:ELSE 1030
1220 B=0:A=A+1
1230 IF A=3 THEN 1240:ELSE 1020
1240 A=0: GOTO 1020
1300 REM WECKVORGANG
1310 BEEP
1320 IF INKEY$<>CHR$(13)THEN 1310:ELSE 1110
```

Listing 16: Digitaluhr

Zeilen 100 bis 180

Das Leistungsangebot der Z 1013-Uhr wird unterbreitet. Wahlweise bietet das Menü die Weckuhr, die Stoppuhr und die Normaluhr an. Die entsprechenden Initialisierungen werden durch eine ON . . . GOTO . . .-Anweisung angesprungen.

Zeilen 200 bis 240

Für die Weckuhr wird eine Hilfsgröße WU benötigt. Sie bekommt den Wert 1 zugeordnet. Dieser Wert ist bedeutungslos; er muß lediglich von Null verschieden sein. Die Aufforderung zur Eingabe der Weckzeit in Sunden und Minuten nimmt unser Programm als Zeichenkette entgegen.

Zeilen 260 bis 320

Da der Z 1013 nicht über einen eigenen Lautsprecher verfügt, nutzen wir den Recorder als Tongeber für das Wecksignal. In diesen Zeilen erfolgt die exakte Nutzerführung zur Vorbereitung des Kassettenrecorders als morgendlicher Ruhestörer.

Zeilen 350 bis 390

Die Zerlegung der Zeichenketten der Stunden und Minuten der Weckzeit und ihre Umwandlung in weiterverarbeitungsfähige numerische Werte wird mittels Zeichenkettenfunktionen vorgenommen. Es werden Zehner- und Einerstellen der Stunden und Minuten gebildet, um sie später mit der Realzeit vergleichen zu können. Ein Sprungbefehl verweist zur Zeile 600, in der der Stellvorgang der Uhr beginnt.

Zeilen 400 bis 450

Für die Organisation des Zeitstoppens werden alle Stunden-, Minuten- und Sekundenmarken auf Null gesetzt. Das ist die Vorbereitung für die exakte Zeitnahme. Die Bestimmung des Zeitfaktors N in Zeile 500 muß experimentell ermittelt werden. Geringfügige Schwankungen von Computer zu Computer können nämlich nicht ausgeschlossen werden. Der Zeitfaktor N wird später in Zeile 1119 benötigt. Starten erfolgt mit "ENTER", Stoppen mit "BREAK".

Zeilen 600 bis 680

Die tatsächliche Uhrzeit ist einzugeben. Der Zeitfaktor N unterscheidet sich von dem der Stoppuhr. Er wurde für die Normaluhr mit N = 260 ermittelt. Das Starten der Z 1013-Uhr erfolgt beim Zeitzeichen des Rundfunks oder nach einer anderen genauen Uhr.

Zeile 1010

Wenn die Hilfsgröße WU = 1 initialisiert ist, wird die gewünschte Weckzeit zur Kontrolle eingeblendet.

Zeilen 1020 bis 1090

Der Aufbau der Uhr erfolgt durch PRINT AT und Zuordnung der eingegebenen Stunden und Minuten. Selbstverständlich läßt sich die Uhr auch an jedem anderen Platz des Bildschirmes positionieren. Dazu sind lediglich die Zeilen- und Spaltenangaben zu ändern.

Zeile 1100

Es erfolgt nach jeder Sekunde eine Prüfung, ob die Realzeit mit der gewünschten Weckzeit übereinstimmt. Auch hier wird wieder die Hilfsgröße WU benötigt. Falls die Realzeit sekundengenau mit der Weckzeit übereinstimmt, erfolgt die Einleitung des Weckvorganges durch Sprung in die Zeile 1300.

Zeile 1110

Die Warteschleife dieser Zeile ist der eigentliche Taktgeber für unsere Zeitbestimmung. Der Zeitfaktor N wurde für Stopp- und Normaluhr an früherer Stelle bestimmt.

Zeilen 1120 bis 1240

Nach jedem unbedingten Sprung in Zeile 1020 wird der Sekundenzähler um den Wert 1 erhöht. Das zieht eine Kette nach sich. Nach 10 Sekunden muß die Zehnerstelle der Sekundenanzeige um den Wert 1 erhöht werden. Gleichzeitig wird natürlich die Einerstelle des Sekundenzählers auf 0 gesetzt. Nach 60 Sekunden muß der Minutenwert erhöht werden usw.

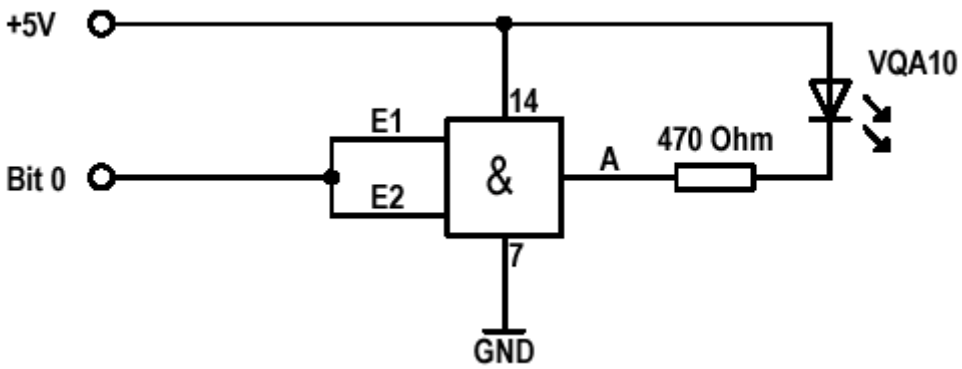
Zeilen 1300 bis 1320

Das Wecksignal wird durch die BEEP-Anweisung ausgelöst. Es kann nur durch die Betätigung der ENTER-Taste abgestellt werden. In dem Fall setzt die Normaluhr ihre Arbeitsweise fort. Die Weckfunktion realisiert der Z 1013 mit dem Recorder auch ohne zugeschaltetes Fernsehgerät, das man vor dem Schlafengehen besser ausschalten sollte.

Programm: Lauflicht (1231 Bytes)

Im Programm Digitaluhr wurde der Bildschirm als Uhrendisplay und der Kassettenmagnetbandanschluß zur Tonausgabe genutzt. Für das "Eingreifen" in Prozesse bietet unser Z 1013 aber noch die Möglichkeit der Ein- und Ausgabe digitaler Informationen. Das erfolgt über den Schaltkreis U 855, kurz PIO-Baustein genannt (PIO = parallel input output). Dieser Baustein hat zwei Datenkanäle (auch Tore oder Ports genannt) mit einer Breite von je 8 Bit. Das Port B benötigt der Z 1013 für sich selbst, mit dem Port A können wir experimentieren.
Dazu sind aber einige Hard- und Softwarevoraussetzungen zu erfüllen. Die Belegung der PIO-Buchse kann dem Handbuch Teil II B für den Z 1013 entnommen werden. Diese Buchse liefert uns die Anschlüsse für Masse, +5 V und die Datenleitungen für Bit 0 bis Bit 7. Die übrigen Anschlüsse werden für das Lauflicht nicht benötigt. Als "Lauflicht Hardware" benötigen wir folgende Bauelemente:

- 8 LED vom Typ VQA 10 oder ähnlich (LED = lichtemittierende Diode)
- 8 Widerstände 470 Ohm
- 2 Schalkreise D 100
- 1 Steckverbinder (dreireihig zu je 5 Anschlüssen).



Bit 1)
:)
:) ebenso wie für Bit 0
Bit 7)

Bild 13: Schaltung für ein Lauflicht mit 8 LED

Die daraus aufgebaute Schaltung zeigt Bild 13. Beim Aufbau der Schaltung sind die LEDs in der richtigen Reihenfolge von Bit 0 bis Bit 7 anzuordnen, anderenfalls entsteht mit dem hier vorgestellten Programm nur "Lichtsalat". Natürlich sind auch andere Schaltungsvarianten möglich, allerdings sollte der Hobby-Elektroniker die geringe Belastbarkeit der PIO-Ausgänge beachten.

Die Softwarevoraussetzungen sind Bestandteil des folgenden Programms (Listing 17), das ebenfalls nur einen Vorschlag mit verschiedenen Lauflichtvarianten darf stellt.

```
5 REM LAUFLICHT
10 WINDOW:CLS
15 OUT 1,207
20 OUT 1,0
25 OUT 0,0
30 PRINT:PRINT:PRINT:PRINT TAB(8);"LAUFLICHT"
35 PRINT TAB(8);STRING$(9,"-")
40 PRINT:PRINT:PRINT:PRINT"1=Lichtpunkte in einer Rich
   tung"
45 PRINT"2=Lichtpunkte in beiden Richt."
50 PRINT"3=zufaellige Lichtpunkte"
55 PRINT"4=Lichterkette (aufbauend)"
60 PRINT"5=Lichtp. von aussen zur Mitte"
65 PRINT"6=Ende"
70 PRINT:PRINT:PRINT"ABBRUCHTASTE=G"
75 WINDOW 19,22,0,31
80 CLS:INPUT"KENNZAHL=";K
85 IF K<1 OR K>6 THEN GOTO80
90 IF K=6 THEN WINDOW: CLS:END
95 PRINT:INPUT"ZEITFAKTOR (0..150)=";Q
100 ON K GOTO200,300,400,500,600
200 REM FALL 1
205 FOR I=0 TO 7
210 OUT 0,2^I
215 FOR Z=0 TO Q:NEXT Z
220 IF INKEY$="G" THEN OUT 0,0:GOTO80
225 NEXT I
230 GOTO205
300 REM FALL 2
305 FOR I=0 TO 7
310 OUT 0,2^I
315 FOR Z=0 TO Q :NEXT Z
320 IF INKEY$="G" THEN OUT 0,0:GOTO80
325 NEXT I
330 FOR I=7 TO 0 STEP -1
335 OUT 0,2^I
340 FOR Z=0 TO Q:NEXT Z
345 IF INKEY$="G" THEN OUT 0,0:GOTO80
350 NEXT I
355 GOTO305
400 REM FALL 3
405 LET C=INT(RND(1)*256)
410 OUT 0,C
415 FOR Z=0 TO Q:NEXT Z
420 IF INKEY$="G" THEN OUT 0,0:GOTO80
425 GOTO405
500 REM FALL 4
505 RESTORE545
510 FOR I=0 TO 41
515 READ A
520 OUT 0,A
525 FOR Z=0 TO Q:NEXT Z
530 IF INKEY$="G" THEN OUT 0,0:GOTO80
535 NEXT I
540 GOTO505
545 DATA 0,1,2,4,8,16,32,64,128,0,1,3,6,12,24,48,96,192,
    128,0,1,3,7,14,28
550 DATA 56,112,224,192,128,0,1,3,7,15,30,60,120,240,224,
    192,128
```

```
600 REM FALL 5
605 RESTORE645
610 FOR I=0 TO 4
615 READ A
620 OUT 0,A
625 FOR Z=0 TO 0:NEXT Z
630 IF INKEY$="G" THEN OUT 0,0:GOTO80
635 NEXT I
640 GOTO605
645 DATA 129,66,36,24,0
```

Listing 17: Lauflicht

Zeilen 5 bis 25

Hier erfolgt die Programmierung (auch als Initialisierung bezeichnet) des PIO-Schaltkreises mit der BASIC-Anweisung OUT Portadresse, Inhalt. Die Portadresse, die auf OUT folgt, darf nicht mit einer Speicherplatzadresse verwechselt werden. Zu jedem Port gehören zwei Portadressen, eine für die Steuerworte zur Programmierung des PIO-Bausteines und eine für das eigentliche Datenwort, das die entsprechenden Informationen enthält. Der HersteIIer des Z 1013 hat für die Steuerworte die Portadresse 1 und für das Datenwort die Portadresse 0 festgelegt. Wir wählen die Betriebsart 3 (siehe Handbuch Teil 1). Sie ermöglicht einen bitweisen Betrieb, wobei jedes der Bits 0 bis 7 auf Ein- oder Ausgabe programmiert werden kann. Diese Betriebsart 3 wird durch OUT 1,207 festgelegt. Bei unserem Lauflicht sollen natürlich alle Bits auf Ausgabe programmiert werden. Für die Ausgabe ist dem entsprechenden Bit die Zahl 0 und für die Eingabe die Zahl 1 zuzuweisen. Da alle Bits Informationen ausgeben sollen, folgt in Programmzeile 20 die Anweisung OUT 1,0. In Zeile 25 belegen wir durch OUT 0,0 jedes der 8 Bits des Datenwortes mit einer 0. Damit werden sämtliche LEDs durch die NAND-Gatter der zwei D100-Schaltkreise ausgeschaltet.

Zeilen 30 bis 100

In gewohnter Weise wird auf dem Bildschirm das Menü zur Auswahl von 5 Lichteffekten geboten. Als Abbruchbedingung wird ein Drücken der Taste G vereinbart. Die Wahl eines Zeitfaktors stellt einen kleinen "Zusatzservice" dar. Die Fallunterscheidung wird in Programmzeile 100 mit der ON GOTO-Anweisung realisiert.

Zeilen 200 bis 230

Im Fall 1 werden durch die geradzahligen Potenzen von 2 die einzelnen Bits von 2^0 bis 2^7 angesprochenen und über die OUT-Anweisung auf das Datenwort mit der Portadresse 0 gebracht. Die Zeile 215 steuert das Tempo über eine FOR NEXT-Schleife. Die Verwendung der PAUSE-Anweisung ist hier nicht sinnvoll, da n Bild kleinste Zeiteinheit mit rund einer zehntel Sekunde leider zu grob ist. Mit dieser "Tempozeile" kann noch vielfältig experimentiert werden, bis hin zu zufällig ausgewählten Zeiten für jedes Bit, also jede LED.

In Zeile 220 wird über die INKEY$-Anweisung die Tastatur abgefragt. Falls durch Drücken der Taste G ein Abbruch gewünscht wird, erfolgt mit OUT 0,0 ein Ausschalten aller LEDs und ein Rücksprung in das Eingabefenster.

Zeilen 300 bis 355

Der Programmaufbau basiert auf dem Fall 1. Hier wurde durch zusätzliche Rückwärtszählung ein Lauf der Lichtpunkte in beiden Richtungen programmiert.

Zeilen 400 bis 425

Der Fall 3 liefert durch die zufällige Ansteuerung einer bestimmten Anzahl von LEDs ein Lichtchaos. Wer es nicht so chaotisch mag, kann sich an dieser Stelle natürlich auch einen 8er oder 6er Würfel bauen, der die erwürfelte Augenzahl für eine gewisse Zeit anzeigt.

Zeilen 500 bis 550

An die vom Jahrmarkt her bekannten Lichterketten erinnert auch der Fall 4. Es wird eine Lichterkette aus zunehmender Anzahl von wandernden Lichtpunkten erzeugt. Die Lichterkette ist durch die Zahlenwerte in den DATA-Zeilen festgelegt. Rechnet man diese Dezimalzahlen in Dualzahlen um, So wird deren Bedeutung Sofort klar. Damit sind auch beliebige Änderungen möglich.

Zeilen 600 bis 645

Im Fall 5 laufen Lichtpunkte vom äußeren Rand zur Mitte. Auch dies wird über ZahIenwerte in einer DATA-Zeile realisiert. Ein Schönheitsfehler im Gesamtprogramm äußert sich im Zeitfaktor. Trotz gleichen Faktors werden die Fälle 1 bis 5 unterschiedlich schnell durchlaufen. Das liegt am unterschiedlichen Zeitbedarf unseres BASIC-Interpreters bei der Abarbeitung der einzelnen Programmteile. Hier könnte man sich mit einem Korrekturfaktor für Q aus der Affäre ziehen.

Programm: Modelleisenbahn (1263 Bytes)

Bei dieser kleinen Modelleisenbahnsteuerung gehen wir einen Schrie weiter, indem über den PIO-Baustein nicht nur Informationen ausgegeben, sondern auch im Datenwort anliegende Informationen (z. B. über eine erfolgreich gestellte Weiche) abgefragt und ausgewertet werden. Dazu muß der PIO-Baustein entsprechend programmiert werden. Vorher sind aber noch einige Überlegungen über die Hardwarevoraussetzungen anzustellen.

Bild 14 zeigt den Aufbau der Modelleisenbahn. Durch zwei Weichen werden ein großer und ein kleiner Kreis gebildet. Die Weichen sind mit Rückmeldekontakten ausgestattet. Der kleine und der große Kreisabschnitt sind mit Unterbrechungsgleisen versehen. Jeder Abschnitt kann über ein Relais Fahrstrom erhalten. Die Einfahrt einer Lok in einen der beiden Gleisabschnitte wird durch eine geeignete Durchfahrtkontrolle signalisiert. Am einfachsten Iäßt sich das mit einem Schaltgleis realisieren. Allerdings können dabei Impulse auftreten, die den Gesamtablauf stören. Zusammenstöße treten dennoch nicht auf, dafür

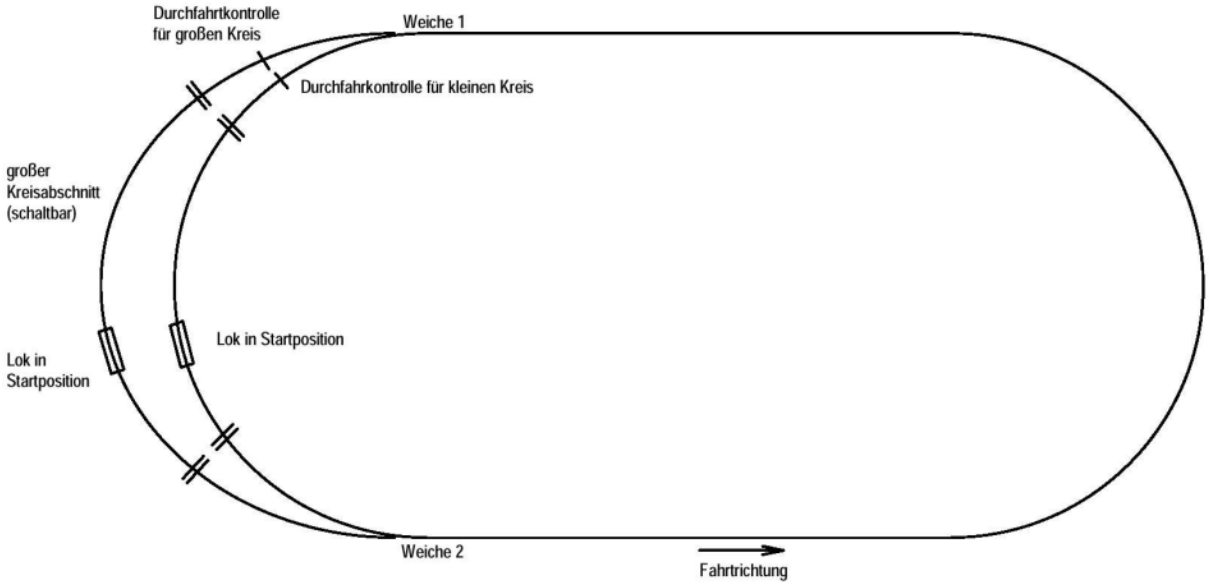


Bild 14: Aufbau der Modelleisenbahn

werden wir im Programm sorgen. Eleganter und störsicherer läßt sich die Durchfahrtkontrolle mit einer Lichtschranke realisieren. Mit dieser Durchfahrtkontrolle ist auch die im Bild 14 angegebene Fahrtrichtung festgelegt.

Folgende einfache Aufgabenstellung soll realisiert werden: Auf den kleinen und großen Kreisabschnitt wird jeweils eine Lok gesetzt. Nach Start des Steuerprogramms wird der zufällig vorliegende Status der Durchlaufkontrolle abgefragt. Je nach Status werden beide Welchen gestellt und über deren Rückmeldekontakte die korrekte Ausführung der Weichenstellung überprüft. Befinden sich beide Weichen z. B. in Schrägstellung, dann wird im kleinen Kreisabschnitt der Strom zugeschaltet, und die Lok setzt sich in Bewegung. Beim Passieren der Durchfahrtkontrolle wird mit einer Zeitverzögerung der kleine Kreisabschnitt stromlos gemacht, worauf die Lok an ihrer Ausgangsposition stehenbleibt.

Jetzt werden beide Weichen in Geradeausstellung gebracht und die korrekte Ausführung überprüft. Anschließend erhält der große Kreisabschnitt Strom, so daß sich diese Lok in Bewegung setzt. Beim Passieren der Durchfahrtkontrolle im großen Kreisabschnitt wird dieser mit Zeitverzögerung abgeschaltet. Das Spiel wiederholt sich nun von neuem mit der Lok im kleinen Kreisabschnitt. Für solch eine einfache Schaltung braucht man natürlich nicht unbedingt einen Computer, und wir reizen damit unseren Z 1013 als Steuermann längst nicht aus. Vielleicht vermittelt unser Beispiel aber den Modelleisenbahnern einige Anregungen, vor allem was die Sicherheit zur Vermeidung von "Unglücken" angeht. Sollte eine Weiche mechanisch versagen (risikofreudige Modellbahner können eine der Weichen auch gewaltsam festhalten), dann gibt das Programm eine Fehlermeldung aus und unterbricht den Gesamtablauf.

Die Hardwarelösung zum Stellen der Fahrströme und Weichen und zur Rückmeldung der Weichenstellungen und Durchfahrtkontrollen können wir hier aus Platzgründen nicht vorstellen. Wir haben dazu Schaltungen benutzt, die an der Päd. Hochschule "N. K. Krupskaja" Halle, Sektion Polytechnik, Wissenschaftsbereich Methodik, erarbeitet worden sind.

Nach Klärung der Hardwarebedingungen gilt es nun, über die Softwarebedingungen nachzudenken. Das 8 Bit breite Datenwort des PIO-Bausteines werden wir so programmieren, wie es Bild 15 zeigt. Zur Ausgabe, also zum Stellen von Weichen und zur

Schaltung von Fahrströmen, dienen die Bits 0 bis 3. Die Bits 4 bis 7 werden zur Eingabe von Prozeßinformationen (Rückmeldung der Weichen, Durchfahrtkontrolle) in den Z 1013 programmiert.

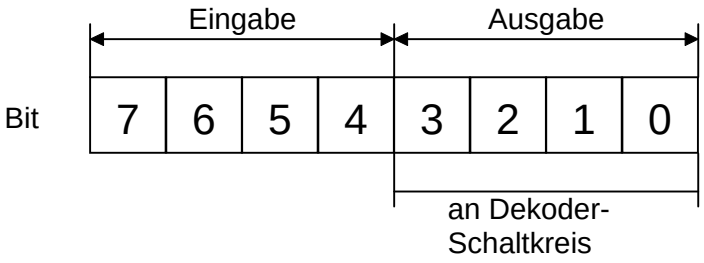


Bild 15: Datenwort für die Modelleisenbahnsteuerung

Allerdings reichen vier einzelne Bits nicht aus, um zwei Weichen in jeweils zwei Stellungen und zwei Fahrströme zu schalten. Deshalb werden die Bits 0 bis 3 an einen Dekoderschaltkreis (z. B. К155УД3) geführt. Mit vier Bits lassen sich ja 16 Zustände (Dezimalzahlen 0 bis 15) darstellen und damit für unsere Modellbahnsteuerung nutzen. Im vorliegenden Beispiel wurde folgende Belegung gewählt, die dann im Programm zu beachten ist:

Ausgabe auf Bit 0 bis Bit 3 über Dekoder, daraus folgt:
0 - keine Aktivität
1 - Weiche 1 in Geradeausstellung bringen
2 - Weiche 1 in Schrägstellung bringen
3 - Weiche 2 in Geradeausstellung bringen
4 - Weiche 2 in Schrägstellung bringen
5 - Fahrstrom für kleinen Kreisabschnitt zuschalten
6 - Fahrstrom für großen Kreisabschnitt zuschalten
7 bis 15 nicht genutzt

Eingabe auf Bit 4 bis Bit 7:
Bit 4 - Durchfahrtskontrolle:

Bit nicht gesetzt (= 0) → kleiner Kreisabschnitt aktiv, passiert die Lok die Durchfahrtskontrolle im kleinen Kreisabschnitt, dann wird Bit 4 gesetzt (= 1)

Bit gesetzt (= 1) → großer Kreisabschnitt aktiv, passiert die Lok die Durchfahrtskontrolle im großen Kreisabschnitt, dann wird Bit 4 zurückgesetzt (= 0)

Bit 5 - nicht genutzt
Bit 6 - Bit gesetzt (= 1), wenn Weiche 1 in Geradeausstellung
Bit 7 - Bit gesetzt (= 1), wenn Weiche 2 in Geradeausstellung

Das zugehörige Programm zeigt Listing 18. Mit der Anweisung OUT 1,207 wird, wie beim Lauflichtprogramm, der bitweise Betrieb des PIO-Bausteines programmiert. Mit OUT 1,240 werden die Bits 4 bis 7 auf Eingabe programmiert ($2^7 + 2^6 + 2^5 + 2^4 = 240$). Die Abfrage des Inhaltes des Datenwortes erfolgt mit der BASIC-Anweisung INP(O). Da der BASIC-Interpreter mit uns in Dezimalzahlen "verhandelt", wir aber den Inhalt der Bits 4, 6 und 7 abfragen müssen, erfolgt in den Programmzeilen 2000 bis 2030 eine Zerlegung der Dezimalzahl in eine Dualzahl. Dabei werden die einzelnen Bits dem Zahlenfeld DU(I) zugewiesen und stehen uns damit für die Auswertung zur Verfügung.

