ZENTRALINSTITUT FÜR BERUFSBILDUNG DER DDR

Stoffsammlung

Informatik

im Grundlagenfach

„Grundlagen der Automatisierung"

Berlin 1988

STAATSVERLAG DER DEUTSCHEN DEMOKRATISCHEN REPUBLIK

1. Auflage
VLN 610 · DDR · Lizenz-Nr. 751 · 3113/88
Printed in the German Democratic Republic
Gesamtherstellung: Staatsdruckerei der Deutschen Demokratischen Republik
(Rollenoffsetdruck)

ZENTRALINSTITUT FÜR BERUFSBILDUNG DER DDR

Stoffsammlung

Informatik

im Grundlagenfach

"Grundlagen der Automatisierung"

Berlin 1988

Vorwort

Für das weitere Leistungswachstum und die Dynamik unserer Volkswirtschaft gewinnt die Anwendung und Beherrschung der Schlüsseltechnologien immer mehr an Bedeutung. In allen Bereichen des gesellschaftlichen Lebens nimmt der Einsatz informationsverarbeitender Technik in starkem Maße zu. Um die Facharbeiter in allen Wirtschaftsbereichen auf die sich damit ergebenden vielfältigen neuen Anforderungen gut vorzubereiten, stehen auch vor der Berufsausbildung neue, anspruchsvolle Aufgaben. Mit der Einführung der Ausbildung auf dem Gebiet der Informatik im Fach "Grundlagen der Automatisierung" für alle Lehrlinge wird der enormen Bedeutung der modernen Informationsverarbeitung für die weitere Entwicklung unserer Gesellschaft Rechnung getragen.

Sie haben sich bereits in der Oberschule erste Kenntnisse zum Umgang mit informationsverarbeitender Technik angeeignet. Im Informatikunterricht im Fach "Grundlagen der Automatisierung" werden Sie nun konkret die Arbeit mit einem Kleincomputer und dessen vielseitige Einsatzmöglichkeiten kennenlernen.

Diese Stoffsammlung enthält den gesamten Unterrichtsstoff, mit dem Sie im Informatikunterricht vertraut gemacht werden und geht in einigen Punkten noch darüber hinaus. Die in diesem Material erläuterten Anweisungen beziehen sich auf die BASIC-Version des KC 85/3 bzw. KC 85/2 mit BASIC-Modul.

Die in der Stoffsammlung formulierten Aufgaben stellen eine Auswahl der Aufgaben dar, die Sie im Informatikunterricht lösen müssen. Für die mit einem Stern (*) gekennzeichneten Aufgaben, können Sie Ihre Lösung in der Stoffsammlung selbst überprüfen. Sie werden die Stoffsammlung also zum Vorbereiten auf den Unterricht, im Unterricht selbst sowie zum Wiederholen und Überprüfen des erworbenen Wissens und Könnens nutzen. Einige Inhalte des Informatikunterrichts sind bereits ausführlich im Lehrbuch "Grundlagen der Automatisierung" dargestellt. (Im folgenden immer mit Lb GdA abgekürzt.) In der Stoffsammlung wird an den entsprechenden Stellen darauf hingewiesen.

Die Verfasser wünschen Ihnen viel Erfolg und gute Ergebnisse beim Lernen mit Hilfe dieser Stoffsammlung. Kritische Hinweise, die zur weiteren Verbesserung der Stoffsammlung beitragen können, bitten wir, dem Zentralinstitut für Berufsbildung zu übermitteln.

Zentralinstitut für Berufsbildung der DDR

Inhaltsverzeichnis

		Seite
1.	Bedeutung und Einsatz der modernen Informationsverarbeitung und Informationsverarbeitungstechnik in unserer Volkswirtschaft	4
2.	Zur Hardware eines Computers	9
2.1.	Aufbau eines Computers	9
2.2.	Aufbau des KC 85/3 bzw. KC 85/2	13
2.3.	Arbeitsweise eines Computers	13
3.	Einführung in die Arbeit mit dem Kleincomputer KC 85/3 bzw. KC 85/2	17
3.1.	Aufbau des Computerarbeitsplatzes und Inbetriebnahme des Computers	17
3.2.	Arbeit mit dem BASIC - Interpreter	19
3.3.	Eingeben, Starten und Speichern von BASIC - Programmen	22
4.	Grundlagen der Programmierung	26
4.1.	Arbeitsschritte der Programmentwicklung	26.
4.2.	Algorithmische Grundstrukturen	31
5.	Einführung in die Programmiersprache BASIC des KC 85/3 bzw. KC 85/2	36
5.1.	BASIC - Vereinbarungen	36
5.2.	Wertzuweisung und Eingabe von Daten	40
5.3.	Anweisungen zur Ausgabe von Daten	45
5.4.	Anweisungen für Programmschleifen und -verzweigungen	47
5.5.	Arbeit mit Unterprogrammen	51
5.6.	Arbeit mit Feldern	52
5.7.	ASCII - Code - Funktionen	53
6.	Spezielle Funktionen des BASIC-Interpreters des KC 85/3 bzw. KC 85/2	54
6.1.	Anweisungen zur Bildschirmgestaltung	54
6.2.	Anweisungen zur Gestaltung einer einfachen Computergrafik	58
6.3.	Anweisungen zur Tonausgabe	68
6.4.	Anweisungen zur Programmierung einfacher Bewegungen auf dem Bildschirm	70
7.	Prozessautomatisierung mit dem Computer	73
7.1.	Der Computer als flexible Steuereinrichtung	73
7.2.	Interface	74
7.3.	Steuerungen mit binären Signalen	77
7.4.	Experimente mit digitalen Signalen	81
8.	Hinweise zur Arbeit mit komplexen Anwenderprogrammen	82
9.	Entwicklungstendenzen des Computereinsatzes	88

Anlagen

Anlage 1: Tastatur des KC 85/3 bzw. KC 85/2 (Arbeitsblatt)
Anlage 2: Übersicht über Arbeitsmodi des Computers und BASIC-Interpreters (Arbeitsblatt)
Anlage 3: Standardfunktionen und mathematische Operationen
Anlage 4: Fehlermeldungen
Anlage 5: Koordinatensysteme des Bildschirmes
Anlage 6: Farbcode-Tabelle
Anlage 7: BASIC-Anweisungen des KC 85/3 und KC 85/2 mit BASIC-Modul
Anlage 8: Hinweise zur Programmoptimierung
Anlage 9: Literaturhinweise
Anlage 10: Lösungen der Übungsaufgaben
Anlage 11: Sachwortverzeichnis
Anlage 12: Merkmale der Rechnergenerationen

1. Bedeutung und Einsatz der modernen Informationsverarbeitung und Informationsverarbeitungstechnik in unserer Volkswirtschaft

Die Entwicklung unserer Volkswirtschaft wird in den nächsten Jahrzehnten maßgeblich durch die Anwendung und Nutzung der Schlüsseltechnologien bestimmt. "Die ökonomische Strategie unserer Partei mit dem Blick auf das Jahr 2000 ist darauf gerichtet, die Vorzüge des Sozialismus noch wirksamer mit den Errungenschaften der wissenschaftlich-technischen Revolution zu verbinden, die selbst in eine neue Etappe eingetreten ist. Mikroelektronik, moderne Rechentechnik und rechnergestützte Konstruktion, Projektierung und Steuerung der Produktion bestimmen mehr und mehr das Leistungsvermögen einer Volkswirtschaft. In enger Wechselwirkung damit breiten sich andere Schlüsseltechnologien aus, wie flexible automatische Fertigungssysteme, neue Bearbeitungsverfahren und Werkstoffe, die Biotechnologie, die Kernenergie und die Lasertechnik."[1]

Die Informatik ist die Lehre von der Verarbeitung, Speicherung und Darstellung von Informationen sowie den dazu erforderlichen Verfahren und Einrichtungen. Sie nimmt bei der Umsetzung der Schlüsseltechnologien eine zentrale Bedeutung ein. Der Einsatz moderner Informationsverarbeitungstechnik wird in Zukunft in allen Bereichen der Volkswirtschaft und der Gesellschaft in immer größerem Tempo erfolgen. Das Hauptziel ist dabei die Steigerung der Arbeitsproduktivität in der Produktion und die Erhöhung der Effektivität in Forschung, Entwicklung, Planung, Leitung und Verwaltung sowie im Transport- und Verkehrswesen.

1.1. Nennen Sie Auswirkungen auf unsere Volkswirtschaft, die sich aus dem Einsatz moderner Informationsverarbeitungstechnik ergeben!

Mit der starken Ausbreitung der Informationsverarbeitungstechnik sind zugleich Veränderungen in den beruflichen Anforderungen für viele Werktätige verbunden. So werden in den nächsten 15 Jahren etwa 50 % aller in unserer Volkswirtschaft Beschäftigten zumindest über ein Grundwissen und über bestimmte Fähigkeiten zum Umgang mit moderner Informationsverarbeitungstechnik verfügen müssen. Der Einsatz von Computern zur Lösung spezieller beruflicher Aufgaben schließt auch eine Reihe neuer Tätigkeiten ein, die mit veränderten Anforderungen an das Wissen und Können der Facharbeiter verbunden sind.

1.2. Welche besonderen Tätigkeiten werden beim Einsatz von Informationsverarbeitungstechnik ausgeführt? Beobachten Sie Werktätige in Ihrem Betrieb, in der Sparkasse, Post oder anderen Bereichen und Einrichtungen!

Die Entwicklung der Informationsverarbeitungstechnik vollzieht sich seit dem Bau der ersten elektronischen Rechenmaschine durch Konrad Zuse in den Jahren 1936-1941 mit ständig steigendem Tempo. Die Etappen der Entwicklung entsprechen etwa den Rechnergenerationen. Diese sind jeweils durch bestimmte Merkmale der jeweiligen Technik bestimmt. (Vgl. Anlage 12: Merkmale der Rechnergenerationen)

1.3. Informieren Sie sich über die Entwicklung der Rechentechnik im Lb GdA S. 78 !

Mit der Entwicklung hoch- und höchstintegrierter Schaltkreise Anfang der 80er Jahre begann eine neue Rechnergeneration (4. Generation) sich zu entwickeln. Die Leistungsfähigkeit dieser Rechner wuchs stetig auf der Grundlage neuer Verarbeitungskonzepte und Nutzungsformen. Wichtige Merkmale der heutigen Rechentechnik sind die dezentrale Informationsverarbeitung, der Dialogbetrieb, der Aufbau von Rechnernetzen und komplexen Datenbanken. Ein typischer Vertreter dieser Technik ist der bedienfreundliche und vernetzungsfähige Computer am Arbeitsplatz.

---

1 Bericht des Zentralkomitees der SED an den XI. Parteitag der SED, Berichterstatter: Gen. Erich Honecker, Dietz Verlag, Berlin 1986, S. 49

Diese Computer

- ermöglichen eine einfache Nutzung durch dialogorientierte Bildschirmarbeit und stellen damit eine wichtige Voraussetzung für rechnergestützte Arbeitsplätze dar;
- lassen sich an die spezifischen Arbeitsaufgaben anpassen, wozu auch die leistungsfähigen peripheren Geräte beitragen;
- können miteinander und mit größeren Rechnern zu sogenannten Rechnernetzen verbunden werden.

Zugleich behalten Klein- und Großrechner ihre Bedeutung und werden in ihrer Leistungsfähigkeit rasch weiterentwickelt.

1981 kündigten japanische Computerhersteller an, bis in die 90er Jahre einen Computer der fünften Generation zu entwickeln. Die neue Qualität dieser Generation soll vor allem darin bestehen, daß die Rechner als Kernstück ein wissensverarbeitendes System besitzen, das selbständig Probleme lösen und logische Schlüsse ziehen kann. Spracheingabe sowie Bildein- und -ausgabe werden wichtige Merkmale dieser Rechnergeneration sein.

Im folgenden sollen einige Beispiele zur Anwendung der Informationsverarbeitungstechnik aus verschiedenen Bereichen unserer Volkswirtschaft zeigen, welche Einsatzbreite erreicht werden konnte.[1]

1 Die Beispiele wurden den Zeitschriften "Jugend und Technik", "Technikus" und "Urania" entnommen (Jahrgänge 1986 und 1987).

Einsatz der Computertechnik zur Optimierung von Produktionsprozessen durch automatische Steuerung. Dabei werden die Computer vorwiegend unmittelbar in der Produktion eingesetzt. Z. B. zur Steuerung von Chemieanlagen, Walzstraßen, Werkzeugmaschinen, Kernkraftwerken, Anlagen für die Mischfutterherstellung.

Beispiele:

Ein neuentwickelter Drehautomat des Werkzeugmaschinenbaus spart bis zu 25 % Fertigungszeit bei der Herstellung von Wälzlagerringen ein. Die Maschine verfügt über eine mikroelektronische Steuerung und kann sechs Werkstücke gleichzeitig aufnehmen. Für den Anwender ergibt sich eine hohe Flexibilität bei der Einrichtung kleiner und mittlerer Serien. Die im Leipziger Drehmaschinenwerk konstruierte und gebaute Maschine besitzt neuartige Getriebe und Lagerungen der Drehspindeln und erreicht eine hohe Fertigungsgenauigkeit.

Geräte und Methoden zum Messen des Korndurchsatzes am Mähdrescher sind im Forschungsinstitut für Mechanisierung der Landwirtschaft Schlieben/Bornim und dem VEB Kombinat Fortschritt Neustadt entwickelt worden. Ein Sensor liefert für den Bordcomputer die Werte, die verrechnet und angezeigt bzw. gespeichert werden. Es lassen sich folgende für die Produktion wichtige Werte ermitteln: Korndurchsatz in Kilogramm je Sekunde, geerntete Kornmasse in Tonnen je Schicht, Schlag, Arbeitstag und Kampagne, Kraftstoffverbrauch je geerntete Tonne Getreide usw. Mit Hilfe dieser Methode können die Arbeitsabläufe rationeller gesteuert werden, was unter anderem zu einer höheren Arbeitsproduktivität, zu vermindertem Kraftstoffverbrauch und zu geringeren Verlusten führt.

Einsatz der Computertechnik zur Lösung ökonomischer und technologischer Probleme. Dabei erfolgt der Einsatz der Computer mittelbar bezüglich der materiellen Produktion in den Bereichen der Ökonomie und Technologie. Im Vordergrund stehen Aufgaben der Rechnungsführung und Statistik, es werden Pläne bilanziert, die Materialbestandhaltung kontrolliert, die Lohnrechnung durchgeführt, die Grundmittelauslastung überprüft, optimale Technologien ermittelt, Transporte optimiert, statistische Erhebungen und Berechnungen durchgeführt u. a.

Beispiele:

Im Betrieb VEB Jugendmode Rostock wird durch eine CAD/CAM-Strecke eine Produktionssteigerung von rund 16 000 Erzeugnissen im Jahr erreicht. Am grafischen Bildschirmarbeitsplatz wird die materialökonomischste Schnittvariante ermittelt. Ein Automat übernimmt danach den Zuschnitt für 100 Stofflagen gleichzeitig. So können 14 000 Stunden Arbeitszeit pro Jahr eingespart werden, weil sich die Arbeiten, die vor dem eigentlichen Nähen liegen durch den Computereinsatz in viel kürzerer Zeit bewältigen lassen.

Fachleute des Institutes für Obstforschung in Dresden-Pillnitz erarbeiteten Computerprogramme mit deren Hilfe Voraussagen zum zu erwartenden Ertrag möglich werden. Für die Planung der notwendigen Arbeitskräfte und der Arbeitsmittel während der Ernte steht den Betrieben das Programm "Kampagneplanung Ernte" zur Verfügung. Auch Informationen über optimale Düngung oder zur chemischen Unkrautbekämpfung können über Computer abgefordert werden.

Seit 1977 fertigt die INTERFLUG ihre Passagiere und das dazugehörige Gepäck mit Hilfe eines Rechnerkomplexes ab. Ausgehend von der Passagieranzahl und dem dazugehörigen Gepäck werden alle weiteren Massen, wie Post und Fracht bis zum größten Anteil, dem Flugtreibstoff, dem Rechner mitgeteilt, um so zur sogenannten Trimmung - der Schwerpunktermittlung und der damit verbundenen Massenverteilung - Aussagen treffen zu können.

Einsatz der Computertechnik in Forschung und Entwicklung. Mit Hilfe des Computers werden Konstruktionswerte berechnet bzw. Konstruktionen ausgeführt, z. B. im Rahmen von CAD-Arbeitsplätzen (CAD-Computer aided design = computergestützte Konstruktion). In vielen Bereichen werden Computer genutzt zur Auswertung von Analysen sowie zur Berechnung umfangreicher mathematisch-technischer Aufgaben.

Beispiele:

Die Analyseautomaten im Qualitätslabor des Instituts für Pflanzenzüchtung Güstrow - Gülzow nehmen den Forschungsingenieuren aufwendige Einzeluntersuchungen ab. Die den speziellen Bedürfnissen der Agrarwissenschaftler angepaßten wissenschaftlich-technischen Geräte führen Analysen automatisch aus, so daß nur noch die Eingabe der Probe und das Entnehmen der ausgedruckten Daten manuell erfolgen.

Ein Gerät, mit dem feingedrehte Metallteile automatisch auf Rauheit und Defekte geprüft werden können, wurde an der Friedrich-Schiller-Universität Jena entwickelt. Das international neue Gerät kann mittels Laser und optischer Prinzipien berührungsfrei die beiden Parameter erfassen. Dabei schließt rechnergestützte Datenauswertung in der Qualitätsprüfung subjektive Fehler aus. Das Gerät läßt beim Anwender eine Steigerung der Arbeitsproduktivität um mehr als 200 Prozent zu.

In der wissenschaftlichen Arbeit sind Experimente unbedingt für die Erkenntnisgewinnung erforderlich. Experimente z. B. mit thermonuklearen Reaktionen, in Windkanälen oder Teilchenbeschleunigern sind jedoch sehr kompliziert und kostenaufwendig. Durch den Einsatz der Computertechnik können einige dieser Experimente simuliert und durch Rechnerexperimente ersetzt werden, wodurch eine hohe Kosten- und Zeiteinsparung erreicht wird.

Die Möglichkeiten der Computergrafik werden nicht nur für neue künstlerische Gestaltungsformen, sondern z. B. auch zum Musterentwurf und zur Schnittgestaltung im Bereich der Textilindustrie genutzt.

Breite Anwendung findet die Computertechnik auch in der Projektierung. Im Dialog mit dem Computer entstehen im Industriebau der DDR mehrgeschossige Mehrzweckgebäude-Projektionen. Durch das CAD-System sinkt der Zeitaufwand der Projektierung um 75 %. Für den Entwurf der Gebäude stehen 4.000 unterschiedliche Elemente zur Verfügung. Daraus wählt der Computer entsprechend dem Zweck der Häuser die notwendigen Teile aus und kombiniert diese optimal, so daß sich der Bauaufwand verringert.

Einsatz der Computertechnik zur Rationalisierung der Verwaltungsarbeit und im Dienstleistungsbereich. In diesem Bereich ist eine große Einsatzbreite zu verzeichnen. Das beginnt mit dem Einsatz von Computern zur Erfassung, Speicherung und Verarbeitung von Daten unmittelbar am Arbeitsplatz im Büro. Das Hauptanliegen ist dabei der Aufbau rechnergestützter hierarchischer Informations- und Leistungssysteme. So erfolgt der Einsatz von Computern z. B. zur Materialrechnung, in Banken, Sparkassen und Postämtern zur Bedienung der Kunden und zur Rationalisierung der Buchungsprozesse.

Reservierungs- und Bestellsysteme werden z. B. im Verkehrswesen (Platzkartenreservierung, Flugkartenbestellung) im Hotelwesen und der Gastronomie (Reservierung von Hotelplätzen bzw. Gaststättenplätzen), im Reisebüro (Verteilung von Campingplätzen) sowie im kommunalen Bereich (Wohnungstausch) eingesetzt.

Eine Rationalisierung der Verwaltungsarbeit kann auch mit dem Einsatz von Textverarbeitungssystemen erreicht werden. Dabei können Texte problemlos korrigiert und verändert werden, mehrfach zu verwendende Texte können gespeichert und bei Bedarf abgerufen werden, die Texte können individuell gestaltet werden,und durch den Einsatz von fertigen Textbausteinen kann der Schreibaufwand verringert werden.

Einsatz der Computertechnik in der Militärtechnik. Der Einsatz erfolgt z. B. zur Berechnung von Richtwerten für Raketenwaffen, Flak und Artillerie, zur Vorbereitung von Entscheidungen zur Truppenführung, zur Berechnung von Entfernungen, Geschwindigkeiten u. a.

Beispiel:

Bei den Landstreitkräften der NVA wurde als Standard-Ausbildungshilfe ein Schießtrainer für Panzerbesatzungen entwickelt. Von einem zentralen Pult kann der Ausbilder verschiedene Bedingungen simulieren sowie die Handlungen mehrerer Richtlenkschützen verfolgen. Fehler werden dem Ausbilder sofort signalisiert.

Einsatz der Computertechnik in der Medizin. Die Methode zur Gewinnung einer Abbildung vom Inneren eines undurchsichtigen Körpers (Tomographie) wird in der Medizin zur Früherkennung von Erkrankungen der inneren Organe bzw. Gewebeveränderungen im menschlichen Körper (Krebs) genutzt. Andere Anwendungsgebiete sind die Geburtenüberwachung, Patientenkontrolle, Diagnose sowie Analyseverfahren.

Auch moderne Behandlungsgeräte unterstützen die Heilung und erleichtern die Arbeit der Mediziner.

Beispiel:

Das mikroprozessorgesteuerte Kurzwellen-Therapiegerät (TuR KWI 5) ist ein Hochleistungsgerät für Wärmebehandlungen mit Hochfrequenzenergie. Je nach therapeutischer Zielstellung kann sie kontinuierlich oder gepulst verabreicht werden. Hierbei erfolgt die Abstimmung in beiden Betriebsarten und für alle Applikationsweisen computergesteuert.

Einsatz der Computertechnik in der Raumfahrt. Mit Hilfe der Computer werden Kopplungsmanöver berechnet, Kurskorrekturen durchgeführt sowie Daten von Raumflugkörpern ausgewertet.

Beispiele:

Bei der Erforschung des Kometen Halley wurden mehrere Raumflugkörper gestartet. Unter Ausnutzung der Gravitation der Venus wurden die VEGA-Sonden auf eine Bahn zu Halley gebracht. Die Raumflugkörper begegneten dem Kometen mit einer hohen Geschwindigkeit. Diese setzt eine entsprechende Präzision der Annäherung sowie der Ausrichtung der Geräte voraus, um eine optimale Informationsausbeute zu erreichen. Bei diesen Sonden sind die Beobachtungsgeräte auf einer Instrumentenplattform montiert, die durch ein Zielsuch-Computersystem horizontal und vertikal ständig auf den Kometenkern fixiert wurde. Eine Fernsteuerung von der Erde ist aufgrund der langen Signallaufzeit nicht möglich.

Eine arbeitsaufwendige Berechnungsaufgabe ist die Wettervorhersage. Zahlreiche Informationen gehen von Satelliten und Wetterstationen ein, die gesammelt und analysiert werden müssen. Umfangreiche Berechnungen sind erforderlich für die Modellierung der Prozesse in der Atmosphäre und im Ozean, um diese schließlich in einer für den Menschen verständlichen Form darzustellen. Auch hier liegt ein breites Einsatzfeld der Computertechnik.

Einsatz der Computertechnik im Freizeitbereich. Zunehmend findet die Computertechnik Anwendung in Haushaltgeräten, im Heimwerkerbedarf, in der Heimelektronik, bei Sportgeräten usw.

Beispiele:

So wird vom VEB Haushaltgeräte Karl-Marx-Stadt ein Elektroherd mit Back- und Bratcomputer hergestellt. Die Automatik erlaubt es, verschiedenste Back-, Brat- und Auftauprozesse im Backraum automatisch ablaufen zu lassen. Im Automatikbetrieb stehen 63 frei wählbare Programme nach vorgegebenen Temperatur-Zeit-Verläufen zur Verfügung. Die ablaufende Garzeit wird elektronisch angezeigt und das Ende des Garprozesses ebenso akustisch signalisiert wie zwischenzeitlich notwendige Arbeitsgänge, z. B. das Wenden des Bratgutes.

Der GERMINA-Boxroboter ist vom Jugendboxer bis zum Superschwergewichtler als Trainingspartner einsetzbar.

In allen Bereichen, in denen moderne Informationsverarbeitungstechnik Anwendung findet, können mechanische und routinemäßige Arbeiten von Computern übernommen bzw. erleichtert werden und der Mensch erhält dadurch mehr Raum für schöpferische Tätigkeiten.

1.4. Suchen Sie weitere Beispiele in Ihrem Ausbildungsbetrieb und in der Literatur (Tageszeitungen u. ä.)!

Mit dem Einsatz der Informationsverarbeitungstechnik für die unterschiedlichsten Aufgaben in allen Bereichen der Volkswirtschaft haben sich gleichzeitig unterschiedliche Organisationsformen entwickelt.

1.5. Informieren Sie sich dazu im Lb GdA S. 85!

2. Zur Hardware eines Computers

Die Hardware ist die gerätetechnische Basis eines jeden Computers. Prinzipiell gleich sind in allen Computern die Hauptbaugruppen und die Arbeitsweise. Unterschiede zwischen den einzelnen Computertypen gibt es z. B. hinsichtlich der Art und Kapazität der internen und externen Speicher sowie der Möglichkeit des Anschlusses verschiedener peripherer Geräte (Drucker, Plotter u. a.).

Vor der Arbeit mit einem Computer, vor der Entwicklung des ersten eigenen Programms, ist es günstig, sich zumindest einen Überblick über den Aufbau des Computers zu verschaffen. Dieser grundsätzliche Überblick genügt zunächst, weil es bei der weiteren Arbeit vor allem um Programmierübungen in einer höheren Programmiersprache (in unserem Fall BASIC), bzw. um die Arbeit mit Anwenderprogrammen geht. Deshalb wird in den Abschnitten 2.1. und 2.2. ein allgemeiner Überblick über den Aufbau eines Computers und über die Hardwarestruktur des KC 85/3 bzw. KC 85/2 gegeben. Dieser Überblick ist für das Verständnis des nachfolgenden Stoffes ausreichend.

Für die Programmierung in Maschinensprache oder in einer maschinennahen Sprache ist jedoch die Kenntnis der Arbeitsweise des Computers, der Struktur seiner zentralen Verarbeitungseinheit und der Organisation der Befehlsausführung erforderlich.

Für Interessenten werden im Abschnitt 2.3. einige vertiefende Fakten zur Arbeitsweise eines Mikrorechners dargestellt.

