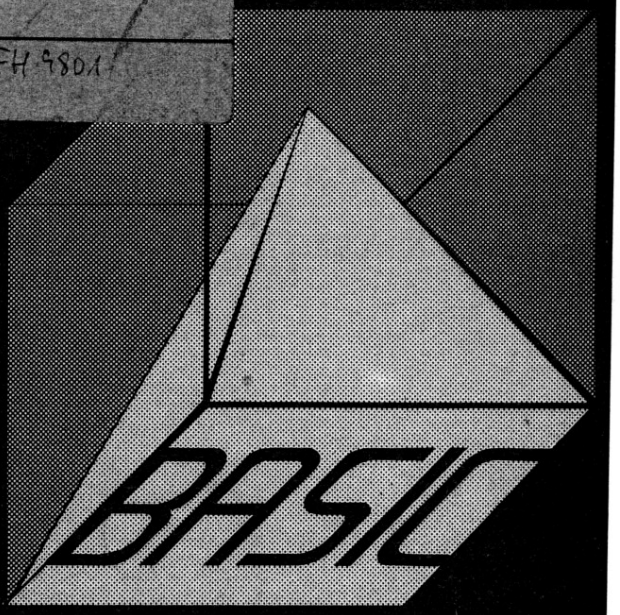
Man kann nicht nur rechnen, Vermögen und Börsenkurse verfolgen, sondern auch mit dem ZX81/Spectrum richtig „arbeiten". Dadurch lernt man nicht allein eine Menge über Programmierung und die Welt, in der wir in Zukunft leben werden; man hat auch einen echten Nutzeffekt für sich persönlich. Im vorliegenden Buch werden die vielfältigen Möglichkeiten klar aufgezeigt. Die Grenzen sind in der leichten Programmsprache BASIC kaum abzustecken. Selbst wer nicht selber programmieren will, kann die Listings für sich voll nutzen. Alles ist narrensicher mit einleuchtender einfacher Benutzerführung ausgestattet. Die wichtigsten Kapitel:

- wie man seinen Lottotip vom Computer bekommt (natürlich ohne Gewinngarantie, aber mit Eingabe von eigenen Vorgabezahlen und umfangreicher Abfrage bisheriger Wochenzahlen)
- wie man Texte einspeichert, editiert und ausdruckt – bis hin zum **komfortablen Wortprogramm mit Blocksatz**
- wie Mitgliedskarteien, Dateien und ähnliches speicherplatzsparend geführt und verwaltet werden (einschließlich einer anwendungsfreundlichen Vereinsstatistik)
- wie Übersicht in die heimischen Finanzen kommt, wie man Vermögenswerte auch grafisch aufbereitet verfolgt
- wie man den ZX81/Spectrum sogar zu **Managementzwecken** einsetzt, d.h. Projekte begrenzter Reichweite kontrolliert und zahlreiche Auswertungen „fährt"

Neben einer dezidierten Einführung in die Welt des Sinclair bietet das Buch, von einem Profi geschrieben und herausgegeben, eine Fülle sehr effektiver Programmtips und -tricks. Mit diesem Buch wird dem Besitzer eines jeden ZX81/Spectrum erst klar, was sich mit seinem Gerät alles anstellen läßt.

ZX 81 / Spectrum-Anwendungen

ZX 81 / Spectrum-Anwendungen



ZX 81 / Spectrum - Anwendungen

. . . was sich mit dem Sinclair
alles machen läßt

Herausgegeben von
Tom Softwell

CIP-Kurztitelaufnahme der Deutschen Bibliothek

(ZX-einundachzig-Spectrum-Anwendungen)
ZX-81-Spectrum-Anwendungen : ... was sich mit d. Sinclair alles machen lässt / hrsg. von Tom Softwell. - Gensingen : Luther, 1985.
ISBN 3-620-00104-9

NE: Softwell, Tom (Hrsg.)

Inhaltsverzeichnis

Kapitel IV: Finanz & Management

Kapitel V: Daten und Verarbeitung

Kapitel VI: Tips & Tricks

Der ZX 81 nur eine Spielmaschine?

Für viele junge Menschen bedeutet der ZX den ersten Kontakt mit der Computerwelt. Noch vor wenigen Jahren unvorstellbar hat Sinclair das Computerzeitalter in die Wohnstube gebracht. Endlich ein richtiger "Micro", der erschwinglich ist ... so werden viele gesagt haben, als sie den ZX 81 sahen. Und nachdem sich die Freude über den eigenen Computer gelegt hatte, fragte man sich: "Was kann ich eigentlich mit dem Wunderwerk der Technik anfangen?"

Wohl zu keinem Heimcomputer gibt es ein ähnlich umfangreiches Angebot an Software und allerlei Zusätzen wie bei Sinclairs Maschinchen. Das Spielangebot überwiegt, ja scheint unerschöpflich zu sein; vor allem seit der Spectrum mit Farbe und HiRes (= hochauflösender Grafik) "glänzt". Dabei verfügen beide Rechner über ein BASIC, das Maßstäbe setzt. Wie in einem nicht ganz ernst zu nehmenden Vergleich gezeigt wird, ist Sinclairs BASIC und Programmhandling IBM's Sprößling, dem PC, teilweise überlegen. Die vielfältigen Möglichkeiten des ZX (und Spectrum) sind durch das Spielangebot z.T. gar nicht bekannt. Es lohnt sich, sie zu entdecken. Eigentlich ist der ZX für Spiele zu schade...

So fängt denn auch das vorliegende Buch da an, wo andere enden - bei Spielen. Das Programmangebot beginnt mit Lotto und reicht bis zur quasi professionellen Datenverarbeitung. Vom Einfachen zu Anspruchsvolleren möchten wir den User zu höherwertigeren Programmbeispielen hinführen. Der ZX ist zu mehr gut als Sternekampf und Froschhüpfen... Ob Haushalt oder Vermögen, ob Wort oder Text, ob Lottozahlen oder (bescheidene) Managementaufgaben - der ZX kann bei alledem eingesetzt werden. Nach einem Streifzug durch "Sinclairs Welt" zeigt das Buch ab Kapitel II, was mit effektiver Programmierung alles im Sinclair steckt. Und das ist dank Dingen wie komfortabler Stringbehandlung und logischer Vergleichsoperatoren nicht wenig. Schon eine einfache Lottozahlenerzeugung wächst sich flugs zur praktischen Wochenstatistik aus. Weitergeht es ab Kapitel III von der Schmierdatei bis hin zum ausgewachsenen Wortprozessor (incl. wanderndem Cursor und Blocksatz). Die Anwendungen werden dann anspruchsvoller, vielseitiger und umfassender. In Kapitel IV reicht die Skala von der häuslichen Vermögensverwaltung bis zur Projektkontrolle mit Terminen, Kosten und Diagrammen. Den Abschluß komplett ausgetestete Programme bildet die Mitgliederverwaltung eines Sportvereins. Mit umfangreicher verständlicher Erläuterung versehen, wird klar, wie man Stammdaten, variable Sätze und komplexe Strings handhabt. Schließlich wurde eine Fülle Tips und Tricks in einem eigenen Abschnitt gesammelt, die Anfänger wie fortgeschrittenem ZXer nützliche Hinweise liefern.

Um es klar und deutlich zu sagen, - das Buch nimmt den ZX ernst.Es zeigt nicht nur, wie man ihn effektiv programmiert, sondern auch, welche Anwendungsfelder sich dem Heimcomputeristen eröffnen. Das Buch ist kein reines Programmierbuch, es erläutert vielmehr die Spezifika von Sinclairs Maschine. Die Programme sind als Lernprogramme zu verstehen. Der Benutzer soll aus dem Stadium des "Kopisten" herauskommen. Dementsprechend breit ist die Programmdokumentation angelegt. In den ersten Kapiteln wurde sie in die Programmbeschreibung integriert, später steht sie (ab Kapitel IV) als geschlossener "Block" hintenan. Vieles ist zudem als Vorschlag, als Anregung und Lösungsidee zu verstehen. Nur selbst erworbenes Programmwissen ist Macht über den Computer. Daher der Weg von Einfachen zum Schwierigen, wenngleich das vorliegende Werk wohl kaum als "Einführung für Anfänger" gedacht ist.

Die Programme stammen ursprünglich von Autoren, welche sich zwecks Veröffentlichung an den Verlag gewandt hatten. Nach kritischer Begutachtung des Gebotenen konnte vom Meisten lediglich die Idee als verwertbar eingestuft werden. Manche Lösungen waren so durftig, daß nur eine totale Überarbeitung durch den Herausgeber etwas Brauchbares zustande brachte. Lediglich die Vorlagen zu Kapitel IV von Autor Rybarczyk verdienen lobende Erwähnung. Sie konnten relativ einfach in "Hochglanzform" gebracht werden und gaben für den Anwendungsbereich "Finanz & Management" eine gute Grundlage ab.

Noch ein Wort zur "Maschine" ZX 81. Ist der 81er nicht dank Spectrum (und demnächst QL) ein "sterbender Rechner"? Was sollen ZX81-Anwendungen beim Spectrum mit Farbe und ausgefuchster Grafik? Dazu ist zweierlei zu sagen:

1. Kein ZX81 braucht bei sinnvoller Anwendung zu vergammeln. Die meisten Programme kommen ohne "HiRes" aus, Farbe ist ein keineswegs notwendiges Attribut. Ein gut ausgebauter ZX mit (mindestens) 16 K RAM und Tastatur ist ebenso sinnvoll und preiswert wie ein Spectrum.

2. Der BASIC-Instruktionssatz ist bei beiden im wesentlichen der gleiche. ZX-Basic stellt ein Subset des Spectrum-Basic dar, alle Programme des Buches laufen auf ZX wie Spectrum (auf welchem sie überwiegend getestet wurden).

Zu letzterem gibt es nur zwei Ausnahmen: die CODES haben beim ZX eine andere Ordnung, was lediglich bei "künstlicher" Inversdarstellung (vgl. Programm PC) für den ZX zum Tragen kommt; da der Spectrum beim PLOT-Befehl in höherer Auflösung arbeitet, muß ein Programm auf dem Spectrum die Blockgrafik simulieren (Vergrößerungsfaktor 4). Ansonsten gibt es sgn. SlowLoader, mit denen man jedes Zx-Programm auf dem Spectrum zum Laufen kriegt. Umgekehrt gehts leider nicht...(Übrigens liegen alle Programme des Buches mit Testdaten gefüllt dem Verlag in Spectrum-Version vor; es geht also, für beide Rechner nahezu identische Programme zu entwickeln). Noch etwas - das vorliegende Werk unterscheidet sich von Nur-Programmbüchern. Unser Ziel war es, *Anwendungen* zu zeigen und nicht inzwischen überholte Dinge wie "Basic für Anfänger". Nur wer erfährt, was der Sinclair alles kann, wir seinen Heimcomputer sinnvoll einsetzen. Und der Sinclair kann eine ganze Menge....

Kapitel I:

Die Sinclair-Welt

1. Mr. Sinclair

Der Markt

Dies Kapitel ist als Einführung für Anwender gedacht. Es möchte Sie mit Person, Produkt und Sinclair spezifischen Eigenheiten vertraut machen. Wir wollen im folgenden nicht nur aufzeigen, was der Mikrozwerg aus England alles kann, sondern auch wie er entstand, was seine Stärken und Schwächen sind. Jeder, der den ZX81 ernsthaft anwendet, wird sich mit seinen Eigenheiten vertraut machen (müssen). Ehe wir zu dezidierten Anwendungsfällen mit kompletten Programm-Listings kommen, sei am Anfang ein wenig Hintergrundmaterial gebracht. Das dürfte nicht nur interessant, sondern auch nützlich für den, der sich ernsthaft mit seinem ZX81 beschäftigt, sein. Ein Anwender ist spätestens, wer sich nach einer Reihe Spiele bemüht, den ZX81 nutzbringend selber zu programmieren. Wir hoffen, daß alle, die das Spielestadium überwunden haben, erkennen werden, daß der ZX81 durchaus eine nützliche Hilfe sein kann. - Aber wenden wir uns zunächst dem Markt der Mikros zu.

Nicht zuletzt durch die Programmiersprache BASIC, was man als "Allzweck-Anfänger-Befehlssprache" übersetzen kann (BEGINNERS ALL PURPOSE SYMBOLIC INSTRUCTION CODE), hat sich inzwischen ein Riesenmarkt für Heimcomputer aufgetan. Die kleineren Brüder der sogenannten PC's (= Personal Computer) gehören heute zum täglichen Bild vieler Menschen. In zahlreichen Haushalten stehen sie zunächst für den Junior zum Spielen. Aber immer mehr Erwachsene beschäftigen sich inzwischen mit den Wunderwerken moderner Technik. Nicht nur, daß man ein Stück Zukunft besitzt, von den "Gedankenzwergen" geht eine beträchtliche Faszination aus. Inzwischen zeichnet sich ein Riesen-Markt ab. Das MAFO Institut IDS schätzt für 1984 in der BRD einen Absatz von 215.000 Kleinrechnern für Heim und Hobby (Verkaufsprognose). Kein Zweifel - "... der Durchbruch hat sich vollzogen", so Commodore-Europachef Hans Speyer (vgl. Spiegel Nr. 50/83 Seite 172 f.) Im Bereich Heim- und Personalcomputer tummeln sich derzeit etwa 350 Hersteller. Nach einhelligen Aussagen von Experten werden nur zehn überleben. SINCLAIR wird mit Sicherheit darunter sein ...

Inzwischen wurden vom ZX80 und seinem Nachfolger ZX81 schätzungsweise 2,5 Millionen verkauft. Das nimmt nicht Wunder, selbst wenn man bedenkt, daß sich die Wunderwerke der Elektronik bisher eher in der falschen (Spiel-) Ecke versteckt haben. Die FAZ vom 01.11.1983: "Allzu bald entpuppt sich das Gros der neuen Heimcomputer als Spielautomaten ohne ... praktischen und didaktischen Nutzwert..." Wirklich ?

Zweifellos wurden zwei Fehler zugleich gemacht, bedenkt man das Desaster bei Texas Instruments und anderen Herstellern:

a) Von Seiten der Industrie wurden die Rechner vom falschen Ende, nämlich für Spiele, verkauft.

b) Beim Käufer hat man sich ebenso Falsches versprochen und lange Zeit an ein "Spielbild" geglaubt.

Inzwischen hat sich ein Wandel vollzogen, wie die FAZ zugibt: "Nach fünf Jahren Kinderzeit... wird die Kleincomputerindustrie jetzt erwachsen." Gefragt sind nicht nur billige Rechner, sondern auch Ideen. Oder, um den Spiegel zu zitieren: "Gute Ideen für den Hausgebrauch der kleinen Computer sind ... gefragt. Fast alles, was die Industrie dem Amateur bislang...empfiehlt, entpuppt sich als...Spielerei." Und weiter: " Die Wirklichkeit der schönen, neuen Computerwelt im Heim wird wohl anders aussehen. Nicht Spaß, Spiel, Spannung werden schließlich mit den Computern ins Wohnzimmer einziehen, sondern der verlängerte Arm von Büro und Betrieb." - Nun denn, genau das ist unsere Absicht mit dem vorliegenden Werk. Wir möchten Sie vertraut machen, was man mit dem ZX81 an ernsthafter Anwendung alles machen kann. Aber dringen wir weiter in die SINCLAIR-Welt ein, schauen wir uns den Mann an, dem das "Marketing-Kunststück" gelang, von seinen Winzigrechnern mehr als zwei Millionen zu verkaufen. Wie kommt es, daß die kleine Firma Sinclair-Research mit etwa fünfzig Mann es fertigbrachte, 1982 bei 54 Millionen Pfund Umsatz 14 Millionen Pfund Gewinn zu machen? Selbst wenn man bedenkt, daß die Fertigung der Elektronikzwerge von Größeren (Timex und Emi) besorgt wird..

Was ist das für ein Mann, was ist das für ein Unternehmen, das die Konkurrenz immer wieder das Fürchten lehrt... und dem Privatmann leistungsfähige, billige Kleinrechner beschert?

Sir Clive persönlich

Mr. Sinclair, der sich seit Juli 1983 Sir nennen darf und nach dem Umzug März 1983 in einem Computerzentrum (Herstellungsaufwand 3/4 Millionen Pfund) residiert, ist zunächst ein typischer Selfmademan. Der Magier von Camberley, wie man ihn nennen könnte, hat es geschafft, von seinem futuristischen Zentrum aus Glas und Stahl Millionen Pfund zu dirigieren. Ähnlich wie in Deutschland Neckermann und Grundig fing er praktisch bei Null an. Darüber hinaus hat der Urheber moderner, billiger Gebrauchselektronik ein Handicap, nämlich keine abgeschlossene Schulbildung. Ganz um Gegenteil, nach einer planlosen Jugend mit rastlosen Stationen (dreizehn Schulen!) war er zunächst als technischer Autor tätig. Im zarten Alter von 17 Jahren schrieb er über Elektronik, seinem Lieblingsgebiet von jung auf, was die Feder hielt. Innerhalb von vier Jahren wurden siebzehn Bücher am laufenden Band produziert. Mit 22 Jahren borgte er sich 50 Pfund und startete seine Sinclair Radionics mit dem Verkauf von Radioteilen. Nicht technisches Wissen, sondern eine gewisse Cleverness schuf die Basis für Größeres. Pfiffig erwarb er billige, ausgemusterte Transistoren, die Industrieanforderungen nicht genügten, für Radioamateure aber einen Segen darstellten. Ebenso kostengünstig wie gekauft, schlug er sie in Massen per Post los. Als technischer Buchautor wußte er bestens über Radiotechnik Bescheid und lieferte gleich Bauanleitungen zum Basteln von allerlei interessanten Baugruppen mit. Wie Grundig ging sein Weg weiter, indem er als nächstes einen Radiobausatz anbot. Auch sein Nerv für publikumswirksame Werbung war erkennbar: Anzeigentitel seines Mini-Selbstbau-Radios "Bauen Sie sich das kleinste Radio der Welt".

Nach allerlei Geschäften mit HiFi und Kleinelektronik erreichte sein Unternehmen fünf Jahre nach Gründung die stattliche Umsatzzahl von 100.000 Pfund Nun war es an der Zeit, sich in das wissenschaftsorientierte Cambridge zu begeben. In der Hochburg englischer Forschung gelang es Mr. Sinclair, einen ersten technischen Alleingang zu starten. Ein Taschenrechner, der Sinclair EXECUTIVE, vereinte ebenso billige wie leistungsfähige Elektronik im Innern. Es gelang binnen kurzem, von diesem Erfolgsmodell eine Riesenstückzahl loszuschlagen. Nach schnellen Millionenumsätzen investierte Sinclair ab 1975 in "Research". Schon damals wird deutlich, daß Mr. Sinclair Geld und Luxus wenig bedeutet; jede verdiente Mark wanderte in die Forschung. Sein Ziel ist, neue revoutionäre Ideen auszubrüten und dem breiten Publikum als Gebrauchsgut zuzuführen. Das erste Forschungsergebnis,

eine Quarzuhr (Black watch), wurde jedoch wegen nicht vorhergesehener- und wohl auch nicht einkalkulierter - Produktions- und Produktschwächen ein Flop. Nur mit Hilfe der englischen NEB gelang es, neue Gelder loszueisen. Und das Projekt war in diesem Fall nichts Geringeres als ein Taschenfernseher! Nachdem Sinclair so drei Jahre weitermachen konnte, in denen er seine Mikrovision sowie fünf Rechnertypen und drei Messgeräte konstruierte, kam es zum Knall mit der NEB. Kurz entschlossen kündigte Mr. Sinclair den Vertrag, ließ sich 7,8 Pfund Entwicklungskosten für das TV-Gerät bezahlen und entwickelte in aller Stille seinen ersten Computer.

Die englische Firma Binatone erhielt von der NEB die Produktionsrechte zum Taschenfernseher, mußte jedoch feststellen, daß eine Serienfertigung ohne Profit bleiben würde... Ein genialer Schachzug von Mr. Sinclair oder nur besessener Forscherdrang ohne Geschäftssinn?

Nach einem letzten freundlichen Händedruck machte sich Mr. Sinclair im Juli 1979 mit seiner Sinclair-Research selbstständig. Schon ein halbes Jahr später ließ er seinen ersten Rechner, den ZX80, vom Stapel. Und rund ein Jahr später kam der ZX81 nach. Seitdem kommen immer neue Gerüchte und Produkte aus dem Ideenbrutkasten von Sinclair-Research, insbesondere von der Abteilung Metalab. Der neueste, tollste Geniestreich für den Zukunftsmarkt Heim-/Geschäftscomputer (mit fließendem Übergang) ist weder ZX81 noch Spectrum und Microdrive, sondern ein geheimnisvoller QL. Was nichts anderes bedeutet als Quantum Leap (= etwa Riesensprung). Nach letztem Stand der Dinge (März 1984) ist Sir Clive mit dem neuen Computer (Verkaufspreis 400 Pfund) gegen den mächtigen IBM PC angetreten. Wieder einmal möchte er Standards setzen: Ein neuer Prozessor, der Motorola 68008 mit interner 32-bit-Struktur, soll den amerikanischen Riesen und andere aus dem Felde schlagen. Für ungerechnet weniger als DM 2.000,- wird eine 128 K-Maschine mit zwei Microdrives, zahlreichen Schnittstellen, eigenem Betriebssystem QDOS (Multiasking!) und Superbasic versprochen. Ein Programmpaket, das Wortprozessor, Spreadsheet, Datenverwaltungs- und Grafik-Software umfaßt, wird gratis mit angeboten. Man darf gespannt sein... Was ist das für ein Mann, der den Markt mit revolutionären Ideen in Bewegung bringt?

Sinclair ist ein Mensch, der risikofreudig und fortschrittsbesessen Träume von Millionen wahrmacht. Was die Kosten seiner Produkte, wie die ganze kaufmännische Seite anbelangt, - Geld spielt für ihn keine Rolle. Auf persönlichen Gewinn weniger aus, möchte er seine Ideen, Träume und Visionen in Wirklichkeit umsetzen. Auch der Weg hierzu ist unkonventionell: Er zieht die besten Leute heran, läßt sie ungestört und ohne auf den Pfennig zu gucken forschen und entwickeln, um anschließend die Produktion in Millionenstück von anderen besorgen zu lassen. Ein Projekt ist für Sinclair nur interessant, bis es eine technische Lösung gibt. Die kaufmännische Seite, wenngleich in letzter Zeit mit reichlich Gewinn beglückt, interessiert ihn weniger. Die Arbeit sollen sozusagen andere machen, seine Einstellung scheint zu sein: "Ich bin der Chef einer Ideenschmiede". Von der bekannten Zeitschrift PCW (siehe Oktober 1983) nach seinem Lebensziel befragt, gab er sinngemäß folgende Antwort: "Ohne finanzielle Beschränkung technische Träume zu verwirklichen". Mr. Sinclair haßt "Fassaden", liebt Kinder, weil diese noch nicht die Fähigkeit erworben haben, zu schauspielern...

Mr. Sinclair alias Sir Clive hat neben der Fähigkeit, Eggheads für seine Computerkonstruktionen heranzuziehen, ein nicht geringes Sendungsbewußtsein. Nach einer Meldung der FAZ vom 12.09.83 will er inzwischen mehr, als das In- und Ausland mit preiswerten Mikros zu

beglücken. Für die Zukunft plant er nichts Geringeres, als den Chinesen auf die Sprünge zu helfen. Mit zwei rotchinesischen Firmen wurde inzwischen ein Abkommen geschlossen, um beim Aufbau einer mikroelektronischen Industrie in China mitzuhelfen. Zunächst sollen Bauteile vom ZX81 und Spectrum ins ferne Guangzhou geschafft werden, um im Rahmen eines Erziehungsprogramms Elektronik unter's Volk zu bringen. FAZ-Tenor: "Sinclair - wieder ein Pionier in seiner Sparte".

Auch sonst peilt Sir Clive neue technische Ufer an. Alle Gewinne von Sinclair Research (Mr. Sinclair besitzt nach wie vor 85 % des Gesellschaftskapitals von 136 Millionen DM) werden in die Forschung gesteckt. Neben Rechnern wird an Vielem geknobelt, was in Zukunft Bedeutung erlangen könnte:

- Fernseh-Satellitenempfang mit einem Adapter (für 100 Pfund) und einem Empfänger mit hoher Bildauflösung (für 500 Pfund)
- Ein Elektro-Auto in Zusammenarbeit mit Hoover, das möglicherweise schon Ende 1984 produktreif sein soll.

Für letzteres wurde inzwischen eine eigene Forschungsfirma gegründet, die Sinclair Vehicle Projects Ltd. (so gemäß FAZ vom 29.02.84). - Aber auch seine Auslandstochter sind nicht untätig. Bis hin in die USA, wo von TIMEX eine eigene Rechnerversion für den amerikanischen Markt gebaut wird, ist Sinclair Research vertreten. In Deutschland peilte die hiesige Vertretung für 1983 einen Umsatz zwischen 10 - 12 Millionen an. Wenngleich oft mehr Technik versprochen als geliefert wird - unvergessen ist die 15monatige Verzögerung bei den Microdrives -, es immer wieder zu Marketingpannen kommt, eins bleibt auf jedem Fall zutreffend:

Mr. Sinclair hat es verstanden, der Masse der Verbraucher preiswerte, leistungsfahige Gebrauchselektronik zu bescheren. Und dabei spielte der ZX81 die Hauptrolle. Kein Wunder, wenn man seine unbestreitbaren Vorzüge naher betrachtet. Was im folgenden geschehen soll.

2. Der ZX81 - besser als IBM ?

Seine Vorteile

Zunächst besitzt der ZX81 Vorzüge wie alle weitverbreiteten Heimcomputer: reichhaltiges Software-Angebot, viele Informationsquellen, viel Hilfe und viel Zusatz-Hardware. Bei kaum einem Kleinrechner gibt es ein so üppiges Angebot wie beim ZX81 (und Spectrum). Neben dem Üblichen - Speichererweiterungen und Druckeranschluß - sind die Grenzen kaum abzustecken. Man kann dem ZX81 Töne entlocken, ihn zur Steuerung von Meß- und Regelapparaten einsetzen, selbst Sprachein- und -ausgabe ist möglich. Wer genug Geld hat, kann sogar eine Floppy-Disk mit eingeschränkter Anwendung anschließen. Ja, in England gibt es gar eine Firma, die ein neues System aus dem ZX81 macht:.BASICARE stockt einen ganzen Turm von Modulen auf, um so ziemlich alles aus dem ZX81 herauszuholen, was im Rahmen seiner Grenzen möglich ist. Ein so "aufgeblasenes" System läßt allerdings die Frage nach dem Sinn stellen. Irgendwo hat der ZX81 seine natürliche Grenzen...

Aber zurück zum eigentlichen Rechner. Zweifellos kann der ZX81 seine Vorzüge nur ausspielen, wenn zumindestens eine brauchbare Tastatur und eine 16 K-Speichererweiterung angeschlossen sind. Die Folienklopferei ist der Hauptkritikpunkt am ZX81; und ohne 16 K-RAM läßt sich mit dem Winzling nicht allzuviel anstellen. Auch der preiswerte und für normale Listings ausreichende ZX-Drucker dürfte zur Minimalausstattung

für eine Anwendung zahlen.

Worin bestehen nun die Vorteile des Kleinpreis-Rechners? Man kann sie in drei Worten zusammenfassen: billig - leicht - vielseitig. Oder, wie es Practical Computing schon im Dezember 1982 formulierte: " Der Computer ist ein Meisterwerk technischer Entwicklung und ein Beitrag zum Massenmarketing." Dreh- und Angelpunkt ist das vielseitige, benutzerfreundliche BASIC der Sinclair-Maschine. Lassen wir zunächst ein kritisches englisches Magazin (Which Micro ? Dezember 1982) zu Worte kommen:

" Der ZX81 hat eine mächtige BASIC-Version.... Das BASIC ist sehr flexibel... Seine Hauptstärke ist der Syntax-Check, der bei Eingabe jeder Funktion erfolgt... Der ZX81 prüft jeden Tastendruck und entscheidet, ob die Zeile tatsächlich zur Interpretation geeignet ist... Die Wort-Eingabe ist ein sehr gutes Verfahren, um Programmieren zu lernen."

Was einen Chef-Kolumnisten der Zeitschrift, David Babsky, zu einem ebenso ketzerischen wie ironischen "Vergleichstest" von ZX81 und IBM PC veranlaßt hat.

ZX81 contra IBM PC

Um Vorzüge beurteilen zu können, muß man Vergleiche anstellen. Mr. Babsky hat es (in der April-Ausgabe 1983 von "Which Micro?", Seite 34 f.) unternommen, die robuste Gebrauchstüchtigkeit des ZX81 einem hochentwickelten Personalcomputer, dem IBM PC, gegenüberzustellen. Natürlich ist ein solcher Vergleich nicht ganz ernst zu nehmen, dürfte aber interessante Hinweise auf die SINCLAIR-Welt liefern. Gehen wir der Reihe nach vor; was passiert zunächst, wenn man eine IBM-Maschine programmieren möchte?

Da muß erst einmal ein ziemlich umfangreiches DOS (Disc Operating System) geladen werden. Wurde die komplizierte, teure Peripherie eingeschaltet, das System geladen und wartet der Benutzer auf seine BASIC-Eingabe - was passiert ? Der Mann vor dem Bildschirm wird aufgefordert, das Datum eingeben. Schon kommt mancher Neuling mit der Technik nicht mehr zurecht. Nach Mr Babsky's Beobachtung hat es einer nach zwanzig Minuten noch nicht geschafft....Kommentar: "Der IBM ist schließlich eine penible Maschine." Weiter - da gibt der BASIC-Programmierer seine Instruktionen ein und hofft nach möglicherweise hundert Zeilen Programm auf ein Ergebnis mit RUN. Was geschieht? Disks fangen an zu schnurren, mit viel Klick-Klack setzt sich die Maschine in Bewegung... und bleibt mit der Meldung "Fehler in Zeile 188" stehen. Der ZX81 kostet zwar keine 10.000,- DM sondern nur etwas über 100,- DM, ist da aber erheblich unempfindlicher. Man setzt sich hin, gibt eine BASIC-Zeile ein und wird höflich auf mögliche Syntax-Fehler hingewiesen. Ja, darüber hinaus zeigt der Sinclair sogar die Stelle an, wo man geirrt hat. Es gibt auch keine Tippfehler durch fehlerhafte Instruktionen, da mit einem Tastendruck ein ganzer Befehl abgerufen wird. Und das sauber mit Leerzeichen vorher und nachher, damit das Ganze schön lesbar bleibt. Kommentar von Mr. Babsky: " Das ist eine Eigenschaft, die ich in jedem Mikro installiert sehen möchte. Ich wünsche mir einen fehlersicheren Computer."

Weiteren Unfug beim Programmanlisten vermeidet der Sinclair ebenfalls. Beim teuren IBM PC erscheint nach LIST flackernd und in schnellem Zeilenzug das Programmlisting auf dem Bildschirm. Stellt man sich die Frage "was war bei Zeile 488 ?", kommt man auch mit STOP (durch

Control Break) nicht zum Ziel.

Beim ZX ist die Sache narrensicher. Nach LIST bekommt man das Programm von Anfang an gelistet, bis der Bildschirm voll ist (beim Spectrum wird höflich "Scroll?" gefragt). Möchte man weitersehen, tippt man LIST plus neue Zeile ein, und wieder wird nur das Gewünschte aufgelistet. Kommentar des "Which-Micro?"-Redakteurs: "Einfach, hinreichend, großartig". Er mutmaßt, daß IBM und andere Maschinen entwerfen, bei denen der Computer im Mittelpunkt steht, ohne daß man sich Gedanken macht, wie der Mensch mit all der Technik fertig werden soll.

Ein weiterer Punkt, zwar Marginalie, aber immerhin etwas, das für den ZX81 spricht - seine Grafiksymbole. Warum werden die Symbole nicht sauber auf der Tastatur gleich mitgegeben wie bei der Sinclair-Maschine?

All dies wird häufig mit dem Argument Geschwindigkeit totgeschlagen. Aber zumindestens in Teilbereichen zeigt der ZX81 gegenüber dem IBM PC gute Seiten. Beispielsweise ist die INKEY$-Funktion (gemäß Mr. Babsky) dreimal so schnell wie die der teuren 16-Bit-Maschine. Die gleiche Routine zur Cursor-Abfrage (und -steuerung) läuft beim ZX erheblich rascher ab. Um auf das bekannte Argument "Laufzeit" weiter einzugehen, hier schneidet der ZX81 im "Vergleichstest" gegenüber dem PC gar nicht schlecht ab. Unbestritten ist die Sinclair-Maschine ein langsamer Arbeiter; aber die Unterschiede sind bei spezifischen Aufgaben, wie sie der bekannte Finsbury Benchmark Test stellt, nicht so gravierend, wie man zunächst annimmt. Babsky-Kommentar: "...zeigt die winzige Sinclair-Maschine eine erstaunlich gute Figur gegenüber dem mächtigen IBM." Und weiter:"... obwohl der Vergleich unsinnig erscheint, zeigt er, daß die Laufzeit eines BASIC-Interpreters weniger vom CPU-Entwurf abhängig ist, als man denken mag." Wen die einzelnen Ergebnisse interessieren, möge sich die nachfolgende Tabelle zu Gemüte führen (wie sie die Februar-Ausgabe von "Which Micro?" 1983 aufführt):

	IBM PC	ZX (FAST)	Abweichung zum PC
Test 1	12,5	43	+ 244 ./.
Test 2	5,6	7,1	+ 27 ./.
Test 3	11,1	16,4	+ 48 ./.
Test 4	12,4	16,1	+ 30 ./.
Test 5	13,2	18,9	+ 43 ./.
Test 6	17,0	50,8	+ 199 ./.
Test 7	37,7	71,9	+ 91 ./.
Test 8	31,8	232,9	+ 633 ./.

Um es nochmal herauszustellen, der ZX81 ist und bleibt eine langsame Maschine. Man darf nicht vergessen, daß der Syntax-Check zu ungunsten der Laufzeit geht. Vielseitiges und komfortables BASIC hat eben seinen Preis. Insbesondere für umfangreichere Programme, wie sie das Buch bringt, muß die Devise gelten: Sie können drauf warten ....

3. ZX-Eigenheiten

Pro und Contra

Nach soviel Lob für den Winzling aus England möchten wir wieder auf den Boden realer Tatsachen zurückkehren. Betrachten wir die Eigenheiten des ZX81, wo neben Glanzlichtern auch Schatten zu finden ist. Neben dem anwenderfreundlichen BASIC-Befehlssatz sollen andere Dinge

kritisch angetippt werden. - Da ist einmal die lange Ladezeit und die Anfälligkeit großer Programme. Generell kann man sagen, daß beim ZX81 so ziemlich alles (außer der INKEY$-Tastenabfrage) langsam geht. Bei einem Laufzeittest der Zeitschrift PWC vom Herbst 1983 schnitten bei Heimcomputern nur ORIC, TI99A, ATARI 400/800 schlechter ab. Die Spitzengruppe bildete der BBC-Mikro, das Mittelfeld (noch weit vor dem ZX81) bestritten Produkte wie VC 20 und DRAGON 32. Kurzum der ZX81 liegt bei Laufzeittests abgeschlagen auf hinteren Plätzen.

Das ist zwar der gewichtigste Nachteil des ZX81, aber bei weitem nicht der einzige. Es gibt genug zu kritisieren, vom Bildschirmflackern bei FAST bis hin zur bekannten Ladeschwierigkeit vom Recorder. Um die Hauptminuspunkte kurz aufzuführen, ehe wir uns Positiverem zuwenden, sei auf Folgenes hingewiesen:

- Die Folientastatur ist schlichtweg indiskutabel. Man sollte sich bereits bei Kauf richtige Tasten als Extra anschaffen. (Gleiches gilt für die kummerliche 1 K RAM Grundausstattung)

- Das Editieren von Programmzeilen liegt unter dem allgemeinen Standard. Wenn schon Corsorbewegungen "up and down" möglich sind, sollten sie (statt umständlichem Zeilen-EDIT) im Programm selber eingesetzt sein.

- DATA und RESTORE fehlen beim ZX81 gänzlich (Der Spectrum hat solches endlich auch)

- Die Sinclair-Blockgrafik ist ebenfalls kein Grund zum Jubeln. Wenngleich man durch Zusätze beim ZX81 Hi Res erreichen kann, es bleibt die Grafik eine schwache Seite des ZX81.

- Womit wir bei der 32-Zeichendarstellung pro Zeile wären. Andere bieten mehr, ja sogar als Option für Monitore die Möglichkeit, auf 80 Zeichen pro Zeile zu gehen.

- Neben weiteren Marginalien (fehlendes ON-Statement, schlechtes Scroll, Werteinitialisierung mit LET etc.) bleibt ein gravierender Punkt noch zu vermerken: Der ZX81 besitzt keine normale Schnittstelle nach außen. Problemlos kann nur der Sinclair-Drucker angeschlossen werden, alles andere kostet einen Haufen Geld und Aufwand.

Nun zu den positiven Seiten. Auch hier seien die wesentlichen Punkte stichwortartig aufgezählt; Erläuterungen zu einzelnen Befehlen finden sich bei den folgenden Programmen bzw. unter dem Kapitel "TIPS & TRICKS".

- Neben der äußeren Handlichkeit und den Haupttrümpfen Syntax-Check und Ein-Tasten-Befehlseingabe sind andere Dinge schlechthin ohne Konkurrenz. Da ist einmal die Möglichkeit, auf den Sinclair-spezifischen Drucker direkt mit LPRINT und LLIST zu gehen. Auch die Kopie einer Bildschirmseite mit COPY ist zweifellos eine unübliche, nützliche Seite des ZX81.

- Die freie Programmierung von Druckstellen mit PRINT AT und TAB dürfte ebenfalls positiv in dem Zusammenhang zu bewerten sein.

- Noch größere Freiheit hat der Hobbyprogrammierer bei Variablen und Strings. Variablen-Namen können frei gewählt werden und sogar Leerzeichen enthalten. Gerade für Anfänger, die sprechende Namen verwenden, ist das eine nützliche Sache. Strings sind zwar dem Namen

nach auf einen Buchstaben (mit folgendem $-Zeichen) beschränkt, aber sowohl in Länge wie in Aufteilung frei.

- Bleibt schließlich noch, auf die vielfältigen Möglichkeiten indirekter Adressierung hinzuweisen. Sowohl Sprungziel wie Wertezuweisung kann durch Formeln, durch "Einbau" anderer Variablen und durch logische Operatoren bestimmt werden.

- Gerade die vielfältigen Manipulationsmöglichkeiten mit der Sinclair-spezifischen Vergleichslogik verdienen, hervorgehoben und näher beleuchtet zu werden.

BASIC-Eigenschaften

Besinnen wir uns auf Stärken, die bei Anwendungsprogrammen eine erhebliche Rolle spielen. Das Sinclair-Basic weist nämlich bedeutsame Abweichungen zum Microsoft-Standard auf, insbesondere dort, wo Strings behandelt werden. Um einen Experten von "Elektronics & Computing Monthly", Mike James (vgl. Oktober-Ausgabe 1983, Seite 78 f.) zu zitieren: "Meiner Meinung nach ist die Art, wie ZX-BASIC Zeichenketten handhabt eine seiner besten Eigenschaften - seine Methode sollte Standard werden". Schauen wir uns einige Sinclair-Stärken des BASIC näher an, als da sind die Möglichkeit, *logische Vergleiche* anzustellen und *Stringgrößen* zu verarbeiten. Sinclair-BASIC handhabt im extremen Maße die Methode, einem Vergleichsausdruck einen einzigen Wert (besser: Wertigkeit) zuzuordnen. Hochentwickelt ist beim ZX81 die Möglichkeit, Formeln zu erkennen und auf Gültigkeit (mit dem Vergleichsergebnis 0 = falsch und 1 = richtig) zu prüfen. Es gibt nicht weniger als vier verschiedene Operationen, Wertigkeiten auszudrucken:

a) als arithmetischer Ausdruck z.B. A + 2 * PI,

b) als relationale Beziehung, z.B. ergibt A <> 2 den Wert 0, wenn sie falsch ist und 1, wenn sie richtig ist,

c) Gleiches gilt für logische Operationen, z.B. A OR B
(näheres siehe Manual, Seite 71)

d) die Möglichkeit, Zeichen und/oder Zeichen und Zahlenausdrücke miteinander zu verbinden, z.B. Addition von Zeichen A + VAL "1"; oder in der Form Zeichen zu Zeichen A$ + "1".

Zusammenfassend sagt Mike James daher: "ZX-BASIC selektiert seine Datenwerte und Datentypen, um es möglich zu machen, Ausdrücke so frei wie möglich miteinander zu vermischen." Gerade die logische Wertigkeit von wahr = 1 und falsch = 0 ermöglicht vielfältige Selektionen. A AND B heißt im Klartext: A wird erkannt, wenn B richtig ist; umgekehrt liefert der Ausdruck den Wert 0, wenn B falsch ist.

Die Mächtigkeit des Sinclair-BASIC geht sogar so weit, Vergleichsoperationen für Strings zuzulassen, wobei allerdings nur ein AND-Vergleich möglich ist. A$ AND B heißt im Klartext: A$ bleibt gultig, wenn B wahr ist; ist B = 0 (d.h. falsch), ist das Ergebnis ein Leerstring. Da die Werte 0 und 1 bzw. Leerstring oder kein Leerstring direkt durch Vergleichsoperation erzeugt werden konnen, kann man unterschiedliche Datentypen beliebig mischen; womit der ZX81 anderen Rechnern weit uberlegen ist. Ein kleines Beispiel soll das veranschaulichen. Der folgende Vergleich auf großer und kleiner mit Zuweisung dieser beiden Worte durfte herkömmlicherweise wie folgt programmiert werden:

```
IF B  > 7 THEN LET A$  =  "GROESSER"
IF B  < 7 THEN LET A$  =  "KLEINER"
```

In Sinclair-BASIC kann das Ganze in einer einzigen Zeile untergebracht werden:

```
LET A$  = ("GROESSER" AND B > 7) + ("KLEINER" AND < 7)
```

Damit sind aber die "Stringstärken" des ZX81 bei weitem nicht erschöpft. Vielmehr gibt es die Möglichkeit, Zahlen in Strings speicherplatzsparend unterzubringen und hin und her zu konvertieren. Insgesamt sind drei Varianten anwendbar, mit den entsprechenden Befehlsklauseln zu arbeiten:

a) Konvertierung einer Zahl in einem String mit STR$:

```
LET  A$  =  STR$ N
```

b) Konvertierung einer Stringzahl zurück in ihren numerischen Wert mit der VAL-Klausel:

```
LET N = VAL A$
```

c) Verwenden der Codezahlen von Zeichen mit den Klauseln CODE (d.h. Zahlenwert eine Stringposition) und CHR$ (d.h. Umwandlung einer Codezahl in ein Zeichen).

Neben dem bereits erwähnten Vorzug indirekt zu adressieren (z.B. GOTO A, GOTO A * 100, GOTO VAL A$ etc.) bleibt schließlich ein Leckerbissen des Sinclair-BASIC gesondert aufzuführen. Der Rechner aus England kann etwas, was sonst nur Große beherrschen, nämlich die dynamische Speicherplatzbestimmung. Möchte man beispielsweise eine Zahlentabelle dimensionieren, ohne zu wissen, wie groß sie einmal sein wird, kann man zunächst indirekt DIM A (N) definieren. Sinnvollerweise wird im Programmablauf nun der Benutzer mittels einer INPUT-Eingabe gebeten, die Zahl der Elemente selbst zu bestimmen. Liegt der Befehl "INPUT N" vor der Zeile mit dem DIM-Befehl, läuft das Ganze reibungslos ab. Erst "at run-Time" wird der Array in seiner endgültigen Größe festgelegt!

Es ist daher Mr. James nur zuzustimmen, wenn er alle genannten Eigenschaften des Sinclair-BASIC so bewertet: " Wie diese Eigenschaften zusammenpassen, um so eine ansprechende und sparsame Sprache abzugeben, das macht ZX-BASIC für Anfänger und Experten gleichermaßen geeignet."

Kapitel II:

Auf dem Weg zur Anwendung

1. Jenseits von Startrek

Ernsthafte Anwendungen

Ernsthafte Anwendungen bewegen sich bei einem Mikrocomputer jenseits von Sternenkrieg und Bombengeballer. Spiele sollten nur den Einstieg bilden für das, was sich alles mit dem ZX anstellen läßt. Zwar wird das Spieleangebot immer raffinierter und der Konkurrenzkampf der Spielehersteller immer erbitterter, aber der Rechner ist zu etwas Besserem gut als zur simplen Augenunterhaltung. Ernsthafte Anwendung bedeutet, daß man die Möglichkeiten des Rechners für sinnvolle Dinge nutzt. Da gibt es eine ganze Menge von simpler Haushaltsbuchführung über allerlei Datenverwaltungsaufgaben bis hin zur Steuerung der gesamten Heizungsanlage. Je umfangreicher die Aufgabenstellung wird, um so stärker muß der ZX arbeiten und um so mehr Speicherplatz wird verbraucht. Wirklich schnelle Programme bekommt man nur in Maschinencodierung (was bekanntlich sehr mühselig ist und umfangreiche Kenntnisse voraussetzt); da die zentrale Recheneinheit (CPU) nur insgesamt maximal 64 K adressieren kann, sind dem ZX spätestens hier natürliche Grenzen gesetzt.

In den Mittelpunkt der folgenden Ausführungen ist die eigentliche Programmierung in BASIC gestellt, wobei wir uns der Beschränkungen des ZX bewußt sind. - Soviel zur Absicht, nun zur Methodik. Hat man sich eine bestimmte Aufgabe vorgenommen, z.B. in Zukunft seine Lottotips aus dem Rechner zu beziehen, so wird man den ZX nur effizient nutzen, wenn man wohlüberlegt ans Werk geht. Statt wild durcheinander zu programmieren (sogenannte Spaghetti-Programmierung) sollte man sich eine gewisse "Technik" aneignen. Diese in Verbindung mit den unbestreitbaren Stärken des Sinclair-BASIC ergibt erst im Laufe der Zeit so etwas wie ein gelungenes Programm. In diesen Zusammenhang gehört das Stichwort Modultechnik - was ist ein Modul?

Modultechnik

Der wesentliche Aufbau sieht im einzelnen etwa wie folgt aus:

A Eingang: Übergabe von Parametern, diese sichern, ausrichten oder retten.

Initialisierung der Anfangswerte (am einfachsten, häufigsten durch eine FOR NEXT-Schleife)

B Verarbeitungsteil:
Abfrage des Endkriteriums bzw. einer vorzeitigen Abbruchbedingung.

Sprung zur Abschlußroutine

Eigentliches Datenhandling mit Sprüngen, Abfragen, Tests von Bedingungen, Prüfungen und "Checking", Formatierung und Konvertierung von Daten.

Ggf. Aufruf weiterer Module/Routinen

Aufbereitung von Daten und optische Darstellung

C Ausgang: Übergabe von Parametern

Sichern von Werten und weiterzuverarbeitenden Daten

Vorbereiten der Parameterübergabe und RETURN

Normalerweise werden die Module gleichrangig sein und als abgrenzbare und abgegrenzte Einheiten selbständig operieren. Der Programmaufbau wird in etwa, wie im Verlaufe dieses Buches noch zu beobachten ist, folgendes Bild ergeben:

Grundsätzlicher Modulaufbau

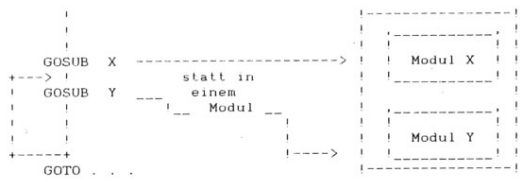
Häufig wird aber eine bestimmte Routine von mehreren Programmpunkten angesprungen. Es gibt also gemeinsame Hilfsroutinen. Andererseits kann die Verbindung von Modulen nach der "Treppenregel" vom Allgemeinen zum Speziellen so aussehen:

Modul - Aufrufkette

Ein Modul ruft weiteres (abhängiges) Modul auf, das wiederum (immer stärker verästelte) kleinere Programmteile aufruft. Selbstverständlich sollte man das Ganze nicht übertreiben. Immerhin kostet jeder GOSUB einen gewissen Rechenaufwand. Dieser ist zwar nicht besonders groß, jedoch wächst im Laufe der Zeit der Verwaltungsaufwand für den Maschinenstapel (im "Innern" des ZX) erheblich an. Wobei in diesem Zusammenhang eine andere Frage auftaucht; Module mit GOSUB oder GOTO ?