```
5 REM EISENBAHNSTEUERUNG
10 WINDOW:CLS:DIM DU(7)
15 PRINT AT(24,0);"ABBRUCHTASTE IST G"
20 WINDOW 0,23,0,31:CLS
25 OUT 1,207
30 OUT 1,240
35 OUT 0,0
40 PRINT AT(2,0);"BEIDE LOKS IN POSITION BRINGEN!"
45 PRINT AT(4,0);"ZUM START S TASTE DRUECKEN!"
50 IF INKEY$="S" THEN CLS:PRINT AT(4,0);"STEUERUNG DURCH
   COMPUTER":GOTO60
55 GOTO50
60 LET Z=INP(0):GOSUB2000
65 IF DU(4)=1 THEN GOTO200
100 REM W1 SCHRAEG STELLEN
105 OUT 0,2:PAUSE 5:OUT 0,0:PAUSE 5
110 LET Z=INP(0):GOSUB2000
115 IF DU(6)=1 THEN GOTO1000
120 REM W2 SCHRAEG STELLEN
125 OUT 0,4:PAUSE 5:OUT 0,0:PAUSE 5
130 LET Z=INP(0):GOSUB2000
135 IF DU(7)=1 THEN GOTO1000
140 PAUSE 5
145 LET Z=INP(0):GOSUB2000
150 IF DU(4)=1 GOTO200
155 REM KL.KR.EINSCHALTEN
160 OUT 0,5
165 LET Z=INP(0):GOSUB2000
170 IF DU(4)=0 THEN GOTO165
175 PAUSE 12
180 OUT 0,0
185 PAUSE 5
190 IF INKEY$="G" THEN WINDOW:CLS:END
200 REM W1 GERADE STELLEN
205 OUT 0,1:PAUSE 5:OUT 0,0:PAUSE 5
210 LET Z=INP(0):GOSUB2000
215 IF DU(6)=0 THEN GOTO1000
220 REM W2 GERADE STELLEN
225 OUT 0,3:PAUSE 5:OUT 0,0:PAUSE 5
230 LET Z=INP(0):GOSUB2000
235 IF DU(7)=0 THEN GOTO1000
240 PAUSE 5
245 LET Z=INP(0):GOSUB2000
250 IF DU(4)=0 THEN GOTO100
255 REM GR.KR.EINSCHALTEN
260 OUT 0,6
265 LET Z=INP(0):GOSUB2000
270 IF DU(4)=1 THEN GOTO265
275 PAUSE 15
280 OUT 0,0
285 PAUSE 5
290 IF INKEY$-" THEN WINDOW:CLS:END
295 GOTO100
1000 REM FEHLER
1005 OUT 0,0
1010 CLS:PRINT AT(2,0);"FEHLER BEI WEICHENSTELLUNG."
1015 PRINT AT(4,0);"NEUBEGINN MIT TASTE S!"
1020 GOTO50
2000 REM ZERLEGUNG
2005 LET R=Z:LET T=128
2010 FOR I=7 TO 4 STEP-1
2015 LET DU(I)=INT(R/T)
```

```
2020 LET R=R-T*DU(I):LET T=T/2
2025 NEXT I
2030 RETURN
```

Listing 18: Modelleisenbahnsteuerung

Bei nicht ordnungsgemäßer Weichenstellung sorgt der Programmteil ab Zeile 1000 für die Bildschirminformation und den Programmabbruch. Bei Programmunterbrechungen, z. B. mit der STOP-Taste, ist stets die Anweisung OUT 0,0 über die Tastatur einzugeben, damit keine Schäden an Weichen, Relais oder entsprechenden Bauteilen auftreten.

Für unseren gemächlichen Modellbahnbetrieb reicht die PAUSE-Anweisung zur Realisierung von Zeitverzögerungen vollkommen aus. Diese Verzögerungszeiten müssen für jede Anlage neu bestimmt werden. Damit wird z. B. auch das Weiterfahren der Lok im entsprechenden Unterbrechungsgleis bis zum Stillstand in Gleismitte realisiert.

Bei etwas Muße kann das Programm auch "in Gedanken" durchfahren werden. So zieht z. B. die Lok aus dem kleinen Kreisabschnitt ihre Bahn, solange in der Anweisungsfolge:

```
165 LET Z=INP(0):GOSUB 2000
170 IF DU(4)=0 THEN GOTO 165
```

eine wahre Aussage vorliegt.

Für viele industrielle Steuerungsaufgaben reicht die Abarbeitungsgeschwindigkeit eines BASIC-Interpreters nicht mehr aus. Solche Steuerprogramme werden dann in der Assemblersprache des jeweiligen Prozessors geschrieben, da diese Programme bis zu 50-, ja sogar 100mal schneller abgearbeitet werden können. Bei der Assemblerprogrammierung kann man auch direkt mit den einzelnen Bits umgehen (siehe Handbuch Teil 1). Der Vorteil unseres BASIC-Programms ist aber dessen Übersichtlichkeit und leichte Änderbarkeit; und darauf kommt es dem "Einsteiger" ja an.

Z 1013 als Partner für Lernen und Forschen

Programm: Meine Heimat DDR - Wissenstest (4347 Bytes)

Der Nachweis geografischen Wissens erfreut sich bei jung und alt großer Beliebtheit. Kein Kreuzworträtsel will darauf verzichten. Mit unserem Z 1013 möchten wir geografische Fakten bildhaft darstellen und durch erläuternde Texte bzw. Fragen systematisieren. Vom Heimatkreis bis zur Weltkarte können entsprechende Ideen entwickelt und umgesetzt werden.

Im vorliegenden Beispiel wollen wir Kenntnisse über unser Heimatland erarbeiten. Der Vorbereitungsaufwand ist allerdings nicht gering. Zunächst ist zu überlegen, wie die Bildschirmaufteilung vorgenommen werden soll. Schließlich brauchen wir Platz für die Landkarte und das Fragenprogramm! Den gesamten Text bringen wir in ein Fenster am unteren Bildschirmrand. Das ist deshalb gut möglich, weil der Z 1013 eben über 32 Zeilen verfügt. Für die Umrißkarte steht dann noch ein Raster von 24 Zeilen und 32 Spalten zur Verfügung. Die Umrisse der DDR benötigen davon zwar 24 Zeilen, aber nur etwa 18 Spalten. Wir grenzen das 32X32 Raster des Bildspeichers auf ein. 24X 18 Raster ein. Darin wird stilisiert der Umriß der Republik eingezeichnet. Bildschirmplatz für Bildschirmplatz sind Zeile, Spalte und mögliches Grafikzeichen für den jeweiligen "Grenzverlauf" zu bestimmen. Für das Programm waren für den Umriß allein 74 solcher Zahlentripel zu bestimmen, die später in DATA-Zeilen abgelegt werden.

Analog ist für die Aufbereitung von Flüssen, Gebirgen und Städten zu verfahren. Wenn man diese Arbeit hinter sich gebracht hat, geht es an das eigentliche Programmieren. Wir stellen die Anforderung, die Umrißkarte der DDR und geografische Hauptfakten durch Erfragen Schritt für Schritt zu vervollständigen. Richtige Antworten und Fehler sollen getrennt gezählt und zum Schluß ausgewertet werden. Ist alles richtig, stellt sich das Computerbild wie auf dem Foto dar (Bild 16). Im folgenden nun zum Listing 19.

Bild 16: DDR-Karte aus Pseudografikzeichen

```
10 CLS
20 REM STRINGVEREINBARUNGEN
30 R$="RICHTIG":NRW="FALSCH"
40 A$=" IST DAS ?":RA$="LAUTET DIE RICHTIGE ANTWORT"
50 L$="WELCHES LAND":S$="WELCHE STADT"
60 I$="WELCHE INSEL":F$="WELCHER FLUSS"
70 4109="WELCHES GEBIRGE"
80 N$="WELCHES NACHBARLAND":B$="WELCHE BEZIRKSSTADT"
90 PRINTAT(12,7);"WISSENSTEST  D D R":PAUSE50
100 REM UMRISS DDR
110 CLS:RESTORE3000:FOR N=1 TO 74:READ X,Y,Z
120 PRINTAT(X,Y);CHR$(Z):NEXTN
130 WINDOW 24,31,0,31:CLS
140 PRINTAT(2,25);"PRUEFE":PRINTAT(8,26);"DEIN":PRINTAT
    (14,25);"WISSEN"
200 REM FRAGEPROGRAMM:R=0:F=0
210 REM ABFRAGE DDR
220 PRINTL$+A$:PRINT:INPUT LL$:PRINT
230 IF LL$<>"DDR" THEN GOSUB2200:ELSE250
240 PRINT"DDR":GOSUB2220,
250 IF LLS="DDR" THEN GOSUB2100
300 REM ABFRAGE BERLIN
310 CLS:PRINTS$+A$:PRINT
320 X=11:Y=12:GOSUB2000
330 INPUTSS$:PRINT
340 IF SS$<>"BERLIN" THEN GOSUB2200:ELSE360
350 PRINT"BERLIN":GOSUB2220
360 IF SS$="BERLIN"THEN GOSUB2100
370 PRINTAT(11,12);CHR$(140):PAUSE10
400 REM ABFRAGE NACHBARLAENDER
410 PRINTN$+A$:PRINT:PRINTAT(9,18);"??":PRINTAT(10,19);
    "????"
420 INPUTNN$:PRINT
430 IF NN$<>"VR POLEN" THEN GOSUB2200:ELSE450
440 PRINT"VR POLEN":GOSUB2220
450 IF NN$="VR POLEN" THEN GOSUB2100
460 PRINTAT(9,18);"VR":PRINTAT(10,19);"POLEN":PAUSE 10
470 PRINTN$+A$:PRINT:PRINTAT(20,16);"???":INPUT NN$:PRINT
480 IF NN$<>"CSSR" THEN GOSUB2200:ELSE500
490 PRINT"CSSR":GOSUB2220
500 IF NN$="CSSR" THEN GOSUB2100
510 PRINTAT(20,16);"CSSR":PAUSE10
520 PRINTN$+A$:PRINT:PRINTAT(10,1);"???":INPUT NN$:PRINT
530 IF NN$<>"BRD" THEN GOSUB2200:ELSE550
540 PRINT"BRD":GOSUB2220
550 IF NN$="BRD" THEN GOSUB2100
560 PRINTAT(10,1);"BRD":PAUSE10
600 REM ABFRAGE FLUESSE
610 RESTORE3130
620 FORN=1 TO 16:READX,Y,Z
630 PRINTAT(X,Y):CHR$(Z):NEXT N
640 PRINTF$+A$:PRINT:INPUTFF$:PRINT
650 IF FF$<>"ELBE"THEN GOSUB2200:ELSE670
660 PRINT"ELBE":GOSUB2220
670 IF FF$="ELBE"THEN GOSUB2100
680 FOR N=1 TO 6 : READX,Y,Z
690 PRINTAT(X,Y):CHR$(Z):NEXT N
700 PRINTF$+A$;PRINT:INPUTFF$:PRINT
710 IF FF$<>"SAALE"THEN GOSUB2200:ELSE730
720 PRINT"SAALE" :GOSUB2220
730 IF FF$="SAALE" THEN GOSUB2100
800 REM ABFRAGE INSELN
810 CLS:PRINTI$+A$:PRINT
820 X=0:Y=14:GOSUB2000
```

```
830 INPUTII$:PRINT
840 IFII$<>"RUEGEN- GOSUB2200:ELSE860
850 PRINT"RUEGEN":GOSUB2220
860 IF II$<>"RUEGEN"THEN GOSUB2100
870 PRINTAT(0,14);"RUEGEN":PAUSE10
880 PRINTI$+A$:PRINT
890 X=3:Y=17:GOSUB2000
900 INPUT II$:PRINT
910 IF II$<>"USEDOM" THEN GOSUB2200:ELSE930
920 PRINT"USEDOM":GOSUB2220
930 IF II$="USEDOM" THEN GOSUB2100
940 PRINTAT(3,17);"USEDOM":PAUSE10
1000 REM ABFRAGE BEZIRKSSTAEDTE
1010 RESTORE1100
1020 FOR N=1 TO 14:READX$,X,Y
1030 PRINTB$+A$:PRINT:GOSUB2000
1040 INPUT BB$:PRINT
1050 IF BB$<>X$ THEN GOSUB2200:ELSE1070
1060 PRINTX$:GOSUB2220
1070 IF BB$=X$ THEN GOSUB2100
1080 PRINTAT(X,Y);"o":PAUSE 10
1090 NEXT N
1100 DATA COTTBUS,15,15,DRESDEN,19,13,ERFURT,18,4
1110 DATA FRANKFURT,12,17,GERA,18,9,HALLE,16,8,KARL-MARX-
     STADT
1120 DATA 20,10,LEIPZIG,17,10,MAGDEBURG,13,7,NEUBRANDEN
     BURG
1130 DATA 6,12,POTSDAM,12,11,ROSTOCK,2,8
1140 DATA SCHWERIN,6,6,SUHL,20,3
1200 REM ABFRAGE GEBIRGE
1210 PRINTG$+A$:PRINT:RESTORE3170
1220 FOR N=1 TO 5: READX,Y
1230 PRINTAT(X,Y);CHR$(199):NEXT N
1240 INPUT GG$:PRINT
1250 IF GG$<>"ERZGEBIRGE" THEN GOSUB2200:ELSE1270
1260 PRINT"ERZGEBIRGE":GOSUB2220
1270 IF GG$="ERZGEBIRGE" THEN GOSUB2100
1280 PRINTG$+A$:PRINT:RESTORE3180
1290 FOR N=1 TO 12:READX,Y
1300 PRINTAT(X,Y);CHR$(199):NEXT N
1310 INPUT GG$:PRINT
1320 IF GG$<>"THUERINGER WALD" THEN GOSUB2200:ELSE1340
1330 PRINT"THUERINGER WALD":GOSUB2220
1340 IF GG$="THUERINGER WALD" THEN GOSUB2100
1350 PRINTG$+A$:PRINT:RESTORE3190
1360 FOR N=1 TO 7:READX,Y
1370 PRINTAT(X,Y);CHR$(199):NEXT N
1380 INPUT GG$:PRINT
1390 IF GG$<>"HARZ" THEN GOSUB2200:ELSE1410
1400 PRINT"HARZ":GOSUB2220
1410 IF GG$="HARZ" THEN GOSUB2100
1500 REM AUSWERTUNG
1510 CLS:A=R+F
1520 PRINT"ANZAHL DER FRAGEN  :  ";A:PRINT
1530 PRINT"RICHTIGE ANTWORTEN :  ";A:PRINT
1540 PRINT"FALSCHE ANTWORTEN  :  ";A:PRINT
1550 END
2000 REM UP BLINKEN
2010 FOR Z=1 TO 5:PRINTAT(X,Y);" ":PAUSE5
2020 PRINTAT(X,Y);"?":PAUSE5
2030 NEXT Z:RETURN
2100 REM UP RICHTIG
2110 PRINTR$:PAUSE30:CLS:R=R+1:RETURN
2200 REM UP FALSCH
2210 PRINTNR$:PAUSE30:CLS:F=F+1:RETURN
```