2.1. Aufbau eines Computers

Ein Computer ist ein technisches System zum Speichern und Verarbeiten von Informationen, welches aus hochintegrierten Schaltkreisbausteinen auf Mikroprozessorbasis aufgebaut ist. Für den Begriff Computer wird aus diesem Grund auch das Synonym Mikrorechner verwendet, obwohl man mit einem Computer weit mehr als nur rechnen kann. Grundlage für die Realisierung der vielgestaltigen Aufgaben mit einunddemselben Gerät sind die verschiedenen Programme (Software). Der Computer bekommt also durch ein Programm die für den jeweiligen Einsatzfall benötigten Eigenschaften.

Bild 2 zeigt den verallgemeinerten Aufbau eines Computerarbeitsplatzes. Der Computer ist Teil eines Systems von Hard- und Softwarekomponenten, die für den jeweiligen Anwendungsfall zugeschnitten sind.

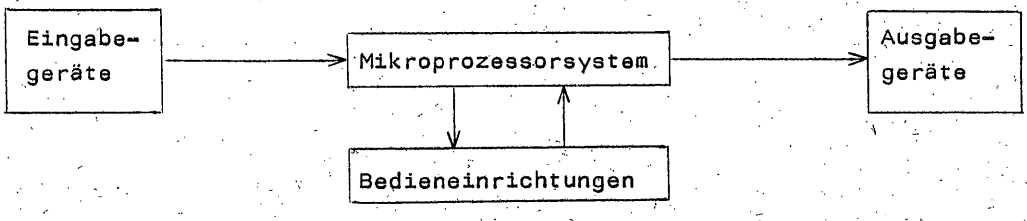


Bild 2: Aufbau eines Computerarbeitsplatzes

HARDWARE ist die Gesamtheit aller technischen Einheiten eines Computers. Sie umfaßt Speicher für Programme und Daten, die zentrale Verarbeitungseinheit (ZVE oder CPU - central processing unit), Baugruppen zur Verbindung der Verarbeitungseinheit mit Eingabe/Ausgabe-Geräten und diese Geräte selbst.

SOFTWARE ist die Gesamtheit der zu einem Computersystem gehörenden Programme, die den Einsatz des Computersystems für eine bestimmte Aufgabe ermöglichen, vereinfachen oder erweitern. Dazu gehören:

- Betriebssysteme, die den inneren Ablauf sowie die Abarbeitung der Anwenderprogramme im Computer organisieren;
- Standardsoftware (auch Basissoftware genannt), die branchenneutrale Anwenderprogramme darstellen;
- Anwendersoftware, Programme, die für spezielle zweig- und betriebsbezogene Probleme erarbeitet werden; u. a.

Für die Entwicklung der Computertechnik ist die weitere Entwicklung der Mikroelektronik eine entscheidende Voraussetzung. Je mehr Funktionen die einzelnen Schaltkreisbausteine realisieren, um so leistungsfähiger und kleiner kann ein Computer werden.

Ein Computer besteht im wesentlichen aus fünf Hauptbaugruppen (vgl. Bild 3):

- dem Mikroprozessor, der zentralen Verarbeitungseinheit (ZVE, CPU), die logische und arithmetische Operationen ausführt und Steuerfunktionen für den gesamten Computer übernimmt,
- dem Speicher für Daten, Programme und Bilder (RAM, ROM und IRM),
- den Eingabe/Ausgabe-Bausteinen (auch Schnittstellen genannt), die die Verbindung zwischen dem Computer und den peripheren Geräten realisieren,
- einem System von Informationsleitungen (BUS - binary unit system), das den Informationsfluß zwischen den Baugruppen realisiert und damit ihr Zusammenspiel ermöglicht,
- einer Systemperipherie, die für den Dialog Mensch-Computer erforderlich ist.

Dazu gehören z. B. die Tastatur als Eingabeeinrichtung für Daten und Programme, der Bildschirm als Sichtgerät und ein Kassettenrecorder als externe Speichereinheit.

Darüber hinaus kann ein Computer an externe Prozesse angeschlossen werden und dort Meß- und Steuerungsaufgaben übernehmen. Hierzu sind weitere Baugruppen erforderlich, die den Computer befähigen, Informationen von den zu steuernden Prozessen entgegenzunehmen und entsprechende Steuersignale nach außen abzugeben. Die hierzu benötigten Baugruppen seien unter dem Begriff Prozeßperipherie zusammengefaßt.

Bild 3 zeigt den Zusammenhang der genannten Hauptbaugruppen am Blockbild zum KC 85/3 bzw. KC 85/2. Das Zusammenschalten dieser Baugruppen ergibt jedoch noch kein arbeitsfähiges System. Es ist noch ein Programm erforderlich, das den Computer gleich beim Einschalten mit bestimmten Eigenschaften ausstattet. Ein solches Programm nennt man das BETRIEBSSYSTEM des Computers. Diese Form der Software ist für Computerfunktionen genauso wichtig wie die Hardware-Baugruppen.

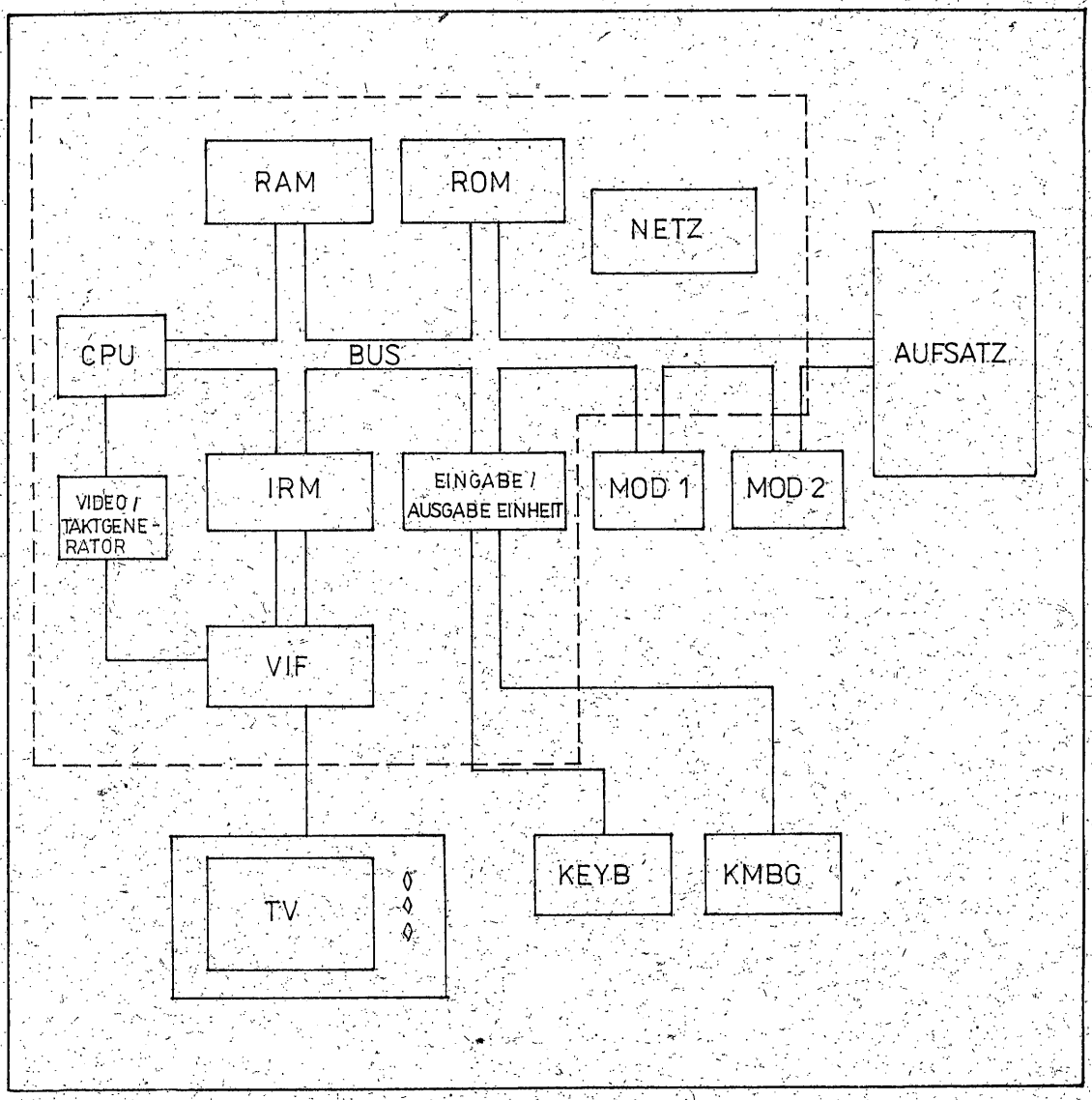


Bild 3: Blockschaltbild des KC 85/3 bzw. KC 85/2 mit Systemperipherie

Beispiel 1:

Der KC 85/3 soll als Textverarbeitungscomputer fungieren. Der Computerarbeitsplatz in Bild 2 bekommt nun folgende Komponenten:

HARDWARE:

Computer-Grundgerät mit Tastatur, Bildschirm und Recorder zur Texteingabe, -bearbeitung und -speicherung
Modul MØØ3 V24 zum Druckeranschluß (Schnittstelle)
Drucker zur Textausgabe

SOFTWARE:

Betriebssystem des Computers zur Einschaltinitialisierung des Computersystems und zur Modulverwaltung
Textverarbeitungssystem z. B. TEXOR auf Modul MØ12 (der TEXOR-Modul selbst würde noch mit zur Hardware gehören)
Treiberroutine zur Textausgabe an den Drucker (ist auf dem TEXOR-Modul mit enthalten)

2.1.1. Welche Komponenten hat der Computerarbeitsplatz in Bild 2, wenn der KC 85/3 eine Steuerungsaufgabe realisiert z. B. die Füllstandsteuerung?

Auf die Systemperipherie und deren Zusammenschaltung mit dem Computer am Arbeitsplatz im Computerkabinett wird im Abschnitt 3.1. näher eingegangen. Das System von Informationsleitungen

(BUS-System) und der Mikroprozessor werden im Abschnitt 2.3. näher beschrieben. Bleiben von den fünf Hauptbaugruppen eines Computers noch die Speicher und die Ein- und Ausgabebausteine.

Die Speicher

Es gibt prinzipiell zwei verschiedene Speichertypen in einem Computer:

- ROM (read-only-memory): Nur-Lese-Speicher und
- RAM (random-ascess-memory): Schreib-Lese-Speicher.

Der ROM dient zur Speicherung von festen Programmen und Konstanten. "Fest" heißt in diesem Fall, daß der Speicher seinen Informationsinhalt (also z. B. das Programm) nicht verliert, wenn der Computer ausgeschaltet wird. Das oben erwähnte Betriebssystem ist ein solches Programm, das auf dem ROM gespeichert ist, denn es muß sofort nach Einschalten des Computers die Steuerung der einzelnen Baugruppen übernehmen.

Es gibt zwei verschiedene Arten von ROM's:

- solche, die vom Hersteller unveränderbar programmiert wurden, sogenannte PROMs (engl.: programmable ROM = programmierbare ROM) und
- solche, die vom Anwender gelöscht und durch Spezialgeräte beschrieben werden können, sogenannte EPROMs (engl.: erasable programmable ROM = löschbare programmierbare ROM).

Der RAM ist ein Speicher, der beschrieben und gelesen werden kann, der aber seinen Informationsinhalt verliert, wenn der Computer ausgeschaltet ist. Der RAM wird z. B. als BASIC-Arbeitsspeicher genutzt und als Speicher, von dem aus die Bildschirmausschrift und -farbe gespeist wird.

Die Art der Informationsspeicherung der beiden ROM-Arten und des RAM-Speichers kann durch folgenden Vergleich verdeutlicht werden: Der Informationsinhalt einer Fotografie oder eines Dias ist unveränderbar. In gleicher Weise wird die Information in einem PROM gespeichert. Der Informationsinhalt einer Polylux-Folie hingegen, kann durch eine Spezialflüssigkeit (Spiritus) gelöscht werden. Danach kann man die Folie neu beschreiben. So kann man sich die Informationsspeicherung auf einem EPROM vorstellen. Hier kann die vorhandene Information durch UV-Licht gelöscht werden. Wird ein Dia oder eine Folie an eine Bildwand projiziert, so hat die Projektionsfläche einen bestimmten Informationsinhalt. Der Informationsinhalt kann durch ein anderes Dia geändert werden. Er wird gleich Null, wenn das Licht ausgeschaltet wird. In ähnlicher Weise verliert ein RAM seine gespeicherte Information, wenn die Versorgungsspannung abgeschaltet wird. Der Informationsinhalt des RAMs kann z. B. durch Einlesen eines anderen Programms geändert werden.

2.1.2. Ordnen Sie den Bildwiederholspeicher des Computers einem der beiden Speichertypen zu!

Die Ein-/Ausgabebausteine

Für die Eingabe bzw. Ausgabe gibt es zwei unterschiedliche Bausteine:

- PIO (parallel input output) parallele Eingabe/Ausgabe.

Der PIO-Baustein ermöglicht den 8-bit-parallelen Datenaustausch zwischen dem Mikroprozessor und seiner Peripherie. Das bedeutet, daß 8-Bit-Daten (2^0, 2^1, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7) gleichzeitig über je eine Leitung ein- bzw. ausgegeben werden.

- SIO (serial input output): serielle Eingabe/Ausgabe.

Der SIO-Baustein realisiert den bitseriellen Datenaustausch zwischen dem Mikroprozessor und seriellen Eingabe/Ausgabe-Geräten. Das bedeutet, daß 8-Bit-Daten (2^0, 2^1, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7) über eine einzige Leitung nacheinander ein- bzw. ausgegeben werden. Z. B. die Tastatur des KC ist ein bitserielles Eingabegerät für den Computer.

2.1.3. Über welchen der oben genannten Bausteine erfolgt die Ankopplung des Experimentiersatzes an den Computer?

2.2. Aufbau des Kleincomputers KC 85/3 bzw. KC 85/2

Bild 3 zeigt den Hardwareaufbau des Kleincomputers KC 85/3 bzw. KC 85/2. Die Strichpunktlinie schließt alles das ein, was zum Grundgerät gehört. Die rechts angegebenen, mit dem Computer verbundenen Geräte bilden die Systemperipherie. Über spezielle Module ist es möglich, auch Prozeßperipherie anzuschließen.

Zentrale Recheneinheit

Die zentrale Recheneinheit des Computers besteht aus dem Mikroprozessor (CPU) U 88Ø, dem Arbeitsspeicher (RAM), dem Bildwiederholspeicher (IRM) und dem Festwertspeicher (ROM), in dem das Betriebssystem und der BASIC-Interpreter enthalten sind. Der Arbeitsspeicher, also der für den Anwender nutzbare Speicherbereich ist im Grundgerät 16 K Byte groß.

Der IRM (image repetition memory) ist ein zweiter Speicher, auf den von der einen Seite der Prozessor lesend und schreibend und von der anderen Seite das Videointerface nur lesend zugreifen kann. Im Bildwiederholspeicher sind die auf dem Bildschirm sichtbaren Informationen als Punkte (Pixel-RAM), als Farbattribute (Color-RAM) und als ASCII-Zeichen (Video-RAM) gespeichert. In einem Feld von 8 x 8 Bildpunkten wird jeweils ein Zeichen abgebildet. Insgesamt können pro Bild 32 Zeilen mit je 4Ø Zeichen abgebildet werden. Jedem Bildfeld von 8 x 4 Bildpunkten ist ein Farbbyte zugeordnet.

Videointerface

Das Videointerface ist die Schnittstelle des Computers zum Datensichtgerät. Es ist so ausgelegt, daß das Fernsehgerät direkt über den RGB-Eingang, über den FBAS-Eingang oder über den Antenneneingang angeschlossen werden kann. Die beiden zuerst genannten Anschlüsse sind als direkter Steckverbinder an der Rückwand des Grundgerätes herausgeführt. Zum Anschluß an den Antenneneingang ist eine Leitung aus dem Computer herausgeführt.

Ein- und Ausgabesteuerung

Die Ein- und Ausgabesteuerung hat die Aufgabe, die von der Tastatur und/oder vom Kassettenrecorder ankommenden seriellen Signale so aufzubereiten, daß sie vom Computer weiterverarbeitet werden können. Weiterhin werden die vom Computer erzeugten seriellen Signale für den Kassettenrecorder aufbereitet.

In der Tastatur ist ein Fernbedienungsschaltkreis zur Serialisierung der Tasteninformationen eingesetzt. Ihr Anschluß zum Computer erfolgt über eine einadrig geschirmte Leitung, über die sowohl die Stromversorgung zur Tastatur als auch die Informationsübertragung zum Computer realisiert wird.

Hardware-Erweiterungen

Das Grundgerät des KC 85/3 u. 2 erlaubt den Anschluß von zwei Erweiterungsmodulen und bis zu 15 Erweiterungsaufsätzen. Für die Module befinden sich an der Vorderseite des Grundgerätes zwei Modulschächte (C und 8). Die Aufsätze besitzen vier Modulschächte und sind an der Rückseite untereinander und mit dem im Grundgerät enthaltenen Bussystem verbunden. Jeder Aufsatz enthält eine eigene Stromversorgung.

2.3. Arbeitsweise eines Computers

Der Computer arbeitet ein Programm Befehl für Befehl ab. Die Programmabarbeitung wird durch die zentrale Verarbeitungseinheit, die CPU (central processing unit) gesteuert. In den einzelnen Befehlen sind Informationen über die auszuführenden Operationen, die beteiligten Daten und gegebenenfalls über den nachfolgenden Befehl enthalten. Diese Informationen sind in einem Binär-

wort (=endliche Folge von binären Zeichen), dem Befehlewort verschlüsselt. (Vgl. Lb GdA S. 16 ff.). Die Ausführung eines Befehls selbst vollzieht sich in verschiedenen Phasen bzw. Zyklen (siehe Bild 4). Zunächst wird der Befehl aus dem Arbeitsspeicher in die CPU geholt (Befehlsholezyklus), dann interpretiert und schließlich ausgeführt, nachdem noch die benötigten Daten in die CPU gebracht worden sind.

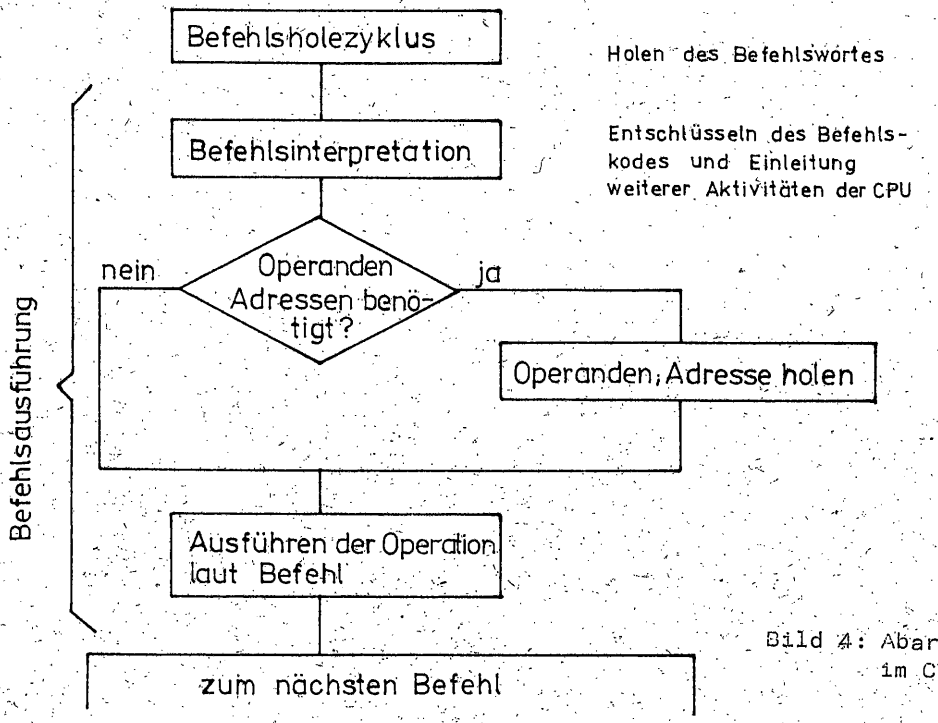


Bild 4: Abarbeitung eines Befehls im Computer

Das alles geschieht unter der Regie des Grundtaktes des Computers. Die Frequenz dieses Grundtaktes bestimmt die Arbeitsgeschwindigkeit des Computers. Der Takt wird vom Taktgenerator vorgegeben und hat beim KC 85/3 eine Frequenz von 1,75 MHz. Die Dauer eines Befehlszyklus richtet sich neben der Taktfrequenz auch nach der Anzahl der Taktperioden, die für die einzelnen Phasen der Abarbeitung eines Befehls benötigt werden. Bild 5 zeigt die Aufgliederung der Abarbeitungsphasen eines Befehls in einzelne Maschinentakte.

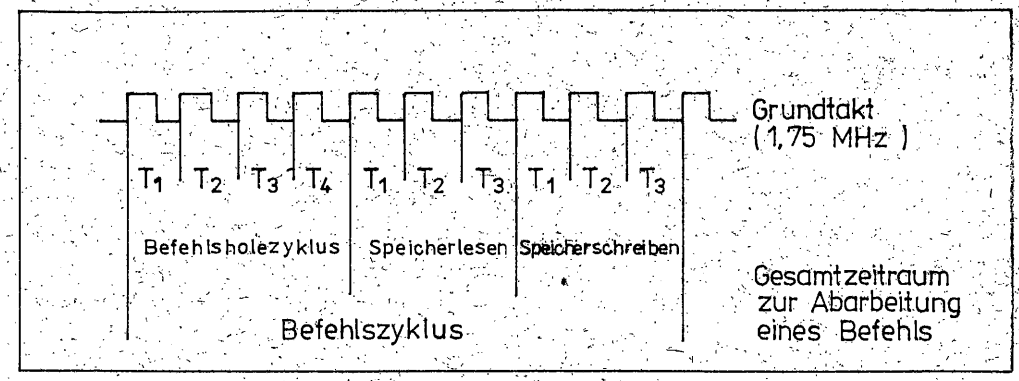


Bild 5: Aufbau eines Befehlszyklus

2.3.1. Berechnen Sie die Dauer des Befehlszyklus aus Bild 4 für den KC 85/3 bzw. KC 85/2! Wieviele solcher Maschinenbefehle kann der KC in einer Sekunde ausführen?

Das Bussystem

Das Bussystem (bus: binary unit system) ist ein System von Leitungsbündeln für den Informationsfluß. Ein Bus ist vorstellbar als eine Reihe von parallelen Drähten, 8 für Daten, 16 für Adressen und 13 für Steuerinformationen. Damit werden unterschieden Datenbus, Adressbus und Steuerbus.

Zum Lokalisieren von Daten und Befehlen enthält jeder Speicherplatz und jede Ein- und Ausgabeeinheit eine Nummer, die Adresse. Auf diese wird dann im Programm immer bezug genommen.

Der Austausch von Informationen (wie Befehlsadresse und Befehlswort, Datenadresse und Datenwort, Ein- und Ausgabeadresse und Ein- und Ausgabewort) erfolgt zwischen der CPU und dem Arbeitsspeicher bzw. den Ein- und Ausgabeeinheiten über das Bussystem.

Der Datenbus überträgt die Daten zwischen den Baugruppen des Computers.

Der Adressbus überträgt die von der CPU erzeugten Adressen an alle Speicher und Ein- und Ausgabeeinheiten.

Der Steuerbus überträgt die zum Abstimmen aller Aktivitäten des Systems erforderlichen Steuersignale zu den jeweiligen Einheiten.

Die Struktur der CPU

Die CPU eines Computers hat zur Bewältigung ihrer Aufgaben im allgemeinen

- ein Adreßwerk zur Ermittlung und Bereitstellung der Befehls- und Operandenadressen,
- ein Rechenwerk zur Zwischenspeicherung und Verarbeitung der Daten und
- ein Steuerwerk zur Steuerung der CPU-internen und externen Abläufe (Bild 6).

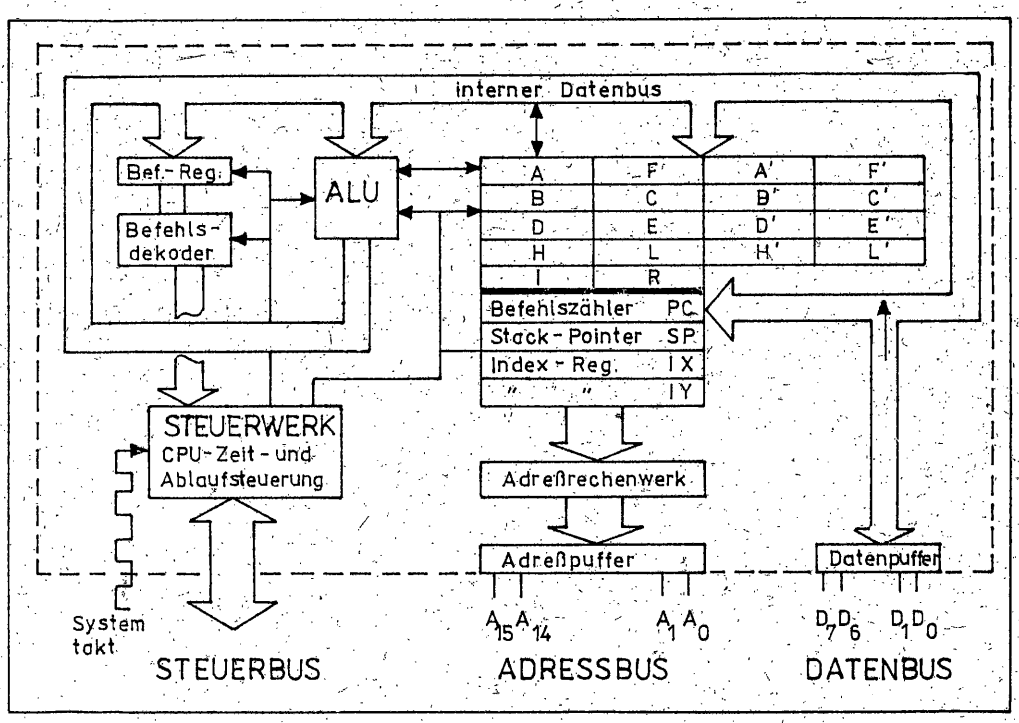


Bild 6: Struktur der CPU des U 880

Die wesentlichsten Funktionsblöcke des Rechenwerks sind die arithmetisch-logische Einheit (ALU: arithmetik logik unit) und die zu einem Registerblock zusammengefaßten internen Datenregister. Die ALU führt arithmetisch-logische Operationen mit einer Wortbreite von 8 Bit aus. Sie kommuniziert mit den Registern des internen Registerblocks und dem externen Datenbus über den internen CPU Datenbus. Die ALU des U 880 kann arithmetische und logische Grundverknüpfungen und Verschiebeoperationen ausführen.