Natürlich ist die Verwendung von GOSUB der Normalfall. Überall dort, wo sich beim Herausarbeiten der Programmlogik voneinander abgrenzbare selbständige Befehlsfolgen ergeben, wird man diese mit einem GOSUB-Befehl aufrufen und abarbeiten. Nicht nur, daß dieser relativ bequem zu handhaben ist, auch schützt er davor, den Rücksprung versehentlich zu vergessen. Aber er geht auch mit GOTO. Ja, es gibt sogar Fälle, wo sich ein GOTO bzw. die Verbindung von GOSUB mit GOTO nicht umgehen läßt. Möchte man es vermeiden, umfangreiche Werte zu retten, zu übergeben und neu abzuspeichern, wird man mit einem simplen GOTO meist besser fahren. Auch die Abprüfung spezieller Bedingungskonstellationen zwingt geradezu zur Verwendung von GOTO. Allerdings können beide Befehle gut miteinander kooperieren, wie das folgende Schaubild veranschaulicht:

Modulverbindung mit GOTO



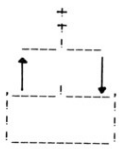
Ein Beispiel für die Aufteilung von aufeinanderfolgenden Routinen in zwei statt einem Modul befindet sich übrigens im Textprogramm (Programme: WORD). Statt zwei nahe beieinanderliegende Routinen zu einem Modul zusammenzufassen, ist es in einem solchen Fall günstiger, beide mit zwei GOSUBS hintereinanderzuschalten und dann von verschiedenen Seiten getrennt anzuspringen.

Darstellungsformen

Es bleibt noch die Frage nach der Darstellungsform. Die bekanntesten Instrumente, um so etwas wie Programmlogik wiederzugeben, sind einmal die Nhassi-Schneidermann-Diagramme und zum zweiten die üblichen DIN-Norm-Symbole. Beide haben spezielle Vor- und Nachteile. Die Nhassi-Schneidermann-Diagramme sind zwar übersichtlich, vollständig und auch gut nachzuvollziehen, aber extrem papieraufwendig. Ähnliches gilt für die üblichen DIN-Norm-Symbole mit Kästchen und Rauten, die bestenfalls am Anfang eine gewisse Berechtigung haben. Natürlich wird im Laufe der Zeit jeder Hobby- und Profiprogrammierer seinen eigenen Stil entwikkeln. Wir möchten an diesem Punkt, quasi als Vorschlag, ein paar Beschreibungselemente einführen, die in der Folge zur Wiedergabe der Programmlogik verwandt werden. Halten wir uns kurz vor Augen, welches notwendige Elemente der Darstellungslogik sein müssen.

Die "Atome" jeder Logik sind immer Verzweigung und Sequenz. Verzweigung heißt Sprung in Abhängigkeit von einer Bedingung. Sequenz heißt festgelegte, hintereinander ablaufende Instruktionsfolge. Anders ausgedrückt - Grundelemente einer "Logik" sind immer Frage und Statement; andere Operationen wie Wiederholung und Zusammenfassung stellen nichts anderes als Ausgestaltung dieser beiden Grundformen dar. Natürlich wird zur Verzweigung, d.h. dem Abprüfen einer Bedingung, der Sprung, d.h. die Bestimmung einer Destination, gehören. Folglich kommt man im wesentlichen, und das beweist auch die im folgenden dargestellte Logik der Programme, mit Sequenz, Verzweigung, Sprung und Zusammenfassung von Befehlsfolgen als Unterprogramm und Modul aus. Dabei gilt folgendes Schema, wie es der Herausgeber entwickelt hat und dem Leser vorstellen möchte:

Darstellungselemente von Programmlogik



Sequenz

Verzweigung

Sprung

Unterprogramm/Modul

Links in der Grafik erscheint jeweils mit einem Kürzel, was geprüpft wird, welche Variable abgefragt wird etc. Rechts hingegen erscheinen in abgekürzter (BASIC-) Form die einzelnen Verarbeitungsschritte. Einrückungen, jeweils immer nach rechts wandernd, machen den Logikzweig von links nach rechts lesbar. Natürlich sollen nie an ein und derselben Stelle Befehle nebeneinanderstehen, so daß man nicht weiß, welchem "Strang" sie zuzuordnen sind. Aber wenn man die entsprechenden Zweige auseinanderzieht, hat die noch im einzelnen vorgeführte Darstellungsform entscheidende Vorteile:

Sie ist klar überschaubar und benötigt wenig Platz. Zudem kann meist von den entwickelten Sequenzen direkt abprogrammiert werden.

Im folgenden wollen wir den Schritt zur ernsthaften Anwendung mit dem ZX zunächst spielerisch angehen. Erst allgemein und ausführlich, dann komplexer und kompakter soll entwickelt werden, wie man zu ernsthaften Programmen kommt. Dabei ist die vorgestellte Lösung jeweils als Vorschlag, als Anregung und als Idee zu verstehen.

Das Buch ist demzufolge auch nicht dazu gedacht, einfach blind kopiert zu werden, sondern soll Hilfe zum Selbstprogrammieren bieten. - Fangen wir zunächst da an, wo andere ZX-Bücher aufhören, nämlich bei einem Spiel. Allerdings nehmen wir uns nicht irgend ein gefräßiges Ungeheuer, das es mit Kraftpillen zu bekämpfen gilt, vor, sondern bleiben beim Glücksspiel und seiner Auswertung. Zunächst soll gezeigt werden, wie man beim Lotto allerlei Nützliches aus dem Rechner gewinnen kann. Dabei sind vorläufig Sortierungen und ähnlich komplizierte Dinge ausgespart worden.

2. Spielzahlen aus dem Rechner

Ein einfaches Lottoprogramm

Eine Idee zu haben, ist meist einfacher als ein Programm. Idee war es, Zufallszahlen für Lottotips zu bekommen. Da der ZX eine eingebaute Randomfunktion hat, ist das Ganze nicht weiter schwierig. Aber doch halt, nun stocke ich schon - eigene Zahlen sollen mit eingemischt werden können. Das heißt, neben den Zufallszahlen sollen auch bestimmte Zahlen (persönliche Glückszahlen, Geburtsdaten etc.) vorgegeben werden. Und damit man nicht alles wieder vergißt, sollen die tatsächlichen Ziehungen festgehalten werden und abrufbereit zur Verfügung stehen. Worauf man natürlich sinnigerweise gleich eine kleine Häufigkeitsstatistik anschließen könnte, um Favoriten und "Zurückgebliebene" zu erkennen... Schon ist aus einer einfachen Idee ein ausgewachsenes Programm bzw. eine stattliche Programmvorgabe geworden.

Gehen wir der Reihe nach vor. Insgesamt werden im folgenden drei

Programme vorgestellt, wobei die Skala von einfachen Zufallszahlen-Programmchen bis hin zur richtigen "Logik" mit allem Drum und Dran reicht. Voraussetzung war, keine allzu komplizierten Dinge, wie Sorts, einzuführen und im übrigen die Sinclair-Spezifika auszunutzen. Und das ist in erster Linie die "starke" Stringverarbeitung des kleinen englischen Rechners. Gehen wir die Sache Schritt für Schritt mit allen drei Programmen an. - Was war Ziel und Lösung bei "Lotto 1" ?

Das Problem bei gezogenen Zahlen ist, Doppelziehungen zu vermeiden. Man muß bei der Generierung von Zufallszahlen verhindern, daß eine bereits "ausgespielte" Zahl nochmals vorkommt. Zum zweiten sollte - hier noch zu Übungszwecken - möglichst wenig Speicherplatz beansprucht werden.

Wenn man bedenkt, daß jede Zahl ohne den Variablen-Namen fünf Bytes kostet, sehen wir schnell, wie ein paar hundert Bytes bei 49 Zahlen verbraten sind. Möchte man noch (später) die gezogenen Zahlen speichern, zeigt sich, was der Komfort eine Gleitkommadarstellung kostet... Speicherplatz!

Die Lösung bei "Lotto 1" sieht denn im Programm vor, daß gezogene Zahlen ein Kennzeichen erhalten, und zum zweiten, daß sie statt als Zahlentabelle in einem String abgelegt werden. Der Weg, um zu einem Zahlentip fürs Lotto zu kommen, läßt sich im einzelnen mit wenigen Worten umschreiben: Direktplazierung in einem 49 Zeichen-String (Z$); die gekennzeichnete Stelle ist die Information über die Zahl selber. Würde beispielsweise das 22. Element (mit "*") gekennzeichnet, so ist dadurch die Information gespeichert, daß der Rechner die 22 als Tip-Zahl ausgewählt hat. Beim Ausprinten brauchen nur noch die Sterne eingesammelt zu werden. Jeder FOR-NEXT-Durchlauf ergibt bei "* = GEFUNDEN" die entsprechende Tipzahl. Einfacher geht's nicht...

Lotto 2 und 3

Lotto 2 ist anpruchsvoller. Da müssen vorgegebene Zahlen mitverarbeitet werden. Die Zufallszahlen sind mit den Vorgabezahlen zu mischen. Werden diese fest auf jede Tipreihe vergeben, ist die Aufteilung problemlos. Was ist aber bei variabel zu verteilenden Vorgabezahlen?

Nun, da entsteht ein Aufteilungsproblem, das bei EDV-Optimierungsaufgaben im übrigen als Allokationsproblem bekannt ist. Beim Programm "Lotto 2" hat es damit folgende Bewandtnis:

Da die Vorgabezahlen nach Anzahl und Wert bekannt sind, ebenso wieviel Tipvorschläge zu liefern sind, müssen die Vorgabezahlen richtig zugemischt werden. Richtig bedeutet, alle Zahlen sind zu verbrauchen, es darf keine übrigbleiben. Richtig heißt aber auch, sie müssen gleichmäßig zugeteilt werden und nicht etwa alle 4 Vorgabezahlen allein auf die letzte Tipreihe.

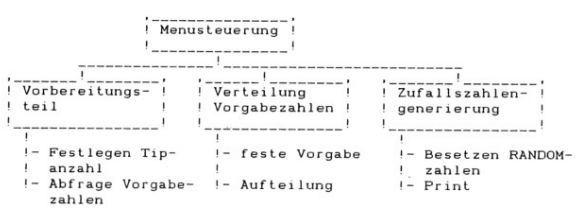
Einfach ist eine solche Lösung immer dann, wenn die Teilerrelation glatt aufgeht: Sollen etwa vier Vorgabezahlen auf vier Tipreihen verteilt werden, erhält jede Reihe eine Zahl. Es müssen nur noch fünf Zahlen mit der Randomfunktion bestimmt werden. Anders bei ungeraden Relationen. Wie sieht die Lösung da aus ?

Die entscheidende Aufteilungsroutine (ab Zeile 180) fügt neben den üblichen Checks das Kennzeichen "+" (für Stelle gleich Zahl) in den Zahlenstring ein. Das geschieht dergestalt, daß am Ende alle Vorgabezahlen verteilt wurden. Im einzelnen sieht die Sache so aus: Die als

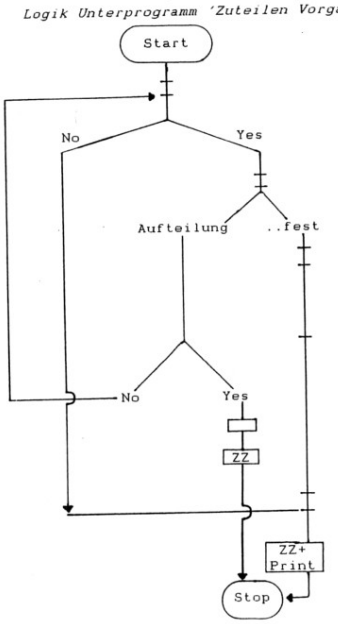
Characters im Vorgabezahlenstring V$ gespeicherten Zahlen werden nacheinander von links nach rechts (aufsteigende Eingabefolge vorausgesetzt!) abgehackt. Der String verkürzt sich so von links nach rechts (Routine ab Zeile 260). Wann eine neue Zahl dran ist, steuert die Variable B, welche die Schrittweite des "Abhackens" bestimmt (ab Zeile 193, wo der GOSUB-Aufruf von 260 liegt). Die Feinlogik ist in einem eigenen Schema aufgezeigt, sie arbeitet relativ simpel. In aller Kürze beinhaltet sie folgende Schritte, Einzelheiten möge man selber anhand des Schaubildes durchspielen: Die Teilerreste aus dem Quotienten "Anzahl Vorgabezahlen: Anzahl Tipreihen" werden bei jedem Durchlauf (ab Zeile 187) so lange fortgezählt, bis eine neue Ganzzahl erreicht ist. Genauer - bis die Differenz der fortgezählten Häufigkeiten (V * N) als Ganzzahl angibt, wieviel weitere Vorgabezahlen in diesem Durchlauf beizumischen sind. Ein einfacher, aber nach unseren Tests wirksamer Zuteilungsalgorithmus, auf den man nicht sofort kommt.

Wie sieht nun die übrige Programmlokik aus? Dazu soll das folgende Schema dienen, das Aufbau und Wirkungsweise des Programmes "Lotto 2 " wiedergibt:

Programmaufbau 'Lotto 2'



Logik Unterprogramm 'Zuteilen Vorgabezahlen'



Vorroutine:
Eingabe Anzahl Tips
(incl. Check)

Vorgabezahlen abgearbeitet ?
(d.h. V$-String leer)

Check VZ + Abspeichern V$

Aufteilen VZ oder VZ fest ?

Prüfung VZ auf Plausibilität

Besetzen Zahlenstring Z$ mit festen Vorgabezahlen
(= Kennzeichen '+')

Prüfung Aufteilungsplausibilität
VZ aufteilbar ?

Bestimmen 'Zuteilung'
Auffüllen mit VZ's

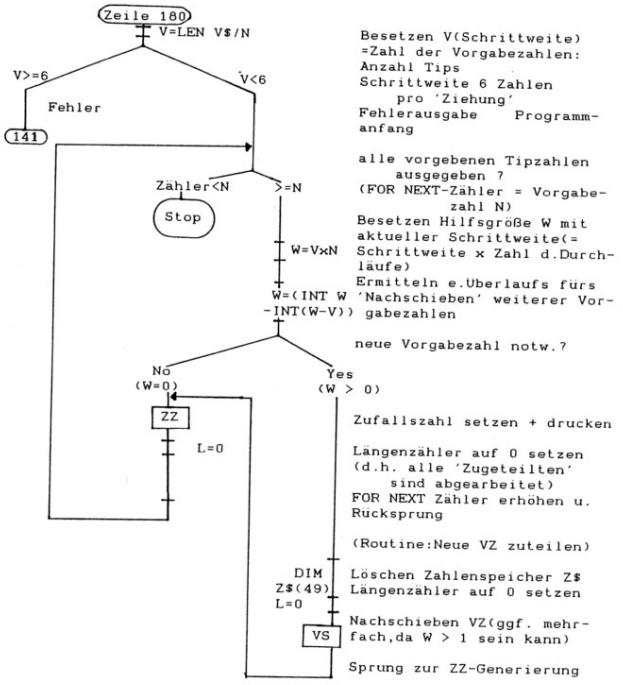
Aufruf Modul 'Zufallszahlen-Generieren'

Löschen String

ZZ Generieren (unter Beachtung fester VZ's)

Ausdruck 'Tip'

Feinlogik 'Aufteilen Vorgabezahlen'



Zum Programm sind noch zwei Dinge anzumerken:

1. Das Programm checkt vieles ab, nur nicht, wenn identische Vorgabezahlen eingegeben werden. Wird also beispielsweise zweimal die Zahl "1" vorgegeben, können vom Rechner bei fester Vorgabe pro Tipreihe nur vier Zahlen zugemischt werden, so daß sich statt sechs nur fünf Tipzahlen ergeben (!).

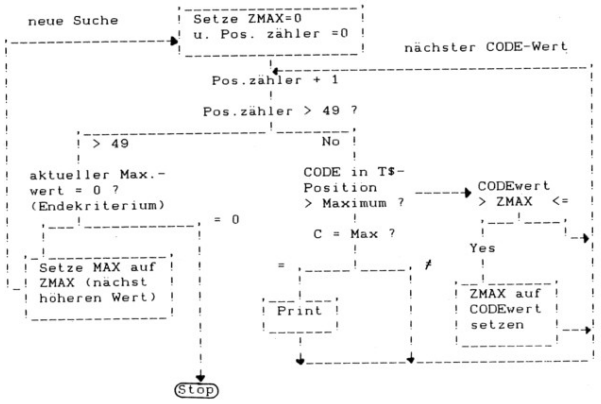
2. Die Addition in Zeile 189, wo es statt INT W heißt INT (W + .0001), dient dazu, den Integerfehler des Sinclair abzufangen. Die interne Zahlendarstellung und Konvertierung führt zu Ungenauigkeiten, die sich hochschaukeln. Der Wert 0,2 z.B. ergibt dann nicht als Ganzzahl "2", sondern nur "1", da er intern als 1,99999 abgelegt ist!

Das Programm "Lotto 3" bietet zusätzlich so etwas wie eine Buchführung. Natürlich kann man es mit der Zufallszahlenerzeugung (aus Lotto 1 oder Lotto 2) zusammen binden. Im Vordergrund stand jedoch bei "Lotto 3" die Eingabe tatsächlicher Ziehungszahlen (mit Wochennummer). Die Ziehungszahlen werden fortgeführt durch eine Wochentabelle, die Lesen und Ändern ermöglicht. Werden neue Zahlen präsentiert, können sie automatisch an die letzte Wocheneingabe angefügt werden. Aber damit nicht genug.

Als Anlistung kann nicht nur jede Wochennummer oder alle bis dahin eingegebenen Wochenzahlen vorgenommen werden, sondern auch die Häufigkeit. Sowohl eine beliebige Woche wie auch die Häufigkeit der Zahlen ist mit "Lotto 3" anzeigbar.

Der Lösungsweg, um sowohl ein Fortschreiben der einzelnen Wochenzahlen (gespeichert in Array W$) als auch einen "sortierten" Häufigkeitsprint zu ermöglichen, läßt sich wie folgt umschreiben: Zunächst wurde Speicherplatz dadurch gespart, daß die maximal 52 x 6 Zahlen nicht in einer Zahlentabelle, sondern in einem String abgelegt wurden. Das ist noch relativ einfach, da die möglichen Zahlenwerte von 1 bis 49 alle als Code-Zahlen transformiert werden und somit nur ein Byte verbrauchen. Schwieriger war es schon, eine Sortierung nach Häufigkeit zu erreichen, ohne komplizierte Sort-Algorithmen zu bemühen. Der Häufigkeitsausdruck ist zwar langsam, erfüllt aber seinen Zweck. Selbstverständlich wird die Häufigkeitstabelle (H$) ebenfalls mit Code-Zahlen (= tatsächliches Vorkommen einer Zahl bis dahin) als String geführt. Allgemein arbeitet die Logik bei der Ausgabe der häufigsten Zahl ganz einfach mit dem Vergleich des jeweils relativ höchsten Wertes. Man schraubt das Feld "MAX" auf den höchsten Wert der Tabelle hoch, um es dann nach Ausgabe der häufigsten Zahl mit der nächsthäufigsten zu füllen. Und so fährt man fort, wie die nachfolgende Logik zeigt:

Logik 'sortierte' Stringwertausgabe (Lotto 3)



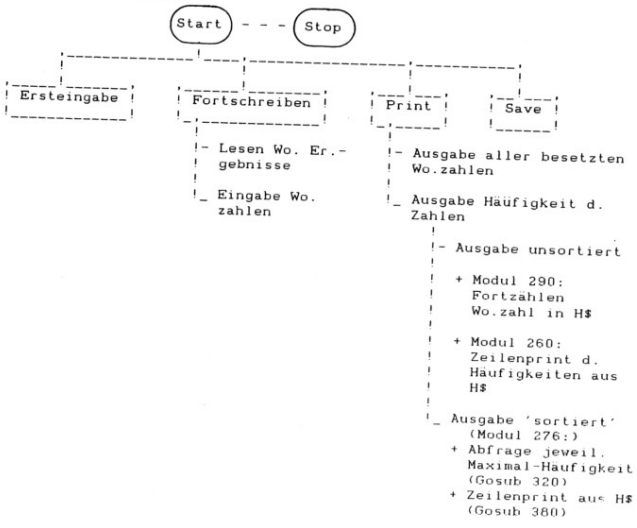
Schließlich könnten wir noch eine kleine Verbesserung einfügen, um die Zahl der Durchläufe zu vermeiden. Da in den Häufigkeiten meist Lücken vorkommen, d.h. die Zahl "17" kommt meinetwegen siebenmal vor und die nächsthäufigere Zahl "3" nur fünfmal, entstehen unnütze Durchläufe. Man könnte nun die Step-Variable (in der Routine Zeile 330) während des Durchlaufes ändern, was leider nicht funktioniert. Um die nächste Häufigkeit abzufragen, mußte vielmehr eine eigene Schleife gebaut werden (in Routine 330).

Im übrigen ermöglicht "Lotto 3" neben der Ausgabe von Häufigkeiten (sortiert oder unsortiert) die übliche Buchführung. Alle Wochenziehungen können entweder zusammen oder für eine bestimmte Woche ausgegeben, fortgeschrieben und schließlich gespeichert werden.

Bevor man etwas fortschreibt, möchte man es in der Regel erst sehen. So können denn alle besetzten Wochen angedruckt werden oder die jeweiligen Häufigkeiten der bisher vorgenommenen Zahlen. Wie man sieht, eine ganz komfortable Orientierung des Lottospielers, ehe er sich darangibt, irgendwelche Zahlen zu tippen...

Nachdem der eigentliche Leckerbissen des Programms, die Logik einer sortierten Stringausgabe ohne Sort-Prozedur dargestellt wurde abschließend noch der allgemeine Programmaufbau:

Programmaufbau Lotto 3



Kapitel III :

Wort & Text

1. Besonderheiten bei Worten

Variable Daten

Worte zeichnen sich dadurch aus, daß sie variabel lang sind. Dem kommt die Speicherung in Zeichenketten, sogenannten Strings, entgegen. Sie ist beim Sinclair besonders gut ausgebaut. Im Gegensatz zu anderen Heimcomputern kann die Positionierung innerhalb eines Strings elegant mit der TO-Klausel vorgenommen werden. Dadurch entfällt die bei anderen Rechnern umständliche Plazierung von Zeichen links- , rechts- und mittelbündig. Darüber hinaus gibt es beim ZX auch keine Längenbeschränkung, wie sie andere Rechner (beispielsweise Maximallänge = 256 Zeichen) besitzen. Mit dem Plus-Zeichen kann zudem ein bestehender String beliebig verkürzt, verlängert und umgruppiert werden. Es gilt, sich bei der Verarbeitung von Wort und Text auf diese unbestreitbaren Vorzüge des ZX zu besinnen.

Von der Bedarfsseite her gesehen, bedeutet jede Verarbeitung von Wort und Text, daß eine Programmlösung problemloses Einfügen, Löschen, Ändern neben dem üblichen positionierten Lesen ermöglichen muß. Man kann diese Funktionen über eine umfangreiche "Buchführung" realisieren: Die jeweilige Stelle innerhalb eines Textfiles wird durch laufendes Mitführen von Seiten-, Zeilen- und Positionsangaben lokalisiert. Statt jedoch auf die beliebte Tabellenlösung mit viel Rechnerei und starrer Eingabebeschränkung zu verfallen, gibt es bessere Ansätze. Statt Texte als 32-stellige Arrays zu behandeln, ist es beim ZX passender, *einen einzigen Textstring* zu verwenden. Er kann, da variabel, beliebig manipuliert werden. Warum keine Array-Lösung? Einerseits kommt man mit einem Textring der schnellen Stringbearbeitung des ZX entgegen (sogenanntes Stringslicing = beliebige Aufteilung von Zeichenketten); andererseits ist man viel flexibler bei der Programmierung. Dies spielt insbesondere eine Rolle, wenn die Druckausgabe geändert werden soll. Ein Normalpapierdrucker wird erst bei mehr als 32 Zeichen pro Zeile "rentabel"...

Wenn man sich so ansieht, was an Textprogrammen offeriert wird, kann man nur staunen. Neben wirklich professionellen Produkten im Maschinencode (wie Tasword) wird auch in sich ernsthaft gebenden Fachzeitschriften allerlei Laienhaftes geboten. Dazu ein paar Beispiele:

- Da bietet eine bekannte Elektronikzeitschrift in ihrem ZX- Sonderheft ein "elektronisches Notizbuch" an. Die Textorganisation ist so aufgezogen, daß maximal fünfzehn Seiten a zwanzig 32-Zeichen-Zeilen upgedatet werden; Suchen nach einem bestimmten Begriff gibt es nicht; als sogenanntes Inhaltsverzeichnis erscheint "automatisch" die erste Zeile einer jeden Seite; Zugriff und Texteingabe sind nur über eine Zeilennummer möglich; es gibt keine Cursorführung, kein In-Text-Insert, keine Find- and Replace-Funktion...

- Die "Textverarbeitung" einer neueren Heimcomputer-Zeitschrift braucht eine ganze Programmseite (d.h. 22 Listingzeilen) für diverse GOTO's, und dann nochmal dasselbe für umfangreiche Cursorerläuterungen. Das simple SAVEN kostet - da viel unnütze Printaufforderungen - elf Zeilen; viele Module bestehen nur aus ein bis drei Programmzeilen (ohne RETURN); vier Zeichen pro Zeile gehen bei der Darstellung für die Anzeige von Zeile und Zeilenanfang verloren ... und ähnlich unnutzer Aufwand mehr.

- Die "Happy"-Textverarbeitung eines Computermagazins arbeitet ebenfalls mit umständlicher Zeilennummerierung (immer mit der Aufforderung "Zeilennummer eingeben!!). Naiv-fröhlich wird dem Benutzer / Hobbyprogrammierer angeraten, so vorzugehen: "Als erstes muß man den Drucker an den Computer anschließen, das Papier einziehen (!) und dann den ZX81 einschalten (!), um mit dem Textprogramm arbeiten zu können (!)".

Das dürfte als Kostprobe genügen. Wenden wir uns besseren Lösungen zu, die man auch finden kann (z.B. in dem flott geschriebenen Buch von I. Stewart und R. Jones, Machine Code and better BASIC, dessen ZEDTEXT uns immerhin eine Anregung zur Dokumentation wert war).

Ein Wort zur Dokumentation

Zu den folgenden Programmen des Komplexes Textverarbeitung noch ein paar kurze Worte, ehe wir näher darauf eingehen. - Zunächst haben wir sie so ausführlich wie möglich dokumentiert, um den Einstieg zu erleichtern. Notwendigerweise umfaßt die komplette Programmbeschreibung folgende Teile:

- Aufbau (Blockbild mit den Hauptprogrammfunktionen)
- Logik wichtiger Module/Routinen
- Variablenliste
- Allgemeiner Text zum Programm-Handling

In Zukunft werden wir nur noch eine Programmübersicht und die Logik dort aufzeichnen, wo es kompliziert wird. Wobei in diesem Zusammenhang die Frage der sogenannten Selbstdokumentation per Programm auftaucht. Immerhin gibt es erfahrene Leute, die behaupten, keinerlei Dokumentation der beschriebenen Art zu benötigen. Ihr Programm sei selber so "sprechend", daß jeder es verstehen könne. Was ist davon zu halten?

Selbstdokumentation bedeutet im wesentlichen zwei Dinge:

a) Gute Lesbarkeit und
b) Erläuterungen zum "Programmgeschehen"

Zur Lesbarkeit:
Natürlich kann die Optik mit formalen Mitteln aufgelockert werden, etwa wie folgt:

- abgesetzte Programmblöcke
- Überschrift zu den einzelnen Modulen
- sprechende Variablenamen

Erläuterungen im Programm selber können in REM's gesetzt werden. Bei komplizierten Abläufen ist dann jeder zweite Befehl mit einer REM-Erläuterung begleitet; ja es kann sein, daß mehr an erläuterndem Text, als an Befehlen im Listing steht. - Natürlich kann man dem abhelfen, indem die Logik in einem Flowchart aufgezeigt wird, welches in die 'Dokumentationsmappe' kommt. Das hat noch zwei weitere Vorzüge:

a) Man steigt nachträglich besser wieder ein (z.B. bei Erweiterungen, Anpassungen und versteckten, erst später sichtbaren bugs (= Programmfehlern)).

b) Es wird deutlicher, wie man Befehle vereinfachen kann, Befehlsfolgen zu Modulen/Unterroutinen zusammenfaßt und allerlei überflüssigen "Textschrott" ausmerzt.

Womit wir bereits die Nachteile einer ausführlichen Dokumentation im Programm selber angesprochen haben: Werden viel REM's verwandt und Programmblöcke mit Spaces aufgefüllt, um einen Absatz zu bilden, kostet das zunächst Speicherplatz. Zudem verlangsamt sich peu à peu die Durchlaufgeschwindigkeit. Das hat wiederum schlaue Leute auf den Trichter gebracht, eine sogenannte REMKILL-Funktion in Toolkits anzubieten. Sicherlich eine nützliche Sache, nur hat man im Endeffekt dann zwei Programmversionen - eine sich selbst dokumentierende und eine, mit der man arbeitet. Für sprechende Namen gilt ähnliches. Variablen-Namen sind ebenfalls je länger je speicher- und laufzeitaufwendiger. Mit anderen Worten - wenn schon viel Dokumentation im Programm selber steht, sollte man eine Hochglanzversion "zum Lesen" und eine Arbeitsversion für tatsächliche Läufe haben.

Nötig ist es in jedem Falle, zumindest den grundsätzlichen Aufbau eines Programmes festzuhalten, und sei es nur als Print-Ausdruck der Menus. Eine gute Benutzerführung macht sich gleichfalls bezahlt, worauf wir bei den folgenden Programmen zum Thema "Wort & Text" Wert gelegt haben.

2. Ein Schmierblock = Scratch pad

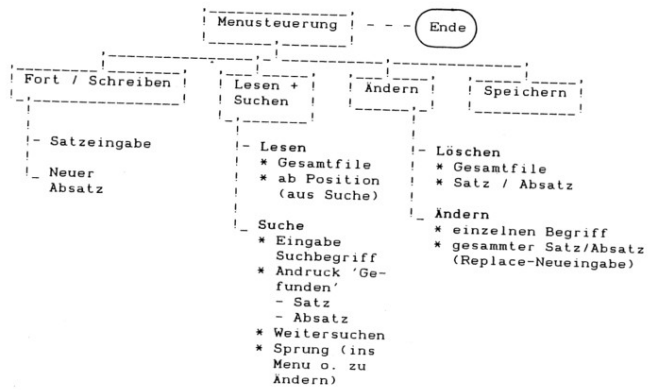
Ein einfaches Texthandling

Das Programm SCPD (Abkürzung für Scratch pad) ging von der Idee aus, zunächst ein einfaches Texthandling zu realisieren. Texte sollen rein- und rausgeschoben werden können, bestimmte Worte gesucht und angezeigt werden. Um der Anforderungen einer Ausgabe in optisch sprechender Form entgegenzukommen, wurde eine kleine Drucksteuerung eingebaut. Man kann jeweils Sätze oder Absätze ausdrucken, wozu bei der Eingabe von Texten ein eigenes Trennungszeichen "Absatz" oder "Satzende" eingefügt wird. So besteht denn die Schreibroutine darin, entweder einen einzelnen Satz einzugeben oder einen neuen Absatz (fort-) zu schreiben.

Wesentlich umfangreicher ist die Lese- und Suchroutine. Einmal kann man sich das gesamte File (= den Textstring) anschauen oder ab einer bestimmten Position Ausdruck bis zum Textende verlangen. Die Position wiederum bekommt man aus der Routine "Suche". Normalerweise wird man im File mit einem Suchbegriff arbeiten. Wurde er gefunden, kann jeweils der Satz oder der Absatz, wo er stand, ausgedruckt werden. Man kann weitersuchen in der Erwartung, daß der Suchbegriff nochmal vorkommt. Danach springt man zurück ins Hauptmenue oder direkt zum Ändern. (Der ursprünglich vorgesehene Weg über die Hauptmenue-Steuerung - siehe Schaubild "Programmaufbau SCPD" - wurde dahingehend abgeändert.)

In jedem Falle wird die Position aus "Suche" zwischengespeichert, so daß man auf ihr jederzeit aufsetzen kann.

Programmsteuerung S C P D



Noch ein Wort zur Suche.
Man sollte sich darauf gefaßt machen, daß beim ZX die Suchzeit beträchtlich ist. Um eine halbe Seite Text "durchzunudeln", braucht der Rechner mindestens fünfzehn Sekunden. Ein weiteres Handicap, das bei vielen Suchprozeduren auftaucht, ist zu beachten. Da gibt es den sogenannten IBM-Effekt. Gemeint ist folgendes.
Da auch Vergleiche mit Teilstrings erfolgen, findet der Rechner Gleichheit bereits bei einer Buchstabenfolge innerhalb eines Strings. Sucht man die Buchstabenfolge "IBM", stößt man unweigerlich auf Worte, die diese Buchstabenfolge enthalten, also beispielsweise "Schreibmaschine". Und noch eine Besonderheit ist zu beachten - wird ein Teilstück eines evtl. größeren Suchbegriffes gefunden, kann nur dieses Teilstück anschließend geändert werden. Hier hilft dann nur, entweder den ganzen Satz zu ändern oder den Suchbegriff auf das gefundene Wort zu erweitern.
Beim Ändern, wo der Weg zweckmäßigerweise als "verkürzter" Sprung von der Suche direkt getätigt wird, kann sowohl ein einzelner Begriff wie auch der gesamte Satz/Absatz geändert werden. Indirekt ist damit eine REPLACE-Funktion realisiert, indem bei gefundenem Suchbegriff der Satz gelöscht und anschließend neu eingegeben wird. Natürlich geht auch der Weg über das Hauptmenue beim Ändern: Nach New Line gelangt man ins Menue "Lesen/Suchen". Nun springt man in die Hauptsteuerung zurück und von dort zum Zweig "Ändern". Suchbegriff und Positionsspeicherung bleiben erhalten! Ein direkter Sprung nach "Ändern" führt zu einem "toten Weg", wenn bereits geändert wurde und der Suchbegriff quasi "verbrannt" ist. Man erkennt dies in der Anzeige dadurch, daß der Suchbegriff als Leerzeichen erscheint. Dann hilft nur ein "Notausstieg", indem man "NL" eingibt, um ins Hauptmenue zurückzugelangen.

Bleibt schließlich noch die Löschoption im Ast "Ändern". Sowohl das Gesamtfile, wie auch ein einzelner Satz oder Absatz kann gelöscht werden. Man sollte bei letzterem jedoch beachten (wie die Menueanzeige deutlich macht), daß die Option "Satz" bzw. "Absatz" während der Suche

für die folgende Prozedur "Ändern" gilt!

Features

Die Features von SCPD seien nochmal zusammengefaßt:

- keine Tabellenlösung, keine festen Textfelder, sondern das Ganze geht mit einem Textstring (A$) über die Bühne.

- Suche wie Änderung erfolgt wortweise, wobei auch ein Wortteil verwandt werden kann (aber Vorsicht: Es wird nur in Länge des Suchwortes abgeschnitten und ersetzt!) Beim Suchen wird wahlweise der gefundene Satz oder Absatz angelistet.

- Löschen und Lesen erfolgt ebenfalls gemäß der gewählten Option satz- oder absatzweise.

- Die Bedienungsführung erscheint ziemlich narrensicher, wobei insbesondere der direkte Sprung "Lesen/Suchen" zu "Ändern" einen gewissen Komfort darstellt.

- Sollte man einmal abstürzen, kann man jederzeit mit "GOTO 16" ins Menue zurückgelangen, ohne daß der Textstring gelöscht wird.

3. Text mit TXT

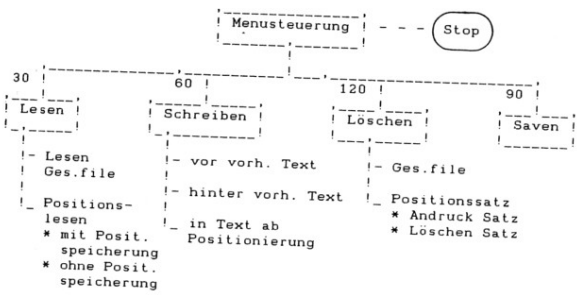
Eine fortgeschrittenere Lösung

In mancher Hinsicht ist das Programm TXT "fortgeschrittener", wenngleich hier keine Suchfunktion realisiert wurde. Wir wollten unterschiedliche Schwerpunkte bei unserem "Programmangebot" herausarbeiten, um so dem Hobbyprogrammierer Anregung und Hilfe für eigene Bemühungen zu geben. Da das TXT-Programm großere Programm- und insbesondere Sinclair-Kenntnissen erfordert, haben wir es ausführlicher dokumentiert. Neben der Modulübersicht (mit Angabe der jeweiligen Programm-Startzeile) findet sich eine Variablenliste, die Auskunft über verwendete Programmgroßen gibt. - Während SCPD als Leckerbissen eine Hauptmenue-Steuerung mit laufendem Cursor brachte, liegt die Raffinesse von TXT in der Leseroutine. Man hat die Wahl, den bestehenden Text langsam oder schnell zu lesen (unabhängig davon, ob man den SLOW- oder FAST-Modus einschaltet). Und das nicht ohne Grund:
Das Langsamlesen dient dazu, durch Tastendruck einen Print-Stop herbeizuführen und gleichzeitig eine Positionsmarke zu setzen. Nachdem das File so "positionsgelesen" wurde, kann mit der gespeicherten Position weitergearbeitet werden.

Mit ihr operiert TXT sowohl bei Schreiben als auch bei Löschen. Im Programm sind diese Hauptfunktionen relativ einfach gehalten. Man kann etwas sowohl vor den Text wie hinter ihn per Eingabe bringen, man kann aber auch ab der festgehaltenen Position in den Text hineinschreiben. Beim Löschen wiederum dient die Positionierung dazu, ab der betreffenden Stelle den Satz angedruckt und gelöscht zu bekommen. Weitere Einzelheiten möge man der folgenden Modulübersicht und der Variablenliste entnehmen.

Programmübersicht T X T

(Ziffern = Programmstartzeile)



Wichtige Unterprogramme

190: Pause (f. Langsamlesen)
200: Andruck positionierter Satz
220: Positionieren für Satzlöschung

Wichtige Variablen für Programm TXT

A$	Textstring
POS	Position in e. Textsatz (d.h. zwischen den Punkten) für Lesen, Schreiben u. Löschen
B$	Eingabestring bei Schreiben
I$	Menueingaben
A, E	Satzanfangs u. -endeposition für die Löschung
OK	Schalter z. richtigen Positionieren (in UPRO 220)
N, M	FOR - NEXT - Laufvariablen

Erläuterungen

Um das Programm etwas genauer zu erläutern, sei auf die Haupt-Features weiter eingegangen:

- Der Aufbau von TXT wurde einfach und kompakt gehalten (d.h. wenig Laufzeitoptimierung!), um eine simple Stringbehandlung zu ermoglichen.

- Löschungen erfolgen satzweise, wobei nur auf eine Stelle im Satz (d.h. zwischen zwei Punkten) postioniert werden muß.

- Umfangreiche Menueführung und zahlreiche Anzeigefunktionen sollen das Handling erleichtern.

- Die Änderung von Text kann nur über Löschen und anschließende

(Insert-) Neueingabe erfolgen.

- Im übrigen wurden ein paar Sinclair-spezifische "Logik"-Tricks verwandt, die dem Eingeweihten nicht ganz unvertraut sein durften.

Natürlich kommt TXT weiterreichenderen Wünschen zur Textverarbeitung nicht allzu entgegen: Es gibt keine buchstaben- oder wortweise Korrekturmöglichkeit, keine Search-Funktion und erst recht nicht so was wie Block-Delete. Damit würde man sich bereits in den professionellen Bereich vorwagen.

Immerhin wäre eine komfortable Cursorsteuerung wünschenswert... aber sehen wir weiter.

4. Wortverarbeitung mit WORD

"Wortfunktionen"

Mit dem Programm WORD gehen wir nun voll ins Eingemachte. Es handelt sich zwar noch nicht um einen ausgewachsenen Wortprozessor, aber die beiden wichtigsten Funktionen, die eine wortweise Verarbeitung (von Texten) haben muß, sind immerhin realisiert:

a) komfortable Cursorsteuerung, die eine exakte Positionierung und ein Lesen über mehrere Seiten ermöglicht

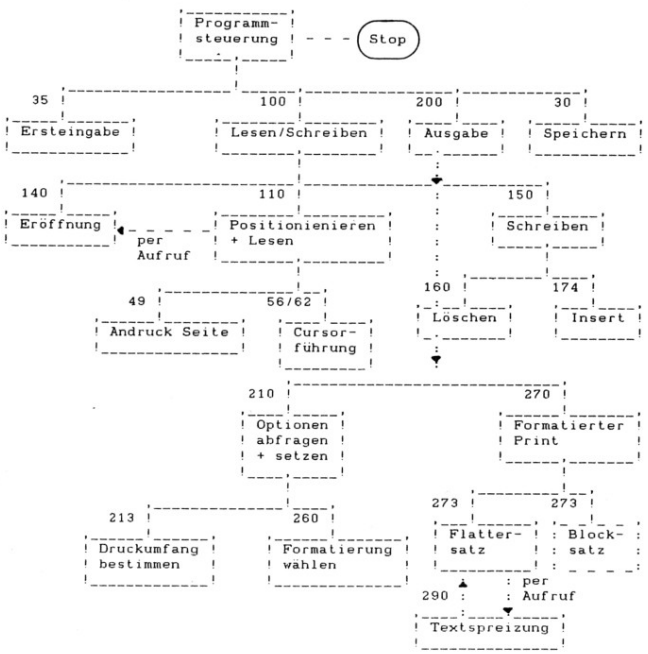
b) umfangreiche Formatierungsoptionen, wobei man bei frei wählbarer Druckbreite im Flattersatz (d.h. linksbündig justiert) oder im Blocksatz (d.h. Spreizung des Textes bis zum rechten Rand) printen kann.

WORD enthält ziemlich alles, was man zur Textverarbeitung (bis auf eine fehlende Find-/Replace-Funktion) braucht. Freilich kann man keine Drucksteuerung eines Normalpapierdruckers (= Einfügen spezieller Sonderzeichen) und Komfort wie Block-Delete und Copy erwarten. Aber immerhin bietet WORD neben CURSOR-gesteuertem Lesen und Positionieren (für Inserten und Deleten) die Wahl unterschiedlicher Ausgabeformate. Beim Andruck über (Normalpapier-) Drucker braucht man nur die formatierte Ausgabe in den folgenden Zeilen von PRINT in LPRINT zu ändern:

- für Flattersatz: Zeile 284
- für Blocksatz: Zeile 300

Eine Grundregel vorweg, ehe wir auf einzelne Programmeigenschaften und -teile eingehen: *Eingefügt* (Insert) wird grundsätzlich *nach* der letzten Cursorposition, *gelöscht* wird grundsätzlich *vor* der letzten Cursorposition. Davon gibt es nur eine Ausnahme-aus berechtigtem Grund: Wird das erste und letzte Zeichen im File angesteuert, kommt der eingegebene Text vor bzw. hinter dem bisherigen Gesamttext zu liegen. Das ist notwendig, um Textfortschreibung über bloßes Einfügen hinaus in diesem Rahmen zu ermöglichen.
Die folgende Modulübersicht zeigt mit exakter Angabe der Programmzeile jeweils, wie das Programm aufgebaut ist und wie es arbeitet.

Modulübersicht von W O R D



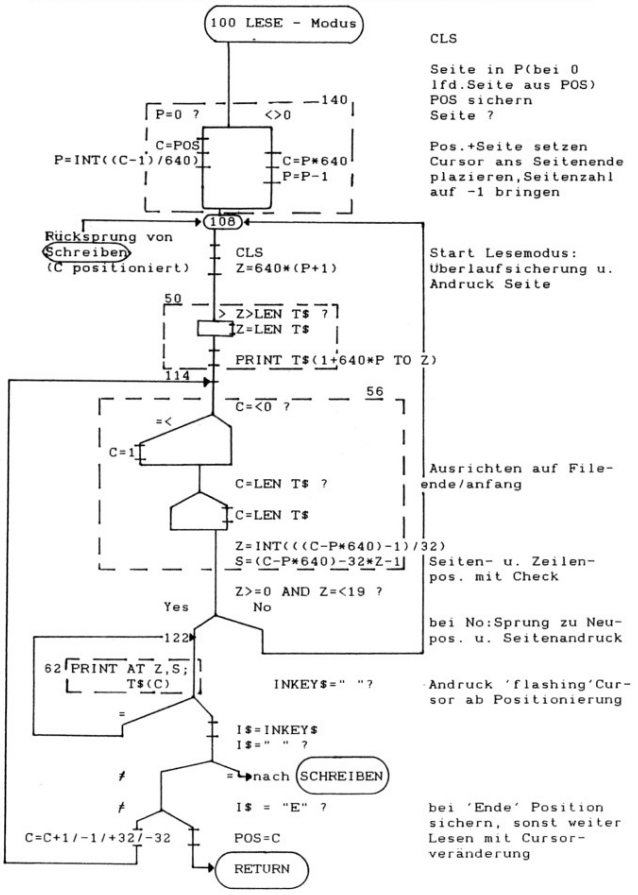
Besondere Leckerbissen sind einmal Benutzerführung mit *flashing Cursor* (d.h. blinkendem Anzeiger) und die Realisierung von *Blocksatz*. Letzterer wird indirekt, d.h. nicht in einer eigenen Routine erzeugt. Man springt vielmehr vom Flattersatz per Aufruf in ein Unterprogramm, dessen Aufgabe darin besteht, den Text zu spreizen. Der Flattersatz bewegt sich bereits innerhalb der vorgegebenen Druckbreite, ohne daß einzelne Worte auseinandergerissen werden. Da lag es nahe, die bestehende "Formatierung" mit Spaces aufzufüllen, bis ein Wortende (oder Satzendezeichen) den rechten Rand erreicht hat. Übrigens ist die Logik der Formatierung in einem eigenen Schaubild festgehalten, wobei insbesondere das Zusammenspiel zwischen linksbündiger Ausgabe und Aufruf der Spreizungs-Unterroutine deutlich wird. (Siehe unten "Logik der Formatierung" und "Logik der Spreizung")

Neben der umfangreichen "Ausgabe" ist das Modul *"Lesen / Schreiben"* ebenfalls recht stattlich. Es zerfällt in die zwei Hauptbereiche "Lesen mit Positionieren" und "Schreiben". Der sogenannte Lesemodus arbeitet mit einzelnen Programmteilen, die jeweils als Modul im Ablaufbild gekennzeichnet und ausgelagert sind.

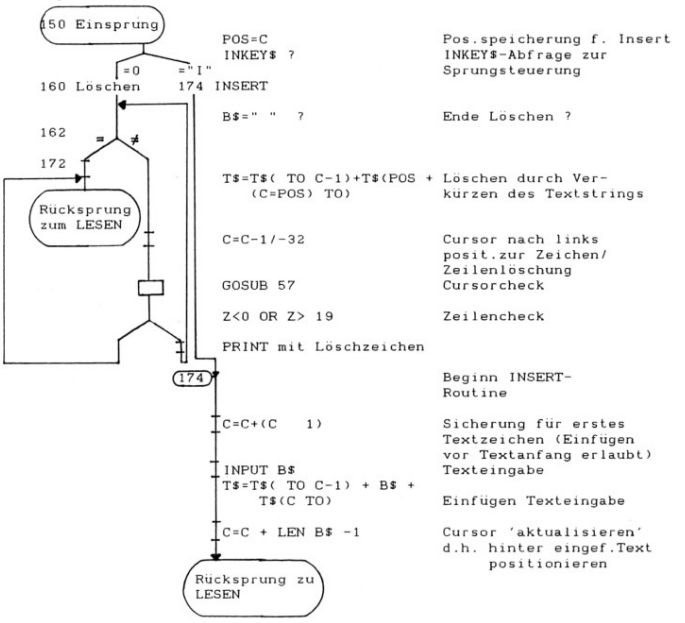
Wichtigster Kernpunkt beim Lesen ist die komfortable Cursorführung. Zahlreiche Überlaufsicherungen sorgen dafür, daß man nie außerhalb des Textstrings T$ gerät. Neben dem Positionieren mittels Cursor kann man jedoch auch eine vorgegebene Seite für die weitere Bearbeitung (Lesen-/Schreiben) vorgeben. Der Grundgedanke bei der ganzen Positioniererei war simpel: Um die Cursor- und damit Zeilen-/Spalten-Steuerung zu ermöglichen, wurde einfach das Textfile in 640er Schritten (bzw. 32 Stellen bei der Zeilenbestimmung) aufgeteilt. Das ist nämlich die Anzahl der pro Druckseite verwandten Zeichen. Eine Seite umfaßt 640 Zeichen, weil zwanzig Zeilen a 32 Zeichen pro Zeile auf dem Bildschirm bequem unterzubringen sind. Die restlichen beiden Zeilen bleiben für Programmausgaben an den Benutzer reserviert.

Programmlogik vom Modul "LESEN/SCHREIBEN"

(Ziffern=Programmzeile,Strichelung=Auslagerung als Untermodul)



Logik des SCHREIB-Modus



Der Schreibmodus, dessen (einfachere) Logik oben ebenfalls aufgezeigt ist, ermöglicht Löschen und Inserten ab der laufenden Position. Man kann nicht nur Zeichen unmittelbar vor dem Cursor - wie durch ein schwarzes Viereck veranschaulicht - löschen, man kann sogar eine ganze Zeile auf einen Schlag "wegbringen". In diesem Falle werden die Steuerungstasten dazu benutzt, einen 31-stelligen Sprung rückwärts (mit dem Cursor) zu machen. Aus gutem Grund ist die Cursorsteuerung beim Löschen immer "rückwärts" gerichtet: man soll nur das löschen, was man zuvor beim Lesen (mit positioniertem Anzeiger) gesehen hat. Zum Einfügen wird der einzugebende Text abgefragt, damit er anschließend ab der bisherigen Position eingefügt werden kann. (Wie im einzelnen die Variablen eingesetzt und verwandt werden, geht aus der Übersicht weiter unten hervor). Allerdings sollte man sich über eines keine Illusion machen - je komfortabler und abgesicherter die Wortbehandlung ist, um so aufwendiger und zeitraubender gestaltet sich der Programmablauf.