```
2220 PRINT:PRINTRA$:PRINT:PAUSE30:CLS:RETURN
3000 DATA 0,11,129,0,12,150,0,13,147,0,14,145,1,8,144,1,9,
     158,1,10,158,1,11
3010 DATA 145,1,12,130,1,13,150,1,14,149,1,15,159,2,6,146,
     2,7,147,2,12,158
3020 DATA2,13,145,2,15,131,3,4,131,3,5,144,3,14,145,3,15,
     145,3,16,150,3,17,149
3030 DATA 4,16,145,4,17,144,5,3,152,5,15,192,6,3,145,4,3,
     153,4,15,145
3040 DATA 6,16,155,7,4,145,7,16,156,8,5,159,8,16,148
3050 DATA 9,4,153,9,16,192,10,4,156,10,17,145,11,5,159
3060 DATA 11,18,155,12,4,144,12,18,152,13,3,153,13,18,159
3070 DATA 14,3,152,14,18,155,15,2,144,15,18,156,16,1,148
3080 DATA16,19,155,17,1,158,17,19,161,18,1,152,18,17,144
3099 DATEI 18,18,150,18,19,132,19,1,159,19,16,144
3100 DATA 20,1,145,20,15,144,21,2,154,21,3,130,21,4,154
3110 DATA 21,5,158,21,6,130,21,7,145,21,8,144,21,9,158
3120 DATA 21,10,130,21,11,158,21,12,158,21,13,150,21,14,147
3130 DATA 9,6,145,10,7,155,11,7,161,12,7,161,13,7,156,14,
     7,242,14,8,164,14,9
3140 DATA 160,14,10,149,15,11,145,16,12,145,17,13,145,18,
     14,155,19,14,152
3150 DATA 20,14,145,9,5,130
3160 DATA 15,8,161,16,8,156,17,8,144,18,7,153,19,7,152,20,
     6,153
3170 DATA 21,9,21,10,20,11,21,11,20,12
3180 DATA 19,1,19,2,19,3,19,4,20,1,20,2,20,4,21,1,21,2,21,
     3,21,4,21,5
3190 DATA 12,4,12,5,13,4,13,5,14,4,14,5,14,6
```

Listing 19: Wissenstest - Meine Heimat DDR

Zeilen 20 bis 80

Um Speicherplatz und Schreibarbeit zu sparen, werden ständig wiederkehrende Begriffe und Sätze als Strings vereinbart. Im weiteren Programmverlauf wird konsequent auf die Stringvereinbarungen zurückgegriffen.

Zeilen 100 bis 120

Der Programmaufbau für die Umrißkarte ist denkbar einfach. Eine Schleifenanweisung wird 74fach durchlaufen. Sie liest dabei aus der DATA-Anweisung ab Zeile 3000 Zeilen- und Spaltenposition sowie den ASCII-Code des entsprechenden Grafikzeichens. Mit PRINT AT wird das Zeichen gesetzt.

Zeilen 200 bis 250

Hier beginnt das Frageprogramm. Richtige Antworten R und falsche Antworten F werden zunächst auf Null gesetzt. Im Fenster erscheint die Frage und die INPUT-Aufforderung zur Antwort. Bei richtiger Beantwortung wird das Unterprogramm in Zeile 2100 angesprungen, bei falscher Antwort das in Zeile 2200. Die Unterprogramme drucken die Bewertung der Antwort und zählen fortlaufend "richtig" oder "falsch".

Zeilen 300 bis 370

Mit der Frage nach der Hauptstadt wird erstmals im Programm ein Blinksignal verwendet.

Dieses Blinken wird im Unterprogramm ab Zeile 2000 erzeugt. Die blinkenden Fragezeichen finden bei nachfolgenden Programmteilen wiederum Verwendung. Nach Eingabe der Antwort wird das Fragezeichen in ein geografisches Symbol für die Hauptstadt umgewandelt und fest in die Umrißkarte eingetragen.

Zeilen 400 bis 560

Es folgen Fragen nach den Nachbarländern der DDR. Die blinkenden Fragezeichen werden nach Eingabe der Antworten jeweils durch die Buchstaben VR POLEN, CSSR und BRD ersetzt.

Zeilen 600 bis 730 und Zeilen 800 bis 940

Die Darstellung der Flüsse Elbe und Saale sowie die entsprechende Abfrage erfolgt in der gleichen Weise wie bei der Umrißkarte (Zeilen 100 bis 120). Die Inseln Rügen und Usedom werden analog zum Vorgehen der Zeilen 400 bis 560 bestimmt.

Zeilen 1000 bis 1140

Alle 14 Bezirksstädte sind nacheinander zu bestimmen und in die Karte einzutragen. Ihre Positionen werden aus DATA-Zeilen ab Zeilennummer 1100 aufgerufen.

Zeilen 1200 bis 1410

Die Darstellung und Ermittlung der Gebirge unserer Republik erfolgt in vorgegebener Reihenfolge unter Rückgriff auf Zeilen, Spalten und Code des Grafikzeichens, die durch PRINT AT und CHR$ gesetzt werden.

Zeilen 2000 bis 2030

Dieses Unterprogramm liefert blinkende Fragezeichen, die später durch kartografische Symbole bzw. Bezeichnungen ersetzt werden. Mit Hilfe einer Schleifenanweisung wird auf die vorgesehene PRINT AT-Position fünfmal im Wechsel ein Fragezeichen und ein Leerzeichen gesetzt. Der Pausenrhythmus von ca. 0,5 s bewirkt den Blinkeffekt.

Programm: Sternbilddarstellungen (2598 Bytes)

Dieses Programm liefert ein vielseitig ausbaubares Demonstrations- und Übungsbeispiel. Zum Können von Hobby-Astronomen gehört es, Sternbilder zu erkennen, sie am Firmament richtig einzuordnen und mit den exakten Bezeichnungen zu benennen. Das Einprägen der Lage der Hauptsterne eines Sternbildes zueinander erfordert entsprechende Übung. Hierbei ist der Z 1013 ein guter Helfer (Bild 17).
Zunächst muß man sich aus der Literatur, z. B. einem Sternatlas, Sternbilder heraussuchen und auf das 32X32 Raster unter Beachtung der exakten Abstandsverhältnisse und -winkel übertragen. Durch die Auswahl eines geeigneten Grafikzeichens (beispielsweise das Zeichen mit dem Code 201) macht man jeden einzelnen Stern über eine POKE-Anweisung auf der ermittelten Bildschirmposition sichtbar. Die Anzahl der Sternbilder für unseren "Z 1013 Sternbildatlas" wird lediglich durch die Speicherkapazität begrenzt. Diese Grenze

könnte für die Computergrundvariante je nach Anzahl der einzelnen Sterne im Sternbild bei etwa 20 bis 30 Sternbilddarstellungen liegen.

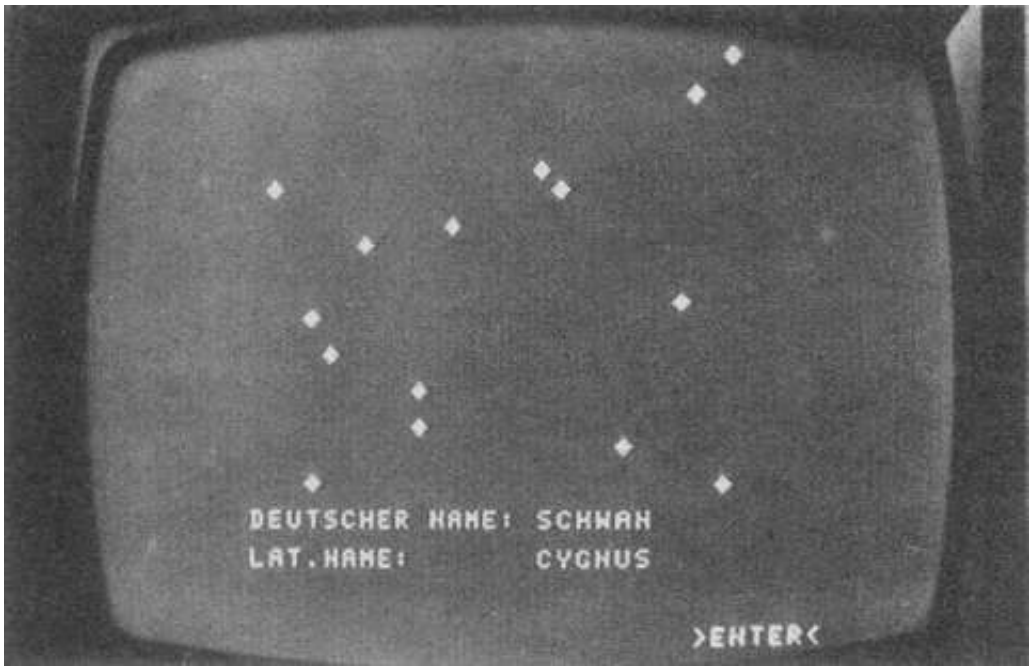


Bild 17: Beispiel einer Sternbilddarstellung

Variationsmöglichkeiten für den Computerastronomen sind im vorliegenden Programm enthalten. So können durch eine geeignete Auswahl von Grafikzeichen unterschiedliche Sterngrößen Berücksichtigung finden. Durch das Einzeichnen von Verbindungslinien zwischen den Sternen kann man die Deutung der Sternbildfiguren hervorheben. Es ist auch möglich, die Hauptsterne eines Sternbildes grafisch besonders zu gestalten und gesondert abzufragen. In diesem Sinn soll das vorliegende Programmbeispiel zur Weiterentwicklung anregen (Listing 20).

```
10 REM   TITELBILD
20 WINDOW:CLS:PRINTAT(0,0);CHR$(32)
30 PRINTAT(15,11);"S T E R N E"
40 FOR X=9 TO 23:PRINTAT(13,X);CHR$(42):PRINTAT(17,X);
   CHR$(42):NEXT
50 FOR Y=13 TO 17:PRINTAT(Y,9):CHR$(42):PRINTAT(Y,23);
   CHR$(42):NEXT
60 PAUSE 30 :CLS
70 WINDOW:CLS
100 REM   VEREINBARUNGEN
110 R$="RICHTIG":F$="FALSCH"
120 DR$="DRACHE":D$="DRACO"
130 KA$="KASSIOPEIA":K$="CASSIOPEIA"
140 SC$="SCHWAN":C$="CYGNUS"
150 AD$="ADLER":A$="AQUILA"
160 DN$="DEUTSCHER NAME : "
170  N$="LAT.NAME:       "
200 REM   WAHLMOEGLICHKEIT
210 PRINT"FOLGENDE AUSWAHLMOEGLICHKEITEN":PRINT
220 PRINT"STEHEN ZUR VERFUEGUNG :"
230 PRINT:BRINT:PRINT:PRINT:PRINT
240 PRINT"  VORFUEHREN DER STERNBILDER = 1":PRINT:PRINT
250 PRINT"  ASTRO - QUIZ               = 2":PRINT:PRINT
```

```
260 INPUT "ZAHL EINGEBEN :";W
270 IF W=2 THEN 500
300 REM   AUFLISTEN D.BILDER
310 CLS:PRINT"ZWISCHEN FOLGENDEN BILDERN":PRINT
320 PRINT"KANN GEWAEHLT WERDEN :"
330 PRINT:PRINT:PRINT:PRINT
340 PRINT"  DRACHE          = 1":PRINT
350 PRINT"  KASSIOPEIA      = 2":PRINT
360 PRINT"  SCHWAN          = 3":PRINT
370 PRINT"  ADLER           = 4":PRINT
380 PRINT:PRINT:PRINT
390 INPUT"ZAHL EINGEBEN :";z
400 REM   VORFUEHREN
410 CLS:WINDOW 25,31,0,31
420 IF Z=1 THEN PRINT DN$;DR$:PRINT:PRINT N$;D$:GOSUB 1000
430 IF Z=2 THEN PRINT DN$;KA$:PRINT:PRINT N$;K$:GOSUB 2000
440 IF Z=3 THEN PRINT DN$;SC$:PRINT:PRINT N$;C$:GOSUB 3000
450 IF Z=4 THEN PRINT DN$;AD$:PRINT:PRINT N$;A$:GOSUB 4000
460 PRINTAT(28,0);CHR$(32):PRINTAT(31,24);">ENTER<"
465 IF INKEY$<>CHR$(13) THEN 465
470 CLS:INPUT "NOCH EIN BILD ?  (J/N)";W2$:WINDOW:CLS
480 IF W2$="J" THEN 300
490 GOTO 799
500 REM     AUFBAU FRAGEBILD
510 WINDOW:CLS:PRINTAT(0,0);CHR$(32)
520 PRINTAT(9,7);"W I E   H E I S S T"
530 PRINTAT(11,11~;"D I E S E S"
540 PRINTAT (13,8);"S T E R N B I L D"
550 RESTORE 580
560 FOR I=1 TO 8:READ X:READ Y:READ Z
570 PRINTAT(X,Y);CHR$(Z)
580 DATA 16,15,174,16,16,173,17,16,172,18,15,174
590 DATA 19,15,171,19,16,172,21,15,129,21,16,128
600 NEXT :PAUSE 20 : CLS
610 WINDOW 25,31,0,31
620 N=1+INT(4*RND(1))
630 ON N GOSUB 1000,2000,3000,4000
640 PRINT "NAME DES STERNBILDES ?":PRINT
650 INPUT S$:PRINT
660 IF N=1 AND S$=DR$ OR S$=D$ THEN PRINT R$
670 IF N=1 AND S$<>DR$ AND S$<>D$ THEN PRINT I I
680 IF N=2 AND S$=KA$ OR S$=K$ THEN PRINT R$
690 IF N=2 AND S$<>KA$ AND S$<>K$ THEN PRINT F$
700 IF N=3 AND S$=SC$ OR S$=C$ THEN PRINT R$
710 IF N=3 AND S$<>SC$ AND S$<>C$ THEN PRINT F$
720 IF N=4 AND S$=AD$ OR S$=A$ THEN PRINT R$
730 IF N=4 AND S$<>AD$ AND S$<>A$ THEN PRINT F$
740 PRINTAT(30,0);CHR$(32):PRINTAT(31,24);">ENTER<"
745 IF INKEY$<>CHR$(13) THEN 745
750 CLS: PRINT"NOCH EIN BILD ?     (J / N)":PRINT
760 INPUTW$
770 WINDOW:CLS
780 IF W$="J" THEN 500
790 IF W$="N" THEN 799
799 END
999 REM  ZEICHNEN DER STERNE
1000 RESTORE 1030
1010 FOR S=1 TO 11:READ B
1015 A=-(5120-B)
1020 POKE A,201: NEXT S
1030 DATA 98,104,135,222,299,432,438,485,520,584,611
1040 RETURN
```

```
2000 RESTORE 2030
2005 A=-(5120-B)
2010 FOR S=1 TO 5 :READ B
2015 A=-(5120-B)
2020 POKE A,201 :NEXT S
2030 DATA 316,355,432,555,600
2040 RETURN
3000 RESTORE 3030
3010 FOR S=1 TO 15 :READ B
3015 A=-(5120-B)
3020 POKE A,201:NEXT S
3030 DATA 26,88,208,225,241,299,326,439,451,516,585,649,
     692,739,761
3040 RETURN
4000 RESTORE 4030
4005 A=-(5120-B)
4010 FOR S=1 TO 6 :READ B
4015 A=-(5120-B)
4020 POKE A,201 :NEXT S
4030 DATA 88,232,326,465,669,705
4040 RETURN
```

Listing 20: Sternbilder

Zeilen 10 bis 70

Der Programmtitel wird mit einer entsprechenden Umrahmung dargestellt. Um ein eventuelles Drucken des Programms mit einer Schreibmaschine zu erleichtern, werden hier wie auch in anderen Programmbeispielen Grafikzeichen im ASCII-Code angegeben und über die CHR$-Anweisung dargestellt.

Zeilen 100 bis 170

. .I

Die Zeichenketten für die Namen von Sternbildern und für häufig wiederkehrende Begriffe werden als Strings vereinbart. Das spart Schreibarbeit und erhöht die Übersichtlichkeit im Programm. Die Namen der Sternbilder werden sinnvollerweise sowohl mit der deutschen als auch der lateinischen Bezeichnung angegeben.

Zeilen 200 bis 600

Das angebotene Menü organisiert zwei Arbeitsweisen. Sternbilder können wahlweise mit ihrem Namen vorgestellt oder im Quizverfahren auf den Bildschirm gerufen werden. Exaktheit der Bezeichnungen und günstiger Bildaufbau sind für ein Übungsprogramm wesentlich. Innerhalb des Menüs erfolgt der Aufruf der Unterprogramme (ab Zeile 1000), die die Sterne zeichnen. Die Zeilen 560 bis 600 dienen lediglich der Herstellung eines großen Fragezeichens auf dem Bildschirm.

Zeilen 620 bis 745

Über den Zufallsgenerator wird die Auswahl der möglichen Sternbilder im Quizverfahren

organisiert. Bei der Erweiterung des vorliegenden Programms im Sinne einer größeren Anzahl von Sternbildern ist zu beachten, daß die mögliche Anzahl der - zahlen und die ON N GOSUB-Anweisung der tatsächlichen Zahl der Sternbilder angepaßt wird. Nach Aufruf des Sternbildes und Eingabe des entsprechenden Namens durch den Nutzer bewertet der Computer die Richtigkeit. Deutsche und Iateinische Bezeichnungen werden parallel als Antwort akzeptiert.

Zeilen 1000 bis 4040

Die Sterne des ersten Sternbildes "Drache" werden gezeichnet. Die RESTORE N-Anweisung sichert, daß nur die jeweils zugehörige DATA-Zeile angesprochen wird. Mit der Schleifenanweisung erfolgt die Festlegung der Anzahl der Einzelsterne. Die entsprechenden Bildspeicherplätze für die POKE-Anweisungen werden über DATA bereitgestellt. Ihre Organisation erfolgt durch die Addition der Anfangsadresse des Bild-Speichers (-5120) und des entsprechenden Wertes des 32X 32 Rasters. Alle weiteren Sternbilder werden nach dem gleichen Prinzip auf den nachfolgenden Zeilen (2000, 3000, 4000) programmiert.

Programm: Simulation eines GALTON·Brettes (1137 Bytes)

Computersimulationen werden genutzt, um möglichst reale Vorstellungen von Vorgängen und Prozessen durch künstliche Nachahmung zu gewinnen. Simulationsprogramme sind ein verbreitetes Anwendungsgebiet der Rechentechnik. Viele Simulationsmodelle lassen sich auch mit dem Z 1013 verwirklichen, obwohl natürlich fehlende Farbtüchtigkeit und Vollgrafik Grenzen setzen. Mit einem GALTON-Brett wird experimentell eine Demonstration der Binomialverteilung (Newtonsche Verteilung) vorgenommen.

Auf ein Brett sind Nägel so nebeneinander in Reihen eingeschlagen, daß eine rollende Stahlkugel auf dem geneigten Brett gezwungen wird, beim Auftreffen auf einen Nagel den Ausweg links oder rechts zu suchen. Nach dem Durchlaufen von Nagelreihen werden die Kugeln in n+1 Fächern aufgefangen und übereinander gestapelt. Bei genügend großer Anzahl der Kugeln wird in den Auffangbehältern eine Binomialverteilung erkennbar.

Das vorliegende Programm simuliert den Kugeldurchlauf durch ein GALTON-Brett. Die auf dem Bildschirm mögliche, aber tatsächlich zu geringe Zahl der rollenden Kugeln macht die Binomialverteilung zwar in der Tendenz immer sichtbar, im Einzelfall aber manchmal nicht so eindeutig beweiskräftig wie auf Bild 18. Nun zum Listing 21.

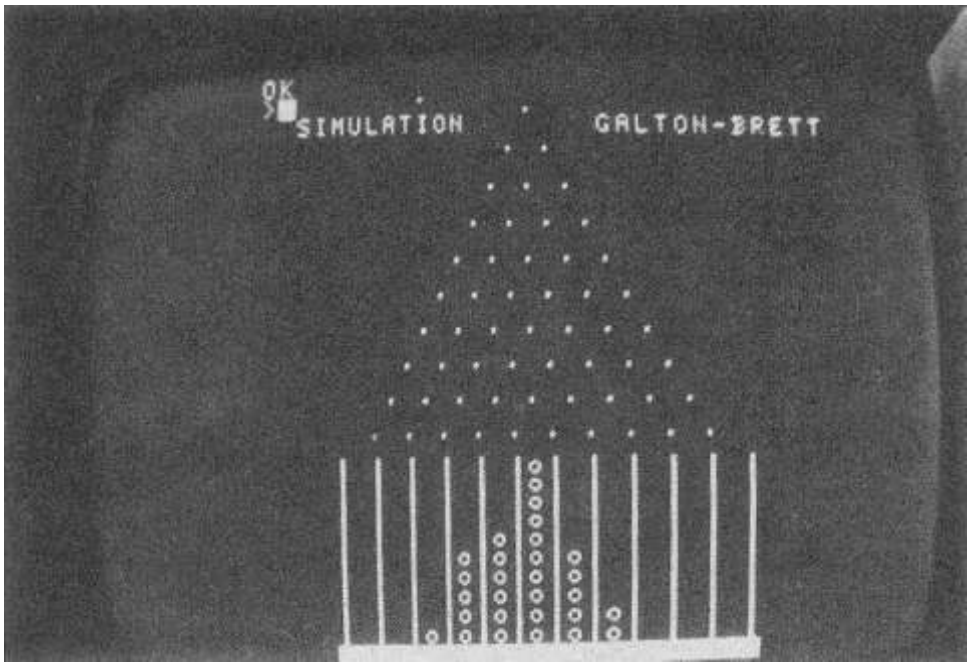


Bild 18: Darstellung eines Galton-Brettes

```
10 REM SIMULATION GALTONBRETT
20 CLS:PRINT"BITTE GESCHWINDIGKEIT DER":PRINT
30 PRINT"SIMULATION WAEHLEN !":PRINT:PRINT
40 PRINT"WERT VON 1 BIS 8 EINGEBEN":PRINT:PRINT
50 PRINT"HOECHSTE GESCHWINDIGKEIT      1":PRINT
60 PRINT"NIEDRIGSTE GESCHWINDIGKEIT    8":PRINT:PRINT
70 INPUT"WERT:";V
90 CLS:PRINTAT(0,0);CHR$(32)
100 REM BEHAELTERBODEN
110 FOR I=0 TO 22
120 A=-4124+I
130 POKE A,255:NEXT I
140 REM SEITENWAENDE
150 FOR I=0 TO 22 STEP 2
160 FOR K=0 TO 9
170 A=-4444+(K*32)+I
180 POKE A,161:NEXT K:NEXT I
190 REM NAGELBRETT
200 FOR I=1 TO 55
210 READ B
220 A=-5120+B
230 POKE A,46 :NEXT I
240 DATA 614,616,618,620,622,624,626,628,630,632,551,553,
    555,557,559,561
250 DATA 563,565,567,488,490,492,494,496,498,500,502,425,
    427,429,431
260 DATA 433,435,437,362,364,366,368,370,372,299,301,303,
    305,307
270 DATA 236,238,240,242,173,175,177,110,112,47
280 PRINTAT(2,2);"SIMULATION"
290 PRINTAT(2,19);"GALTON-BRETT"
300 REM START DER SIMULATION
310 A=-5105:POKE A,111 :PAUSE V
320 FOR N=1 TO 10
330 POKE A,32
```

```
340 Z=1+INT(2*RND(1))
350 IF Z=1 THEN A=A+31
360 IF Z=2 THEN A=A+33
370 POKE A,111:PAUSE V:POKE A,32
380 A=A+32:POKE A,111:PAUSE V:POKE A,32
390 NEXT N
400 IF PEEK(A+32)=111 THEN 450
410 A=A+32
420 IF PEEK(A)=32 THEN 430:ELSE 310
430 POKE A-32,32 :PAUSE 1:POKE A,111
440 GOTO 410
450 END
```

Listing 21: Galton-Brett

Zeilen 20 bis 90 ,

Die Durchlaufgeschwindigkeit der Kugeln ist vor Versuchsbeginn veränderbar. Die Geschwindigkeit hat Einfluß auf die Dauer des Gesamtexperimentes. Ihre Beeinflussung erfolgt durch eine PAUSE-Anweisung in Zeile 370. Durch die individuelle Wahl einer Geschwindigkeitsstufe von 1 (schnell) bis 8 (langsam) bestimmt der Nutzer den zeitlichen Verlauf des Experimentes.

Zeilen 100 bis 180

Der Auffangbehälter wird über einzelne POKE-Anweisungen aus entsprechend positionierten Grafikzeichen zusammengesetzt. Seine Konstruktion wird sinnvollerweise zuvor auf dem 32X32 Raster erprobt.

Zeilen 190 bis 290

Das Nagelbrett wird analog zum Auffangbehälter errichtet. Es ist zu berücksichtigen, daß die "Nägel" so gesetzt werden, daß sie den gleichen Abstand in der Reihe aufweisen, die Nagelzahl in jeder weiteren Reihe nach unten um zwei erhöhen und jeweils links und rechts neben dem Nagel Ausweichmöglichkeiten für die "Kugel" bieten.

Zeilen 300 bis 450

In diesen Zeilen wird der eigentliche Simulationsvorgang erzeugt. Die "Kugel" geht in der Bildschirmzeile 0 auf die Reise. Trifft sie auf kein Hindernis, setzt sie ihren Weg nach unten fort und wählt dafür den Speicherplatz A + 32 (A ist die Position des vorangegangenen Speicherplatzes). Damit wird ihr ein senkrechter Weg zugewiesen. Trifft die "Kugel" auf einen "Nagel", so wird über die RND-Funktion ein Wert 1 oder ein Wert 2 bereitgestellt. Dem Wert wird der Speicherplatz A -1 **oder** A +1 zugewiesen. Das bedeutet die Fortsetzung des Weges um das Hindernis nach links **oder** nach rechts. Nach neunmaliger Wiederholung dieses Vorganges fällt die "Kugel" in einen der zehn Auffangbehälter. In jeder Phase des Fallens erfolgt mit Hilfe der PEEK-Anweisung die Abfrage, ob der entsprechende Platz im Behälter für die Kugel noch frei ist.
Falls der darunterliegende Platz bereits besetzt ist, ist der Fall der Kugel beendet, und sie nimmt die darüberliegende Position im Behälter ein. Ist ein Auffanggefäß vollständig gefüllt, tritt die Abbruchbedingung der Zeile 400 in Kraft.

Programm: Simulation einer Auslese (1542 Bytes)

Unter Nutzung festgelegter Spielregeln simuliert der Computer mit diesem Programm ein Ausleseprinzip. Das Ganze könnte auch von Hand mit Würfeln erfolgen. Dazu wären einige Stunden erforderlich, während der Computer die Simulation in rund 10 Minuten ausführt. Auf einem Spielbrett mit 6 mal 6 Feldern werden vier Farben je neunmal wahllos verteilt. Da der Z 1013 keine Farben verarbeiten kann, nehmen wir dazu die Buchstaben B, G, R und V.

Im ersten Schritt werden die Koordinaten eines Spielfeldes mit dem Computer erwürfelt. Die auf diesem Spielfeld befindliche Farbe wird entfernt und vernichtet. Im zweiten Schritt werden die Koordinaten eines weiteren Spielfeldes zufällig ermittelt. Falls sich das soeben entstandene Leerfeld ergibt, muß der Wurf wiederholt werden. Auf dem im zweiten Schritt erwürfelten Spielfeld befindet sich also eine bestimmte Farbe. Sie wird auf dem Spielfeld belassen, diese Farbe aber zugleich auf das Leerfeld gesetzt, das nach dem ersten Wurf erzeugt wurde. Dieses Vorgehen wird solange wiederholt, bis sich auf dem Spielbrett nur noch eine Farbe befindet. Man mag zunächst kaum glauben, daß es mit diesen einfachen Spielregeln hier wirklich ein Ende gibt (Listing 22).