Das Steuerwerk sorgt für die ordnungsgemäße Programmabarbeitung und steuert die externen Leitabläufe. Die Basis hierfür bilden der Systemtakt, die aktuellen Befehlsfolgen und die CPU-externen Steuersignale.

Im Adreßwerk dient der Befehlszähler zum Speichern der Befehlsadressen. Zur Ermittlung der Adressen von Operanden und Befehlen sind besondere Adreßregister und ein Adreßrechenwerk vorhanden.

Die Register der CPU

Die CPU U 880 ist vorwiegend für die Verarbeitung von Wörtern mit 8 Bit Breite vorgesehen. Der Adreßbus enthält 16 Bit oder 2 Byte. Auf diese Weise kann ein Speicherbereich von 64 K ($64x2^{10}$) Speicherstellen adressiert werden.

Der Prozessor kann für Datenmanipulationen auf insgesamt 208 Bit eines internen Lese-Schreib-Speichers zurückgreifen. Dieser Speicher besteht aus achtzehn 8-Bit-Registern und vier 16-Bit-Registern (Bild 6).

Hauptregistersatz

Der Hauptregistersatz besteht aus 8-Bit-Registern; von diesen sind die Register A und F für spezielle Aufgaben vorgesehen. Die restlichen sechs Register B, C, D, E, H, L werden für allgemeine Verwendung genutzt.

Register A und F: Das Register A, der Akkumulator, enthält nach logischen oder arithmetischen Operationen das Ergebnis. Entstehen bei diesen Operationen besondere Bedingungen (z. B. das Ergebnis ist negativ oder das Ergebnis ist gleich Null oder es wird durch die Operation im Ergebnis die vorgegebene Wortlänge überschritten) oder es werden besondere Operationen ausgeführt, so wird dies in 6-Bedingungs-Flip-Flops des Flagregisters oder F-Registers (flag: Flagge oder Signal) registriert.

Die weiteren sechs 8-Bit-Register des Hauptregistersatzes lassen sich auch in den Paaren BC, DE und HL als 16-Bit-Register verwenden, um 16-Bit-Operationen zu realisieren.

Tauschregistersatz

Der Hauptregistersatz ist im U 880 zweimal vorhanden. Der zweite Registersatz beinhaltet die Register A', F', B', C', D', E',H', L'. Durch einen einfachen Austauschbefehl können die Inhalte der Registersätze komplett vertauscht werden. Dadurch ist es möglich, einen bestimmten Programmabschnitt einem der Registersätze zuzuordnen. Wechselt das Programm, dann können die dazugehörigen Registersätze umgetauscht werden.

Zweckregister

Die Register I, R, PC, IX und IY heißen Zweckregister der CPU, weil ihre Verwendung speziellen Aufgaben vorbehalten bleibt.

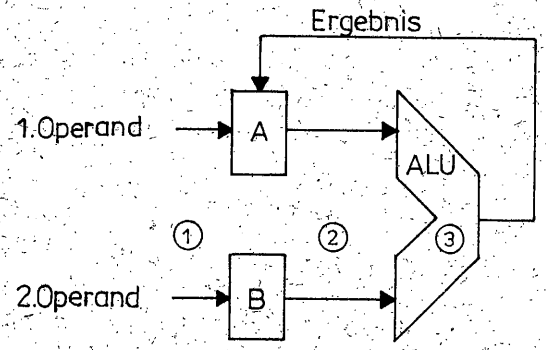
Organisation der Operationsausführung

Jeder Befehl eines Programms veranlaßt bei seiner Ausführung bestimmte Operationen des Rechners. Bei der Computerprogrammierung muß darum die Lösung eines Problems auf ganz konkrete Elementaroperationen zurückgeführt werden. Zur Ausführung dieser Elementaroperationen müssen die entsprechenden Daten in die CPU gebracht und die Ergebnisse wieder wegtransportiert werden. Hierfür gibt es spezielle Operationen

- zum Datentransport (z. B. innerhalb der CPU und zwischen Speicher und CPU)
- zur Datenmanipulation (z. B. arithmetische und logische Operationen)
- zur Steuerung und Beeinflussung des Programmablaufs (z. B. Sprungoperationen).

Der CPU wird über die Adressierung mitgeteilt, wo sich die Daten befinden, mit denen operiert werden soll. Im Computer werden durch den Adreßbus die Speicherzellen angesprochen, aus denen Daten gebraucht werden bzw. in die Daten gebracht werden sollen. Der Datenfluß erfolgt über den Datenbus. Die erforderlichen Signale zur Steuerung der einzelnen Vorgänge fließen über den Steuerbus.

Bei der Ausführung einer Operation werden die Operanden über ihre Adressen aufgerufen und in der CPU verknüpft. Im Bild 7 sind die Operanden in den Registern A und B bereitgestellt. Das Resultat entsteht während des Durchlaufes der Datenwörter durch die ALU und wird nach A gebracht (daher der Registername "Akkumulator"). Von dort kann es wieder zum Speicher transportiert werden.



1. Bereitstellung der Operanden (Daten vom Speicher)
2. Operandentransfer
3. Operation
4. Ergebnistransfer

Bild 7: CPU-interne Operation

2.3.2. Lesen Sie dazu auch im Lb GdA S. 82 ff.!

3. Einführung in die Arbeit mit dem Kleincomputer KC 85/3 bzw. KC 85/2

3.1. Aufbau des Computerarbeitsplatzes und Inbetriebnahme des Computers

Ein Arbeitsplatz im Computerkabinett ist mit einem Kleincomputer, einem Kassettenrecorder und einem Fernsehgerät ausgestattet. Diese Geräte sind entsprechend der Darstellung im Bild 8 miteinander und mit dem Netz verbunden:

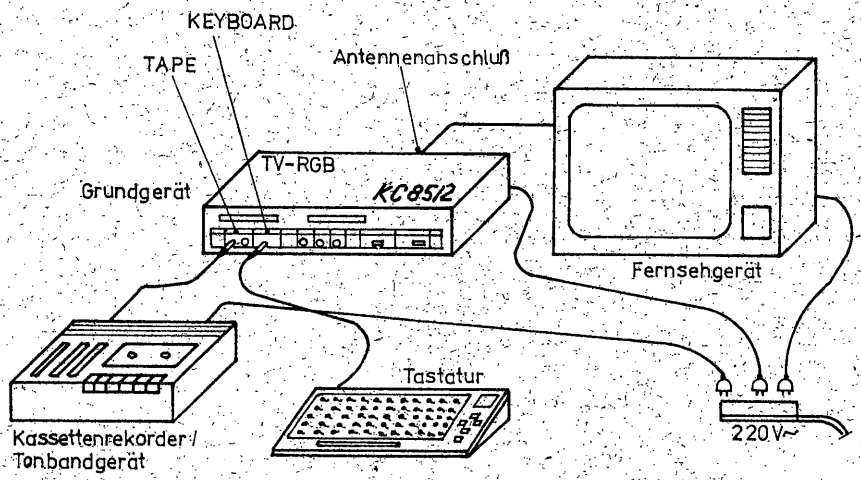


Bild 8: Anschlußschema des Kleincomputers

Die Tastatur wird mit dem Kabel über die Buchse "KEYBOARD" am Grundgerät angeschlossen.
Der Kassettenrecorder wird mit dem Diodenkabel über die Buchse "TAPE" mit dem Grundgerät verbunden.
Das Fernsehgerät wird mit der Leitung, die an der Rückseite des Computers herausgeführt ist, über den Antenneneingang (VHF) mit dem Grundgerät verbunden. Der Kanalwähler am Fernsehgerät ist auf Kanal 8 einzustellen.

D e r C o m p u t e r m u ß s t e t s a l s l e t z t e s G e r ä t e i n - u n d a l s e r s t e s a u s g e s c h a l t e t w e r d e n .

Der Computer wird durch Betätigen der Taste "POWER", rechts am Grundgerät, eingeschaltet. Danach leuchten die rote Netzkontrollanzeige und drei grüne Leuchtdioden, die die Arbeitsbereitschaft der Speicher (ROM, RAM, IRM) anzeigen. Leuchtet keine dieser vier Kontrollampen, muß die Netzverbindung des Computers überprüft werden.

3.1.1. Überprüfen Sie die Richtigkeit der Anschlüsse und schalten Sie die Geräte ein!

Der Computer meldet sich mit folgendem Grundmenü (Bildschirmanzeige) arbeitsbereit:

KC 85/3 bzw. KC 85/2 mit aktiviertem BASIC-Modul

```
*HC-CAOS 3.1*
%BASIC
%REBASIC
%
%SWITCH
%JUMP
%MENU
%SAVE
%VERIFY
%LOAD
%COLOR
%MODIFY
%DISPLAY
%KEYLIST
%KEY
% ■
```

KC 85/2

```
*HC 900-CAOS 2.2*
>SWITCH
>JUMP
>MENU
>SAVE
>VERIFY
>LOAD
>COLOR
>MODIFY
> ■
```

Erscheint dieses Bild nicht, ist die Kanaleinstellung am Fernsehgerät zu überprüfen und gegebenenfalls neu einzustellen. Das Grundmenü gehört zum Betriebssystem CAOS (Cassette Aided Operation System), das die Programme zur Steuerung der angeschlossenen Geräte enthält. Jedes Menüwort beinhaltet jeweils ein Unterprogramm.

BASIC	Kalt-Start des BASIC-Interpreters
REBASIC	Warm-Start des BASIC-Interpreters
SWITCH	Ein- und Ausschalten von Modulen
JUMP	Sprung in ein anderes Betriebssystem
MENU	Aufruf des aktuellen Menüs des Betriebssystems
SAVE	Ausgabe vom Speicherinhalt auf Magnetband
VERIFY	Kontrollesen von auf Magnetband gespeicherten Daten und Programmen
LOAD	Laden des Speichers vom Magnetband
COLOR	Festlegung der Vorder- und Hintergrundfarbe
MODIFY	Speicheranzeige und -veränderung
DISPLAY	Ausgabe von Speicherinhalten auf den Bildschirm
KEYLIST	Auflisten der Funktionstastenbelegung
KEY	Belegung von Funktionstasten

Die aktuelle Position für eine mögliche Eingabe über die Tastatur wird mit dem Cursor (■) angezeigt. Auf der Position des Cursors erscheint das eingegebene Zeichen und der Cursor rückt um eine Position nach rechts weiter. Vom Zeilenende springt er automatisch an den Zeilenanfang der nächsten Zeile. Der Cursor kann mittels der Cursorsteuertasten (↓, ↑, →, ←; in BASIC nur →, ←; vgl. Anlage 1: Tastatur) über den Bildschirm bewegt werden.

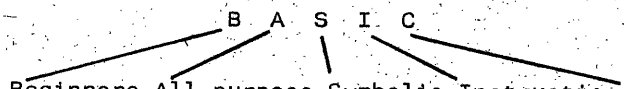
Durch Betätigung der ENTER-Taste (↵) übernimmt der Computer die über Tastatur eingegebenen Zeichen und führt die geforderten Operationen aus. Die ENTER-Taste ist beim Dialog mit dem Computer eine sehr häufig benutzte Taste.

3.1.2. Prägen Sie sich die Lage der Tasten ein! Verwenden Sie dazu auch Anlage 1 und beschriften Sie dort die betreffenden Tasten!

Nach Inbetriebnahme des Computers befindet sich dieser zunächst im Betriebssystemmodus mit dem Betriebssystem CAOS. Durch Eingabe von Anweisungen in der Maschinensprache kann der Mikroprozessor direkt programmiert werden. Die Informationsverarbeitung im Rechner selbst erfolgt immer durch Folgen von Binärzeichen, die jeweils den Wert 1 oder Ø annehmen können. Die Informationsdarstellung durch solche Folgen von Binärzeichen nennt man Maschinensprache. Es entstehen sofort abarbeitungsfähige Programme. Nachteile der Programmierung in Maschinensprache sind, daß ein Maschinenprogramm sehr aufwendig und unübersichtlich ist. Dadurch können sich leichter Programmierfehler ergeben, deren Korrektur sehr schwierig ist. Deshalb wurden sogenannte höhere Programmiersprachen entwickelt, deren Wortschatz der menschlichen Sprache ähnlich ist. Diese Sprachen sind entschieden einprägsamer und leichter handhabbar, weil sich der Programmierer voll auf das zu lösende Problem konzentrieren kann.

3.2. Arbeit mit dem BASIC-Interpreter

Es gibt verschiedene höhere Programmiersprachen, die jeweils für typische Einsatzgebiete entwickelt wurden. Eine leicht erlernbare und deshalb auch sehr verbreitete Sprache ist BASIC. Diese Programmiersprache wurde 1964 am Darthmouth College in Hanover (New Hampshire USA) entwickelt.



(engl.) Beginners All-purpose Symbolic Instruction Code
(deut.) Anfänger All-Zweck symbolischer Befehlscode (Programmiersprache)

Wie der Name sagt, ist BASIC geeignet, sich sehr schnell mit den Grundlagen der Programmierung und dem Umgang mit Computern vertraut zu machen. Mit dem Kleincomputer kann ebenfalls in der Sprache BASIC gearbeitet werden. Dazu ist jedoch ein Programm zur Übersetzung von BASIC-Anweisungen in die Maschinensprache, die der Mikroprozessor verarbeiten kann, erforderlich. Sehr häufig ist dieses Programm ein BASIC-Interpreter (Interpreter = Übersetzer). Er ist entweder im Computer als festes Programm (z. B. in Form eines ROM) installiert oder er muß nach dem Einschalten des Computers geladen werden. Der Interpreter übersetzt das vorhandene Programm zeilenweise in der Reihenfolge der Bearbeitung. Bei jeder Abarbeitungszeit des Programms ist deshalb immer die Übersetzungszeit enthalten. Die Vorteile des Interpreters bestehen in einfachen Korrekturmöglichkeiten und einem geringeren Speicherbedarf.

Bei leistungsfähigen Computern für berufliche Anwendung, für die zur Abarbeitung umfangreicherer Programme größere Geschwindigkeiten verlangt werden, stehen Compiler zur Verfügung. Ein Compiler übersetzt das vorhandene Programm der höheren Programmiersprache insgesamt und einmalig und speichert es in der Maschinensprache. Es kann dann beliebig oft abgearbeitet werden, ohne nochmals übersetzt werden zu müssen. Der Compiler braucht zur Abarbeitung des Programms nicht im Speicher zu sein. Dadurch steht mehr Platz für das eigentliche Programm und für Daten zur Verfügung. Da die Anweisungen nicht wie beim Interpreter während des Programmlaufs übersetzt werden müssen, sind Compiler-Programme wesentlich schneller. Ein Interpreter oder Compiler ist für jede höhere Programmiersprache notwendig und ist für jede der höheren Programmiersprachen verschieden. Für die Arbeit mit dem KC 85/3 bzw. KC 85/2 steht ein leistungsfähiger BASIC-Interpreter zur Verfügung.

Aufruf des BASIC-Interpreters über Steckmodul am KC 85/2

Der BASIC-Modul wird in den rechten Modulschacht des Grundgerätes bei ausgeschaltetem Computer gesteckt. Nach dem Einschalten des Computers wird der Cursor im Grundmenü auf die Anweisung JUMP mittels der Cursorsteuertasten positioniert und die Adresse des Modulsteckplatzes ergänzt:

JUMP Ø8 ↵ (Abschluß der Eingabe erfolgt mit ENTER).

Die Leuchtdiode am Modul leuchtet auf und gleichzeitig wird der ROM-Speicher des Grundgerätes abgeschaltet, dessen Kontrollanzeige erlischt automatisch.

Der KC 85/2 mit aktiviertem BASIC-Modul ist in der Funktionsweise mit der des KC 85/3 identisch, und er meldet sich auch mit dem Grundmenü * HC-CAOS 3,1 *. Die weiteren Tätigkeiten zur Aktivierung des BASIC-Interpreters sind mit denen für den KC 85/3 identisch und im folgenden nachzulesen.

Aufruf des BASIC-Interpreters im Grundgerät des KC 85/3

Im Grundmenü werden

% BASIC
und % REBASIC

angeboten. Durch Positionieren des Cursors auf die Anweisung
% BASIC und Betätigen der ENTER-Taste
wird der BASIC-Interpreter aufgerufen und meldet sich mit der Bildschirmausschrift:

```
HC-BASIC
MEMORY END?  : ■
```

Mit der Eingabe einer dezimalen Speicheradresse, z. B. 32768, kann die obere Grenze des BASIC-Speichers herabgesetzt werden. Das ist jedoch eine Ausnahme und nur notwendig, wenn Speicherplätze für spezielle Funktionen reserviert werden sollen. Für den Normalfall wird die volle Anzahl der Speicherplätze genützt. Nach Betätigen der Entertaste wird die Anzahl der freien Speicherplätze angezeigt:

```
15 Ø 86 BYTES FREE
```

Zugleich meldet der Computer die Arbeitsbereitschaft in BASIC mit der Ausschrift

```
OK
>■
```

Soll der BASIC-Interpreter aus irgendeinem Grund (z. B. Auslesen eines Speicherbereiches) wieder verlassen werden, so geschieht das durch Eingabe der Anweisung BYE. Nach Betätigung der ENTER-Taste kann mit der Anweisung MENU das Grundmenü aufgerufen werden. Es ist jedoch darauf zu achten, daß außer dem Zeilenanfangszeichen % und der Anweisung MENU kein weiteres Zeichen auf dem Bildschirm in dieser Zeile steht. Am Zeilenanfangszeichen kann man die Maschinenebene mit % und die BASIC-Interpreterebene mit > erkennen.

Reagiert der Computer einmal auf keine Eingabe, läßt sich der Cursor nicht bewegen bzw. erscheint er nicht, auch nicht nach Betätigung der ENTER-Taste oder der BRK-Taste, kann ein "Absturz" des Interpreters vorliegen. In diesem Fall ist die RESET-Taste (direkt am Grundgerät) zu betätigen. Damit wird der BASIC-Interpreter verlassen und es erscheint wieder das Grundmenü. Der BASIC-Interpreter kann nun auf zwei Wegen wieder gestartet werden:

1. Durch einen Warm-Start mit der Anweisung REBASIC.
   Dabei bleibt ein vorhandenes BASIC-Programm erhalten.

2. Durch einen Kalt-Start mit der Anweisung BASIC.
   Dabei wird jedoch ein vorhandenes BASIC-Programm gelöscht. (Wurde der Computer abgeschaltet, durch Betätigen der POWER-Taste, ist immer beim ersten Start mit BASIC zu beginnen. Der BASIC-Speicher wird vom Interpreter organisiert und muß aufgeteilt werden.)

Für den KC 85/2 mit BASIC-Modul muß nach Betätigen der RESET-Taste der Modul mit JUMP Ø8↵ erneut aktiviert werden. (Vgl. Anlage 2)

Laden des BASIC-Interpreters von Kassette

Sollte nur das Grundgerät KC 85/2 ohne BASIC-Modul zur Verfügung stehen, so ist der BASIC-Interpreter von Kassette zu laden.

Der Cursor wird dazu im Grundmenü auf die Anweisung LOAD positioniert. Die Interpreterkassette wird in den Kassettenrecorder eingelegt. Der Recorder wird auf Wiedergabe geschaltet und beim Ertönen des Signaltones, der den Programmbeginn anzeigt, wird die ENTER-Taste betätigt. Es beginnt die Übernahme des Programms. Das kann am Bildschirm an Hand der übernommenen Blöcke verfolgt werden (vgl. auch Abschnitt 3.3.).

Ist der Interpreter vollständig und richtig vom Computer übernommen, erscheint auf dem Bildschirm folgendes Bild:

```
MC-BASIC
MEMORY END?:■
```

Damit kann wie beim KC 85/3 bzw. KC 85/2 mit BASIC-Modul weitergearbeitet werden.

3.2.1. Legen Sie sich eine Karteikarte mit der Handlungsfolge (in Stichpunkten) für die Inbetriebnahme des Computers und für den Start des BASIC-Intertreters an!

Arbeit im Direktmodus

Nach dem Start des BASIC-Interpreters kann sofort im Direktmodus gearbeitet werden, d. h. es können Anweisungen eingegeben werden, die nach Betätigen der ENTER-Taste ausgeführt werden.

Es gelten folgende BASIC-Vereinbarungen:

- Eine Null wird als Ø geschrieben
- Für ein Dezimalkomma wird der Dezimalpunkt verwendet
- Führende Nullen können weggelassen werden (z. B. Ø.3 in BASIC .3)
- Die Darstellung der Zahlen erfolgt in Exponentialschreibweise (über 6 Stellen - wie beim Taschenrechner) z. B.:

$3{,}84 \cdot 10^{-20}$ in BASIC 3.84E-2Ø

$0{,}92 \cdot 10^{13}$ in BASIC .92E13

Der Kleincomputer kann nun im Direktmodus genau wie ein Taschenrechner benutzt werden.
Zur Durchführung einer einfachen mathematischen Berechnung ist folgende Eingabe erforderlich:

```
PRINT 177 + 927
```

Die Anweisung PRINT bewirkt, daß das Ergebnis auf dem Bildschirm angezeigt wird. Es ist eine Ausgabeanweisung. Das Gleichheitszeichen entfällt. Im angegebenen Beispiel erscheint nach Betätigen der ENTER-Taste das Ergebnis (11Ø4) auf dem Bildschirm. An Stelle der Anweisung PRINT kann auch ein Fragezeichen verwendet werden.

```
Beispiel: ? 3.22 * 4,78
```

In Anlage 3 sind die zur Verfügung stehenden Standardfunktionen sowie die mathematischen Operatoren aufgeführt. Für die Abarbeitung von verschiedenen Operationen ist folgende Rangfolge festgelegt:

Mathematische Operatoren

1. Klammerausdrücke
2. Potenzieren und Radizieren (^ und SQR (x))
3. Multiplikation und Division (* und /)
4. Addition und Subtraktion (+ und -)

Vergleichsoperatoren

5. < , > , >= , = , <>

Logische Operatoren

6. NOT
7. AND
8. OR

Bei gleichrangigen Operationen ist die Reihenfolge von links nach rechts festgelegt.

Auch mit Zeichenketten können mathematische und logische Operationen ausgeführt werden. Im Abschnitt 5.1. wird darauf näher eingegangen.

Soll der Bildschirm nach den ausgeführten Berechnungen wieder gelöscht werden, so erfolgt das mit der Anweisung CLS und dem Betätigen der ENTER-Taste. Die gleiche Wirkung wird durch gleichzeitiges Drücken der Tasten SHIFT (⇧) und HOME erreicht.

Beim Arbeiten mit dem Computer sind folgende Hinweise zu beachten:

- Der Kleincomputer darf nur zur Nutzung am Fernsehgerät betrieben werden.
- Durch das Ein- und Ausschalten des Kassettenrecorders entstehen Störimpulse. Es ist deshalb keine Schaltung der Netzspannung des Recorders vorzunehmen, wenn die Verbindung Recorder - Kleincomputer über das Diodenkabel besteht.
- Die Lüftungsschlitze an den Geräten dürfen nicht bedeckt werden, um die notwendige Kühlung zu sichern.
- Da die Aufzeichnungsdichte der Programme und Daten hoch ist, ist darauf zu achten, daß sich das Kassettenmagnetbandgerät in einem einwandfreien technischen Zustand befindet und daß nur Magnetbandkassetten ohne Klebe- und Knitterstellen verwendet werden.
- Die Kassetten sind nicht auf dem Kassettenrecorder oder dem Fernsehgerät abzulegen, da die gespeicherten Daten und Programme durch magnetische Beeinflussungen zerstört werden können.

3.2.2. Lösen Sie folgende Aufgaben im Direktmodus des BASIC-Interpreters: (Suchen Sie die kürzeste Schreibweise dieser Aufgaben!)

1. a) $23 - 61$ b) $42 \cdot 0{,}3$ c) $12 : 1{,}1$

2. a) $\frac{12{,}1 + 20}{47 - 63{,}1}$ b) $\frac{1}{12{,}3 \cdot 17}$ c) $\frac{0{,}25 \cdot 66}{4}$

3. a) $\sin \frac{\pi}{2}$ b) $\sqrt{\frac{2{,}2^2}{4} - 1}$ c) $\frac{1}{\sqrt[3]{0{,}3 \cdot 12}}$

3.2.3. Nutzen Sie das Arbeitsblatt (Anlage 1) und prägen Sie sich die Lage und Funktion der einzelnen Tasten ein! Vervollständigen Sie die Beschriftung des Arbeitsblattes!

3.3. Eingeben, Starten und Speichern von BASIC-Programmen

Den Nutzern von Computern werden in der Regel die von Experten erarbeiteten Programme (Software) zur Verfügung gestellt. Es wäre sehr zeitaufwendig bzw. teilweise unmöglich, die Programme immer neu über die Tastatur einzugeben. Die Anwenderprogramme sind deshalb auf maschinenlesbaren Datenträgern gespeichert. Im Falle der Kleincomputer stehen die Programme auf Magnetbandkassetten zur Verfügung. Der Anwender hat die Aufgabe, die Programme ordnungsgemäß einzuladen sowie die erforderlichen Daten korrekt bereitzustellen. Das verlangt ein gewissenhaftes und fehlerfreies Arbeiten, Kenntnisse über den Umgang mit den Datenträgern und Fähigkeiten in der Bedienung des Computers und der peripheren Geräte. In Anlage 7 sind wichtige BASIC-Anweisungen für die Arbeit mit dem Computer zusammengestellt.

Programmeingabe von Kassette

Zum Laden von BASIC-Programmen ist die Anweisung

```
CLOAD "Programmname"
```

einzugeben. Die Programmkassette wird in den Recorder eingelegt und mit Hilfe des Bandzählwerkes wird das Band auf den Programmbeginn gebracht. Beim Ertönen des Signaltons wird die ENTER-Taste betätigt und die Programmübernahme beginnt. Der Name des gefundenen Programms wird auf den Bildschirm geschrieben und stimmt er mit dem nach CLOAD geschriebenen Namen überein, erfolgt die Übernahme. Zur Kontrolle werden die Blocknummern angezeigt (Hexadezimal). Wird der Datenblock (jeweils 128 Bytes) vom Computer fehlerfrei erkannt, erscheint nach jeder Blocknummer das Zeichen>(z. B. 1Ø>) auf dem Bildschirm. Wird die Blocknummer mit einem Fragezeichen verbunden gedruckt (z. B. ?1Ø), liegt ein Fehler vor und das Magnetband muß kurz zurückgespult werden. Das Sternchen mit der Blocknummer (z. B. *1Ø) weist darauf hin, daß der Datenblock nicht in den Speicher übernommen wird. Das geschieht, wenn die Blocknummer bereits

vorhanden ist (nach dem Rückspulen des Bandes) und wenn ein Block in der Reihenfolge fehlt (nach einem Block mit Fragezeichen).