Das Wichtigste von WORD und seine Logik

Fassen wir noch einmal das Wichtigste zu WORD zusammen:

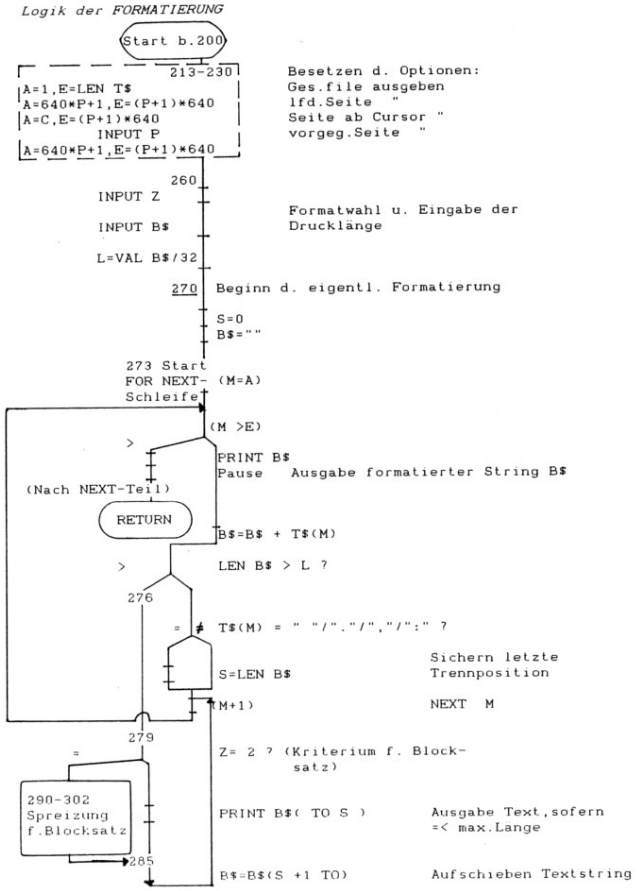
- Das Textfile ist in Seiten (engl. Page, daher die Variable P) aufgeteilt. Wählt man eine bestimmte Seite vor - man kann auch auf die laufende Seite zurückgreifen, falls man die Seitennummer vergessen hat - , muß diese innerhalb des gegebenen Textvolumens liegen. Der Cursor erscheint in diesem Falle blinkend am Ende der Seite.

- Gelöscht wird vor der letzten Cursorfunktion, eingefügt hinter derselben. Ausnahme: erste und letzte Textstelle!

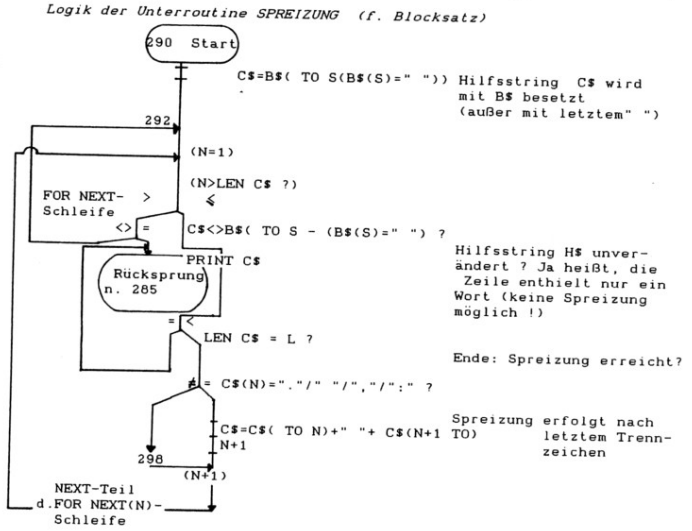
- Beim Lesen kann über eine Seite hinaus (vor- oder rückwärts) positioniert werden. Die aktuelle Seite wird jeweils angezeigt.

- Die Formatierung wowohl als Flatter- wie auch als Blocksatz ist so "sicher", daß es zu keinen Verklemmungen kommt. Wenngleich die Routine nicht gerade die schnellste ist, verkraftet sie auch überlange Worte, z.B. wenn in einer Zeile kein zweites Wort mit Leerzeichen mehr aufgefüllt werden kann. (Näheres ist der folgenden "Logik" zu entnehmen)

Logik der FORMATIERUNG



Logik der Unterroutine SPREIZUNG (f. Blocksatz)



Übersicht der in WORD verwandten Variablen

Name	Funktion	Anwendung (location)
String-Variablen		
A$	Menuevariable zur Programmsteuerung	Menu (20)
B$	Variable zur Aufnahme Texten (als Eingabe o. zur Ausgabeformatierung) daneben zur Optioneneingabe vom Benutzer (wie I$)	Insert b.Schreiben(175) Flattersatz/Print b.Format.(270) Input b.163,266
C$	Textaufnahmestring f. Blocksatz (d.h. nach Spreizung von B$)	'Spreizung'(290)
I$	Allg. Abfragestring z.Benutzerführung (vor allem bei d. Cursorsteuerung)	Lesen/Schreiben(124,153) Ausgabe (213)
T$	Textstring als 'Gesamtfile'	
Numerische Variablen		
A	Anfangsposition f. formatierten Print (Gegenstück zu E)	zur Druck/Seitenbegrenzung (224)
C	Cursor,der im Zusammenspiel mit T$ die zentrale Positionierungsrolle besitzt!	
E	Endeposition f. formatierten Print	(wie oben)
POS	dient der hilfsweisen Speicherung einer Textposition	Initialisierung bei 18,38,140
P	Seite (=page) innerhalb d. Textfiles	Lese/Schreiberöffnung (140)
S	Spalte) beim Lese/Schreib-	Seitenprint(50)
Z	Zeile) Modul daneben auch als Hilfsvariablen beim Formatieren (s. 263,271)	Cursorcheck(57) Cursor-(59,63,168) u. Seitenprint(50)
FOR NEXT-Laufvariablen		
M	verschiedene Hilfsfunktionen	Pause (40), Formatierung(273)
N	allgemeine Funktion	Spreizung (292)

Kapitel IV:

Finanz & Management

1. Programm VERM

Der ZX als "Rechner"

Mit den folgenden Programmen ziehen wir einen weiteren Anwendungskreis in unsere Nutzungsüberlegungen ein. Die folgenden Beispiele - wiederum als Vorschlag und Anregung gedacht - sollen zeigen, wie man seinen ZX zum rechnen kriegt. Das Gerät muß dabei kräftig "schuften". Es sind weniger Daten zu verwalten, als vielmehr allerlei Rechenoperationen zu vollführen. Auch bei Erledigung umfangreicher Zahlenwerke ist der Sinclair zu nützlichem Tun fähig. Allerdings wird schnell klar, daß das Arbeiten mit (Fließkomma-) Zahlen Speicherplatz kostet. So kann ein Programm mittleren Kalibers schnell auf 8 K RAM und mehr kommen, wenn es stattliche Zahlenarrays benutzt. Der Luxus einer 9-Stellen-Genauigkeit hat seinen Preis. Da meist die volle Zahlenvarianz gar nicht gebraucht wird, könnte man auf die Idee verfallen, mit "String-zahlen" zu operieren. Da kostet zwar weniger Speicherplätze, erfordert jedoch umfangreiche Konvertierungen (intern), die in die Laufzeit gehen. Bei der extrem platzsparenden CODE-Verwendung ist man zum anderen an den Zahlenbereich (Ganzzahlen positiv) von 0 - 225 laut Sinclair-Codetabelle gebunden...

Wie man es dreht und wendet, man stößt auf irgendwelche Grenzen. Daher wurde zunächst auf die "übliche" Zahlendarstellung zurückgegriffen, ehe im folgenden Kapitel Tricks zu Zahlen und Positionierungen verraten werden. So machen denn in den Programmen dieses Kapitels IV Zahlenarrays den Hauptteil aus. Letztere haben einen Vorteil - sie sind leicht zu adressieren und upzudaten. Um die Übersichtlichkeit zu erhöhen, wurden verschiedenen Zahlentabellen ihrem Zweck entsprechend gebildet. Man hätte auch mehrere Tabellen zusammenbinden können, allerdings ist die Adressierung schwer nachzuvollziehen. Ferner muß man sich dann mit dem Problem unterschiedlicher Dimensionen herumschlagen. Möchte man beispielsweise (wie in Vermögen) in einer Zahlentabelle "Kurs, Bestand, Zu-/Abgang und Wert" unterbringen, benötigt aber nur "Kurs, Bestand und Wert", bleibt eine Dimension ungenutzt. Also ist es angebracht, zwei verschiedene Arrays zu schaffen und sie in getrennten Schleifen abzuarbeiten. Dies und vieles mehr wird in den folgenden Listings gezeigt. Dabei wurde Wert auf ausgiebige Dokumentation gelegt. Was man in manchen Zeitschriften (und Programmbüchern) vorgesetzt bekommt, kann bisweilen nur als Zumutung bezeichnet werden. Es ist dem Verständnis nicht dienlich, nur ein paar Hinweise zu bringen, was das Programm alles kann und wie toll es ist. Manchmal fehlen gar Variablenliste und Programmübersicht. Was soll der Hinweis, " Das Programm erklärt sich von selbst" dann?

Die Vorlagen zum Themenkreis "Finanz und Management" stammen übrigens von Herrn Rybarczyk. Er lieferte Idee und Programmgerüst, die Programme selber wurden vom Herausgeber gründlich überarbeitet. Es gelang nicht nur, eine Reihe von "Ungradheiten" auszubügeln, vielmehr wurden auch Zusatzfunktionen eingebaut. Insgesamt gelang es bei VERM, Programmumfang und Speicherbedarf um etwa ein Viertel zu straffen. Auf weitere Besonderheiten wird im folgenden näher einzugehen sein.

Status contra "Buchungen"

Das Programm VERM (Abkürzung für VERMöGEN und Ladename) geht von der Idee einer einfachen Bestandsführung aus. Alle Vermögensarten werden

monatlich upgedatet. Diese Monatseingaben werden gespeichert und stehen auf Abruf zur Ausgabe bereit. Es wird der jeweilige Status ausgeworfen, umfangreiche Buchungen entfallen. Da es unser Ziel ist, den ZXer von einfachen zu komplexeren Anwendungen zu führen, war ein solches Vorgehen sinnvoll.

Selbstverständlich hätte man auch "Buchungen" einbauen können. Bei "Vollausbau" wäre dann mit Gegenkonten zu arbeiten, was den Überblick erschwert hätte. - In einfachster Form bedeutet Buchführung die Einbeziehung von Zu- und Abgängen. Das ist in VERM teilweise (bei Anlageart Gold) geschehen. Um jedoch Unterpositionen einer Anlageart zu führen, wurde auf eine exakte Buchführung im übrigen verzichtet. Sonst hätte man für alle Positionen eigene Monatstabellen anlegen müssen. Hier ist zu einem Kunstgriff Zuflucht genommen worden: Es gibt zwei Arten von Anlagekategorien - 1. Positionswerte und 2. Globalwerte. Positionswerte sind Arten, die bis zu 10 Einzelwerte aufnehmen können. In VERM sind das Aktien, Renten, Spar- und sonstige Anlagen. Nun der Kunstgriff:

Statt die Positionen Monat für Monat in eigenen Arrays zu speichern wird nur eine, die letzte, Eintragung zwischengespeichert. Man hat dergestalt immer die Positionen im Griff, kann sie löschen oder mit neuen Namen versehen, die aktuelle Kursnotierung und den Bestand eingeben etc. Andererseits wandert der Monatsgesamtwert je Anlageart sowieso in die Monatstabelle, sodaß nur *ein* Zwischenspeicher für aktuelle Dinge benötigt wird. Da dort nur zuletzt eingegebene Positionswerte stehen, sei man gewahr, daß durch Zugriff auf einen anderen als den letzten Monat die Optik falsch erscheinen kann. Hat man beispielsweise zuletzt unter "Aktien" VW im Juni upgedatet und fragt den Mai ab, erscheinen Kurs, Menge und Wert von Monat Juni! Die Summen stimmen natürlich, da nach Anlageart addiert wird.

Bei Globalwerten gibt es keine Unterpositionen. Die Anlagearten Gold und Bargeld kennen keine Unterteilung. Bei Gold findet lediglich eine Unterscheidung zwischen Monats- und Gesamtbestand statt. Das heißt, die Monatsmengen werden als Gesamtmenge jeweils kumuliert. Da man den betreffenden Monatskurs eingibt, kann man die Wertentwicklung gut verfolgen. Bei anderen Anlagearten erfolgt keine "Fortschreibung", was auch beim "Luxus" von 10 Einzelpositionen nicht unbedingt notwendig erscheint. Zu- und Abgänge werden halt "netto", d.h. mit dem neuen Bestand eingegeben. Und wie steht es mit Positionslöschungen bzw. -änderungen ?

Kein Problem - gelöscht wird, indem als "neuer Name" ein Leerstring eingegeben wird und die Menge bzw. der Kurs auf Null gesetzt wird. Bei Änderungen "quittiert" man den Namen (als gültig) mit Space, um anschließend neue Menge und neuen Kurs einzugeben. Aber Achtung: Unterläßt man bei einer Position das monatliche Updating (d.h. man übergeht die Aufforderung "Menge/Kurs" mit NL), erscheint sie in der Monatsberechnung als Nullwert! Im Monatsstatus kann nur stehen, was richtig und vollständig eingegeben wurde.

Noch ein Hinweis, ehe wir genauer auf Bedienung und Programmeinzelheiten eingehen: Eigentlich selbstverständlich dürfte sein, daß "Vermögen" immer positive Gesamtwerte bedeutet. Theoretiker mögen darüber streiten, ob es einen negativen Bargeldbestand geben kann... Auch hüte man sich vor anderem "Unsinn", wie das Diagramm der Vermögensentwicklung bei nur einer Monatseintragung abzufragen!

Womit wir bei den "Features" des Programms sind, auf die es sich lohnt, kurz einzugehen. Das Programm ist etwa 8,4 K RAM "schwer" und

arbeitet mit drei Menues: Monatseingabe, Aktueller Status, Vermögensdiagramm. An Vermögensarten können "monatsscharf" upgedatet und abgerufen werden: Aktien / Renten (=festverzinsliche Wertpapiere mit Börsenkurs), Spar- / Sonstige Anlagen (mit Konto/Bezeichnung und Fälligkeit), Gold und Bargeld. Für alle außer Gold und Bargeld sind bis zu 10 Positionen angebbar, die als aktuelle Information mit der letzten Eintragung zwischengespeichert werden. Bei Statusabfrage werden sie zusammen mit den Monatswerten (je Anlageart) ausgeworfen. Als Unterscheidung ist vom Benutzer wählbar: Bei der Gesamtübersicht nur die Monatssummen ohne Positionsanlistung; werden Detailinformationen gewünscht, kommen die Positionseintragungen auf den Schirm. Bei letzterem kann nach Anlageart selektiert werden - entweder man bekommt die betreffende Art präsentiert oder alle Arten hintereinander. Durch Eingabe von NL kann man dann - wie auch sonst - weiterblättern.

Benutzerfreundlich ist ebenfalls die Anzeige der Monatsentwicklung (global des Vermögens). Das Diagramm skaliert maßstabgerecht Ober- und Untergrenze der Summen. Wird ein neuer Höchstwert erreicht, schiebt sich die Darstellungsgrenze nach oben, auch die Unterteilung in Zehner-, Hunderter- oder Tausenderschritte geschieht "automatisch". Die Skalierungsroutine ist ziemlich ausgefuchst, sie arbeitet mit bisherigen Min(imal-) und Max(imal-)werten, benutzt den 10er Logarithmus um die Stepkoordinate zu bestimmen und gewinnt so den passenden Darstellungsmaßstab (Variable F) fürs eigentliche Plotten. Man kann sie in anderen Programmen verwenden, wenn es ums Zeichnen einer Zahlentabelle geht. Neben zahlreichen benutzergerechten Pausefunktionen (mit zwei verschiedenen Lösungen) kommt VERM dem User mit einer Reihe "special effects" entgegen. Sie seien kurz angetippt:

- Monatskürzel wie Menuetext wird aus Texttabelle bzw. Zeichenstring geladen und immer gleichlautend präsentiert

- beim Monatsupdate gibt es nachträgliche Korrekturmöglichkeit (bei Spar/Sonstige Anlagen), zu überlesendes wird durch Tastendruck (NL) übergangen

- der "aktuelle Status" arbeitet mit raffinierter Printselektion; sowohl Anlageart wie Positionswerte werden direkt angesteuert

- Das Diagramm skaliert mit ansprechender Maßstabsunterteilung, wobei bei Überschreiten der Tausendergrenze Ganzzahl plus "TS" erscheint

- schließlich - beim Saven wird invers blinkend der Benutzer (ähnlich wie es der Spectrum macht) auf Inbetriebnahme seines Recorders aufmerksam gemacht

Auf folgende Besonderheit sei noch hingewiesen, ehe wir ausführlich das Programm in Einzelheiten dokumentieren: Bezeichnungen bei Aktien / Renten sind maximal 10 Stellen lang; bei Spar / Sonstigen Anlagen sieht die Feldaufteilung so aus - Konto = 1. - 8. Stelle, Fälligkeit Stelle 9 - 12 (Eingabe in einem Zug hintereinander).

Programmdokumentation

Auf den folgenden Seiten findet der Leser eine umfangreiche Dokumentation des Programms. Sie enthält alles, was man für einen tieferen "Einstieg" wissen muß (und oft vermissen muß). Zuvor sei jedoch aufgeführt, wie eine Positionseingabe erfolgt. Das ist erforderlich, falls es aus dem Blatt "Ablauflogik" nicht hervorgehen mag. In Zusammenhang mit dem Anwendungsbeispiel (s. weiter unten) wird klar, wie die Prozedur vonstatten geht. Also - jemand ruft Monat Januar auf und will

unter Aktien VW eingeben. Wie geht das?

Die Schritte im einzelnen:

- Handelt es sich um Ersteintragung, wird "*VW" eingetippt, dann quittiert man letzteres mit "_"; bei vorhandener gültiger Benamung geht man gleich mit "_" weiter

- nun werden Anzahl und Kurs eingegeben; möchte man die Positionen übergehen (man kann ohne weiteres in einem Monat mehrmals neue, aktuellere Daten eingeben!), springt man gleich per NL zur nächsten Position

- nicht besetzte (Leer-) Positionen werden desgleichen mit NL "überblättert", bis Position 10 erreicht ist

- danach erscheint die Summe der Positionswerte; letztere errechnet das Programm selber, d.h. man braucht nur Anzahl und Kurs einzutippen

- mit Tastendruck geht es zurück ins Menue, um weitere Anlagearten zu aktualisieren

- mit Option 7 (Ende Eingaben) erfolgt Monatssummierung, die man sich als "Übersicht Gesamt" (unter Menue STATUS) anschauen kann

Abschließend noch ein paar Memos:

1. Überschreiben von Monatswerten ist jederzeit möglich

2. Monatsaktualisierung erfordert mindestens Fortschreiben des Kurses, sofern sich ansonsten nichts geändert hat (Achtung - andernfalls bleiben Nullwerte "stehen"!).

3. Bei "Anwahl" von "Akt.Status" mit einem anderen als dem letzten Monat erscheinen als Monatssummen die des vorgegebenen Monats, aber die "aktuellen" Einzelpositionen des zuletzt eingegebenen.

4. Dezimalwerte müssen mit Punkt (statt Komma) vorgegeben werden.

Die folgende Dokumentation besteht aus den Blättern:

- *Programmaufbau*

- Grobschema *Ablauflogik* - sie gibt die beiden Hauptzweige "Monatseingaben" und "Aktueller Status" wieder

- weitere Programmeinzelheiten enthält die *Modul/Routinenbeschreibung*; wie bei der "Logik" sind die betreffenden Startzeilen (im Listing) als Nummer mitgeteilt

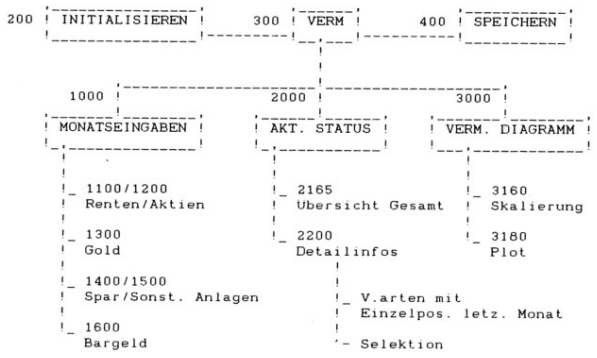
- die *Variablenliste* zeigt in Unterteilung, welche Definitionen den Programmroutinen zugrunde liegen; Feldinhalt und - aufteilung erlautern, wie im einzelnen gerechnet und zugeordnet wird

- abschließend veranschaulicht ein *Anwendungsbeispiel*, was alles vom Benutzer vorzugeben ist und was der Rechner dann errechnet

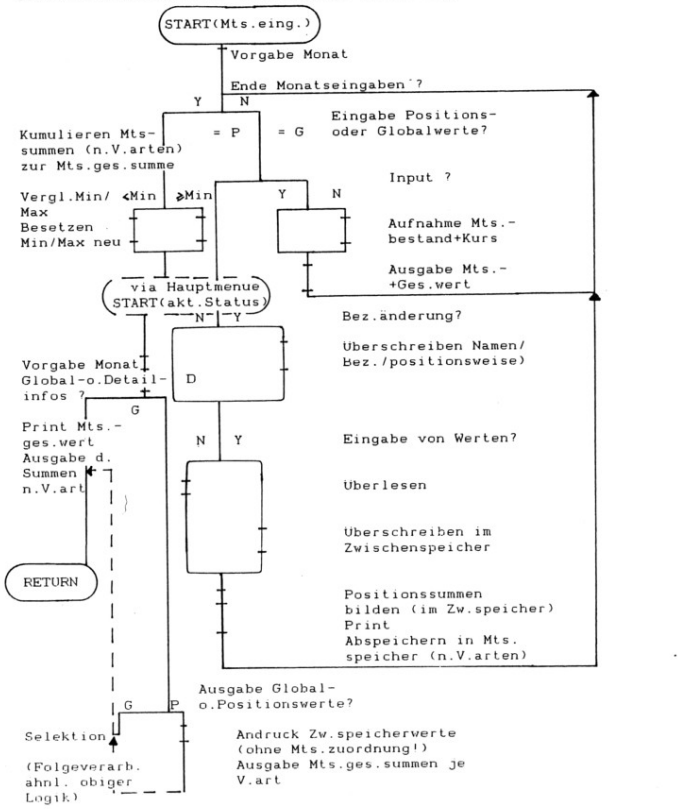
Wie man sehen kann, ist Programmdokumentation nicht mit Bemerkungen wie "Das Programm dokumentiert sich selbst" abzutun. Je umfangreicher und komplexer ein Listing ist, umso mehr Hilfestellung braucht Be-

nutzer wie Nachvollzieher. Selbstverständlich gibt es in der Ausführlichkeit Grenzen. Wollte man die gesamte Logik umfangreicher Programme wiedergeben, brauchte man viel Papier; aber das Gerippe sollte schon deutlich werden. Bei einfachen, überschaubaren Lösungen kann der eine oder andere Punkt wegfallen. Immer gehört zur Dokumentation - so kann man mit Fug und Recht verlangen - Variablenliste und Aufbauschema. Womit wir hoffen, ein gutes Beispiel bei allen Programmen dieses Buches gegeben zu haben.

Programmaufbau von VERM



Grobschema ABLAUFLOGIK (Menuezweige 'Monatseingaben'+'Akt.Status')



MODUL/ROUTINEN-BESCHREIBUNG

Menue (Unter-menue)	Progr.-Zeile	Bezeichnung	Beschreibung	Anmerkung
0	200	*Initialisieren*	Laden Texttabelle Anlagearten Besetzen Monatsabkürzungen Dimensionieren von a)Textarrays b)Wertearrays Initialisieren Min/Max f. Grafik	
1	1000	*Monatseingaben*	Vorgabe Monat Wahl Anlageart + Updating Errechnen mtl.Gesamtwerte aller V.-Arten (nach Eingabe-Ende)	Unterprogramm Upro 130
(1) +(2)	1100 +1200	Mts.eingaben Renten + Aktien	Ggf.Namensänderung,Eingabe Mts.bestand u.Kurs je Position Ausgabe Positionssumme Abspeichern ders.als Monats-Summe V.-Art	Upro 110
(3)	1300	Mts.eingabe Gold	Eingabe Kurs,Monatsbestand Errechnen Ges.bestand(=kumulierte Mts.bestände bis dahin) Ausgabe u. Abspeichern aktualisierten Gesamtwert d.Mts.	Upro 100
(4) +(5)	1400 +1500	Mts.eingabe Spar/Sonst.A.	Ggf.Änderung v. Bezeichnung und Fälligkeit Eingabe Mts.wert je Position Errechnen mtl. Gesamtwert Ausgabe dess. u.Abspeichern	mit Korrekt-möglichkeit Upro 120
(6)	1600	Mts.eingabe Bargeld	Eingabe Mts.bestand Anzeige u. Abspeichern dess.	
2	2000	*Akt.Status* *(letzter Monat)*	Eingabe Monat Wahl der Wiedergabeart Sprung z.Untermenue	(nur dieser wird angez. danach Rücksprung ins Hauptmenue

(1)	2165	Übersicht Gesamt	Andruck aller 6 Vermögensarten mit den Monatssummen Ausgabe aktuelle Gesamtsumme	
(2)	2200	Detailinfos aktueller Status	Wahl der Anlageart Unterprogrammaufrufe :	auch alle Arten ohne
	2300		Renten/Aktien ltzt.Monat +Ausgabe aller besetzten Positionen(Anz.,Kurs,Wert) +Andruck d. Monatsgesamts	Selektion Konsistenz nur wenn Monatswahl= letzter Mon.
	2500		Gold +Ausgabe akt.Kurs,Monats-u.Gesamtbestand +Anzeige Gesamtwert	
	2600		Spar/Sonstige Anlagen +Anlisten aller besetzten Einzelpositionen(Anz.,Fälligkeit,Wert) +Andruck d. Monatsgesamts	Konsistenz s.o.
	2700		Bargeld Ausgabe d.Monatsbestandes	
3	3000 3135 +3160 3180	*Diagramm Vermögensentwicklung*	Zeichnen u.Skalieren d.Koordinaten(n.bish.Min/Max) Plot d. besetzten Monats-gesamtwerte(=Summe V.arten)	mit Festlegen von Darstellg. Maßstab
4	4000	*Speichern*	Startnachricht, dann Abspeichern unter Prgr.namen VERM	als Invers blinken

VARIABLENLISTE

1.Strings: M$=Monatskurztexte
E$=Eingabestring

2.Numerische Variablen:
A,B =Menuewahl-Variablen
M =Monat
N =Laufvariable (dito auch Y)
Z =Zwischenwert für Monatskumulation
F,Y,X=Skalierungsvariablen für Diagramm
MAX =Maximum der Monatswerte) f.Skalierung
MIN =Minimum der Monatswerte)

3.Arrays

a)Zeichenarrays:
N$ =Texttabelle f. V.-Arten
A$ =2dimensionale Positionsbezeichnungstabelle (Renten/Aktien) *
S$ =desgleichen für Spar/Sonstige Anlagen *

b)Zahlenarrays:

```
        G(..)=Werte für Gold  *
        A(..)=Positionswerte f.Renten/Aktien  *
        W(..)=Positionswerte f.Spar/Sonstige Anlagen  *

        T(M) =Monatswert Renten
        O(M) =     "     Aktien
        X(M) =     "     Spareinlagen
        V(M) =     "     Sonstige Anlagen
        Y(M) =     "     Bargeld
        Z(M) =     "     Summe alle V-Arten

*  Erläuterungen zu den Arrays:
- - - - - - - - - - - - - - - - -

DIM N$(7,13)     = Anlagearten

          1='RENTEN         '
          2='AKTIEN         '
          3='GOLD           '
          4='SPAREINLAGEN   '
          5='SONST.ANLAGEN  '
          6='BARGELD        '
         (7='ENDE EINGABEN ')
DIM A$(2,10,10) = Bezeichnung Renten/Aktien
                  1='Dimension' Renten
                  2=    "       Aktien
                  1...10=Einzelpositionen
                  10 Stellen Text (Name/Bez.)

DIM A(2,10,3)    = Aktuelle Monatswerte Renten/Aktien
                  1='Dimension' Renten
                  2=    "       Aktien
                  1...10=Einzelpositionen
                  1=letzte Stückzahl
                  2=letzter Kurs
                  3=letzter Positionswert (= 1 * 2)

DIM S$(2,10,12) = Bezeichnungen Spar/Sonstige Anlagen
                  1='Dimension' Spareinlagen
                  2=    "       Sonstige Anlagen
                  1...10=Einzelpositionen
                  12 Textstellen u.z.
                     Stelle 1-8=Konto/Bez.
                     Stelle 9-12=Fälligkeit

DIM W(2,10)   =  Aktuelle Monatswerte Spar/Sonstige Anlagen
                  1=Spareinlagen
                  2=Sonstige Anlagen
                  1...10=letzter Postionswert
```



```
DIM G(12,4)   =  Goldwerte monatlich
                 1..12=Monat
                 1=Aktueller Kurs
                 2=Monatsmenge
                 3=akt.Gesamtbestand (= kumuliertes 2)
                 4=akt.Gesamtwert (= 3 * 4)
```

Anwendungsbeispiel

Kategorie	Monate / Vermögensangaben								
	April			Mai			Juni		
Positions-werte	Anz.	Kurs	Wert:	Anz.	Kurs	Wert	Anz.	Kurs	Wert
Renten									
8 RHPF 72	10	101	1010:	20	100	2000	20	103	2060
9 BUND82II	20	105	2100:	20	104	2080	20	106	2120
			3110:			4080			4180
Aktien									
DEGUSSA	5	400	2000:	0	380	0	0	350	0
VW	10	200	2000:	10	220	2200	0	210	0
BMW	7	400	2800:	7	410	2870	7	420	2940
			6800:			5070			2940
Global-werte	Mts. Kurs	Ges. Best.	Best:	Wert:	Kurs	MB	GB:	Wert:	Kurs	MB	GB:	Wert
Gold	32.2	50	50	1610:	34	0	50	1700:	33.8	0	50	1690
Bargeld	-	-	-	520:	-	-	-	1420	-	-	-	3520
Monats-summen				1240:				12270:				12330:

Vorgänge
- - - - -

April : Eingabe der Positions- und Globalwerte mit Namen, Kurs und Mengen

Mai : DEGUSSA wird zu 380 verkauft (neuer Bestand also als Nullmenge eingegeben), der Erlös von DM 1900 wird zum Erwerb von 10 Stück RHPF-Anleihe (Kurs 100) verwandt, der Rest von DM 900 wandert in den Bargeldbestand.

Juni : VW wird zum Kurs 210 verkauft, die Position wird als Nullposition (wie DEGUSSA) fortgeführt. Der Verkaufserlös von DM 2100 wird dem Bargeldbestand zugeführt, der nun mit 3520 als Monatseingabe aktualisiert wird. Gold bleibt mit Ges.bestand 50 Einheiten erhalten, es erfolgt (wie im Juni) keine Monatszu- bzw. -abschreibung.

Anmerkung

Die Werte, welche vom Rechner eingesetzt bzw. fortgeschrieben werden, sind in der Aufstellung gestrichelt kenntlich gemacht.

2. Programm AKT

Zahlenwerke

Das Programm "Aktien" führt die "Vermögens"-Linie fort. Es ist aus einem Börsenprogramm von Autor Rybarczyk hervorgegangen, wurde jedoch völlig umgestaltet und weist nun einen beträchtlichen Komplexitätsgrad auf. Wie bei VERM geht es um die Verwaltung großer Zahlenwerke im Rahmen der häuslichen Finanzen. Noch nicht so speicherplatzsparend wie Programm MKART arbeitet es mit beträchtlicher "Programmtechnik". Einige Tricks bei der Ausnutzung von Fließkommazahlen und eine ausgedehnte Unterprogrammstruktur stellen einige Ansprüche an fortgeschrittene Hobbyprogrammierer.

Zeigen wir einmal in gebotener Kürze, wie solche Zahlenwerke für den ZX zu handhaben sind und wie man zu besserer Speicherplatzausnutzung kommt. - Grundlage der Verwaltung von Zahlen sind die oft verwendeten Arrays, welche in unterschiedlicher Dimensionierung (mit DIM) angelegt werden. Bei "Aktien" interessieren Kurse und Werte von Titeln. Üblicherweise wird man es bei folgender Zielsetzung mit diesen Zahlenmengen zu tun haben: Es soll ein Portefeuille von bis zu 10 Titeln verwaltet werden, dazu kommt ein Index, der ebenso mit seiner Notierung fortzuführen ist. Da der Gesamtwert des Depots verlangt wird, braucht man einen speziellen Summenarray. Für die Darstellung müssen bei richtiger Skalierung mögliche Mini- bzw. Maximalwerte gespeichert werden. Letztere sind sowohl für Höchst/Tiefst der Aktientitel wie auch für den Portefeuillewert anzulegen. Selbst wenn man auf Einzelwerte pro Titel verzichtete, da diese jederzeit aus Menge mal Kurs errechnet werden können, ergibt sich bei 52 upzudatenden Wochen folgender "Zahlenbedarf":

```
10 Einzelaktien jeweils mit Menge + Kurs für 52 Wochen
   = 20 Positionen a 52 Wochen ---------------- 1.040 Zahlen
1 Portefeuille-Gesamtwert pro Woche ----------    52 "
1 Indexwert je Woche fortgeführt -------------    52 "
Zuzüglich Minimal/Maximalwerte zur Darstellung
bedeuten im einzelnen:
10 Einzelaktien mit Höchst- und Tiefstwert ----    20 "
1  Indexwert desgleichen über die gesamte Zeit-     2 "
1  Portefeuillewert ebenfalls als MiniMax------     2 "
                                                -------------
                gesamter Zahlenbedarf           1.168 Zahlen
```

Da bekanntlich jede Zahl in Gleitkommadarstellung 5 Bytes braucht (ohne Namensfelder), ist der Speicherbedarf mindestens mit 5.840 Bytes zu veranschlagen.

Hinzutreten noch Zwischen- und Hilfsgrößen zur Berechnung bzw. Zwischenspeicherung. - Wie kann man das reduzieren? Wie kommt es, daß im Programm AKT der Variablenbereich bloß 4.038 Bytes belegt?
Da beim Programm in erster Linie Verwaltung und Handhabung von Daten zu regeln war, wurde noch nicht alles, was an "Spartechnik" möglich ist, angewandt. (Das ist in MKART geschehen, von weitergehenderer spezieller "Datenkompression" mal abgesehen). Eine Technik sei jedoch

an dieser Stelle vorgestellt, wie sie in AKT Anwendung findet.

Obwohl der Sinclair eine Rechengenauigkeit von 9 1/2 Stellen aufrecht zu erhalten vermag, wird diese Stellenzahl meist gar nicht benötigt. Wenn man nicht gerade Millionär ist, wird man nicht mit fünfstelligen Mengen und Kursen zu "rechnen" haben. Also teilt man den Zahlenvorrat einer Gleitkommazahl auf in Vor- und Nachkommawert. Wenngleich der Verarbeitungsaufwand durch De-/Codierung steigt, hält er sich in Grenzen. Auch die Geschwindigkeit für Portefeuille-Verwaltung und Diagramme fällt nicht allzu ins Gewicht. Folglich sieht die praktische Nutzanwendung im Programm AKT so aus, daß zwei Maßnahmen für Entlastung sorgen:

1. Alle Höchst / Tiefstangaben werden unter der Bezeichnung MIN und MAX auf ein Tabellenelement verteilt:
   MIN-----> Vorkommawert; MAX -----> Nachkommawert

2. Menge und Kurs werden bei Einzeltiteln (Aktien) ebenfalls in einer Zahl untergebracht:
   Kurs ------> Vorkommawert; Menge ------> Nachkommawert

Nachteil der Prozedur ist, daß nur mit positiven Ganzzahlen operiert wird. Vorteil ist die spürbare Speicherplatzeinsparung. Nun sieht nämlich die Rechnung des Zahlenbedarfs anders aus:

```
  10 Einzelaktien a 52 Wochen mit Kurs + Menge ------ 520 Zahlen
   1 Index- und Portefeuillegesamtwert (2 x 52 Wo.)-- 104 "
 Minimax für 10 Einzelaktien und 2 sonstige Werte --  12 "
                                                     ----------
                                      Gesamtbedarf   636 Zahlen
```

Was eine Reduktion zu oben von 46 % ergibt! - Allerdings treten noch hinzu ein Einzelwert für die Zwischenspeicherung der aktuellen Woche (zur Darstellung im Diagramm) - gleich 52 Wochenzahlen - und das dazugehörige Minimum/Maximum; sodaß noch 53 Zahlen zu addieren sind. Aber selbst bei 689 Zahlen bleibt der Zahlenspeicher von nunmehr 3445 Bytes deutlich unter der vorher angestellten Rechnung.

An sonstigen Variablen, die den Variablenbereich dann auf die gemessenen 4038 Bytes bringen, gehören:

- eine Namenstabelle für Aktien, Index etc mit jeweils 13 Zeichen Text als Maximum (ergibt 13x13 =) 169 Bytes

- eine Nachrichtentabelle für Printkonstanten, um direkte Textwiederholungen im Programm zu vermeiden (s. Variablenliste weiter unten) - 15 Texte a 11 Zeichen = 165 Bytes

Summe bis dahin ist dann 3779 Bytes, also noch weniger als sich bei herkömmlicher "Zahlenarbeit" ergeben würde. Der Rest des Variablenbedarfs verteilt sich auf wenige Zwischenrechengrößen, Laufvariablen etc., wie sie die Variablenliste ausweist. Ehe wir auf einzelne Programmfeatures von AKT eingehen, noch ein Hinweis zur VK/NK-Behandlung. Bekanntlich holt man sich den Vorkommawert mit der INT-Funktion, was nicht weiter problematisch ist. Schwieriger ist der Nachkommawert. Zunächst muß er durch Division (mit beispielsweise 10.000) "klein"gemacht werden. Zweckmäßigerweise wird man die 10er Potenz per E-Funktion benutzen. Will man den NK-Wert wieder aus seinem "Versteck" hinter dem Komma hervorzaubern, ist die Multiplikation mit 1E5 nicht das einzige, was zu tun ist. Vielmehr muß erst der VK-Wert eleminiert werden, um zur richtigen Zahl zu kommen. Und da lauert eine gefähr-

liche Integer-Falle beim Sinclair auf den Benutzer:

Dank interner Rundungs-/Rechenungenauigkeiten erscheint beispielsweise statt abgespeicherter 1.200 nur der Zahlenwert 1.999,99.. Wer dann forsch die Rechenreste mit INT abschneidet, erhält eine falsche Zahl, was insbesondere bei Tausenderüberschreitung u.a. unangenehm wird. Abhilfe - so auch im Programm AKT - ist einfach, mal besondere Umstände außen vor gelassen: Die Zahlenklippe wird mit Addition eines Kleinstwertes (etwa 0,1) *vor Anlegen der INT-Klausel umschifft*.

Programmfeatures

Allgemeine Funktionsweise

Das Programm verwaltet nicht allein einen umfangreichen Zahlenpark, es übernimmt auch zahlreiche Verwaltungs- und Darstellungsaufgaben. Die Struktur ist daher kompliziert und kein idealer Einstieg für Anfänger. Unserem Ziel getreu - von Einfachen zum Anspruchsvollen - haben wir eine Reihe recht trickreicher Routinen verwandt. Neben der von Programm VERM bereits bekannten automatischen Skalierung bei Diagrammen ist in AKT die Unterprogrammtechnik vorherrschend. Sie ist nicht leicht zu durchschauen, weshalb sich weiter unten eine Cross-Reference-Liste der Aufrufe findet. Wieso das, dienen doch "Upros" nach herkömmlicher Meinung der Übersichtlichkeit? Nun, bekanntlich kann man nicht alles zugleich haben. In diesem Fall kollidierten sprechende Variablenamen mit For-Nrext-Schleifenanforderungen. NR (= Aktiennummer) und MIN, MAX gehen nicht als Laufvariablen für FOR-Next; sie müssen folglich dauernd umgeshiftet werden. Auch permanente Aktualisierung von Höchst/Tiefstwerten für Einzel- und Gesamtdiagramme erfordert viel Programmaufwand. Gleiches gilt für die beschriebene VK/NK-Rechnerei. Immerhin blieb das Programm insgesamt noch beträchtlich unter dem Umfang der ursprünglichen Vorlage. Dafür besitzt es eine Reihe bemerkenswerter Leistungsmerkmale. Seine eigentliche Funktionsweise kann wie folgt beschrieben werden:

- Zu allererst werden die Aktiennamen eingegeben, - oder es wird ein neuer Name nachträglich zugewiesen. Durch Anzeige der Aktienliste kann der Benutzer wählen, was er in Diagrammen angezeigt haben möchte. Der Index Dow Jones ist vorgegeben, man kann aber auch (per Programmänderung) einen andern "Text" (z.B. FAZ-Index) einfügen. Fußend auf der "Aktienliste" arbeitet das Programm fortan mit einer Aktiennummer (NR).

- Diese eigentliche Verarbeitung ist wochenweise aufgezogen. Nach der Wochenzahl richtet sich Updatingmenue, Diagramm der Portefeuilleaufteilung und Dow Jones-Indexstand. Lesen und Schreiben sind bei Aktien zusammengefaßt. Nach Wochenvorgabe erscheinen vorhandene Titel entweder mit Kurs, Menge und Einzelwert zum Lesen oder (falls leer) zur Zahleneingabe. Besetzte Positionen können mit Tastendruck durchblättert werden, ansonsten erwartet das Programm die Eingabe von Wochenkurs und Bestandsmenge. Beim Index (Dow Jones) kann recht elegant vor- und rückwärts geblättert werden. Neben Kurs wird die Veränderung zur Vorwoche ausgegeben. Unbesetzte Wochen erscheinen mit Nullwert.

- Nach Kurseingabe errechnet das Programm neue Höchst/Tiefst's und aktualisiert die gespeicherten MiniMax'e. Bei Aktien wird nach jeder Aktualisierung der Wochenportefeuillewert errechnet und gespeichert. Aber nur, falls man im Updatemodus nicht den "Notausstieg" (="Keine Gesamtwertfortschreibung") wählt. Beim Lesen wird man nach Kurs-Anlisten auf die Option "Keine Fortschreibung" verfallen. (Sonst werden die vorhandenen Wochenwerte mit ihrem richtigen Wert überschrieben, was kein Beinbruch ist). Nach Gesamtwertfortschreibung erfolgt für den Portefeuillewert ebenfalls eine MiniMax-Prüfung, um für Diagramme immer den richtigen Skalierungsfaktor zur Verfügung zu halten.

- Löschungen sind auch kein Problem. Hat man eine Position verkauft, d.h. die aktuelle Menge auf "Null gebracht", bieten sich zwei Möglichkeiten: entweder der Titel wird nur mit Kurs weitergeführt oder man entschließt sich zur Totaloperation "Löschung".Bei Option "Wochenlöschung" werden nur alle Wochenwerte eleminiert (neben dem Namen selbstverständlich), der Portefeuillewert bleiben davon unberührt. Bei Option "Portefeuillebereinigung" wird der bisherige Gesamtwert aller Titel im nachhinein so gestellt, als habe es die betreffende Aktie nie gegeben. Ansonsten können einmal eingegebene Wochenwerte jederzeit nachträglich mit Kurs und Menge überschrieben werden. Da im Anschluß an jedes Wochenupdate eine Neuberechnung (sofern nicht per "Notausstieg" unterbunden) des Portefeuilles erfolgt, bleibt alles in der Reihe.

- Sowohl Index wie Aktien werden maßstabsgerecht in einer Darstellungsroutine mit aktuellem Wert abgebildet. Skalierung und wochenweise Besetzung erfolgen nach vorhandenen Werten, d.h. ist eine Woche bzw. Position (noch) leer, wird nicht geplottet. Aktien werden nach Kurs wie nach Wert im Diagramm abgebildet, ebenso Portefeuillegesamtwert und aktuelle Portefeuilleaufteilung (Einzelpositionen in % zum Gesamtwert). Letzteres wurde als Balkendiagramm realisiert und beim "Updating" aufgehängt, um zur aktuellen Woche die "Schwere" einzelner Aktienpositionen zu haben. - Dazu kommen zur "Benutzerunterstützung" zahlreiche Anhaltefunktionen, d.h. der User kann Diagramm oder Tabellierung anschauen, bis er durch NL oder Tastendruck fortzufahren wünscht.

Leistungsmerkmale

Nähere Einzelheiten zur Programmstruktur kann man Programmaufbauschema und aufgeführter Programmlogik (beim Hauptzweig "Updating Aktien")entnehmen. Bevor wir Benutzungshinweise geben, soll auf die Hauptleistungsmerkmale von AKT eingegangen werden.

Neben der angesprochenen Speicherplatzreduktion von Zahlen und der "starken" Darstellungskomponente wurde für Datensicherheit und Benutzerfreundlichkeit etwas getan. *Sicherheit* wird bei AKT erreicht durch die nach jedem Updating erfolgende Neuberechnung der Wochenwerte. Als Erstes wurde ein umfangreicher Wochencheck eingebaut. Da die Wochenzahl wichtigster "Zuordner" ist, wird die Eingabe nicht nur auf "Zahl" sondern auch auf Höchstwert geprüft. Ein zu hoher Wert wird auf 52, ein zu niedriger auf 1 gesetzt, damit das Programm weiter arbeiten kann. Zudem kann (beim Dow Jones) mit "V" vor- und mit "N" nachgeblättert werden - ausgehend vom letzten Wochenwert. Damit der User keine unbesetzte Aktien-Nr. verwendet, erhält er bei Änderung / Neueingabe die aktuelle Liste präsentiert. In der wochenweisen Verarbeitung des Inputs werden die "besetzten" Titel vorgegeben, die man auch mit NL überlesen kann. Sofern kein Eintrag vorhanden ist, wird der User gezwungen, pro Titel Kurs und Menge einzugeben. Hat er sich in der Woche geirrt, kommt er mit "RETURN" wieder aus dem Menue; hat er falsche Kurse oder Mengen eingegeben, wird in der Routine solange "rotiert", bis die letzte Eingabe quittiert wurde. Nur sie gelangt in die Weiterverarbeitung. Für den Fall, daß versehentlich die falsche Woche ugedatet wurde, kann der User sie neu anwahlen und überschreiben. Allerdings ist gegen "gewollte" Fehler kein Programm ge-

feit. Wenn etwa versucht wird, eine leere Woche abzurufen (Woche ohne Kurs- und Mengeneintrag) und man nicht rasch mit RETURN aussteigt, tut selbiges das Programm. Multiplikation mit Null nimmt der Rechner dem User übel.., weshalb man auch keine Null- Kurse bzw. Null-Mengen eingeben sollte.

A Konto *Benutzerfreundlichkeit* geht nicht nur die DIA (spricht Diagramm-) - Optik. Auch die Staffelung in mehrere Untermenues mit Rücksprung zur nächst höheren Ebene kommt dem User entgegen. Er kann gleichsam zwischen Einzelwert-, Index,- Kurs- und Gesamtwertdarstellung rotieren. Bei Dow Jones (=Index) kann schnell in Sprüngen zwischen Wochenangaben vor- und rückwärts gelesen werden. Ebenfalls der raschen Übersicht dient das DIA '% Aufteilung Portefeuille'. Es gibt die wochenweise, aktuelle Verteilung der Titel (in %) aufs Gesamtportefeuille wieder. Damit es schnell im gleichen Menue abgerufen werden kann,wurden es beim 'Ast' Updating Aktien (und nicht unter 'Diagramme') aufgehängt. Zahlreiche Quittierungsfunktionen und Möglichkeiten, den Menuezweig 'mittendrin' zu verlassen, dürfen ebenfalls als bedienerfreundlich einzustufen sein.
Anderseits ist es notwendig, der Fairness halber auf Grenzen und Beschränkungen des Programms hinzuweisen. Sie rühren einerseits von Setzungen der gewählten (speicherplatzsparsamen) Zahlenbehandlung, anderseits von den natürlichen Grenzen der Programmierung her. Wollte man alles abfangen und alle Wünsche erfüllen, wären mehr als 7 Seiten Listing und 11,6 K erforderlich - was so gesehen schon reichlich ist.Im einzelnen kann man als Kurs und Menge nur positive Ganzzahlen eingeben. Auch erscheint es ratsam,Gesamtwerte über 5 Stellen zu meiden.