```
10 REM SIMULATION AUSLESE
20 WINDOW:CLS
30 DIM FF$(6,6):LET N=1
40 DEF FN ZU(X)=INT(RND(X)*6+1)
50 DEF FN Z(X) =3*X+9:DEF FN S(X)=3*X+11
60 REM BILDSCHIRMAUFBAU
70 PRINT TAB(11);"SIMULATION":PRINT TAB(11);STRING$
   (10,"-")
80 PRINT TAB(3);"EINER NATUERLICHEN AUSLESE":PRINT TAB(3);
   STRING$(26,"-")
90 PRINT:INPUT"EILTEMPO (J/N)?";IN$:PRINT
100 PRINT TAB(6);". SCHRITT":PRINT
110 PRINT"1. WURF=":PRINT STRING$(8,"-"):PRINT:PRINT
120 PRINT"VERNICHTUNG":PRINT:PRINT"DIESER":PRINT:PRINT"FAR
    BE"
130 POKE 113,30:PRINT" ":REM CURSOR POSITIONIEREN
140 PRINT AT(21,0);"2. WURF=":PRINT AT(22,0);STRING$
    (8,"-")
150 PRINT AT(25,0);"FARBE   AUF":PRINT AT(27,0);"FELD"
160 FOR I=12 TO 31:PRINT AT(10,I);CHR$(199):PRINT AT
    (29,I);CHR$(199):NEXT I
170 FOR I=11 TO 28:PRINT AT(I,12);CHR$(199):PRINT AT
    (I,31);CHR$(199):NEXT I
180 REM ZUFALLSFELD ERZEUGEN
190 FOR I=1 TO 9
200 LET F$="B":GOSUB 1000
210 LET F$="G":GOSUB 1000
220 LET F$="R":GOSUB 1000
230 LET F$="V":GOSUB 1000
240 NEXT I
250 REM 1. WURF
260 LET Z1=FN ZU(1):LET S1=FN ZU(1)
270 PRINT AT(7,0);N:PRINT AT(11,0);STR$(Z1+" :"+STR$(S1)
280 PRINT AT(FN Z(Z1),FN S(S1)+1);"*"
290 IF IN$="N" THEN PAUSE 50
300 LET FF$(Z1,S1)=" ":REM LEERZEICHEN
310 PRINT AT(FN Z(Z1),FN S(S1));FF$(Z1,S1)+" "
320 REM 2. WURF
330 LET Z2=FN ZU(1):LET S2=FN ZU(1)
340 IF FF$(Z2,S2)=" " THEN GOTO330:REM LEERZEICHEN
350 PRINT AT(23,0);STR$(Z2)+" :"+STR$(S2)
360 PRINT AT(25,6);FF$(Z2,S2)
```

```
370 PRINT AT(27,4);STR$(Z1)+" :"+STR$(S1)
380 LET FF$(Z1,S1)=(Z2,S2)
390 PRINT AT(FN Z(Z2),FN S(S2)+1);"*"
400 IF IN$="N" THEN PAUSE 50
410 PRINT AT(FN Z(Z2),FN S(S2)+1);" "
420 PRINT AT(FN Z(Z1),FN S(S1));FF$(Z1,S1)
430 REM ENDEBEDINGUNG
440 FOR I=1 TO 6
450 FOR J=1 TO 6
460 IF FF$(1,1)<>FF$(I,J) THEN LET N=N+1:GOTO260
470 NEXT J,I
480 PRINT AT(29,0);"GESCHAFFT!":END
1000 REM UP ZUFALLSFELD
1010 LET Z1=FN ZU(1):LET S1=FN ZU(1)
1020 IF FF$(Z1,S1)<>"" THEN GOTO 1010
1030 LET FF$(Z1,S1)=F$
1040 PRINT AT(FN Z(Z1),FN S(S1));FF$(Z1,S1)
1050 RETURN
```

Listing 22: Simulation einer Auslese

Zeilen 10 bis 50

Zunächst werden erforderliche Vereinbarungen vorgenommen. Zur Auswahl von Zufallszahlen wird die "Würfelfunktion" ZU(X) definiert. Hier ist zu beachten, daß in Ermangelung der RANDOMIZE-Anweisung der Computer nach Neustart des Interpreters stets mit der gleichen Anfangszahl beginnt und damit auch die gleiche Zufallszahlenreihe erzeugt. Das läßt sich z. B. durch eine Eingabe von RND (neg. Zahl) vor dem Programmstart verhindern (siehe Bedienhandbuch). Eine elegantere Methode wird im Programm Schiebepuzzle vorgeschlagen. Die selbst definierten Funktionen Z(X) und S(X) organisieren das korrekte Setzen der Buchstaben B, G, R und V mit Hilfe der PRINT-AT-Anweisung auf dem Bildschirm.

Zeilen 60 bis 240

Mit Hilfe der durch TAB und AT erweiterten PRINT-Anweisungen wird der gesamte Bildschirm aufgebaut. Als Umrahmung wird das Pseudografikzeichen mit dem Code 199 benutzt. Die Zufallsverteilung der vier Buchstaben auf jeweils neun Felder erfolgt mit Hilfe des Unterprogramms ab Zeile 1000. In Zeile 1020 wird gefragt, ob das ausgewählte Feld schon belegt ist. Falls ja, wird die Zufallsauswahl wiederholt.

Zeilen 250 bis 310

Hier wird das Feld erwürfelt, dessen Buchstabe zur Vernichtung vorgesehen ist. In diesen Programmzeilen erfolgt sowohl die Aktualisierung des Spielfeldes FF$ im Computer als auch die Spielfelddarstellung über PRINT-AT-Anweisungen auf dem Bildschirm. Die Zeile 290 nimmt Bezug auf die Eingabe in Zeile 90, wo nach einem gewünschten Eiltempo gefragt wird. Der Anfänger sollte sich erst einmal in Zeitlupe das Vorgehen ansehen, deshalb die Pause zu 5 Sekunden. In Zeile 280 wird das ausgewählte Feld durch ein danebengesetztes Sternchen kenntlich gemacht. Eine blinkende Darstellung des Feldes wäre schöner gewesen, aber das kann der Interpreter des Z 1013 nicht.

Zeilen 320 bis 420

Hier werden Auswahl und Anzeige des Feldes organisiert, dessen Buchstabe in das Leerfeld kopiert werden soll. Auch hier ermöglicht die Zeitlupe eine ausgiebige Beobachtung.

Zeilen 430 bis 480

Die Endbedingung vergleicht den Inhalt des ersten Feldelementes von FF$ mit allen weiteren Feldelementen. Sobald eine Ungleichheit auftritt, also noch andere Buchstaben auf dem Spielfeld sind, wird der Programmablauf ab Zeile 260 mit dem ersten Wurf eines weiteren Durchlaufes wiederholt. Bis zum Programmabschluß können rund 400, aber auch über 1000 Schritte erforderlich werden . Wer gern wetten will, soll vor dem Programmstart fragen: Welcher Buchstabe gewinnt?

Programm: Simulation des Spiels "Leben" (2090 Bytes)

Das SimuIationsspieI Life wurde 1970 von einem britischen Mathematiker erfunden. Die Begeisterung für dieses Spiel ist deshalb so groß, weil nur mit wenigen und einfachen Spielregeln eine nicht vorhersehbare Zahl verschiedener Entwicklungen auf dem Bildschirm vonstatten gehen kann. Im Lebensspiel kommen tote und lebende Zellen vor. Im Programm werden tote Zellen als Leerfelder und lebende als Sternchen dargestellt. Jede Zelle hat acht gleichberechtigte Nachbarzellen. Der gesamte Zellenverband unterliegt einem ständigen Generationswechsel, wobei die folgenden drei Zustände auftreten können:

1. Überleben: Wenn eine lebende Zelle zwei oder drei lebende Nachbarzellen hat, dann fühlt sie sich wohl und bleibt am Leben.
2. Tod: Hat eine lebende Zelle keinen oder nur einen Nachbarn, dann stirbt sie an Vereinsamung. Sie stirbt aber auch, wenn sie vier oder mehr Nachbarn hat (Erstickung).
3. Geburt: Auf jedem Leerfeld, das genau drei lebende NachbarzeIIen hat, entsteht eine lebende Zelle.

Besonders wichtig für den SpieIverIauf ist es, alle drei Regeln gleichzeitig für alle lebenden und toten Zellen anzuwenden. Bei einem "handbetriebenen" Lebensspiel sind zunächst alle Zellen auf Tod oder Überleben zu untersuchen. Die sterbenden Zellen dürfen aber zunächst nur durchgestrichen werden, denn alle Zellen müssen anschließend auf eine mögliche Geburt hin untersucht werden. Zum Schluß wird die entstandene Generation dann auf ein neues Blatt übertragen. Das ist mühsam, zum Glück haben wir unseren Z1013, in dem wir aber neben dem Spielfeld SF noch ein Zwischenfeld ZF aufbauen müssen (Listing 23). Bild 19 zeigt ein Beispiel der Entwicklung mehrerer Generationen aus einer Grundfigur.

```
 * *
 ***
 * *

 1 . GENERATION
ES LEBEN JETZT 8 ZELLEN


  * *
 ** **
  * *

 2 . GENERATION
ES LEBEN JETZT 12 ZELLEN

  ** **
  ** **
  ** **


 3 . GENERATION
ES LEBEN JETZT 10 ZELLEN

  ** **
 *    *
  ** **
```

Bild 19: Einige Generationen aus dem Spiel "Leben"

```
10 REM SIMULATION LEBEN
20 WINDOW:CLS:CLEAR 56
30 DIM SF(17,17):DIM ZF(17,17)
40 LET AN=0 :LET FB=0
50 PRINT"WIEVIEL GENERATIONEN SOLLEN"
60 INPUT"BERECHNET WERDEN?";GE
70 PRINT"GIB ZEILENWEISE DIE AUSGANGS-"
80 PRINT"STELLUNG MIT FOLGENDEN SYMBOLEN":PRINT"EIN:"
90 PRINT"ZELLE=* LEERFELD=+":PRINT"ABSCHLUSS=ENDE"
100 FOR ZE=1 TO 17
110 INPUT M$
120 IF M$="ENDE" THEN GOTO 200
130 FOR SP=1 TO LEN(M$)
140 IF MID$(M$,SP,1)<>"*"THEN GOTO 170
150 LET SF(ZE,SP)=1
160 LET AN=AN+1
170 LET FB=(SP+FB+(SP-FB)*SGN(SP-FB))/2:REM MAX. SPALTEN
    ZAHL ERMITTELN
180 NEXT SP
190 NEXT ZE
200 LET G=0
210 LET ZZ=ZE-1
220 GOSUB 1000
230 IF AN=0 OR G=GE THEN PRINT"PROGRAMMENDE":CLEAR 256:END
240 REM SPIELREGELABFRAGE
250 LET G=G+1
260 LET AN=0
270 FOR ZE=1 TO ZZ
280 FOR SP=1 TO FB
290 LET FU=0
300 IF ZE<17 AND SP<17 THEN GOTO 330
310 PRINT"RAND ERREICHT!"
```

```
320 PRINT"PROGRAMMABBRUCH":CLEAR 256:END
330 IF NOT(-ZF(ZE,SP+1)) OR SP=FB THEN GOTO 350
340 LET PU=PU+1
350 IF NOT(-ZF(ZE+1,SP+1)) OR ZE=ZZ OR SP=FB THEN GOTO 370
360 LET PU=PU+1
370 IF NOT(-ZF(ZE+1,SP)) OR ZE=ZZ THEN GOTO 390
380 LET PU=PU+1
390 IF ZE=ZZ OR SP=1 THEN GOTO 420
400 IF NOT(-ZF(ZE+1,SP-1)) THEN GOTO 420
410 LET PU=PU+1
420 IF SP=1 THEN GOTO 450
430 IF NOT(-ZF(ZE,SP-1)) THEN GOTO 450
440 LET PU=PU+1
450 IF ZE=1 0R SP=1 THEN GOTO 480
460 IF NOT(-ZF(ZE-1,SP-1)> THEN GOTO 480
470 LET PU=PU+1
480 IF ZE=1 THEN GOTO 510
490 IF NOT(-ZF(ZE-1,SP)) THEN GOTO 510
500 LET PU=PU+1
510 IF ZE=1 OR SP=FB THEN GOTO 540
520 IF NOT(-ZF(ZE-1,SP+1)) THEN GOTO 540
530 LET PU=PU+1
540 IF PU=3 THEN GOTO 560:REM GEBURT
550 IF PU<>2 OR NOT(-ZF(ZE,SP)) THEN GOTO 580
560 LET SF(ZE,SP)=1
570 LET AN=AN+1
580 NEXT SP
590 NEXT ZE
600 GOSUB 1000
610 IF AN== OR G=GE THEN PRINT"PROGRAMMENDE":CLEAR 256:END
620 GOTO 240
1000 REM UP DARSTELLUNG UND ZWISCHENFELD
1010 PRINT:PRINT G;". GENERATION"
1020 PRINT"ES LEBEN JETZT";AN;"ZELLEN":PRINT
1030 IF AN=0 THEN RETURN
1040 LET VS=17:LET VZ=17:LET MZ=0:LET MS=0
1050 FOR ZE=1 TO ZZ
1060 FOR SP=1 TO FB
1070 IF NOT(-SF(ZE,SP)) THEN GOTO 1130
1080 LET VZ=(VZ+ZE+(VZ-ZE)*SGN(ZE-VZ))/2
1090 LET VS=(VS+SP+(VS-SP)*SGN(SP-VS))/2
1100 LET MZ=(MZ+ZE+(MZ-ZE)*SGN(MZ-ZE))/2
1110 LET MS=(MS+SP+(MS-SP)*SGN(MS-SP))/2
1120 PRINT TAB(SP);"*";
1130 NEXT SP
1140 PRINT
1150 NEXT ZE
1160 IF G=GE THEN RETURN
1170 LET MZ=MZ-VZ+3
1180 LET MS=MS-VS+3
1190 FOR I=1 TO 17
1200 FOR K=1 TO 17
1210 LET ZF(I,K)=0
1220 NEXT K
1230 NEXT I
1240 FOR ZE=1 TO ZZ
1250 FOR SP=1 TO FB
1260 IF NOT(-SF(ZE,SP)) THEN GOTO 1290
1270 LET ZF(ZE-VZ+2,SP-VS+2)=1    .
1280 LET SF(ZE,SP)=0
```

```
1290 NEXT SP
1300 NEXT ZE
1310 LET ZZ=MZ:LET FB=MS
1320 RETURN
```

Listing 23: Simulation des Spiels "Leben"

Zeilen 10 bis 230

Beim Z 1013 ohne 1260 IF NOT (ist die Simulation des Lebensspiels auf Darstellungen von 17 Zeilen und 17 Spalten eingeschränkt (Zeile 30). Hier muß sogar noch der Textspeicherbereich auf 56 Zeichen reduziert werden (Zeile 20), um keine Probleme mit dem Speicherplatz zu bekommen. Bei vorhandenen Speichererweiterungsmodulen wäre eine Erweiterung auf 32 Spalten und, unter Berücksichtigung des erklärenden Textes, auf etwa 25 Zeilen sinnvoll. Hierzu müßten die Programmzeilen 30, 100, 300, 1040, 1190 und 1200 entsprechend geändert werden. Vielleicht findet der Leser aber generell eine sparsamere Programmversion, denn bei der Dimensionierung in Zeile 30 gehen wir doch recht großzügig mit den knapp bemessenen Bytes in unserem Z 1013 in der Grundversion um.

Ab Zeile 40 erfolgen weitere Vereinbarungen (AN = Anzahl lebenden Zellen, FB = Feldbreite, also Spaltenanzahl, GE = Anzahl der zu berechnenden Generationen, G = Generationszähler, ZZ = Anzahl der Zeilen) und die Realisierung aller erforderlichen Eingaben.

Zeilen 1000 bis 1320

In diesem Unterprogramm erfolgt die Darstellung des Textes und des aktuellen Generationsmusters auf dem Bildschirm. Es enthält auch die Abbruchbedingungen für die Fälle, bei denen keine lebenden Zellen mehr vorhanden sind (AN = 0) oder die vorgegebene Anzahl von Generationen (G = GE) erreicht wurde. Die zweifache Laufanweisung (Zeilen 1050 bis 1150) fragt den Spielfeldinhalt SF(ZE, SP) ab. Ein Sternchen für eine lebende Zelle ist im Spielfeld SF als Ziffer 1 abgespeichert (siehe Zeile 150). In Zeile 1070 wird gefragt, ob das Feldelement ZE, SP eine Eins enthält. Falls ja, dann ergibt sich mit -1 eine wahre Aussage, die durch NOT negiert wird. In diesem Fall werden die Zeilen 1080 bis 1120 abgearbeitet. Hier wird das Maximum der Variablen VZ, VS (Verschiebefaktoren für Zeile und Spalte) und MZ, MS (maximale Zeilen- und Spaltenzahl) ermittelt und das aktuelle Generationsfeld auf dem Bildschirm ausgegeben.

Nach Löschung des Zwischenfeldes ZF wird das neue Spielfeld SF auf das Zwischenfeld übertragen und zugleich das Spielfeld gelöscht. Der Variablen ZZ wird dann die maximale Zeilenzahl MZ und der Variablen FB die maximale Spaltenzahl MS zugewiesen.

Zeilen 240 bis 620

Nach Überprüfung auf eventuelle Randüberschreitung wird die relativ aufwendige Spielregelabfrage eingeleitet. Die Abarbeitung der doppelten Laufanweisung von Zeile 270 bis 590 macht die zeitlichen Schwächen eines BASIC-Interpreters deutlich, denn jede einzelne Zelle muß auf vorhandene Nachbarzellen untersucht werden. Die Variable mit dem Namen PU zählt dabei die Nachbarzellen. In Zeile 540 wird bei drei lebenden Nachbarzellen eine Geburt (Zeile 560) ausgelöst. Auch im Überlebensfall erfolgt das Setzen einer lebenden Zelle in das entsprechende Feld des Spielfeldes SF. Als Ergebnis dieser Spielregelabfrage ist das neue

Spielfeld SF entstanden, mit dem wieder in das Unterprogramm ab Zeile 1000 gesprungen wird.

Programm: Satz des Pythagoras (1920 Bytes)

Der Satz des Pythagoras gehört zu den bekanntesten Lehrsätzen der Mathematik (Bild 20). Jeder Schüler lernt ihn bei Dreiecksberechnungen anzuwenden. Seine mathematische Grundlage besteht darin, daß sich in einem rechtwinkligen Dreieck eine Seitenlänge berech-

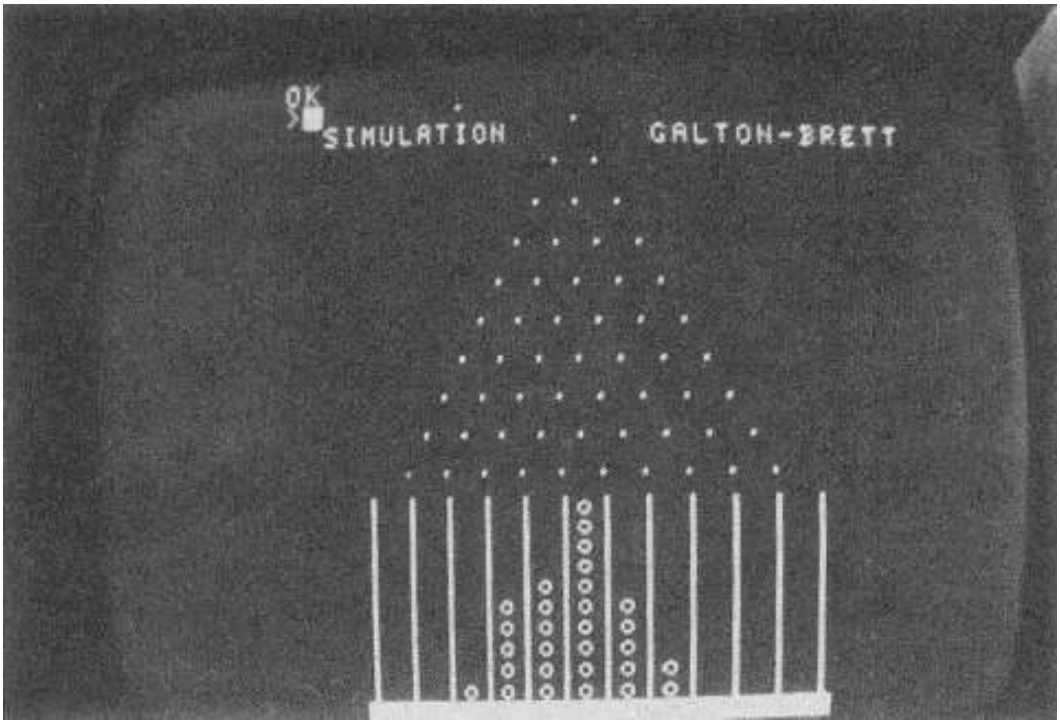


Bild 20: Satz des Pythagoras

nen läßt, wenn die Längen der beiden anderen Seiten bekannt sind. Das vorliegende Programm ist die erste Stufe eines Übungsprogrammes. Es bietet die grafische Darstellung des Lehrsatzes sowie die Grundformel und fordert dazu auf, wahlweise eine der drei Seiten mit selbstgewählten Werten für die anderen zwei Seiten zu berechnen (Listing 24). Erweiterungsmöglichkeiten sind denkbar. Durch das Zufallsprinzip könnte der Computer selbst die zu berechnende Seite auswählen. Es ist dann zu beachten, daß die Seitenlänge c stets größer als die von a oder b sein muß. Der Z 1013 wäre bei entsprechender Programmveränderung auch in der Lage, die Aufgabe selbst zu stellen, den Nutzer zur Berechnung mit anderen Hilfsmitteln aufzufordern und das errechnete Ergebnis auf Richtigkeit zu überprüfen.