Fehlermeldungen (vgl. auch Anlage 4):

IO-ERROR - Es wurde ein falscher Programmname eingegeben.
. Der Programmname ist zu korrigieren (neue Eingabe - Ausschrift auf dem Bildschirm beachten: SSSName).
. Das Magnetband ist auf richtige Positionierung zu überprüfen (falsches Programm).

- Es ist kein BASIC-Programm (an der Stelle von SSSName stehen andere Zeichen bzw. sofort der Name).

OM-ERROR - Der vorhandene Speicherplatz reicht nicht aus.
. Es wurde zuvor der Stringspeicherbereich vergrößert. Mit CLEAR255 wird der Normalzustand wieder hergestellt.
. Es ist ein Zusatzmodul (RAM) erforderlich. Das Programm kann ohne diesen nicht genutzt werden.

Programmstart

Ist der Ladevorgang beendet, meldet sich der Computer mit FF (FILE FOUND) und OK. Das Kassettenmagnetbandgerät kann gestoppt werden.

Das Programm kann nun mit der Anweisung RUN und dem nachfolgendem Betätigen der ENTER-Taste gestartet werden. Die Programmabarbeitung wird auch als Programmodus bezeichnet.

Programmunterbrechung

Wird es erforderlich, die Programmabarbeitung zu unterbrechen, z. B. zur längeren Betrachtung einer Bildschirmanzeige, so kann das durch Betätigen der Taste STOP erfolgen. Soll die Programmabarbeitung fortgesetzt werden, ist die Taste "Cursor down" (↓) zu drücken.

Eine Unterbrechung des Programms wird auch erreicht durch Betätigen der BRK-Taste. In diesem Fall zeigt der Computer an, in welcher Programmzeile die Programmabarbeitung unterbrochen wurde. Das ist für Programmveränderungen vorteilhaft. Es kann danach wieder im Direktmodus gearbeitet werden oder das Programm durch Eingabe der Anweisung CONT fortgesetzt werden. Ein neuer Start des Programms erfolgt wieder mit RUN.

3.3.1. Tragen Sie die Funktion der neu kennengelernten Tasten in das Arbeitsblatt (Anlage 1) ein!

3.3.2. Ergänzen Sie in Anlage 2 die Anweisungen, wie Sie vom Direktmodus in den Programmodus gelangen und umgekehrt!

Programmeingabe über Tastatur

Für die Eingabe eines Programms über die Tastatur sind einige Kenntnisse über den Aufbau eines BASIC-Programms erforderlich.

Ein Programm besteht aus Programmzeilen, die mit Zeilennummern gekennzeichnet sind. Diese Programmzeilen werden vom Computer nacheinander abgearbeitet. Damit ist die aufsteigende Zeilennummerierung für die Reihenfolge der Abarbeitung der Anweisungen entscheidend.

Eine Programmzeile besteht aus der Zeilennummer N (N kann zwischen Ø und 65529 liegen) und einer bzw. mehreren BASIC-Anweisungen. Werden mehrere BASIC-Anweisungen verwendet, was anfangs jedoch aus Gründen der Übersichtlichkeit vermieden werden sollte, so sind diese durch Doppelpunkt voneinander zu trennen.

Beispiel: 1Ø CLS : PRINT "HALLO"

Eine Programmzeile darf maximal 72 Zeichen lang sein (fast zwei Bildschirmzeilen).

Eine eingegebene Programmzeile wird mit Betätigen der ENTER-Taste beendet und damit zunächst im Programmspeicher abgelegt. Die Anweisungen werden nicht sofort ausgeführt, sondern erst nach dem Programmstart.

3.3.3. Welche Anweisung muß für den Programmstart eingegeben werden?

Die Anweisung AUTO erleichert die Eingabe. Es werden dabei durch den Computer die Zeilennummern in Zehnerschritten bereitgestellt. Das Eingeben des Programms in Zehnerschritten ermöglicht ein problemloses Einfügen neuer Programmzeilen im Falle einer Überarbeitung bzw. Erweiterung des Programms. Nach Eingabe einer Programmzeile im AUTO-Modus (mit: AUTO Zeilennummer, kann der Beginn der automatischen Zeilennummerierung frei gewählt werden) wird diese mit der ENTER-Taste vom Computer übernommen und die nächste Zeilennummer bereitgestellt. Nach Eingabe der letzten Zeile und Betätigen der ENTER-Taste kann der AUTO-Modus mit der BRK-Taste wieder verlassen werden.

Die gespeicherten Programmzeilen können mit Hilfe der Anweisung LIST auf dem Bildschirm wieder sichtbar gemacht werden.

Nach dem Starten des Programms mit der Anweisung RUN liest der Interpreter die Programmzeilen der Reihe nach, übersetzt Zeile für Zeile und stellt das Programm zur Abarbeitung bereit. Findet der Interpreter bei der Übersetzung einen Fehler, so erfolgt eine Fehlermeldung mit Angabe des Fehlertyps und der Zeilennummer, in der der Fehler auftrat. (Vergleiche dazu Anlage 4!)

3.3.4. Führen Sie folgende Übungen am Computer aus:

a) AUTO (ENTER)
```
1Ø CLS
2Ø PRINT
3Ø PRINT "KC 85/3"
4Ø END
BRK-Taste
```

b) Geben Sie das Programm mit AUTO 15Ø ein!

c) Sehen Sie sich das Programm gelistet an und starten Sie das Programm!

Korrekturmöglichkeiten

Bei jeder Eingabe über die Tastatur können Fehler auftreten. Um diese möglichst schnell beheben zu können, besitzt der BASIC-Interpreter verschiedene Korrektur- bzw. Editiermöglichkeiten. Während der Eingabe über die Tastatur, bevor die ENTER-Taste gedrückt wurde, können innerhalb einer BASIC-Zeile folgende Möglichkeiten genutzt werden:

Tastenbetätigung	Funktion
←	Cursor nach links
→	Cursor nach rechts
↕ ← (gleichzeitig)	Cursor an den Zeilenanfang
↕ DEL "	Zeile löschen
↕ HOME "	Bildschirm löschen
DEL	Zeichen, auf dem der Cursor steht, löschen und Zeile verdichten
INS	Leerzeichen an der Stelle, an der der Cursor steht, einfügen

Durch Überschreiben können Fehler ebenfalls beseitigt werden. Erscheint nach dem Programmstart mit RUN eine Fehlermeldung, dann wird vom Computer die Programmzeile angegeben (z. B. SN-ERROR in 2Ø). In diesem Fall kann die zu korrigierende Programmzeile mit der Anweisung EDIT Zeilennummer aufgerufen werden. Der Cursor befindet sich dann am Ende der aufgerufenen Programmzeile

und es können die oben genannten Editiermöglichkeiten genutzt werden. Durch Betätigung der ENTER-Taste wird die Programmzeile in der veränderten Form in den Programmspeicher übernommen und die folgende Programmzeile zur Änderung bereitgestellt. Muß in dieser Zeile keine Veränderung vorgenommen werden, kann durch die BRK-Taste der Editiermodus verlassen werden.

Weitere Korrekturmöglichkeiten sind:

- Eingabe der Zeilennummer + ENTER-Taste: löscht die genannte Zeile.
- Eingabe der Zeilennummer und der neuen Programmzeile + ENTER-Taste: überschreibt die alte Programmzeile.
- Eingabe einer neuen unbelegten Zeilennummer mit Anweisungen + ENTER-Taste: Die neue Zeile wird automatisch in das Programm entsprechend der Zeilennummer eingefügt.
- Die Anweisung DELETE n, m löscht alle Programmzeilen von der ersten Zeilennummer n bis zur letzten Zeilennummer m, also ganze Programmabschnitte. Beispiel: DELETE 5Ø, 1ØØ (Alle Zeilen zwischen 5Ø und 1ØØ, einschließlich der beiden, werden gelöscht.)

3.3.5. Geben Sie folgendes fehlerhafte Programm ein, und korrigieren Sie auf verschiedene Weise!

```
1Ø CLS PRINT
2Ø PRINT"KORREKTURUEBUNG"
3Ø END
```

Speichern von Programmen auf Magnetbandkassette

Zur Speicherung eines im Speicher des Computers vorhandenen BASIC-Programms auf Magnetbandkassette ist die Anweisung

```
CSAVE "Name"
```

einzugeben. Der "Name" des Programms darf maximal acht Zeichen lang sein, z. B. CSAVE "MITTEL". In den Kassettenrecorder wird die Magnetbandkassette eingelegt und der Recorder auf Aufnahme eingestellt. Erst wenn das Band läuft, kann durch Betätigen der ENTER-Taste die Übernahme beginnen. Die Blocknummern werden auch beim Abspeichern von Programmen zur Kontrolle angezeigt. Es ist darauf zu achten, daß auf der Magnetbandkassette schon vorhandene Daten und Programme nicht gelöscht werden. Deshalb ist eine genaue Buchführung über die aufgenommenen Programme unbedingt erforderlich.

Nach dem Abschluß der Übernahme erscheint auf dem Bildschirm die Frage VERIFY? (Y) zur Kontrolle des gespeicherten Programms. Durch Zurückspulen der Kassette und dem Betätigen der Y-Taste beim Kennton werden die gespeicherten Blöcke überprüft. Werden Fehler signalisiert (Blocknummer mit Fragezeichen), muß der Speichervorgang wiederholt werden. Der Abschluß des Vergleichs ist mit OK auf dem Bildschirm zu erkennen. Wird keine Überprüfung gewünscht, ist eine beliebige Taste, außer Y, zu drücken.

In der Praxis der Computeranwendung hat sich bewährt, an Stelle des Kontrollesens eine zweite Aufzeichnung des Programms anzufertigen.

3.3.6. Legen Sie sich je eine Karteikarte für das Vorgehen beim Laden und Speichern von Programmen mit Magnetbandkassetten und für die Editiermöglichkeiten an!

3.3.7. Ergänzen Sie im Arbeitsblatt (Anlage 2) die Anweisungen, mit denen der Programmodus, der Editiermodus, der Zeilennumerierungsmodus und der Programmlistendruckmodus aus dem Direktmodus erreicht werden kann und wie in diesen wieder zurückgekehrt werden kann!

4. Grundlagen der Programmierung

Soll eine Aufgabe, ein Problem mit Hilfe eines Computers gelöst werden, muß zunächst eine Folge von Anweisungen, ein Algorithmus in Form eines Programms erarbeitet und eingegeben werden. Das Programm enthält die speziellen Lösungsvorschriften, die den Computer erst in die Lage versetzen, das gewünschte Ergebnis zu liefern.

Die Erarbeitung dieser Programme wird Programmierung oder Programmentwicklung genannt. Die Art und Weise des Vorgehens der Programmierung ist die Technologie der Programmierung. Das Ziel besteht darin, mit möglichst geringem Aufwand Programme mit hoher Qualität, Zuverlässigkeit, Bedienfreundlichkeit und Wartungsfreundlichkeit zu entwickeln. In der Praxis gewinnt gegenwärtig die strukturierte Programmierung als eine Technologie der Programmierung (Softwarherstellung) an Bedeutung.

Sie ist durch folgende Merkmale gekennzeichnet:

- Die Erarbeitung des Programms erfolgt in festgelegten Arbeitsschritten (Stufen der Programmierung). Mit jedem Arbeitsschritt werden bestimmte Voraussetzungen für die Lösung des Problems geschaffen.
- Der dem Programm zugrunde liegende Lösungsalgorithmus wird immer mehr zergliedert und verfeinert, so daß in sich abgeschlossene Bausteine entstehen (top down und buttom up Methode).
- Zur Darstellung des Lösungsalgorithmus werden standardisierte Algorithmenbausteine verwendet.

Das Vorgehen nach der strukturierten Programmierung, also einem hierarchischen Aufbau der Programme vom komplexen zum einzelnen Problem, bringt mehrere Vorteile. Die Programme sind sehr übersichtlich und änderungsfreundlich. So kann auch ein anderer Programmierer sich leicht in das Programm hineinfinden, um z. B. Erweiterungen vorzunehmen. Durch das systematische, schrittweise Vorgehen bei der Programmierung werden Fehler weitgehend ausgeschlossen bzw. die Testung des Programms und die Fehlersuche erleichtert.

4.1. Arbeitsschritte der Programmentwicklung

Bei der Erarbeitung von Programmen müssen prinzipiell zwei Fragen beantwortet werden:
WAS? soll der Computer leisten und
WIE? soll der Computer dabei vorgehen.
In der Praxis hat sich bei der Entwicklung von Programmen und der Beantwortung dieser Fragen folgende Schrittfolge, auch Stufen der Programmierung genannt, bewährt:

1. Aufgabenstellung und Problemanalyse
2. Programmentwurf - Aufstellen der algorithmischen Grobstruktur und Verfeinerung
3. Umsetzung des Programmentwurfs in eine bestimmte Programmiersprache
4. Testung des Programms
5. Dokumentation

Diese Schrittfolge sollte auch bei der Entwicklung eigener Programme angewendet werden. Aus diesem Grund wird auf das Wesen der einzelnen Schritte im Folgenden noch genauer eingegangen.

Aufgabenstellung und Problemanalyse

Bei der Formulierung der Aufgabenstellung, der Beantwortung der Frage WAS ? soll das Programm leisten, ist bereits große Aufmerksamkeit und Genauigkeit erforderlich. Das zu lösende Problem muß genau beschrieben werden. Dabei sind alle Anforderungen, die vom künftigen Nutzer an das Programm gestellt werden, zu erfassen. Alle Daten und Parameter, die in das Programm eingehen, sind zu berücksichtigen. Je genauer die Aufgabenstellung formuliert wird, umso besser kann das zu erarbeitende Programm die gestellten Anforderungen erfüllen und das gewünschte Ergebnis liefern.

Die Aufgabe muß auch daraufhin analysiert werden, ob sie durch einen Computer zu lösen ist und welcher Typ zur Verfügung steht. Dabei spielen z. B. Speicherart (Kassette, Diskette), Speicherumfang, Programmiersprache und Anforderungen an die Rechenzeit eine Rolle. Auch Überlegungen zum mathematischen Lösungsweg gehören zur Problemanalyse sowie auch Überlegungen zur Gestaltung der Ein- und Ausgabe der Daten.

Programmentwurf

Der Entwurf des Algorithmus für ein Programm erfolgt in mehreren Schritten, wobei er immer verfeinert wird. Die Erarbeitung des Lösungsalgorithmus entspricht der Beantwortung der Frage WIE ? soll das Problem gelöst werden.

Allgemein versteht man unter einem Algorithmus eine Folge eindeutig formulierter Anweisungen bzw. Vorschriften, die nach einer bestimmten Reihenfolge auszuführen sind und zur Lösung aller Aufgaben eines bestimmten Typs führen.

Beim Algorithmieren, dem Entwerfen eines Algorithmus, sind entsprechend der Definition folgende Merkmale zu beachten:

- die Anweisungen eines Algorithmus sind eindeutig,
- die Anweisungen sind in einer Folge angeordnet und
- die Reihenfolge der Ausführung der Anweisungen ist angegeben.

Wenn der Algorithmus durch einen Computer abgearbeitet werden soll, so muß er als eine Folge von Anweisungen geschrieben werden, die vom Computer verarbeitet werden kann.
Einen in einer bestimmten Programmiersprache geschriebenen Algorithmus nennt man Programm.
Zu dem in einer Programmiersprache geschriebenen Algorithmus gelangt man vielfach schrittweise. Entsprechend dem Vorgehen bei der strukturierten Programmierung wird zunächst in einem hierarchischen Entwurf das zu lösende Problem in Teilprobleme oder Teilaufgaben zerlegt. Die dabei entstehenden Module können unabhängig voneinander und bei sehr umfangreichen Problemen auch von verschiedenen Bearbeitern weiter verfeinert, algorithmiert und schließlich in ein Programm umgesetzt werden. Bei der Algorithmierung wird im allgemeinen zunächst eine sprachliche (umgangssprachliche) Formulierung des Algorithmus vorgenommen. Diese wird in Abhängigkeit vom Umfang und von der Komplexität des zu lösenden Problems in weiteren Schritten immer mehr verfeinert und vervollkommnet. Danach wird der Algorithmus in einer übersichtlichen Form dargestellt. In der Praxis werden dabei vor allem drei Formen der Darstellung des Algorithmus im Programmentwurf verwendet:

- Grafische Darstellung in Form eines Programmablaufplanes
  (festgelegte Symbole werden durch Linien, die den Programmablauf kennzeichnen miteinander verbunden).

- Grafische Darstellung in Form eines Struktogramms
  (Strukturblöcke werden aneinandergesetzt, nebeneinandergestellt, oder verschachtelt).

- Verbale Formulierung der einzelnen Algorithmenstrukturen
  (festgelegte Vokabeln (Pseudocode) werden zur Kennzeichnung der Algorithmen verwendet).

Besonders die Darstellung in Form von Struktogrammen unterstützt die strukturierte Programmierung.

Umsetzung des Programmentwurfs in eine Programmiersprache

Bei diesem Schritt der Programmierung erfolgt die Umsetzung des im Programmentwurf festgelegten Algorithmus in eine ganz bestimmte Programmiersprache. Die Aufgabe verlangt also die Kenntnis einer konkreten Programmiersprache. Im Informatikunterricht wird dabei die höhere Programmiersprache BASIC angewendet.

4.1.1. Informieren Sie sich im Lb GdA, Seite 88 über die verschiedenen Arten der Programmiersprachen!

Im folgenden Beispiel werden die bisher beschriebenen Schritte der Programmentwicklung noch einmal deutlich gemacht:

Aufgabenstellung und Problemanalyse

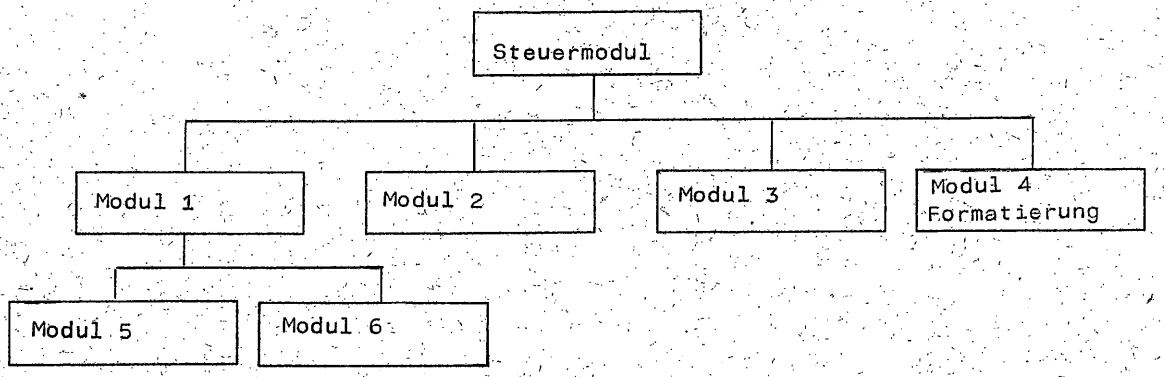
Es ist ein Programm zu entwickeln, welches eingegebene Festkommazahlen so formatiert, daß die Dezimalpunkte der Zahlen genau untereinander stehen (auf der Bildschirmspalte 21).

Bei Eingabe einer Ø soll auf dem Bildschirm "Ø.ØØ" erscheinen und bei Zahlen, die kleiner sind als Ø,Ø1 oder größer als 999999, die Ausschrift: "Festkommadarstellung nicht möglich!"
Es ist zu beachten, daß eine Stelle für das Vorzeichen automatisch reserviert wird. Der Bildschirm soll vor der Abarbeitung des Programms gelöscht werden. Weiterhin ist zu beachten, daß die Spaltennumerierung bei Ø beginnt. Das Programm soll immer wieder zur Eingabeaufforderung zurückkehren.

Das Programm soll als Unterprogramm in größeren Programmen z. B. für tabellarische Zahlendarstellungen nutzbar sein.

Programmentwurf

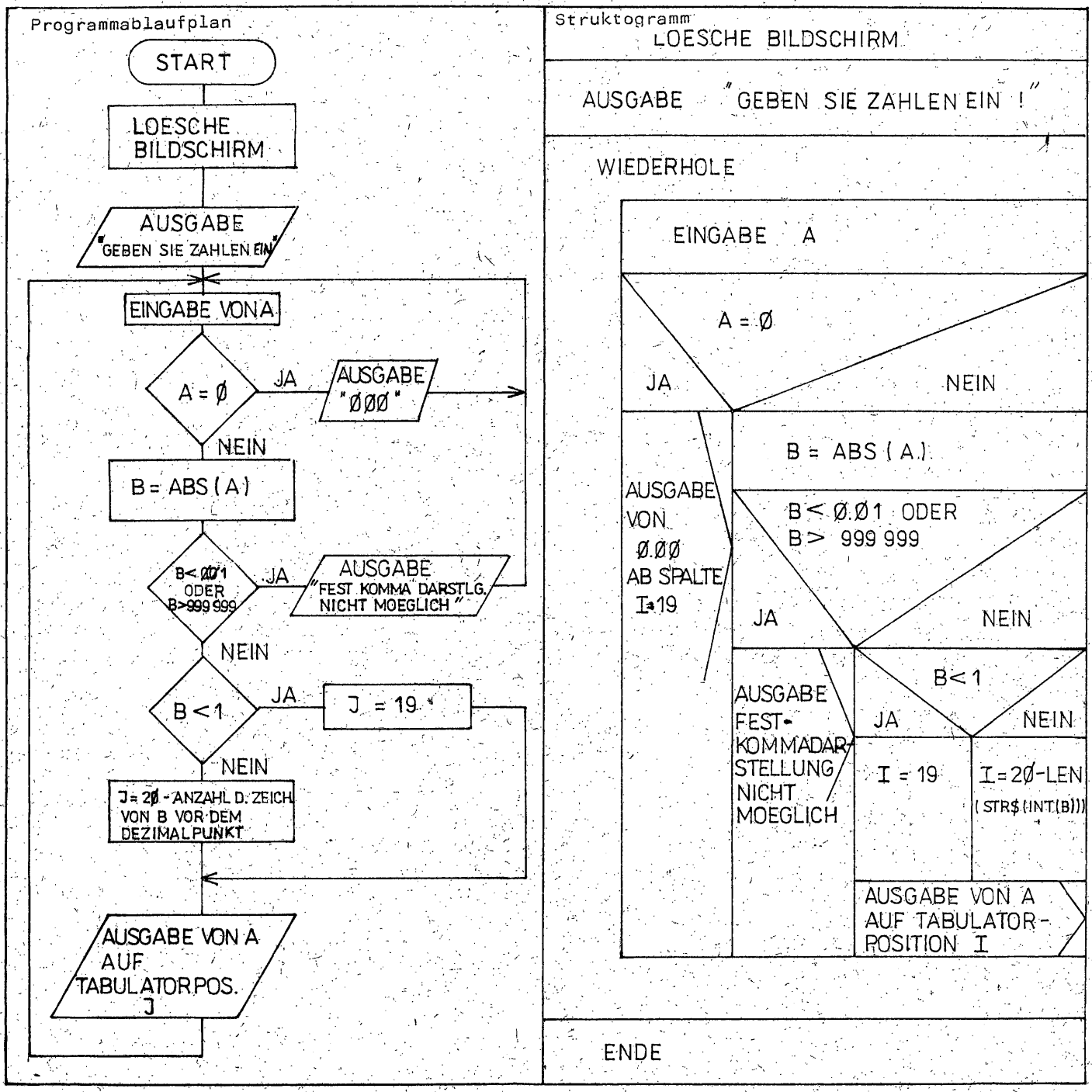
Ein hierarchischer Entwurf des Lösungsalgorithmus ist für dieses kleine Problem nicht notwendig. Es wäre jedoch als ein Modul in einem größeren Programm denkbar.



- Umgangsprachliche Formulierung des Algorithmus:

1. Beginn, Bildschirm löschen
2. Wiederhole
3. Eingabe : Beliebige Zahl A
4. Falls A = Ø
   Dann Ausgabe "Ø.ØØ" ab Spalte I=19
   Zurück zur Eingabe
5. Bildung des Absolutbetrages von A
   B = Abs(A)
6. Falls B < Ø.Ø1 oder B > 999 999
   Dann Ausgabe "Festkommadarstellung nicht möglich!"
   Zurück zur Eingabe
7. Falls B < 1
   Dann I = 19
   Sonst I = (2Ø-(nächstkleinere ganze Zahl in B))
8. Ausgabe von A ab Spalte I
9. Zurück zur Eingabe
1Ø. Ende

- Grafische Darstellung des Algorithmus als Programmablaufplan oder Struktogramm:
  (Hier werden beispielhaft beide Varianten vorgestellt.)



Umsetzung in die Programmiersprache BASIC

```
1Ø CLS
2Ø PRINT:PRINT "GEBEN SIE ZAHLEN EIN!"
3Ø INPUT A
4Ø IF A=Ø THEN PRINT TAB(19)"Ø.ØØ": GOTO 3Ø
5Ø B = ABS(A)
6Ø IF B< Ø.Ø1 OR B > 999999 THEN 9Ø
7Ø IF B< 1 THEN I=19:ELSE I=2Ø-LEN(STRØ(INT(B)))
8Ø PRINT TAB(I)A : GOTO 3Ø
9Ø PRINT "FESTKOMMADARSTELLUNG NICHT MOEGLICH !":GOTO 3Ø
1ØØ END
```

Testung des Programms

Bei der Testung des Programms kommt es darauf an, möglichst alle Fehler zu ermitteln, zu beseitigen und zu prüfen, ob das Programm den gestellten Anforderungen genügt.

Bei der Fehlersuche sind vor allem

- syntaktische Fehler (Verstoß gegen die Sprachvereinbarung) und
- logische Fehler (es wird nicht das erwartete Ergebnis erreicht),

zu korrigieren.

Syntaktische Fehler werden vom Rechner angezeigt. Der KC z. B. meldet sich dann mit "SN - ERROR" (Syntax-Fehler). Vergleichen Sie dazu auch Anlage 5.