- Kurs und Menge sind zwingend für Abfrage und Update bei Aktien. Wenn versucht wird, 'Nullwerte' einzugeben oder zu lesen, steigt das Programm aus!
- Die Skalierung arbeitet maßstabsgerecht, jedoch kann es zu 'Sprüngen' kommen. Die groben Plotschritte lassen sich nicht ganz mit der tatsächlichen Wertgenauigkeit synchronisieren.
- Eigentlich selbstverständlich dürfte sein, daß bei nur einem Titel bzw. Wochenwert kein DIA gezeichnet werden kann (Ober-und Untergrenzen sind noch nicht festgelegt).Auch eine Nullwert-Prozentverteilung des Portefeuilles stößt auf natürliche Grenzen.

Benutzungshinweise

Um an das zuletzt Gesagte anzuknüpfen, sei bei den folgenden Hinweisen zunächst vor allerlei "Unsinn" gewarnt. An und für sich liegt es auf der Hand, dem Programm nichts Unmögliches abzuverlangen; aber der menschlichen Schwäche sind bekanntlich keine Grenzen gesetzt. Daher zuerst ein paar Warnungen:

- Außer bei Dow Jones vermeide man Null-Kurse und Mengen; man kann zwar Nullkurse eingeben und sofort mit RETURN aussteigen, wird jedoch bei Lesen der Position mit Crash belohnt. (Man kommt ohne weitere Verluste mit GOTO 990 wie auch sonst ins Hauptmenue zurück)

- Ebenfalls meide man Lücken in der Aktienliste. Da nach der Reihenfolge der Liste (sprich NR) wochenweise Lese-/Updatingpositionen präsentiert werden, gelangt man sonst an die erwähnte "Nullposition". Wer beispielsweise Woche Nr. 1 und dann Woche Nr. 3 füllt, erhält als Nr. 2 eine "Lücke" fortan geboten.

- Updating einer "Aktienwoche", ohne einen einzigen Titel im Portefeuille ist ein Ding, das Programm nicht schluckt. Zu Darstellungszwecken muß immer mindestens ein Titel mit Werten, die Höchst und

Tiefstpunkte abgeben, vorhanden sein.

- Zur Rubrik "Kein Dia/Update ohne gültige Werte" gehört eine weitere Erfordernis. Wenn etwa 5 Werte alle mit Kurs 100 vorhanden sind, kann der Rechner kein MiniMax berechnen (es wäre Null!). Um dem entgegen zu wirken ein Trick, mit dem es auch möglich ist, den Skalierungsmaßstab "zusammenzudrücken":

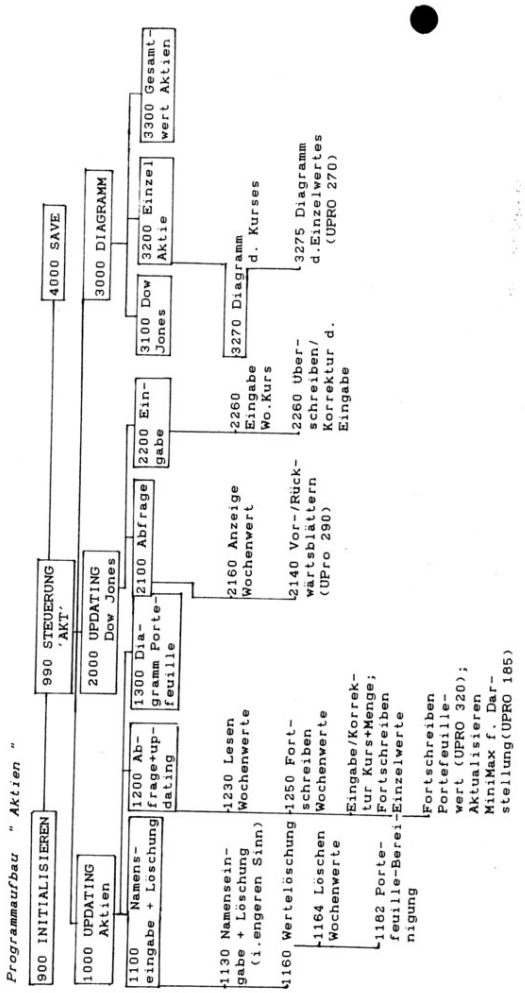
  Mit entfernter Wochenzahl wird ein künstliches Maximum des betreffenden Titels (=Aktiennummer) eingegeben, das man anschließend wieder mit Nullkurs "löschen" kann. Hierzu muß nach 0-Kurseingabe mit dem angebotenen RETURN das Updating-Menue sogleich verlassen werden - ehe es zur Wertberechnung kommt.

Ein regulärer Eingabevorgang wird bei der Kurseingabe wie folgt vonstatten gehen, um damit den Irregulär-Zweig zu verlassen:
- nach Kursanzeige neuen Kurs eingeben
- denselben mit NL quittieren oder mit neuem Betrag korrigieren
- Menge (= aktueller Wochenbestand) eingeben
- ebenso mit NL quittieren oder korrigieren
- nun nächsten Titel "kommenlassen" und wie beschrieben verfahren
- nach Ende Wochengesamtwertfortschreibung (mit NL) quittieren oder "Notausstieg" (bei Lesen) wählen
- neue Woche wählen oder mit RETURN ins Hauptmenue
- Aktienwerte wie beschrieben weiter eingeben bzw. korrigieren

Da der Benutzer jederzeit vorhandene Wochen überschreiben kann, gibt das Programm ihm viel Macht über seinen Datenbestand. Eine ganze Menge Tricks sind zusätzlich über die Namenslöschung möglich. Insgesamt hat der User drei Manipulationsmöglichkeiten:

1. Einfache Umbenennung eines Titels. Beispielsweise heißt "VW" ab sofort "VAG". Weg: Namensänderungszweig anwählen, unter der NR den neuen Namen (max. 13 Stellen) eingeben, fertig.

2. Löschung mit Wochenwerteleminierung. - Wird beispielsweise VW in Woche 12 verkauft, kann der Titel unter der betreffenden NR mit allen bisherigen Wochenwerten gelöscht werden. Nun möchte man aber das Portefeuille nicht mit den Gesamtwerten verfälschen. Keine Bange, bei Option "Wochenlöschung" (Input "-") ändert sich nichts an den Wochensummen - es sei denn, man datet sie nachträglich up! Unter gleicher Nummer kann beispielsweise nun BMW geführt werden. Was der Realität sicher entsprechen dürfte.

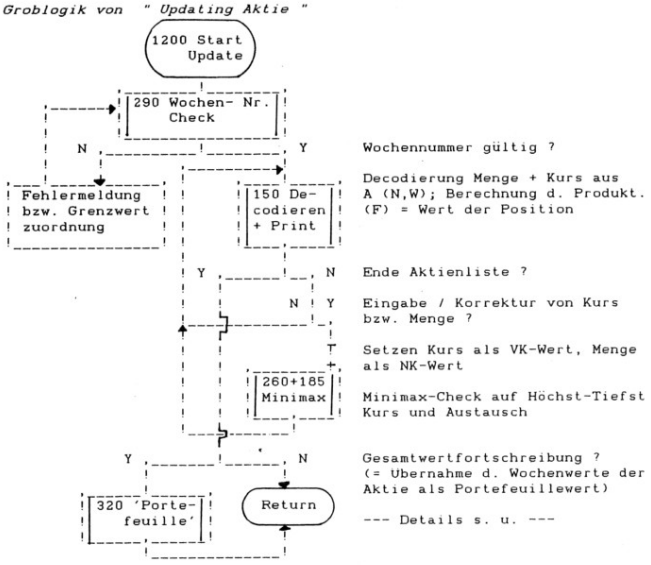


3. Mit der Option "Portefeuillebereinigung" werden hingegen alle vergangenen Wochenwerte aus dem Wertbestand herausgerechnet. Da Menge und Kurs "wochenscharf" gespeichert sind, kein Problem für den Rechner...Sicher eine Radikaloperation für ernste Fälle - z.B. wenn man mehrere Wochen falsch eingegeben hat. Natürlich kann man so "rückwärts" das ganze Aktienfile löschen...Wie gesagt, dem User sind viele Mittel in die Hand gegeben.

Auf eins jedoch sollte man sich beim Programm AKT einstellen, - wenn ein besetztes Portefeuille über die volle Distanz von 52 Wochen upgedatet / abgefragt wird, kommt der Faktor Rechenzeit zum Zuge. Man wappne sich mit etwas Geduld.

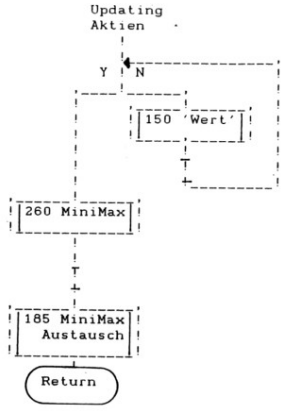
Auf den folgenden Seiten findet der Leser noch eine Reihe von Angaben, die ihm das "Durcharbeiten" des Programms AKT erleichtern sollen. Dazu gehört - der schematische Programmaufbau
- die Groblogik "Updating Aktie"
- eine Aufruftabelle für die zahlreichen Unterroutinen
- und schließlich die übliche, notwendige Variablenliste

Groblogik von " Updating Aktie "



Modul " Portefeuille "

(320 Fortscheiben Gesamtwert)



Löschen Wochenwert aller Aktien
A (11,W)

Letzte Aktie d. Liste ?

Decodieren Einzelwert u.
Berechnen Produkt

Addieren Aktienwert (F) auf
Wochengesamtwert A (11,W) + F

Besetzen der Felder MINI u. MAX
aus bisherigem Gesamtwert
A (11,W)

Besetzen Vergleichsfeld F aus
neuem Wochengesamtwert:A (11,W)

Werteaustausch, sofern neues
Mini-/Maximum vorliegt
(F < MIN; F > MAX)

Abruftabelle (Cross Reference d. GOSUB's)

Zeilen-Nr.	Bezeichnung	Aufgerufene Routine	160 Säulengrafik	200 Wochendia	270 Einzelwertsetzung	310 MiniMax-Check Dow Jones	320 Portefeuille	1000 Updating	1100 Aktiennamen	1200 'Aktien-Woche'	1300 Portefeuille-aufteilung (%)	2100 DJ-Wo.werte	2200 DJ-Wo.eingabe	3100 DJ-Dia	3200 Dia Einzelaktie
150	Wertberechnung (Kurs * Menge)		X		X		X		X	X					
160	Säulengrafik										X				
170	Aktienliste (m. Nr. Wahl)								X						X
185	MiniMax Austausch (-> F)				X		X		X	X					
200	Wochen-Dia													X	(X)
260	MiniMax-Besetz. (aus M( ))			X		X	X		X	X					
270	Einzelwert-Besetzung Aktie (aus Zwischenarray)														X
290	Inputcheck Wo.Nr									X	X	X	X		
310	MiniMax-Check bei Dow Jones												X		
320	Fortschreiben Gesamtwert(=Portefeuille)							X							

Variablenliste

1. Einzelne Variablen
   . . . . . .
   W = Wochennummer (Wert 1-52)
   NR = Nummer der Aktie (gem. Liste)
   Y,X = Positionen für Plot
   (auch zur Wertermittlung der Wochenwerte der Aktien)
   N,M = Laufvariablen
   F = Skalierungsfaktor (auch zum Vergleich in MINIMAX benutzt)
   MIN = Minimum von Werten (zur Skalierung erforderlich)
   MAX = Maximum von Werten (desgleichen)
   I$,E$ = Eingaben

2. Arrays

. . . .

a) Zahlenarrays

DIM A(NR,W) = Wochenwerte Aktien,Index etc. wie folgt

Zuordnung	Inhalt	(VK) Vorkomma-	(NK) Nachkommawert
1 - 10	einzelne Aktie	Kurs (ganz-zahlig)	Menge (= Wochen-bestand)
11	Portefeuille-wert	Wochenwert (aller Aktien)	---
12	Dow Jones	Kurs	---
13	Einzelwert (akt. Woche)	einzelner Wert einer Aktie	---

DIM M(NR) = MiniMax Tabelle

Zuordnung	Inhalt	(VK) Vorkomma-	(NK) Nachkommawert
1 - 10	Höchst/Tiefst-Kurs e. Aktie	Minimum	Maximum
11	Höchst/Tief Portefeuille (als Wert aller Aktien)	"	"
12	Höchst/Tief des Dow Jones	"	"
13	H/T e. aktuell. Aktie (dient d. Zwischenspeich.	"	"

b) Zeichenarrays

DIM A$(NR;13) = Namenstabelle ('Liste')

Nr	Inhalt
1 - 10	Aktiennamen
11	"Portefeuille")
12	"Dow Jones") als Konstanten
13	"W e r t")

DIM N$(15,11) = Nachrichtentabelle

Nr	Inhalt
1	UPDATING
2	AKTIEN
3	DOW JONES
4	ENTWICKLUNG
5	GESAMT
6	DIAGRAMM
7	LISTE DER
8	EINGABE
9	+ LOESCHUNG
10	ABFRAGE
11	WOCHE(N)
12	MENGE
13	WERT
14	EINZELN
15	ALLE

3. Programm P C (Project Control)

Sinclair for Management

Mit Programm PC als Abschluß dieses Kapitels soll gezeigt werden, daß man seinen ZX sehr wohl für Managementaufgaben einsetzen kann. Natürlich erfolgt solches wegen fehlendem Massenspeicher nur in eingeschränktem Rahmen; aber kleinere Verwaltungsaufgaben von Zeiten und Kosten sind lösbar. Wer an seinen ZX81 einen größeren RAM-Speicher als die standardmäßigen 16 K angeschlossen hat, oder einen 48 K Spectrum einsetzt, kann den Anwendungsbereich noch ausweiten. Da alle Programme auch auf dem Spectrum lauffähig sind - der BASIC-Wortschatz ist als Subset im Spectrum-Befehlsvorrat enthalten - können die Grenzen dann weiter gezogen werden. Im Programm PC wurde zudem die ZX81 typische Blockgrafik (beim Balkenplan) so eingesetzt, daß der Spectrum das Programm ohne weiteren Änderungen "frißt". (Lediglich bei den CODEs gibt es bekanntlich zwischen den Rechnern Unterschiede).

Management per Sinclair ist möglich, so sei eingangs festgestellt, ehe wir uns näher mit Einzelheiten der "Projekt-Kontrolle" beschäftigen. Solange man sich im gegebenen "Datenrahmen" bewegt, kann die Verarbeitung von Managementdaten recht zügig erfolgen. Lediglich, wenn viel gerechnet, konvertiert und vor allem sortiert wird, kommt der Faktor Zeit zum Zuge. Im Prinzip gilt auch für Heimcomputer wie ZX und Spectrum - alles was formalisierbar ist, kann vom Rechner erledigt werden. Warum also nicht auch Kosten- und Terminüberwachung? Naturgemäß bewegen sich Lösungen im Rahmen vorgegebener Sinclair Grenzen : Rechen- und Suchzeiten sind lang, bei größeren Datenmengen stößt man schnell an Speichergrenzen; es fehlt vor allem ein fixer Massenspeicher. Immerhin zeigt das Programm PC, daß man bescheidene Managementaufgaben auf seinen Sinclair bringen kann. Zwar wurde gemäß Vorlage nur ein Umfang von 25 Arbeitspaketen ins "Programm-Kleid" gebracht, aber damit sind eine ganze Menge Daten im Zugriff. - Da Projekte meist in Untereinheiten zerlegt werden, kann man mit dem AP-Konzept Termine überwachen, Kosten verfolgen und sogar Meilensteine abfragen. Eine Fülle von Listen ist abrufbar und gibt jederzeit Auskunft über den Stand der Dinge. Damit das Auge mitlesen kann, wurde das Diagramm "Balkenplan" realisiert, welches der optischen Kontrolle von Einzelaktivitäten dient. Allerdings sei vor zu hohen Erwartungen gewarnt: PC kann nur ausgeben, was der Benutzer eingegeben hat. Der Rechen- und Weiterverarbeitungsteil hält sich in Grenzen. Böse Zungen sprechen in dem Zusammenhang von Datenverarbeitung der Methode "garbage in - garbage out" (Unsinn rein - Unsinn raus)! Im wesentlichen funktioniert Programm PC nach dem Prinzip der Karteikarte: Jedes Arbeitspaket wird mittels einer Druckmaske vom User mit Daten "bestückt". Diese bleiben abrufbar und können jederzeit geändert und "gesichtet" werden. Gerechnet wird außer bei den Gesamtkosten nicht

viel, dafür ist der Darstellungsteil gut entwickelt.

PC geht auf eine Vorlage von Autor Rybarczyk zurück, die vom Herausgeber umgearbeitet wurde. Interne Arbeitslogik wie auch die "Philosophie" blieben unangetastet. Das Programm konnte dank einiger Kniffe um ein Viertel vom ursprünglichen Umfang her gekürzt werden. Insbesonders ein neuer Daten-Array-Bereich half viele umständliche Abfragen zu vereinfachen. Neu eingebaut bzw. verbessert wurden:

- ein Maskengenerator (verbessert)
- ein PLOT PRINT, d.h. die Erzeugung von Blockgrafik ohne PLOT-Befehl (!)
- ein Termindatensort (neu), der es gestattet, Arbeitspakete in freier Folge einzugeben, welche trotzdem nach Projektbeginn sortiert erscheinen

Damit sich der Leser bei dem gewiß anspruchsvollen Programm zurechtfindet, ist die Dokumentation stark ausgeweitet worden. Die Zuordnung von Arraywerten zu Nummern der Eingabemaske (siehe Layout weiter unten) ist in einer eigenen Übersicht enthalten - neben der üblichen Variablenliste. Programmaufbau und Routinen sind ebenfalls dokumentiert. Schwierige Logikteile wurden separat aufgeführt. So sei den das Programm PC Lern- und Anwendungsbeispiel dafür, daß man seinen Sinclair auch zu Management-Aufgaben heranziehen kann.

Das Auge arbeitet mit

Im Mittelpunkt programmtechnischer Besonderheiten steht diesmal die Optik. Da wir es uns angedeihen ließen, im Laufe dieses Buches das Augenmerk des Lesers auf bestimmte Lösungsideen zu lenken, stand bei PC dieser Aspekt als "Programmleckerbissen" Pate. Während die "hohe Kunst der platzsparenden Speicherung" dem folgenden Kapitel vorbehalten bleibt, sollte quasi am Rande des Hauptgeschehens die optische Aufbereitung gezeigt werden. Dabei wurden zwei Lösungen, etwas benutzerfreundlich zu präsentieren, realisiert. Zum einen geht es um Anzeige und Eingabe von Daten, zum anderen um Darstellung von Vorgängen als visuelle Größen. Mit anderen Worten - beim Programm PC sind als besondere Routinen ein Maskengenerator und ein Plot-Print eingebaut.

Der *Maskengenerator* kann unabhängig von PC auch in anderen Programmen eingesetzt werden. Zunächst ist ein String (in PC ist das I$) einzusetzen, welcher folgender Konvention genügt:

a) zum Zeilenvorschub dient - hinter dem auszuprintenden Text - das $-Zeichen
b) Zeichen "*" stellt den Positionierer dar für einen Separator. (In PC steht die Zeichenkette, welche auf "*" vom Generator eingefügt wird, in E$).

Die Routine selber ist ab Zeile 20 lokalisiert (s. auch Routinenbeschreibung weiter unten). Nachdem mit I$ und E$ die Zeilen der Druckmaske bestimmt sind, macht der sgn. Maskengenerator (ab Zeile 30) folgendes:

1. Das Listbild wird mit Zeilenvorschub und Einfügen von Text wie Separatoren aufgebaut.
2. Die Positionen, wie sie per Separator gekennzeichnet worden sind, stehen danach in Tabelle W (N) gespeichert (Vorkommawert=Zeile, Nachkommawert=Spalte).

Mit Positionsspeicher W können nun Daten exakt an die Stelle der Separatoren plaziert werden. Eingaben wie Speicherdaten lassen sich ohne viel Aufwand an immer die gleiche Printposition plazieren. Das Ganze erfolgt, ohne umständliche Print-AT-Ketten abzuarbeiten. In PC wurde der Mechanismus mehrfach eingesetzt. Auch das Hauptmenue wird

mittels Maskengenerator aufgebaut, was den ungewohnten "rollenden" Andruck erklärt. Die gesamte Arbeit mit Arbeitspaket-Daten erfolgt über die festgelegte Maske.

Beim zweiten "Leckerbissen" wurde bewiesen, daß exakter "Plot" von Daten nicht unbedingt über den PLOT-Befehl zu erfolgen braucht. *Routine PLOTPRINT* (ab Zeile 446, näheres in der Routinenbeschreibung) bringt das Kunststück fertig, ein sauberes Balkendiagramm allein aus ZX-Grafikzeichen zu erzeugen. Der entscheidende Dreh ist die Zuordnung entsprechender Zeichen für Anfang und Ende. Die eigentliche Routine hat es in sich und ist nur 10 Zeilen lang. Sie ist das Ergebnis mehrerer Stunden Knobelei, ehe auch die Klippe umschifft war, nur einen einzelnen Wochenwert darzustellen. Die Lösung läuft über einen "Gerade"-Schalter, welcher die Wahl des richtigen Grafikzeichens steuert. Zum leichteren Nachvollzug wurde die Logik von PLOTPRINT - ebenso wie die der Maskengenerierung - auf einem Extraschaubild veranschaulicht.

Im übrigen erfolgt die Verwaltung aller AP-Daten durch ein gestaffeltes "Werk" von Zahlenarrays (s. eigene Übersicht). Durch die Mehrfachdimensionierung konnten Abläufe vereinheitlicht und die genaue Zuordnung von Input wie Berechnungen erreicht werden. Da User-Eingaben mittels Masken-NR eingehen, Speicherung der AP-Daten sich hingegen nach dem Array-Subskript richtet, war die Umwandlung von ersterem zu letzterem notwendig. (Die Übersicht der Daten-Arrays enthält in den letzten beiden Spalten diesbezügliche Hinweise). Wie wurde die Zuordnung im Programm realisiert?
Selbstverständlich hätte man über eine Folge von IF-Abfragen gehen können. Das wäre umständlich, aufwendig und verarbeitungsintensiv gewesen. In PC wurde ein anderer Weg eingeschlagen: Die Ansteuerung von Masken-Input-NR zur Elementzuordnung erfolgte über einen *Konvertierungsstring*. Er ist in Zeile 62 mit Besetzung von C$ definiert. Wie man sehen kann, ist die dortige Zahlenfolge (zweistellig) identisch mit der Masken-NR-Folge in der Datenarray-Übersicht. Wählt der Benutzer beispielsweise Nr 9, um die Kurzbezeichnung von Meilenstein 1 einzugeben, erhält das Programm die Zahl 1 zur weiteren internen Verarbeitung. PC weiß nun, daß von Array Y$ (...) das erste Datenelement anzusteuern ist, AP-Nr. X vorausgesetzt, welche die erste Dimension abgibt. Mit der geschilderten Konversion (vereinfacht dargestellt!) werden NR des User-Inputs mit Verarbeitungsnummer bei den Daten-Arrays synchronisiert. Das ist nötig, um elegant das Ursprungskonzept von Arbeitspaketen, Meilensteinen und Projektdaten bzw. -kosten auf eine Tabellenlösung auszurichten. Wenden wir uns nun vom Inneren des Programms zum Äußeren - der sgn. Benutzeroberfläche - zu.

Benutzungshinweise

Wir können uns im Folgenden kurz fassen, da die ausführliche Programmdokumentation manches deutlich macht. Worauf hat der User zu achten, ohne daß er das Programm genauer zu kennen braucht?

Will man *AP's* ändern oder lesen, muß unbedingt der AP-Titel eingegeben sein. Daten können ohne das nicht gelesen werden. Gleiches gilt für Auswertung und Anlistung von Firmen, Meilensteinen und Termindaten. Leerfelder bei der Bezeichnung wertet das Programm als Indiz, daß sonst nichts gespeichert ist. Mit einem Trick ist ein Lesen dennoch möglich: Man steigt mit AP-Nr. im *Input-Modus* ein und schaut sich Vorhandenes an. Es ist nämlich so, daß von Lesen wie Schreiben die gleiche Routine benutzt wird; ergo kann beim Zweig INPUT eine Eingabe unterbleiben und der User sich alles anschauen, ohne etwas zu ändern.

Das *Datum* muß im Prinzip 6stellig eingegeben werden, jedoch kann der Tag auch einstellig bleiben. Beispiel: 10184 ist fürs Programm mit 010184 gleich (= 1. Januar 1984). Jahresübergänge sind ebenfalls möglich, nur muß dann die Projektwoche richtig fortgezählt werden. Also - soll das Projekt ab 1.7.84 starten, erhält die Beginnwoche die Zahl 1, die Woche ab 9.7.84 die Zahl 2 usf. Enddatum 30.6.85 würde als Woche 52 geführt werden. Es kann maximal ein ganzes Jahr (von Woche 1-52) verfolgt werden.

Unbedingt ist auf zeitliche *Parallelität* von Projektdatum und Wochennummer zu achten. Da die Sortierung nach Termindatum geschieht, die Darstellung im Balkenplan aber als Maßstab den Wocheneintrag benutzt, dürfen beide Größen nicht differieren! Der Terminsort ordnet nach AP-Datum, das Diagramm bildet hingegen nach Projektwochen ab. Und der Sort dauert... Sind mehr als 10 AP's zu verwalten, geht das sichtbar in die Zeit. Um das Programm kurz zu halten, wurde als Sort (ab Zeile 920) ein einfacher Austauschort gewählt. Und der ist keiner der schnellsten...

Bei der *Kostenübersicht* wundere man sich nicht über eine Prozentangabe 0,005 %, wenn die Bezugsgröße Null ist. Diese Zahl kommt zustande, weil damit gerundet wird. Damit die Division durch Null vermieden wird (Systemstop!), erscheint bei "abgefahrenen Kosten" = 0 die Zahl 0.005.

Die *Eingabeprozedur* des Benutzers weist wenig ungewöhnliche Eigenheiten auf. Im Prinzip der Sicherheit lag es, möglichst viele Ausstiegspunkte anzubieten. Im einzelnen sind folgende Einzelsteps zu beachten:

- *Ersteingabe*: Mit Eingabe der AP-Nr. wird die Druckmaske aufgebaut, die noch leer ist. Nach NL erscheinen Inputnummern in der Maske (invers), mit denen Texte und Zahlen einzugeben sind (in der Abfolge - NR, NL, Dateninput, NL). Nach jeder Dateneingabe (d.h. nach dem "NewLine") erscheint die Maske mit a l l e n derzeit vorliegenden Daten; Mehrfachkorrektur derselben "items" (Informationselemente) ist also möglich. - Nun kann der zweite, dritte Eintrag mit der angezeigten NR gemacht werden . Die Nummernvorgabe erscheint jedoch nur beim ersten Print; möchte man sie wieder sehen - etwa, weil man die NR-Ordnung vergessen hat -, steigt man mit "RETURN" (="") einfach aus, um mit NL (statt der AP-Nr) sofort wieder einzusteigen. Das Programm nimmt nämlich die letzte AP-Nr, wenn keine neue vorgegeben wird.

- *Fortschreiben*: Der Vorgang ist der gleiche wie oben, lediglich die Vorgabe der leeren Maske unterbleibt. Statt dessen erscheint die mit Daten gefüllte Maske. Durch Tastendruck kommen die Eingabe NR an die Stelle der Daten, um nach erstem Update wieder zu verschwinden. Daß die Input-Nummern nur am Anfang erscheinen, hat einen Sinn: zum einen soll die Updating-Prozedur beschleunigt werden, zum anderen ist realistischerweise davon auszugehen, daß der User die Nummerierung rasch "intus" hat. Die Datenfelder sind nämlich in der AP-Maske fortlaufend durchgezählt...

- *Löschen*: Wie schon erwähnt richten sich die Auswerteprozeduren (Kostenübersicht, Terminliste, AP- und Meilensteinliste, Firmenübersicht und Balkendiagramm) danach, ob die Textpositionen besetzt sind oder nicht. Ein Datenandruck unterbleibt, wenn jeweils AP-Titel, Firmenbezeichnung und Meilensteintext fehlen. Ergo kann eine Löschung (indirekt) vorgenommen werden, indem diese Positionen (NR=1,2,9,12,15,18 - s. Layout der Druckmaske) mit Leerzeichen (Input " ") "überschrieben" werden. Es genügt die Eingabe eines Space. - Möchte man alle Daten "clear" (d.h. weg) haben, kann das gesamte Arbeitspaket per Einzelwerteingabe (mit 0) gelöscht werden; was hinreichende, nicht aber notwendige Bedingung zur Löschung ist.

Natürlich hat das Programm PC auch Grenzen. Beim Umfang von 12 K RAM (allein die Daten für 25 AP's schlucken 4.850 Bytes = 194 Bytes pro Arbeitspaket) konnte nicht alles Wünschenswerte zum "Projektmanagement" realisiert werden. Daher die folgenden Hinweise zur Fortentwicklung und Erweiterung für den Sinclair-Heimprogrammierer:

- PC verarbeitet nur 25 AP's, bei voller 16 K RAM'Ausnutzung könnten bis zu 40 Arbeitspakete verwaltet werden; auch die Arrays sind (neben dem Hilfsarray W (26)) "heraufzusetzen".

- Sort- und Suchzeiten schlagen bei voller Auslastung negativ zu Buche. Änderungen greifen jedoch (z. B. falls das Shell-Melzer-Verfahren angewandt wird) in die Programmstruktur ein und sind recht tiefgreifend.

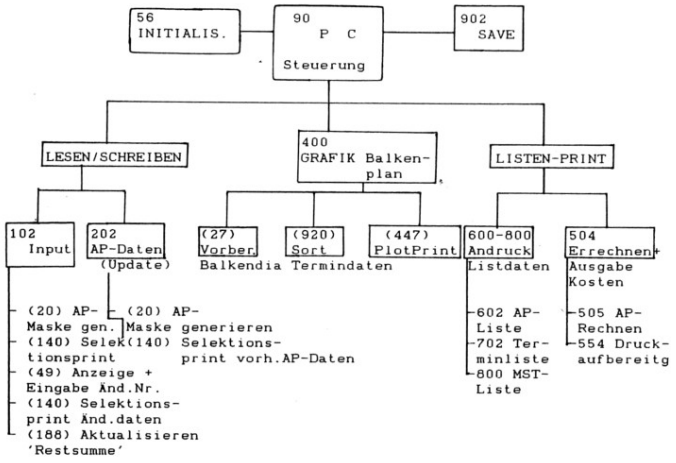
- Alle Einzeldatenänderungen ziehen das "overall"-Überschreiben nach sich; das geschah, um die Array-Verwaltung zu vereinheitlichen. Man sei sich darüber im klaren, daß solches nicht ideal ist. Ideal wäre die gezielte Einzeländerung mit Einzelandruck. Auch die Kostenrechnungen sind unterentwickelt. Eigentlich müßte für ein echtes "Project Control" mehr her als als die simple Summenrechnung bei der "Kostenübersicht".

- Als ersten Schritt zur PC-Programmerweiterung sei die Umrechnung der Wochentermine in Projektwochen angeregt. Das ist sicher nicht einfach (selbst wenn man nur Kalender- und keine Feiertage zugrunde legt), aber machbar. Erwägenswert wäre in dem Zusammenhang auch der "Einbau" einer Nachfolge/Vorgängerbeziehung (bei AP's).

- Verbesserbar sind zudem die Plausibilitäts- und Vollständigkeitsprüfungen. Weder Datum noch Kosten werden formal oder auf Grenzwerte gecheckt, mal ganz von der internen Schlüssigkeit abgesehen. Auf die Synchronisation von Projekbeginn und Wochenangabe ist bereits hingewiesen worden.

Im Folgenden ist nun als "Programmdokumentation" das Notwendige zum Lesen und Verstehen von PC nachgetragen worden.

Programmaufbau von PROJECT - CONTROL



Programmschema von 'P C'

```
18-55   Häufige UPro's   (Unterprogramm-
                          Routinen)
        ------------------------------------------
          20      Generieren
           .      Druck-
           .      Maske
          47
        ..........................
          49      Menue-
           .      Zahlen-
           .      Print
           .
          55
        ------------------------------------------
57-88   Werte-Initialisierung
        ------------------------------------------
          57      Dimensionieren
           .      Arrays und
           .      Laden Konvertierungs-
          67      String (C$)
        ------------------------------------------
          80      'Welcome'
        ------------------------------------------
           .      Print Begrüssungs-
           .      text und Eingabe Projekt-
          88      titel und -Nummer
        ------------------------------------------
90-101  | H P - Steuerung
        |       (Menueanzeige + Wahl)
        ------------------------------------------
102-910         Haupt-Programmzweige
        - - - - - - - - - - - - - - - - - - - - -
          von     0=Initialisieren
           .
           .
          bis     10=Stop
        ------------------------------------------
920     UPro    'Termindatensort'
        - - - -
          940     Datum umgruppieren,
           .      (Austausch-)Sort und
          946     Abspeichern Reihenfolge (sortiert)
        ------------------------------------------
```

Erläuterungen zu besonderen Programm-Routinen

1. Routinen
. . . . . .

447-476 PLOTPRINT

Zeichen eines Balkendiagramms mit Blockgrafikzeichen, die damit verbundene Zeilen-/Spaltenführung erfolgt einzig mittels der TAB-Funktion. Im einzelnen:

- Entnahme der AP-Nummer aus Hilfstabelle W(M) als Nachkommawert
- Bestimmen Anfangs- und Endewoche mit Check auf Grenzwerte
- eigentliche "Plot-Routine" (Zeile 455-468) wie folgt:
  . Zuordnen des richtigen Grafikzeichens für Anfang und Ende des "Terminbalkens" (s. Einzellogikbild)
  . Sicherung gegen Vorgabe eines einzelnen Wochenwertes
  . maßstabgerechte Tabulierung und Print

550-559 Bei Kostenübersicht --> Druckaufbereitung Arrayzahlen

Rundung und längenmäßige Ausrichtung der Zahlen für Ausgabe Einzel- und Gesamtkostenwerte

920-939 TERMINSORT

Austauschort für Terminliste und Balkendiagramme; Sortkriterium ist der jeweilige Projekt-Beginn (AP-Masken-Nr. 3), wie er in Array X (X, 1, 1) gespeichert ist (X = jeweilige AP-Nummer). Da das Datum lediglich in Monat und Jahr zweistellig einzugeben ist, dreht die Routine das Datum um, um in der Folge "Jahr/Monat/Tag" zu sortieren. Der Sort benutzt den Arbeitsarray W (26) - s. Einzellogikbild - und arbeitet wie folgt:

. (920-930) Besetzen von Array W (..) mit dem "umgedrehten" Datum als Vorkommawert und mit der AP-Nummer als Nachkommawert
. (940-946) Elementaustausch, wobei das unbenutzte 26. Element als Zwischenfeld benutzt wird (Memo: Programm PC arbeitet nur mit maximal 25 AP's, der Array wird mit DIM W (26) ein Element höher dimensioniert).
. Nach der Sortierung stehen die AP-Nummern (als NK-Werte) nun im W-Array (als "richtige" Terminsortierung) und können für Terminliste bzw. Balkendia benutzt werden.

2. Von Routinen aufgerufene Unterroutinen

20 - 47 Aufbau Druckmaske

Erzeugen einer Maske nach vorgegebenem Text (in I$), mit Einfügen von Separatoren (in E$), Positionsspeicherung Zeile, Spalte (in W-Array als VK-/NK-Wert) und Vorbereiten des Prints in der Routine (ab 31); wesentlliche Schritte sind (Einzelheiten s. Einzellogikbild):

. Aufbau Maske und Speichern der Druckpositionen in W (..); Regel: $=Zeilenvorschub, *= Positionsmarkierung für Separator
. (ab Listingzeile 30) eigentlicher Maskengenerator, wie in Einzellogikbild erläutert

49 - 55 Druckmaskenprint für Menue (als Variante zu oben)

140 - 180 SELEKTIONSPRINT

Andruck aller vorhandenen AP-Daten (gem. AP-Nr.), Vergleich mit Änderungsnummer (=Eingabe-NR gem. AP-Maske) und Austausch der Daten durch User-Input; Erläuterung:

. Jede "Dimension" der Datenarrays hat gem. Konvertierungsstring C$ eine Nummer zugeordnet erhalten.
. Die Input-Nr. des Users, die seinen Eingaben gem. AP-

Maske voranstellt wird, wird per Konversion mit der "Arraynummer" verglichen; ist sie synchron und liegt ein Dateninput vor (Bedingungen - 1.B-NR und 2. I$ gefüllt), erfolgt Datenaustausch.
. Bei Nur-Lesen setzt das Programm NR auf Null, sodaß nur Andruck und kein Austausch erfolgt.

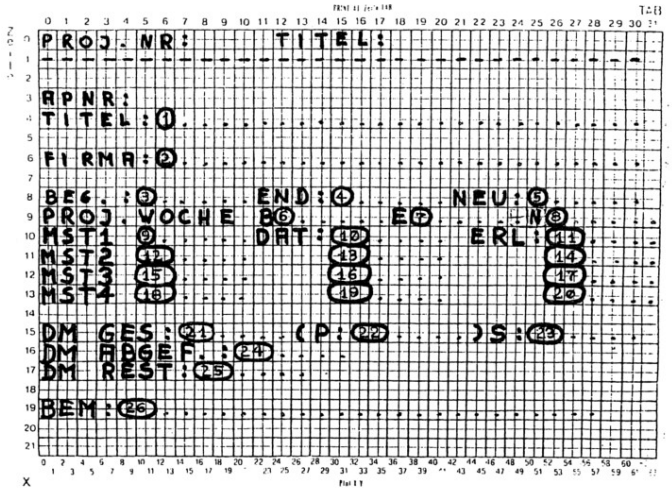
182 - 186 Codetransformation (für Selektionsprint)

Mittels der Änderungsnummer des Users (gem. AP-Maske) wird aus Codestring C$ die Codezahl (B) entnommen, die einen Zugriff zum passenden Array ermöglicht; ist sie gleich der laufenden (Print)Position im Selektionsprint (d.h. B=A), so erfolgt die oben erläuterte Datenänderung.

188 - 192 Errechnen "Restsumme"

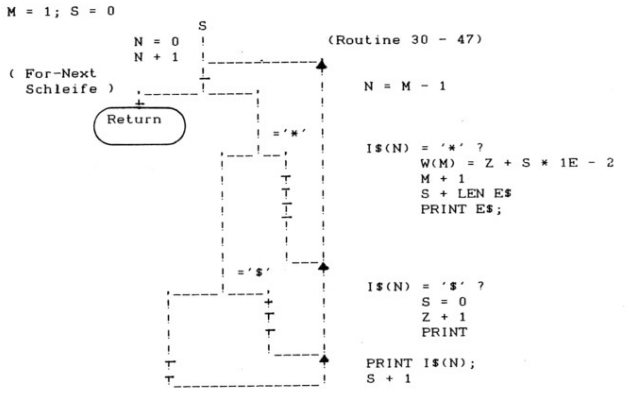
Damit nach jedem möglichen Input im "Arbeitspaket" - genauer: AP-Datenbestand - die richtigen Summen stehen, erfolgt die Aktualisieren nach jedem Durchlauf des "Input-zweigs"! Dies erfolgt daher immer nach Routine 140, nicht jedoch nach Nur-Lesen im Rahmen des Programmzweigs "AP-Daten" (Programm ab Zeile 202).

Druckmaske für Arbeitspaketdaten (Nummer = Input-Nr.)

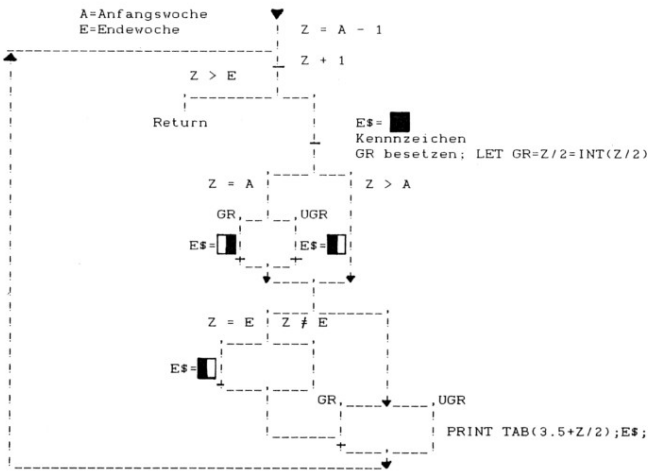


(Die Legende befindet sich in der Übersicht Daten-Arrays)

Einzellogik 'Drucksteuerung Maske' (Masken-Generator)



Einzellogik PLOT PRINT (Kernstück Zeile 455 - 468)



	Bedingung	Zeichen für Anf	Zeichen für Ende	
A = E	Zähler Z=GRade	◨	◨	Logikhilfe für Zuordnung der Anfangs- und Endezeichen
A = E	Z=UGR	◧	◧	
E - A > 1	GR	◨	■	
E - A > 1	UGR	■	◧	

Variablenliste (Daten-Arrays s. gesonderte Übersicht)

1. Stringvariablen

A$	Projektnummer (dreistellig)	) werden nur bei
BS	Projekttitel	) Initialisierung vergeben
C$	(Konstante) Konvertierungsstring f. Besetzung bzw. deren Zugriff (gem. AP-Input-Nr)	
E$	String für Eingaben, Druckfunktionen und zur Datumsumbesetzuung	
I$	String für alle User-Eingaben und zur Maskenerzeugung	

2. Numerische Variablen

W (26) Arbeitsarray zur Positions- und Termindatenspeicherung
A Hilfsvariable - Anwendung: Maskenkonvertierung, Auswahl Datenarray, Termindatensort, Anfangswert f. Balkendiagramm etc.
B Nummer Datenarray bei Selektionsprint
C Schalter bei Selektionsprint; Variable bei Druckaufbereitung Kostenübersicht
GR Schalter "GERADE" bei Plot Print
NR Maskennummer bei AP-Input
N Laufvariable f. For Next-Schleifen
M dito, sowie weitere Sonderverwendungen (vgl. Routine 20)
X aktuelle AP-Nummer (= Nummer des Arbeitspaketes)
S Spaltenzähler bei Maskenaufbau
Z dito, ferner Laufvariable bei Plot Print

Übersicht DATEN-ARRAYS (X = 25, d.h. vorgegebene Anzahl AP's)

1. String-Arrays

DIM Y$ (..) + X$ (..) = Texte pro Arbeitspaket

DIM Y$ (X, 4,6)

Elemente	Masken-Kürzel	Bezeichnung	Masken-NR	Array-Verarb.Nr.
Y$(X,1-4)	MST 1-4	Kurzbezeichnung von Meilenstein 1 bis 4	9 12 15 18	1 2 3 4

DIM X$ (X, 3,25)

X$ (X,1)	AP-TITEL	Titel des Arbeitspakets	1	5
X$ (X,2)	FIRMA		2	6
X$ (X,3)	BEM	Bemerkung	26	7

2. Zahlen-Arrays

DIM Y (..) + X (..) = AP u. MST-Daten pro "Paket"

DIM Y (X,5,2)

Y (X,1,1)	MST1-DAT	Datum Meilenstein 1	10	8
Y (X,1,2)	MST1-ERL	Erledigungszeitp.MST1	11	9
Y (X,2,1)	MST2-DAT	Datum Meilenstein 2	13	10
Y (X,2,2)	MST2-ERL	Erl.zeitpunkt MST 2	14	11
Y (X,3,1)	MST3-DAT	Datum Meilenstein 3	16	12
Y (X,3,2)	MST3-ERL	Erl.zeitpunkt MST 3	17	13
Y (X,4,1)	MST4-DAT	Datum meilenstein 4	19	14
Y (X,4,2)	MST4-ERL	Erl.zeitpunkt MST 4	20	15
Y (X,5,1)	DM ABGEF	Kosten in DM abgefahren	24	16
Y (X,5,2)	DM REST	Restsumme Kosten (wird v.Progr.errechnet)	(25)	17

DIM X (X,3,3)

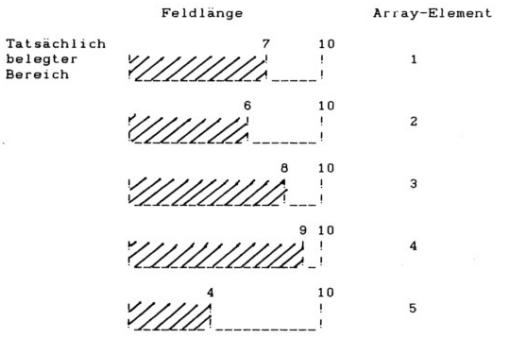
X (X,1,1)	AP-BEG	Beginn Arbeitspaket	3	18
X (X,1,2)	AP-END	Ende Arbeitspaket	4	19
X (X,1,3)	AP-NEU	neues Datum AP	5	20
X (X,2,1)	PROJ.WO B	AP-Beginn in Proj.Wo.	6	21
X (X,2,2)	PROJ.WO E	A-Ende in Projekt-Wo.	7	22
X (X,2,3)	PROJ.WO N	neues Datum in P.Wo.	8	23
X (X,3,1)	DM GES	Ges.kosten in DM	21	24
X (X,3,2)	P	aktuelle Proj.Woche	22	25
X (X,3,3)	S	Summe AP-Wochen	23	26

Kapitel V:

Daten und Verarbeitung

1. Die große Verschwendung

Geht es um die Verwaltung großer Datenbestände (Zeichenmengen), wird häufig die bequeme Array-Lösung verwandt. Da jedes Feld (indiziert) ansprechbar ist, kann ein entsprechender "Feldvorrat" leicht mit einer FOR-NEXT-Schleife durchsucht werden. Auch durch direkte Adressierung jedes Feldes mit Index ist ein Updaten einfach. Andruck und Durchsuchen eines solchen "Files" ist unproblematisch. Allerdings muß immer die entsprechende Tabellen-Nummer vorgegeben werden, weshalb man sie häufig in einer getrennten Zahlentabelle speichert. Möchte man beispielsweise bei 50 Textfeldern einen anderen Zahlenbereich als die Nummern 1 bis 50 verwenden, muß die abweichende Nummerierung an anderer Stelle festgehalten werden. Daneben haben feste Feldlängen - sprich eine Tabellenlösung - einen viel gravierenderen Nachteil: Es wird viel Platz verschenkt. Gesetzt den Fall, man hat einen Array mit einer maximalen Feldlänge von zehn Zeichen, so wird eine Besetzung folgendes Bild ergeben:



Bei einer Feldlänge von 10 Zeichen und nur 5 "Records" (= Sätze als logische Informationseinheiten) wird an Speicherplatz mindestens 50 Bytes benötigt. Beim aufgezeigten Beispiel werden davon nur eine geringe Anzahl ausgenutzt. Es ergibt sich lediglich ein echter Platzbedarf von (7 + 6 + 8 + 9 + 4 =) 34 Zeichen. Folge einer solch bequemen Lösung - ein Drittel des Speicherplatzes wird nicht genutzt, steht leer.

Auch beim Handling gibt es Probleme. Sie gruppieren sich vor allem um zwei Dinge:

1. Buchführung - Welcher Satz ist besetzt, welcher leer? Welches ist der nächste freie Satz? Wo sind Lücken? Was wurde geändert? Wo soll ein neuer Satz hin?

2. Löschen und Updaten - Neben der generellen Frage "Wo steht was?" kommt es insbesondere auf die ökonomische Verwaltung rein- und rausgeschobener Sätze an.Fragen sind dann: Wo soll ein neuer Satz hin? Was kann überschrieben werden? Wie ist der Array neu zu organisieren (Kennzeichen setzen, Neusortierung, "Hochschieben" etc.)?

Oftmals ist zur Buchführung sogar eine Zahlentabelle erforderlich. Eine bessere Lösung wird nun so aussehen, daß mit variabel langen Feldern gearbeitet wird. Lediglich die Satzlänge (= als Summe der einzelnen Felder) ist fest vorgegeben. Nach wie vor arbeitet man mit einem Array. Da die Einzelfelder in ihrer Länge variieren, erreicht man hingegen eine bessere Platzausnutzung. Das wiederum muß mit einem eigenen Verwaltungsteil erkauft werden. Er sieht vor, daß die (variablen) Feldpositionen getrennt in einer Positionstabelle festgehalten werden. Die Trennung zwischen Suchargument (d.h. den Feldern) und Positionierung ist der eigentliche Dreh- und Angelpunkt zum Speicherplatzsparen. Der Vorteil variabel langer Einzelfelder bedeutet erhöhten Aufwand bei der Bestimmung des einzelnen Feldes. Es muß getrennt nach Lage und Länge bestimmt und qualifiziert werden. Beides ist vorzugeben. Insgesamt sind also - so wird man bei einem Ansatz in dieser Richtung sagen können - für jedes einzelne Feld zwei Angaben notwendig:

1. Anfangsposition innerhalb des Arrays (Welches Element? Wievieltes Feld?) und
2. Endeposition, was gleichbedeutend mit Länge und Anfang des Feldes ist.