```
10 REM SATZ DES PYTHAGORAS
20 WINDOW:CLS:PRINTAT(0,0);CHR$(32)
30 PRINTAT(1,1);"SATZ"
40 PRINTAT(3,1);"DES"
50 PRINTAT(5,1):"PYTHAGORAS"
100 REM AUFBAU DES DREIECKS
110 A=-5120:FOR I=0 TO 11
120 POKE A+244+I*32,159:NEXT I
130 FOR I=0 TO 5
140 POKE A+206+I,248:NEXT I
150 B=155:C=156
160 POKE A+238,B:POKE A+270,C
170 POKE A+303,B:POKE A+335,C
180 POKE A+368,B:POKE A+400,C
190 POKE A+433,B:POKE A+465,0
200 POKE A+498,B:POKE A+530,0
210 POKE A+563,B:POKE A+595,C
220 FOR I=1 TO 30:READ B
230 POKE A+B,218
240 DATA 239,240,241,242,243,271,272,273,274,275,304,305,
    306,307
250 DATA 336,337,338,339,369,370,371,401,402,403,434,435,
    466,467,499,531
260 NEXT I
270 PAUSE 20
300 REM SEITENBEZEICHNUNG
310 PRINTAT(12,21);"a"
320 PRINTAT(12,15);"c"
330 PRINTAT(5,16);"b"
340 PAUSE 20
400 REM AUFBAU SEITENQUADRATE
410 FOR I=0 TO 11
420 POKE A+212+I ,249
430 POKE A+628+I,158
440 POKE A+255+I*32,244
450 NEXT I
460 FORI=0 TO 5
470 POKE A+45+I*32,192
480 POKE A+52+I*32,159
490 POKE A+46+I,158
500 NEXT I:POKE A+212,136
510 FOR I=1 TO 6:READ B:READ C
520 D=155:E=156
530 POKE A+B,D:POKE A+C,E
540 DATA 418, 450,483,515,548,580,613,645,678,710,743,775
550 NEXT I
560 FOR I=1 TO 12:READ B:READ C
570 D=147:E=146
580 POKE A+B,D:POKE A+C,E
590 DATA 237,236,267,266,297,296,327,326,357,356,387,386
600 DATA 627,626,657,656,687,686,717,716,747,746,777,776
610 NEXT I
620 PAUSE 20
630 PRINTAT(2,23);"2  2  2"
640 PRINTAT(3,22);"a +b =c"
650 PAUSE 50
700 REM SEITENBERECHNUNG
710 WINDOW 24,31,12,31
720 PRINT"WELCHE SEITE SOLL":PRINT
730 PRINT"BERECHNET WERDEN ?":PRINT
740 INPUT"(EINGABE a,b oder c)";S$
750 CLS
```

```
760 IF S$="A" OR S$="a" THEN 800
770 IF S$="B" OR S$="b" THEN 860
780 IF S$="C" OR S$="c" THEN 920
790 GOTO 720
800 INPUT"LAENGE b =";B
810 PRINT:INPUT"LAENGE c =";c
820 A=SQR(C*C-B*B)
830 PRINT:PRINT"LAENGE a =";A
840 PRINT"==========="
850 GOTO 980
860 INPUT"LAENGE a =";A
870 PRINT:INPUT"LAENGE c =";C
880 B=SQR(C*C-A*A)
890 PRINT:PRINT"LAENGE b =";B
900 PRINT"==========="
910 GOTO 980
920 INPUT"LAENGE a =";A
930 PRINT:INPUT"LAENGE b =";B
940 C=SQR(A*A+B*B)
950 PRINT:PRINT"LAENGE c =";C
960 PRINT"==========="
970 GOTO 980
980 PRINTAT(31,24);">ENTER<"
990 IF INKEY$<>CHR$(13) THEN 990
1000 CLS
1010 PRINT"NOCH EINE AUFGABE ?"
1020 PRINT:INPUT"(J/N)";F$
1030 IF F$="J" THEN 710
1040 END
```

Listing 24: Satz des Pythagoras

Zeilen 10 bis 50

Mit diesen Zeilen wird der Titel so in den Bildschirm gesetzt, daß der nachfolgende Aufbau des Dreiecks und seiner Seitenquadrate nicht behindert wird.

Zeilen 100 bis 270

Die Konstruktion des rechtwinkligen Dreiecks erfolgt durch das Setzen ausgewählter Grafikzeichen auf entsprechende Bildschirmplätze. Dazu nutzen wir die POKE-Anweisung. Bezugspunkt A ist immer der linke obere Bildspeicherplatz mit dem Wert -5120. Durch vorheriges Aufzeichnen im 32 x 32 Raster wird der entsprechende Platz im Bildspeicher ermittelt. Für fortlaufende Bildspeicherplätze werden Schleifenanweisungen zum Bildaufbau genutzt. Wo dies nicht möglich ist, werden einzelne Bildspeicherplätze direkt angesprochen. Das ist allerdings wesentlich aufwendiger und benötigt auch mehr Programmzeilen. Die Punktierung der Dreiecksfläche erfolgt ebenfalls mit Hilfe von POKE. Die zugehörigen Werte der Bildspeicherplätze sind in einer DATA-Zeile abgelegt.

Zeilen 300 bis 340

Die Kennzeichnung der Seiten a, b und c des rechtwinkligen Dreiecks wird durch PRINT AT-Anweisungen vorgenommen.

Zeilen 450 bis 650

Der Aufbau der Quadrate über den Seiten a, b und c erfolgt in Anwendung und Kombination der gleichen Arbeitsmöglichkeiten, wie sie auch in den Zeilen 100 bis 270 zur Realisierung des grafischen Bildaufbaus genutzt wurden.

Zeilen 700 bis 990

Der Berechnungsvorgang wird durch Erfragen der zu berechnenden Seite und die Aufforderung zur Eingabe der Werte der bekannten Seiten eingeleitet. Dies erfolgt mit INPUT-Anweisungen. In den Zeilen 820, 880 und 940 finden wir jeweils die umgestellte Formel des Satzes des Pythagoras in BASIC-Schreibweise. Der Ergebnisausdruck erfolgt in dem vorliegenden Programm ungerundet. Durch Einbau des Unterprogramms zum Runden (siehe "Werkzeugkiste") kann das Programm sinnvoll ergänzt und verbessert werden.

Zeilen 1000 bis 1040

Bei gewünschter Wiederholung einer Übung zum Satz des Pythagoras erfolgt ein Rücksprung in Zeile 710. Damit wird nur das aktuelle Fenster innerhalb der WINDOW-Festlegung gelöscht und der Berechnungsvorgang erneut eingeleitet. Ein Neuaufbau der Grafik ist für das Grundverständnis nicht erforderlich, und deshalb verzichten wir darauf.

Programm: Computerfragebogen (3412 Bytes)

Oft werden Fragebögen bei Forschungsaufträgen zur Ermittlung von interessierenden Sachverhalten eingesetzt. Damit sind immer Schreibaufwand, Papierverbrauch, nachträgliches Auszählen der Antworten und gesondertes Berechnen der Ergebnisse verbunden. Der Z 1013 kann es auch anders. Das vorliegende Programm stellt ein erweiterungsfähiges Beispiel eines Computerfragebogens dar. Ordentliche Nutzerführung, klare Struktur der Fragen und der entsprechenden antworteninternen Zählung und Berechnung der statistischen Werte sowie die Möglichkeit einer externen Speicherung der ermittelten Daten für die spätere Wiederverwendung gehören dazu. Falls unser Z 1013 über eine Speichererweiterung verfügt, kann man statistische Verfahren (z. B. CHI-Quadrat) gleich an das Programm koppeln. Für die konkrete Anwendung des Fragebogenprogramms müssen die "Fragen n" durch Sachfragen ersetzt und entsprechende Antwortmöglichkeiten ausformuliert werden (Listing 25).

```
10 CLS:REM FRAGEBOGEN
20 PRINTAT(12,6);"COMPUTER-FRAGEBOGEN"
30 FOR K=4 TO 26:PRINTAT(10,K);CHR$(42):PRINTAT(14,K);CHR$
   (42):NEXT K
40 FOR K=11 TO 13:PRINTAT(K,4);CHR$(42):PRINTAT(K,26);
   CHR$(42):NEXT K
```

```
50 PRINTAT(30,23);">ENTER<":PRINTAT(0,0);CHR$(32)
60 IF INKEY$<>CHR$(13) THEN 60
100 CLS:REM FELDDIMENSIONIERUNG
110 PRINT:PRINT:PRINT
120 PRINT"VORBEREITUNG DES DATENFELDES":PRINT:PRINT:PRINT
130 PRINT"WIEVIEL FRAGEN ENTHAELT DER":PRINT:PRINT"FRAGE
    BOGEN":PRINT
140 INPUT I
150 PRINT:PRINT
160 PRINT"GROESSTE ANZAHL MOEGLICHER":PRINT:PRINT"ANT
    WORTEN INNERHALB EINER FRAGE"
170 PRINT:INPUT J
180 DIM A(I,J)
200 CLS:REM MENUE-ANGEBOT
210 PRINT:PRINT:PRINT
220 PRINT"DAS PROGRAMM BIETET FOLGENDE":PRINT:PRINT"LEI
    STUNGEN AN :"
230 :PRINT:PRINT:PRINT
240 PRINT"  1.EINLESEN VON DATEN AUS":PRINT
250 PRINT"    FRUEHERER BEFRAGUNG    =1"
260 PRINT:PRINT:PRINT"  2.NEUE BEFRAGUNG          =2"
270 PRINT:PRINT:PRINT"  3.AUSLADEN DER DATEN      =3"
280 PRINT:PRINT:PRINT
290 INPUT"WELCHE VARIANTE ?";L
300 ON L GOTO 400,1000,3000
400 CLS:REM DATEN EINLESEN
410 PRINT:PRINT:PRINT
420 PRINT"D A T E N   E I N L E S E N"
430 PRINT"=========================="
440 PRINT:PRINT:PRINT
450 PRINT"1.KASETTE POSITIONIEREN"
460 PRINT:PRINT:PRINT
470 PRINT"2.REKORDER STARTEN"
480 PRINT:PRINT:PRINT
490 PRINT"3.BEI VORTON  >ENTER<"
500 IF INKEY$<>CHR$(13) THEN500
510 CLOAD*"DATEN";A
520 GOTO 200
1000 CLS:REM BEFRAGUNG
1010 PRINT:PRINT:PRINT
1020 PRINT"DIES IST EIN COMPUTER-FRAGEBOGEN":PRINT:PRINT
1030 PRINT"TIPPE NUR IMMER DIE FUER DICH":PRINT
1040 PRINT"ZUTREFFENDE ANTWORTMOEGLICHKEIT":PRINT
1050 PRINT"IN DEN COMPUTER EIN" :PRINT:PRINT:PRINT
1060 PRINT"DRUECKE DANN DIE ENTER-TASTE":PRINT:PRINT:PRINT
1070 PRINT"STARTE DAS PROGRAMM MIT TASTE S"
1080 IF INKEY$<>CHR$(83) THEN 1080
1090 LET A(0,0)=A(0,0)+1
1100 CLS:PRINT:PRINT"FRAGE 1"
1110 PRINT:PRINT:PRINT
1120 PRINT"  1.ANTWORTMOEGLICHKEIT     =1":PRINT:PRINT
1130 PRINT"  2.ANTWORLMOEGLICHKEIT     =2":PRINT:PRINT
1140 PRINT"  3.ANTWORTMOEGLICHKEIT     =3":PRINT:PRINT
1150 PRINT"  4.ANTWORTMOEGLICHKEIT     =4":PRINT:PRINT:
    PRINT:PRINT
1160 GOTO 1180
1170 PRINT"DIESE ANTWORT GIBT ES NICHT !":PRINT
1180 INPUT"WELCHE ANTWORT TRIFFT ZU ?";N
1190 IF N<1 OR N>4 THEN 1170
1200 ON N GOTO 1210,1220,1230,1240
1210 A(1,1)=A(1,1)+1:GOTO 1300
1220 A(1,2)=A(1,2)+1:GOTO 1300
1230 A(1,3)=A(1,3)+1:GOTO 1300
```

```
1240 A(1,4)=A(1,4)+1:GOTO 1300
1300 CLS:PRINT:PRINT"FRAGE 2"
1310 PRINT:PRINT:PRINT
1320 PRINT"  1.ANTWORTMOEGLICHKEIT     =1":PRINT:PRINT
1330 PRINT"  2.ANTWORTMOEGLICHKEIT     =2":PRINT:PRINT
1340 PRINT"  3.ANTWORTMOEGLICHKEIT     =3":PRINT:PRINT
1350 PRINT"  4.ANTWORTMOEGLICHKEIT     =4~:PRINT:PRINT
     :PRINT:PRINT
1360 GOTO 1300
1370 PRINT"DIESE ANTWORT GIBT ES NICHT !":PRINT
1380 INPUT"WELCHE ANTWORT TRIFFT ZU ?";N
1390 IF N<1 OR N>4 THEN 1370
1400 ON N GOTO 1410,1420,1430,1440
1410 A(2,1)=A(2,1)+1:GOTO 2000
1420 A(2,2)=A(2,2)+1:GOTO 2000
1430 A(2,3)=A(2,3)+1:GOTO 2000
1440 A(2,4)=A(2,4)+1:GOTO 2000
2000 CLS
2010 PRINTAT(12,3);"DANKE FUER DIE MITARBEIT !":PAUSE 50
2020 CLS:PRINT:PRINT:PRINT:PRINT:PRINT
2030 PRINT"WIE GEHT ES WEITER ?":PRINT:PRINT
2040 PRINT"   NEUER PROBAND        =1":PRINT:PRINT
2050 PRINT"   ERGEBNISLISTE        =2":PRINT:PRINT
2060 PRINT"   MENUE ANZEIGEN       =3":PRINT:PRINT:PRINT
2070 INPUT"WELCHE VARIANTE ?";V
2080 IF V<1 OR V>3 THEN 2070
2090 IF V=1 THEN 1000
2100 IF V=2 THEN 2500
2110 IF V=3 THEN 200
2500 CLS:PRINT"E R G E B N I S S E"
2510 PRINT"=================":PRINT:PRINT
2520 PRINT"PROBANDENZAHL =";A(0,0):P=100/A(0,0):PRINT
2530 PRINT"FRAGE 1"
2540 PRINT"  ANTWORT 1 =";A(1,1);TAB(18)A(1,1)*P;"X"
2550 PRINT"  ANTWORT 2 =";A(1,2);TAB(18)A(1,2)*P;"X"
2560 PRINT"  ANTWORT 3 =";A(1,3);TAB(18)A(1,3)*P;"X"
2570 PRINT"  ANTWORT 4 =";A(1,4);TAB(18)A(1,4)*P;"X"
2580 PRINT:PRINT"FRAGE 2"
2590 PRINT"  ANTWORT 1 =";A(2,1);TAB(18)A(2,1)*P;"X"
2600 PRINT"  ANTWORT 2 =";A(2,2);TAB(18)A(2,2)*P;"X"
2610 PRINT"  ANTWORT 3 =";A(2,3);TAB(18)A(2,3)*P;"X"
2620 PRINT"  ANTWORT 4 =";A(2,4);TAB(18)A(2,4)*P;"X"
2900 PRINTAT(30,23);">ENTER<"
2910 IF INKEY$<>CHR$(13) THEN 2910
2920 GOTO 200
3000 CLS:REM DATEN AUSLAGERN
3010 PRINT:PRINT:PRINT
3020 PRINT"D A T E N   A U S L A G E R N"
3030 PRINT"=============================="
3040 PRINT:PRINT:PRINT
3050 PRINT"1.KASSETTE POSITIONIEREN"
3060 PRINT:PRINT:PRINT
3070 PRINT"2.REKORDER AUF AUFNAHME"
3080 PRINT:PRINT:PRINT
3090 PRINT"3.REKORDER STARTEN"
3100 PRINT:PRINT:PRINT
3110 PRINT">ENTER<"
3120 IF INKEY$<>CHR$(13) THEN 3120
3130 CSAVE*"DATEN";A
3140 GOTO 200
```

Listing 25: Computerfragebogen

Zeilen 100 bis 180

Das Datenfeld ist vor der ersten Nutzung des Fragebogens zu dimensionieren. Der einzugebende Wert I entspricht der Anzahl der Fragen im Computerfragebogen. Dem Wert J ist die größte Zahl der vorgesehenen Antwortmöglichkeiten innerhalb einer Frage zugeordnet. Erreicht eine Fragestellung nicht diese höchste Anzahl von Antworten, werden dafür im Feld automatisch die Werte auf Null gesetzt, ohne daß wir dabei etwas zu beachten haben.

Zeilen 200 bis 300

Der angebotene Leistungsumfang wird im Menü dargestellt und besteht aus dem Einladen des Datenfeldes einer vorangegangenen Befragung, der Durchführung und Auswertung einer neuen Befragungsrunde und der Möglichkeit, die Daten nach der Befragung extern auf einer Kassette zu speichern.

Zeilen 400 bis 520

Das Einlesen von Daten wird unter Nutzung der Anweisung CLOAD * "DATEN";A organisiert. Der Dateiname kann anders gewählt werden.

Zeilen 1000 bis 2000

In diesen Zeilen wird der eigentliche Fragebogen untergebracht. Jeder gestellten Frage können beliebig viele Antwortmöglichkeiten zugeordnet werden. Die Höchstgrenze dafür legt der Nutzer durch die Eingabe des Wertes in den Zeilen 100 bis 180 selbst fest. Der gewählten Antwort folgt durch ON N GOTO . . . die Erhöhung des zugeordneten Feldelementes A (I,J) um den Wert 1. Das Computerprogramm wird gegen ein Abstürzen bei falscher Eingabe gesichert. Ein Unterbrechen des Programmablaufes würde zum Verlust aller im Speicher befindlichen Werte führen. Deshalb werden mit Hilfe der Zeilen 1160 bis 1190 u. a. nur solche Werte zugelassen, die im Feld vereinbart wurden.
Die im Beispiel vorgesehenen zwei Frage-Antwort-Komplexe lassen sich auf eine größere Anzahl erweitern. Der vorgesehene Programmablauf ist ab Zeile 1150 analog zu den Zeilen ab 1100 und 1300 aufzustocken.

Zeilen 2500 bis 2920

Die aktuellen Werte der Feldelemente A (1, J) werden den Fragen und ihren Antwortmöglichkeiten zugeordnet und in tabellarischer Anordnung ausgedruckt. Wert 1. Gleichzeitig erfolgt die Berechnung aller zugehörigen Prozentwerte auf der Grundlage der Gesamtzahl der Probanden. Da das Feldelement A (0, 0) für die Frage-Antwort-Kombinationen nicht verwendet wird, dient es als Zähler für die Menge der Versuchspersonen.

Zeilen 3000 bis 3140

Die ermittelten Daten können für eine spätere Weiterverarbeitung oder ein Neueinlesen auf einer Kassette abgespeichert werden. Wir verwenden dazu die Anweisung CSAVE *

"DATEN"; A. Es ist sinnvoll, das ordnungsgemäße Ausladen des Datenfeldes mittels VERIFY zu überprüfen.

Programm: Kurzzeitgedächtnis (842 Bytes)

Mit diesem Programm kann man das Kurzzeitgedächtnis testen und trainieren. Das Kurzzeitgedächtnis gehört ebenso wie das Ultrakurzzeitgedächtnis zu den "flüchtigen" Informationsspeichern des Menschen. Während das Ultrakurzzeitgedächtnis seine Informationen nur Bruchteile von Sekunden behält, kann die Speicherzeit beim Kurzzeitgedächtnis bis zu einigen Minuten betragen. In dieser Zeit sind dann auch die Entscheidungen über ein eventuelles "Einspeichern" in das Langzeitgedächtnis zu treffen.

Der Computer präsentiert auf dem Bildschirm kurzzeitig eine Folge zufällig ausgewählter Buchstaben oder Ziffern. Diese Folge muß anschließend über die Tastatur wieder eingegeben werden. Bei richtiger Eingabe erhöht der Computer bei der folgenden Auswahl die Anzahl der Zeichen um Eins, der Schwierigkeitsgrad erhöht sich also von Schritt zu Schritt. Bei falscher Eingabe wird die richtige Lösung und die Anzahl der richtig gemerkten Zeichen angezeigt. Wer es auf etwa acht korrekt gemerkte Zeichen bringt, kann mit seinem Kurzzeitgedächtnis zufrieden sein. Vielleicht wird sich der eine besser die Buchstabens und der andere besser die Ziffernfolgen merken können. Das Merken von Buchstaben erscheint zunächst schwieriger, da immerhin aus 26 Buchstaben zufällig ausgewählt wird (im Gegensatz zu den Ziffern 0 bis 9.) Andererseits ergeben mit etwas Glück einige Buchstabenkombinationen auch leicht zu merkende "Eselsbrücken" (Listing 26).