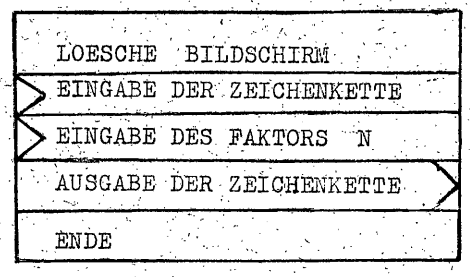
Logische Fehler sind meist schwieriger aufzudecken. Durch die Eingabe von Daten, die leicht zu kontrollieren und z. B. mit dem Taschenrechner nachzurechnen sind, können logische Fehler gefunden werden.

Je umfangreicher das Programm ist, desto komplizierter sind die Fehler zu finden. So kann es in Ausnahmefällen vorkommen, daß durch den Programmtest nicht alle Fehler aufgedeckt werden. Wenn Fehler erst nach längerer Nutzung des Programms auftreten, müssen sie durch eine sogenannte Software-Wartung beseitigt werden.

Bei der Testung des Programms ist auch zu prüfen, ob das Programm "absturzsicher" ist. Das heißt, daß der Computer trotz einer fehlerhaften Eingabe weiterarbeitet und auf Eingaben reagiert. Für diesen Fall müssen sogenannte Fluchtfunktionen eingebaut werden. Am folgenden Beispiel soll das demonstriert werden.

Beispiel: Eine eingegebene Zeichenkette soll N-mal vervielfacht werden.

Struktogramm



BASIC-Programm

```
1Ø CLS
2Ø INPUT "ZEICHENKETTE EINGEBEN:"; A$
3Ø INPUT "FAKTOR N:" ; N
5Ø PRINT STRING $ (N,A$)
6Ø PRINT : PRINT
8Ø END
```

Nach dem Start mit der Anweisung RUN wird die Zeichenkette "HALLO" und der Faktor 1ØØ eingegeben. Auf dem Bildschirm erscheint daraufhin "LS-ERROR", weil eine Zeichenkette maximal 255 Zeichen lang sein darf. Das Programm wird unterbrochen.

Der Einbau der folgenden Fluchtfunktion ermöglicht eine höhere Zuverlässigkeit des Programms und verhindert, daß das Programm abgebrochen wird.

```
4Ø E=LEN (A$) : F = N * E : IF F > 255 THEN 7Ø
7Ø PRINT "AUSSERHALB MEINES BEREICHES!" : GOTO 2Ø
```

Dokumentation

Die Dokumentation eines Programms soll die problemlose Nutzung durch beliebige Anwender ermöglichen. Dazu wird in der Praxis eine Anwenderdokumentation erarbeitet. Sie enthält die Hinweise, die ein Nutzer des jeweiligen Programms benötigt, um das Programm zu starten und um mit ihm zu arbeiten.

Gleichzeitig wird eine Programmdokumentation erarbeitet. Sie enthält Hinweise zum Programm selbst, zum Beispiel Informationen über die verwendeten Dateien, über Feldvariablen, das Struktogramm u. a. Diese Dokumentation wird vor allem bei Veränderungen am Programm benötigt, d. h. bei der Wartung der Software.

Die Dokumentation erfolgt nicht am Schluß der Programmentwicklung. Sie wird in den einzelnen Arbeitsschritten erstellt und immer mehr angereichert.

In eine Dokumentation gehören neben dem Namen des Programms auch Angaben zum Einsatzgebiet, zum Rechnertyp, zum Inhalt, zur Programmiersprache, zur Handhabung des Programms u. a.

4.1.2. Erläutern Sie die Vorgehensweise bei der Entwicklung eines Programms!

4.1.3. Was ist das Wesentliche in den einzelnen Stufen der Programmierung?

4.2. Algorithmische Grundstrukturen

Jeder Algorithmus besteht aus einer oder mehreren verschiedenen algorithmischen Grundstrukturen. Durch die Verknüpfung dieser Grundstrukturen kann jeder Algorithmus realisiert werden. Bereits in dem relativ einfachen Programm des Beispiels im vorangegangenen Abschnitt sind die drei algorithmischen Grundstrukturen enthalten:

- die Reihung (Sequenz) als eine Aneinanderreihung einzelner Arbeitsschritte
- die Auswahl (Selektion) als eine Alternative oder Fallentscheidung und
- die Wiederholung (Zyklus) als eine Schleife.

Für die Darstellung dieser algorithmischen Grundstrukturen wird im folgenden die grafische Form der Struktogramme gewählt, da diese besonders die strukturierte Programmierung unterstützt. Nach den Entwicklern dieser Beschreibungsmittel werden sie auch NASSI - SHNEIDERMAN - Diagramme genannt.

Die Struktogramme bestehen aus Strukturblöcken, die aneinandergereiht, nebeneinandergestellt oder ineinandergeschachtelt werden können. Die Strukturblöcke haben folgende Eigenschaften:

- Jeder Block bildet eine abgeschlossene funktionale Einheit.
- Jeder Block besitzt nur einen Eingang und einen Ausgang.
- Der Steuerfluß verläuft von oben nach unten.

Damit ergeben sich für das Zusammensetzen der Strukturblöcke zwei grundlegende Regeln:

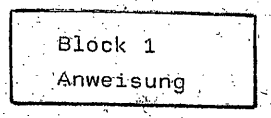
1. Ein Block ist entweder vollständig in einem anderen enthalten oder befindet sich außerhalb von diesem. Zwei Blöcke dürfen sich nicht überlappen.
2. In einem Struktogramm erhält ein Block die Steuerung nur von dem ihm übergeordneten Block und gibt sie dorthin wieder zurück. Er kann sie weitergeben (und wieder zurückerhalten), jedoch nur an ihm untergeordnete Blöcke.

Der einfachste Strukturblock ist der Elementarblock, der nur aus einer Anweisung besteht.

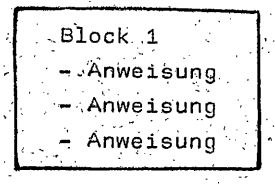
Reihung

Die algorithmische Grundstruktur Reihung ist die einfachste Grundstruktur. Sie besteht aus einer Folge von Anweisungen, die nacheinander abgearbeitet werden. Dabei kann jede Anweisung einem Strukturblock entsprechen (Elementarblöcke) oder es werden mehrere Anweisungen zu einem Strukturblock zusammengefaßt. Damit werden lineare Programmverläufe realisiert.

Elementarblock



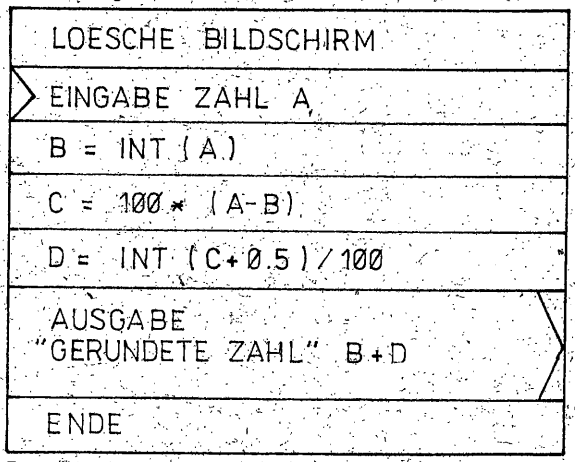
Zusammengefaßter Strukturblock



Diese einfache Grundstruktur ist die Basis für lineare Programme. Es gibt nur wenige Programme, die ausschließlich diese Grundstruktur enthalten. Sehr viel häufiger tritt diese Grundstruktur als ein Teil eines Programms auf.

Beispiel: Es ist ein Programm zu erstellen, welches eine eingegebene Festkommazahl auf zwei Stellen hinter dem Komma rundet.

Struktogramm:



BASIC - Programm:

```
10 CLS
20 INPUT "ZAHL EINGEBEN:";A
30 B=INT(A)
40 C=100*(A-B)
50 D=INT (C+0.5)/100
60 PRINT:PRINT"GERUNDETE ZAHL="; B+D
70 PRINT:PRINT
80 END
```

In der Praxis wird diese Rundung mit nur einer Programmzeile realisiert:

```
B=INT (A*100 + .5)/100
```

Damit wird nur eine Variable benötigt. Zu viele Variablen würden den Speicher belasten und die Effektivität eines Programms verringern.

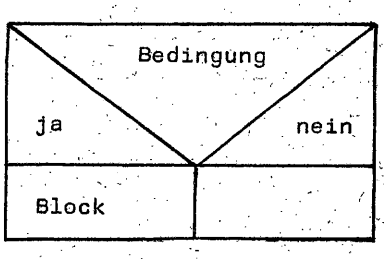
4.2.1. Sehen Sie sich verschiedene gelistete Programme an, und versuchen Sie, diese Grundstruktur in den Programmen zu erkennen! (Besonders in den Programmen Telefon und Musik!)

Auswahl

Die algorithmische Grundstruktur Auswahl ist die Basis für Programme mit Verzweigungen, für die Arbeit mit Unterprogrammen und die Menütechnik. In der Grundstruktur Auswahl kann zwischen einer, zwei oder mehreren Möglichkeiten des weiteren Programmverlaufs ausgewählt werden. Die Entscheidung für die Auswahl wird immer auf der Basis einer vorgegebenen Bedingung getroffen. Je nachdem, ob eine Bedingung erfüllt ist (die Aussage wahr ist) oder nicht erfüllt ist (die Aussage falsch ist), wird mit den Anweisungen in dem dazugehörigen Block das Programm fortgesetzt.

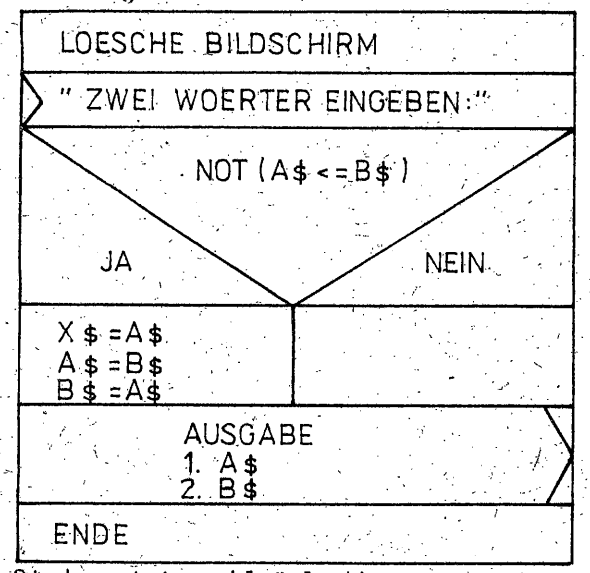
Unter der Bedingung, daß nur eine Auswahlmöglichkeit gegeben ist (unvollständige Alternative), spricht man auch von dem bedingten Sprung. Die Anweisungen im anschließenden Block werden nur dann ausgeführt, wenn die Bedingung erfüllt ist. Ist die Bedingung nicht erfüllt, wird das Programm mit dem nächsten Strukturblock fortgesetzt.

Struktogramm:



Beispiel: Es soll ein Programm erstellt werden, welches zwei Wörter alphabetisch ordnet. Vor Abarbeitung der Sortierroutine soll der Bildschirm gelöscht werden.

Struktogramm:

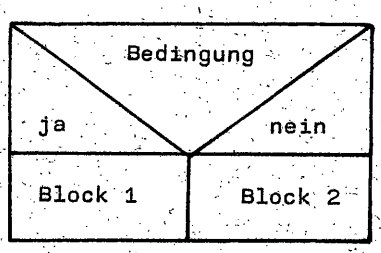


BASIC - Programm:

```
1Ø CLS
2Ø PRINT:INPUT"ZWEI WOERTER EINGEBEN:"; A$, B$
3Ø IF NOT (A$ <=B$) THEN
   X$=A$:A$=B$:B$=X$
4Ø PRINT:PRINT,"1.";A$
5Ø PRINT:PRINT,"2.";B$
6Ø PRINT:PRINT
7Ø END
```

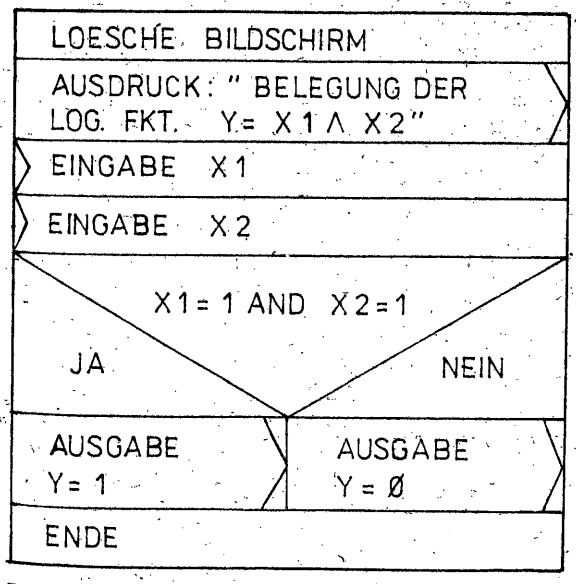
Sind zwei Auswahlmöglichkeiten gegeben (vollständige Alternative), wird das Programm mit dem Strukturblock fortgesetzt, für den die Bedingung erfüllt ist.

Struktogramm:



Beispiel: Es ist ein Programm zur Überprüfung des Wahrheitsgehaltes der logischen Funktion Y = X1 ∧ X2 zu entwickeln.

Struktogramm:

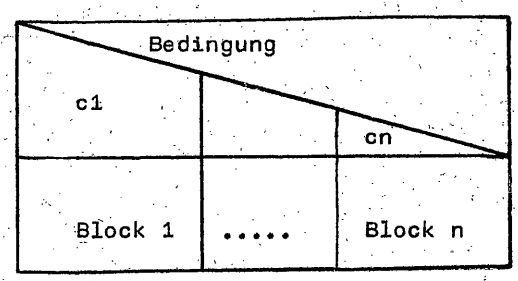


BASIC - Programm:

```
1Ø CLS
2Ø PRINT:PRINT"BELEGUNG DER LOG. FKT. Y=X1 ∧ X2"
3Ø PRINT:PRINT"X1 UND X2 SIND MIT 1 ODER Ø
   ZU BELEGEN!":PRINT:PRINT
4Ø INPUT"X1=";X1
45 IF X1<> 1 AND X1 <>ØTHEN4Ø
5Ø INPUT"X2=";X2
55 IF X2 <> 1 AND X2 <> ØTHEN5Ø
6Ø IF X1=1 AND X2=1 THEN PRINT,"Y=1": ELSE PRINT,
   "Y=Ø"
7Ø END
```

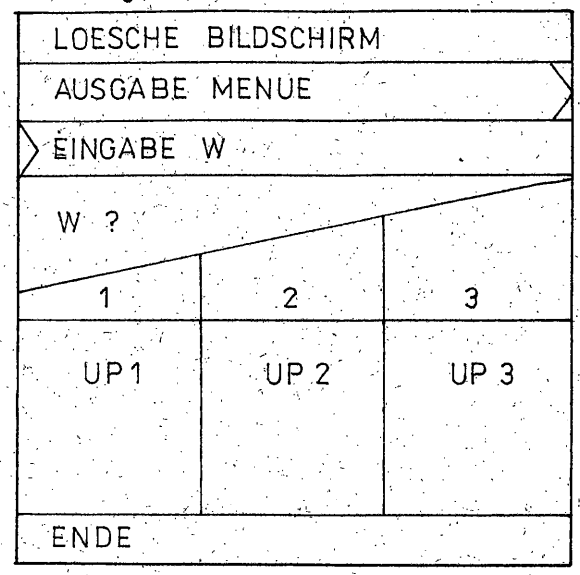
Sind mehr als zwei Auswahlmöglichkeiten gegeben, so wird von einer Fallunterscheidung gesprochen. Es wird als nächstes der Strukturblock bearbeitet, der die vorgegebene Bedingung erfüllt.

Struktogramm:



Beispiel: Es ist ein Programm für ein Menü zu entwickeln, welches die Möglichkeit bietet, drei verschiedene Unterprogramme zur Berechnung von Körpern (Kugel, Quader, Kegel) zu nutzen. Die Berechnungen in den Unterprogrammen sollen an dieser Stelle nicht interessieren.

Struktogramm:



BASIC-Programm:

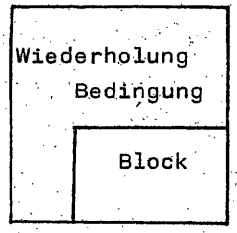
```
1Ø  CLS
2Ø  PRINT:PRINT"VOLUMENBERECHNUNG"
3Ø  PRINT:PRINT"KUGEL  1"
4Ø  PRINT"QUADER  2"
5Ø  PRINT"KEGEL   3"
6Ø  INPUT W
7Ø  ON W GOSUB 1ØØ,4ØØ,7ØØ
8Ø  END
1ØØ  PRINT:PRINT"UNTERPROGRAMM KUGEL"
39Ø  RETURN
4ØØ  PRINT:PRINT"UNTERPROGRAMM QUADER"
69Ø  RETURN
7ØØ  PRINT:PRINT"UNTERPROGRAMM KEGEL"
89Ø  RETURN
```

Wiederholung

Die algorithmische Grundstruktur der Wiederholung ist die Basis für Programmschleifen. Bestimmte einzelne Anweisungen oder ganze Strukturblöcke sollen mehrfach durchlaufen werden. Dabei gibt es prinzipiell zwei Möglichkeiten der Art der Schleifen:

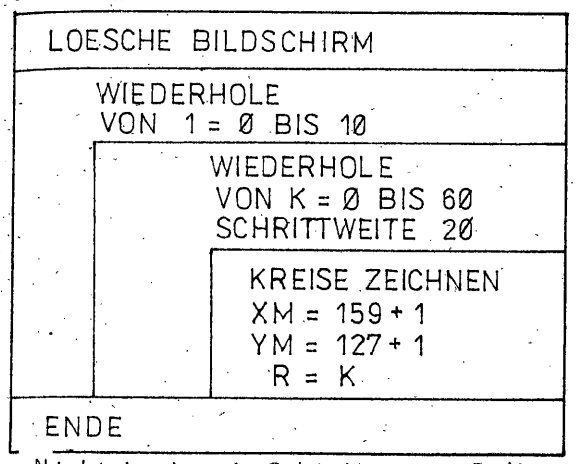
- Abweisende Schleife: Es wird erst eine Bedingung überprüft. Solange diese erfüllt ist, wird der entsprechende Strukturblock wiederholt abgearbeitet. Die Bedingung steht somit am Schleifenanfang.

Struktogramm:



Beispiel: Erarbeitung eines Programms für die grafische Darstellung von Kreisen mit unterschiedlichen Radien.

Struktogramm:

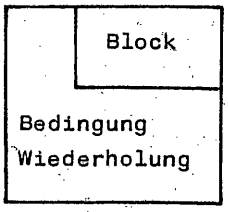


BASIC - Programm:

```
1Ø CLS
2Ø FOR I=Ø TO 1Ø
3Ø FOR K=Ø TO 6Ø STEP 2Ø
4Ø CIRCLE 159+I,127+I,K,7
5Ø NEXT K
6Ø NEXT I
7Ø END
```

- Nichtabweisende Schleife: Die Bedingung steht in diesem Fall am Schleifenende. Der Strukturblock wird solange wiederholt durchlaufen, bis die Bedingung erfüllt ist. Der Strukturblock wird also in jedem Fall erst einmal durchlaufen.

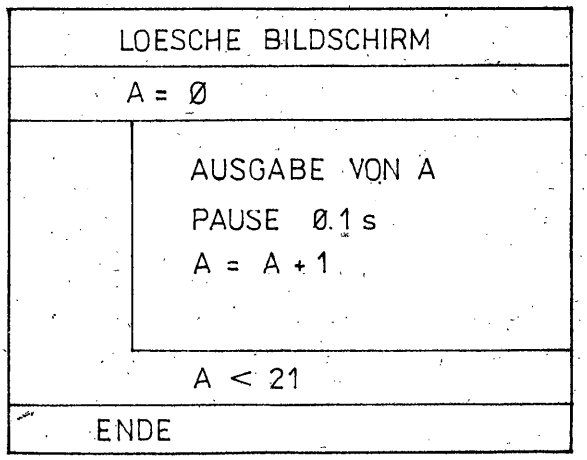
Struktogramm:



Ein Sonderfall der nichtabweisenden Schleife ist die Zählschleife. In diesem Fall wird die Anzahl der Wiederholungen des Anweisungsblockes in einer Bedingung festgelegt. Durch die Laufvariable wird die Anzahl der Wiederholungen gezählt, und die Schleife wird beim Erreichen des Endwertes abgebrochen.

Beispiel: Programmierung einer Zählschleife

Struktogramm:



BASIC - Programm:

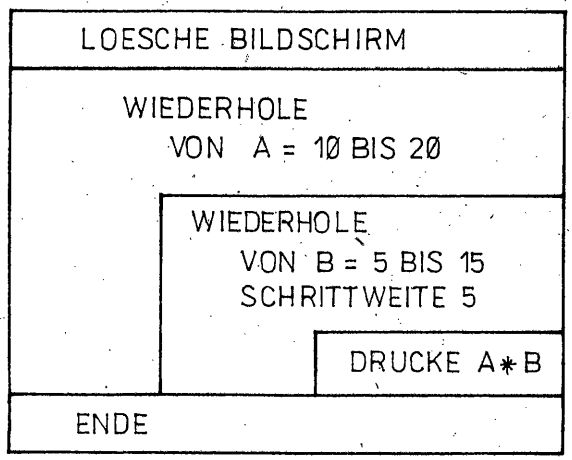
```
1Ø CLS
2Ø A = Ø
3Ø PRINT AT(15,1Ø);A
4Ø PAUSE 1Ø
5Ø A=A+1
6Ø IF A < 21 THEN 3Ø
7Ø END
```

Bei der Arbeit mit Schleifen sind folgende Hinweise zu beachten:

1. Es darf nie von außen in den Anweisungsblock einer Schleifenanweisung hineingesprungen werden. Herauszuspringen ist dagegen jederzeit möglich.

2. Es darf keine Schleifenüberlappung geben. Bei verschachtelten Schleifen muß immer erst die innere und danach die äußere Schleife abgeschlossen werden.

Beispiel:




```
1Ø CLS
2Ø FOR A=1Ø TO 2Ø
3Ø FOR B=5 TO 15 STEP 5
4Ø PRINT A*B,
5Ø NEXT B
6Ø NEXT A
7Ø END
```

4.2.2. Geben Sie das Beispielprogramm ein, ändern Sie die Variablen Zeile 5Ø NEXT A und Zeile 6Ø NEXT B! Wie reagiert der Computer und warum?

In der im KC 85/3 bzw. KC 85/2 vorliegenden BASIC-Version können die Variablen hinter NEXT auch weggelassen werden. Damit erhöht sich die Geschwindigkeit des Programms.

4.2.3. Sehen Sie sich gelistete Programme an und versuchen Sie, die genannten Strukturen zu finden! (Besonders in den Programmen KÖRPER und ÜBUNG.)

4.2.4. * Erarbeiten Sie ein Programm, das auf dem Bildschirm erkennen läßt, wann der Computer in einer inneren oder einer äußeren Schleife arbeitet!

5. Einführung in die Programmiersprache BASIC des KC 85/3 bzw. KC 85/2

Die höhere Programmiersprache BASIC ist eine von vielen Programmiersprachen, die bisher entwickelt wurden. Sie ist geeignet, um sich sehr schnell einige Grundlagen der Programmierung anzueignen.

5.1. Informieren Sie sich im Lb GdA S. 88 über weitere Programmiersprachen und ihre Einsatzgebiete!

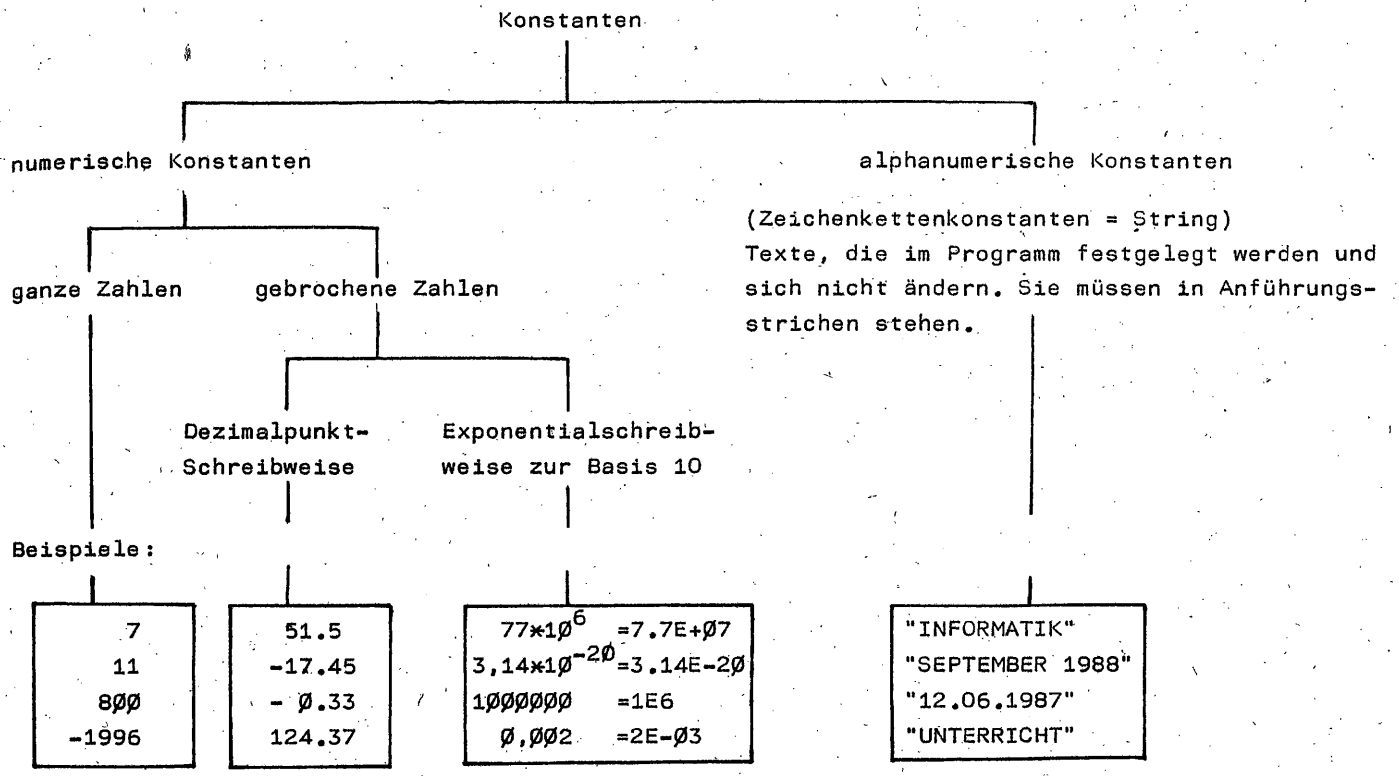
Nachdem bereits im Abschnitt 3 dieser Stoffsammlung einige BASIC - Anweisungen erläutert wurden und auch erste Übungen am Computer dazu erfolgten, werden nun weitere BASIC - Anweisungen eingeführt, mit denen Programme unter Verwendung der im vorangegangenen Abschnitt erläuterten Schrittfolge entwickelt werden können.