Obgleich man bei solch halb-optimaler Lösung zu besserer Speicherausnutzung kommt, bleibt der Nachteil umfangreicher Positionsrechnungen. Das läßt sich allerdings bei SINCLAIR-String-Operationen kaum vermeiden. Möchte man den Vorteil variabler Felder nutzen, muß man sich schon die Mühe genauer, aktueller Qualifizierung (= "Stellenbestimmung") machen. Derartige Operationen lassen sich nicht vermeiden. - Etwas anderes läßt sich jedoch verbessern, weswegen wir aus anderem Grund von einer halb-optimalen Lösung gesprochen haben:
Die Positionsspeicherung in einer numerischen Tabelle. Jede Verwendung von Floating-Point-Zahlen kostet Speicherplatz. Da in der Regel mit Ganzzahlen operiert wird, ist eine supergenaue Zahlenwahl auch garnicht notwendig. Verzichtet man auf den Komfort von Gleitkommazahlen, kann man auf dem Gebiet eine Menge tun. Das sei im folgenden näher beleuchtet, ehe wir zur - unseres Erachtens - optimalen Lösung bei der Verwaltung großer Daten kommen.

2. Stringpositionierungen

Zweierlei Daten

Zur Erinnerung - Strings eignen sich vorzüglich zur Datenspeicherung, seien es Zahlen, seien es Zeichen. Ehe wir die Lösung in einem Beispielprogramm präsentieren, möchten wir den Leser weiter Schritt für Schritt dorthin (und insbesondere zum "Warum so?") hinführen.
Strings haben beim Sinclair drei Vorteile:

Erstens - sie ermöglichen die vollvariable Datenspeicherung, d.h. verschieden lange Felder

Zweitens - mit ihnen ist, dank der Aufteilungsmethode des Sinclair, ein elegantes Handling möglich (Updating durch Verkürzen, Einfügen und Verlängern der Zeichenkette).
Drittens - Da kaum verschenkter Platz bleibt, wird weniger Speicher benötigt.
Einziger Nachteil - umfangreiche Buchführung zur Positionierung und damit ein anspruchsvoller Programm-Verwaltungsteil ist von Nöten.

Zunächst sind die Daten in einen variablen und festen Teil aufzuspalten. (Dies ist der erste Schritt). Beispielsweise wird man Anschriften oder andere variabel lange Textteile in einen eigenen Gesamt-Datenstring packen. Wie im Textprogramm werden Dinge wie Name, Vorname, Straße und Stadt dort abgelegt.

Man könnte auch mit Separatoren arbeiten und alle variablen Felder hintereinanderhängen, wobei sie durch zwei unterschiedliche Trennzeichen voneinander unterschieden werden.

Der bessere Weg

Es geht auch anders, als im String selber herumzusuchen und jedes Zeichen auf Separatoren abzuprüfen; vielmehr werden alle Eingaben vollvariabel im String hintereinander abgelegt. Man bewerkstelligt aber die Positionierungsarbeit durch eine reine Längentabelle. Sie enthält (neben der Länge der im Beispiel gebrachten 5 Felder) als erstes ein zusätzliches sechstes Feld. Es nimmt die Gesamtlänge des RECORDS auf. Die Positionstabelle wird nun bei RECORDS mit dem DIM P (N,6) dimensioniert. Die sechs Längenangaben jedes Satzes sehen etwa so aus:

```
P (N,1) = Länge RECORD
P (N,2) = Länge Name
P (N,3) = Länge Vorname
P (N,4) = Länge Stadt
P (N,5) = Länge Straße
P (N,6) = Länge Berufsbezeichnung
```

Ist bekannt, der wievielte RECORD gesucht wird, findet man die Anfangsstelle im String durch folgende Zählschleife. Dabei gilt ST = N ter RECORD, POS = Anfangsposition.

```
LET POS  =  1
FOR X    =  1 TO ST-1
LET POS  =  POS + P (ST,1)
NEXT X
```

Anmerkung: Die Routine funktioniert auch für die erste und letzte Stelle bzw. für den ersten und letzten Satz im String.

Nach Durchlauf enthält POS den Beginn des gesuchten Nten Satzes. Er kann in einem Zwischenfeld Z$ wie folgt gespeichert werden:

```
LET Z$ = A$ (POS TO POS + P (X, 1) - 1)
```

Möchte man eine Feldauswahl haben, braucht man lediglich die Folge-Längen aus P (N,2,usf.) und holt die Felder aus dem Zwischenstring heraus. Statt sich mit "echten" Zahlen und damit viel Speicher abzugeben, gibt es noch einen anderen Weg. Der zweite *Schritt* sieht folglich so aus:
Die variablen Daten werden hintereinander abgelegt, ihre Feldlängen

plus Satzlänge kommt nun aber in eine Code-Tabelle. Da diese Zeichen- und nicht Zahlencharakter hat, wird sie als Positions-Array angelegt. Zusätzlich enthält sie weitere feste, d.h. der Länge nach nicht variierende, Daten. Folglich wird man festen und variablen RECORD-Teil etwa so anlegen:

Variabler Datenteil

```
1. Record    Feld 1    !    2    !  3   !    4     !! 2.Record
-------------------------------------------------------------------
             Name          Vorname     Stadt    Strasse
```

Feste Stammdaten (String N$)

```
1. Record  ! Rec.  !Längenfelder     !KZ   !   Geb.Datum !  (codiert)
           ! länge !----!---!----!--!--!--!---!-----!----!
                   ! Name   Vorn. Stadt  ! Geschl.       !
                                    Strasse

           ! 1 ! 2 ! 3 ! 4 ! 5 ! 6 ! 7 ! 8 ! 9 !
2. Record  !---!---!---!---!---!---!---!---!---!
```

Bezeichnen wir den so geschaffenen String N$ als Stammdaten-String, wird er im Endzustand die obigen Angaben in folgender Reihenfolge enthalten:

- Länge RECORD
- Länge Name
- Länge Vorname
- Länge Stadt
- Länge Strasse
- Kennzeichen Geschlecht
- Geburtsdatum

Das Geburtsdatum kann wie der Rest als Code eingegeben werden, da seine wesentlichen Bestandteile (Jahr, Monat und Tag) die Code-Grenze 255 nicht überschreiten. Folglich kommt man mit drei Bytes beim Geburtsdatum aus:

1. CODE für Jahr (nur letze zwei Stellen)
2. CODE für Monat
3. CODE für Tag

Das Kennzeichen für Geschlecht enthält beispielsweise als Zeichen-information =
+ -> weiblich
- -> männlich.

Aber man kann hier noch mehr Information in das eine Byte packen. Da der gesamte CODE-Bereich von 0 bis zu 255 zur Verfügung steht, können beispielsweise zusätzliche Statusinformationen hineingebracht werden, ohne daß die Information (weiblich/männlich) verloren geht.

Eine Decodierungsroutine für beispielsweise den 2. Satz sieht dann wie folgt aus (Die Statusinformation steht im String-Array N$ an sechster Stelle, sie wird durch die Variable STATUS aufgenommen und ausgedruckt):

```
LET STATUS  =  CODE N$ (2, 6)
IF STATUS > 10 THEN PRINT "WEIBL."
LET STATUS  =  STATUS - 10 * (STATUS >10)
```

```
IF STATUS   =  1 THEN PRINT "ANFAENGER"
IF STATUS   =  2 THEN PRINT "FORTGESCHR."
IF STATUS   =  3 THEN PRINT "PROFI"
```

Positionierungen

Zurück zur eigentlichen Positionierung. Wie erwähnt hilft das Längenfeld in der Positions-(String-)-Tabelle mittels schneller Zählroutine, die passende Stelle im Datenstring zu finden. Wie aber weiß man, welcher-genauer der wievielte-RECORD gesucht werden soll? Damit stehen wir auch vor einem anderen Problem. Wie soll man Ordnung Mitgliedsnummernkreis bringen, ohne dauernd sortieren zu müssen? Die Lösung ist einfach - wie im LOTTO-Programm wählt man den Weg der Direkt-Adressierung. Da Strings speicherarm sind, sind nur ein paar Leerbytes zu verkraften. Soll beispielsweise der Bereich 1 bis 100 bei den Mitgliedsnummern abgedeckt werden, kommt man mit einem String, der mit DIM M$ (100) definiert wird, aus. Zunächst wird er bei leerem Datenfile aus Nullen bestehen. Wird ein RECORD eingetragen ,erhält es die zugehörige Mitgliedsnummer, seine Stelle wird direkt in den M$-String eingetragen. Abfragen erfolgen ebenfalls durch direkte Indizierung. Möchte man beispielsweise wissen, als wievielter Datensatz Mitgliedsnummer 13 abgespeichert wurde, ergibt folgende Operation den passenden Wert (5) im Feld RECNUM:

```
LET RECNUM = CODE M$ (13)
```

Was passiert nun, wenn Mitglied-Nummer 13 neu bzw. wieder mit Daten besetzt werden soll?
Die in der Folge laufenden Operationen müssen sich um drei Strings kümmern:

Variabler Datenstring A$ - Hier wird der fünfte RECORD aufgesucht und angesprochen.

Fester Datenstring N$ (5) - Er liefert die einzelnen Feldlängen und ermöglicht erst eine Positionierung von A$.

Mitgliedsnummern-String M$ - Er liefert durch Abfragen der 13. Stelle (=Mitgliedsnummer)die Information, daß auf den fünften RECORD in A$ (und mittels M$) positioniert werden muß.

Das Ganze läuft im Programm in umgekehrter Reihenfolge ab. Zur Veranschaulichung des Prinzips sei das Gesagte noch einmal kurz rekapituliert.
1. Abfrage nach Mitglied Nummer 13 ergibt - Ja, das Mitglied ist vorhanden, sofern in M$ (13) etwas anderes als 0 steht. Die entsprechende Stelle (= MST), d.h. der wievielte RECORD, kann im Programm wie folgt abgefragt werden:

```
IF NOT CODE M$ (MNR) THEN GOTO...
PRINT "MTGL.VORH."
LET MST = CODE M$ (MNR)
```

2. Aufsuchen der RECORD-Stelle im Datenstring - Dazu wird die Stelle MST benutzt; der feste Datenstring enthält im ersten Byte die Länge eines jeden RECORDS. Also wird einfach bis zur fünften Stelle (genauer -RECORD-Anfang) aufgezählt. Die folgende Routine liefert die Position, wo im Datenstring (A$) der fünfte Satz beginnt.


```
LET POS   =  1
FOR N     =  POS TO MST - 1
LET POS   =  POS + CODE N$ (N,1)
NEXT N
```

3. Einlesen des RECORDS - Da POS auf RECORD-Anfang zeigt, kann nun beispielsweise der Name als erstes Feld des Satzes ausgelesen und verändert werden. Die Feldlänge bekommt man wiederum aus M$. Steht sie als zweiter CODE in der Stringtabelle, wird die Länge zur Anfangspositionierung hinzugezählt, um das Zwischenfeld Z$ richtig zu besetzen:

```
LET Z$ = A$ (POS TO POS + CODE N$ (N,2) - 1)
```

Anmerkung: N ist korrekt, da nach Schleifen-Ende N um 1 höher steht als bei der POS-Besetzung. Damit hat N dann die MST- Zahl erreicht.

Um das Updating zu vereinfachen und Neueintragungen eleganter (als beim Längen-Array N$) zu gestalten, kann man eine weitere Verbesserung vornehmen: Statt als String-Array wird N$ undimensioniert angelegt. Eintragungen werden durch Verkürzen und Verlängern (bei fester Feldlänge!) bewerkstelligt. Für die Programmroutinen bedeutet das keine Umstellungen, wohl jedoch beim Handling von Änderungen. Die obige FOR-NEXT-Schleife arbeiten bei einem fünfstelligen Längenfeld lediglich den N$-String in vorgegebener STEP-Größe ab. Die Routine sieht dann so aus:

```
LET POS   =  1
FOR N     =  POS TO 5 * (MST - 1) STEP  5
LET POS   =  POS + CODE N$ (N)
NEXT N
```

Diese kleine Änderung wird sich noch als sehr hilfreich erweisen. Im Folgenden möchten wir abschließend zeigen, wie Löschungen und Neueintragungen ohne hohen Verwaltungsaufwand erfolgen. Anschließend wird der geneigte Leser gebeten, solchermaßen vorbereitet das Programm "Mitgliederverwaltung" zu " genießen".

Das Verkürzungs- / Verlängerungsprinzip

Weswegen von der ursprünglichen Positionstabelle abgegangen wurde und die Feldlängen in einem String hintereinander abgespeichert werden, hat einen triftigen Grund: Änderungen nutzen die vorgegebenen Sinclair-Möglichkeiten der Stringmanipulation voll aus. Die beiden variabel angelegten Strings - Daten- und Positions-String - arbeiten nach dem Prinzip der Verkürzung und Verlängerung. Der Vorgang ist in einem kleinen Schaubild weiter unten für die drei Möglichkeiten Löschung, Neuaufnahme und Austausch näher aufgezeigt. Zunächst zum Einfachen - der Neuaufnahme von Sätzen.
Soll ein neues Mitglied mit seinen variablen Daten (Name, Anschrift etc.) eingegeben werden, wird zunächst im Stellenstring nachgefragt, ob die angegebene Mitgliedsnummer frei ist. Zeigt der String eine 0 an, kann die Nummer vergeben werden. (Die freie Stelle, d.h. als wievielter RECORD die Neueintragung angelegt wird, kommt man einfach durch Division der Länge des Positionsstrings durch die jeweilige Zahl Stellen, die jeder RECORD als Längenfelder beansprucht.) Der Positionsstring bekommt nun die neuen Feldlängen einfach hinten drangeklebt. Gleiches geschieht mit den variablen Daten - sie werden an den bestehenden Datenstring angehängt. Das ist alles... Bei der Löschung ist es etwas schwerer. Zunächst wird der entsprechende RECORD aufge-

sucht, d.h. die Einzellängen werden aufaddiert, bis die richtige Position erreicht ist. Dann wird der Datenstring um die im Positionsstring gespeicherte Länge verkürzt und der Positionsstring ebenfalls um seinen festen Teil "amputiert". Da jedoch die Positionsangaben im "Stellenstring" nicht mehr stimmen, müssen alle Eintragungen überprüft werden. Auch das ist unproblematisch: Ergibt sich bei der Abfrage, daß eine RECORD-Nummer größer als die geloschte ist, wird sie kurzerhand um eins herabgesetzt und schon stimmt die gesamte Buchführung!

Das eigentliche Updating, d.h. der Austausch von Informationen, ist nicht ganz so schwer. Zunächst ändert sich ja im "Stellenstring" nichts, die alte RECORD-Nummer bleibt erhalten. Auch der Positionsstring ändert sich nur insofern, als Gesamt/Feldlänge den neu eingegebenen Daten angepaßt werden. Diese erscheinen im Datenstring mittels der bekannten Sinclair-Aufteilungstechnik.
Wie man sehen kann, läuft das Ganze im Prinzip einfach ab;aber durch dauernde Codierung und Decodierung begibt man sich wieder des Vorteils, der durch speicherplatzarme Ablage gewonnen wurde. Eine Menge Routinen sind notwendig, um zu positionieren, die Längen hochzurechnen, Feldauswahl und Feldbesetzung vorzunehmen und anderes mehr. Insbesondere wenn man verschiedene Such- und Auswerte-Operationen anschließt, wird die Programmarbeit kompliziert und aufwendig. Eigentlich müßte dann Maschinencodierung her, um das Programm schnell und speicherplatzsparend zu gestalten.

3. Programm MKART

Seine Leistungsfähigkeit

Nachdem wir Sie behutsam zu anspruchsvolleren Aufgaben bei ZX81 hingeführt haben, soll das zuletzt Gedachte nun in die Tat umgesetzt werden. Das Programm MKART versteht sich als Höhepunkt und Abschluß der Programmsammlung dieses Buches. Es ist nicht nur umfangreich, sondern auch anspruchsvoll programmiert; es arbeitet mit allen Tricks, die beim Sinclair derzeit möglich sind. Da primär *Lehr- und Lernprogramm* steht die praktische Nutzanwendung bei der Verwaltung von Mitgliedsdaten eines Sportvereins demgegenüber zurück. Wer MKART versteht, darf sich mit Fug und Recht als Profi bezeichnen. Das Ganze ist dermaßen ausgefuchst angelegt (hoffentlich bug free = ohne Programmfehler), daß bei 10 K Größe nur eine bescheidene Datenverwaltung möglich ist. Immerhin können durch extreme Speicherplatzausnutzung bis zu 100 Sätze, d.h. die Daten von bis 100 Mitgliedern, verwaltet werden. Bei ernsthafter Anwendung würden wir dringend zur RAM-Erweiterung raten oder aber gleich den Spectrum (mit 48 K) einsetzen. Wir hoffen, daß die zehn Seiten Listing dem kundigen, fortgeschrittenen ZXler zeigen, was man alles an Tricks und Tips verwenden kann. Trotz umfangreicher Dokumentation (siehe die folgenden Seiten) wurde auf Darlegung der Logik und insbesondere der Einzellogik von Programmteilen verzichtet. Der Grund ist einfach - das würde einfach zu weit führen. Im folgenden können nur Eigenschaften, Wirkungsweise und Grobaufbau wiedergegeben werden.

Wonach bestimmt sich die Leistungsfähigkeit eines Programms, was soll es und was kann es? Am Beispiel des Programms "Mitgliederkartei" sei gezeigt, wie man Programme beurteilen kann; die folgenden Maßstäbe sind als exemplarische Hilfe (sowohl für MKART wie für andere Programme) gedacht. Diese Eigenschaften bestimmen in etwa die Leistungsfähigkeit eines Programms:

a) speicherplatzsparsam

MKART geht vom Konzept der Datenverdichtung aus, um möglichst viele Mitgliedsdaten unterzubringen. Das geschieht durch zwei Maßnahmen

- Vercodung und Verschlüsselung
- Arbeiten mit variablen Daten, die hintereinander (in einem String) abgelegt werden

Halten wir uns vor Augen, welcher Speicherplatzbedarf entstehen würde, würden folgende Angaben in normaler Zahlendarstellung gespeichert werden:

- Um Name und Adresse in einem String hintereinander abzulegen, sind insgesamt vier Längenangaben erforderlich, sonst findet man die entsprechende Eintragung nicht mehr. Dazu kommt die Gesamtlänge des variabel langen Records; macht zusammen fünf Zahlen
- Kennzeichen für männlich/weiblich, das bei der Auswertung (nachher) eine Rolle spielt,
- Kennzeichen aktiv/passiv gleichermaßen,
- Statuskennzeichen, welches wiedergibt, ob es sich beim Mitglied um Neuaufnahme, Abgang oder Vollmitgliedschaft handelt,
- Dazu kommen die möglichen Sparteneingaben, d.h. es können maximal fünf Sportarten gleichzeitig gewählt sein,
- Fürs Geburtsdatum sind ebenfalls drei Zahlenangaben erforderlich (Jahr, Monat, Tag).

Speichert man alle Kennzeichen, Längen etc. als Zahl, so sind insgesamt 16 Zahlen erforderlich. Da jede Zahl "nackt" fünf Bytes kostet(soviel benötigt der Rechner für seine Floating-Point-Darstellung ohne Variablennamen), ergibt sich ein Speicherplatzbedarf von achtzig Bytes pro Mitglied. Bei einer durchschnittlichen Textmenge (für Name und Anschrift) von 60 Bytes würde jedes Mitglied 140 Bytes "kosten". Da bliebe bei 100 Mitgliedern und einer Normalausstattung des ZX81 mit 16 K RAM nicht viel für's Programm übirg.

Das Programm MKART benötigt für all die genannten Angaben nur ganze 10 Bytes. Das wird durch extreme Verschlüsselung und Vercodung erreicht. Zum einen werden die Kennzeichen (männlich/weiblich, aktiv/passiv, Neuaufnahme, Abgang..) ebenso wie die Sparte (=Sportart) in einen numerischen Schlüssel überführt und benötigt so nur ein Byte! Zum zweiten werden alle Längenangaben als Codes gespeichert. Was lediglich voraussetzt, daß kein Satz oder einzelnes Feld länger als 255 Bytes ist. Davon ist in der Regel auszugehen. - Aufgrund dieser Maßnahmen kommt MKART pro Satz mit 70 Bytes aus. Die Erfordernis "Speicherplatzsparen" ist also erfüllt, wenngleich ein umfangreicher, komplizierter Programmaufwand vonöten war.

b) modularisiert

Sowohl für Entwicklung wie für nachträgliche Änderung eines Programms ist das Modulkonzept tragender Stützpfeiler. Kleine Routinen oder Programmblöcke erleichtern nicht nur das Programmieren, sondern auch Übersicht und spätere Wartung. MKART verfügt über eine stattliche Menge Routinen, Module und größere Programmblöcke. Das meiste wurde an den Anfang gepackt, um für die maschineninterne Verwaltung kurze Zugriffszeiten zu ermöglichen. Andererseits konnte nicht alles separat programmiert werden, so daß sich eine Reihe Routinen überlappen. Integriert wurden zwei wesentliche - die Lese/Schreib-Routine und die Routine Positionieren. Integration war nicht Selbstzweck, sondern sollte mit einem Min-

destmaß Programmzeilen unterschiedliche Aufgaben bewältigen. Bei der Lese/Schreib-Routine bedeutet dies, daß sowohl der Inhalt bestimmter Felder gelesen wie auch geändert werden kann. Lediglich der Ansprung ist unterschiedlich. Bei der Positionierungs-Routine gilt ähnliches; hier kann der Mitglieds-Satz sowohl mit Suchbegriff Name wie mit Mitgliedsnummer (MNR) gefunden und positioniert werden. Da er sich im vollvariablen M$-Mitgliedsdatenstring befindet, liegt solches auf der Hand und wurde an anderer Stelle erläutert.

c) benutzerfreundlich

Die Benutzefreundlichkeit kommt bei vielen Programmlösungen, mögen sie noch so elegant sein, zu kurz. Dem Benutzer genügend Anzeigen und Bedienungsmöglichkeiten zu bieten, bedeutet viele Menues und häufiges Quittieren von Eingaben. In MKART wurde darauf großer Wert gelegt. In der Regel kann man mit New Line (NL) nicht nur Eingaben checken und ins übergeordnete Menue springen, sondern auch schrittweise lesen (z.B. bei variablen Namensdaten = Name und Anschrift).

Benutzerfreundlichkeit bedeutet aber auch, daß Ergebnisse ansprechend und verwendungsfähig sind. In MKART wurde beispielsweise bei der Besetzung der Spartenstrings eine Sortierung nach Mitgliedsnummer vorgenommen, so daß das Ergebnis hübsch "aufsteigend" auf dem Bildschirm erscheint. - Dabei lauert Gefahr, dem Benutzer "zuviel" an Information vorzusetzen. In MKART wurde zwecks schnellen Lesens der Mitglieder-Namensdaten das sogenannte ungezielte Lesen realisiert. Es bewirkt, daß das Mitglied nur mit Vornamen und Namen erscheint. Was eine schnelle Orientierung gestattet, ehe der Name als Suchbegriff zur Gewinnung weiterer Information verwandt wird. - Natürlich konnte nicht alles, was ein Benutzer sich wünschen mag, realisiert werden; beispielsweise unterblieben weitere Auswertungsmöglichkeiten, um das Programm nicht noch mehr aufzublähen, (z.B.: Jugendriege Fußball oder Handball Damen). Sogenannte Überkreuz-Selektionen mehrerer Merkmale dürften eher in den Bereich der Personalcomputer gehören.

d) flexibel

Mit MKART können zwar nur bestimmte Sportarten gewählt werden, aber es ist ohne weiteres möglich, auch anderes vorzugeben. (Die Sportart steht als "Fußball", "Handball" etc. in Texttabelle P$, welche mit der Initialisierung aus String I$ geladen wird). Wer andere Sportarten verwenden möchte, braucht lediglich im Programm (Zeile 103) die Sportart-Bezeichnungen zu ändern. - Flexibel ist ein Programm aber auch, wenn man die Zahl der zu verarbeitenden Elemente ändern kann. Dies ist grundsätzlich in MKART möglich, indem der Mitglieds-Nummernstring O$ (in Zeile 121) auf eine andere Zahl als 100 dimensioniert wird. Man bleibe jedoch unter der Höchstzahl von 100 Mitgliedern, sofern man nicht mit einer Speichererweiterung mehr Platz im Rechner hat... Zur Flexibilität gehört schließlich, daß ein Programm auf verschiedenen Rechnern läuft. MKART ist wie alle Programme dieses Buches so angelegt, daß es sowohl auf dem ZX81 wie dem Spectrum ablauffähig ist. Es wurde nur der gemeinsame BASIC-Wortschatz strikt verwandt.

Das nur zu den wichtigsten Merkmalen, die die Leistungsfähigkeit eines Programms ausmachen. Wenden wir uns nun weiteren Eigenschaften von MKART zu.

Features

MKART kann immerhin eine ganze Menge, wenngleich der Programmaufwand beträchtlich ist; man werfe nur einen Blick aufs Programmgerüst (siehe unten). Um die wesentlichsten Eigenschaften anzusprechen, seien sie stichwortartig wiedergegeben:

- Alle Daten sind der Art nach in zwei Blöcken organisiert -
  1. variable Mitgliedsdaten (Name und Anschrift),
  2. feste (Stamm-) Daten (Status, Sparte, Sportart).
  Flexible Eingabe und Änderung setzt allerdings entsprechende Längenfelder und exaktes Positionieren voraus, was dank der komfortablen Stringbehandlung des ZX81 elegant gelingt...

- Durch Abspeichern aller Namensdaten mit effektiver Länge (und deren Festhalten in getrennten Längenfeldern) wird ein Höchstmaß an Datenverdichtung erreicht. Auch die Codierung aller Zahlen (fester Mitgliedsdaten) gemäß Sinclair-Codetabelle bedeutet zwar Verschlüsselungsaufwand, jedoch höchste Speicherplatzausnutzung.

- Jede Positionierung eines Satzes erfolgt direkt mit Mitgliedsnummer und relativer Position im Mitgliederstring M$. Die entsprechende Stelle, d.h. als wievielter Record ein Mitgliedersatz steht, wird mittels Zwischenstring und der Variablen MST gewonnen (Einzelheiten zum Satzkonzept und Funktionsschema des Zugriffsverfahrens finden sich weiter hinten).

- Die Verwaltung ist dank flexibler Stringverarbeitung (durch Ändern, Verkürzen und Hinzufügen am bestehenden String) relativ einfach.

- "Stark" ist auch die Spartenauswertung. Die Ausgabe der Spartenbesetzung (d.h. wieviel und welche Mitglieder beispielsweise Fußball betreiben) erfolgt entweder durch Längenmessung des entsprechenden Spartenstrings (in dem Fall F$ = Fußball) oder durch Ausgabe der in ihm als Codes vertretenen Mitgliedsnummern.

Auf einem anderen Blatt steht der Programmaufwand. Das Mengengerüst sei kurz wiedergegeben.

```
Programmlänge:  ca. 10 K
Programmumfang: über 400 Zeilen
Module: 3 Hauptmodule, 6 Untermodule, zahlreiche Hilfsmodule
        /Routinen; insgesamt 26 Module / Routinen
Menues mit allen Optionen:  15
Sonstiges: 3 Auswertungsstatistiken, zweierlei komplette
           Mitgliederlisten (als Ausgabe)
```

Um Arbeitsweise und was das Programm bei der Mitgliederverwaltung eines Sportvereins bringt darzulegen, sei kurz auch dieser Teil abgehandelt (Näheres kann man dem Schaubild und weiteren Angaben entnehmen).

- Lesen:

Was im Mitgliederfile steht, kann sowohl gezielt wie ungezielt gelesen werden. Ungezielt bedeutet, daß kein Suchbegriff vorgegeben ist. Dann wird entweder eine kurze Namensausgabe (gemäß Speicherung) oder eine nach Mitgliedsnummern sortierte Liste gebracht. Zwischen beiden kann der Benutzer wählen; ein jederzeitiger Lesestop ist möglich. Beim gezielten Lesen muß vorher positioniert werden, d.h. entweder der Name oder die Mitgliedsnummer vorgegeben werden. Dann

erscheinen alle Namens- oder Stammdaten des Betreffenden auf dem Schirm.

- Fortschreiben:

  Fortschreiben bedeutet Eingabe eines neuen Mitglieds, Updating oder Löschen eines vorhandenen. Bei Neueingabe werden alle variablen und festen Mitgliederdaten abgefragt; jedoch braucht eine Mitgliedsnummer nicht vorgegeben werden, das System sucht sich andernfalls die nächste freie Nummer. Beim Updating ist wichtig, daß richtig positioniert wurde. Demzufolge legt das Menue nahe, bei jedem Ändern vorher zu positionieren, d.h. das entsprechende Mitglied mit Name oder Mitgliedsnummer herauszusuchen. Jedoch kann selektiv upgedatet werden, d.h. entweder im variablen (Name / Anschrift) oder im festen Teil (Status und Sparte). Beim Löschen besteht ebenfalls eine Wahlmöglichkeit: Möchte man das gesamte File weghaben, wird sicherheitshalber noch mal nachgefragt, ob das auch ok ist. Ansonsten besteht die Möglichkeit, einen Mitgliedersatz zu eleminieren, wobei alle bisherigen Spartenbesetzungen (d.h. Wahl der entsprechenden Sportart) ebenfalls mitgelöscht werden.

- Statistik:

  Bei den Stammdaten wird neben der Gesamtzahl Mitglieder auch die Verteilung auf einzelne Statusarten angezeigt. Als da sind die Rubriken männlich/weiblich, aktiv/passiv, Neuaufnahme, Vollmitglied, Abgang, Jugendlicher/Erwachsener. Der zweite Ast der Gesamtstatistik umfaßt die Spartenbesetzung. Nicht nur die Gesamtzahl Mitglieder pro Sportart (= Sparte) kann gewählt werden, es ist auch möglich, pro Sparte eine nach Mitgliedern sortierte Liste zu bekommen. Selbstverständlich kann man daneben auch einzelne Sparten "anwählen"; also z.B. "Ich möchte alle Mitglieder haben, die Handball betreiben."

Ehe im folgenden eine Reihe Lesehilfen zum Programm MKART kommen - quasi als Programmdokumentation -, seien noch zwei Anmerkungen gemacht:

1. In einer Reihe von Programmteilen wird als "Notausstieg" dem Benutzer angeboten, die Routine sozusagen fluchtartig zu verlassen. Dies sollte man nicht ohne schwerwiegenden Grund tun. Wird eine GOSUB-Routine "mit Gewalt" (d.h. durch GOTO) abgebrochen, besteht Gefahr, daß der Maschinen-Stack des Sinclair zu stark anwächst. Was heißt, der Adressstapel, wo der Sinclair die Rücksprungadressen ablegt, wird nicht wieder abgetragen. Irgendwann füllt er dann den gesamten Speicherbereich auf, so daß das Programm "überläuft".
2. Bei aller Raffinesse und aller trickreichen String- wie Codeverarbeitung, eins kann ein BASIC-Programm nicht überspielen - die natürliche Langsamkeit des ZX81. Man wappne sich bei umfangreicheren Daten und falls man sich die Mühe macht, alle Programmzeilen einzutippen, mit Geduld!

Lesehilfen für MKART

Der folgende Teil dient überwiegend dokumentarischen Zwecken. Eigentlich sollte er zu jedem Programm in der gebrachten Ausführlichkeit gehören. Nun kurz den Sinn der folgenden Seiten zur Erläuterung:

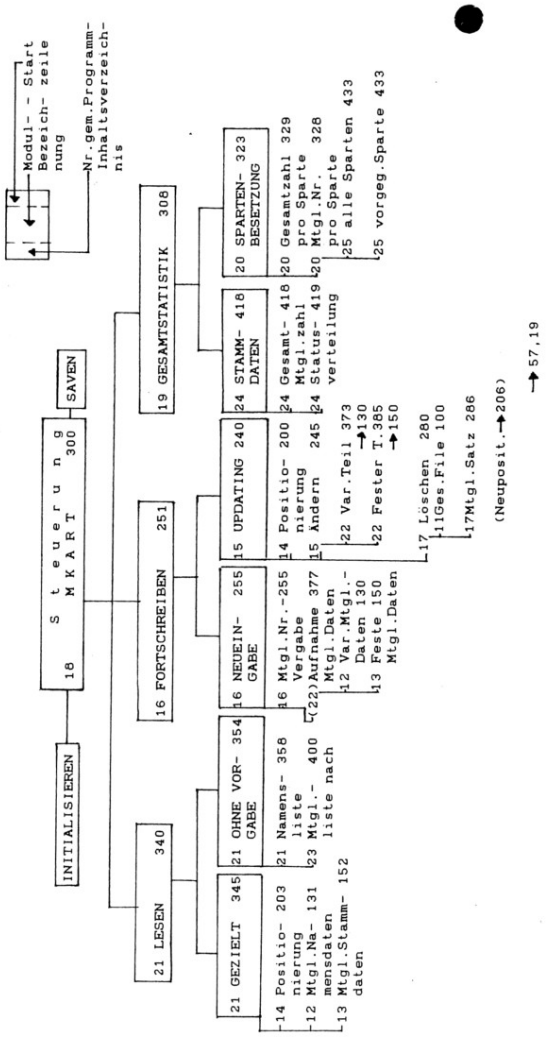
- Das *Programmgerüst* zeigt an, aus welchen Blöcken das Programm besteht, welche Nummer (im folgenden Verzeichnis) es hat und mit welcher Zeile es beginnt.

- Das *"Satzkonzept"* soll noch einmal in komprimierter Form Zusammenhang und Wirkungsweise der weiter oben beschriebenen Sinclair-Programmierung erläutern. Satzkonzept bedeutet, wie die Sätze aufgebaut sind, was unter File, Record (= Satz) und Feld zu verstehen ist und wie das Ganze bei entsprechender Positionierung funktioniert. Die spezifische Stringbehandlung mit allen Möglichkeiten, Daten aufzunehmen, zu ändern und zu löschen, ist an ein paar kleinen Schaubildern gezeigt.

- Wie einzelne Felder und Variablen benutzt werden, welchen Sinn sie haben und welche Daten sie aufnehmen, ist unter Punkt *Satzaufbau* und *Variablenliste* ebenfalls ausreichend dokumentiert.

- Wer über das Schaubild "Programmgerüst" hinaus tiefer einsteigen möchte, bekommt ein ausführliches *Programm-Inhaltsverzeichnis* geboten. Das gesamte Programm ist mit den wichtigsten Routinen/-Modulen in Stichworten wiedergegeben. Auch die Unterprogrammaufrufe und weitere Anmerkungen sollen das Lesen des eigentlichen Programms erleichtern.

- Da das Programm Daten im starken Maße verschlüsselt, findet sich für die beiden Felder "KZ" (Statuskennzeichen) und "S" (Sparte) der entsprechende *Verschlüsselungsmechanismus* auf einem eigenen Blatt. Codewert und dahinterstehende Informationen sind jeweils angegeben. Bei Sparte wird auch der dazugehörende Spartenstring genannt, wo jedes Mitglied, das diese Sportart gewählt hat, mit seiner Mitgliedsnummer (als Code) "angehängt" wird.



- Bleibt schließlich noch, wenn man nicht ins eigentliche Programm einsteigen möchte, eine Zusammenstellung der wichtigsten *Bedienungsregeln*. Diese enthalten das unbedingt Notwendige, um mit dem Programm zu arbeiten.

Die folgenden Seiten sind daher nicht nur als Dokumentation zum Programm gedacht, sondern auch, um das "Einsteigen" in MKART zu erleichtern. Für denjenigen, der sich mit allen Kniffen und Tricks des Sinclair beschäftigt, eröffnet sich so etwas wie eine Fundgrube nützlicher Routinen. Womit wir unserem Ziel, einen komplexen Anwendungsfall als Lern- und Lehrprogramm zu bringen, glauben gerecht geworden zu sein.

Satzkonzept

1. Zusammenhang der Sätze bei MKART
   . . . . . .

a) Begriffserklärung
   . . . . . .

File (entspricht im Deutschen etwa "Datei")
Sammlung logisch zusammengehörender Daten, auf die physikalisch (durch Positionieren) zugegriffen werden kann; das Mitgliedsfile M$ enthält alle variablen Namensdaten (Name und Anschrift) als "Sätze"

Record (deutsch "Satz")
Logische Untereinheit eines Files, d.h. Felder, welche insgesamt die Informationseinheit Satz ergeben; in MKART besteht ein Mitgliedssatz z.B. aus Name, Vorname, Stadt, Straße.

Felder (engl. items)
Variabel- oder gleichlange Zeichen-/Zahlenfolgen, die als logische Arbeitseinheiten verwandt werden (z.B. Name, Statuscode)

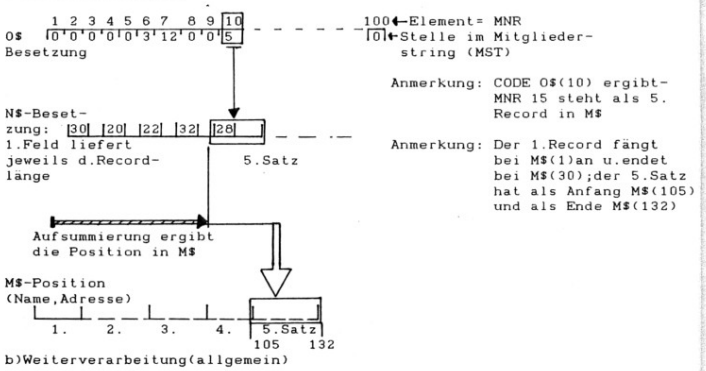
b) Definitionen
   . . . . . .

O$ = 100-stelliger String (-Array), der die Stelle (MST) der Mitgliedsnummer MNR im Feldlängenstring (N$) enthält.
N$ = Feldlängenstring; er hat je Mitgliedssatz 10 Felder, welche die Längen der variablen Namensdaten enthalten plus weiterer codierte Festdaten
M$ = Mitgliedsdatenfile; er enthält fortlaufend alle variablen Namensdaten, die als Records (= Sätze) angelegt sind.

2. Funktionsschema

(Beispiel - Mitglied Nr. 10 soll positioniert werden)

a)Grundsätz..Zugriff



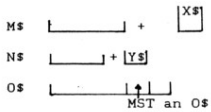
b)Weiterverarbeitung(allgemein)

M$-Position → POS (Vorkommawert = Anfang, Nachkommawert = Ende)
M$-Teilstring(=Record) → X$
N$-Teilstring(=feste Stammdaten) → Y$

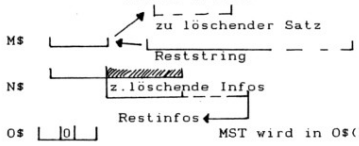
Einzelfeldlänge der variablen Namensdaten → N(5) so:
N(1)= Gesamtlänge (= Länge von X$)
N(2)= Länge Name
N(3)= "" Vorname
N(4)= "" Stadt
N(5)= "" Strasse

c)Einzelverarbeitung

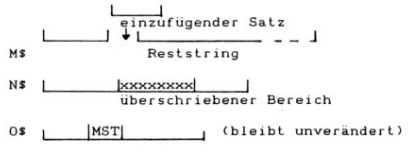
1.Neuaufnahme



2.Löschen



3.Updating

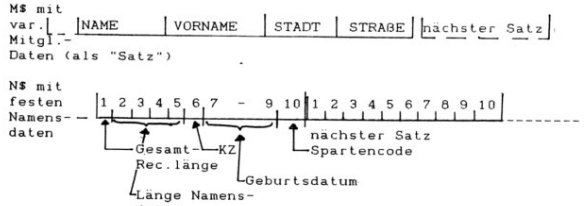


3. Satzaufbau

a) Klassifikation

Bezeichnung	Format/Stelle	Inhalt
1. Mitgliedersatz in M$	V(ariabel)	variable Mitgliedsdaten
. Name	V	Zuname
. Vorname	V	Vorname
. Stadt	V	PLZ und Stadt
. Straße	V	Straße und Nummer
2. Stammdaten in N$	F(est)	feste Stammdaten Mitglied (alle als Code)
Längen	1	Gesamtlänge Record (in M$)
	2	Länge Name
	3	Länge Vorname
	4	Länge Stadt
	5	Länge Straße
Statusinfos	6	KZ = Statuskennzeichen (s. u.)
	7 - 9	Geburtsdaten wie folgt:
	7	Jahr (letzte beiden Stellen)
	8	Monat
	9	Tag
	10	S = Sparte (s. u.)
3. Mtgl. Stellenstring O$	F(est) 1,...100	MST (als Code) =Stelle in N$, d.h. als wievielter Record die MNR gespeichert ist. Beispiel: Ist O$ (15) = 4, heißt das: die Infos von Mitglied Nr. 15 stehen in Record 4

b) Darstellung



Programm-Inhaltsverzeichnis (nach Zeilen aufsteigend)

Lfd. Nr.	Progr.-Zeilen	Routine/Module	Unterprog. aufruf (Start)	Anmerkungen
1	6 - 8	Programmstart	-> 10	Eingabe Tagesdatum und Sprung zur Hauptsteuerung
2	9 - 15	Datumseingabe (DATUM IN)		Datum in D$ mit Check
3	16 - 18	Datumsausgabe (DATUM OUT)		Decodierung D$
4	19 - 27	Löschen Sparten strings	-> 28 -> 40 54	MNR vorgegeben, Sparte in A$, Zwischenspeicherung in Z$
5	28 - 29	Zwischenspeichern Spartenstrings in Z$		Auswahl nach Sparte in B$
6	30 - 38	Anzeigeprint Sparten aus Texttabelle P$ (NEW S PRINT)		Printauswahl und -steuerung durch A$
7	39 - 55	Einzelsparten-rückbesetzung		wird selektiv angesprungen und aus Zwischenstring Z$ "geladen"
8	56 - 66	Decodieren Sparte		Spartenvariable S nach Zweierpotenzen entschlüsselt und Sparten in A$ laden
9	69 - 82	Codieren Sparte	-> 28 -> 40, 54	Verschlüsselung Spartenseingabe (B$) und Besetzen Spartenstrings mit MNR
10	84 - 90 91 - 95	Stringsort Elementaustausch	-> 92	Austauschsort der Zwischenspartenstrings Z$ nach MNR ... für Stringsort (84)

11	99 - 128	Wertinitialisierung		Löschen Variablen und alle Spartenstrings, Besetzen Texttabelle P$
12	129 - 149	Ein-/Ausgabe variabler Mitgliedsdaten (VAR MTGL.DATEN)		Anzeige GESCHLECHT, NAME, VORNAME, STADT, STRASSE, Führen Längentabelle N(5), Eingabe mit B$ und Zwischenspeicherung in X$
13	150 - 199	Ein-/Ausgabe feste Mitgliedsstammdaten	-> 10 17 19 30 70	Anzeige/ Veränderung von Statuskennzeichen KZ (MAENNL/WEIBL; AKTIV/PASSIV; NEUAUFNAHME, VOLLMITGLIED, ABGANG), Geb.-Datum und Sparte (= Sportart) in S
14	200 - 239	Positionierung Mtgl.string M$ (POSITIONIERUNG) nach Mitgliedsnummer oder Namen (NAMENSSUCHE)		Suche mit MNR (203-223) oder mit NAME (225-235), Besetzen MNR und Mitgliedssatz MST, Positionsspeicherung in POS und Zwischenspeicherung variabler Daten (in X$) + fester Mtgl.Daten (in Y$) aus M$ bzw. N$
15	240 - 250	Modul UPDATING	-> 200 370	Ändern mit/ohne Positionierung; Steuerung f. Updaten, Löschen und Untermenues
16	251 - 275	Hauptmodul FORTSCHREIBEN mit eingebundener NEUAUFNAHME	-> 377	Steuerung f. Neueingabe + Updating, bei Neueingabe: Mtgl.Nr.-Vergabe, Positionieren + Füllen mit allen Mtgl.daten (Unterprogramme 130 +
17	279 - 299	Modul LOESCHEN (ohne RETURN, da Hauptsteuerung direkt folgt	-> 100 206 57	File- u. Mtgl.Satz-Löschen; bei letzterem mit Mtgl.Namensandruck, Sparten-Stringlöschen und Neupositionieren von M$ und N$
18	300 - 307	HAUPTPROGRAMM-Steuerung	-> 100	Berechneter Sprung zu den Hauptmodulen LESEN, FORTSCHREIBEN, STATISTIK usw.
19	308 - 321	Steuerung GESAMT-STATISTIK	-> 17 10 -> 418 323	Datumsein-/ausgabe und Verzweigung zu Mtgl. Stammdaten /Spartenbesetzung
20	322 - 339	Modul SPARTEN-STATISTIK (nach Anzahl u. Besetzung	-> 30 28 -> 433	Spartensteuerung aus A$, Anlistung Summen als Länge Zwischenstring Z$; für Ausgabe Mtgl.Nummern (sortiert) Aufruf Unterprogramm (433)

21	340 - 368	Hauptmodul LESEN un-/gezielt mit	-> 203 395 391 131 152 -> 400	Hauptaste: Lesen mit Vorgabe -> (345) = Positionieren u. Ausgabe Stamm-/var. Namensdaten . Lesen ohne Vorgabe ->(354) = Namensausgabe unsortiert oder Print Mtgl. Nr.-Liste (400)
22	369 - 397	Modul MTGL.DATEN-AENDERUNG	-> 391 130 -> 395 150	Nach Satzpositionieren-Lesen /Ändern var. Namensdaten (Name, Adresse) und/oder - Lesen/Ändern feste Stammdat. (Mtgl. Status, Spartenwahl)
23	399 - 416	Erstellen Mitgliederliste (nach MNR)	-> 218 - 391	Abfragen Codestring O$ auf Mtgl.Nr., Positionieren und Zwischenspeichern Namensdat. Besetzen Längentabelle, Ausdruck Name/Anschrift aus X$ gem. Länge N (N)
24	417 - 432	Ausdruck Sparten-statistik		Durchprüfen Stammdatenstring N$, Besetzen der Vari. f. Geschlecht und Mtgl.Status, Ausdruck mit Texttabelle P$
25	433 - 440	Ausdruck Sparte mit Mitglieds-Nr. (sortiert)	-> 85	Sortierung der zwischengespeicherten Spartenstrings Z$, Angabe Sparte aus P$ und der Mtgl.Nr. (decodiert)

Variablenliste

1. Arrays

DIM P$ (20, 10)		Nachrichtentabelle
DIM N (5)		Längentabelle für Namensdaten (zur Zwischenspeicherung)

2. Strings

Hauptstrings:	M$ =	Mitgliederfile (variable Namensdaten)
	N$ =	Feste Stammdaten Mitglieder und Satzlängen
Hilfsstrings:	X$ =	Zwischenspeicher f. variable Namensdaten (hauptsächlich)
	Y$ =	Zwischenspeicher f.feste Mtgl.Daten (hauptsächlich)
	Z$ =	Zwischenspeicher f. Spartenstrings u.a.
	K$...G$ =	Spartenstrings (s. Verschlüsselung)
	C$ =	Hilfsfeld f. Stringsort
	I$ + B$ =	Benutzereingaben
	DS =	Datum
	A$ =	Hilfsstring f. Spartenbesetzung, Nachrichtenprint und die Leseroutine

3. numerische Variablen

S =	Sparte
KZ =	Statuskennzeichen Mitglied
MNR =	Mitgliedsnummer
MST =	Record Nr. ("Stelle" Mitglied in M$-File)
POS =	Positionierung Record in M$
A =	Zwischenspeicher f. Benutzeroption beim Updating u.ä.
N,M =	Allgemeine Laufvariablen

Verschlüsselungen

KZ = Statuskennzeichen

Codewert	Information (Besetzung gem. Legende)							
	M	W	A	P	N	V	A	
1	X			X	X			
2	X			X		X		
3	X			X			X	
4	X		X		X			
5	X		X			X		
6	X		X				X	
11		X		X	X			
12		X		X		X		
13		X		X			X	
14		X	X		X			
15		X	X			X		
16		X	X				X	

Legende:

M = Männlich
W = Weiblich
A = Aktiv
P = Passiv
N = Neuaufnahme
V = Vollmitglied
A = Abgang

S = Spartencode

Sparte (=Sportart)	zugehöriges Anzeigen-Tabellenelement P$ (N)	zugehöriger Spartenstring	Codierung **	Spartenwert (nur einf. Wahlen) *
keine Sparte	-	K$	-	0
Fußball	1	F$	2^0	1
Handball	2	H$	2^1	2
Volleyball	3	V$	2^2	4
Tennis	4	T$	2^3	8
Schwimmen	5	S$	2^4	16
L. Athletik	6	L$	2^5	32
Gymnastik	7	G$	2^6	64

* Es können daher mehrere Sportarten zugleich vorkommen bzw. gewählt

werden - Beispiel: S = 5 heißt "Fußball" und "Volleyball" (als Sparte)

** Decodierung erfolgt durch CODE - 2^n und Weiterrechnen mit dem Rest.

Bedienungsregeln

1. Programmstart ist immer mit GOTO 1 anzugehen - es sei denn, man möchte alle bisherigen Daten verlieren! (Dann muß initialisiert werden, da das Programm bestimmte Anfangszustände benötigt)

2. Man halte sich bitte (zunächst) an die Vorgaben des Menues und hüte sich davor, "logischen Unsinn" zu versuchen (z.B. bei Neueingabe nicht vorhandene Daten zu lesen oder beim Updaten nichts einzugeben). Ist einmal "Datenschrott" im Mitgliederfile, hilft nur noch Satzlöschen und Neueingabe.

3. Da das Programm recht aufwendig und komfortabel ist, richte man sich auf eine gewisse Such- und Antwortzeit ein.

4. Das Mitgliederfile ist auf einen Nummernkreis von 1 - 100 (=Mitgliedsnummern) angelegt. Möchte man anders - eher kleiner als größer - dimensionieren, braucht lediglich Zeile 121 geändert zu werden.