```
10 REM KURZZEITGEDAECHTNIS
20 WINDOW:CLS
30 PRINT"TEST DES KURZZEITGEDAECHTNISSES"
40 PRINT STRINK$(31,"-"):PRINT
50 PRINT TAB(5):"TEST MIT":PRINT:PRINT
60 PRINT TAB(5);"BUCHSTABEN= B":PRINT
70 PRINT TAB(5);"     ZAHLEN= N"
80 LET IN$=INKEY$:IF IN$<>"B" AND IN$<>"N"
   THEN GOTO80:ELSE CLS
90 FOR I=6 TO 26:PRINT AT(4,I);CHR$(199):PRINT AT(8,I);
   CHR$(199):NEXT I
100 FOR I=5 TO 7:PRINT AT(I,6);CHR$(199):PRINT AT(I,26);
    CHR$(199):NEXT I
110 WINDOW 5,7,7,25:CLS
120 PRINT AT(6,9);"ACHTUNG":PAUSE 5
130 IF IN$="B" THEN LET A=26:LET B=65:ELSE LET A=10:
    LET B=48
140 LET Z=1:CLS
150 LET A$=""
160 FOR I=1 TO Z
170 LET B$=CHR$(INT(B+RND(1)*A))
180 PRINT AT(6,I+8);" "+B$
190 LET A$=A$+B$:PAUSE 9
200 NEXT I
210 PRINT AT(6,I+8);"  "
230 PRINT"GEBEN SIE DIE RICHTIGE"
240 INPUT"ZEICHENFOLGE EIN:";AN$
250 IF AN$=A$ THEN CLS:LET Z=Z+1:GOTO150
260 PRINT:PRINT"SIE HABEN FALSCH BEOBACHTET!"
270 PRINT"ICH ZEIGTE IHNEN:";A$
280 PRINT:PRINT"SIE MERKTEN SICH":PRINT Z-1;"ZEICHEN
    RICHTIG!"
290 PRINT-"WOLLEN SIE WEITER UEBEN (J/N)?";J$
300 IF J$="J" THEN GOTO20
310 WINDOW:CLS:END
```

Listing 26: Training des Kurzzeitgedächtnisses

Zeilen 10 bis 80

Die Programmüberschrift und das Menü werden auf dem Bildschirm dargestellt. Die Unterstreichung der Überschrift erfolgt mit der Anweisung STRING$. In Zeile 80 wird mit Hilfe der INKEY$-Anweisung die Eingabe des Buchstaben B für die Arbeit mit Buchstaben oder des Buchstaben N für die Arbeit mit Ziffern realisiert. Der Buchstabe Z ist über die Flachfolientastatur nur durch zusätzliches Drücken der Taste S1 eingebbar.

Zeilen 90 bis 140

Zunächst wird mit Hilfe des Pseudografikzeichens mit dem Code 199 ein kleiner Bildrahmen gesetzt, in dem dann die Anzeige der Zeichen erfolgen soll. In diesem Rahmen erscheint zunächst das Wort ACHTUNG etwa eine halbe Sekunde lang. Die Variable mit dem Namen B enthält den Anfangswert der ASCII-Zeichencodetabelle, bei der die Auswahl von Zeichen beginnen soll (A = Code 65, 0 = Code 48). Die Variable mit dem Namen A enthält die Anzahl der in Frage kommenden Zeichen. Die Variable Z repräsentiert den Zeichenzähler, der, bei eins beginnend, schrittweise erhöht wird.

Zeilen 150 bis 210

Hier erfolgen zufällige Auswahl und Anzeige der Buchstaben- oder Ziffernkombinationen. In Zeile 190 ist die Anzeigezeit mit 3/10 Sekunden festgelegt. Diese Anzeigezeit kann beliebig erhöht oder verkürzt werden. Es ist auch möglich, das Programm so umzubauen, daß über die Anzeigezeit zum Programmbeginn eine wählbare Schwierigkeitsstufe eingegeben werden kann.

Zeilen 220 bis 250

Die von der "Testperson" eingegebene Zeichenfolge wird der Variablen AN$ zugeordnet. Diese Zeichenfolge wird in Zeile 250 mit der vom Computer erzeugten Folge verglichen. Stimmen beide Folgen überein, dann wird der Zeichenzähler um Eins erhöht und der Programmablauf ab Zeile 150 wiederholt.

Zeilen 260 bis 310

Diese Programmzeilen werden dann ausgeführt, wenn eine falsche Buchstabenkombination eingegeben wurde. Mit der Anzeige der richtigen Folge und der insgesamt richtig gemerkten Zeichen wird der Programmablauf beendet.

Der Spielmeister Z 1013

Programm: Schiebepuzzle (1811 Bytes)

Der ungarische Architekt E. Rubik hatte vor einigen Jahren die Idee, ein flächiges Schiebepuzzle durch ein räumliches zu ersetzen. Von diesem Zauberwürfel wurden in fünf Jahren rund 100 Millionen Stück gefertigt. Es gibt auch Computerprogramme, die einen Zauberwürfel auf dem Bildschirm darstellen und sogar seine korrekte Ordnung vornehmen. Da dies aber ziemlich kompliziert ist, gehen wir mit unserem Z 1013 wieder auf die Fläche zurück und stellen ein Schiebepuzzle vor, das aus fünf mal fünf Feldern besteht. Auf diese Felder werden bei Spielbeginn in Ermangelung von Farbe fünf verschiedene Pseudografikzeichen verteilt. Jedes der fünf Zeichen belegt also fünf zufällig ausgewählte Felder (Bild 21). Durch zyklisches Verschieben der Pseudografikzeichen in den Zeilen und Spalten

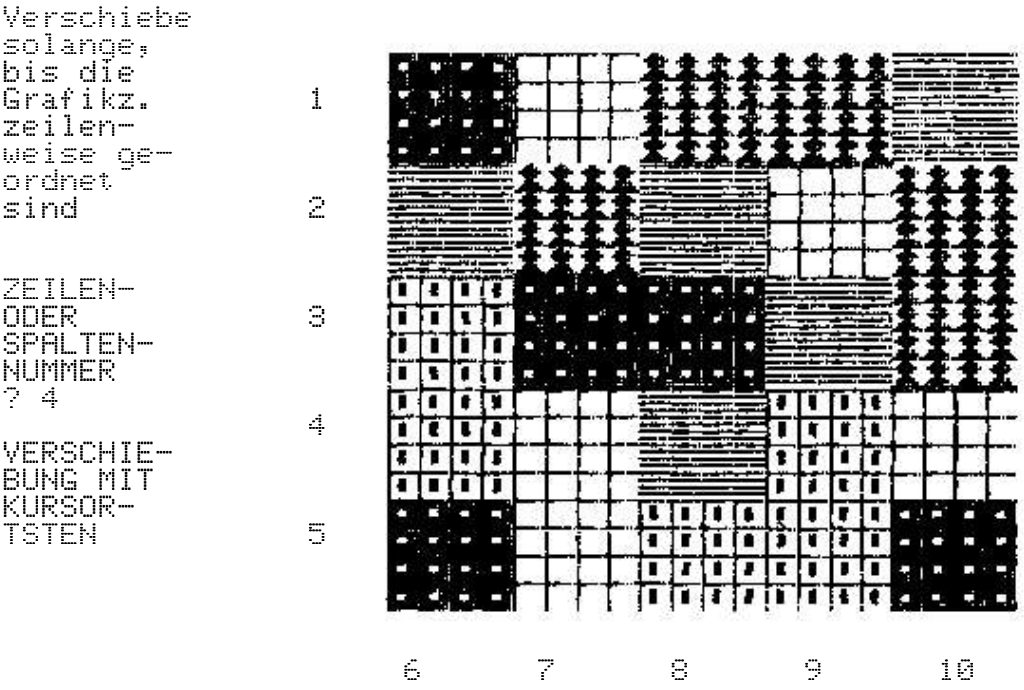


Bild 21: Zufällige Startsituation für das Schiebepuzzle

soll erreicht werden, daß im Endergebnis fünf Zeilen mit jeweils gleichen Zeichen vorliegen. Im Programm wird die Anzahl der Verschiebungen bis zur Lösung gezählt, so daß damit auch eine Wettspielsituation gegeben ist (Listing 27).

```
10 REM SCHIEBEPUZZLE
20 WINDOW:CLS
30 REM ZUFAELLIGE ANF.ZAHL
40 PRINT"ZUM PROGRAMMSTART BELIEBIGE"
50 PRINT"TASTE DRUECKEN!":LET I=0
60 LET I=1:LET A=RND(I)
70 IF INKEY$="" THEN GOTO60:ELSE CLS
80 DIM GF(6,6):DEF FN ZU(I)=INT(RND(I)*5+1):LET AN=0
90 REM ZUFAELLIGE GRAFIKVERTEILUNG
100 FOR I=1 TO 5
110 FOR J=1 TO 5
120 LET GR=J+192:GOSUB 1000
130 NEXT J,I
140 REM ZEILEN-UND SPALTENNUMMERN
150 LET J=1
```

```
160 FOR I=3 TO 19 STEP 4
170 PRINT AT(I,9);J:LET J=J+1
180 NEXT I
190 FOR I=12 TO 28 STEP 4
200 PRINT AT(23,I);J:LET J=J+1
210 NEXT I
220 WINDOW 0,9,0,9
230 PRINT"Verschiebe";:PRINT"solange,":PRINT"bis die":
    PRINT"Grafikz."
240 PRINT"zeilen-":PRINT"weise ge-":PRINT"ordnet":PRINT
    "sind !"
250 WINDOW 10,31,0,9
260 REM GRAFIKFELD
270 FOR Z=1 TO 5
280 LET P=Z*4-2
290 FOR I=0 TO 3
300 LET J=1
310 FOR S=12 TO 28 STEP 4
320 LET MU=GF(Z,J)
330 PRINT AT(P+I,S);CHR$(MU)+CHR$(MU)+CHR$(MU)+CHR$(MU)
340 LET J=J+1
350 NEXT S,I,Z
360 REM ENDEABFRAGE
370 FOR I=1 TO 5
380 FOR J=2 TO 5
390 IF GF(I,1)<>GF(I,J) THEN LET AN=AN+1:GOTO460
400 NEXT J,I
410 WINDOW 0,31,0,9:CLS
420 PRINT"GESCHAFFT":PRINT"IN";AN:PRINT"SCHRITTEN!"
430 PRINT:PRINT"NOCH":PRINT"EINMAL":INPUT"(J/N)?";A$
440 WINDOW:CLS
450 IF A$="J" THEN CLEAR:GOTO80:ELSE END
460 PRINT"ZEILEN-":PRINT"ODER":PRINT"SPALTEN-":
    PRINT"NUMMER"
470 INPUT NR
480 IF NR<1 OR NR>10 THEN CLS:GOTO460
490 PRINT:PRINT"VERSCHIE-"
500 PRINT"BUNG MIT":PRINT"KURSOR-":PRINT"TASTEN"
510 IF NR>5 THEN LET NR=NR-5:GOTO560
520 LET KU$=INKEY$
530 IF KU$=CHR$(9) THEN CLS:GOSUB 1100:GOTO270
540 IF KU$=CHR$(8) THEN CLS:GOSUB 1200:GOTO270
550 GOTO520
560 LET KU$=INKEY$
570 IF KU$=CHR$(10) THEN CLS:GOSUB 1300:GOTO270
580 IF KU$=CHR$(11) THEN CLS:GOSUB 1400:GOTO270
590 GOTO560
1000 REM UP ZUFALLSZAHLEN
1010 LET Z=FN ZU(1):LET S=FN ZU(1)
1020 IF GF(Z,S)<>0 THEN GOTO 1010
1030 LET GF(Z,S)=GR
1040 RETURN
1100 REM UP NACH RECHTS
1110 FOR I=6 TO 2 STEP -1
1120 LET GF(NR,I)=GF(NR,I-1)
1130 NEXT I
1140 LET GF(NR,1)=GF(NR,6)
1150 RETURN
1200 REM UP NACH LINKS
1210 LET GF(NR,6)=GF(NR,1)
1220 FOR I=1 TO 5
1230 LET GF(NR,I)=GF(NR,I+1)
1240 NEXT I
```

```
1250 RETURN
1300 REM UP - UNTEN
1310 FOR I=6 TO 2 STEP -1
1320 LET GF(I,NR)=GF(I-1,NR)
1330 NEXT I
1340 LET GF(1,NR)=GF(6,NR)
1350 RETURN
1400 REM UP NACH OBEN
1410 LET GF(6,NR)=GF(1,NR)
1420 FOR I=1 TO 5
1430 LET GF(I,NR)=GF(I+1,NR)
1440 NEXT I
1450 RETURN
```

Listing 27: Schiebepuzzle

Zeilen 10 bis 80

Die Zeilen 40 bis 70 realisieren in Ermangelung der RANDOMIZE-Anweisung eine zufällige Anfangszahl für die Reihe der Pseudozufallszahlen. Anschließend wird das Spielfeld dimensioniert, wobei ein sechstes Feld (beim Z 1013 hätten wir auch das Feld 0 nehmen können) als Puffer für die zyklische Vertauschung dient. Die Definition einer eigenen Funktion zur Erzeugung der Pseudozufallszahlen ist nicht zwingend erforderlich, wir wollten hier nur das Prinzip zeigen (was obendrein mehr Rechenzeit kostet).

Zeilen 90 bis 250

Die erforderlichen Informationen werden auf den Bildschirm gebracht. Zur Erzeugung des Feldes der zufällig verteilten Pseudografikzeichen wird das Unterprogramm ab Zeile 1000 genutzt. dieses Unterprogramm sorgt auch dafür, daß alle fünf Zeichen genau fünfmal vorkommen.

Zeilen 260 bis 350

Nach jedem Zug erfolgt der Programmdurchlauf ab Zeile 270. In Zeile 320 wird der aktuelle Wert des im Computer gespeicherten Spielfeldes GF (Z, J) der Variablen MU zugewiesen. Das Setzen des Pseudografikzeichens mit der Codezahl MU auf den Bildschirm erfolgt in Zeile 330 mit der PRINT AT-Anweisung. Zur Vervielfachung des Zeichens werden geschachtelte Laufanweisungen genutzt.

Zeilen 360 bis 450

Es wird geprüft, ob jede Zeile jeweils gleiche Pseudografikzeichen enthält. Wer auch eine spaltenweise Gleichheit zulassen möchte, müßte das Programm, einschließlich erklärendem Text, entsprechend ändern. Die Variable AN zählt die Versuche. Ist die Endebedingung noch nicht erreicht, wird in Zeile 460 fortgesetzt.

Zeilen 460 bis 590

Im aktuellen Bildschirmfenster erfolgt die Eingabe der Zeilen- oder Spaltennummer und die gewünschte Verschiebungsrichtung. Die Zeile 480 sorgt dafür, daß nur Eingaben für die Zeilennummern 1 bis 5 und die Spaltennummern 6 bis 10 akzeptiert werden. Bei Eingabe einer Zeilennummer wird in einer Schleife zwischen den Zeilen 520 und 550 die Abfrage der Tastatur über die INKEY$-Anweisung durchgeführt. Damit werden automatisch nur waagerechte Verschiebungen (Kode 9 = Kursor nach rechts, Kode 8 = Kursor nach links) zugelassen. Das gleiche gilt sinngemäß für die spaltenweise Verschiebung. Leider besitzt die Flachfolientastatur des Z 1013 nicht die Tasten "Kursor nach oben" und "Kursor nach unten". Die Kursorbewegung nach unten wird durch die Kodezahl 10 realisiert. Dazu sind auf der Flachfolientastatur die Tasten S4 und R zu drücken. Für die Kursorbewegung nach oben gilt die Kodezahl 11. Das entspricht den Tasten S4 und S. Bei ausschließlicher Verwendung der Flachfolientastatur kann natürlich auch eine andere Tastenbelegung im Programm vereinbar werden.

Zeilen 1100 bis 1450

Für jede der vier möglichen Verschiebungsrichtungen existiert ein Unterprogramm. Jedes Unterprogramm aktualisiert das Spielfeld GF (6,6) gemäß der erfolgten Eingabe. Die Darstellung auf dem Bildschirm wird aber nicht hier, sondern im Hauptprogramm vorgenommen.

Programm: CAD-Spiel (4537 Bytes)

Ingenieure, Technologen und technische Zeichner nutzen heute in großem Umfang die Fähigkeiten von Computern zum effektiven Konstruieren und Projektieren. CAD (Computer aided design) hilft dem Konstrukteur, schneller, präziser und kostengünstiger zu optimalen Arbeitsergebnissen zu gelangen. Das gelingt vor allem deshalb, weil dem Computer ingenieurtechnische Routineaufgaben übertragen werden, die er mit hoher Zuverlässigkeit erfüllt. Wesentlicher Bestandteil einer CAD-Konstruktion ist die Möglichkeit, genormte Konstruktionselemente aus dem Computerspeicher abzurufen und in das zu bearbeitende Projekt einzufügen.

Die Idee unseres CAD·Spieles besteht darin, im Programm für den Z 1013 wesentliche Bauelemente zu speichern, die wahlweise zum Bau eines Hauses zusammengefügt werden können (Bild 221. Dieses kleine Spiel ermöglicht dem Hobbykonstrukteur bereits 320 Varianten für die Gestaltung der Bauzeichnung. Mit geringem Aufwand können weitere Ergänzungen oder Abänderungen vorgenommen werden. Knobellustige Z 1013-Fans können sich ein ähnliches CAD-SpieI zum Bau eines Schiffes, eines Autos oder einer Werkzeugmaschine ausdenken. Auch Entwürfe von elektronischen Schaltungen oder Planentwürfe für die Kleingartengestaltung wären möglich.

Zunächst sind wesentliche, variierbare Elemente des Projektes zu bestimmen. Bei unserem Haus sind das Etagenzahl, Dach-, Fenster- und Türformen sowie mögliches Nebengelaß.

Mit Hilfe unserer 32 x 32 Rasterfolie und der verfügbaren Grafikzeichen "bauen" wir jedes Teil für sich zusammen und legen es in einem Unterprogramm ab. Zur besseren Erkennbarkeit wird dabei durchgängig mit der PRINT AT-Anweisung gearbeitet und auf das speicherplatzsparende POKE verzichtet.

Die Konstruktion wird über die Menümöglichkeiten des Hauptprogramms so geführt, daß über entsprechende Aufrufe der Unterprogramme Schritt für Schritt eine Komplettierung des Bauwerkes vorgenommen wird (Listing 28).

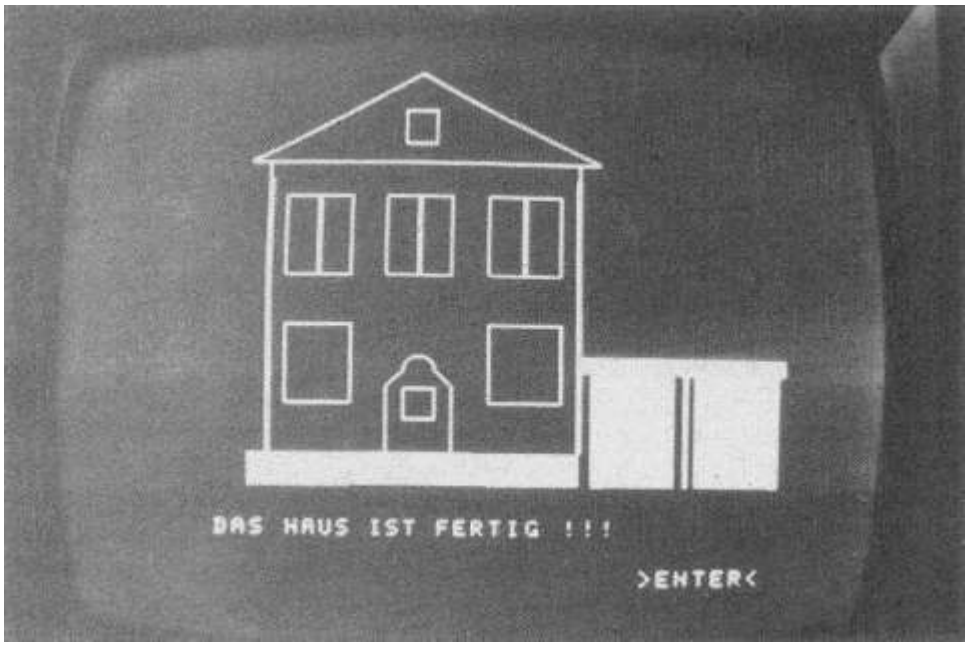


Bild 22: Beispiel zum CAD-Spiel

```
10 WINDOW:CLS:REM CAD-SPIEL
20 PRINTAT(12,7);"C A D - S P I E L"
30 FOR N=5TO25:PRINTAT(10,N);CHR$(42):PRINTAT(14,N);
   CHR$(42):NEXT N
40 FOR N=10TO13:PRINrAT(N,5);CHR$(42):PRINTAT(N,25);
   CHR$(42):NEXT N
50 PRINTAT(29,24);">ENTER<":PRINTAT(0,0);CHRS(32)
60 IF INKEY$<>(13)THEN 60
100 WINDOW:CLS:PRINT:PRINT
110 PRINT"WIR BAUEN EIN HAUS NACH DEM":PRINT:PRINT"BAU
   KASTENPRINZIP"
115 PRINT:PRINT:PRINT:PRINT
120 PRINT"DIE GRUNDMAUER IST VORHANDEN.":PRINT:PRINT:PRINT
130 PRINT"ANZAHL DER ETAGEN,DACHFORM,":PRINT:PRINT"GROESSE
    UND ANZAHL DER"
135 PRINT:PRINT"FENSTER,TUERFORM UND":PRINT
140 PRINT"NEBENGELASS WAEHLEN WIR SELBST."
150 PRINTAT(29,24);">ENTER<":PRINTAT(20,0);CHR$(32)
160 IF INKEY$<>CHR$(13)THEN 160
200 CLS:WINDOW26,31,0,31:GOSUB1000
210 PRINT"WAEHLE DIE ETAGENZAHL!"
220 PRINT"     EINE ETAGE  = 1":PRINT"     ZWEI ETA
   GEN = 2"
230 INPUT"EINGABE DER ZAHL:";E
240 IF E=1 THEN ZE=14:GOSUB 2000:GOTO300
250 IF E=2 THEN ZE=5:GOSUB2000:ZE=14:GOSUB2000:ZE=5:
   GOTO300
300 CLS:PRINT"WAEHLE DIE DACHART!"
310 PRINT"     FLACHDACH = 1":PRINT"     SPITZDACH = 2"
320 INPUT"EINGABE DER ZAHL:";D
330 IF D=1 THEN GOSUB3000:ELSE GOSUB 4000
400 CLS:PRINT"WAEHLE DIE HAUSTUER!"
410 PRINT"     RECHTECKIGE TUER = 1":PRINT"     RUNDBOGEN
   TUER    = 2"
420 INPUT"EINGABE DER ZAHL:";T
430 IF T=1 THEN GOSUB 5000:ELSE GOSUB 6000
500 CLS:PRINT"FENSTER FUER HAUSTUER ?"
510 PRINT"     JA   = 1":PRINT"     NEIN = 2"
```