Doch zunächst müssen noch einige BASIC - Vereinbarungen betrachtet werden. Während einer Programmabarbeitung werden verschiedene Daten transportiert, berechnet, ein- und ausgegeben. Es ist wichtig zu wissen, wie diese Daten aussehen können.

5.1. BASIC-Vereinbarungen

Konstanten

Konstanten sind feste Größen. Der BASIC-Interpreter kann zwei Arten unterscheiden:



Ganze Zahlen zwischen - 999 999 und + 999 999 werden normal, als Festkommazahl, ausgegeben, gebrochene Zahlen $< Ø,Ø1$ und Zahlen $> 999\ 999$ als Gleitkommazahlen.

Die größte darstellbare Zahl ist $1,7Ø141^{38}$ (Computerschreibweise: 1.7Ø141E+38) und die kleinste ist $9,4Ø396 * 10^{-39}$ (Computerschreibweise: 9.4Ø396E-39). Diese Größen ergeben sich aus der internen dualen Zahlencodierung, die der BASIC-Interpreter verwendet und sind damit fest vorgegeben.

5.1.1. Geben Sie folgende Anweisungen in den Computer ein, und kommentieren Sie den Ausdruck auf dem Bildschirm!

```
PRINT 19+8 , 3∧.5 , 2/25Ø
PRINT 3/9.4Ø396E-39
```

Die Fehlermeldung OV ERROR besagt, daß die kleinste Computerzahl unterschritten wurde.

```
PRINT, SQR(1.7Ø141E+38)
PRINT, "KC 85/3", "Computer"
```

Variablen

Variable sind veränderbare Dateneinheiten. Sie erhalten im Programmablauf einen Wert zugewiesen, den sie so lange behalten, bis ein neuer Wert zugewiesen wird. Eine Variable besteht immer aus dem Variablennamen und dem Variablenwert. Solange einer Variablen kein Wert zugewiesen wird, ist ihr Wert Ø (Null).

Der Variablenname kann frei gewählt werden. Dabei ist es günstig, den Namen möglichst so zu wählen, daß ein Bezug zum Begriff, für den die Variable steht, hergestellt werden kann. Folgende Regeln sind zu beachten:

1. Ein Variablenname muß immer mit einem Buchstaben beginnen. Danach können auch Zahlen folgen, jedoch ohne Leerzeichen.

   Z. B.: Mittelwert MW
   Fläche F oder A
   Kreisfläche FK
   1. Platz P1

2. Variablennamen können beliebig lang sein, doch nur die ersten zwei Zeichen werden vom Computer erfaßt und erkannt. Er kann somit "MALER" und "MAXIMUM" nicht unterscheiden, da beide Variablennamen als erste zwei Zeichen ein "MA" enthalten.

   Z. B.: "MIETE" und "MINIMUM"
   "KRF" (Kreisfläche) und "KRU" (Kreisumfang), aber wenn dafür verwendet wird "KF" und "KU" besteht keine Gefahr!

3. Reservierte Worte, die einer BASIC-Anweisung entsprechen, darf ein Variablennamen nicht enthalten. Es ist deshalb zu empfehlen, nur zwei Zeichen als Variablenname zu verwenden. Die möglichen reservierten Worte reduzieren sich dann auf: TO, IF, AT, ON, OR und PI.

   Z. B.: "EDITH" enthält das reservierte Wort "EDIT"
   "TELEFON" enthält das reservierte Wort "ON"

4. Soll dem Variablennamen eine Zeichenkette zugewiesen werden (Stringvariable), so wird der Variablenname mit einem Dollarzeichen "$" abgeschlossen.

   Z. B.: Name NA$
   Wohnort WO$

5.1.2. *Welche der angegebenen Variablennamen sind unzulässig?
a) VO b) 1A c) LISTE d) DATUM e) ENDWERT f) B5

Standardfunktionen

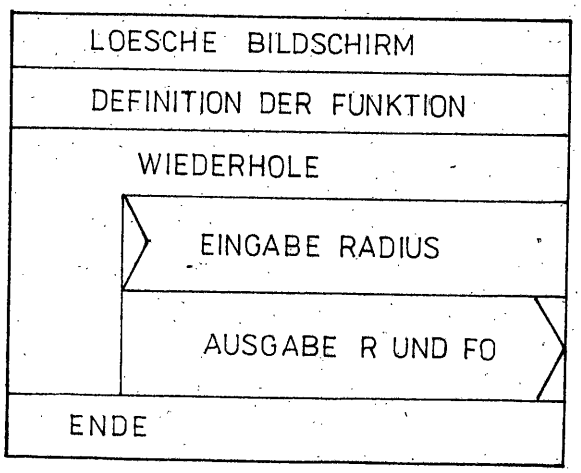
Jeder Rechner, z. B. auch der Taschenrechner, verfügt über ganz bestimmte sofort nutzbare Funktionen. Diese sogenannten Standardfunktionen sind im jeweiligen Rechner durch die Hardware fest eingebaut. Auch der im KC 85/3 bzw. KC 85/2 enthaltene BASIC-Interpreter besitzt eine Reihe von Standardfunktionen, die sofort nutzbar sind, z. B. ABS(X), SIN(X), LOG(X) u. a. (vergleiche dazu Anlage 3).

Darüber hinaus bietet der BASIC-Interpreter mit der Anweisung DEF FN die Möglichkeit, umfangreiche Formeln und im Programm mehrfach verwendete Berechnungen als Funktion zu definieren und unter einem selbst zu wählenden Funktionsnamen wieder aufzurufen.

Format: DEF FN Name (verwendetes Argument) = Ausdruck

Beispiel: Es ist die Oberfläche einer Kugel zu berechnen und für verschiedene Radien eine Wertetabelle aufzustellen!

Struktogramm



BASIC-Programm

```
1Ø CLS
2Ø DEF FNFO(R)=4*PI*R*R
3Ø INPUT "RADIUS:";R
4Ø PRINT:PRINT, "R=";R,"FO=";FNFO(R)
5Ø PRINT
6Ø GOTO 3Ø
7Ø END
```

Die Anwendung der Anweisung DEF FN(X) wirkt sich auf die Effektivität eines Programms positiv aus. Das Programm wird schneller, wenn häufig verwendete komplizierte mathematische Verknüpfungen am Anfang des Programms definiert werden.

5.1.3. Definieren Sie andere Funktionen und stellen Sie dazu Wertetabellen auf!

Ausdrücke

Ausdrücke werden in BASIC nach strengen Regeln aus Konstanten und/oder Variablen, die mit Operatoren verknüpft sind, gebildet. Die Operatoren führen mit den Werten mathematische und/oder logische Operationen durch.

1. Arithmetische Ausdrücke sind z. B. (4+A)*5
   SIN(A)
   2*A+2*B

   Nach Belegung aller Variablen mit Werten wird ein arithmetischer Ausdruck im Computer zu einer Zahl verarbeitet.

2. Logische Ausdrücke sind z. B.: 24 < 1ØØ
   Y = X
   MAX > 1ØØ
   A$+B$ <> "AUTOMOBIL"

   Diese Vergleichsoperatoren liefern nach Belegung aller Variablen mit einem Wert eine Aussage, die "wahr" oder "falsch" sein kann. Das Ergebnis eines solchen Vergleichs hat Einfluß auf den weiteren Verlauf eines Programms.

```
Beispiel: 1Ø CLS
          2Ø INPUT "A=";A
          3Ø IF A < 1Ø THEN PRINT "A < 1Ø": ELSE PRINT "A > 1Ø"
          4Ø END
```

Sind in einem Ausdruck arithmetische und logische Operatoren enthalten, so werden erst die arithmetischen Operationen ausgeführt und danach der logische Vergleich.

Z. B.: X + Y > Z + X

5.1.4. Geben Sie das kleine Beispielprogramm ein und beobachten Sie, was geschieht, wenn Sie unterschiedliche Werte für A eingeben!

5.1.5. Bei der Eingabe der Zahl 1Ø wird in diesem Programm "A > 1Ø" ausgedruckt. Warum ist das so, und wie muß das Programm verändert werden, damit bei der Eingabe der Zahl 1Ø der Ausdruck "A=1Ø" auf dem Bildschirm erscheint?

Operationen mit Zeichenketten (Stringoperationen)

Auch Zeichenketten können addiert (Stringaddition) bzw. logisch miteinander verknüpft werden.

```
Beispiel: 1Ø CLS
          2Ø A$="FEIER" : B$="BROT" : C$="ABEND"
          3Ø PRINT A$ + C$, C$ + B$, C$ + A$
          4Ø PRINT:PRINT:END
```

5.1.6. Geben Sie das Beispielprogramm ein und erläutern Sie die Wirkungsweise!

Neben diesen Stringoperationen gibt es noch eine Reihe von Stringfunktionen, die eine effektive Nutzung von Zeichenketten gestatten.

LEN(String)	Berechnet die Anzahl der Zeichen eines Strings (einschließlich Leerzeichen).
STR$(A)	Wandelt eine Zahl A in eine Zeichenkette um.
VAL(Zahlenstring)	Wandelt einen String, der aus Zahlen besteht, in einen numerischen Wert um.
STRING$(n,String)	Vervielfacht einen String n - mal.
LEFT$(String,A)	Bildet einen neuen String, der aus A Zeichen besteht, beginnend von links.
RIGHT$(String,A)	Bildet einen neuen String, der aus A Zeichen besteht, beginnend von rechts.
MID$(String,A,B)	Bildet einen neuen B Stellen langen String beginnend mit der A-ten Stelle
INSTR(A$,B$)	Errechnet die Position, ab welcher A$ in B$ enthalten ist.
VGET$	Der Inhalt der Cursorposition wird als Zeichenkette ausgegeben.

Beispiele: PRINT, LEN("COMPUTER")
PRINT, VAL("1ØØ")-1
A$="TEILNEHMER":B$="TEIL":PRINT,INSTR(A$,B$),INSTR(B$,A$)

Bildung neuer Zeichenketten:

```
1Ø CLS
2Ø A$ = "MAMAIA"
3Ø PRINT, LEFT$(A$,4)
4Ø PRINT, MID$(A$,3,3)
5Ø PRINT, RIGHT$(A$,2)
6Ø END
```

Vervielfachung einer Zeichenkette:

```
1Ø CLS
2Ø INPUT "ZEICHENKETTE EINGEBEN!";A$
3Ø INPUT "FAKTOR   N:";N
4Ø PRINT STRING$(N,A$)
5Ø PRINT:PRINT
7Ø END
```

5.1.7. Geben Sie die Beispiele in den Computer ein, und beobachten Sie die Wirkung der einzelnen Anweisungen! Versuchen Sie, eigene Beispiele zu finden!

5.2. Wertzuweisung und Eingabe von Daten

Wie bereits erläutert, erhalten Variable im Programmverlauf Werte zugewiesen. Das kann auf zwei prinzipiellen Wegen erfolgen:

1. Wertzuweisung im Programm selbst, ohne Programmunterbrechung. Das heißt, die Werte, die den Variablen zugewiesen werden sollen, wurden bereits im Programm festgelegt. Dazu werden die Anweisungen LET und READ DATA verwendet.

2. Wertzuweisung erfolgt von außen - z. B. über die Tastatur oder den Kassettenrecorder. Dabei finden die Anweisungen INPUT und INKEY$ Anwendung.

LET - Anweisung

Bei der Zuweisung eines Wertes an eine Variable im Programmverlauf durch Anwendung der Anweisung LET ist darauf zu achten, daß links vom Gleichheitszeichen der Variablenname und rechts der Wert, der der Variablen zugewiesen werden soll, steht. Das Gleichheitszeichen hat hier eine andere Bedeutung als in der Mathematik. Die Seiten sind deshalb nicht vertauschbar.

Format: LET Variable = Ausdruck

```
Beispiel: LET A = 124
          LET MAX = 2ØØ
          LET NA$ = "OSKAR"
          LET B = 4*A+C
```

Bei den meisten Computern kann das Schlüsselwort LET auch weggelassen werden. So auch beim KC 85/3 und KC 85/2. Deshalb wird von nun an die Schreibweise ohne Schlüsselwort verwendet.

```
Beispiel: 2Ø Y = SIN (X)
          5Ø F = A * B
```

5.2.1 Welche Wertzuweisungen sind richtig?

a) LET 3*A = B
b) V = 25Ø
c) Y = 1/X
d) O$ = "BERGEN"
e) MA = MATHEMATIK

READ DATA - Anweisung

Die zweite Möglichkeit, ohne Programmunterbrechung den Variablen Werte zuzuweisen, ist die Anwendung der READ DATA - Anweisung. Diese Anweisung greift auf Daten zurück, die in bestimmten Programmzeilen abgelegt sind. Sie werden durch Komma getrennt und durch das Schlüsselwort DATA am Anfang der Programmzeile gekennzeichnet.

Format: DATA Konstante (,Konstante...)

```
Beispiel: DATA Georg, Ralf, Katja
          DATA 15,12Ø,15Ø
          DATA Friedrich Gerber, Lange Straße, 3, Ebersbach, 246Ø
```

Mit der READ - Anweisung werden den Variablen die Werte aus der DATA - Zeile zugeordnet.

Format: READ Variable (,Variable...)

```
Beispiel: READ X
          READ A,B,C
          READ NA$, WO$
```

Die in der DATA - Zeile abgelegten Werte können sowohl numerische Werte als auch Zeichenketten sein. Es ist jedoch wichtig, daß einer numerischen Variablen kein Text zugeordnet wird und umgekehrt.

Beispiel:
```
1Ø CLS
2Ø READ NA$,WO$
3Ø PRINT NA$
4Ø PRINT WO$
5Ø PRINT
6Ø I=I+1: IF I=3 THEN 1ØØ
7Ø GOTO 2Ø
8Ø DATA Herbert Sachs, Berlin, Eva Berger, Potsdam
9Ø DATA Bernd Friedrich, Halle
1ØØ END
```

5.2.2. Geben Sie das Beispielprogramm ein, und starten Sie es!

5.2.3. Verändern Sie das Programm, indem Sie weitere Daten in die DATA - Zeilen eingeben! (Beachten Sie dabei den Zähler I!)

5.2.4. Verändern Sie das Programm so, daß auch die Straße und die Postleitzahl abgerufen werden können!

5.2.5. Sehen Sie sich die Anwendung der DATA - READ - Anweisung in den Programmen TELEFON und TAUSCH an!

Am dargestellten Beispiel ist zu erkennen, daß mit jeder READ-Anweisung der Variablen der nächste Wert aus der DATA-Zeile zugewiesen wird. Die Ursache dafür ist ein "Zeiger", der so steht, daß bei einer erneuten Zuweisung über READ der nächste Wert zugeordnet wird.

Die Wirkung des Zeigers wird auch an folgendem Beispiel deutlich, wenn der Programmverlauf schrittweise verfolgt wird:

Beispiel:

```
 5 CLS
1Ø READ X          DATA-Zeile wird aufgerufen, um der Variablen X den ersten Wert zuzuweisen.
                   Der Zeiger steht auf dem ersten Wert:
                   DATA 1Ø,15,2Ø,25,3Ø
                        ▲
                   Nach Ausführen der READ-Anweisung rückt der Zeiger zum nächsten Wert:
                   DATA 1Ø,15,2Ø,25,3Ø
                           ▲
2Ø READ A,B,C      Die DATA-Zeile wird wieder aufgesucht, um den drei Variablen die drei
                   folgenden Werte zuzuweisen. Danach steht der Zeiger auf dem letzten Wert:
                   DATA 1Ø,15,2Ø,25,3Ø
                                 ▲
3Ø READ Y          Der letzte Wert wird angefordert.
                   DATA 1Ø,15,2Ø,25,3Ø
                                    ▲
4Ø PRINT X,A,B,C,Y Die Werte werden auf dem Bildschirm angezeigt.

5Ø DATA 1Ø,15,2Ø,25,3Ø

6Ø END
```

Würde nach Zeile 3Ø nochmals eine READ-Anweisung folgen, erschiene eine Fehlermeldung, da kein Wert in der DATA-Zeile mehr vorhanden ist.

Soll jedoch erneut auf die Werte zurückgegriffen werden, muß der Zeiger mit der Anweisung RESTORE wieder an den Anfang der Datenliste gesetzt werden.

Format: RESTORE Zeilennummer

Mit dieser Anweisung können die mit den DATA-Anweisungen vereinbarten Werte ab der ersten oder der angegebenen Zeilennummer erneut gelesen werden. Dabei greift die nächste READ-Anweisung auf das erste Datenelement der ersten bzw. der DATA-Anweisung mit der angegebenen Zeilennummer zurück.

5.2.6. Geben Sie das Beispielprogramm ein, und verändern Sie es nach dem ersten Durchlauf durch die Eingabe der Zeile

```
25 RESTORE
```

und beobachten Sie, was passiert!

5.2.7. Geben Sie ein:
35 READ E,F,G,H,I und 45 PRINT E;F;G;H;I
Welcher Fehler liegt vor? Wie ist dieser Fehler zu beseitigen?

INPUT - Anweisung

Wird der Wert der Variablen nicht bereits im Programm fest zugewiesen, sondern soll das durch eine Eingabe über die Tastatur erfolgen, ist die Anweisung INPUT zu verwenden. Dabei erwartet der Computer an dieser Stelle des Programms eine Eingabe eines Wertes oder eines Strings, der der Variablen zugewiesen werden soll.

Format: INPUT Variable (,Variable.....)

Beispiel:
```
INPUT A
INPUT A,B,C
INPUT Q$
```

Bei dieser Form der Eingabe erscheint auf dem Bildschirm ein Fragezeichen.

Soll bei der Eingabeaufforderung zugleich ein Text ausgedruckt werden, ist die Anweisung wie folgt zu verwenden:

Format: INPUT "String"; Variable (,Variable....)

Beispiel:
```
INPUT "Name=";NA$
INPUT "Radius=";R
```

Bei dieser Form der Eingabe erscheint auf dem Bildschirm: Name=
Durch die Form der Eingabe INPUT" ";X kann somit das Erscheinen des Fragezeichens unterdrückt werden.

5.2.8. Geben Sie das folgende Programm ein, und testen Sie es mit unterschiedlichen Werten!

```
10 CLS
20 INPUT A,B,C
30 MW = (A+B+C)/3
40 PRINT MW
50 END
```

5.2.9. Verändern Sie das Programm so, daß bei der Eingabeaufforderung nicht nur das Fragezeichen erscheint, sondern die Bezeichnungen der einzugebenden Werte auf dem Bildschirm angezeigt werden!

INKEY - Anweisung

Mit der Anweisung INKEY$ erfolgt eine Abfrage der Tastatur. Der Computer reagiert auf einen bestimmten Tastendruck und arbeitet dann das Programm dementsprechend ab.

Beispiel:
```
10 CLS
20 A$ = INKEY$:IF A$=""THEN 20
30 IF A$ = "1" THEN PRINT" Taste 1 wurde betaetigt"
40 IF A$ = "2" THEN PRINT" Taste 2 wurde betaetigt"
50 IF A$<>"1" AND A$ <>"2" THEN PRINT "Es wurde eine andere Taste betaetigt"
60 GOTO 20
70 END
```

Diese Anweisung wird vorwiegend zur Steuerung des Programmablaufs eingesetzt.

CLEAR - Anweisung

Die Anweisung CLEAR hat die Funktion, die den Variablen im Programmverlauf zugewiesenen Werte wieder zu löschen und sie mit Ø zu belegen. Diese Funktion ist immanent auch in der Anweisung zum Programmstart RUN enthalten. Wird ein Programm mit RUN gestartet, so werden alle Variablen vom Interpreter mit Ø belegt. Damit ist es in dieser BASIC-Version nicht erforderlich, die Variablen am Programmanfang mit Ø zu belegen.

Sollen nun im Programmverlauf die Variablen wieder gelöscht werden, wird die Anweisung CLEAR verwendet.

Format: CLEAR

Beispiel: Programm zur Demonstration der Anweisung CLEAR
```
10 CLS
20 PRINT:PRINT"EINGABE DER WERTE FUER X UND Y":PRINT
30 INPUT "X=";X
40 INPUT "Y=";Y
50 S=X+Y
60 PRINT:PRINT "Summe=";S
70 CLEAR
80 D=X-Y
90 PRINT:PRINT "Differenz=";D
100 PRINT:PRINT:PRINT:END
```

5.2.10. Geben Sie das Programm ein, und beobachten Sie, was passiert!

5.2.11. Verändern Sie das Programm so, daß auch die Differenz von X und Y berechnet und ausgegeben wird!

Eine zweite Funktion hat die Anweisung CLEAR, wenn sie mit einem Argument verbunden ist (z. B. CLEAR 1ØØØ). Dann bewirkt sie zusätzlich eine Veränderung des internen Speicherbereiches für Strings.

5.2.12. Sehen Sie sich die Programme SORT und UNIDATEI gelistet an! Dort finden Sie diese zweite Funktion der Anweisung CLEAR.

REM - Anweisung

Diese Anweisung ist eine sogenannte Kommentaranweisung. Sie bewirkt, daß alles, was hinter der Anweisung REM in einer Programmzeile steht, vom Rechner ignoriert wird. Es ist damit möglich, in ein Programm Erläuterungen einzubauen, die die Lesbarkeit und Verständlichkeit des Programms erhöhen.

Format: REM Kommentar

Der Kommentar besteht aus einem Text mit beliebigen Zeichen. Auch darin enthaltene BASIC-Anweisungen werden nicht ausgeführt.

Zu REM gibt es als äquivalentes Zeichen das Ausrufezeichen(!). Es hat die gleiche Wirkung wie die Anweisung REM.

5.2.13. Sehen Sie sich dazu gelistete Programme an, und üben Sie an eigenen kleinen Programmen die Anwendung dieser Anweisung!

5.3. Anweisungen zur Ausgabe von Daten

PRINT - Anweisung

Die PRINT-Anweisung wurde bereits im Abschnitt 3.2. bei der Arbeit im Direktmodus verwendet. Sie bewirkt, daß Daten oder Strings auf dem Bildschirm erscheinen. Doch die PRINT-Anweisung kann zugleich helfen, die Bildschirmausschrift gut zu gestalten.

Im folgenden Beispiel sind verschiedene Funktionen der PRINT-Anweisung erläutert:

```
 1Ø CLS
 2Ø READ A,B,C
 3Ø PRINT
 4Ø PRINT A,B,C
 5Ø PRINT:PRINT
 6Ø PRINT A: PRINT B: PRINT C
 7Ø PRINT:PRINT
 8Ø PRINT A;B;C
 9Ø PRINT:PRINT
1ØØ DATA 1Ø, 1ØØ, 1ØØØ
11Ø END
```

PRINT - Ausgabe einer Leerzeile
(Zeilen 3Ø, 5Ø, 7Ø, 9Ø)

PRINT Ausdruck - Der Ausdruck wird in eine Zeile geschrieben. Ist der Ausdruck länger als die Zeile, so wird er in der folgenden fortgesetzt, und der Cursor geht an den Anfang der darauf folgenden Zeile. Ist der Ausdruck gerade so lang wie die Zeile, so überspringt der Cursor die nächste Zeile. Der Ausdruck kann auch ein String sein (z. B. 25 PRINT "PRINT-Beispiele")

PRINT Ausdruck, - Komma nach dem Ausdruck (Zeile 4Ø)
Der Ausdruck beginnt am Anfang einer Zeile. Das Komma bewirkt, daß der folgende Druck 13 Spalten vom Anfang des vorhergehenden beginnt. 3 Zahlen können so in 40 Spalten plaziert werden. Für jede der 3 Zahlen stehen damit 13 Zeichen zur Verfügung, das entspricht dem Platz für die längste Gleitkommazahl (12 Zeichen) plus einem Leerzeichen. Sollte der Ausdruck länger als 12 Zeichen sein, so verliert das Komma seine Wirkung.

PRINT Ausdruck; - Semikolon nach dem Ausdruck (Zeile 8Ø)
Das Semikolon bewirkt, daß der nachfolgende Ausdruck unmittelbar anschließt, wenn er ein String ist; nach einer Zahl wird noch ein Leerzeichen gesetzt.

5.3.1. Geben Sie das Beispielprogramm ein, und erläutern Sie die Wirkung der PRINT-Anweisung!

5.3.2. Entwickeln Sie ein ähnliches Programm zur Demonstration der PRINT-Anweisungen unter Verwendung von Strings!

Für eine gute Gestaltung der Ausgabe der Daten und Texte bietet die Programmiersprache BASIC aber noch andere PRINT-Anweisungen. Um diese zu nutzen, ist es erforderlich, die Einteilung des Bildschirmes genau zu kennen.

Auf dem Bildschirm können 32 Zeilen mit je 4Ø Zeichen beschrieben werden. Wie im Bild 9 dargestellt ist, beginnt die Zählung der Zeilen (Z) und der Spalten (S) im ersten Feld am linken oberen Rand mit Null und es gilt:

Ø <= Z <= 31
Ø <= S <= 39

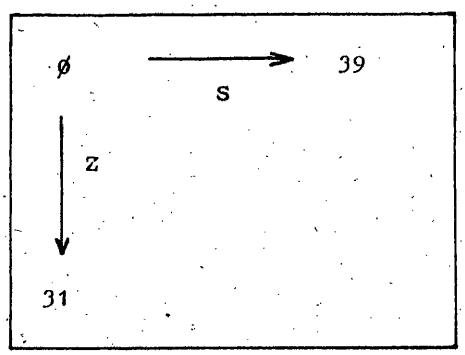


Bild 9: Zeichenkoordinatensystem

Dieses Koordinatensystem mit 32 Zeilen und 4Ø Spalten wird "Zeichenkoordinatensystem" genannt. (Siehe auch Anlage 5 und Abschnitt 6.1.). Der Cursor kann in diesem Koordinatensystem durch die Angabe der Zeilen- und Spaltennummer genau positioniert werden und damit die Ausgabe auf dem Bildschirm beeinflußt werden.

PRINT AT - Anweisung

Diese Anweisung wird zur Positionierung eines Ausdrucks in eine bestimmte Spalte und Zeile verwendet.