5. Bei verunglücktem Updating kann man ohne weiteres wieder "von vorne" einsteigen ; da das Programm mit kombinierter Schreib/Lese-Routine arbeitet, ist leicht zu sehen, wie weit Daten eingegangen sind. Auch die Spartenbesetzung kann zusätzlich (mit Untermenue STATISTIK) gecheckt werden.

6. Beim Lesen vorhandener Daten unterlasse man tunlichst - trotz gelegentlicher Aufforderung (gleiche Programmroutinen!) - den Versuch, Neues einzugeben. Dafür ist Menue Updating da, es stellt hierzu den richtigen Einstieg dar.

7. Da viel mit der NL-Taste gearbeitet wird, bekommt man rasch heraus, wann Daten zunächst nicht angezeigt werden und wann sie mit NL erscheinen. Das hat den Sinn, a) den Bildschirm in Ruhe zu studieren, ehe ein neues Menue erscheint und b) gültige Daten einfach zu "überlesen", falls Änderung nicht gewünscht wird. NL hat auch eine Quittierungsfunktion, d.h. solange nach einem Benutzerinput kein NL folgt, kann nochmal etwas anderes eingegeben werden.

8. Besondere Vorsicht ist bei Status- und Sparteneingabe angebracht. Statusprints (Geschlecht, Neuaufnahme etc.) sollten entweder quittiert und bei ok "weitergeblättert" oder gleich mit der gewünschten Eingabe korrigiert werden. (Bei "Vollmitglied, Abgang, Neuaufnahme" wird solange rotiert, bis sich der Benutzer entschieden hat). Bei Spartenkorrektur gilt, es muß nach Ausgabe Sportart eine Spartenwahl stattfinden (was eigentlich selbstverständlich ist).

9. Zwar ist ein Ändern im Anschluß an die letzte Positionierung möglich, jedoch gilt umgekehrt - gelesen werden kann nur, was vorher im File positioniert wurde. In jedem Falle muß auf eine aktuelle, gültige Positionierung geachtet werden.

10. In allen Menues ist ein Notausstieg vorgesehen - in der Regel mit NL (New Line); man vermeide gewaltsamen Arbeitsabbruch. Bei eventuellem Systemstop kommt man mit GOTO 300 immer zur Hauptsteuerung zurück.

KAPITEL VI:

TIPS & TRICKS

1. Eingaben und Ausgaben

Speicher contra Laufzeit

Im Zusammenhang mit dem ZX81 gibt es eine Unzahl von Tips & Tricks. Das beginnt bei Ratschlägen zu Entwurf und Programmaufbau und reicht bis zur Maschinencodierung. Eigentlich wäre das Ausloten jeden Sinclairbefehls, seiner vielfältigen Möglichkeiten und Grenzen ein weiteres Buch wert (Ein solches erscheint im gleichen Verlag ebenfalls aus der Feder des Herausgebers).

Was heißt nun eigentlich TIPS & TRICKS?
Das bedeutet doch wohl im wesentlichen zweierlei:

a) Unbekannte Features des Rechners entdecken und ausnutzen; mit anderen Worten andere Dinge als im Manual oder in mehr oder weniger naiv-unbedarften Werken für Programmeleven enthalten sind. Meist werden Tips in User Clubs unter der Hand gehandelt: "Wußten Sie schon, daß man..."

b) Vereinfachungen, die umständliche Programmierung vermeiden und das Gleiche eleganter ausdrücken. - Das bedeutet im wesentlichen, Programmroutinen schneller zu machen oder den Speicheraufwand ökonomischer zu gestalten.

Bei dem Bemühen, durch Kniffe den eigenen oder den Aufwand des Rechners klein zuhalten, befindet man sich jedoch in einem Dilemma. Zwischen folgenden drei Punkten besteht nämlich eine Diskrepanz:

- Eleganz einer Lösung; wobei nicht eine gute Idee, in die man sich leicht verliebt, gemeint ist, sondern ein gewünschtes Ergebnis mit möglichst kurzer Befehlsfolge zu realisieren. Aber auch da muß man Obacht walten lassen; meist trügt die Optik. Ein kurzer Programmbefehl braucht nicht dem Rechner im Innern Entlastung zu verschaffen...

- Speicherplatz-sparsame Instruktionen; weniger Speicher bedeutet fast immer erhöhten Programmaufwand. Insbesondere im Zusammenhang mit der speziellen Vergleichslogik des Sinclair kann man zwar etliche Bytes sparen, aber mit tödlicher Sicherheit sinkt die Verarbeitungsgeschwindigkeit rasant. Womit wir beim nächsten Punkt wären...

- Laufverhalten; die Ablaufgeschwindigkeit ist die Kehrseite der Medaille "Speicherplatz sparen". Günstigeren Durchsatz von Programmbefehlen kann nur erreichen, wer - so die Regel - erhöhten Befehlsaufwand betreibt.

Eine elegante Lösung, speicherplatzarm und mit wenig Rechnerverarbeitung, ist eher die Ausnahme als die Regel. Man kann allerdings unter "Eleganz" beides verstehen - schnelle und ökonomische Programmierung. - Da die meisten Hobbycomputeristen ungern viele Tasten drücken und aus Passion "faul" sind, soll natürlich ein und derselbe Programmeffekt mit möglichst wenig Aufwand erreicht werden. Das ist durchaus verständlich. Aber sogenannte elegante Lösungen ecken meist mit den anderen Größen-Laufzeit und Speicherplatz - an.

Als allgemeine Regel, ehe wir die Rubrik "TIPS & TRICKS" weiter ab-

handeln, gilt daher zweierlei: Vermeiden Sie *Zahlen*, unnützen *Ballast* und als ersten Schritt - konstruieren Sie eine *saubere Logik*! Das sogenannte Tuning (etwa: Schnellermachen des Programms) sollte erst im zweiten Akt folgen. Wurde alles durchgecheckt und die zu erledigenden Dinge durchdacht, kann man sozusagen als "finish" darangehen, mit ein paar Kniffen das Ganze zu optimieren. TIPS & TRICKS sind allgemein als Beigabe, Erfahrungprodukt und Abschlußmaßnahme gedacht. Und nicht als Selbstzweck...

Gehen wir über zur ersten Rubrik, wie sie die Überschrift ankündigt: Ein- und Ausgaben.

INPUT oder INKEY$?

Generell kann die Frage nicht eindeutig beantwortet werden. Vieles ist von Reaktionsgeschwindigkeit und Datensicherheit abhängig. Generell gilt, das INKEY$ für schnelle Tastaturabfragen geeignet ist. Da die Funktion eine rasante Keyboard-ROM-Routine anstößt, muß man Vorsorge treffen, daß buchstäblich der Rechner nicht davonläuft. Das heißt, es sind zusätzliche Programmzeilen einzufügen (worauf wir noch näher eingehen werden). Andererseits hat INKEY $ - neben seiner Schnelligkeit - zwei unbestreitbare Vorzüge:

a) Es braucht nicht nach jeder Eingbe "ENTER" bzw. "New line" gedrückt zu werden und

b) man kann INKEY$ direkt, ohne ein Feld/eine Variable zu bemühen, abfragen.

INPUT hingegen hat als Vorteil, daß sowohl nach Zeichen wie nach Zahl die entsprechende Variable gewählt werden kann. Und noch etwas kommt hinzu, was bei einzugebenden Texten im Rahmen einer Textverarbeitung nicht ohne Bedeutung ist - man kann die Eingabe, ehe sie mit "NL" abgeschickt wird, am unteren Bildschirmrand editieren!

Wenden wir uns nun ein paar Einzelheiten zu, die vielleicht nicht jedem Hobbyprogrammierer des ZX81 bekannt sind. Für die übliche JA/-NEIN-Abfrage braucht man keineswegs immer ein Character-Feld zu bemühen. Vielmehr läßt die INPUT-Funktion auch Eingabe einer Variablen direkt zu. Wurden beispielsweise die Worte "JA" und "NEIN" (als Variable) vorab definiert, können sie als Buchstabenfolge mit "INPUT" vom Benutzer eingegeben werden, obwohl das Programm nach der numerischen Variablen (A) fragt. Probieren Sie am besten die folgenden Zeilen aus:

```
LET JA = 0
LET NEIN = 1
PRINT "ANDRUCK? JA/NEIN"
INPUT A
IF A = JA THEN PRINT "JA"
IF A = NEIN THEN PRINT "NEIN"
```

Die Abfrage von Zahlenvariablen ist ohne weitere Checks nicht ungefährlich. In der Regel wird man daher mit einer Stringeingabe arbeiten. In diesem Zusammenhang hilft die segensreiche VAL-Klausel, die bekanntlich "Stringzahlen" in echte, vom Rechner zu verarbeitende Zahlen umgewandelt. Also etwa so:

```
INPUT I$
 :
LET X = VAL I$
```


Aber mit Verwendung einer Stringvariablen hat man sich weitere Risiken eingehandelt: Was ist, wenn ein Leerzeichen eingegeben wird, d.h. der Benutzer einfach "NL" druckt? Sofern man mit dem zulässigen Wert "0" arbeiten mochte, kann man eine Besonderheit des ZX81 ausnutzen: Wird eine Leerkette eingegeben, hat diese den internen Code 0. Man fangt also den NL-Fall ab, indem man weitere Bearbeitungen auf numerische Werte in Form der CODE's abstellt. - In diesem Zusammenhang kann man eine weitere Besonderheit des Sinclair-BASIC ausnutzen: Unabhängig, wie lang eine eingegebene Kette ist, man bekommt immer das erste Zeichen der Kette mit folgender, zunächst unsinnig erscheinender Klausel:

```
LET A$ = CHR$ CODE I$
```

Worauf beruht der genannte Effekt? Zunächst erscheint es abwegig, den CODE eines Zeichens abzufragen, um ihn ins Ursprungszeichen zurückzutransformieren. Aber die Eigenheit des Sinclair ist eben, mit dieser Klausel das erste Zeichen einer Kette zu nehmen, so daß man sich nicht mit umständlicher Qualifizierung (= Positionsbestimmung innerhalb der Kette) abplagen muß. Wieso das praktischen Nährwert hat, wird am Beispiel deutlich: Um die "Andruck-Frage" fortzuführen, als Antwort soll statt JA und NEIN auch J oder N (oder weitere mit J oder N gebildete Verbindungen)statthaft sein. Der Benutzer kann statt JA einfach J eingeben und statt NEIN nur N oder NICHT, NO und ähnliches. Wie verkraftet eine Routine diese Aufgabenstellung? Schauen wir uns die folgende, noch etwas umständliche Lösung an:

```
PRINT "ANDRUCK ? JA/NEIN"
INPUT I$
IF I$ (1) = "J" THEN PRINT "JA"
IF I$ (1) = "N" THEN PRINT "NEIN"
```

Nun die Verbesserung mit dem Character-Code, wobei zudem eine spezielle Drucklogik mit AND angewandt wird:

```
PRINT "JA" AND CHR$ CODE IS = "J";
      "NEIN" AND CHR$ CODE I$ = "N"
```

Das schnelle INKEY$ bietet sich demgegenüber bei simulierter CURSOR-Steuerung an, wie von vielen Spielprogrammen her hinreichend bekannt. Aber auch bei "ernsthafter" CURSOR-Steuerung in Textverarbeitungsprogrammen (siehe Programm WORD) und bei vielen PRINT-/PLOT-Routinen ist sinnvollerweise INKEY$ zu gebrauchen. Allerdings muß man zwei Vorsorgemaßnahmen treffen, damit einem die blitzschnelle Tastaturabfrage nicht davonläuft:

a) Entweder wird die Abfrage durch eine Selbstblockade, wie sie im Manual Seite 128 geschildert wird, gezügelt, oder

b) man setzt die Pausenfunktionen ein, um zu "greifbaren" Werten zu kommen.

Die folgende kleine Routine arbeitet mit INKEY$ und malt je nach ZX81-CURSOR-Pfeil ein schwarzes Viereck auf den Schirm. Sehen wir uns die folgenden Zeilen näher an, die insbesondere in Verbindung mit der SINCLAIR-spezifischen Vergleichslogik die Werte-Steuerung veranschaulichen.

```
90  LET ZL  = 10        )  Zentrieren Ausgangs-
95  LET  SP  = 15       )  punkt (Zeile  10,Spalte 15)
```

```
100  IF INKEY$ <> ""
        THEN GOTO 100      )  Tastatur-
110  IF INKEY$ = ""         )  blockade
        THEN GOTO 110

120  LET A$ = INKEY$
130  IF A$ <"5" OR A$ > "8"          Ausstieg
     THEN GOTO 220

140  LET SP = SP - (A$ = "5") + (A$ = "8")   ) Spaltenwert
150  IF SP <0 THEN LET SP = 0                ) u.Überlauf-
160  IF SP >31 THEN LET SP = 31              ) sicherung

170  LET ZL = ZL - (A$ = "7") + (A$ = "6")   ) Zeilenwert
180  IF ZL > 21 THEN LET ZL = 21             ) u.Überlauf-
190  IF ZL <0 THEN LET ZL = 0                ) sicherung

200  PRINT AT ZL, SP; " "                    ) PRINT und
210  GOTO 110                                ) Rücksprung
```

Und noch ein Trick, mit dem man dem Benutzer die Möglichkeit offeriert, durch Eingabe von NL das Programm nach einer beliebig langen Pause fortzusetzen. Man spricht in diesem Zusammenhang von der ewigen Pause, engl. forever. 4E4 klingt so ähnlich. Mit anderen Worten, eine ewige Pause erreicht man durch :PAUSE 4E4. Solange keine Taste gedrückt wird, hält der Computer an; jeder weitere Tastendruck einschließlich NL läßt das Programm mit der folgenden Programmzeile weiterlaufen. Bekanntlich bringt eine Pausenzahl von >= 32767 den ZX81 zum Stehen. Aber statt diese lange Zahl einzugeben, hat die geschilderte Klausel (welche sich einer besonderen Zahlendarstellung bedient) drei Vorteile, die ein Musterbeispiel für elegante Programmierlösung darstellen:

a) weniger Speicher als die volle Zahl 32767

b) weniger "Tipparbeit" (nur 3 Tastenhübe),

c) leichter zu merken, da "four E four" so ähnlich klingt wie "forever"...

Verlassen wir den Bereich der Eingaben und wenden uns der Ausgabe und insbesondere dem Komplex "Print-Steuerung" zu.

PRINT-Steuerung

Zunächst ein Negativ-Beispiel, welches Anzeige und Eingabe in ziemlich aufwendiger Form veranschaulicht:

```
100  PRINT "MENU"
110  PRINT
120  PRINT "---"
130  PRINT
140  PRINT "1-NEUEINGABE"
150  PRINT "2-LESEN BEST."
160  PRINT "3-AENDERN"
 :
190  INPUT  I
200  IF I < 1 OR I > 3
     THEN GOTO 190
210  IF I = 1 THEN GOTO 300
220  IF I = 2 THEN GOTO 600
```



```
230  IF I = 3 THEN GOTO 900
```

Eine Verbesserung der Print-Statements kann durch verschiedene Maßnahmen erreicht werden. Im wesentlichen sind folgende Forderungen zu erfüllen:

a) weniger Lines of code,
b) damit im Zusammenhang weniger Abfragen zur Positionierung und
c) kein sich einschleichender Mehraufwand.

Das oben gezeigte Beispiel (bzw. der Ausschnitt aus einer Eröffnungs-Routine) braucht für die beschriebenen Prüfungen und Anzeigen zwölf Zeilen. Wie kommt man mit weniger aus? Zunächst läßt sich die Printsteuerung vereinfachen, indem man mit Kommata arbeitet. Dadurch nutzt man die Aufteilung des Bildschirms in zwei Print-Anfangspositionen (1. und 16. Spalte) aus. - Viele IF-Abfragen lassen sich durch die spezielle SINCLAIR-Logik abfangen. Mit ihr prüft man zum einen I auf vorgegebene Grenzwerte ab, zum zweiten übergibt man ihr die GOTO-Steuerung. Da ein Sprungziel auch ein arithmetischer Ausdruck sein kann (im vorliegenden Beispiel die Multiplikation von I mit 100), wird mit der spezifischen ZX81-Wahrheitslogik der gesamte Abfrageteil auf eine einzige Zeile reduziert. Die Printsteuerung kann ebenfalls auf eine Zeile (mit zahlreichen Kommata) zusammengedrückt werden. All das geschieht, ohne daß mehr Zeichen verbraucht werden oder die Laufzeit spürbar leidet. Die optimierte 3-Zeilen-Lösung sieht so aus:

```
180 PRINT "MENU",,,,"----",,,,"1-NEUEINGAB",,
    "2-LESEN BEST",,"3-AENDERN"

200 INPUT I

210 GOTO 200 * (I<1 OR I> 3) + I * 300 * (I>0 AND I<4)
```

Man hätte natürlich auch einen anderen Weg gehen können, insbesondere wenn der Anzeigetext in gleicher Form mehrfach im Programm auftaucht. Beispielsweise könnte man ihn in einem String ablegen, der jeweils mit unterschiedlicher Positionierung selektiert und ausgedruckt wird. Aber nach unserer Erfahrung ist die vorgestellte 3-Zeilen-Lösung die bei weitem kürzeste und für den Normalfall ausreichenste Lösung.Ökonomisch sind in diesem Zusammenhang auch die vorgegebenen Sinclair-Tokens (=Befehlscodes). Da sie Worte wir STOP, NEW, INPUT, IF, NEXT etc. enthalten, braucht man diese Begriffe nicht mehr Byte für Byte einzugeben; vielmehr greift man direkt auf den entsprechenden Code (siehe Code-Tabelle im Anhang des Manuals) zurück. Dabei gibt es zwei Wege:

a) Man gibt die Befehlscodes im EDIT-Modus direkt ein, indem man einen kleinen Trick anwendet: Um in den Command-Modus (nur in diesem sind die Tokens anlistbar) zu kommen, drückt man die Taste "THEN", wählt das Token aus, positioniert anschließend zurück und radiert "THEN" wieder aus.

b) Man fügt die entsprechenden Characters direkt ins Print Statement ein (CHR$-Klausel!).

Weitere Ausgabetricks

Damit ist die Printsteuerung bei weitem nicht erschöpft. Die zentrale Funktion der SINCLAIR-spezifischen Abfragelogik kann nutzbringend mit ins Spiel gebracht werden. Um nur zwei Hauptvarianten vorzustellen:

1. Vorschubsteuerung und
2. selektives Drucken von Nachrichten

Zu 1.: Neben dem üblichen Kommasetzen gibt es eine andere Möglichkeit der Vorschubsteuerung (vergleiche Programm SCPD, Zeile 182). - Zunächst das Problem: Wenn im Textstring ein bestimmtes Zeichen (z.B. "*") vorkommt, soll ein Absatz gemacht werden. Ansonsten werden Zeichen fortlaufend in die Zeile "geprintet". - Lösung: Der Vorschub wird durch NL erreicht, etwa als direktes Printcommand (mit CHR$ 118) und durch ein AND-Abfrage (auf bezeichnetes Sternchen). Bekanntlich funktioniert die Vergleichslogik auch bei Zeichenketten, jedoch nur für die AND-Klausel.

Die Lösung, den Textstring T$ auszudrucken und bei "*" Vorschub zu machen, geschieht im Programm durch folgende Programmzeile:

```
PRINT CHR$ 118 AND T$ (N) = "*";
      T$ (N) AND T$ (N) <> "*";
```

Soll nun jedoch die Länge einer Kette (als Repräsentant einer Elementzahl) abgedruckt werden, versagt die vorgestellte Printsteuerung mit AND. Um einen Trick aus dem Programm MKART aufzugreifen - wie bekommt man den numerischen Wert LEN X$ für den Zeichen-Vergleich in den Griff?

Bei Einsatz der LEN-Funktion würde bei negativen Vergleichsergebnissen eine 0 erscheinen, bei positivem Vergleich eine 1. Es interessiert aber im Zusammenhang mit der Printsteuerung nur der richtig positionierte, echte Längenwert. Die Lösung ist relativ einfach - man transformiert die (numerische) Längenangabe in eine Stringgröße. Dazu dient die Funktion STR$. Ist nun der Vergleichswert 0, wird das Printzeichen zu einem Leerzeichen, bei Vergleichswert 1 erscheint hingegen die gewünschte Längenangabe ohne Zusatz. Was die folgende Programmzeile zeigt; ihr liegt die Annahme zugrunde, wenn N = 1 wird die Länge von X$ ausgedruckt, ist N = 2, erscheint die Länge von Y$. Und nun der Befehl in seiner vollen Komplexheit:

```
PRINT STR$ LEN X$ AND N = 1;
      STR$ LEN Y$ AND N = 2
```

Zu 2.: Das selektive Drucken von Nachrichten ist immer dann nützlich, wenn es darum geht, Fehlertexte auszugeben (vergleiche LOTTO 3, Zeile 668). - In Verbindung mit AND kann man etwa zur Lösung einen einfachen OK-Schalter einsetzen. Mit anderen Worten, die Variable OK bringt die Information, ob alles in Ordnung war (OK = 1). Ist OK = 0, lag ein Fehler vor. Bekanntlich kann das Abprüfen auf 0 mit der Klausel NOT OK realisiert werden, beispielsweise wenn eine Tabelle leer ist. Das sieht dann einfach so aus:

```
PRINT "TABELLE LEER" AND NOT OK
```

Man braucht aber keineswegs erst einen Schalter einzubauen, sondern kann direkt beim entsprechenden Vergleich das Vergleichsergebnis im Klartext ausdrucken:


```
PRINT "ELEMENT "; ("NICHT" AND X <> TAB (N);
      " GEFUNDEN"
```

Selbstverständlich sind auch Kombinationen zulässig, indem man mehrere Abfragen "schachtelt" (vergleiche Programm SCPD, Zeile 132). Die folgende Programmzeile etwa verlangt eine Eingabe, wenn vorher die String-Variable I$ den Wert "B" oder "S" erhalten hatte:

```
PRINT "EINGABE" AND (I$ = "B" OR I$ = "S")
```

Es ließen sich noch jede Menge weiterer TIPS & TRICKS zum Problemkreis "Eingabe/Ausgabe" bringen. Um den Hobbyprogrammierer, insbesondere den noch nicht gewieften, nicht zu verwirren, sei dieser Punkt hier abgeschlossen. Wenden wir uns weiteren Kniffen zu, die insbesondere auf die zwei Hauptstärken des ZX81 abstellen - die vielfältigen Varianten der Vergleichs-/Abfrage-Logik und die herausragenden Möglichkeiten bei der Stringbehandlung.

2. Stringbehandlung

SINCLAIR's Stärke

Die Hauptstärke des ZX81 liegt zweifellos auf dem Gebiet der Zeichenketten. Nicht nur Bildung und Aufteilung ermöglicht eine Flexibilität, die man bei anderen Rechnern suchen muß (Man beachte die einzelnen Features des Programms WORD). Darüber hinaus können Zeichenketten - neben bequemer Aufteilung, Verkürzung und Verlängerung - dazu verwandt werden, aufwendige Zahlendarstellung zu vermeiden. Bekanntlich kostet jede Zahl unabhängig vom Namen mindestens 5 Bytes, sobald per Programm installiert. Gelingt es, eine Zahl in einem Character unterzubringen, braucht man nur ein einziges Byte. Insbesondere, wenn es darum geht, Speicherplatz zu sparen, dürfte das Thema "Zahlen im String" in den Mittelpunkt der Betrachtung rücken. Allerdings muß man sich darüber im klaren sein, daß jede Konvertierung (insbesondere auf CODE-Ebene) Aufwand bedeutet; nicht nur beim Schreiben im Programm, sondern erst recht im Rechner. Die beiden "Optimalziele" Speicherplatzsparen und Laufzeitbeschleunigung stehen im Gegensatz zueinander. Beides zugleich kann nur in seltenen Fällen erreicht werden. Schauen wir uns zunächst ein paar Besonderheiten der Sinclair-Strings an, ehe wir den Schwerpunkt auf das Thema "Zahlen in Strings" legen.

Worauf man zunächst bei der Stringbehandlung achten muß, sind gleiche Längen. Besonders beim Vergleichen läßt Prokrustes grüßen...Bei ungleichen Längen geht der ZX81 rigoros mit dem "Material" um: Was nicht paßt, wird abgeschnitten (Vergleichsargument zu lang) oder mit Spaces in die Länge gezogen (Vergleichsargument zu kurz). Möchte man Vergleiche anstellen, ohne die Länge der beiden Zeichenketten zu kennen und indem man auf Richtigkeit der ersten Charactere abprüft, kommt man mit der folgenden Programmzeile nicht weit:

```
IF I$ = X$ THEN LET OK = 1
```

Hier hilft die nützliche Längenklausel "LEN". Möchte man zudem die bekannten Unsinnsvergleiche "" und " " vermeiden, wird eine einfache Programmroutine in etwa so aussehen:

```
LET X$  =  "GG"
LET OK  =  0
```

```
INPUT I$
IF I$ = " " OR I$ = ""
   THEN GOTO...
IF I$ (TO LEN X$) = X$
   THEN LET OK = 1
```

Vorsicht ist auch bei der Dimensionierung von Strings angebracht. Es ist etwas gänzlich anderes, ob man einen String mit vorgegebener Zeichenkette direkt besetzt, oder die DIM-Klausel benutzt. Die beiden folgenden Formatierungen sind in ihrer Art weder gleichwertig noch in der Behandlung später identisch:

```
LET A$  = "____"
DIM A$ (4)
```

Man beachte - bei DIM bleibt der String in seiner vollen Länge erhalten und muß qualifiziert werden! Arbeitet man hingegen mit der LET-Klausel kann die Variable (A$) beliebig verkürzt, verlängert, gedruckt und anderen Stringvariablen zugewiesen werden. Folgendes ist also problemlos möglich:

```
LET A$  = "___"
  :
LET A$  = "____"
PRINT A$ etc.
```

Im Gegensatz dazu bleibt bei der Dimensionierung (mit DIM) die ursprüngliche Länge und Aufteilung erhalten. Es wird nicht mit Spaces aufgefüllt, vielmehr muß man die Elemente einzeln ansprechen. Kein Wunder, statt einer Kette hat man sich einen Array gebildet!

Wobei in diesem Zusammenhang die Frage auftaucht, wie man bei Vergleich oder Inputcheck die Spaces rausbekommt. Da hilft eine einfache FOR NEXT-Schleife, allerdings nur bei reinen (d.h. nicht mit DIM angelegten) Strings. Soll der zu prüfende String I$ "gereinigt" werden, sieht die entsprechende Routine etwa wie folgt aus:

```
LET X$  = ""
FOR N = 1 TO LEN I$
IF I$ (N) <> " "
   THEN LET X$ = X$ + I$ (N)
NEXT N
LET I$ = X$
```

Natürlich läßt sich die IF-Abfrage ganz umgehen, indem man einen direkten AND-Zeichenvergleich anstellt (dazu weiter unten mehr). Die IF-Zeile läßt sich also ersetzen durch:

```
LET X$ = X$ + (I$ (N) AND I$ (N) <> " ")
```

Zahlen in Strings

Wenden wir uns dem Hauptanwendungsgebiet "Zahlen in Strings" zu. - Am einfachsten läuft die Geschichte, die 5 Bytes pro Floatingpoint-Zahl zu sparen, wenn der Wertbereich 0 bis 9 beträgt. Dann kann man numerische Variablen mit der STR$-Klausel wieder aus diesem herauslesen. Möchte man beispielsweise eine PRINT-Positionierung vornehmen, wird eine herkömmliche Lösung wie folgt mit vielen Zahlen operieren:

```
DIM  A (4)
LET  A (1)  =  1
LET  A (2)  =  7
LET  A (3)  =  5
LET  A (4)  =  3
  :
PRINT AT A(1), 0; A$; AT A(4), 0; B$; AT A (3), 0;
      "BITTE EINGABE"
```

Wesentlich eleganter und speicherplatzärmer ist es, die Ganzzahlen in einem String - hier C$ - abzulegen. Die Lösung kommt auf wesentlich weniger Programmzeilen, wenngleich das Ganze nicht unbedingt schneller werden dürfte:

```
LET C$  = "1753"
  :
PRINT AT VAL C$ (1), 0; A$; AT VAL C$ (4), 0;
      B$; AT VAL C$ (3), 0; "BITTE EINGABE"
```

Im übrigen können alle Vergleiche und auch Sprungziele (wie bei numerischen Variablen) mit der VAL-Klausel aus dem String entnommen werden. Einzige Beschränkung ist, daß jeweils pro Stelle nur ein Zahlenwert von 0 bis 9 vorkommt.

Wie sieht es nun mit größeren Zahlen aus? Man behalte bitte im Gedächtnis, daß Stringzahlen immer ganzzahlig positiv sein müssen und sich innerhalb bestimmter Grenzwerte bewegen. Wir gehen darauf noch näher ein.

Zunächst gibt es zwei Tricks, um mehr als eine Zeichenstelle mit Zahlen zu besetzen bzw. Zahlen aus ihr zu entnehmen:

a) Indem man den String wie oben erwähnt dimensioniert, oder eine bestehende Zeichenkette je nach Ziffernanzahl aufteilt. Bleibt man bei der Regel "1 Zahl = 1 Zeichen", muß man allerdings ein bißchen Arithmetik bei der Abfrage walten lassen. Gesetzt den Fall, es sollen die Werte 13, 42, 63 und 0 zum Plotten aus einem String genommen werden; da wird die Lösung ungefähr so aussehen:

```
LET A$  =  "13426300"
FOR N   =  1 TO LEN A$ STEP 2
PLOT N, VAL A$ (N TO N+1)
NEXT N
```

   Möchte man die STEP-Variable außen vor lassen, kann man natürlich auch einen String-Array anlegen, der dann die Plotpositionen liefert.

b) Die Strategie "Zahlen aus Strings" kann noch weiter auf die Spitze getrieben werden - durch die CODE-Verwendung. Dabei besteht die Möglichkeit, auf die unterste Ebene des Rechners zuzugreifen, nämlich auf ein Byte. Das hört sich zunächst selbstverständlich an; aber wer sich im Manual Anhang A anschaut, wird feststellen, daß in einem Byte (durch unterschiedlich besetzte Bits) Codewerte von 0 bis 255 unterzubringen sind. Es gibt also 256 Zustände, die man in einem Byte speichern kann. Das ist mehr, als eine Zahl in Gleitkommadarstellung pro Byte ermöglicht. Bewegt man sich im positiven Ganzzahlenbereich von 0 bis 255, hilft die CODE-Klausel weiter. Ein solcher Zahlenwert kostet jeweils nur ein Byte, das Ganze ist zudem schneller als die Verwendung von VAL.

   Viele gebräuchliche Zahlen beim ZX sind in diesem Bereich angesiedelt, z.B. Zeilen /Spalten beim Drucken (Werte von 0 bis 21 bzw.

31), Plotposition (von 0 bis 43 bzw. 64). So kommt man mit Codes zurecht, wenn es gilt, große Datenmengen ökonomisch zu speichern. Die Frage der Laufzeit steht dem wohlgemerkt unter Umständen entgegen.

Wer größere Zahlen speichersparend unterbringen will, kann durchaus auf mehr als einen CODE-Wert (zur Unterscheidung) zurückgreifen. Werden zwei CODE-Werte, d.h. zwei Bytes verwandt, ist der Bereich von 0 bis 65025 (=265 x 255) abgedeckt. Zählt man die 0 mit, lassen sich 256 x 256 verschiedene Zustände speichern. Die Systematik wird einem deutlich, wenn man mit PEEK'S und POKE'S arbeitet: Da zeigt sich, wie die Speicherplatzverwaltung bei der 8-Bit-CPU des SINCLAIR arbeitet.

Beispiel "Aktienkurse"

Werden wir konkret und veranschaulichen, wie man sowas nutzbringend anwendet. Wir haben z.B. Aktienkurse, die sich innerhalb des genannten Wertebereiches (0 bis 255) gewegen. Wie sieht nun die Ein-Byte-Speicherung per CODE aus?

Da ist beispielsweise die Aktie AEE, deren Name im String A$ stehen soll. Am einfachsten wäre es, die Kurse einfach an diesen String anzuhängen. Möchte man noch das jeweilige Datum aufnehmen, kann das ebenfalls per CODE-Zahl erfolgen: Bleiben doch der Monat (Ziffer 1 bis 12) und der Monatstag (Zahlenwert 1 bis 31) innerhalb des gegebenen CODE-Bereichs. Womit man ein Datum in zwei Bytes unterbringen könnte. Selbstverständlich geht es auch kürzer, jedoch muß man dann bestimmte Konventionen treffen. Etwa kann man sich die Tatsache zunutze machen, daß Samstags und Sonntags keine Geschäftstätigkeit erfolgt, ebenso wie an Feiertagen. Zählt man so die laufenden Tage fort, (Montag bekommt die Zahl 1, Freitag die Zahl 5) und die Wochen ebenfalls, so kommt man in einen Wertebereich, der um die Zahl 20 liegen dürfte. Diese Zahl mit dem jeweiligen Monat multipliziert, bleibt unterhalb der Höchstgrenze von 255, was als CODE-Zahl maximal gespeichert werden kann. Mit anderen Worten, man kann das Datum bei geschickter "Konfektionierung" in einem einzigen Byte unterbringen. - Wenden wir uns wieder den Kursen zu.

Nun werden die Notierungen dem Hobbyprogrammierer nicht den Gefallen tun, immer zwischen 0 und 255 zu liegen. Größere Kurse können jedoch untergebracht werden, wenn man die Untergrenze einer Bewegung vorab speichert. Lag die niedrigste Notierung der AEE-Aktie beispielsweise bei 481 und die bisherige Höchstnotierung bei 593, werden nicht diese Zahlen, sondern die Differenz als CODE-Wert abgelegt. Bei den vorgegebenen Zahlen bleibt der Differenzwert von 112 noch innerhalb des CODE-Bereichs. Am besten geht man so vor, die Untergrenze am Anfang des "Aktienstrings" (A$) direkt als 3-stelligen Zahlenwert abzuspeichern. Im folgenden werden dann die aktuellen Notierungen (als Differenzen) angefügt und je nach Bedarf ausgegeben. Der Ausdruck des Strings für die AEE-Aktie ist relativ einfach, wenn man sich deren Aufbau vor Augen hält. A$ ist wie folgt angelegt:

```
A$ = "AEE-481------"
      !    !      !
 Aktien-  Kurs-  Kurscodes als
 name     Mini-  Differenz z. Minimum
          mum
```

Werte, die solchermaßen gespeichert sind, können in einer simplen FOR NEXT-Schleife ausgedruckt werden:


```
PRINT "KURSENTWICKLUNG VON" ; A$ ( TO 3)
LET M  = VAL A$ (4 TO 6)
FOR N  = 7 TO LEN A$
PRINT M + CODE A$ (N)
NEXT N
```

Und wie bekommt man die laufende Kursentwicklung in den Rechner herein? Nun, die entsprechende INPUT-Routine berechnet einfach - nachdem Aktienname und Kursminimum eingegeben wurden - die Kursdifferenzen und setzt diese in CHR$-Codes um. Darauf werden sie an den String A$ als "laufende Notierung" angehängt. Die nächstenn Programmzeilen sind etwas umständlich programmiert, aber das geschah wegen der besseren Verständlichkeit. Mit der folgenden Codierungs-Routine hat man zumindestens einen Eindruck, wie sich mit CODE-Zahlen äußerst speicherplatzsparend und noch hinreichend schnell arbeiten läßt:

```
PRINT "KURSEINGABE: AKTIE? MINIMUN?"
INPUT I$
LET A$ = I$
INPUT M
LET I$ = STR$ M
LET A$ = A$ + I$
PRINT "EINGABE LFD NOTIERUNG (0 = ENDE)"
(200)INPUT N
IF N = 0 THEN GOTO ...
LET N = N - M
LET I$ = CHR$ N
LET A$ = A$ + I$
GOTO 200
```

3. Vergleichslogik

Unbekannte Möglichkeiten

Die Möglichkeiten beim SINCLAIR, mit einer spezifischer Vergleichslogik zu arbeiten, sind weithin unbekannt. Diese Stärke des englischen Winzlings dürfte gleich hinter seiner außergewöhnlichen Stringverarbeitung rangieren. In anderen BASIC-Dialekten unmögliche Abfragen, wie

```
IF I = D = D THEN GOTO....
```

oder raffinierte logische Operatoren bei Zeichenketten (siehe weiter oben die Ausführungen zur Printsteuerung) können nur mit Erstaunen zur Kenntnis genommen werden. Ohne im folgenden alle Möglichkeiten und Anwendungsfälle auszuloten (hierzu und zu anderen Besonderheiten des SINCLAIR-BASIC gibt es ein eigenes Buch des Herausgebers), soll auf einige wesentliche Dinge eingegangen werden. Das einfachste ist zunächst immer der Vergleich, ob richtig oder falsch. Eine solche Prüfung bekommt beim ZX81 einen eigenen Vergleichswert. Grundsätzlich erhält jeder Vergleich das Ergebnis 0 = falsch oder 1 = richtig. "0" und "1" als Vergleichswert können aber auch direkt aus einer Operation abgeleitet werden. Mit anderen Worten, die Ziffer 0 oder 1 kann durch einen Vergleich erzeugt werden. Statt beispielsweise zu schreiben:

```
LET X = 1 oder
LET X = 0
```

setzt der Rechner diese Werte ein, wenn folgende Form einer bestehenden Variablen gewählt wurde:

```
LET X = D = D (ergibt die Zahl 1)
LET X = NOT D = D (ergibt die Zahl 0)
```

Wenngleich solche Vergleiche auch bei Strings möglich sind (dann jedoch als Ergebnis "in Zeichenform" ein Leerzeichen liefern), dürften sie bei numerischen IF-Abfragen eine große Rolle spielen. Die verkürzte Abfrage, welche die Vergleichslogik des Rechners ausnutzt, hat ziemlich alle Vorteile auf ihrer Seite:

a) ist weniger einzutippen,
b) wird das Listing kürzer,
c) benötigt man weniger Programmspeicher und
d) läuft die Operation schneller ab.

Man befleißige sich also folgender Ausdrücke:

```
Statt IF X = 0 THEN GOTO 100 schreibt man
      IF NOT X THEN GOTO 100
Statt IF X = 1 THEN GOTO 200 schreibt man
      IF X THEN GOTO 200
```

Selbstverständlich sind damit die Grenzen der Vergleichslogik bei weitem nicht erreicht. Man kann vielmehr die Boolschen Operatoren AND und OR mit einbeziehen. Dazu ein Zahlenbeispiel:

Die Forderung lautet - wenn ein Warengewicht größer oder gleich einem vorgegebenen Grenzwert ist, soll zur Gesamtsumme ein Frachtzuschlag von 10 % geschlagen werden. Für das Beispiel seien folgende Größen vorgegeben:

A = Warengewicht
B = Grenzwert
N = Rechnungsbetrag

Die simple Formel, um die gestellte Forderung zu erfüllen, lautet ganz einfach:

```
LET N = N * (1.1 OR A < B)
```

Wie kommt das zustande? Analysieren wir kurz den Vorgang. - Wenn A kleiner B ist, bleibt das Warengewicht unter dem Grenzwert. Ist die Beziehung A kleiner B wahr, bekommt der Vergleich den Wert 1. Die Zuweisungsbeziehung wird dann zum Befehl

```
LET N = N * 1
```

Ist hingegen der Warenwert größer oder gleich dem Grenzwert, erhält der Vergleich A kleiner B die Klassifikation "falsch". Der Vergleichswert wird 0. In diesem Falle ist der rechte Wert von OR unzutreffend, erhält also die Wertigkeit 0. Also kommt der linke Klammerwert zur Anwendung (vergleiche Manual Seite 71). Damit stellt sich folgende Zuweisung ein:

```
LET N = N * 1.1
```

Stringlogik

Bei Strings kann ebenfalls die Vergleichslogik trickreich eingesetzt werden. Dabei ist egal, ob es sich um einen Vergleich numerischer oder nicht numerischer Größen handelt. Allerdings müssen diese mit der AND-Bedingung verbunden sein. Andere Boolsche Operatoren sind nicht zulässig (siehe Manual). Hat nun ein Vergleich bei einer Zeichenkette das Ergebnis 1, d.h. die Beziehung wird als richtig erkannt, so wird der erste Operant genommen. Sonst liefert der Vergleich einen 0-String. Das Ganze wird deutlich an einem praktischen Beispiel:

Die Größer-/Kleiner-Beziehung der Variablen X und A soll dazu verwandt werden, den Ausdruck "GROESSER" oder "KLEINER" zu printen. Nun kann man ein solches Vorhaben mit zwei kurzen Abfragen realisieren. Es geht aber ohne IF und vor allen Dingen in nur einer Programmzeile. Dazu wird der Ausgabestring A$ "manipuliert". Der Text "GROESSER" oder "KLEINER" wird wie "GLEICH" direkt in Abhängigkeit vom numerischen Vergleich ein- bzw. ausgeschaltet. Die beiden Programmzeilen, um das auszugeben, sehen so aus:

```
LET A$ = ("GROESSER" AND X > A) +
         ("KLEINER" AND X < A)  +
         ("GLEICH" AND X = A)
PRINT "ERGEBNIS IST "; A$
```

Ansonsten ließen sich eine Menge Möglichkeiten aufzeigen, mit Vergleichsoperationen "zu schalten". Insbesondere im Zusammenhang mit der indirekten Adressierung (bei GOTO's und GOSUB's) gibt es eine ganze Menge kniffliger Tricks, um der Rechnerlogik auf die Spur zu kommen. Allerdings ist die Frage, ob solches immer angebracht ist. An Stellen, wo es auf Geschwindigkeit nicht ankommt, wie im Eröffnungsteil eines Programms, kann man eine Menge Zeilenschreiberei mit folgenden indirekten Adresszuweisungen vermeiden. Womit wir denn diesen Teil spezieller "Sinclair-Logik" verlassen möchten:

```
200  INPUT N
     IF N < 1 OR N > 3 THEN GOTO 200
     GOTO 220 * (N = 1) + 256 * (N = 2) + 306 * (N = 3)
```

4. Weitere nützliche Tips

Rund ums System

Es würde zu weit führen, im Rahmen dieses Kapitels alle Möglichkeiten trickreicher und vor allen Dingen effizienter Programmierung aufzuzeigen. Das folgende daher nur zur Abrundung für den, der es halt noch nicht weiß.

Wenden wir uns Dingen zu, die bei längeren Arbeiten mit dem ZX81 positiv wie negativ herauskommen. Da sind zunächst eine ganze Reihe sogenannter BUG's (= Systemfehler). Nun sollte man nicht bei Programmabsturz gleich den Sinclair-ROM für die eigene Dusseligkeit verantwortlich machen. Fast immer gibt es bei unerwartetem Programmende oder Teststop eine plausible, naheliegende Erklärung. Und die geht in der Regel, d.h. zu 99 %, zuungunsten des Hobbyprogrammierers aus...Nichtsdestoweniger existieren einige Dinge, bei denen ROM und Betriebssystem Schwächen zeigen. Sie seien in aller Kürze hier - ohne Anspruch auf Vollständigkeit - stichwortartig wiedergegeben:

- Man hüte sich bei der STEP-Variablen einer FOR NEXT-Schleife vor Werten < 1 oder ungeraden, gebrochenen Zahlen. Interne Rechenungenauigkeiten schaukeln sich schnell hoch, so daß die gewünschte Anzahl Durchläufe nicht in jedem Falle erreicht wird.

- Damit im Zusammenhang vermeide man allzu leichtfertiger Verwendung der INT-Funktion. Nimmt aufgrund interner Rechenungenauigkeit und

dauernder Zahlenkonvertiererei des ZX81 eine "glatte Zahl" plötzlich den Wert 1,99999 an, liefert die INTEGER-Klausel bedauerlicherweise nicht den Wert 2, sondern nur eine 1 !

- Da "INTEGER" wiederum dazu benutzt wird, "Rechenreste" totzuschlagen, kann man buchstäblich vom Regen in die Traufe kommen. - Was nun gar nicht funktioniert, ist eine Operation mit dem "Rechenrest" 0,5. Gerade beliebte Verwendung dieses Ausdrucks bei Dezimal-Rundungsroutinen kann ins Auge gehen. - Lösung: Entweder wechselt man in eine andere Zahlendarstellung über, die der Rechner sowieso "lieber" hat, oder man behilft sich mit anderen Ausdrücken, z.B. 1/2.

- Andererseits kann man seinerseits viel tun, um unangenehme Dinge wie Tabellenüberlauf oder den gefürchteten "Integer out of range" zu vermeiden. Etwa indem man mit glatten Zahlen arbeitet, grundsätzlich auf GROESSER/GLEICH abfragt und Rechendifferenzen durch Addition eines Minimalwertes (z.B. 0.001) wegfegt. Da wir gerade bei dem sind, was "den ZX81 im Innersten zusammenhält", ein paar nützliche Systemroutinen ...

- Zeile 23 und 24 kann benutzen, wer vorher Platz geschaffen hat mit POKE 16418,0. Es gilt allerdings, die erforderlichen Restriktionen zu beachten:

  a) Es darf kein SCROLL oder CLS verwandt werden,
  b) man darf keine System-Nachrichten provozieren (notfalls unterdrückt man diese mit PAUSE 4E4),
  c) der übliche INPUT-Befehl muß durch INKEY$ ersetzt werden.

- Möchte man dem, der unbefugt in ein Programm einbricht, einen üblichen Streich spielen, kann mit RAND USR 0 eine entsprechende ZX81-RAM-Routine dafür sorgen, den gesamten RAM-Bereich bis zum Maschinenstapel zu löschen!

- Womit wir beim Thema "Speicherplatzgröße" sind. Um zu erfahren, wieviel an Speicher belegt bzw. was noch frei ist, ist zu unterscheiden:

  a) Das reine Programm mit seinen Variablen, was man mit PRINT PEEK 16404 + 256 * PEEK 16405 - 16509 herausbekommt.

  b) Den Platz, welchen Programm plus Variablen und Systemvariablen sowie zusätzlich der Bildschirm gekostet haben, was mit PRINT PEEK 16404 + 256 * PEEK 16405 - 16384 aufgezeigt wird.

  c) Wieviel vom gesamten RAM-Speicher noch übrig ist, wozu die folgende Anweisung dient: PRINT PEEK 16386 - PEEK 16412 + 256*(PEEK 16387 - PEEK 16413) - 50.

Programmschutz

Das Kapitel TIPS & TRICKS kann nicht abgeschlossen werden, ohne ein Wort zum Programmschutz zu verlieren. Erwarten Sie nun bitte nicht ausgefuchste Insider-News, wie man fremde Programme knackt. Wir werden uns nicht auf die Ebene spezialisierter Systemfreaks herabbegeben und nur für eine Minderheit Interessantes bringen. Halten wir uns bei Dingen auf, die von allgemeinem Interesse sind.
Programmschutz kann man unterteilen in *Listschutz* und *Copyschutz*. Ersteres bedeutet, daß sich ein Programm nicht mehr anlisten läßt, so daß man (zumindest im BASIC) die Befehlsfolge nicht zu Gesicht bekommt. Copyschutz bedeutet, daß ein Programm ohne besondere Maßnahme nicht kopiert werden kann. Bei letzterem dürfte eine absolut "einbruchsichere" Vorkehrung nur darin bestehen, dem Laden eine eigene modifizierte Laderoutine vorherzuschicken. Es mögen sich die Spezialisten darüber streiten, ob es in dem Zusammenhang absolut sichere Programme gibt oder nicht...