```
520 INPUT"EINGABE DER ZAHL:";TF
530 IF TF=1 THEN540:ELSE 600
540 X=19:Y=11:GOSUB7000
600 CLS : PRINT"FENSTER FUER ERDGESCHOSS?"
610 PRINT"      NORMALE FENSTERGROESSE = 1"
620 PRINT"      FENSTER GETEILT        = 2"
630 INPUT"EINGABE DER ZAHL:";FE
640 IF FE=1 THEN 650:ELSE670
650 X=15:Y=4:GOSUB 8000
660 X=15:Y=16:GOSUB 8000:GOTO 700
670 X=15:Y=4:GOSUB 9000
680 X=15:Y=16:GOSUB 9000
700 IF ZE=5 THEN 710:ELSE 900
710 CLS:PRINT"WAEHLE FENSTER FUER OBERETAGE!"
720 PRINT"      NORMALE FENSTER     = 1"
730 PRINT"      FENSTER GETEILT     = 2"
740 PRINT"      GROSSE FENSTERFRONT = 3"
750 INPUT"EINGABE DER ZAHL:";FO
760 IF FO=1 THEN 790:ELSE 770
770 IF FO=2 THEN 820:ELSE 780
780 IF FO=3 THEN 850
790 X=7:Y=4:GOSUB 8000
800 X=7:Y=10:GOSUB 8000
810 X=7:Y=16:GOSUB 8000:GOTO 900
820 X=7:Y=4:GOSUB 9000
830 X=7:Y=10:GOSUB 9000
840 X=7:Y=16:GOSUB 9000:GOTO 900
850 GOSUB 10000
900 CLS:PRINT"ANBAU NEBEN DEM HAUS ?"
910 PRINT"      GARAGE = 1" :PRINT"      MAUER  = 2"
920 INPUT"EINGABE DER ZAHL:":AN
930 IF AN=1 THEN GOSUB 11000:ELSE GOSUB 12000
940 CLS:PRINT:PRINT"DAS HAUS IST FERTIG !!!":PRINT:PRINT
950 PRINTAT(30,23);">ENTER<":PRINTAT(30,0);CHR$(32)
955 IF INKEY$<>CHR$(13) THEN 955
960 CLS:PRINT"SPIEL WIEDERHOLEN ? (J/N)":PRINT
970 INPUT S$
980 IF S$="J" THEN 100
990 WINDOW:CLS:END
1000 REM GRUNDMAUER
1010 FOR K=0TO1:FOR N=0TO19:PRINTAT(24-K,2+N);CHR$(198):
     NEXT N:NEXT K
1020 FOR N=0TO7:PRINTAT(24,8+N);CHR$(197):NEXT N
1030 FOR N=0TO5:PRINTAT(23,9+N);CHR$(197):NEXT N:RETURN
2000 REM SEITENWAENDE/REM ZE=ZEILENNUMMER DER ETAGE
2010 FOR N=0TO8
2020 PRINTAT(ZE+N,3);CHR$(232)
2030 PRINTAT(ZE+N,20);CHR$(244)
2040 NEXT N
2050 RETURN
3000 REM FLACHDACH/ZE=ZEILENNUMMER DER DACHEBENE
3010 ZD=ZE-1:FOR N=0TO21:PRINTAT(ZD,1+N);CHR$(255):NEXT N
3020 RETURN
4000 REM SPITZDACH/REM ZE=ZEILENNUMMER DER ETAGE
4010 FOR N=0TO15
4020 PRINTAT(ZE,4+N);CHR$(158)
4030 NEXT N
4040 PRINTAT(ZE,2);CHR$(158)
4050 PRINTAT(ZE,21);CHR$(158)
4060 PRINTAT(ZE,3);CHR$(198)
4070 PRINTAT(ZE,20);CHR$(187)
4080 RESTORE4120
```

```
4090 FOR N=0TO19:READ A,Z
4100 PRINTAT(ZE-A,);CHR$(146+Z)
4110 NEXT N
4120 DATA 1,0,1,1,2,0,2,1,3,0,3,1,4,0,4,1,5,0,5,1
4130 DATA 5,4,5,3,4,4,4,3,3,4,3,3,2,4,2,3,1,4,1,3
4140 CLS:PRINT"FENSTER IM DACHGESCHOSS ?"
4150 PRINT"      JA    = 1":PRINT"      NEIN = 2"
4160 INPUT"EINGABE DER ZAHL"LFD
4170 IF FD=1 THEN 4180:ELSE4190
4180 X=ZE-3:Y=11:GOSUB 7000
4190 RETURN
5000 RESTORE 5030: REM RECHTECKIGE TUER
5010 FOR N=1TO12:READX,Y,Z
5020 PRINTAT(X,Y);CHR$(Z):NEXT N
5030 DATA 18,10,193,18,13,137,18,11,158,18,12,158,19,10,
     159,20,10,159,21,10
5040 DATA 159,22,10,159,19,13,192,20,13,192,21,13,192,22,
     13,192
5050 RETURN
6000 RESTORE 6030: REM TUER MIT RUNDBOGEN
6010 FOR N=1TO12:READX,Y,Z
6020 PRINTAT(X,Y);CHR$(Z):NEXTN
6030 DATA 22,10,159,21,10,159,20,10,159,19,10,159,18,10,
     144,17,11,174,17
6040 DATA 12,173,18,13,145,19,13,192,20,13,192,21,13,192,
     22,13,192
6050 RETURN
7000 RESTORE 7030:REM FENSTER IN TUER/DACHGESCHOSS
7010 FOR N=1TO4:READA,B,Z
7020 PRINTAT(X+A,Y+B);CHRS(Z):NEXT N
7030 DATA0,0,193,0,1,137,1,0,136,1,1,200
7040 RETURN
8000 RESTORE 8030:REM NORMALE FENSTER/P(X,Y>OBERER LINKER
      ECKPUNKT
8010 FOR N=0TO13:READA,B,Z
8020 PRINTAT(X+A,Y+B);CHR$(Z) :NEXTN
8030 DATA 0,0,193,0,1,158,0,2,158,0,3,137,1,0,159,1,3,192,
     2,0,159,2,3,192
8040 DATA 3,0,159,3,3,192,4,0,136,4,1,248,4,2,248,4,3,200
8050 RETURN
9000 RESTORE8030:REM NORMALES FENSTER,GETEILT
9010 FOR N=0TO23:READA,B,Z
9020 PRINTAT(X+A,Y+B);CHR$(Z):NEXT N
9030 DATA 0,1,137,0,2,193,1,1,192,1,2,159,2,1,192,2,2,159
9040 DATA 3,1,192,3,2,159,4,1,200,4,2,136
9050 RETURN
10000 REM GROSSE FENSTERFRONT(NUR FUER OBERE ETAGE
10010 FOR N=0TO9:PRINTAT(7,7+N);CHR$(158)
10020 PRINTAT(11,7+N);CHRS(248):NEXT N
10030 PRINTAT(7,6);CHR$(193):PRINTAT(11,6);CHR$(136)
10040 PRINTAT(7,17);CHR$$(137):PRINTAT(11,17);CHR$(200)
10050 FOR N=0TO2:PRINTAT(8+N,6);CHR$(159):PRINTAT(8+N,17);
      CHR$(192):NEXT N
10060 RETURN
11000 REM GARAGE
11010 FOR N=0TO6:PRINTAT(24-N,31);CHR$(180):PRINTAT(24-N,
      21);CHR$(181):NEXT N
11020 FOR N=0TO10:PRINTAT(17,21+N);CHR$(255):NEXT N
11030 FOR K=0TO6:FOR N=0TO8:PRINTAT(24-K,22+N);CHR$(190):
      NEXT N:NEXT K
11040 FOR K=0TO6:PRINTAT(18+K,26);CHR$(161):NEXT K
11050 RETURN
```

```
12000 REM MAUER
12010 FOR K=0TO4:FOR N=0TO10
12020 PRINTAT(24-K,21+N);CHR$(184):NEXT N:NEXT K
12030 FOR N=0TO10:PRINTAT(19,21+N);CHR$(157):NEXT N
12040 RETURN
```

Listing 28: CAD-Spiel

Zeilen 100 bis 160

Zur Nutzerführung wird eine knappe Darstellung der Möglichkeiten des Spielprogrammes gegeben.

Zeilen 200 bis 260

Die Grundmauer wird über das Unterprogramm ab Zeile 1000 fest vorgegeben. Der Baumeister hat die Gelegenheit, zwischen einem ein- oder zweigeschossigen Haus zu wählen. Nach der Wahl ordnet das Programm jeder Etage eine entsprechende Zeilennummer ZE für die obere Begrenzung des Hausgeschosses zu. Diese Zeilennummer bestimmt im Unterprogramm die Position, auf der mit dem Zeichnen der Seitenwände begonnen wird. Bei zweigeschossiger Bauweise wird das Unterprogramm doppelt aufs gerufen. Der Aus- und Rücktausch des Wertes für ZE macht sich erforderlich, weil auf diese Werte beim Bau des Daches noch einmal zurückgegriffen wird.

Zeilen 300 bis 970

Analog zur Vorgehensweise der Zeilen 200 bis 260 erfolgt sukzessive die Fortführung des Baugeschehens durch entsprechende Auswahl der Dachform, der Haustür, des Fensters in der Haustür, der Fenster der ersten Etage, der Fenster der zweiten Etage sowie eines Anbaues neben dem Haus. Für die Fenster ist jeweils die Angabe eines Punktes P (x, y) erforderlich. Dieser Punkt bestimmt die Lage von Zeile und Spalte der linken oberen Ecke des zu zeichnenden Fensterelementes.

Zeilen 1000 bis 12040

Immer ein Bauelement wird übersichtlich in einem Unterprogramm abgelagert. In jedem Fall beginnt ein neues Unterprogramm mit einer Tausender Zeilennummer.
Der Aufbau der Konstruktionselemente erfolgt durch die Kombination von Bildschirmplätzen und Grafikzeichen, die mittels entsprechender Programmschleifen aneinandergereiht werden. Bei umfangreicheren Konstruktionen werden Zeile, Spalte und Grafikcode in DATA-Anweisungen abgelegt und durch READ eingelesen.

Programm: Flächengestaltung (1052 Bytes)

Mit dem CAD-Spiel konnten Häuser nach Wunsch auf dem Bildschirm "gebaut" werden. Jetzt soll der Z 1013 dabei helfen, die Muster für Tapeten, Vorhänge und Fußbodenbeläge zu gestalten. Allerdings sind wir auf die im Mikrorechnerbausatz vorhandenen Pseudografikzeichen angewiesen, da es leider nicht möglich ist, eigene Zeichen zu definieren. Unser Programm arbeitet nach dem Prinzip des Kartoffeldruckes. In einem Menü werden acht verschiedene Grundmuster aus dem Grafiksatz des Z 1013 zur Auswahl angeboten. Durch kleine Änderungen im Programm können natürlich auch andere

Pseudografikzeichen gewählt werden. Ein einzelnes Grundmuster viermal (2 Zeilen, 2 Spalten) dargestellt, um eine günstige Bildschirmgestaltung zu erreichen. Das Menü in Bild 23 macht dies deutlich.

Bild 23 zeigt auch, daß der Bildschirm zwei Funktionen übernehmen muß. Er ist Druckfläche und zugleich Anzeigeeinrichtung für den Dialog zwischen Nutzer und Computer. Die Druckfläche umfaßt die Bildschirmspalten 0 bis 23 und die Bildschirmzeilen 0 bis 31. Die Fläche ist in 12 kleinere Flächen unterteilt, die alle den gleichen Inhalt haben. Der Nutzer

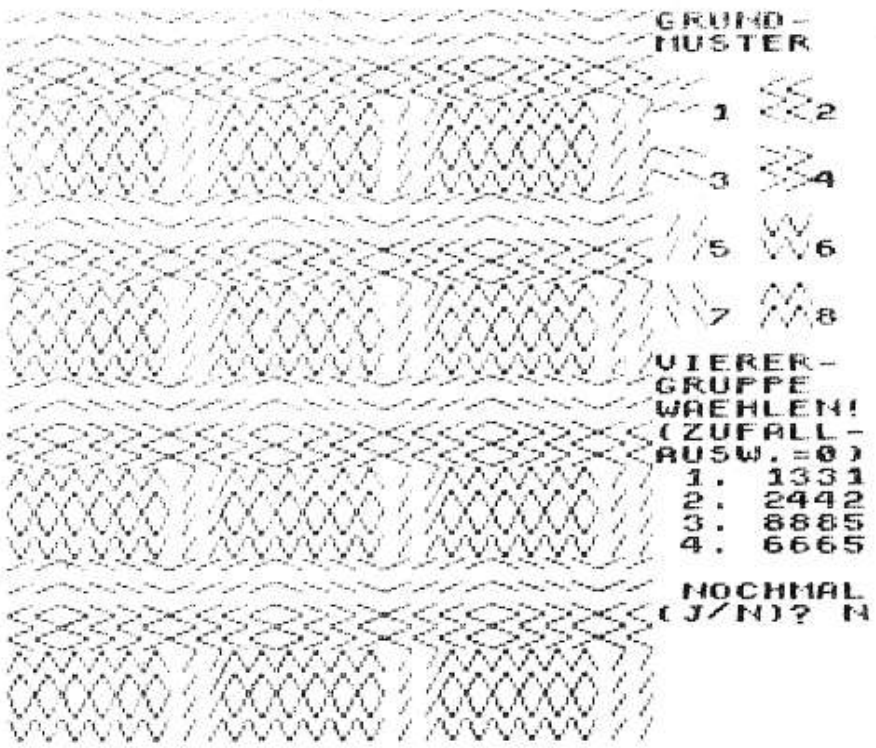


Bild 23: Beispiel zum Programm Flächengestaltung

legt mit seinen Eingaben den Inhalt eines dieser zwölf Felder fest. Dieses eine Feld wiederum umfaßt vier mal vier wählbare Grundmuster. Die gewünschte Anordnung wird über die Kennzahlen 1 bis 8 in den Z 10l3 eingegeben. Mit etwas Überlegung lassen sich so recht hübsche Flächen gestalten. Möchte sich jemand einer optischen Täuschung hingeben, dann sollte er folgende Anordnung probieren: 1. Reihe = 1111, 2. Reihe = 2222, 3. Reihe = 3333 und 4. Reihe = 8888.

Wer die Unregelmäßigkeit liebt, kann dem Z 1013 das "Erwürfeln" von Mustern übertragen. Mit Hilfe des Zufallszahlengenerators entstehen zufällig und manchmal auch ansprechende Flächenmuster. Doch nun zum Listing 29.

```
10 REM FLAECHENGESTALTUNG
20 REM VEREINBARUNGEN
30 DIM FE(4,4)
40 WINDOW:CLS
50 REM BEDIENFENSTER
60 WINDOW 0,31,24,31:CLS
70 PRINT"GRUND-":PRINT"MUSTER"
80 LET I=48
90 FOR Z=3 TO 12 STEP 3
100 LET I=I+1:LET IM=I*3
110 PRINT:PRINT CHR$(IM);CHR$(IM);"  ";CHR$(IM+1);
    CHR$(IM+1)
```

```
120 PRINT CHR$(IM);CHR$(IM);RIGHT$(STR$(I*2-97),1);" ";
130 PRINT CHR$(IM+1);CHR$(IM+1);RIGHT$(STR$(I*2-96),1)
140 NEXT Z
150 PRINT:PRINT"VIERER-":PRINT"GRUPPE":PRINT"WAEHLEN!";
160 PRINT"(ZUFALL-";:PRINT"AUSW.=0)";
170 REM EINGABEN
180 FOR I=1 TO 4
190 PRINT" ";RIGHT$(STR$(I),1);:INPUT".";I4$:POKE 113,
    PEEK(113)-1
200 IF I4$="0" THEN GOSUB1000:GOTO280
210 IF LEN(I4$)<>4 THEN GOTO60
220 FOR J=1 TO 4
230 LET IN=VAL(MID$(I4$,J,1))
240 IF IN<1 OR IN>8 THEN GOTO60
250 LET FE(I,J)=(INT(IN/2+.5)+48)*8-(IN-INT(IN/2)*2)+1
260 NEXT J
270 NEXT I
280 REM DARSTELLUNG NACH VORGABE
290 FOR ZB=0 TO 24 STEP 8
300 FOR SB=0 TO 16 STEP 8
310 LET I=0:LET J=0
320 FOR ZE=ZB TO ZB+6 STEP 2
330 LET I=I+1:LET J=0
340 FOR SP=SB TO SB+6 STEP 2
350 LET J=J+1
360 FOR Z=ZE TO ZE+1:FOR S=SP TO SP+1
370 PRINT AT(Z,S);CHR$(FE(I,J))
380 NEXT S,Z,SP,ZE,SB,ZB
390 PRINT:PRINT" NOCHMAL";:INPUT"(J/N)?";A$
400 WINDOW:CLS
410 IF A$="J" THEN GOTO60:ELSE END
1000 REM UP ZUFAELLIGE VERTEILUNG
1010 FOR I=1 TO 4
1020 FOR J=1 TO 4
1030 LET FE(I,J)=INT(RND(1)*4+49)*8+INT(RND(1)*2)
1040 NEXT J,I
1050 RETURN
```

Listing 29: Flächengestaltung

Zeilen 10 bis 160

Das Feld FE (4, 4), das in Zeile 30 dimensioniert wird, nimmt die Codezahlen der 16 durch den Nutzer zu wählenden Pseudografikzeichen auf. Nach Festlegung des Menüfensters werden in den Zeilen 90 bis 140 die acht möglichen Grundmuster angeboten. Da es hier etwas eng zugeht, mußte mit der RIGHT$-Funktion in Zeile 130 eine Vorzeichenstelle beseitigt werden. Soll das Programm mit anderen Pseudografikzeichen benutzt werden, dann müssen in den Zeilen 80, 100, 250 und 1030 entsprechende Änderungen vorgenommen werden; eine schöne Übung für den Anfänger.

Zeilen 170 bis 270

Hier werden die Eingaben für das 4 mal 4-Feld realisiert. Wegen der Kleinheit des Menüfensters mußte mit der RIGHT$-Funktion in Zeile 190 wieder eine Vorzeichenstelle eliminiert werden. Die Anweisungen POKE und PEEK organisieren ein korrektes Setzen des Kursors. Bei Eingabe einer Null wird die Würfelvariante gewählt. Die dazu erforderlichen Zufallszahlen werden im Unterprogramm ab Zeile 1000 erwürfelt.

Die Zeile 210 überprüft die Eingabe für jede der vier Reihen auf Vollständigkeit. Wurden die vier Ziffern als Zeichenkette I4$ für eine Reihe eingegeben, erfolgt in Zeile 230 deren Umwandlung in vier einzelne Zahlen IN. In Zeile 240 wird die zweite Eingabekontrolle durchgeführt. Anschließend werden aus den Eingaben die zugehörigen Codezahlen für die Pseudografikzeichen ermittelt und dem Feld FE (1, j) zugewiesen.

Zeilen 280 bis 410

Hier muß der Programmieranfänger etwas nachdenken, denn es stehen immerhin sechs ineinandergeschachtelte Laufanweisungen zur Debatte. Damit wird das vollständige Flächenmuster in der eingangs beschriebenen Weise auf den Bildschirm gebracht. Hierzu dient die PRINT AT-Anweisung, mit der die Kursorposition beim Z 1013 nicht beeinflußt wird. Aus diesem Grund Kann auch in Zeile 390 problemlos wieder Text in das Menüfenster geschrieben werden.

Programm: Jagd (1264 Bytes)

Bei unserer Entenjagd fliegt eine Ente in verschiedenen Höhen über den Bildschirm von rechts nach links. Der Jäger hat die Aufgabe, durch Betätigung der Schußtaste G auf der Tastatur die Ente zu erlegen. Der im - 1013 festgelegte Grafiksatz bietet natürlich weder eine Flugente noch einen Jäger mit Flinte. Auch hier wäre es schön, wenn wir uns die Figuren aus eigenen Pseudografikzeichen selbst bauen könnten. Da dies nicht möglich ist, suchen wir den Grafiksatz nach brauchbaren Zeichen ab. Bild 24 zeigt eine Möglichkeit. Bessere Ideen sind durch kleine Änderungen jederzeit realisierbar.

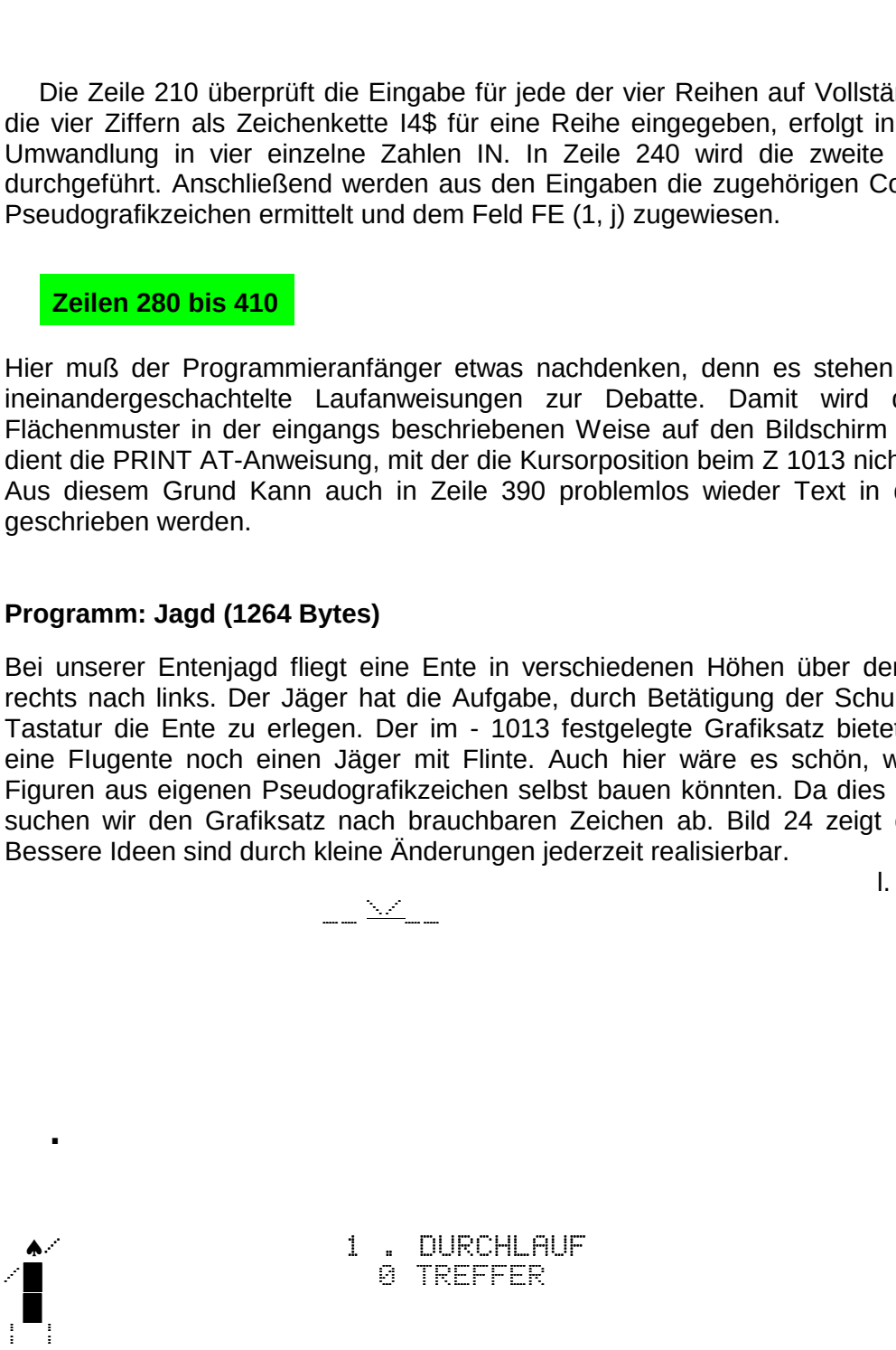


Bild 24: Bildaufbau im Programm Entenjagd

Im Programm werden die Durchläufe (besser Durchflüge) und die Treffer gezählt, wobei nach 50 Durchläufen abgebrochen wird. Damit ist eine Wettbewerbssituation geschaffen, bei der es auf Konzentration und Reaktionsschnelligkeit an kommt, wenn uns auch die Flachfolientastatur etwas zu schaffen macht.

Das Programm zeigt einige Grundprinzipien zur Gestaltung von Aktionsspielen auf dem Bildschirm. Es zeigt aber auch die Tempogrenzen bei der Arbeit mit einem BASIC-Interpreter. Die Nutzung eines BASIC-Compilers (z. B. für den PC 1715), des schnelleren FORTH-Interpreters, einer Compilersprache (z. B. PASCAL) oder gar eine Programmierung in Assembler würden die Zeitverhältnisse. deutlich machen. Dann könnten sich z. B. auch mehrere Enten, und diese vielleicht noch im "Schlängelflug", über den Bildschirm bewegen. Doch bleiben wir zunächst beim Listing 30 für unseren Z 1013 mit BASIC-Interpreter.