Format: PRINT AT (Zeile, Spalte); Ausdruck

Beispiel: In einem Programm soll während umfangreicher interner Arbeiten im Programm in der Mitte des Bildschirmes der Hinweis "Moment bitte!" erscheinen.

```
1ØØ PRINT AT(15,1Ø); "Moment bitte!"
```

5.3.3. Bauen Sie diese Programmzeile in ein Programm ein!

5.3.4. Ergänzen Sie ein von Ihnen ausgewähltes Programm so, daß am Ende des Programms in der Mitte des Bildschirmes der Hinweis "Programmende" erscheint!

PRINT TAB- und PRINT SPC-Anweisung

Mit den Anweisungen PRINT TAB und PRINT SPC ist die Möglichkeit der Darstellung in Tabellenform gegeben. Dabei wird der entsprechende Ausdruck genau in einer vorgeschriebenen Spalte plaziert.

Format: PRINT TAB(I); Ausdruck

I gibt den Abstand der Ausgabe zum linken Bildschirmrand in Zeichen an.

Beispiel: PRINT TAB(12);"Spalte 1"

Der Ausdruck des Wortes "Spalte 1" beginnt im 13. Zeichen.

Format: PRINT SPC(I); Ausdruck

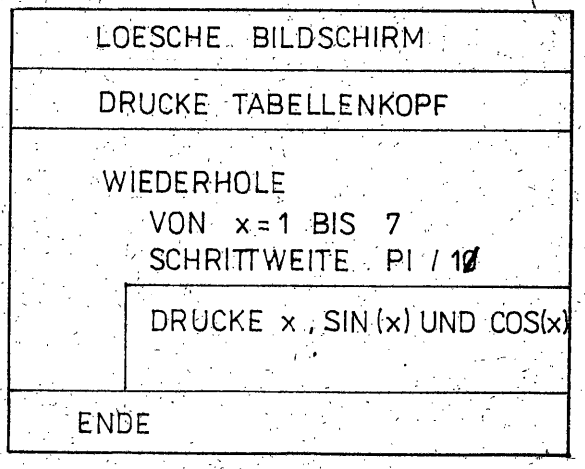
I gibt die Anzahl der zu erzeugenden Leerzeichen zwischen der aktuellen Cursorposition und der neuen Ausgabe an. Damit bezieht sich PRINT TAB immer in der aktuellen Zeile auf den linken Bildschirmrand, und PRINT SPC bezieht sich immer auf die aktuelle Cursorposition.

Beispiel: PRINT SPC(7);"Spalte 1"; SPC(3); "Spalte 2"

Der Ausdruck "Spalte 1" beginnt im 8. Zeichen und der Ausdruck "Spalte 2" beginnt im 19. Zeichen.

Beispiel: Die Sinus- und Cosinuswerte von X sind in einer Tabelle darzustellen.

Struktogramm:



BASIC-Programm:

```
10 CLS
20 PRINT:PRINT"X";TAB(13);"SIN X";TAB(26);"COS X":
   PRINT
30 FOR X=1 TO 7 STEP PI/10
40 PRINT X;TAB(13);SIN(X);TAB(26);COS(X)
50 NEXT X
60 END
```

5.3.5.*Es ist ein Programm zu erarbeiten für die Berechnung der Fläche, des Umfangs und des Volumens einer Kugel. Die Ergebnisse sind in einer Tabelle darzustellen!

5.3.6.*Es ist ein Programm zu erarbeiten für das Raster eines Stundenplanes!

5.4. Anweisungen für Programmschleifen und Programmverzweigungen

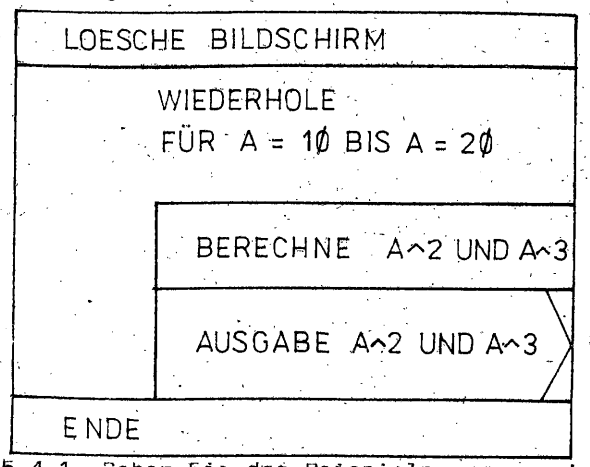
FOR .... TO .... NEXT - Anweisung

Soll eine Anweisung oder eine Folge von Anweisungen mehrmals durchlaufen und ausgeführt werden, wird für diesen Zyklus die Anweisung FOR ... TO ... NEXT verwendet. Die Schlüsselworte FOR und NEXT schließen den Anweisungsblock ein, der wiederholt ausgeführt werden soll. Mit der Schrittweite Z kann festgelegt werden, in welchen Schritten die Laufvariable A vom Anfangswert X den Endwert Y erreicht. Für die Schrittweite 1 kann die Angabe STEP entfallen. Wird der Endwert Y erreicht, wird die Schleife automatisch abgebrochen.

Format: FOR A = X TO Y STEP Z ..... NEXT

Beispiel: Es ist ein Programm zu erarbeiten, mit dem alle Quadrat- und Kubikzahlen für die Werte von 10 bis 20 ausgedruckt werden.

Struktogramm:



BASIC - Programm:

```
1Ø CLS
2Ø FOR A=1Ø TO 2Ø
3Ø Q=A ^ 2
4Ø K=A ^ 3
5Ø PRINT A,Q,K
6Ø NEXT A
7Ø END
```

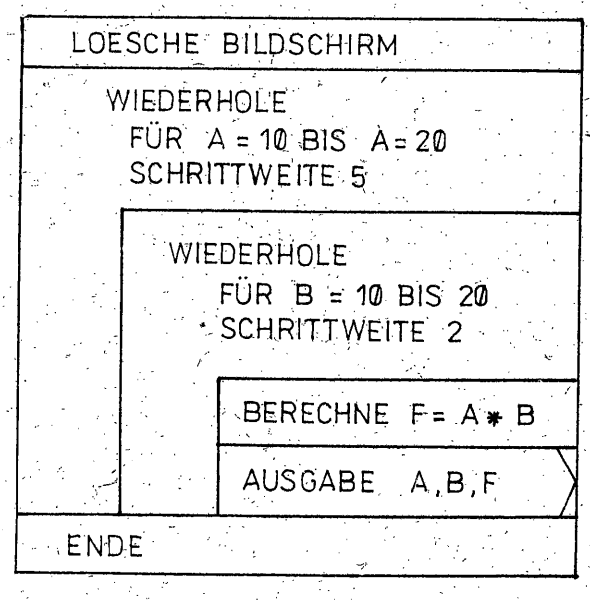
5.4.1. Geben Sie das Beispielprogramm ein, und beobachten Sie den Programmablauf!

5.4.2. Verändern Sie das Programm so, daß nur jeder zweite bzw. jeder dritte Wert berechnet wird!

In einem Programm können auch mehrere Schleifen verschachtelt werden. Das findet z. B. Anwendung beim Aufbau einer Matrix oder der Ausgabe von Tabellen. Wichtig ist, daß bei Verwendung mehrerer Schleifen immer erst die innere und dann die äußere Schleife beendet wird.

Beispiel: Es sollen die Flächeninhalte berechnet werden für die Werte A=1Ø, A=15, A=2Ø und B von 1Ø bis 2Ø in zweier Schritten.

Struktogramm:



BASIC - Programm

```
1Ø CLS
2Ø FOR A=1Ø TO 2Ø STEP 5
3Ø FOR B=1Ø TO 2Ø STEP 2
4Ø F=A*B
5Ø PRINT A,B,F
6Ø NEXT B
7Ø NEXT A
8Ø END
```

5.4.3. Verändern Sie im Beispielprogramm beliebig die Schrittweiten von A und B!

5.4.4. Was passiert, wenn Sie in Zeile 6Ø NEXT A und Zeile 7Ø NEXT B eingeben?

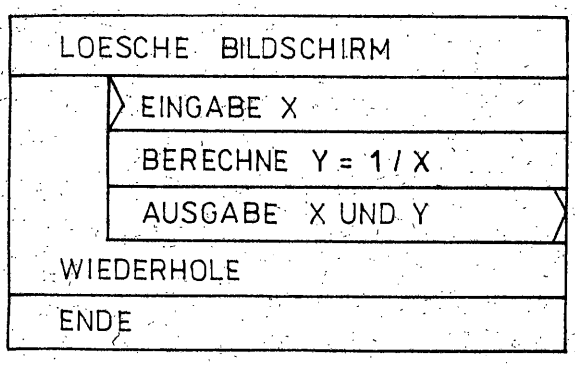
5.4.5.* Erarbeiten Sie ein Programm, mit dem ein Kästchen von 2Ø Zeichen und 5 Zeilen mit * ausgefüllt wird!

GOTO - Anweisung

Von der zeilenweisen Abarbeitung eines BASIC-Programms kann mit Hilfe der GOTO-Anweisung abgewichen werden. Diese Anweisung entspricht einem unbedingten Sprung, d. h. es muß in jedem Fall, unabhängig von einer bestimmten Bedingung, der lineare Verlauf des Programms unterbrochen werden. Sie unterstützt in keiner Weise das strukturierte Programmieren. Mit der GOTO-Anweisung ist deshalb sehr sparsam umzugehen, um die Übersichtlichkeit des Programms zu erhalten.

Format: GOTO Zeilennummer

Beispiel: Verwendung der GOTO-Anweisung, um ein Programm beliebig oft abarbeiten zu lassen.




```
1Ø CLS
2Ø INPUT"X=";X
3Ø PRINT
4Ø Y=1/X
5Ø PRINT,"X=";X,"Y=";Y
6Ø GOTO 2Ø
7Ø END
```

In diesem Beispiel wird durch die Anweisung GOTO in Zeile 6Ø eine Schleife ohne Ende erzeugt. Das Programm kann nur mit BRK abgebrochen werden.

IF ... THEN ... Anweisung

Sehr häufig tritt die GOTO-Anweisung in BASIC-Programmen in Verbindung mit einer Bedingung auf. Es handelt sich dabei um einen sogenannten bedingten Sprung. Nur wenn die Bedingung erfüllt ist, wird der Sprung ausgeführt, ansonsten wird die nächste Anweisung bearbeitet. Damit kann eine Programmverzweigung, aber auch eine Programmschleife mit Abbruchbedingung programmiert werden. Die Bedingung wird durch das Schlüsselwort IF eingeleitet und kann mit den Schlüsselwörtern GOTO und THEN gekoppelt werden.

Beispiel: Es ist ein Programm zu erarbeiten, durch welches vom Computer nur Werte größer als 1Ø ausgedruckt werden. Wird eine kleinere Zahl als 1Ø eingegeben, soll die Eingabe wiederholt werden.

```
1Ø CLS
2Ø INPUT "A=";A
3Ø IF A > 1Ø THEN PRINT A
4Ø IF A < 1Ø GOTO 2Ø
5Ø END
```

5.4.6. Was geschieht, wenn Sie bei vorliegendem Beispiel die Zahl 1Ø eingeben? Begründen Sie Ihre Antwort!

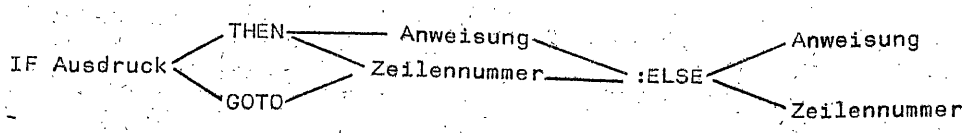
Das Beispielprogramm läßt sich vereinfachen durch die Verbindung der Anweisung IF ... THEN mit dem Schlüsselwort ELSE. Es kann damit angegeben werden, welche Anweisung auszuführen ist, wenn die vorgegebene Bedingung nicht erfüllt ist. Die Anweisung IF - THEN - ELSE stellt somit eine alternative Sprunganweisung dar.

Beispiel:
```
1Ø CLS
2Ø INPUT "A=";A
3Ø IF A > 1Ø THEN PRINT A : ELSE 2Ø
4Ø END
```

5.4.7. Was geschieht, wenn Sie im Beispiel die Zahl 1Ø eingeben? Welchen Unterschied gibt es zum vorherigen Beispiel?

Für die beschriebenen Anweisungen gilt folgendes:

Format:



Beispiele: IF X > Ø THEN = 1/X : ELSE Y = Ø
IF Q$ ="ENDE" GOTO 1ØØ
IF INKEY$ = X THEN 3ØØ

5.4.8. Sehen Sie sich gelistete Programme an, und suchen Sie weitere Beispiele für die Verwendung dieser Anweisungen!

ON ... GOTO - Anweisung

Sind mehr als zwei Möglichkeiten für die Fortsetzung eines Programms vorgesehen, so liegt eine Fallentscheidung vor. Sie wird mit der Anweisung ON ... GOTO realisiert.

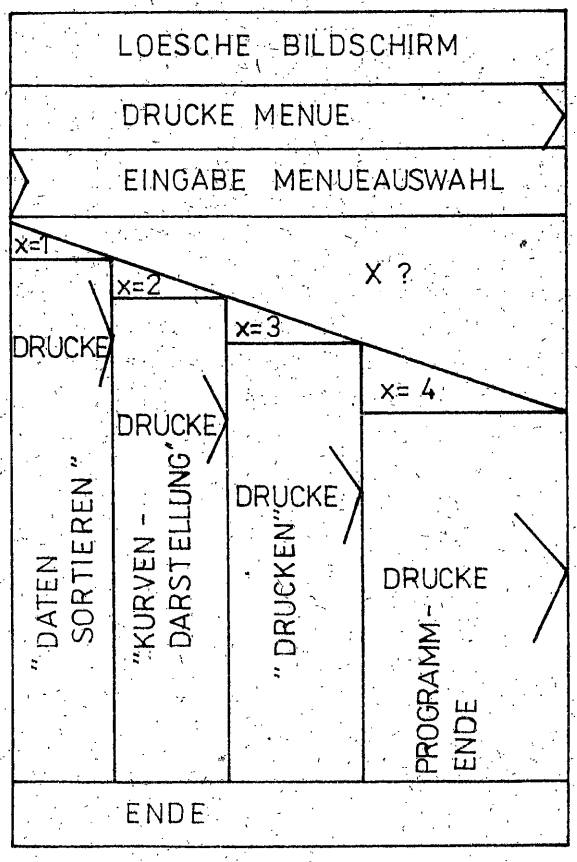
Format: ON X GOTO Liste von Zeilennummern

Der Wert von X bestimmt die anzuspringende Zeilennummer. Ist z. B. X = 3, so wird die an dritter Stelle der Liste angegebene Zeilennummer angesprungen und das Programm dort fortgesetzt. Dabei kann X im Programm vorgegeben, über Tastatur eingegeben oder berechnet werden.

Beispiel: Erarbeitung eines Programms für ein Menü mit folgenden Auswahlmöglichkeiten:

1. Daten sortieren,
2. Kurvendarstellung,
3. Drucken,
4. Programmende.

Struktogramm:



BASIC - Programm:

```
1Ø CLS
2Ø PRINT AT(3,18);"MENUE"
3Ø PRINT AT(4,18);"====="
4Ø PRINT:PRINT:PRINT:PRINT
5Ø PRINT"1 = Daten sortieren"
6Ø PRINT"2 = Kurvendarstellung"
7Ø PRINT"3 = Drucken"
8Ø PRINT"4 = Programmende"
9Ø INPUT" MENUENUMMER:";X
1ØØ IF X >4 OR X < 1 THEN PRINT "Nicht im Menue":
    GOTO 9Ø
11Ø ON X GOTO 2ØØ,3ØØ,4ØØ,5ØØ
2ØØ CLS:PRINT:PRINT"Daten sortieren"
29Ø PAUSE 2Ø:GOTO 1Ø
3ØØ CLS:PRINT:PRINT"Kurvendarstellung"
39Ø PAUSE 2Ø:GOTO 1Ø
4ØØ CLS:PRINT:PRINT"Drucken"
49Ø PAUSE 2Ø:GOTO 1Ø
5ØØ CLS:PRINT:PRINT"Programmende"
59Ø PAUSE 2Ø:END
```

In diesem Programmbeispiel wird der Wert für X über die Tastatur eingegeben. Der Nutzer des Programms entscheidet mit der Eingabe über INPUT (Zeile 9Ø), welche Zeilennummer (Menüpunkt) angesprungen werden soll.

X muß immer im Bereich zwischen 1 und 255 liegen. Ist X = Ø oder größer als die Anzahl der Listenelemente, so wird die Anweisung ignoriert und in der nächsten Programmzeile das Programm fortgesetzt. Damit bei falschen Eingaben oder bei für X errechneten Werten außerhalb der Anzahl der Listenelemente das Programm nicht unkontrolliert weiterläuft bzw. "abstürzt" (z. B. Fehlermeldung bei X<Ø), sind besondere Maßnahmen zu treffen.

Eine Maßnahme ist als Beispiel in Zeile 1ØØ getroffen worden, indem der zulässige Bereich für X festgelegt wurde und Bereichsüberschreitungen zu einer neuen Eingabe auffordern. Dieses Programm kehrt von selbst ins Menü zurück.

5.5. Arbeit mit Unterprogrammen

GOSUB - Anweisung

Wird ein bestimmter Programmabschnitt in einem Programm mehrfach benötigt, so ist es effektiver, diesen Abschnitt nur einmal zu programmieren und ihn als Unterprogramm zu nutzen. Durch entsprechende Anweisungen im Hauptprogramm kann dieser Programmabschnitt dann jederzeit angesprungen und abgearbeitet werden. Eine Möglichkeit für einen Sprung in einen anderen Programmteil wurde bereits mit der Anweisung GOTO vorgestellt. Diese Anweisung hat aber den Nachteil, daß für den Rücksprung in das Hauptprogramm eine Adresse angegeben werden muß. Werden nun im Hauptprogramm Veränderungen vorgenommen, die sich auf die Zeilennummer auswirken, so müßten auch die mit GOTO anzuspringenden Programmteile und die entsprechenden Rücksprungadressen korrigiert werden. Bei sehr umfangreichen Programmen ist das sehr aufwendig, und die Möglichkeit von Fehlern ist sehr groß. Es gibt daher eine günstigere Lösung mit der Anweisung GOSUB.

```
Format:          GOSUB Zeilennummer
                   .
                   .
                   .
        Zeilennummer ..... RETURN
```

Mit der Anweisung GOSUB N wird zu dem auf der Programmzeile N beginnenden Unterprogramm gesprungen und dieses abgearbeitet. Nach der Abarbeitung des Unterprogramms, das immer mit der Anweisung RETURN abschließen muß, kehrt der Computer in das Hauptprogramm zur Aufrufstelle des Unterprogramms zurück und führt die nachfolgenden Anweisungen aus. Es ist also keine Rücksprungadresse erforderlich, was ein großer Vorteil bei der Erarbeitung der Programme und der Nutzung von Unterprogrammen ist.

Im Zusammenhang mit der Arbeit mit Unterprogrammen wird noch einmal deutlich, wie wichtig es ist, jedes Programm mit der Anweisung END abzuschließen. Wird das Hauptprogramm nicht mit der Anweisung END abgeschlossen, kann es passieren, daß der Computer in einem sich daran anschließenden Unterprogramm weiterarbeitet und es dann zur Fehlermeldung kommt.

Beispiel:
```
1Ø CLS
2Ø PRINT "DAS IST DAS HAUPTPROGRAMM"
3Ø GOSUB 1ØØ
4Ø PRINT "WIEDER IM HAUPTPROGRAMM"
5Ø GOSUB 2ØØ
6Ø PRINT "NOCH EINMAL DAS HAUPTPROGRAMM"
7Ø PRINT "SCHLUSS"
8Ø END
1ØØ PRINT "HIER IST UNTERPROGRAMM 1"
11Ø RETURN
2ØØ PRINT "HIER IST UNTERPROGRAMM 2"
21Ø RETURN
```

5.5.1. Geben Sie das Beispielprogramm ein, und beobachten Sie die Abarbeitung des Programms!

5.5.2. Was geschieht, wenn Sie die Zeile 8Ø löschen?
Warum reagiert der Computer so?

Die Arbeit mit Unterprogrammen wird vor allem im Zusammenhang mit der Menütechnik genutzt. Auch die Anweisung GOSUB kann mit der Anweisung ON verknüpft werden.

Format: ON X GOSUB Liste von Zeilennummern

5.5.3. Verändern Sie das vorletzte Beispielprogramm so, daß an Stelle der Anweisung ON ... GOTO mit der Anweisung ON ... GOSUB gearbeitet wird!

5.5.4. Sehen Sie sich die Programme "WZM", "KOHLE" und "LAGER" an hinsichtlich der Verwendung der Anweisungen ON ... GOSUB!

5.6. Arbeit mit Feldern

Werden in einem Programm größere Mengen von Variablen benötigt, dann reicht die bisher verwendete Form X, A3, Q$, F u. ä. nicht mehr aus. Der Computer bietet dafür die Möglichkeit, wie es in der Mathematik üblich ist, indizierte Variablen (X1, X2, usw.) zu verwenden.

Eine Gruppe von Variablen, die sich nur durch ihre Indizes unterscheiden, wird ein Variablenfeld genannt (Y_1, Y_2, Y_3 oder A_1, A_2, A_3). Bei der Computerschreibweise wird der Index nicht nach unten versetzt angegeben, sondern wird in Klammern hinter die Variable geschrieben (z. B. X(Ø), X(1), X(2) usw.).

Dieses Variablenfeld X(I) ist eindimensional. Es enthält nur einen Index und entspricht einer Variablenliste. Es können aber auch 2- oder n- dimensionale Variablenfelder vereinbart werden. Diese besitzen dann zwei bzw. n Indizes. Im Unterschied zur Mathematik beginnen die Indizes mit Ø. Die Festlegung bzw. Dimensionierung der Variablenfelder erfolgt mit Hilfe der Anweisung DIM.

Format: DIM Variable (Index, Index, ...)

Beispiel: DIM X(3,4)
Das mit dieser Anweisung dimensionierte zweidimensionale Variablenfeld besteht konkret aus folgenden Variablen:

X(Ø,Ø)	X(Ø,1)	X(Ø,2)	X(Ø,3)	X(Ø,4)
X(1,Ø)	X(1,1)	X(1,2)	X(1,3)	X(1,4)
X(2,Ø)	X(2,1)	X(2,2)	X(2,3)	X(2,4)
X(3,Ø)	X(3,1)	X(3,2)	X(3,3)	X(3,4)

Mit der DIM-Anweisung wurden für diese 2Ø Variablen Speicherplätze reserviert und mit Ø belegt.

Im weiteren Verlauf des Programms kann jede einzelne Variable mit einem Wert belegt werden und damit wie gewohnt gearbeitet werden.

Beispiel: Es ist ein Programm zu erarbeiten, mit dem eine Matrix mit drei Zeilen und vier Spalten aufgebaut wird. Die Variablen A(1,2) und A(2,3) sind mit den Werten 25 und 15 zu belegen.

```
1Ø CLS
2Ø DIM A(2,3)
3Ø A(1,2)=25:A(2,3)=15
4Ø FOR I=Ø TO 2: FOR J=Ø TO 3
5Ø PRINT A(I,J)
6Ø NEXT J: NEXT I
7Ø END
```

So wie diese numerischen Variablenfelder vereinbart werden, können auch Stringvariablenfelder angelegt werden. Diese Art von Dateien können auch über den Recorder eingelesen oder gerettet werden.

5.6.1 ✗Entwickeln Sie ein Programm zur Anfertigung einer Anwesenheitsliste!

5.7. Die ASCII-Code Funktionen

In jedem Computer werden verschiedene Zeichen verwendet. Es sind z. B. Steuerzeichen für die Programmsteuerung, Ziffern und Sonderzeichen, Groß- und Kleinbuchstaben und Grafikzeichen. Jedem Zeichen, jeder Taste und jeder Steuerfunktion wird im Computer nach einem internationalen ASCII-Code (American Standard Code for Information Interchange) umkehrbar eindeutig ein bestimmter Code zugeordnet. Die Codes sind Zahlen zwischen Ø und 255. Zur Ermittlung des zu einem bestimmten Code gehörenden Zeichens dient die Anweisung CHR$(I). Mit dieser Anweisung wird es somit möglich, Zeichen auszudrucken, die der Computer sonst nicht ausdrucken würde. Das trifft z. B. für die Anführungszeichen zu.

Beispiel: Soll auf dem Bildschirm der Ausdruck "TEST" erscheinen, dann ist das nur über die Anweisungen PRINT CHR$(34); "TEST"; CHR$(34) zu realisieren.

5.7.1. Geben Sie die folgenden Programmzeilen ein und vergleichen Sie!

```
1Ø CLS
2Ø PRINT:PRINT, "TEST"
3Ø PRINT:PRINT
4Ø PRINT, CHR$(34); "TEST"; CHR$(34)
5Ø END
```

5.7.2. Ermitteln Sie zu folgenden Codezahlen die dazugehörigen Zeichen: 6Ø, 72, 91, 37!

Mit der Anweisung ASC (String) wird die Codezahl des ersten Zeichens eines String bestimmt.

Beispiel: PRINT ASC("MUTTER") es erscheint 77

5.7.3. Geben Sie folgende Übungsbeispiele ein und verfolgen Sie die Wirkung dieser Anweisungen!

```
PRINT,CHR$(65),CHR$(136)
PRINT,CHR$(34)
PRINT,CHR$(12)
PRINT,ASC("U88Ø")
```

Mit folgendem kleinen Programm kann der gesamte druckbare Zeichenvorrat des Computers aufgelistet werden:

```
1Ø CLS
2Ø FOR I=32 TO 255          (Das Programm beginnt bei I=32, weil mit dem Code
3Ø PRINT CHR$(I);           von 1 bis 31 nichtdruckbare Steuerzeichen belegt wurden.)
4Ø NEXT I
5Ø END
```

Die Codes werden vom Interpreter z. B. beim logischen Vergleich der Größe zweier Strings benötigt, zum alphabetischen Ordnen.

5.7.4. Sehen Sie sich dazu das Programm SORT an!

Bei der Verwendung der bisher erläuterten Anweisungen und der Erarbeitung eines BASIC-Programms gibt es verschiedene Möglichkeiten, ein Programm optimal zu gestalten. In Anlage 8 werden einige Hinweise zur Programmoptimierung angeführt. Sie sind bei der Erarbeitung umfangreicher Programme sehr nützlich.