Der Listschutz läßt sich in schwächster Form durch eine eigene Copyright-Zeile herbeiführen. Wenn schon jemand unbefugt das Programm listet, soll er auf das Urheberrecht hingewiesen werden. Natürlich sind diese und weitere Maßnahmen bestenfalls dazu geeignet, Anfänger vom üblen Tun abzuschrecken. Aber zurück zum Copyright-Vermerk. Mit folgendem Poke kann die vorhergende Zeile 10 zur nicht editierbaren und löschbaren Null-Zeile gemacht werden:

```
10 REM COPYRIGHT XYZ
POKE 16510,0
```

Eine stärkere Maßnahme gewährt einen gewissen Listschutz dadurch, daß das High-Byte der ersten Zeile höher als die höchstzulässige Zeilennumerierung gesetzt wird. Man erreicht dies beispielsweise mit folgendem Poke:

```
POKE 16509,200
```

Allerdings hat die Sache den Nachteil, daß das Programm nicht mehr mit GOTO's angesprungen werden kann, da sozusagen die erste Zeile die letzte geworden ist. Außerdem dürfte jeder "Kenner" des ZX schnell wissen, wie man die Maßnahme rückgängig macht, nämlich durch:

```
POKE 16509,0
```

Stärker entwickelt als der Listschutz ist ein BREAK-Schutz, welcher verhindern soll, daß bei Programmablauf die Bestätigung der BREAK-Taste einen ungewollten Stop und damit die Möglichkeit, das Programm in Augenschein zu nehmen, eröffnet. Nun hat die BREAK-Taste beim ZX81 die unangenehme Eigenschaft, für die Eingabe von SPACE benötigt zu werden! Außerdem funktioniert ein BREAK-Schutz nicht für den Ladevorgang. Zwar hat ein "Einbrecher" nicht viel davon, wenn er nur das vollständig geladene Programm untersuchen kann. Aber - ein Sprung in die Load-Routine des ROM kann auch so geschehen, daß sauber geladen wird, anschließend das Programm mit einer provozierten Fehlermeldung stehenbleibt und angelistet werden kann. Der beliebte Autostart (hinter SAVE im Programm) hilft dagegen nichts. Wen es übrigens interessiert, der entsprechende Ladestop wird erreicht mit dem Befehl

```
RAND USR 836
```

nachdem der ZX81 in den FAST-Modus geschaltet wurde. Selbstverständlich gibt es eine ganze Reihe von Methoden, ein geschütztes und mit Copyschutz versehenes Programm aufzubrechen. Um nur einige Dinge in diesem Zusammenhang anzudeuten: Eine Methode geht dahin, das RAMTOP so niedrig zu setzen, daß das Programm geladen, aber nicht gestartet wird. Man kann nach einem solchen Stop die Speichergrenze wieder heraufsetzen, um das Programm anzulisten und zu analysieren. Allerdings ist dazu eine mühselige Programmgrößenbestimmung notwendig (welche beim Spektrum freundlicherweise der Header mitliefert). Jeder Systemcrack wird in Verbindung mit einem leistungsfähigen Disassembler wohl ziemlich alle Programm-Schutzmaßnahmen überwinden können. Simple Dinge, wie beabsichtigter Absturz bei nicht "vorschriftsmäßiger" Eingabe, sind nur was für weniger fortgeschrittene Gemüter. Selbst wenn

man Schlüsselworte abfragt, die der berechtigte Benutzer getrennt zur Programmkopie erhält,stellt das kein unüberwindliches Hindernis dar. Lediglich, wenn eine abzufragende verzwickte Speicherzahl bei einer bestimmten Konstellation im Rechner vorhanden ist, dürfte die Arbeit gewisser "Programmklauer" erschwert sein.

Mit diesen, normale Vorstellungen eines Anwenders überschreitenden Ausführungen möchten wir den Kreis schließen. Wie sicher dem ein oder anderen Leser aufgegangen sein wird, steckt mehr im ZX81, als man zunächst annimmt. Nicht nur von der Anwendungsseite her, sondern auch im Bezug auf das System im Innern des SINCLAIR. Ziel und Absicht dieses Buches war, für den Benutzer des kleinen Rechners aus England Anwendungsfelder aufzuzeigen. Und wir hoffen, daß der Leser neue Anregungen und Ideen geschöpft hat, um sich verstärkt mit seinem Mikrozwerg zu beschäftigen.