```
10 REM JAGD
20 WINDOW:CLS
30 PRINT AT(2,10);"ENTENJAGD"
40 PRINT AT(10,6);"DIE SCHUSSTASTE IST G"
50 PRINT AT(0,0);" ":PAUSE 30
60 CLS:LET P=0:LET D=1:PRINT AT(0,0);" "
70 REM TEXT UND JAEGER DARSTELLEN
80 PRINT AT(28,15);D:PRINT AT(28,18);". DURCHLAUF"
90 PRINT AT(28,17);P:PRINT AT(30,20);"TREFFER"
100 PRINT AT(28,2);CHR$(204)
110 PRINT AT(29,1);CHR$(158)+CHR$(255)+CHR$(155)
120 PRINT AT(30,20);CHR$(255)
130 PRINT AT(31,1);CHR$(192):PRINT AT(31,3);CHR$(159)
140 REM AKTION
150 LET Y=INT(RND(1)*22):LET Z=27:LET F=0
160 PRINT AT(29,3);CHR$(155):PRINT AT(28,3);" "
170 PRINT AT(Y,30);CHR$(156)+CHR$(146):PRINT AT(Y+1,30);
    CHR$(238)+CHR$(238)
180 FOR I=30 TO 1 STEP-1
190 PRINT AT(Y,I+1);" ":PRINT AT(Y+1,I+1);" "
200 PRINT AT(Y,I-1);CHR$(156)+CHR$(146)
210 PRINT AT(Y+1,I-1);CHR$(238)+CHR$(238)
220 REM ABFRAGE DER G-TASTE
230 IF Z=27 AND INKEY$="0" THEN LET F=1:GOTO 240:ELSE GOTO 260
240 PRINT AT(29,3);" "
250 PRINT AT(28,3);CHR$(152):PRINT AT(27,3);CHR$(209)
260 IF F<>1 THEN FOR ZV=1 TO 15:NEXT ZV:GOTO 310:REM ZEITVER
    ZOEGERUNG
270 PRINT AT(Z,3);" ":PRINT AT(Z-1,3)1CHR$(209)
280 IF Z-1=Y+1 AND (I-1=3 OR I=3) THEN GOSUB 500:GOTO 330
290 IF Z=1 THEN LET F=0:PRINT AT(Z-1,3);" ":GOTO 310
300 LET Z=Z-1
310 NEXT I
320 PRINT AT(Y,0);"  ":PRINT AT(Y+1,0);"  ":PRINT AT(Z,3);" "
330 IF D=50 THEN POKE 113,5:POKE  112,5: INPUT" WIEDE(J/N?";A$:
    GOTO 360
340 LET D=D+1
350 PRINT AT(28,17-LEN(STR$(D)));D:GOTO 150
360 IF A$="J" THEN GOTO 60
370 CLS:END
500 REM UP TREFFER
510 LET P=P+1
520 PRINT AT(30,19-LEN(STR$(P)));P
530 PRINT AT(Y,I-1);"**":PRINT AT(Y+1,I-1);"**)
540 PAUSE 10
550 PRINT AT(Y,I-1);"  ":PRINT AT(Y+1,I-1);"  "
560 RETURN
```

Listing 30: Entenjagd

Zeilen 10 bis 130

Die kurze Erklärung in den Zeilen 30 und 40 bleibt etwa drei Sekunden (Pause 301 auf dem Bildschirm, wobei mit der Anweisung PRINT AT (0, 0);" " das Kursorzeichen gelöscht wird. Dies erfolgt nochmals In Zeile 60, wo auch der Durchlaufzähler D mit 1 und der Trefferzähler P mit 0 belegt werden. Der aus Pseudografikzeichen zusammengesetzte Jäger wird in den Zeilen 100 bis 130 auf den Bildschirm gebracht. Er bleibt während des gesamten Programmablaufes an dieser Stelle. Nur beim Schuß erhebt er einen Arm (Zeilen 240 und 250). Natürlich muß auch das "Rücksetzen" des Armes für einen neuen Spieldurchlauf ordnungsgemäß programmiert werden.

Zeilen 140 bis 170

Der Variablen Y wird die zufällig ausgewählte Flughöhe der Ente zugewiesen. Die Variable Z beinhaltet die Zeilenposition des Geschosses, und die Variable F fungiert als Flag, das zunächst nicht gesetzt ist (F = 0). Die Zeile 160 besorgt bei den späteren Programmdurchläufen die "Armkorrektur", und die Zeile 170 enthält die aus insgesamt vier Pseudografikzeichen zusammengesetzte Ente. Soll die Ente aus anderen Zeichen erzeugt werden, so sind die Zeilen 170, 200 und 210 gleichermaßen zu ändern.

Zeilen 180 bis 210

Die Laufvariable I kennzeichnet den Entenflug von rechts nach links über den Bildschirm. Die Darstellung des Fluges erfolgt durch Löschung der beiden rechten Positionen der Ente in Zeile 190 und deren erneutes Setzen auf den Bildschirm in den Zeilen 200 und 210. Damit täuscht der Z 1013 eine Bewegung vor.

Zeilen 220 bis 320

Die Bedingung in Zeile 230 sorgt dafür, daß die Schußtaste G bei einem Durchlauf nur ein einziges Mal gedrückt werden kann. Unser Jäger verfügt also nicht über eine Zwillings- oder Drillingsflinte. Sobald nämlich das Geschoß abgefeuert wurde, ist der Zeilenzähler Z für das Geschoß kleiner als 27.
Bei Auslösung des Schusses erhält das Flag den Wert 1. Damit wird die in Zeile 260 eingebaute Zeitverzögerung außer Kraft gesetzt. Der Sinn des Ganzen besteht darin, daß mit und ohne Geschoß auf dem Bildschirm ein Entenflug mit gleichem Tempo gewährleistet ist. Das Geschoß selbst besteht aus dem Pseudografikzeichen mit der Codezahl 209. Es bewegt sich stets nur senkrecht nach oben. Die Trefferüberprüfung erfolgt in Zeile 280. Dabei werden nur die beiden unteren Zeichen, aus denen die Ente zusarnmengesetzt ist, als Treffer akzeptiert, FIügeIschüsse also nicht. Im Trefferfall wird das Unterprogramm ab Zeile 500 aufgerufen.
Wurde kein Treffer erzielt, dann rückt das Geschoß um eine Zeilenposition weiter nach oben. Bei Erreichung des oberen Bildschirmrandes sorgt die Programmzeile 290 für das Löschen des Geschosses vom Bildschirm.

Zeilen 330 bis 370

In Zeile 330 wird gefragt, ob die Ente schon 50 Durchflüge absolviert hat. Ist dies nicht der Fall, wird der Durchlaufzähler um Eins erhöht, der Inhalt der Variablen D auf dem Bildschirm angezeigt und mit Sprung zur Programmzeile 150 ein neuer Durchlauf eingeleitet. Durch Verwendung der LEN- und STR$-Funktion wird in Zeile 350 eine rechtsbündige Darstellung erreicht.

Zeilen 500 bis 560

Dieses Unterprogramm tritt bei erfolgreichem Schuß in Aktion. Dazu wird zunächst der Trefferzähler P um Eins erhöht und die Gesamtzahl der Treffer auf dem Bildschirm angezeigt. Auf die Darstellung fliegender Federn und einer herabstürzenden Ente haben wir verzichtet und auf die vier Zeichenpositionen nur einfache Sternchen gesetzt. Mit etwas Phantasie und Programmieraufwand könnte natürlich auch der Hund des Jägers die

herabstürzende Ente ergreifen und seinem Herrn vor die Füße legen. Leider bietet unser Z 1013 keine Tonausgabe. Sonst hätte in Zeile 540 auch ein Siegessignal programmiert werden können. In unserem Programm werden nach etwa einer Sekunde die Sternchen wieder vom Bildschirm entfernt. Nach Rückkehr in das Hauptprogramm wird mit der Programmzeile 330 fortgesetzt.

Programm: Rallye (1699 Bytes)

Rallyesport verlangt vom Fahrzeugführer höchste Aufmerksamkeit und die Fähigkeit, im richtigen Augenblick in der erforderlichen Weise zu reagieren . Vom Auto wird während der Rallye hohe Zuverlässigkeit erwartet. Diese Zuverlässigkeit bietet für unser Spiel der Z 1013. Rallye ist einfach zu programmieren. Durch die Kombination verschiedener Fähigkeiten des Computers verwandeln wir den Bildschirm in eine sich bewegende Straße, die mit zahlreichen Windungen dem "Rallyefahrer" Aufmerksamkeit und Reaktionsschnelligkeit abverlangt, um Karambolagen mit den Seitenplanken zu vermeiden (Bild 25). Wir nutzen die Tasten P (nach links) und W (nach rechts), um das Fahrzeug zu steuern.

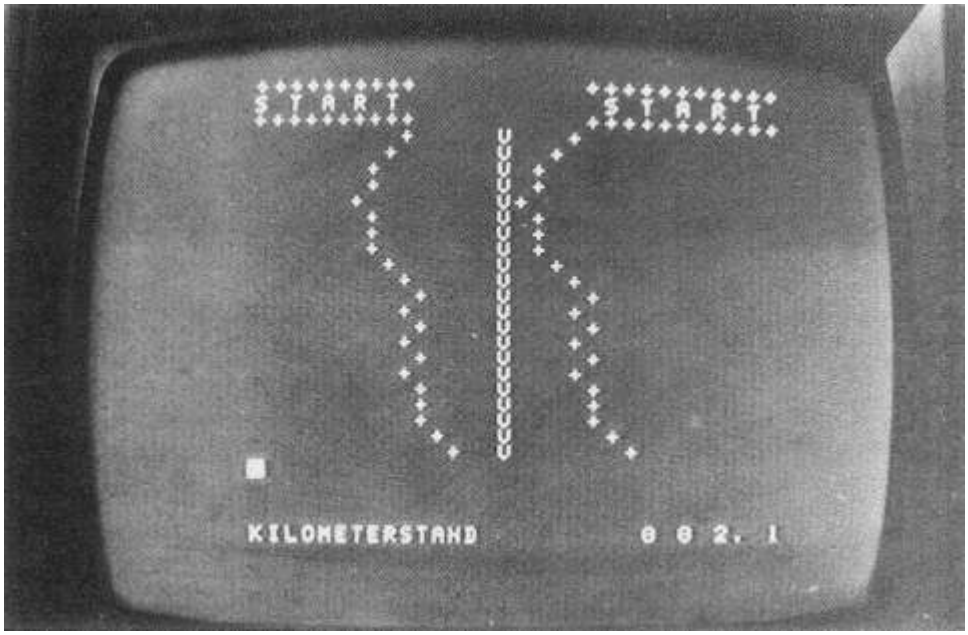


Bild 25: Situation im Spiel

Ihre Lage auf der Flachfolientastatur bietet sich dafür an, weil man anatomisch günstig mit beiden Zeigefingern arbeiten kann.

Vor dem Start ist eine Geschwindigkeitsstufe zu wählen, die dann über den gesamten Spielverlauf die Schnelligkeit der Straßenbewegung konstant steuert. Ein Kilometerzähler registriert exakt die unfallfrei zurückgelegte Wegstrecke. Einhundert Meter entsprechen in unserem Spiel einer Bildschirmzeile (Listing 31).

```
10 WINDOW:CLS:REM RALLYE
20 PRINTAT(12,10);"R A L L Y"
30 FOR N=8 TO 22:PRINTAT(10,N);CHR$(42):PRINTAT(14,N);
   CHR$(42):NEXT N
40 FOR N=11 TO 13:PRINTAT(N,8);CHR$(42):PRINTAT(N,22);
   CHR$(42):NEXT N
```

```
50 PRINTAT(30,24);">ENTER<":PRINTAT(0,0);CHR$(32)
60 IF INKEY$<>CHR$(13) THEN 60
100 CLS:PRINT:PRINT:PRINT"S P I E L R E G E L
    :":PRINT:PRINT
110 PRINT"EIN FAHRZEUG IST SICHER DURCH":PRINT
120 PRINT"EINE VERWINKELTE STRASSE ZU":PRINT
130 PRINT"FUEHREN.":PRINT:PRINT"MIT DER TASTE P LENKT
    MAN NACH":PRINT
140 PRINT"LINKS, MIT DER TASTE W NACH":PRINT
150 PRINT"RECHTS.":PRINT:PRINT"DIE GESCHWINDIGKEIT IST
    VOR":PRINT
160 PRINT"SPIELBEGINN WAEHLBAR.":PRINT
170 PRINT"BEI BERUEHRUNG MIT DEM STRASSEN-"
180 PRINT"RAND WIRD DAS SPIEL ABGEBROCHEN."
190 PRINTAT(30,24);">ENTER<":PRINTAT(25,0);CHR$(32)
200 IF INKEY$<>CHR$(13) THEN200
300 CLS:PRINT:PRINT:PRINT"WAHL DER GESCHWINDIGKEIT :":
    PRINT:PRINT
310 PRINT"ES GIBT DIE STUFEN 1,2,3,4 UND 5":PRINT
320 PRINT"STUFE 5 IST DIE SCHNELLTSTE !":PRINT:PRINT:PRINT
330 INPUT"EINGABE DER STUFE:"
340 IF G=1 THEN GE=5:GOTO 400
350 IF G=2 THEN GE=4:GOTO 400
360 IF G=3 THEN GE=3:GOTO 400
370 IF G=4 THEN GE=2:GOTO 400
380 IF G=5 THEN GE=1:GOTO 400
400 CLS:WINDOW 0,24,0,31
410 M=0:H=0:I=0:J=0:GOSUB 870
420 FOR N=1 TO 10:PRINTTAB(15)"V":FOR Z=0 TO 150:
    NEXT Z:NEXT N
430 PRINT"++++++++++           ++++++++++"
440 PRINT"S T A R T             S T A R T"
450 PRINT"++++++++++           ++++++++++"
500 B=10:L=10:A=15
510 R$=INKEY$
520 IF R$<>""THEN 540:ELSE 530
530 FOR Z=0 TO GE*30:NEXT Z:GOTO 600
540 IF NOT (R$="P" OR R$="W") THEN 510
550 IF R$="P" THEN A=A-1
560 IF R$="W" THEN A=A+1
600 X=INT(3*RND(1))-1:L=L+X
610 IF L<1 THEN L=1
620 IF L+B>30 THEN L=30
630 IF A=L OR A=L+B THEN 700
640 PRINTTAB(L)"+";TAB(A)"V";TAB(L+B)"+"
650 GOSUB 800:GOTO 510
700 IF A=L THEN PRINTTAB(L)CHR$(202):GOTO 720
710 IF A=L+B THEN PRINTTAB(L+B)CHR$(202)
720 END
800 REM ANZEIGEN DER STRECKE
805 M=M+1
810 IF M=10 THEN 820:ELSE 870
820 M=0:H=H+1
830 IF H=10 THEN 840:ELSE 870
840 H=0:I=I+1
850 IF I=10 THEN 860:ELSE 870
860 I=0:J=J+1
870 PRINTAT(28,0);"KILOMETERSTAND":PRINTAT(28,22);J:
    PRINTAT(28,24);I
880 PRINTAT(28,26);H:PRINTAT(28,28);".":PRINTAT(28,29);M
890 RETURN
```

Listing 31: Rallye-Spiel

Zeilen 10 bis 200

Die übliche Programmüberschrift wird durch Sternchen umrahmt, die mit Hilfe von zwei Schleifenanweisungen einfach auf den Bildschirm zu bringen sind. Die Erläuterung der Spielregeln erfolgt ausschließlich mit PRINT-Anweisungen.

Zeilen 300 bis 380

Die Wahl der Geschwindigkeit wird für jedes Spiel durch den Nutzer erneut vorgenommen. Es stehen fünf Stufen zur Verfügung. Der eingegebene Wert G wird in einen Wert GE umgewandelt. Das ist erforderlich, weil GE später als Faktor in einer Schleifenanweisung benötigt wird.

Zeilen 400 bis 450

Zur Startvorbereitung wird der Kilometerzähler über eine PRINT AT-Anweisung auf den Wert 0 gesetzt. Das Symbolfahrzeug "rollt" mit konstantem Tempo zum Start. In dieser Phase ist noch kein Lenkeinfluß gefordert.

Zeilen 500 bis 560

In der Zeile 500 werden die Ausgangsgrößen initialisiert. B entspricht der Straßenbreite, L dem linken Straßenrand und A der Startposition des Fahrzeugs in der Straßenmitte. Der rechte Fahrbahnrand ergibt sich aus L + B. Durch Nutzen der INKEY$-Abfrage wird das Fahrzeug geführt. Liegt keine Tastatureingabe vor, verbleibt das Fahrzeug bei fortgesetzter Bewegung in seiner ursprünglichen Position, obwohl sich der Fahrbahnrand verändert. Durch die Tastatureingaben P bzw. W wird das Fahrzeug gesteuert. An dieser Stelle sei auf das Programm Punktsteuerung (Werkzeugkiste für Software des Z 1013) verwiesen. Der vorher bereitgestellte Wert GE wird jetzt für die geschwindigkeitsregulierende Schleife benötigt.

Zeilen 600 bis 650

Über den ZufaIlsgenerator wird die Änderung des Straßenverlaufes bewirkt. Durch die zufällige Bestimmung der Werte 0,1 oder 2, von denen dann der Wert 1 subtrahiert wird, erhält man die Straßenführung nach links, geradeaus oder nach rechts. Das Verlassen des Bildschirms wird verhindert, indem mögliche Werte L< 1 bzw. L + B>30 in konstante Werte 1 bzw. 30 umgewandelt werden.

Zeilen 700 bis 720

Im Falle einer Karambolage mit der Straßenleitplanke wird die Berührungsstelle optisch angezeigt und das Spiel beendet.

Zeilen 800 bis 850

Dieses Unterprogramm liefert die aktuelle Kilometeranzeige. Das Prinzip entspricht dem Programm der Digitaluhr.

Mit der vorliegenden Broschüre wurden Anregungen zur unmittelbaren Nutzung des Z 1013 gegeben. Für alle interessierten Leser gibt es eine Fülle von Literatur zur weiteren Beschäftigung mit der Programmiersprache BASIC und dem Mikrocomputer. Eine kleine Auswahl der Literatur, die in der DDR vertrieben wurde und in den Bibliotheken zur Verfügung steht, soll empfohlen werden.

Boon, K. L.:
BASIC für Tischcomputer. Pflaum Verlag KG, München 1983
Bückner, U.:
Kleincomputer leicht verständlich. VEB Fachbuchverlag, Leipzig 1986
Claßen, L.; Oefler, U.:
Wissensspeicher Mikrorechnerprogrammierung. VEB Verlag Technik, Berlin 1986
Franke, K.:
Einführung in die Mikrorechentechnik. VEB Verlag Technik, Berlin 1986
Groh,J.:
Kleincomputerfibel. Akademie-Verlag, Berlin 1986
Gutzer, H.:
Das kann der Mikrocomputer. Urania-Verlag, Leipzig-Jena-Berlin 1985
Heblik, P.:
Wissensspeicher BASIC. Volk und Wissen Volkseigener Verlag, Berlin 1986
Hopfer, R.:
Mikrorechentechnik - allgemeinverständlich. VEB Fachbuchverlag, Leipzig 1985
Hopfer, R.; Müller, R::
BASIC - Einführung in das Programmieren. VEB Fachbuchverlag, Leipzig 1986
Müller, S.:
Programmieren mit BASIC. VEB Verlag Technik, Berlin 1986
Strelocke, K.; Hoffmann, D.:
Die Dialogprogrammiersprache BASIC. Verlag Die Wirtschaft, Berlin 1981
Werner, D.:
BASIC für Mikrorechner. VEB Verlag Technik, Berlin 1986

Periodika und Zeitschriften

edv aspekte. Verlag Die Wirtschaft, Berlin
Jugend und Technik. Serie ABC der Mikroprozessortechnik, Hefte 1/84 bis 9/86. Verlag Junge Welt, Berlin
Kleinstrechner-Tips. VEB Fachbuchverlag, Leipzig
MEGA-BIT. Verlag Junge Welt, Berlin
MP Mikroprozessortechnik. VEB Verlag Technik, Berlin
Rechentechnik/Datenverarbeitung. Verlag Die Wirtschaft, Berlin