Programmverzeichnis

Name/Bezeichnung	Merkmale	Umfang Progr.-Zeilen (ca.)	Sonstiges	Seite
LOTTO 1	Lotto-Zufallszahlen	17		113
LOTTO 2	Lotto-Tips mit Vorgabezahlen	73		113/114
LOTTO 3	Lottostatistik	140		114/115
SCPD Scratch Pad	einfache Textdatei	120		116/117
Text	satzweises Update eines Textfiles	100		118/119
WORD	Wortprozessor mit Formatierung und Cursorführung	150		119-121
Vermögen	Verwaltung und Darstellung der Entwicklung von Aktien, Renten, Sparkonten u.Gold	110		122-124
Aktien	Bestandsführung und Charts von Aktien plus Index	310	11,6 K RAM	125-128
PC Project Control	Kontrolle von Arbeitspaketen mit Termindaten, Kosten und Meilensteinen (Verwaltung+Darstellung)	280	12,0 K RAM	128-131
MKART Mitglieder-kartei	Führen der Mitgliedsdaten eines Sportvereins (Adres. und Sportarten) plus Gesamt- u. Spartenstatistik	340	variable Satzverarbeit.	132-136


```
150 REM ** L O T T O  1 *******
        **********************

160 PRINT "WIEVIEL TIP""S ?"
170 INPUT N
175 IF NOT N THEN  GO TO 160
180 FOR N=1 TO N
190 DIM Z$(49)
200 FOR M=1 TO 6
210 LET R=INT (RND*49)+1
220 IF Z$(R)<>" " THEN  GO TO 2
10
230 LET Z$(R)="*"
240 NEXT M

245 REM >>AUSDRUCK D. ZAHLEN <<

250 FOR M=1 TO 49
255 IF Z$(M)="*" THEN  PRINT M;
" ";
260 NEXT M
265 PRINT
270 NEXT N
100 REM *** L O T T O  2 *****
        **********************

105 REM TIP'S MIT VORGABEN

110 PRINT "LOTTO 2",,"-------",
,
120 REM ---VORBEREITUNGSTEIL---
          -----------------

125 PRINT "WIEVIEL TIP'S ?"
130 INPUT N
135 IF NOT N THEN  GO TO 125
140 PRINT "=>";N
141 REM >>DEF.F.VORGABEN/ZAHLEN

142 DIM Z$(49)
143 LET L=0
144 LET V$=""
145 PRINT "VORGABEZAHLEN ?  ","
KEINE/ENDE EINGABE=0"
148 INPUT V
150 IF NOT V THEN  GO TO 160
152 IF V>49 THEN  GO TO 145
154 PRINT V;" ";
156 LET V$=V$+CHR$ V
158 GO TO 148
160 IF V$="" THEN  GO TO 175
161 REM ->VERWENDUNG V.VORGABEN
          ......................

162 PRINT "AUFTEILUNG D.VORGABE
ZAHLEN AUF  TIP'S? (0=NEIN,1=JA)
",
163 INPUT V
164 IF V<0 OR V>1 THEN  GO TO 1
62
166 IF V THEN  GO TO 180
167 IF LEN V$<6 THEN  GO TO 170
168 PRINT "VORGABEZAHLEN>=6,  "
,"VERTEILUNG NOTW."
169 GO TO 162

170 REM >> FESTE VORGABEZAHLEN
          (OHNE AUFTEILUNG)

171 LET L=LEN V$
172 FOR M=1 TO L
173 LET Z$(CODE V$(M))="+"
174 NEXT M

175 FOR N=1 TO N
176 GO SUB 210
177 NEXT N
178 STOP

180 REM >>VERARBEITUNG VON VOR-
       GABEZAHLEN M. AUFTEILUNG

182 LET V=LEN V$/N
184 IF V<6 THEN  GO TO 187
185 PRINT "VORGABEN PRO TIP>6
       ->NEUEINGABE"
186 GO TO 141
187 FOR N=1 TO N
188 LET W=V*N
189 LET W=INT (W+.0001)-INT (W-
V)
190 IF NOT W THEN  GO TO 196
191 DIM Z$(49)
192 LET L=0
193 FOR X=1 TO W
194 GO SUB 260
195 NEXT X
196 GO SUB 208
197 LET L=0
198 NEXT N
199 STOP

208 REM ->ZUFALLSZAHLEN + PRINT
          ......................

209 IF NOT L THEN  DIM Z$(49)
210 FOR M=1 TO (6-L)
212 LET R=INT (RND*49)+1
```

```
 215 IF Z$(R)<>" " THEN  GO TO 2
12
 220 LET Z$(R)="*"
 225 NEXT M
 230 FOR M=1 TO 49
 235 IF Z$(M)<>" " THEN  PRINT M
;" ";
 240 IF Z$(M)="*" THEN  LET Z$(M
)=" "
 245 NEXT M

 247 PRINT
 250 RETURN

 260 REM >> VORGABEZAHLEN ZUTEI-
       LEN UND IN Z$ EINFUEGEN

 270 LET Z$(CODE V$(1))="+"
 275 IF LEN V$>1 THEN  LET V$=V$
(2 TO )
 280 LET L=L+1
 285 RETURN


 500 REM *** L O T T O  3.1 ****
       **********************

 502 REM >> PROGRAMMVORLAUF <<<

 504 PRINT
 506 LET M$="LOTTO 3 (STATIKSTIK
)***1.ERSTEINGABE*2.FORTSCHREIBE
N*3.PRINT*4.SAVE*5.STOP*"
 508 FOR N=1 TO LEN M$
 510 IF M$(N)<>"*" THEN  PRINT M
$(N);
 512 IF M$(N)="*" THEN  PRINT
 514 NEXT N

 516 INPUT N
 518 IF N>0 AND N<6 THEN  GO TO
522
 520 GO TO 516
 522 CLS
 523 GO TO 526*(N=1)+574*(N=2)+6
08*(N=3)+632*(N=4)+640*(N=5)
 524 REM >> HAUPTPROGRAMM <<
       ------------------

 526 REM ->E R S T E I N G A B E

 528 DIM W$(52,6)
 530 PRINT "ERSTEINGABE WOCHENER
GEBNISSE",,,
 532 PRINT "EINZUGEBENDE WOCHE?
","(0=ENDE)"
 534 INPUT W
 536 IF W=0 THEN  GO TO 564
 538 IF W<1 OR W>52 THEN  GO TO
532
 540 PRINT W;".WOCHE,WELCHE ZAHL
EN ?"
 542 IF W$(W)="       " THEN  GO
TO 550
 544 PRINT "WERT VORH.-KORREKTUR
?(Y/N)"
 546 INPUT A$
 548 IF A$="N" THEN  GO TO 532
 550 FOR M=1 TO 6
 552 INPUT N
 554 IF N<1 OR N>49 THEN  GO TO
552
 556 PRINT N;" ";
 558 LET W$(W,M)=CHR$ N
 560 NEXT M
 562 PRINT
 564 PRINT "ENDE EINGABE?(Y/N)"
 566 INPUT A$
 568 IF A$="N" THEN  GO TO 532
 570 CLS
 572 GO TO 500
 574 REM F O R T S C H R E I B E
N
 576 PRINT "FORTSCHREIBEN WOCHEN
ZAHLEN",,,
 578 PRINT "LESEN VON WO.ERGEBNI
SSE?(J/N)"
 580 INPUT A$
 582 IF A$="J" THEN  GO TO 586
 584 GO TO 532
 586 PRINT "WOCHE ?"
 588 INPUT W
 590 IF W<1 OR W>52 THEN  GO TO
588
 592 IF W$(W)<>"       " THEN  GO
 TO 598
 594 PRINT "KEINE ZAHLEN IN WO.
;W
 596 GO TO 578
 598 GO SUB 789
 600 PRINT AT 21,0;"WEITER=>TAST
E DRUECKEN"
 602 PAUSE 4E4
 604 CLS
 606 GO TO 578
 608 REM -> P R I N T  TABELLEN

 610 PRINT "TABELLEN-PRINT:",,"1
=ALLE BESETZTEN WOCHEN","2=NUR H
AEUFIGKEITEN","3=BEIDE",,"4=RUEC
KKEHR INS MENUE"
 612 INPUT N
 614 IF N>0 AND N<5 THEN  GO TO
618
 616 GO TO 610
 618 IF N=4 THEN  GO TO 500
 620 IF N=1 OR N=3 THEN  GO SUB
646
 622 IF N>=2 THEN  GO SUB 672
 624 PRINT AT 21,0;"WEITER=>TAST
E DRUECKEN"
 626 PAUSE 4E4
```



```
 628 CLS
 630 GO TO 610
 632 REM S P E I C H E R N+ STOP

 634 PRINT "SAVE MIT NAMEN ""LOT
TO 3""","RECORDER STARTEN"
 636 SAVE "LOTTO 3"
 638 GO TO 500
 640 REM -> S T O P

 642 STOP

 644 REM -- UNTERPROGRAMME -----
       ------------------------

 646 REM - PRINT  WO.ZIEHUNGEN

 648 CLS
 650 LET OK=0
 652 PRINT "WOCHENZIEHUNGEN",,,,
 654 FOR W=1 TO 52
 656 IF W$(W)="       " THEN  GO
TO 664
 658 GO SUB 788
 660 LET OK=1
 662 PRINT
 664 NEXT W
 666 IF OK THEN  GO TO 670
 668 PRINT ,,!"WOCHENTABELLE LEE
R" AND NOT OK!,,,
 670 RETURN

 672 REM - PRINT HAEUFIGKEITEN

 674 PRINT
 676 PRINT "HAEUFIGKEITEN UN/SOR
TIERT ?","Y=SORTIERT"
 678 INPUT A$
 680 IF A$="Y" THEN  GO TO 708
 682 REM *BESETZEN H.TABELLE
       +PRINT
 684 GO SUB 718
 686 CLS
 688 PRINT "HAEUFIGKEITEN UNSORT
IERT"
 690 PRINT "ZAHL = VORKOMMEN",,,
 692 LET Z=3
 694 LET S=0
 696 FOR W=1 TO 49
 698 LET C=CODE H$(W)
 700 GO SUB 774
 702 NEXT W
 704 RETURN

 706 REM ***SORTIERTER H.PRINT

 708 IF NOT MAX THEN  GO SUB 718
 710 PRINT "SORTIERTE HAEUFIGKEI
TEN","------------------------",
 712 GO SUB 740
 714 RETURN

 716 REM - AUFFUELLEN H.TABELLE

 718 LET MAX=0
 720 DIM H$(49)
 722 FOR W=1 TO 52
 724 IF W$(W)="       " THEN  GO
TO 736
 726 FOR M=1 TO 6
 728 LET C=CODE W$(W,M)
 730 LET H$(C)=CHR$ (CODE (H$(C)
)+1)
 732 IF CODE H$(C)>MAX THEN  LET
 MAX=CODE H$(C)
 734 NEXT M
 736 NEXT W
 738 RETURN

 740 CLS
 742 REM "SORTIERUNG" VON H$ +
       PRINT
 744 PRINT "SORTIERTE HAEUFIGKEI
TEN","------------------------"
 746 LET Z=3
 748 LET S=0
 750 LET ZMAX=0
 752 FOR W=1 TO 49
 754 LET C=CODE H$(W)
 756 IF C>MAX THEN  GO TO 766
 758 IF C=MAX THEN  GO TO 764
 760 IF C>ZMAX THEN  LET ZMAX=C
 762 GO TO 766
 764 GO SUB 774
 766 NEXT W
 768 IF NOT MAX THEN  RETURN
 770 LET MAX=ZMAX
 771 GO TO 750
 772 REM +++ UNTERROUTINEN+++++

 774 REM ..ZEILENPRINT H.TABELLE

 776 PRINT AT Z,S;W;"=>";C;
 778 LET Z=Z+1
 780 IF Z<16 THEN  RETURN
 782 LET Z=3
 784 LET S=S+7
 786 RETURN

 788 REM ..AUSGABE E.WOCHE AUS
       DER WOCHENTABELLE

 790 PRINT W;".WOCHE"
 792 FOR M=1 TO 6
 794 PRINT CODE W$(W,M);" ";
 796 NEXT M
 798 RETURN
```

```
  10 REM *S C R A T C H - P A D*
        **********************

  12 REM >> STEUERUNG <<<

  14 LET A$=""
  16 GO SUB 146
  18 LET B$=""
  20 IF A$<>"" THEN  GO TO 30
  22 IF N=5 OR N=13 THEN  GO TO
30
  24 PRINT "TEXTSTRING LEER"
  26 PAUSE 90
  28 GO TO 16
  30 CLS
  32 GO TO 36*(N=5)+60*(N=7)+100
*(N=9)+140*(N=11)+144*(N=13)
  34 REM >>FORT/SCHREIBEN <<<

  36 PRINT ,,"  FORT/SCHREIBEN",
"  ▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬
,,,
  38 IF A$="" THEN  LET A$="*"
  40 PRINT "NL=EINGABEENDE",,
  42 PRINT "SPACE=NEUER ABSATZ",
,,
  44 INPUT I$
  46 IF I$="" OR I$=" " THEN  GO
 TO 52
  48 LET A$=A$+I$+"." AND I$(LEN
 I$)<>"."
  50 GO TO 44
  52 LET A$(LEN A$)="*"
  54 IF I$=" " THEN  GO TO 42
  56 IF A$="*" THEN  LET A$=""
  58 GO TO 16

  60 REM >> LESEN + SUCHEN <<<

  62 CLS
  64 IF I$<>"" THEN  LET S$=""
  66 PRINT "  LESEN + SUCHEN","
▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬",,,
  68 PRINT "1=LESEN OHNE SUCHE",
"2=SUCHEN",,"3=RUECKKEHR Z.HAUPT
MENUE",,,,,"(WEITER B.LESEN=>NL)
",,,
  69 INPUT N
  70 IF N<1 OR N>2 THEN  GO TO 1
6
  71 IF N=2 THEN  GO TO 86

  72 PRINT ,,"L E S   E N ",,,,"
0=GESAMTFILE",,"1=ABSATZ AB POSI
TION(>SUCHE)",,,,
  73 INPUT N
  74 LET N=1+N*(INT POS)

  76 REM .. GESAMT PRINT ...

  78 GO SUB 178
  79 PRINT "NL=>WEITER MENUE"
  80 PAUSE 4E4
  82 GO TO 62

  84 REM ...  SUCHE  .......

  86 PRINT "ES FOLGT S U C H E",
,,"SUCHBEGRIFF ?","(NL=ENDE)",
  88 INPUT S$
  90 IF S$="" THEN  GO TO 62
  92 PRINT S$,
  94 GO SUB 188
  96 IF I$="N" THEN  GO TO 86
  98 IF I$<>"S" THEN  GO TO 62

 100 REM >>AENDERN + LOESCHEN<<<

 102 CLS
 104 PRINT "AENDERN + LOESCHEN",
"  NACH SUCHE",,,,"L-OESCHEN","A
-ENDERN",,,"SUCHBEGRIFF=>";S$,,,
 106 INPUT I$
 108 IF I$="A" OR I$="L" THEN  G
O TO 112
 110 GO TO 106-90*(S$="")
 112 LET P=INT ((POS-INT POS)*10
E3+.01)
 114 LET POS=INT POS
 116 IF I$="A" THEN  GO TO 128

 118 PRINT ,"LOESCHEN:","G-ESAMT
FILE",,"E-INZELN AB/SATZ",,
 120 INPUT I$
 122 IF I$="G" THEN  GO TO 12
 124 LET A$=A$( TO POS)+A$(P+1 T
O )
 126 GO TO 16

 128 PRINT ,"AENDERN:","B-EGRIFF
>SUCHE",,"S-ATZ/ABSATZ GES.",,
 130 INPUT I$
 132 PRINT "EINGABE ?" AND (I$="
B" OR I$="S")
 134 INPUT B$
 136 IF I$="B" THEN  LET A$=A$(
TO A-1)+B$+A$(A+LEN S$ TO )
 137 IF I$="S" THEN  LET A$=A$(
TO POS)+B$+A$(P TO )
 138 GO TO 16

 140 SAVE "SCPD"
 142 GO TO 16

 144 STOP
```



```
 146 REM >> M E N U E - PRINT <<

 148 CLS
 150 LET B$=" FORT/SCHREIBEN LES
EN + SUCHEN AEND. N. SUCHE ABSPE
ICHERN    ENDE VERARB.  "
 152 PRINT ,,,,"  MENUE SCRATCH
- PAD"
 154 PRINT "  ------------------
-",,,
 156 FOR N=1 TO 74 STEP 15
 158 PRINT B$(N TO N+14),,,,
 160 NEXT N
 162 PRINT AT 19,0;"WAEHLEN=>STO
P M.TASTE"
 164 FOR N=5 TO 13 STEP 2
 166 PRINT AT N,0;"■"
 168 PAUSE 50
 170 PRINT AT N,0;" "
 172 IF INKEY$<>"" THEN  RETURN
 174 IF INKEY$="" THEN  NEXT N
 176 GO TO 164

 178 REM ...GESAMT PRINT ...

 180 FOR N=N TO LEN A$-1
 182 PRINT CHR$ 118 AND A$(N)="*
";A$(N) AND A$(N)<>"*";
 184 NEXT N
 186 RETURN

 188 REM .SEARCH N.SUCHBEGRIFF..

 190 PRINT "BEI ""GEFUNDEN"":",,
"1= PRINT SATZ",,"2= PRINT ABSAT
Z",,"GILT F.KORREKTUR",,
 192 INPUT M
 194 IF M<1 OR M>2 THEN  GO TO 1
92
 196 LET I$="W"
 198 LET POS=1
 200 LET L=LEN S$-1
 202 FOR N=POS TO LEN A$-L
 204 LET B$=A$(N)
 206 IF B$<>S$(1) THEN  GO TO 21
4
 208 IF A$(N TO N+L)=S$ THEN  GO
 SUB 224
 210 IF I$<>"W" THEN  RETURN
 212 GO TO 216
 214 IF B$="." AND M=1 OR B$="*"
 THEN  LET POS=N
 216 NEXT N
 218 PRINT S$;" NICHT GEFUNDEN"
,,,
 220 PAUSE 160
 222 RETURN

 224 REM ....FOUND.........

 225 LET A=N
 226 PRINT S$;" GEFUNDEN :",A$(P
OS+1 TO N+L);
 228 FOR P=N+L+1 TO LEN A$-1
 230 IF A$(P)="." AND M=1 OR A$(
P)="*" THEN  GO TO 236
 232 PRINT A$(P);
 234 NEXT P
 236 PRINT ,,,"W-EITERSUCHEN",,"
S-PRUNG->AENDERN","N-EUES SUCHEN
",,,,"(NL=ENDE)"
 238 INPUT I$
 240 IF I$="S" OR I$="" THEN  LE
T POS=POS+P*10E-5
 242 IF I$="W" THEN  LET POS=P
 244 IF I$="W" THEN  LET N=P
 246 RETURN
```

```
  10 REM ****T E X T F I L E****
         **********************

  15 LET A$=""
  18 LET POS=1
  20 CLS
  22 PRINT "   T E X T",,"
     ",,,,"1=LESEN",,"2=SC
     HREIBEN",,"3=SAVEN",,"4=LOESCHEN
     ",,"5=STOP",,"(SAETZE > 3 ZEICHE
     N)"
  24 INPUT N
  26 IF N>0 AND N<6 THEN  GO TO
     N*30
  28 GO TO 20

  30 REM **** L E S E N ********

  32 CLS
  34 PRINT "   L E S E N",,,,"0=
     VON A - Z",,"1=AB POSITION",,,,
  36 INPUT N
  38 LET N=(N=0)+POS*(N=1)
  40 PRINT "S-CHNELLES LESEN","L
     -ANGSAMLESEN",,"  MIT POSITIONS-
     ","   SPEICHERUNG",,,,"( STOP =>
     TASTE DRUECKEN)",,,
  42 INPUT I$
  43 PAUSE 10
  44 FOR N=N TO LEN A$
  46 PRINT A$(N);
  48 IF I$="S" THEN  GO TO 50
  49 GO SUB 190
  50 IF INKEY$<>"" THEN  GO TO 5
     2
  51 NEXT N
  52 IF I$="L" THEN  LET POS=N
  53 PAUSE 20
  54 PRINT
  55 PRINT ,,"*** ENDE LESEN ***
     "
  56 PRINT
  57 PRINT ,"( RETURN =>TASTE","
      DRUECKEN)"
  58 PAUSE 4E4
  59 GO TO 20

  60 REM * S C H R E I B E N ***

  61 CLS
  62 PRINT "   S C H R E I B E N
     ",,,"A-NFANG TEXT()DAVOR)","E-ND
     E TEXT()DAHINTER)","I-N TEXT()EI
     NGEFUEGT","  AB POSITION)",,,,
  64 INPUT I$
  66 IF I$<>"I" THEN  GO TO 73
  68 IF POS=0 THEN  LET I$="A"
  70 IF POS=LEN A$ THEN  LET I$=
     "E"
  71 IF I$="I" THEN  GO SUB 200
  73 PRINT "EINGABE:",,
  74 INPUT B$
  75 PRINT B$
  76 IF I$="A" THEN  LET A$=B$+A
     $
  78 IF I$="I" THEN  LET A$=A$(
     TO POS)+B$+A$(POS+1 TO )
  79 IF I$="E" THEN  LET A$=A$+B
     $
  80 PRINT ,,"***ENDE SCHREIBEN*
     *"
  82 GO TO 56

  90 REM ** S A V E N **

  92 CLS
  94 PRINT "RECODER STARTEN"
  96 PAUSE 200
  98 SAVE "TXT"
  99 GO TO 18
 120 REM *** L O E S C H E N ***

 121 CLS
 122 PRINT "   L O E S C H E N",
     ,,"G-ESAMTFILE",,"P-OSITIONSSATZ
     ",,," (POS AUS LESEN)",,,
 124 INPUT I$
 126 IF I$="G" THEN  GO TO 15
 128 GO SUB 220
 130 IF NOT OK THEN  GO TO 20
 132 PRINT ">>SATZ(<",,A$(A+1 TO
     E);"(= LOESCHUNG OK? => NL"
 134 INPUT I$
 136 PRINT "SATZ WIRD ";("NICHT
     " AND I$<>"")+"GELOESCHT"
 138 IF I$="" THEN  LET A$=A$( T
     O A)+A$(E+1 TO )
 139 IF I$="" THEN  LET POS=A
 140 GO TO 56

 150 STOP
 185 REM ***HAUPTPROGRAMM-ENDE**
         ***(ES FOLGEN UPRO"S)**

 190 REM >>> PAUSE <<

 192 FOR M=1 TO 15
 194 NEXT M
 196 RETURN

 200 REM >>ANDRUCK POSITIONSSATZ

 202 PRINT "POS PRINT ","POSITIO
     N=";POS,,,
 204 LET M=POS-15
 206 LET M=M*(M>0)+(M<1)
 208 LET N=POS+15
 210 LET N=N*(N<=LEN A$)+(LEN A$
     )*(N>LEN A$)
 212 FOR N=M TO N
 214 PRINT CHR$ (CODE A$(N)+128*
     (N=POS));
```



```
 216 NEXT N
 218 PRINT
 219 RETURN

 220 REM >>POSITIONIEREN SATZ<<

 221 REM *OK=1 => FEHLER
 222 LET OK=POS<4 OR POS>=LEN A$
 223 IF NOT OK THEN  LET OK=A$(P
     OS)="."
 224 REM *UMKEHRUNG,UM OK "SPREC
     HEND" ZU MACHEN
 225 LET OK=NOT OK
 226 IF NOT OK THEN  GO TO 242
 228 LET A=LEN A$-1
 230 LET M=1
 231 FOR N=POS TO A STEP M
 232 IF A$(N)<>"." THEN  NEXT N
 233 LET A=N
 234 IF M<>1 THEN  GO TO 242
 235 LET M=-1
 236 LET E=A
 237 LET A=1
 238 GO TO 231
 242 PRINT ,,"POSITIONIERG. "+("
     NICHT " AND NOT OK)+"OK","(WEITE
     R=>NL)"
 244 PAUSE 4E4
 246 RETURN

  12 REM **********************
         *   W O R D          *
         **********************

  13 LET T$="DIES PROGRAMM DIENT
      DER EINFACHEN HANDHABUNG VON TE
     XT.ALLE BEWEGUNGEN ERFOLGEN UEBE
     R CURSOR - WIE BEIM EDITIEREN IM
      PROGRAMM.BEIM LESEN WIRD AUTOMA
     TISCH MIT POSITIONIERT.DIE AUSGA
     BE KANN FORMATIERT WERDEN.(ENDE
     PROBETEXT =>WEITER MIT NL)"
  14 CLS
  15 PRINT AT 8,4;"W O R D",,,,,
     T$
  16 PAUSE 4E4
  18 LET POS=1

  19 REM ** PROGRAMM-STEUERUNG *
         **********************

  20 CLS
  21 PRINT AT 4,2;"M E N U E",,,
     ,"E-RSTEINGABE",,"L-ESEN/SCHREIB
     EN","A-USGABE (FORMATIERUNG)","S
     -PEICHERN",,,"(NL= STOP )"
  22 INPUT A$
  23 IF A$="" THEN  STOP
  24 IF A$="L" THEN  GO SUB 100
  25 IF A$="A" THEN  GO SUB 200
  26 GO TO 20+10*(A$="S")+15*(A$
     ="E")
  30 REM >>> S A V E  <<<
  32 SAVE "WORD"
  33 GO TO 20

  35 REM >> ERSTEINGABE <<<<
  36 PRINT "E R S T E I N G A B
     E",,,"TEXT:"
  37 INPUT T$
  38 LET POS=LEN T$
  39 GO TO 20

  40 REM *** R O U T I N E N ***

  41 REM >>>> P A U S E  <<<

  42 FOR M=1 TO 8
  44 NEXT M
  46 RETURN

  49 REM >>  SEITEN PRINT <<

  50 LET Z=640*(P+1)
  52 IF Z>LEN T$ THEN  LET Z=LEN
     T$
  54 PRINT AT 0,0;T$(1+640*P TO
     Z)
  55 RETURN

  56 REM >> CURSOR PRINT CHECK<<
```

```
  57 IF C>LEN T$ THEN  LET C=LEN
 T$
  58 IF C<=0 THEN  LET C=1
  59 LET Z=INT (((C-P*640)-1)/32
)
  60 LET S=(C-P*640)-32*Z-1
  61 RETURN

  62 REM >CURSOR PRINT (FLASH)<

  63 PRINT AT Z,S;CHR$ (CODE T$(
C)+128)
  64 GO SUB 40
  65 PRINT AT Z,S;T$(C)
  66 GO SUB 40
  67 RETURN
 100 REM *** M O D U L E ******

 101 REM *MODUL LESEN/SCHREIBEN*

 102 CLS
 104 PRINT AT 20,0;"SEITE ?","0=
LFD POS.SEITE"
 106 GO SUB 140
 108 CLS
 110 PRINT AT 20,0;"L E S E N SE
ITE: "¡P+1¡"=>CURSOR Z.POS (""E"
"=ENDE,"" ""=SCHREIB/LOESCHEN)"
 112 GO SUB 50
 114 GO SUB 57
 116 IF Z>=0 AND Z<=19 THEN  GO
TO 122
 118 LET P=P+(Z>19)-(Z<0)
 120 GO TO 108
 122 GO SUB 63
 124 IF INKEY$="" THEN  GO TO 12
2
 126 LET I$=INKEY$
 128 GO TO 132+4*(I$="E")+18*(I$
=" ")
 132 LET C=C+(I$="8")-(I$="5")+3
2*((I$="6")-(I$="7"))
 134 GO TO 114
 136 LET POS=C
 138 RETURN

 139 REM >LES/SCHREIBEROEFFNUNG<

 140 INPUT P
 141 LET POS=POS+(POS=0)
 142 IF P THEN  GO TO 147
 143 LET C=POS
 145 LET P=INT ((C-1)/640)
 146 RETURN
 147 LET C=P*640
 148 LET P=P-1
 149 RETURN

 150 REM * UNTERMODUL SCHREIBEN*

 151 LET POS=C
 153 PRINT AT 20,0;"SCHREIBEN SE
ITE ";P+1," I=INSERT,0=LOESCHEN
                "
 154 IF INKEY$<>"" THEN  GO TO 1
54
 156 IF INKEY$="" THEN  GO TO 15
6
 158 GO TO 154+20*(INKEY$="I")+6
*(INKEY$="0")
 160 REM >> LOESCHEN <<<
 161 PRINT AT 20,0;"LOESCHEN SEI
TE:"¡P+1,"STELLEN/ZEILEN<=CURSOR
(ENDE="" "")"
 162 GO SUB 40
 163 LET B$=INKEY$
 164 IF B$=" " THEN  GO TO 172
 165 LET C=C-(B$="5")-32*(B$="7"
)
 166 GO SUB 57
 167 IF Z<0 OR Z>19 THEN  GO TO
172
 168 PRINT AT Z,S;("■" AND POS>
C)
 169 GO TO 162
 172 LET T$=T$( TO C-1)+T$(POS+(
C=POS) TO )
 173 GO TO 108

 174 REM >>> INSERT-SCHREIBEN <<

 175 PRINT AT 20,0;"SCHREIBEN SE
ITE:"¡P+1,"TEXT INPUT -DANN N.LE
SEN ZURUECK"
 176 LET C=C+(C<>1)
 177 INPUT B$
 178 LET T$=T$( TO C-1)+B$+T$(C
TO )
 179 LET C=C+LEN B$-1
 180 GO TO 108

 200 REM ** MODUL FORMATIERUNG**

 205 REM >>> AUSGABE-OPTIONEN <<

 210 CLS
 212 PRINT AT 5,2;"FORMATIERTE A
USGABE",,,"  =>OPTIONEN :",,,,"G
=ESAMTFILE",,"L-AUFENDE SEITE",,
"C-URSORPOS. SEITE","S-EITENVORG
ABE",,,,"( RETURN V. PRINT > NL)
"
 213 INPUT I$
 214 IF I$="" THEN  RETURN
 215 IF I$="L" OR I$="S" OR I$="
C" OR I$="G" THEN  GO TO 217
 216 GO TO 213
 217 GO SUB 218+(I$="C")+(I$="S"
)*3+(I$="L")*6+(I$="G")*10
 218 GO TO 260
 219 LET A=C
 220 GO TO 225
 221 PRINT AT 20,0;"WELCHE SEITE
?"
 222 INPUT P
 223 LET P=P-1
 224 LET A=1+640*P
 225 LET E=(P+1)*640
 226 IF E>LEN T$ THEN  LET E=LEN
T$
 227 RETURN
 228 LET A=1
 229 LET E=LEN T$
 230 RETURN

 260 REM >WAHL DES FORMATS <<

 261 CLS
 262 PRINT "FORMAT EINGEBEN:",,,
"1=LINKSBUENDIG",,"2=BLOCKSATZ"
 263 INPUT Z
 264 PRINT ,,"NUN DIE DRUCKBREIT
E","(NL=STANDARD VON 32 Z/ZEILE)
"
 265 PRINT
 266 INPUT B$
 267 IF B$="" THEN  LET L=32
 268 IF B$<>"" THEN  LET L=VAL B
$
 269 PRINT "---";("BLOCKSATZ" AN
D Z=2);("FLATTERSATZ" AND Z=1);"
----": PRINT "DRUCKBREITE:";L;"
ZEICHEN/ZEILE",,,

 270 REM *** FORMATIERUNG AUF **
        ****BLOCK/FLATTERSATZ**

 271 LET S=0
 272 LET B$=""

 273 FOR M=A TO E
 274 LET B$=B$+T$(M)
 276 IF LEN B$>L THEN  GO TO 279
 277 IF T$(M)=" " OR T$(M)="." O
R T$(M)="," OR T$(M)=":" THEN  L
ET S=LEN B$
 278 GO TO 286
 279 IF Z=2 THEN  GO TO 290
 284 PRINT B$( TO S)
 285 LET B$=B$(S+1 TO )
 286 NEXT M
 287 PRINT B$
 288 IF INKEY$="" THEN  GO TO 28
8
 289 RETURN

 290 REM >> SPREIZUNG <<

 291 LET C$=B$( TO S-(B$(S)=" ")
)
 292 FOR N=1 TO LEN C$
 293 IF LEN C$=L THEN  GO TO 300
 294 IF C$(N)="." OR C$(N)=" " O
R C$(N)="," OR C$(N)=":" THEN  G
O TO 296
 295 GO TO 298
 296 LET C$=C$( TO N)+" "+C$(N+1
TO )
 297 LET N=N+1
 298 NEXT N
 299 IF C$<>B$( TO S-(B$(S)=" ")
) THEN  GO TO 292
 300 PRINT C$
 302 GO TO 285
```

```
  10 REM START DES PROGRAMMS
        V E R M O E G E N
  <ZUM HAUPTMENAU MIT GOTO 300>
  20 GO TO 300
 100 REM MONATSEINGABEN
 102 REM KUMULIEREN GOLDMENGE
 104 LET Z=0
 106 FOR N=1 TO M
 107 LET Z=Z+G(N,2)
 108 NEXT N
 109 RETURN
 110 REM RENTEN/AKTIEN SUMME
        LETZTE EINZELWERTE
 112 LET Z=0
 114 FOR N=1 TO 10
 116 LET Z=Z+A(B,N,3)
 118 NEXT N
 119 RETURN
 120 REM SONST/SPARANLAGEN SUMME
        EINZELWERTE
 122 LET Z=0
 124 FOR N=1 TO 10
 126 LET Z=Z+W(B-3,N)
 128 NEXT N
 129 RETURN
 130 REM MONATSSUMMEN ALLE ARTEN
 132 LET Z(M)=Y(M)+V(M)+X(M)+O(M
)+T(M)+G(M,4)
 134 IF Z(M)<MIN THEN  LET MIN=Z
(M)
 136 IF Z(M)>MAX THEN  LET MAX=Z
(M)
 138 RETURN
 199 REM
  -------M-O-D-U-L-E------------
 200 REM INITIALISIEREN
 220 DIM N$(7,13)
 222 LET M$="RENTEN        AKTIEN
      GOLD          SPAREINLAGEN
 SONST.ANLAGENBARGELD      ENDE
EINGABEN"
 224 FOR N=1 TO 7
 226 LET N$(N)=M$( TO 13)
 228 LET M$=M$(14 TO )
 229 NEXT N
 230 LET M$="JANFEBMRZAPRMAIJUNJ
ULAUGSEPOKTNOVDEZ"
 240 DIM G(12,4)
 245 DIM A$(2,10,10)
 250 DIM A(2,10,3)
 255 DIM T(12)
 260 DIM O(12)
 265 DIM S$(2,10,12)
 270 DIM W(2,10)
 275 DIM X(12)
 280 DIM V(12)
 285 DIM Y(12)
 290 DIM Z(12)
 292 LET MIN=999999
 294 LET MAX=0
 300 REM HAUPTPROGRAMM
 305 CLS
 310 PRINT "  VERMOEGEN",,"0=INI
TIALISIEREN","1=MONATSEINGABEN",
"2=AKT.STATUS(LTZ.MONAT)","3=ENT
WICKLUNG(DIAGRAMM)","4=SPEICHERN
","  (NL=STOP)"
 315 INPUT E$
 320 IF E$="" THEN  STOP
 325 IF E$<"0" OR E$>"4" THEN  G
O TO 315
 330 LET A=VAL E$
 335 IF A THEN  GO TO 1000*A
 340 GO TO 200
1000 REM MONATSEINGABEN
1010 PRINT "MONAT ?"
1020 INPUT M
1030 IF M<0 OR M>12 THEN  GO TO
1020
1040 CLS
1045 PRINT " MONATSEINGABEN ";M$
(3*(M-1)+1 TO M*3),,,
1050 FOR N=1 TO 7
1055 PRINT N;". ";N$(N)
1060 NEXT N
1065 INPUT B
1070 IF B<0 OR B>7 THEN  GO TO 1
065
1071 IF B=7 THEN  GO TO 1080
1073 CLS
1075 PRINT N$(B);"MONAT ";M$(3*(
M-1)+1 TO M*3)
1077 GO SUB 1000+100*B
1078 GO TO 1040
1080 IF B=7 THEN  GO SUB 130
1085 GO TO 300
1100 REM MTS-EINGABEN RENTEN
1200 REM  BZW. AKTIEN
1214 PRINT AT 20,0;"NAMENSAEND.M
IT*,UBERGEHEN<>*";AT 21,0;"BLAET
TERN MIT NL";AT 2,0;"NR NAME/BEZ
   ANZ KURS WERT","-------------
--------------",,,
1220 FOR N=1 TO 10
1222 PRINT
1225 PRINT N;" ";A$(B,N);"    ";
1230 INPUT E$
1235 IF E$="" THEN  GO TO 1265
1238 IF E$(1)<>"*" THEN  GO TO 1
245
1240 LET A$(B,N)=E$(2 TO )
1244 GO TO 1222
1245 INPUT A(B,N,1)
1250 PRINT A(B,N,1);"  ";
1252 INPUT A(B,N,2)
1254 PRINT A(B,N,2);"  ";
1256 LET A(B,N,3)=A(B,N,1)*A(B,N
,2)
1260 PRINT A(B,N,3)
1265 NEXT N
1270 GO SUB 110
```



```
1272 PRINT ,"GESAMT DM",Z
1274 PAUSE 4E4
1275 IF B=1 THEN  LET T(M)=Z
1280 IF B=2 THEN  LET O(M)=Z
1290 RETURN
1300 REM GOLD MONATL.
1310 PRINT AT 21,0;"(NL = KEIN I
NPUT )";AT 2,0;"KURS MTS.BEST.+-
 GES.BEST. WERT","--------------
----------------",,,
1320 INPUT E$
1330 IF E$<>"" THEN  LET G(M,1)=
VAL E$
1335 PRINT " ";G(M,1);"     ";
1340 INPUT E$
1345 IF E$<>"" THEN  LET G(M,2)=
G(M,2)+VAL E$
1350 PRINT G(M,2);"       ";
1355 GO SUB 100
1360 PRINT Z;"       ";
1365 LET G(M,3)=Z
1370 LET G(M,4)=Z*G(M,1)
1375 PRINT G(M,4)
1380 PAUSE 4E4
1390 RETURN
1400 REM SPAREINLAGEN
1500 REM BZW SONSTIGE ANLAGEN
1510 PRINT AT 21,0;"BEZ.AENDERUN
G MIT*,NL=NO INPUT ";AT 2,0;"NR
KONTO/BEZ FAELLIGK. WERT","-----
-------------------------"
1520 FOR N=1 TO 10
1525 PRINT N;".  ";S$(B-3,N, TO
8);"  ";S$(B-3,N,9 TO );"    ";
1530 INPUT E$
1535 IF E$="" THEN  GO TO 1560
1540 IF E$(1)="*" THEN  LET S$(B
-3,N)=E$(2 TO )
1556 INPUT W(B-3,N)
1560 PRINT W(B-3,N)
1562 PRINT "***EINGABE OK ?=>NL,
SONST NEU**"
1564 INPUT E$
1565 IF E$<>"" THEN  GO TO 1525
1567 NEXT N
1570 GO SUB 120
1575 PRINT ,,,"GESAMT ";Z
1580 IF B=4 THEN  LET X(M)=Z
1585 IF B=5 THEN  LET V(M)=Z
1588 PAUSE 4E4
1590 RETURN
1600 REM BARGELD
1610 PRINT "MONATSBESTAND,0= NL"
,"-------------------------"
1620 INPUT E$
1630 IF E$="" THEN  GO TO 1650
1640 LET Y(M)=Y(M)+VAL E$
1650 PRINT "BESTAND ";Y(M)
1660 PAUSE 4E4
1670 RETURN
2000 REM STAND LETZTER MONAT
2100 CLS
2110 PRINT "WELCHER MONAT ?"
2120 INPUT M
2130 IF M<0 OR M>12 THEN  GO TO
2120
2140 PRINT AT 0,0;"AKTUELLER STA
TUS ";M$(3*(M-1)+1 TO M*3),"----
----------------",,,"1=UEBERSICH
T GES.","2=DETAILINFOS",,"(NL=EN
DE)"
2150 INPUT E$
2155 IF E$="" THEN  GO TO 300
2160 IF E$="2" THEN  GO TO 2200
2165 FOR N=1 TO 6
2170 PRINT AT N+2,0;N$(N),STR$ T
(M) AND N=1;STR$ O(M) AND N=2;ST
R$ G(M,4) AND N=3;STR$ X(M) AND
N=4;STR$ V(M) AND N=5;STR$ Y(M)
AND N=6
2180 NEXT N
2185 PRINT ,,"GESAMT",Z(M)
2190 PAUSE 4E4
2195 GO TO 2000
2200 REM DETAILINFOS
2205 CLS
2210 PRINT "DETAILS AKT.STATUS "
;M$(3*(M-1)+1 TO M*3),"---------
-------------";AT 21,0;"0-7 WAEH
LEN "
2215 PRINT AT 3,0;"0=ALLE ARTEN"
2220 FOR N=1 TO 7
2230 PRINT AT N+3,0;N;"-";N$(N)
2240 NEXT N
2250 INPUT N
2252 PRINT AT N+3,16;"<=GEWAEHLT
"
2253 PAUSE 40
2254 IF N>6 THEN  GO TO 300
2256 IF NOT N THEN  GO TO 2270
2260 GO SUB 2280
2262 PAUSE 4E4
2264 GO TO 2200
2270 FOR N=1 TO 6
2273 GO SUB 2280
2274 IF INKEY$="" THEN  GO TO 22
74
2276 NEXT N
2278 GO TO 300
2280 CLS
2282 PRINT N$(N);" AKTUELL",,,
2284 GO SUB 2200+100*N
2286 RETURN
2300 REM RENTEN  UND
2400 REM AKTIEN LETZTER MONAT
2410 PRINT "NR NAME/BEZ  ANZ KUR
S  WERT","----------------------
------"
2420 FOR Y=1 TO 10
```

```
2430 IF A$(N,Y,1)<>" " THEN  PRI
NT Y;")";A$(N,Y);TAB 13;A(N,Y,1)
;TAB 17;A(N,Y,2);TAB 23;A(N,Y,3)
2440 NEXT Y
2450 PRINT ,,"GESAMTBETRAG-";STR
$ T(M) AND N=1;STR$ O(M) AND N=2
2455 PAUSE 4E4
2458 RETURN
2500 REM GOLD AKTUELL
2510 PRINT "KURS MTS- GES.MENGE
WERT","-------------------------
"
2520 FOR Y=1 TO 4
2530 PRINT TAB (Y-1)*6;G(M,Y);
2540 NEXT Y
2550 IF INKEY$="" THEN  GO TO 25
50
2555 RETURN
2600 REM SPAR- UND
2700 REM SONST.ANLAGEN LTZ MONAT
2710 PRINT "NR KONTO/BEZ FAELLIG
K WERT","-------------------------
---"
2720 FOR Y=1 TO 10
2730 IF S$(N-3,Y,1)<>" " THEN  P
RINT Y;"  ";S$(N-3,Y, TO 8);"
  ";S$(N-3,Y,9 TO );"  ";W(N-3,Y
)
2740 NEXT Y
2750 PRINT ,,"GESAMTBETRAG ";STR
$ X(M) AND N=4;STR$ V(M) AND N=5
2760 IF INKEY$="" THEN  GO TO 27
60
2770 RETURN
2800 REM BARGELD
2810 PRINT "------------------","
DERZ.BESTAND ";Y(M)
2820 IF INKEY$="" THEN  GO TO 28
20
2830 RETURN
3000 REM GES.VERMOEGEN(DIAGRAMM)
3100 CLS
3110 PRINT "        VERMOEGENSENT
WICKLUNG"
3115 REM KOORDINATEN
3135 PRINT AT 20,6;"
"

3136 PRINT AT 21,0;"MONAT  1 2 3
 4 5 6 7 8 9101112"
3150 FOR N=1 TO 20
3152 PRINT AT N,5;"■"
3155 NEXT N
3160 REM SKALIERUNG
3162 LET X=MAX-MIN
3164 LET F=19/X
3165 LET X=10^INT (LN X/LN 10)
3168 LET Y=X*INT ((MIN+X)/X)
3170 FOR N=Y TO MAX STEP X
3172 LET E$=STR$ N
3175 IF LEN E$)3 THEN  LET E$=E$
( TO LEN E$-3)+"TS"
3176 LET X=19-(N-MIN)*F
3177 PRINT AT X+.5,0;E$;AT X+.5,
5;"—"
3178 NEXT N
3180 REM PLOT
3182 LET F=38/(MAX-MIN)
3184 FOR N=1 TO 12
3186 IF Z(N) THEN  PLOT 10+N*4,(
Z(N)-MIN)*F+3.5
3188 NEXT N
3190 PAUSE 4E4
3192 GO TO 300
4000 REM SPEICHERN
4100 CLS
4200 LET E$="RECORDER STARTEN (W
EITER MIT NL)"
4300 PRINT AT 10,0;E$
4400 FOR N=1 TO 17
4500 LET E$(N)=CHR$ (CODE E$(N)+
128-256*(E$(17)<>" "))
4600 NEXT N
4700 IF INKEY$="" THEN  GO TO 43
00
4750 SAVE "VERM"
4800 GO TO 300
```



```
  5 REM **********************
      *  P R O G R A M M    *
      *  A K T I E N        *
      *VERWALTUNG VON WOCHEN*
      *EINGABEN,WERTBERECH- *
      *UND PORTEFEUILLE-    *
      *VERWALTUNG(MAX.10 TI-*
      *TEL+INDEX=DOW JONES) *
      **********************
 10 GO TO 990
 149 REM WERT=KURS*MENGE
 150 LET X= INT A(N,W)
 152 LET F=INT (.1+(A(N,W)-X)*1E
5)
 154 LET F=X*F
 155 RETURN
 160 REM SAEULENGRAFIK PORTEFEUI
LLE
 161 IF NOT A(N,W) THEN  RETURN
 162 GO SUB 150
 163 LET F=F/Y+.5
 164 LET X=N*2+3
 165 LET F=20-F/5
 166 FOR M=F TO 19.5
 167 PRINT AT M,X;"■"
 168 NEXT M
 169 RETURN
 170 REM AKTIENLISTE
 171 PRINT N$(7);N$(2),"---------
--------"
 173 FOR N=1 TO NR
 174 IF A$(N,1)<>" " THEN  PRINT
N;"-";A$(N)
 175 NEXT N
 176 PRINT "NR.WAEHLEN (NL= RETU
RN )"
 177 INPUT I$
 178 IF I$="" THEN  RETURN
 179 IF I$<"1" OR VAL I$>NR THEN
  GO TO 176
 180 LET NR=VAL I$
 181 RETURN
 184 REM MINIMAX-AUSTAUSCH
 185 IF NOT F THEN  RETURN
 186 IF MIN>F THEN  LET MIN=F
 187 IF MAX<F THEN  LET MAX=F
 188 LET M(NR)=MIN+MAX*1E-5
 189 RETURN
 200 REM UPRO DIA
 205 CLS
 210 PRINT AT 0,6;N$(6);A$(NR)
 214 REM SKALIEREN
 215 PRINT AT 20,5;"■"
 216 FOR N=1 TO 19
 217 PRINT AT N,5;"▌"
 218 NEXT N
 219 GO SUB 260
 220 IF X THEN  GO TO 223
 221 PRINT AT 10,8;"NO  PRINT (M
INIMAX=0)"
 222 GO TO 258
 223 LET F=19/X
 224 LET X=10^INT (LN X/LN 10)
 226 LET Y=X*INT ((MIN+X)/X)
 228 FOR N=Y TO MAX STEP X
 230 LET I$=STR$ (N/(1000 OR N<1
000))
 231 IF LEN I$)5 THEN  LET I$=I$
( TO 5)
 232 IF LEN I$<4 THEN  LET I$=I$
+("TS" AND N>=1000)
 235 LET X=19-(N-MIN)*F
 236 PRINT AT X+.5,0;I$;AT X+.5,
5;"■"
 237 NEXT N
 239 REM WOCHENACHSE
 240 FOR N=1 TO 13
 242 PRINT AT 20,4+N*2;"┳";AT
21,4+N*2;N*4
 244 NEXT N
 245 PRINT AT 20,0;"WOCHE"
 249 REM WERTEPLOT
 250 LET F=37/(MAX-MIN)
 252 FOR N=1 TO 52
 254 IF A(NR,N) THEN  PLOT 10+N*
4,(A(NR,N)-MIN)*F+3.5
 256 NEXT N
 258 RETURN
 260 REM MINIMAX
 262 LET MIN=INT M(NR)
 264 LET MAX=INT (.1+(M(NR)-MIN)
*1E5)
 266 LET X=MAX-MIN
 268 RETURN
 269 REM EINZELWERT AKTIE
 270 LET MIN=999999
 271 LET MAX=0
 272 FOR W=1 TO 52
 274 GO SUB 150
 277 LET A(13,W)=F
 279 IF F THEN  GO SUB 185
 282 NEXT W
 284 LET M(13)=MIN+MAX*1E-5
 286 RETURN
 289 REM  INPUT CHECK WOCHE
 290 INPUT I$
 291 IF I$="" THEN  RETURN
 292 IF I$="V" OR I$="N" THEN  G
O TO 295
 293 IF (LEN I$=2 AND (I$(1)<"1"
 OR I$(1)>"5")) OR (I$(LEN I$)<"
0" OR I$(LEN I$)>"9") THEN  GO T
O 299
 294 LET W=VAL I$
 295 LET W=W-(I$="V")+(I$="N")
 296 LET W=W OR (W<1)
 297 IF W>52 THEN  LET W=52
 298 IF W THEN  RETURN
 299 PRINT N$(11);"FALSCH =>";I$
```

```
 300 GO TO 290
 310 REM MINIMAX CHECK
 312 GO SUB 260
 314 IF INT (A(NR,W))<MIN THEN
LET MIN=INT (A(NR,W))
 316 IF INT (A(NR,W))>MAX THEN
LET MAX=INT (A(NR,W))
 318 LET M(NR)=MIN+MAX*1E-5
 319 RETURN
 320 REM FORTSCHREIBEN GESAMTWER
TE (PORTEFEUILLE)
 321 LET NR=11
 322 LET A(NR,W)=0
 323 FOR N=1 TO 10
 325 GO SUB 150
 327 LET A(NR,W)=A(NR,W)+F
 329 NEXT N
 330 GO SUB 260
 331 LET F=A(NR,W)
 332 GO SUB 185
 335 RETURN
 900 REM INITIALISIERUNG
 905 LET W=52
 920 DIM A(13,W)
 930 DIM M(13)
 931 FOR N=1 TO 13
 932 LET M(N)=999999
 936 NEXT N
 940 DIM A$(13,13)
 950 DIM N$(15,11)
 951 REM  NEW S-TABELLE FUELLEN
          + MINI/MAX-TAB.BESETZE
N
 952 LET I$="UPDATING*AKTIEN*DOW
 JONES*ENTWICKLUNG*G E S A M T*D
IAGRAMM*LISTE DER*EINGABE*+LOESC
HUNG*ABFRAGE*WOCHE(N)*AUFTEILUNG
*W E R T*EINZELN*A L L E*"
 954 LET W=1
 956 FOR N=1 TO 12
 957 IF I$(N)<>"*" THEN  NEXT N
 958 LET N$(W)=I$( TO N-1)
 960 LET I$=I$(N+1 TO )
 962 LET W=W+1
 966 IF I$<>"" THEN  GO TO 956
 967 LET A$(11)="PORTEFEUILLE "
 968 LET A$(12)=N$(3)
 969 LET A$(13)=N$(13)
 990 REM H A U P P R O G R A M M
 991 CLS
 992 PRINT " HAUPTMENUE AKTIEN",
,,"1";N$(1);N$(2),"2";N$(1);N$(3
),"3";N$(6);N$(5),,"+ ";N$(14),"
4 SAVE ",,"5INITIALISIEREN",,"(N
L= STOP )"
 993 INPUT I$
 994 IF I$="" THEN  STOP
 995 IF I$>"5" OR I$<"1" THEN  G
O TO 993
 996 IF I$="5" THEN  GO TO 900
 997 GO SUB 1000*VAL I$
 998 GO TO 991
 999 REM UP DATE AKTIEN
1000 CLS
1010 PRINT N$(1);N$(2),"--------
----------","1NAMENS";N$(8),N$(9)
,,"2";N$(10);"+";N$(1),"3";A$(11
);N$(6)," IN O/O JE ";N$(11),"(N
L=> RETURN )"
1020 INPUT E$
1030 IF E$="" THEN  RETURN
1050 IF E$<"1" OR E$>"3" THEN  G
O TO 1020
1060 GO SUB 1000+100*VAL E$
1065 IF E$<>"2" THEN  RETURN
1070 PRINT AT 20,0;" NOTAUSSTIEG
? *=>KEINE GESAMTWERT-FORTSCHREI
BUNG*"
1080 INPUT I$
1085 IF I$<>"*" THEN  GO SUB 320
1090 RETURN
1099 REM AKTIENNAMEN
1100 CLS
1105 PRINT N$(1);"NAMEN","------
---------------"
1110 LET NR=10
1120 GO SUB 170
1125 PRINT " - =>NAMENSLOESCHUNG
","...=>BEZEICHNUNG AENDERN"," +
 =>WERTELOESCHUNG","(NL= RETURN
)"
1130 INPUT I$
1135 IF I$="" THEN  RETURN
1140 IF I$(1)="+" THEN  GO TO 11
60
1145 IF I$(1)="-" THEN  LET I$="
"
1150 LET A$(NR)=I$
1155 RETURN
1159 REM *WERTELOESCHUNG**
1160 CLS
1162 PRINT N$(9);A$(NR),,,"1 ";N
$(11);N$(13),"2 BEREINIGUNG ";A$
(11),"(NL= RETURN )"
1164 INPUT I$
1166 IF I$="" THEN  RETURN
1168 LET A$(NR)=""
1170 LET M(NR)=999999
1171 LET N=NR
1172 LET NR=11
1173 IF I$="1" THEN  GO TO 1176
1174 LET M(11)=999999
1175 GO SUB 260
1176 FOR W=1 TO 52
1177 GO SUB 150
1178 LET A(N,W)=0
1180 IF I$="1" THEN  GO TO 1190
1182 LET A(NR,W)=A(NR,W)-F
1184 LET F=A(NR,W)
1186 GO SUB 185
1190 NEXT W
1192 RETURN
1199 REM AKTIEN-WOCHE
1200 CLS
1210 PRINT N$(1);N$(2);N$(14);AT
 1,0;"WOCHE ?"
1212 GO SUB 290
1214 PRINT AT 1,0;W;".WOCHE";AT
21,0;" ("" ""=> RETURN , NL=>UEB
ERLESEN)";AT 3,0;"NR NAME (*=NEU
)  KURS MENGE WERT----------
----------"
1225 FOR N=1 TO 10
1230 IF A$(N,1)=" " THEN  GO TO
1275
1235 PRINT N;TAB 3;A$(N);
1240 GO SUB 150
1245 PRINT TAB 17;X;
1250 INPUT I$
1252 IF I$=" " THEN  RETURN
1254 IF I$="" THEN  GO TO 1262
1255 LET A(N,W)=VAL I$+A(N,W)-X
1256 LET NR=N
1257 GO SUB 260
1258 LET F=INT A(N,W)
1259 GO SUB 185
1260 PRINT TAB 16;"*";
1261 GO TO 1240
1262 PRINT TAB 22;F/X;TAB 27;F
1264 INPUT I$
1266 IF I$=" " THEN  RETURN
1268 IF I$="" THEN  GO TO 1275
1270 LET A(N,W)=X+(VAL I$)*1E-5
1272 PRINT TAB 21;"*";
1273 GO SUB 150
1274 GO TO 1262
1275 NEXT N
1280 RETURN
1299 REM PORTEFEUILLE O/O-AUFTEI
LUNG
1300 CLS
1302 PRINT "O/O-AUFTEILUNG=>WOCH
E ?"
1304 GO SUB 290
1310 LET Y=A(11,W)/100
1312 PRINT AT 0,0;"100■   ";A$(
11);":";Y/10;" TS";AT 1,7;"AUFTE
ILUNG ";W;".WOCHE";AT 21,0;N$(2)
;AT 20,0;"O/O";AT 10,1;"50"
1313 IF NOT Y THEN  GO TO 1350
1315 FOR N=1 TO 10
1320 PRINT AT N*2,3;"■";AT 20,2
+N*2;"■";AT 21,3+N*2;N;AT 10,
2+N*2;"--"
1330 GO SUB 160
1340 NEXT N
1350 PRINT AT 2,7;("E N T F A E
L L T" AND NOT Y)
1355 PAUSE 4E4
1360 RETURN
2000 REM UPDATE DOW JONES
2010 CLS
2020 PRINT N$(3);N$(4),,,"1";N$(
10);N$(11),"2";N$(8);N$(3);N$(11
);"3";N$(9);N$(5),"(NL= RETURN )
"
2030 INPUT I$
2040 IF I$="" THEN  RETURN
2050 IF I$<"1" OR I$>"3" THEN  G
O TO 2030
2060 GO SUB 2000+100*VAL I$
2070 PAUSE 4E4
2080 GO TO 2010
2099 REM ABFRAGE DOW JONES-WOCHE

2100 CLS
2110 PRINT N$(3);N$(4),,,N$(11);
N$(13);"VERAEND.","--------------
---------------"
2120 PRINT AT 20,0;N$(11);N$(8);
"DANN=>"," V-ORWO./N-ACHWO.(NL=
RETURN )"
2130 PRINT AT 4,0;""
2140 GO SUB 290
2150 IF I$="" THEN  RETURN
2160 PRINT TAB 2;W;TAB 12;A(12,W
);TAB 24;A(12,W)-A(12,W-(W>1))
2170 GO TO 2140
2199 REM  INPUT DOW JONES
2200 CLS
2205 LET NR=12
2210 PRINT N$(8);N$(3),N$(11);N$
(13),"------------------   (NL=E
NDE)"
2220 GO SUB 290
2230 IF I$="" THEN  RETURN
2250 PRINT TAB 2;W;
2255 IF A(NR,W) THEN  PRINT TAB
12;A(NR,W);TAB 20;"<=VORHANDEN",
N$(13);"UEBERSCHREIBEN ?","(NO=N
L)"
2260 INPUT I$
2265 IF I$="" THEN  GO TO 2220
2270 LET A(NR,W)=VAL I$
2275 PRINT TAB 12;A(NR,W)
2280 GO SUB 310
2285 GO TO 2220
2300 REM DOW JONES-LOESCHUNG
2310 PRINT "L O E S C H U N G",N
$(3);N$(15);N$(11);"  O K ? (YES
=1,SONST NL)"
2320 INPUT I$
2330 IF I$<>"1" THEN  RETURN
2340 LET N=12
2350 LET M(N)=999999
2360 GO SUB 1176
2370 PRINT N$(9);"BEENDET"
2380 RETURN
3000 REM GESAMTENTWICKLUNG
```

```
3010 CLS
3020 PRINT N$(6),,,,"1";N$(3),,"
2";N$(2);N$(14),"3";N$(2);N$(5);
" ";N$(13),,"(NL= RETURN )"
3030 INPUT I$
3040 IF I$="" THEN  RETURN
3050 IF I$<"1" OR I$>"3" THEN  G
0 TO 3030
3060 GO SUB 3000+100*VAL I$
3065 PAUSE 4E4
3070 GO TO 3010
3100 REM DOW JONES DIA
3110 LET NR=12
3120 GO SUB 200
3150 RETURN
3199 REM DIA EINZELAKTIE
3200 CLS
3205 LET NR=10
3210 PRINT N$(2);N$(4)
3215 GO SUB 170
3260 PRINT ,,"KURS(=NL) ODER ";N
$(13),"(<>NL) ?"
3265 INPUT I$
3270 IF I$="" THEN  GO TO 3120
3272 REM EINZELWERT-DIA
3275 CLS
3277 PRINT N$(13);A$(NR)
3278 LET N=NR
3279 GO SUB 270
3281 LET NR=13
3283 GO SUB 216
3285 RETURN
3300 REM GESWERT.DIA
3310 LET NR=11
3320 GO TO 3120
4000 CLS
4100 PRINT AT 8,12;" SAVE "
4110 PRINT AT 10,0;"RECORDER STA
RTEN+TASTE DRUECKEN"
4115 PAUSE 40
4120 IF INKEY$<>"" THEN  GO TO 4
140
4130 PRINT AT 10,0;"
                    "
4133 PAUSE 40
4135 GO TO 4110
4140 SAVE "AKT"
4150 GO TO 990
```

```
  10 REM ************************
         *PROGR.PROJECT-CONTROL*
         *--------------------*
         *MAX.25 ARBEITSPAKETE *
         *TERMINSORT BEG.DATUM *
         *(NO CHECK),BALKENDIA *
         *NACH PROJ.WO.,LISTEN *
         *NACH APNR(FREIE VERGA*
         *BE VOM USER 1-25)    *
         ************************

  15 GO TO 90
  18 REM U P R O S
  19 REM AUFBAU AP-DRUCKMASKE UN
D LADEN POSIT.TAB W(VK=ZEILE,NK=
SP)
  20 CLS
  21 PRINT "PROJ.NR:";A$(1);" TI
TEL:";B$(1),"--------------------
-----------",,,"AP-NR:";X
  22 LET I$="TITEL:**$FIRMA:**$B
EG.:*END:*NEU:**$PROJ.WOCHE B*E*N
**MST1 *DAT:* ERL:**MST2 *    *
    **MST3 *    *     **MST4 *
  *     **$DM GES:*(P:*)S:**DM A
BGEF.:*$DM REST:*$$BEM:*$"
  23 REM  DIM POS.ZAEHLER/SEPARA
TOR
  24 LET N=26
  25 LET E$="......"
  26 LET Z=4
  27 DIM W(N)
  28 LET M=1
  29 LET S=0
  30 REM MASKEN-GENERATOR
  31 FOR N=1 TO LEN I$
  32 IF I$(N)<>"*" THEN  GO TO 3
8
  33 LET W(M)=Z+S*1E-2
  34 LET M=M+1
  35 LET S=S+LEN E$
  36 PRINT E$;
  37 GO TO 45
  38 IF I$(N)<>"$" THEN  GO TO 4
3
  39 LET S=0
  40 LET Z=Z+1
  41 PRINT
  42 GO TO 45
  43 PRINT I$(N);
  44 LET S=S+1
  45 NEXT N
  46 LET N=M-1
  47 RETURN
  49 LET N=26
  50 FOR N=1 TO N
  51 LET I$=STR$ N
```



```
  52 LET I$( TO LEN I$)=CHR$ (12
8+CODE I$(1))+CHR$ (128+CODE I$(
LEN I$))
  53 PRINT AT (INT W(N)),(W(N)-I
NT (W(N)))*1E2;I$
  54 NEXT N
  55 RETURN
  56 REM WERTEINITIALISIERUNG
  57 LET X=25
  58 DIM Y$(X,4,6)
  59 DIM X$(X,3,25)
  60 DIM Y(X,5,2)
  61 DIM X(X,3,3)
  62 LET C$="0912151801022610111
31416171920242503040506070821222
3"
  63 FOR N=0 TO 25
  64 LET C$(N+1)=CHR$ VAL C$(1+N
*2 TO 2+N*2)
  65 NEXT N
  66 DIM A$(1,3)
  67 DIM B$(1,13)
  80 LET Z=0
  81 LET I$="WELCOME TO$  P R O
J E C T  C O N T R O L$$  TITEL
*B.13 STELLEN$  PROJ.NR. *3ST$$
 (BITTE INPUT )"
  82 LET E$=">"
  83 LET N=2
  84 GO SUB 27
  87 INPUT B$(1)
  88 INPUT A$(1)
  89 REM HP - STEUERUNG
  90 CLS
  91 PRINT "  PC-HAUPTMENUE ","
 -------------- ",,,"0 INITIALIS
IEREN"
  92 LET E$=""
  93 LET N=10
  94 LET I$="*  INPUT$*  AP-DATE
N$*  TERMINLISTE$*  BALKENPLAN$*
  KOSTEN$*  AP-LISTE$*  FIRMEN$*
  MST-LISTE$*  SAVE$*   STOP"
  95 GO SUB 26
  96 GO SUB 50
  97 INPUT I$
  98 IF I$<"0" OR LEN I$>1 AND I
$>"10" THEN  GO TO 90
  99 IF I$="0" THEN  GO TO 56
 100 IF I$="10" THEN  STOP
 101 GO TO (2+100*VAL I$)
 102 REM  INPUT
 103 PRINT AT 20,0;" INPUT AP-NR
 VORGEBEN;LETZTE=";X,"OK=>NL,SON
ST=>NR("" ""=RETURN)"
 104 INPUT I$
 105 IF I$="" THEN  GO TO 109
 106 IF I$=" " THEN  GO TO 90
 107 LET X=VAL I$
 108 IF X<0 OR X>25 THEN  GO TO
104
 109 GO SUB 20
 110 LET I$=""
 111 LET NR=0
 112 IF X$(X,1)=" " THEN  GO TO
120
 116 GO SUB 140
 117 GO SUB 188
 118 PAUSE 4E4
 120 IF I$="" THEN  GO SUB 49
 122 PRINT AT 20,0;"AEND.NR EING
EBEN,DANN DATEN - INPUT(0=NUR PR
INT,NL=>RETURN)"
 124 INPUT I$
 126 IF I$="" THEN  GO TO 103
 128 LET NR=VAL I$
 129 IF NR AND NR>26 THEN  GO TO
 122
 130 IF NR=25 THEN  PRINT AT 20,
0;"POSITION WIRD ERRECHNET ;KEIN
"
 132 INPUT I$
 134 GO TO 112
 138 REM SELEKTIONSPRINT
 140 LET A=0
 142 FOR N=1 TO 4
 144 GO SUB 182
 145 IF NR=B AND I$<>"" THEN  LE
T Y$(X,N)=I$
 146 PRINT AT Z,S;Y$(X,N)
 147 NEXT N
 152 FOR N=1 TO 3
 154 GO SUB 182
 155 IF NR=B AND I$<>"" THEN  LE
T X$(X,N)=I$
 156 PRINT AT Z,S;X$(X,N)
 158 NEXT N
 161 LET C=A=A
 162 FOR N=1 TO 5
 163 LET C=C=0
 164 GO SUB 182
 165 IF NR=B AND I$<>"" THEN  LE
T Y(X,N,1+C)=VAL I$
 166 PRINT AT Z,S;"......";AT Z,
S;Y(X,N,1+C)
 168 IF NOT C THEN  GO TO 163
 169 NEXT N
 173 FOR N=1 TO 3
 174 FOR M=1 TO 3
 175 GO SUB 182
 176 IF NR=B AND I$<>"" THEN  LE
T X(X,N,M)=VAL I$
 177 PRINT AT Z,S;"......";AT Z,
S;X(X,N,M)
 178 NEXT M
 179 NEXT N
 180 RETURN
 181 REM PRINTPOS.
 182 LET A=A+1
 183 LET B=CODE C$(A)
 184 LET Z=W(B)
 185 LET S=(Z-INT Z)*1E2
```

```
 186 RETURN
 187 REM ERRECHNEN "RESTSU."
 188 LET Y(X,5,2)=X(X,3,1)-Y(X,5
,1)
 189 LET A=17
 190 GO SUB 183
 191 PRINT AT Z,S;".....";AT Z,
S;Y(X,5,2)
 192 RETURN
 200 REM AP-LESEN
 202 PRINT AT 20,0;"LESENAP-NR V
ORGEBEN;LETZTE=";X,"OK=>NL,SONST
 NR("" ""=RETURN)"
 203 INPUT I$
 205 IF I$=" " THEN  GO TO 90
 210 IF I$<>"" THEN  LET X=VAL I
$
 215 GO SUB 20
 220 LET NR=0
 225 IF X$(X,1,1)<>" " THEN  GO
TO 240
 227 PRINT AT 20,0;"AP-NR ";X;"
L E E R"
 230 PAUSE 100
 235 GO TO 202
 240 GO SUB 140
 245 PAUSE 4E4
 250 GO TO 202
 300 REM TERMINLISTE
 302 CLS
 306 PRINT "  TERMINLISTE (SORTI
ERT)",,,"APNR BEGINN   ENDE   NE
U",,,
 308 GO SUB 920
 310 FOR N=1 TO M
 315 LET M=(W(N)-INT (W(N)))*1E2
 320 LET M=INT (M+.01)
 325 PRINT M;TAB 5;X(M,1,1);TAB
14;X(M,1,2);TAB 21;X(M,1,3)
 330 NEXT N
 335 PAUSE 4E4
 340 GO TO 90
 401 REM BALKENDIA
 402 CLS
 404 LET I$="$       B A L K E N
P L A N$$APNR    PROJ.WOCHEN$$
.*********"
 406 LET N=9
 408 LET E$="■"
 410 LET Z=0
 412 GO SUB 27
 414 LET I$="0 6 12182430364248"
 415 FOR N=1 TO 9
 416 PRINT AT 4,(W(N)-INT (W(N))
)*1E2;I$( TO 2);
 417 LET I$=I$(3 TO )
 418 NEXT N
 419 GO SUB 920
 420 PRINT AT 6,0;
 421 GO SUB 447
```



```
 422 LET E$=STR$ INT W(1)
 423 PRINT "  ..................
..........";TAB 3;E$(5 TO );TAB
 5;E$(3 TO 4);TAB 7;E$( TO 2);
 424 LET E$=STR$ INT W(M)
 425 PRINT TAB E/2;E$(5 TO );TAB
 E/2+2;E$(3 TO 4);TAB E/2+4;E$(
TO 2)
 431 PAUSE 4E4
 432 GO TO 90
 446 REM  PLOT  PRINT
 447 FOR N=1 TO M
 448 LET NR=(W(N)-INT (W(N)))*1E
2
 449 LET NR=INT (NR+.01)
 451 PRINT NR;TAB 3;".";
 452 LET A=X(NR,2,1)
 453 LET E=X(NR,2,2)
 454 IF A>E OR A<1 OR A>52 OR E<
1 OR E>52 THEN  GO TO 474
 455 FOR Z=A TO E
 456 LET GR=Z/2=INT (Z/2)
 457 LET E$="■"
 458 IF Z=A THEN  LET E$=(" ▛ A
ND GR)+("▜ " AND NOT GR)
 459 IF Z<>E THEN  GO TO 463
 460 IF NOT GR THEN  LET E$="▜ "
 462 GO TO 466
 463 IF NOT GR THEN  GO TO 468
 466 PRINT TAB INT (3.5+Z/2);E$;
 468 NEXT Z
 469 PRINT TAB 30;"."
 470 NEXT N
 473 RETURN
 474 PRINT "WO.EINTRAG FALSCH"
 475 NEXT N
 476 RETURN
 500 REM KOSTEN
 502 CLS
 504 PRINT "  KOSTENUEBERSICHT"
,,,"APNR GES.SU. ABGEF. IN O/O",
,,
 506 LET E=0
 508 LET A=E
 510 FOR N=1 TO 25
 515 IF NOT X(N,3,1) THEN  GO TO
530
 520 GO SUB 550
 525 LET E=E+C
 528 LET A=A+M
 530 NEXT N
 532 PRINT
 535 LET C=E
 537 LET M=A
 540 GO SUB 554
 545 PAUSE 4E4
 548 GO TO 90
 549 REM ARRAYZAHLEN + DRUCK-AUF
BEREITUNG
 550 LET C=X(N,3,1)
```



```
 552 LET M=Y(N,5,1)
 554 LET E$=STR$ (.005+M*1E2*C)*
1)
 556 IF LEN E$>5 THEN  LET E$=E$
( TO 5)
 558 PRINT N*(N<26);TAB 5;C;TAB
13;M;TAB 25-LEN E$;E$
 559 RETURN
 600 REM ARBEITSPAKET-
 700 REM UND FIRMENLISTE
 702 CLS
 704 PRINT "      ";("ARBEITSPAKE
T-" AND I$="6");("FIRMEN-" AND I
$="7");"LISTE",,,"APNR ";("FIRMA
" AND I$="7");("TITEL" AND I$="6
"),,,,
 708 LET M=VAL I$-5
 710 FOR N=1 TO 25
 715 IF X$(N,M,1)<>" " THEN  PRI
NT N;TAB 5;X$(N,M)
 720 NEXT N
 725 PAUSE 4E4
 730 GO TO 90
 800 REM MST-LISTE
 802 CLS
 804 PRINT "   MEILENSTEINLISTE"
,,,"APNR MST    DATUM  ERL.",,,
 806 FOR N=1 TO 25
 808 IF X$(N,1,1)=" " THEN  GO T
O 820
 812 FOR M=1 TO 4
 814 IF Y$(N,M,1)<>" " THEN  PRI
NT N;TAB 5;Y$(N,M);TAB 12;Y(N,M,
1);TAB 19;Y(N,M,2)
 816 NEXT M
 820 NEXT N
 822 PAUSE 4E4
 824 GO TO 90
 900 REM  SAVE
 902 CLS
 904 PRINT AT 10,5;"RECORDER STA
RTEN","     DANN NL DRUECKEN"
 906 INPUT I$
 908 SAVE "PC"
 910 GO TO 90
 919 REM TERMINDATENSORT
 920 DIM W(26)
 922 LET M=0
 923 FOR N=1 TO 25
 924 IF NOT X(N,1,1) THEN  GO TO
 930
 925 LET M=M+1
 926 LET E$=STR$ X(N,1,1)
 927 IF LEN E$=5 THEN  LET E$="0
"+E$
 928 LET E$=E$(5 TO )+E$(3 TO 4)
+E$( TO 2)
 929 LET W(M)=VAL E$+N*1E-2
 930 NEXT N
 932 FOR N=1 TO M-1
 934 FOR A=1 TO M-N
 936 IF W(A)>W(A+1) THEN  GO SUB
940
 937 NEXT A
 938 NEXT N
 939 RETURN
 940 LET W(26)=W(A)
 942 LET W(A)=W(A+1)
 944 LET W(A+1)=W(26)
 946 RETURN
```

```
   6 PRINT "START PROGRAMM",,"M
ITGLIEDERKARTEI","
*****************************"
,,,"BITTE TAGES";
   7 GO SUB 10
   8 GO TO 300
   9 REM DATUM IN
  10 PRINT "DATUM SO:01.08.84"
  11 INPUT D$
  12 IF LEN D$<>8 THEN  GO TO 11
  13 IF D$( TO 2)<"01" OR D$( TO
 2)>"31" OR D$(4 TO 5)<"01" OR D
$(4 TO 5)>"12" OR D$(7 TO )<"00"
 OR D$(7 TO )>"99" THEN  GO TO 1
1
  14 LET D$=CHR$ VAL D$(7 TO )+C
HR$ VAL D$(4 TO 5)+CHR$ VAL D$(
TO 2)
  15 RETURN
  16 REM DATUM OUT
  17 PRINT STR$ CODE D$(3);"."+S
TR$ CODE D$(2);".19";STR$ CODE D
$(1)
  18 RETURN
  19 FOR N=1 TO LEN A$
  20 LET B$=A$(N)
  21 GO SUB 28
  22 FOR M=1 TO LEN Z$-1
  23 IF CODE Z$(M)<>MNR THEN  NE
XT M
  24 LET Z$=Z$( TO M-1)+Z$(M+1 T
O )
  25 GO SUB 40+(VAL B$)*2
  26 NEXT N
  27 RETURN
  28 LET Z$=(K$ AND B$="0")+(F$
AND B$="1")+(H$ AND B$="2")+(V$
AND B$="3")+(T$ AND B$="4")+(S$
AND B$="5")+(L$ AND B$="6")+(G$
AND B$="7")
  29 RETURN
  30 REM  NEW S PRINT
  31 FOR N=1 TO 7
  32 LET A$=A$+STR$ N
  33 NEXT N
  34 IF A$(1)<>" " THEN  LET A$=
"0"+A$
  35 FOR N=2 TO LEN A$
  36 PRINT STR$ (N-1) AND A$(1)=
" ";"-";P$(VAL A$(N))
  37 NEXT N
  38 RETURN
  39 REM SPARTENSTR$  OUT
  40 LET K$=Z$
  41 RETURN
  42 LET F$=Z$
  43 RETURN
  44 LET H$=Z$
  45 RETURN
  46 LET V$=Z$
  47 RETURN
  48 LET T$=Z$
  49 RETURN
  50 LET S$=Z$
  51 RETURN
  52 LET L$=Z$
  53 RETURN
  54 LET G$=Z$
  55 RETURN
  56 REM DECODIEREN SPARTE
  57 LET A$="0" AND NOT S
  58 FOR N=7 TO 0 STEP -1
  59 IF (S-2^N)<0 THEN  GO TO 65
  60 LET S=S-2^N
  61 IF S<0 THEN  GO TO 65
  62 PRINT P$(N+1)
  63 LET A$=A$+STR$ (N+1)
  65 NEXT N
  66 RETURN
  69 REM CODIEREN SPARTE
  70 LET S=0
  71 PRINT "SPARTENWAHL(NL=ENDE/
KEINE SPARTE=0)"
  72 INPUT B$
  73 IF B$="" THEN  GO TO 82
  74 IF B$<"0" OR B$>"7" THEN  G
O TO 72
  75 LET S=S+INT (2^(VAL B$-1))
  76 GO SUB 28
  77 LET Z$=Z$+(CHR$ MNR AND MNR
)
  78 GO SUB 40+(VAL B$)*2
  79 PRINT "NOCH MEHR ?"
  80 GO TO 72
  82 RETURN
  84 REM STRINGSORT
  85 FOR N=1 TO LEN Z$
  86 FOR M=1 TO LEN Z$-N
  87 IF Z$(M)>Z$(M+1) THEN  GO S
UB 92
  88 NEXT M
  89 NEXT N
  90 RETURN
  91 REM ELEMENTAUSTAUSCH
  92 LET C$=Z$(M)
  93 LET Z$(M)=Z$(M+1)
  94 LET Z$(M+1)=C$
  95 RETURN
  99 REM WERTEINITIALISIERUNG
 100 DIM P$(20,10)
 101 LET MNR=1
 102 LET X$=""
 103 LET I$="FUSSBALL*HANDBALL*V
OLLEYBALL*TENNIS*SCHWIMMEN*L.ATH
LETIK*GYMNASTIK*STATISTIK*ALTER*
NEUAUFN.*VOLLMITGL.*ABGANG*PASSI
V*AKTIV*WEIBLICH*MAENNLICH*JUGEN
DL.*ERWACHSEN*AUSWERTUNG*OPTIONE
N:"
```



```
 104 LET A=1
 105 LET M=A
 106 FOR N=A TO LEN I$
 107 IF I$(N)<>"*" THEN  NEXT N
 108 LET P$(A)=I$(M TO N-1)
 109 LET A=A+1
 110 LET M=N+1
 111 NEXT N
 112 LET M$=""
 113 LET N$=""
 114 DIM N(5)
 115 LET A$=""
 116 LET POS=1
 117 LET Z$=""
 118 FOR N=0 TO 7
 119 GO SUB 40+N*2
 120 NEXT N
 121 DIM O$(100)
 122 FOR N=1 TO LEN O$
 123 LET O$(N)=CHR$ O
 124 NEXT N
 128 RETURN
 129 REM VAR MTGL.DATEN IN
 130 PRINT "VARIABLE NAMENSDATEN
","--------------------",,,"E I
N G A B E N :","(VORH.DATEN UEBE
RGEHEN M. NL)",,,
 131 LET B$="NAME  VORNA.STADT S
TR/NR"
 132 LET N(1)=0
 133 PRINT "GESCHL.(+=HERR,-FRAU
)"
 134 INPUT I$
 135 IF I$<>"" THEN  LET KZ=10*(
I$="-")
 136 PRINT "HERR" AND KZ<10;"FRA
U" AND KZ>=10
 137 FOR N=2 TO 5
 138 PRINT B$( TO 6);" : ";
 139 LET B$=B$(7 TO )
 140 INPUT I$
 141 IF I$="" THEN  GO TO 144
 142 LET X$=X$( TO N(1))+I$+X$(N
(1)+N(N)+1 TO )
 143 LET N(N)=LEN I$
 144 PRINT X$(N(1)+1 TO N(1)+N(N
))
 145 INPUT I$
 146 IF I$<>"" THEN  GO TO 142
 147 LET N(1)=N(1)+N(N)
 148 NEXT N
 149 RETURN
 150 CLS
 151 PRINT "FESTE MTG.STAMMDATEN
","--------------------",,,"(NL
=OK/WEITER;SONST INPUT :","+MAEN
NL/AKTIV,-WEIBL/PASSIV)",,,
 152 LET M=14*(KZ=0)+KZ
 153 LET KZ=KZ+4*(KZ=0)
 154 PRINT "WEIBL" AND KZ>6;"MAE
NNL" AND KZ<10,
 155 INPUT I$
 156 IF I$="" THEN  GO TO 159
 157 LET KZ=KZ-VAL (I$+"10")
 158 GO TO 154
 159 LET M=M-10*(M>6)
 160 PRINT P$(13) AND M<4;P$(14)
AND M>3,
 161 INPUT I$
 162 IF I$="" THEN  GO TO 166
 163 LET KZ=KZ+VAL (I$+"3")
 164 LET M=M+VAL (I$+"3")
 165 GO TO 160
 166 LET M=M-3*(M>3)
 167 LET KZ=KZ-M
 168 PRINT P$(M+9);" =>WEITER M.
<>NL"
 169 INPUT I$
 170 IF I$="" THEN  GO TO 173
 171 LET M=M*(M<3)+1
 172 GO TO 168
 173 LET KZ=KZ+M
 174 PRINT P$(9);">GEB.";
 175 LET B$=D$
 176 IF Y$(8)<>" " THEN  GO TO 1
79
 177 GO SUB 10
 178 LET Y$(7 TO 9)=D$
 179 LET D$=Y$(7 TO 9)
 180 GO SUB 17
 181 INPUT I$
 182 IF I$<>"" THEN  GO TO 177
 183 LET D$=B$
 184 IF S=128 THEN  GO TO 188
 185 PRINT "SPARTENAUSGABE ?(NO=
NL)"
 186 INPUT I$
 187 IF I$<>"" THEN  GO SUB 57
 188 PRINT "EINGABE SPORTART (NO
=NL)"
 189 INPUT I$
 190 IF I$="" AND S<128 THEN  RE
TURN
 191 IF S<128 THEN  GO SUB 19
 192 LET A$=" "
 193 GO SUB 30
 197 GO SUB 70
 198 LET Y$(10)=CHR$ S
 199 RETURN
 200 REM POSITIONIERUNG
 201 CLS
 203 PRINT "SUCHE M.NAME/MNR(END
E=NL)"
 204 INPUT I$
 205 IF I$="" THEN  GO TO 300
 206 FOR N=1 TO LEN I$
 207 IF I$(N)>="0" AND I$(N)<="9
" OR I$(N)>="A" AND I$(N)<="Z" T
HEN  GO TO 210
 208 PRINT "FALSCHER INPUT "
 209 GO TO 204
 210 NEXT N
```

```
 211 IF I$(1)>"9" THEN  GO TO 22
5
 212 LET MNR=VAL I$
 213 IF MNR=0 OR MNR>LEN O$ THEN
  GO TO 208
 214 IF CODE O$(MNR) THEN  GO TO
 217
 215 PRINT "MTGL. NICHT VORH."
 216 GO TO 202
 217 PRINT "MTGL.MIT ";MNR;" VOR
HANDEN"
 218 LET MST=CODE O$(MNR)
 219 LET POS=1
 220 FOR N=POS TO 10*(MST-1) STE
P 10
 221 LET POS=POS+CODE N$(N)
 222 NEXT N
 223 GO TO 236
 224 REM NAMENSSUCHE
 225 LET POS=1
 226 FOR N=POS TO LEN N$ STEP 10
 227 IF M$(POS TO POS+LEN I$-1)=
I$ THEN  GO TO 231
 228 LET POS=POS+CODE N$(N)
 229 NEXT N
 230 GO TO 215
 231 LET MST=1+INT (N/10)
 232 FOR M=1 TO LEN O$
 233 IF CODE O$(M)<>MST THEN  NE
XT M
 234 LET MNR=M
 235 PRINT "MTGL.MIT ";MNR;" VOR
HANDEN"
 236 LET POS=POS+(POS+CODE N$(N)
-1)*1E-4
 237 LET X$=M$(POS TO ((POS-INT
POS)+1E-6)*1E4)
 238 LET Y$=N$(1+(MST-1)*10 TO M
ST*10)
 239 RETURN
 240 REM AENDERN MIT/OHNE POSIT.
 241 CLS
 242 PRINT "U P D A T I N G","MI
TGLIED ";MNR,,,"A-ENDERN NACH PO
SITIONIEREN","P-OSITIONIEREN+AEN
DERN","L-OESCHEN",,"(NL=> RETURN
)"
 243 INPUT I$
 244 GO TO 245+55*(I$="")+35*(I$
="L")
 245 IF I$="P" THEN  GO SUB 200
 247 PRINT "UNTERMENUE AENDERN",
,,"1=NAME/ADR.(VARIBLER TEIL)","
2=STATUS/SPARTE(FESTER TEIL)","3
=ALLE MTGL.DATEN(CHECK MOEGL.)",
"(NL= RETURN )"
 248 INPUT I$
 249 IF I$<>"" THEN  GO SUB 370
 250 GO TO 300
 251 CLS
 252 PRINT "F O R T S C H R E I
B E N",,,"N-EUEINGABE",,"U-PDATI
NG",,"(NL= RETURN )"
 253 INPUT I$
 254 GO TO 300-45*(I$="N")-60*(I
$="U")
 255 PRINT ,,"MITGLIEDS-NR.:",,"
VORGABE=> INPUT ","NL=>PER PROGR
AMM"
 256 INPUT I$
 257 IF I$="" THEN  GO TO 260
 258 LET MNR=VAL I$
 259 IF NOT CODE O$(MNR) THEN  G
O TO 263
 260 FOR N=1 TO LEN O$
 261 IF CODE O$(N) THEN  NEXT N
 262 LET MNR=N
 263 LET MST=1+LEN N$/10
 264 LET O$(MNR)=CHR$ MST
 265 PRINT ,,"MITGLIED BEKOMMT N
R.=>";MNR
 266 DIM N(5)
 267 LET X$=""
 268 LET Y$="                "
 269 LET POS=LEN M$+1+(LEN M$+1)
*1E-4
 270 LET Y$(6)=CHR$ 0
 271 LET Y$(10)=CHR$ 128
 272 LET N$=N$+Y$
 273 LET A=3
 274 GO SUB 377
 275 GO TO 300
 279 REM LOESCHROUTINE
 280 CLS
 281 PRINT "  L O E S C H E N",,
,"1-GESAMTFILE",,"2-MTGL.SATZ",,
"(NL=>RETURN)"
 282 INPUT I$
 283 IF I$="" THEN  GO TO 300
 284 IF I$="1" THEN  GO SUB 100
 285 IF I$<>"2" THEN  GO TO 282
 286 PRINT "MTGL ";MNR;" WIRD GE
LOESCHT - OK ?","(NL=JA;SONST IN
PUT MNR/NAME)",,,
 287 INPUT I$
 288 IF I$="" THEN  LET I$=STR$
MNR
 289 GO SUB 206
 290 PRINT X$,"..WIRD GELOESCHT"
 291 LET M$=M$( TO POS-1)+M$(1+(
POS-INT POS)+1E-6)*1E4 TO )
 292 LET S=CODE Y$(10)
 293 GO SUB 57
 294 GO SUB 19
 295 LET N$=N$( TO (MST-1)*10)+N
$(MST*10+1 TO )
 296 FOR N=1 TO LEN O$
 297 IF CODE O$(N)>MST THEN  LET
O$(N)=CHR$ (CODE O$(N)-1)
 298 NEXT N
```



```
 299 LET O$(MNR)=CHR$ 0
 300 REM H A U P T P R O G R A M
M
 301 CLS
 302 PRINT "H A U P T P R O G R
A M M",,,"I=NITIALISEREN(GES.LOE
SCHUNG)","F=ORTSCHREIBEN",,"L=ES
EN",,"G=ESAMTSTATISTIK","S=AVEN"
,,"(NL=ENDE VERARBEITUNG)"
 303 INPUT I$
 304 IF I$="" THEN  STOP
 305 IF I$="S" THEN  SAVE "MKART
"
 306 IF I$="I" THEN  GO SUB 100
 307 GO TO 301+40*(I$="L")+8*(I$
="G")-50*(I$="F")
 308 REM GESAMTSTATISTIK
 309 CLS
 310 PRINT P$(8),"VOM ";
 311 IF D$<>"" THEN  GO SUB 17
 312 PRINT "---------","(DATUM O
K?NL=JA)"
 313 INPUT I$
 314 IF I$<>"" THEN  GO SUB 10
 315 PRINT ,,P$(20),,"1=MTGL.STA
MMDATEN","2=SPARTENBESETZUNG","3
=BEIDES",,,"(NL= RETURN )"
 316 INPUT I$
 317 IF I$="" THEN  GO TO 300
 318 IF I$="1" OR I$="3" THEN  G
O SUB 418
 319 IF I$="2" OR I$="3" THEN  G
O SUB 323
 320 IF INKEY$="" THEN  GO TO 32
0
 321 GO TO 309
 322 REM SPARTENSTATISTIK
 323 CLS
 324 PRINT "SPARTENBESETZUNG","-
---------------",,,"1=MTGL.ZAHL
ALLE SPORTARTEN","2=SPARTE M.MTG
L.NUMMERN","(WEITER+ RETURN => N
L)"
 325 INPUT Y$
 326 IF Y$="" THEN  RETURN
 327 IF Y$="1" THEN  GO TO 329
 328 PRINT "SPARTE VORGEBEN :","
ALLE-NL/0"
 329 PRINT "-KEINE SPARTE"
 330 INPUT A$
 331 IF A$="" THEN  GO SUB 30
 332 FOR A=1 TO LEN A$
 333 LET B$=A$(A)
 334 GO SUB 28
 336 IF Y$="1" THEN  PRINT AT A+
5,14;LEN Z$
 337 IF Y$="2" THEN  GO SUB 433
 338 NEXT A
 339 GO TO 325
 340 REM LESEN
 341 CLS
 342 PRINT "L E S E N/SUCHEN",,,
"G-EZIELT M. VORGABE","O-HNE VOR
GABE(VON A-Z)",,"(NL=RETURN)"
 343 INPUT I$
 344 GO TO 300+54*(I$="O")+45*(I
$="G")
 345 GO SUB 203
 346 PRINT ,,"1=NAMENSDATEN",,"2
=NAMENS- + STAMMDATEN","3=STAMMD
ATEN",,"(NL= RETURN /BLAETTERN)"
 347 INPUT A$
 348 IF A$="" THEN  GO TO 340
 349 GO SUB 395
 350 IF A$<"3" THEN  GO SUB 391
 351 IF A$<"3" THEN  GO SUB 131
 352 IF A$>"1" THEN  GO SUB 152
 353 GO TO 346
 354 PRINT "1=LESEN UNSORTIERT",
"2= PRINT MTGL.NUMMERN","(NL=BLA
ETTERN/<>NL= RETURN )"
 355 INPUT A$
 356 IF A$="2" THEN  GO TO 365
 357 IF A$<>"1" THEN  GO TO 340
 358 LET POS=1
 359 FOR M=POS TO 10*(LEN N$/10)
 STEP 10
 360 PRINT "HERR " AND CODE N$(M
+5)<10;"FRAU " AND CODE N$(M+5)>
10;M$(POS+CODE N$(M+1) TO POS+CO
DE N$(M+1)+CODE N$(M+2)-1);" ";M
$(POS TO POS+CODE N$(M+1)-1)
 361 LET POS=POS+CODE N$(M)
 362 INPUT A$
 363 IF A$="" THEN  NEXT M
 364 GO TO 354
 365 CLS
 366 PRINT "  MITGLIEDERLISTE","
  ---------------",,,"MNR..NAME"
,,,,
 367 GO SUB 400
 368 GO TO 340
 369 REM MTGL.DATEN-AENDERUNG
 370 LET A=VAL I$
 371 IF A<1 OR A>3 THEN  RETURN
 372 IF A=2 THEN  GO TO 385
 373 GO SUB 391
 376 LET KZ=CODE Y$(N)
 377 GO SUB 130
 378 FOR N=1 TO 5
 379 LET Y$(N)=CHR$ N(N)
 380 NEXT N
 381 LET M$=M$( TO POS-1)+X$+M$(
1+((POS-INT POS)+1E-6)*1E4 TO )
 382 LET POS=(POS+LEN X$-1)*1E-4
+INT POS
 383 LET N$(1+(MST-1)*10 TO MST*
10)=Y$
 384 IF A=1 THEN  RETURN
 385 GO SUB 395
 387 GO SUB 150
```

```
 388 LET Y$(6)=CHR$ KZ
 389 LET A=1
 390 GO TO 383
 391 FOR N=1 TO 5
 392 LET N(N)=CODE Y$(N)
 393 NEXT N
 394 RETURN
 395 LET KZ=CODE Y$(6)
 396 LET S=CODE Y$(10)
 397 RETURN
 399 REM ALLE MITGL.NACH MNR
 400 FOR M=1 TO LEN O$
 402 IF NOT CODE O$(M) THEN  GO
TO 414
 403 LET MNR=M
 404 PRINT M;
 405 GO SUB 218
 406 GO SUB 391
 407 FOR N=2 TO 5
 408 PRINT TAB 5;X$( TO N(N))
 409 LET X$=X$(N(N)+1 TO )
 410 NEXT N
 411 PRINT
 412 INPUT A$
 413 IF A$<>"" THEN  RETURN
 414 NEXT M
 415 PAUSE E4E
 416 RETURN
 417 REM STATISTK SPARTE
 418 PRINT "MITGLIEDER ";LEN N$/
10,"DAVON",,,
 419 DIM N(6)
 420 LET A$=CHR$ (CODE D$(1)-18)
+D$(2 TO )
 421 FOR N=6 TO LEN N$ STEP 10
 422 LET KZ=CODE N$(N)
 423 LET N(5)=N(5)+(KZ>6)
 424 LET KZ=KZ-10*(KZ>6)
 425 LET N(4)=N(4)+(KZ<4)
 426 LET KZ=KZ-3*(KZ>3)
 427 IF KZ THEN  LET N(KZ)=N(KZ)
+1
 428 LET N(6)=N(6)+(A$>N$(N+1 TO
 N+3))
 429 NEXT N
 430 PRINT P$(10),N(1),P$(11),N(
2),P$(12),N(3),P$(13),N(4),P$(14
),(LEN N$/10)-N(4),P$(15),N(5),P
$(16),(LEN N$/10)-N(5),P$(17),(L
EN N$/10)-N(6),P$(18),N(6)
 431 PAUSE 4E4
 432 RETURN
 433 GO SUB 85
 434 IF B$>"0" AND B$<"8" THEN
PRINT P$(VAL B$);
 435 IF B$="0" THEN  PRINT "KEIN
E SPARTE";
 436 FOR N=1 TO LEN Z$
 437 PRINT TAB 15;CODE Z$(N)
 438 NEXT N
 439 PRINT
 440 RETURN
```


Literaturhinweis

1. CHIP (Hrsg.) 'Sinclair Programme', Vogel Verlag, Würzburg 1983

2. E. Floegel 'Programmieren in BASIC und Maschinencode mit dem ZX81', Hofacker, 8150 Holzkirchen 1982

3. Tim Hartnell 'Entdecken Sie die unheimlichen Dimensionen Ihres ZX81', Cooperation, München 1982

4. H. Hergert 'Mein Sinclair ZX81', SYBEX, Düsseldorf 1983

5. R. G. Hülsmann '35 Programme für den ZX81', Hofacker, 8150 Holzkirchen, 1983

6. J. Stewart + R. Jones 'Sinclair ZX81' und 'Maschinencode und bess. BASIC', Birkhäuser, Basel 1983

7. Trevor Toms 'Das ZX81-Buch', Cooperation, München 1983