6. Spezielle Computerfunktionen des KC 85/3 bzw. KC 85/2

6.1. Anweisungen zur Bildschirmgestaltung

Mit grafischen Darstellungen können verschiedene Aussagen und Zusammenhänge anschaulicher gemacht werden. Berechnungen gewinnen an Wert, wenn die Ergebnisse übersichtlich in Tabellen oder in einer Grafik auf dem Bildschirm dargestellt werden. So gehören das Zeichnen und eine sorgfältige Bildschirmgestaltung untrennbar zum Programmieren.

Generell werden fünf Verfahren der Erzeugung einer Grafik auf dem Bildschirm unterschieden:

1. Mit Hilfe der Schriftzeichen, indem Buchstaben, Zahlen oder Sonderzeichen mit Leerzeichen so verbunden werden, daß ein Bild entsteht.

2. Durch Anordnung spezieller im Computerzeichensatz vorhandener Grafiksymbole.

3. Durch Anwendung der Pseudografik, bei der es möglich ist, Zeichen in einer Matrix von meist 8 x 8 Punkten frei zu gestalten.

4. Durch Anwendung der Pixelgrafik, bei der jeder einzelne Bildpunkt gesondert angesprochen werden kann.

5. Durch Anwendung der Vektorgrafik, die aber nur bei speziellen Computern existiert, bei der ein Lichtpunkt direkt bewegt wird (ist nicht auf Fernseher übertragbar).[1]

Der KC 85/3 bzw. KC 85/2 verfügt über die Möglichkeit der Pseudografik und Pixelgrafik sowie in begrenztem Maße auch über die Möglichkeit, mit Schriftzeichen Bilder zu erzeugen. Im folgenden werden einige Anweisungen zur Nutzung dieser Grafikeigenschaften erläutert.

Koordinaten des Bildschirmes

Der Computer schreibt 32 Zeilen mit je 40 Zeichen auf den Bildschirm. Wie im Bild 10 skizziert ist, beginnt die Zählung der Zeilen (Z) und der Spalten (S) im Feld links oben mit Null (Ø), und es gilt:

$$Ø <= Z <= 31$$
$$Ø <= S <= 39$$

Dies ist das Zeichenkoordinatensystem.

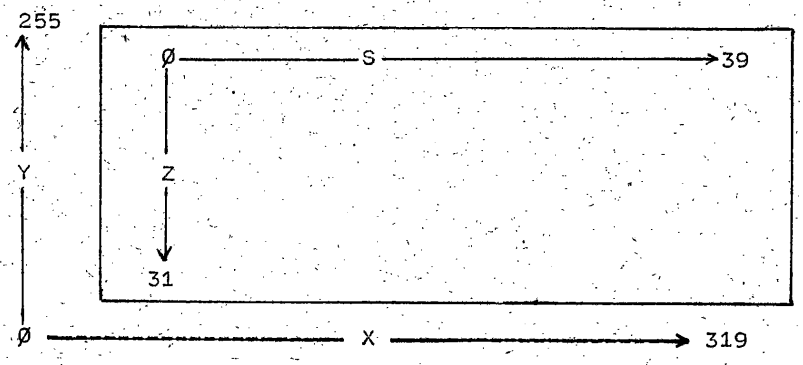


Bild 10: Koordinatensysteme des Bildschirms (vgl. Anlage 5)

1 Vgl. U. Grote, H. Völz: BASIC-Einmaleins des Programmierens. In URANIA-EXTRA, 1987

Ein zweites Koordinatensystem ergibt sich bei Betrachtung der einzelnen Bildpunkte (Pixel). Es ist zu erkennen, daß sich die Ziffern, Buchstaben und Sonderzeichen aus Punkten zusammensetzen. Der Cursor bedeckt gerade ein Feld, in das e i n Zeichen paßt. Dieses Feld besteht aus 8 x 8, also aus 64 Pixel. Somit ergibt sich, daß der Computer über insgesamt 8192Ø Pixel auf dem Bildschirm verfügt.

(32 Zeilen x 8 Pixel + 4Ø Spalten x 8 Pixel = 8192Ø Pixel)

Jeder dieser Bildpunkte kann einzeln angesprochen werden durch Angabe der Koordinaten X und Y. Dieses Koordinatensystem ist das Pixelkoordinatensystem. Sein Ursprung (Nullpunkt) liegt linksunten auf dem Bildschirm.

Die X-Koordinate gibt den Abstand des Pixel vom linken Bildschirmrand (von links nach rechts von Ø bis 319) an, die Y-Koordinate den Abstand vom unteren Bildschirmrand (von unten nach oben von Ø bis 255). Es gelten die Bedingungen:

Ø <= X <= 319
Ø <= Y <= 255 (vgl. Bild 10)

Zwischen Pixel- und Zeichenkoordinatensystem bestehen eindeutige Beziehungen. Es ist einfach zu errechnen, welches Pixel (X, Y) in welchem Zeichenfeld (Z, S) liegt, bzw. welches Zeichenfeld aus welchen Pixeln besteht. Die Umrechnungsformeln lauten:

```
S = INT(X/8)
Z = INT((256-Y)/8)
```

und

```
X = 8*S + Position im Byte
Y = 256-8*Z - Position im Zeichen
```

WINDOW - Anweisung

Mit dieser Anweisung ist es möglich, einen bestimmten Teil des Bildschirmes, ein Fenster festzulegen, in welchem geschrieben wird und auch die Steuerzeichen wirken. Alle Farb- und Schreibanweisungen, die auf eine WINDOW-Anweisung folgen, wirken nur in dem "aktuellen", sprich zuletzt definierten, Fenster. Auch INPUT, LIST, CLOAD u. ä. schreiben nur in das aktuelle Fenster. Nur mit der Anweisung PRINT AT kann ein Ausdruck außerhalb des Fensters positioniert werden.

Format: WINDOW ZA, ZE, SA, SE

Es wird ein Fenster definiert mit den Koordinaten

ZA = erste Zeile des Fensters (oben)
ZE = letzte Zeile des Fensters (unten)
SA = erste Spalte des Fensters (links)
SE = letzte Spalte des Fensters (rechts)

und es gilt: Ø <= ZA <= ZE <= 31
Ø <= SA <= SE <= 39

```
Beispiele: WINDOW Ø,31,Ø,39     : der gesamte Bildschirm ist Schreibfläche
           WINDOW 1Ø,24,Ø,39    : die 15 Zeilen von Zeile 1Ø bis Zeile 24 sind Schreibfläche
           WINDOW               : (also ohne Parameter) ist identisch mit
           WINDOW 1,3Ø,Ø,39.
```

Hinweis: Mit der Anweisung CLS wird immer das Fenster mit den Koordinaten 1,3Ø, Ø,39 gelöscht.

LOCATE - Anweisung

Zur Positionierung des Cursors innerhalb eines definierten Fensters findet die LOCATE-Anweisung Verwendung.

Format: LOCATE Z, S

Z = Zeile im Fenster (Ø<= Z<= ZE-ZA)
S = Spalte im Fenster (Ø<= S<= SE-SA)

Der Cursor wird in bezug zur linken oberen Ecke des aktuellen Fensters positioniert. Der nächste Druck beginnt dann in der Cursorposition, egal, ob eine PRINT-, INPUT- oder LIST-Anweisung den Druck bewirkt oder ob eine Meldung vom Betriebsesystem auf dem Bildschirm erscheint.

Nur PRINT AT ... setzt sich darüber hinweg, oder eine nächste LOCATE-Anweisung positioniert den Cursor neu.

Beispiel: 1Ø WINDOW 1Ø, 15, 5, 2Ø
2Ø LOCATE 2,2:PRINT "FENSTER"

6.1.1 ✱ Entwickeln Sie ein Programm, in dem der Bildschirm in 4 Fenster eingeteilt wird, und beschriften Sie diese Fenster!

Die Verbindung der WINDOW-Anweisung mit den Farbanweisungen bietet wieder neue Möglichkeiten der Bildschirmgestaltung.

Farbgestaltung des Bildschirmes

Auch die farbliche Gestaltung trägt wesentlich zur Erhöhung der Anschaulichkeit eines Bildschirmausdruckes bei. Der KC 85/3 bzw. KC 85/2 besitzt
8 Hintergrundfarben (Code Ø bis 7)
und 14 Vordergrundfarben (Code Ø bis 15, schwarz und weiß sind zweimal vorhanden).
Die Vordergrundfarben können auch blinken, wenn ihr Code um den Wert 16 erhöht wird (Farbcode-Tabelle siehe Anlage 6).

Jedes Zeichenfeld (Größe des Cursors) ist in zwei Farbfelder unterteilt, ein oberes und ein unteres. Das wird beim Cursor sichtbar, wenn die SHIFTLOOK-Dauertaste gedrückt wird. Jedes einzelne Farbfeld ist 8 x 4 = 32 Pixel groß. Es gibt auf dem Bildschirm
4Ø x 32 x 2 = 2560 Farbfelder.

Beim Einfärben des Hintergrundes eines Fensters werden stets beide Farbfelder gleich getönt. Beim Schreiben erhalten auch beide Farbfelder dieselbe Vordergrundfarbe. Aber wird ein einzelnes Pixel eingefärbt, dann erhalten noch die 31 angrenzenden Pixel des Farbfeldes, zu dem das Pixel gehört, diese Farbinformation (über die Vordergrundfarbe).

Das bedeutet nun, daß nur dann zwei Pixel verschieden eingefärbt werden können, wenn sie verschiedenen Farbfeldern angehören, sonst nicht. Schneiden sich zwei oder mehrere Kurven oder kommen sie durch dieselben Farbfelder, dann nehmen alle eingefärbten Pixel die (Vordergrund-) Farbe des Pixel an, welches als letztes eingefärbt wird. Das ergibt dann fleckige Kurven.

Natürlich wäre es besser, wenn jedes Pixel seine eigene Farbe hätte. Doch dazu reicht ein Speicherplatz von 2,5 k Byte nicht aus.

6.1.2. ✱ Berechnen Sie, wieviel Speicherplatz mehr nötig ist, damit jedes Pixel seine Farbe haben kann!

Mit folgenden Anweisungen kann die farbliche Gestaltung realisiert werden:

Format: PAPER H — bewirkt, daß der Hintergrund, auf welchem geschrieben wird, die mit dem Code Ø< = H< =7 festgelegte Farbe erhält. ("paper" zu deutsch "Papier")

Beispiel: PAPER 5

Format: INK V — bewirkt, daß der Vordergrund die mit dem Code Ø <= V <=15 (31) fixierte Farbe erhält. Alle Zeichen werden damit gedruckt. ("ink" zu deutsch "Tinte")

Beispiel: INK 7

Format: COLOR V, H — verknüpft die Anweisungen INK und PAPER und stellt Vorder- und Hintergrundfarbe ein. ("color" zu deutsch "Farbe")

Beispiel: COLOR Ø,2

Erst nach der Anweisung CLS wird das aktuelle Fenster bzw. der gesamte Hintergrund in der zuletzt definierten Hintergrundfarbe eingefärbt.

6.1.3. Sehen Sie sich die möglichen Farbkombinationen im Programm HALLO an!

6.1.4. Geben Sie jedem Fenster aus der Aufgabe 6.1.1. eine andere Farbe!

Es wird empfohlen, Fenster- und Farbanweisungen im BASIC-Programm an den Anfang eines Anweisungsblockes und so nahe beieinander zu schreiben, daß man auf einen Blick erkennt, welcher Bildschirmausschnitt behandelt wird.

Die folgenden Beispielprogramme demonstrieren noch einmal die Verwendung der Anweisungen zur Bildschirmgestaltung. Beispiel 1 demonstriert die Arbeit mit Fenstern und Beispiel 2 die Darstellung einer Tabelle.

Beispiel 1:

```
 1Ø ! WINDOW-Beispiel
 2Ø WINDOW Ø,31,Ø,39:COLOR7,1:CLS
 3Ø FOR A=Ø TO 15 STEP 2
 4Ø WINDOW A,31-A,A,39-A:PAPER7-A/2:CLS
 5Ø NEXT
 6Ø FOR XY=Ø TO 127
 7Ø PSET XY,XY,7:PSET XY,255-XY
 8Ø PSET 319-XY,XY:PSET 319-XY,255-XY
 9Ø NEXT
1ØØ WINDOWØ,31,Ø,39:COLOR7,1
11Ø PRINTAT(16,18);"ENDE":PAUSE3Ø
12Ø CLS:LOCATE15,18:END
```

Beispiel 2:

```
 1Ø ! Tabellen-Beispiel
 2Ø WINDOWØ,31,Ø,39:COLOR7,1
 3Ø FOR K=Ø TO 4
 4Ø     CLS:LOCATE2,Ø
 5Ø     PRINT" n";TAB(5);" n  2";TAB(11)" n  3"
 6Ø     PRINT STRING$Ø(18,"-")
 7Ø FOR I=2Ø*K+1 TO 2Ø*(K+1)
 8Ø     PRINT I;TAB(5);I*I;TAB(11);I*I*I
 9Ø NEXT
1ØØ  LOCATE14,2Ø:PRINT COLORØ,7;"Tabelle";K+1
11Ø  LOCATE16,2Ø:PRINT"2. und 3. Potenzen"
12Ø  LOCATE17,2Ø:PRINT"von n = ";2Ø*K+1;"bis";
     2Ø*(K+1)
13Ø  LOCATE25,Ø:INPUT"";Q$
14Ø NEXT
15Ø INPUT"";Q$:GOTO2Ø
```

Hinweis zum Tabellen-Beispiel:

Mit Zeile 13Ø wird erreicht, daß die Rechnung unterbrochen wird, wenn der Bildschirm voll ist. Ist ein Drucker angeschlossen, so kann man jetzt auch eine Hardcopy anfertigen. (Mit der Anweisung PAUSE anstelle von INPUT wäre das nicht möglich.) Diese Programmzeile ist auch in den folgenden Grafikprogrammen zu finden.

6.1.4. Analysieren Sie Ihre bisher erstellten Programme hinsichtlich der Möglichkeit einer besseren grafischen Gestaltung!

6.2. Anweisungen zur Gestaltung einer einfachen Computergrafik

Die folgenden Anweisungen zur grafischen Gestaltung bauen auf der Pixelgrafik des KC 85/3 bzw. KC 85/2 auf. Das sind die Anweisungen, die sich auf das Pixelkoordinatensystem beziehen. Ihnen allen ist gemeinsam, daß sie sich nicht um WINDOW- und Coloranweisungen kümmern. Sie positionieren die Pixel absolut und bestimmen unabhängig von allem die Farbe der Pixel.

PSET - Anweisung

Mit der Anweisung PSET wird ein einzelnes Pixel gesetzt (engl. point set = Punkt setzen).

Format: PSET X,Y,F — X und Y geben die Koordinaten des Pixelkoordinatensystems an und F ist die Vordergrundfarbe, in der das Pixel eingefärbt wird.

Für die Parameter X, Y und F gelten folgende Einschränkungen:

$0 <= X <= 319$
$0 <= Y <= 255$
$0 <= F <= 31$

Werden diese Zahlenbereiche über- oder unterschritten, so bricht das Programm mit SN-ERROR ab!

6.2.1. Geben Sie Bildpunkte mit unterschiedlichen Koordinaten ein!

PRESET - Anweisung

Mit der Anweisung PRESET wird ein einzelnes Pixel gelöscht (engl. point reset = Punkt zurücksetzen).

Format: PRESET X,Y — X und Y sind die Koordinaten des Pixels, welches gelöscht, radiert wird.

Es wird damit möglich, in einer Zeichnung oder einer Konstruktion Korrekturen vorzunehmen. Mit diesen Anweisungen können jedoch nicht nur einzelne Punkte gesetzt und gelöscht werden, sondern auch Linien und ganze Figuren gezeichnet werden. Das Zeichnen einer Linie wird z. B. durch eine Schleifenanweisung erreicht.

Beispiel:
```
10 CLS
20 FOR X=0 TO 100:Y=50
30 PSET X,Y,7
40 NEXT X
50 END
```

Der Computer zeichnet, indem er Pixel für Pixel der Linie mit der Vordergrundfarbe einfärbt. Das geht mit dem BASIC-Interpreter relativ langsam. Aber es hat den Vorteil, daß man zusehen kann wie die Grafik entsteht.

Zwei weitere Grafikanweisungen ermöglichen ein schnelleres Zeichnen:

LINE - Anweisung

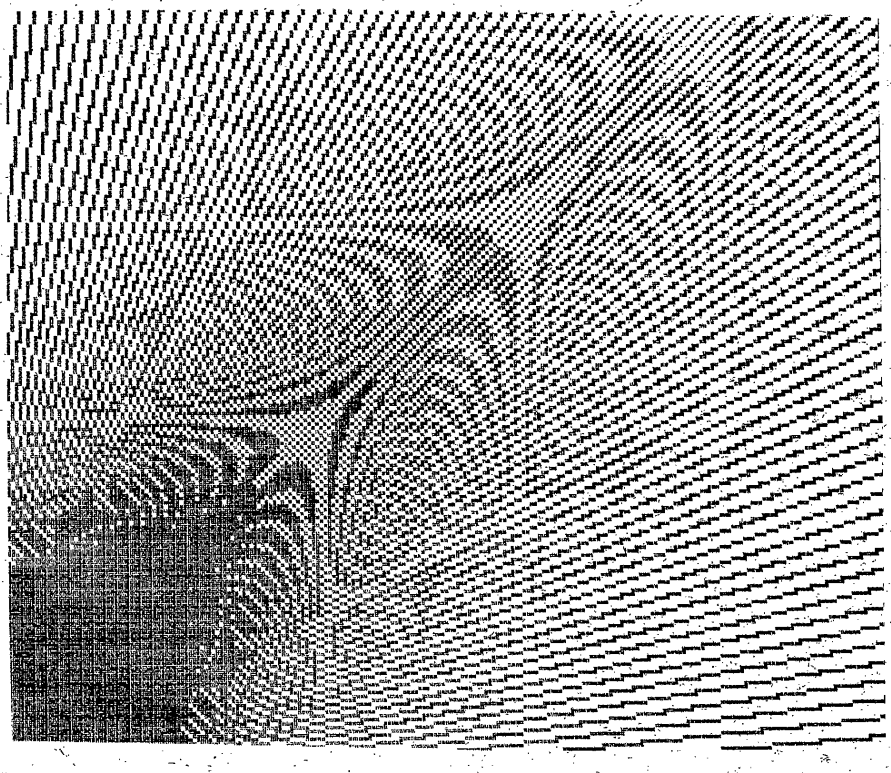
Diese Anweisung zeichnet eine Gerade zwischen zwei anzugebenden Punkten.

Format: LINE XA,YA,XE,YE(F)

Die Linie wird gezeichnet vom Anfangspunkt mit den Koordinaten XA und YA bis zum Endpunkt mit den Koordinaten XE und YE. F ist der Code der Farbe, in welcher gezeichnet werden soll. Wurde schon in vorangegangenen Anweisungen eine Vordergrundfarbe vereinbart, die beibehalten werden soll, so kann dieser Farbcode entfallen.

Das Programm "Linien" zeichnet gleich zwei Scharen dieser Geraden, und es entstehen Interferenzbilder, was schon eine erste Computergrafik ergibt:

```
1Ø ! Linien
2Ø WINDOW Ø,31,Ø,39:COLOR7,Ø:CLS
3Ø FOR I=Ø TO 255 STEP 5
4Ø LINE Ø,Ø,I,255,7
5Ø LINE Ø,Ø,255,I
6Ø NEXT
7Ø LOCATE3Ø,35:INPUT"";Q$:GOTO 1Ø
8Ø END
```



6.2.2. Entwickeln Sie ein eigenes Programm unter Verwendung der Anweisung LINE!

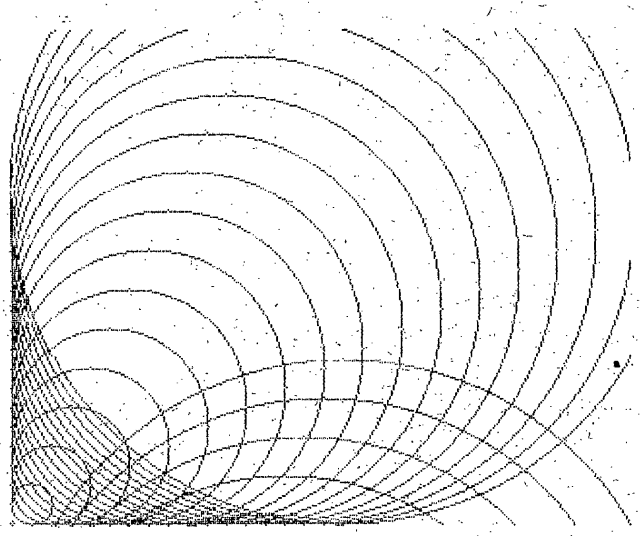
CIRCLE - Anweisung

Zum schnellen Zeichnen von Kreisen kann die Anweisung CIRCLE verwendet werden.

Format: CIRCLE X,Y,R(,F)

Es wird ein Kreis mit dem Mittelpunkt X,Y und dem Radius R gezeichnet. Mit dem Farbcode F kann die gewünschte Vordergrundfarbe bestimmt werden. Das Beispielprogramm "CIRCLE" demonstriert diese Anweisung. Auch dabei entsteht eine einfache Computergrafik.

```
1Ø ! CIRCLE
2Ø WINDOW Ø,31,Ø,39:COLOR 7,1:CLS
3Ø FOR I=1 TO 18Ø STEP 1Ø
4Ø CIRCLE I,I,I,7
5Ø NEXT
6Ø INPUT"";Q$: GOTO 1Ø
7Ø END
```



6.2.3. Entwickeln Sie ein eigenes Programm unter Verwendung der Anweisung CIRCLE!

Hinter beiden BASIC-Anweisungen stehen Routinen im Maschinencode, durch welche das pixelweise Einfärben erledigt wird. Obwohl das dieselbe Technik ist, wie das Setzen der Einzelpunkte in BASIC-Programmen, geht das mehr als hundertmal so schnell.

6.2.4. Sehen Sie sich das Programm MC/BASIC der Programmsammlung an, und vergleichen Sie die Zeichengeschwindigkeit der einzelnen Linien!

Das Programm "Koordinatenkreuz durch Mx,My mit Hilfslinien" zeichnet ein Netz auf den Bildschirm, wie es etwa vom Millimeterpapier bekannt ist. Es soll deutlich machen, daß die Anweisungen LINE und PSET nebeneinander selbst beim Zeichnen einfacher Geraden volle Berechtigung haben. Die Hilfsgeraden ergeben sich dadurch, daß nicht Punkt neben Punkt eingefärbt wird, sondern mit STEP in der FOR-TO-NEXT-Schleife eine 'dünne' Linie erzeugt wird. Mit LINE ist das nicht möglich. Wie bereits an anderer Stelle erklärt wurde, ist es nicht ratsam, die Hilfslinien durch eine zweite Farbe vom Koordinatenkreuz abzuheben. Das Koordinatensystem würde dadurch fleckig und damit auch die Konstruktion, die man später in das Netz zeichnen würde.

```
 1Ø ! Koordinatenkreuz mit Hilfslinien
 2Ø WINDOWØ,31,Ø,39:CLS
 3Ø LOCATE2,1:INPUT"mx(<32Ø)=";MX:IF MX<Ø OR MX>319 GOTO3Ø
 4Ø LOCATE3,1:INPUT"my(<256)=";MY:IF MY<Ø OR MY>255 GOTO4Ø
 5Ø CLS
 6Ø LINEØ,MY,319,MY,7:LINE MX,Ø,MX,255
 7Ø IF MY+2Ø>255 GOTO9Ø
 8Ø FOR Y=MY+2Ø TO 255 STEP 2Ø:GOSUB 2ØØ:NEXT
 9Ø IF MY-2Ø< Ø  GOTO11Ø
1ØØ FOR Y=MY-2Ø TO Ø STEP-2Ø:GOSUB 2ØØ:NEXT
11Ø IF MX-2Ø< Ø GOTO13Ø
12Ø FOR X=MX-2Ø TO Ø STEP-2Ø:GOSUB 18Ø:NEXT
13Ø IF MX+2Ø> 319 GOTO15Ø
14Ø FOR X=MX+2Ø TO 319 STEP 2Ø:GOSUB 18Ø:NEXT
15Ø PRINTAT(1,1);"mx=";RIGHT$(STR$(MX),3);",my=";
    RIGHT$(STR$(MY),3)
16Ø INPUT"";Q$:GOTO2Ø
17Ø ! Waagerechte Hilfslinien
18Ø FOR Y=Ø TO 255 STEP 3:PSET X,Y:NEXT:RETURN
19Ø ! Senkrechte Hilfslinien
2ØØ FOR X=Ø TO 319 STEP 3:PSET X,Y:NEXT:RETURN
```

Die vielen IF-Anweisungen im Programm bedürfen der Erklärung. Sie alle dienen dem Zweck, daß nur solche Werte von X und Y in die Anweisung PSET gelangen, die im oben genannten Zahlenbereich liegen. In den Zeilen 3Ø und 4Ø reagiert das Programm einfach abweisend auf solche Zahlen. Es nimmt sie gar nicht erst an, wenn der Nullpunkt des Koordinatenkreuzes nicht im Bildschirm liegt.

In Zeile 7Ø wird geprüft, ob die erste senkrechte Hilfslinie rechts vom Nullpunkt des Achsenkreuzes noch auf dem Bildschirm liegt. Ist das nicht der Fall, dann wird Zeile 8Ø übersprungen. Das Programm weicht der Absturzgefahr aus, es 'flieht' vor ihr. Daher haben diese Programmteile den Namen "Fluchtfunktionen" oder auch "Escape-Funktionen".

Ein Programmabsturz wegen Verletzung der Regeln der Programmiersprache gilt als einer der schwersten Programmierfehler. Der Programmierer muß diese Fallen kennen und dafür sorgen, daß kein Benutzer seines Programms in solche Falle geraten kann. Er baut also Fluchtfunktionen ein. Nur wenige Programme kommen ohne sie aus, aber in Grafikprogramme gehören sie mit Sicherheit.

PTEST - Anweisung

Die letzte hier zu erklärende Grafikanweisung ist PTEST.

Format: PTEST(X)

Sie prüft, ob ein Pixel X,Y gesetzt ist oder nicht. Ist das Pixel gesetzt, ergibt sie den Wert 1, ist das Pixel nicht gesetzt, den Wert Ø.

Den X-Wert des zu prüfenden Pixel kann man in die Anweisung einsetzen. Den Y-Wert entnimmt das Programm der zuletzt abgearbeiteten Grafikanweisung.

In folgendem Beispielprogramm wird die Wirkung dieser Anweisung demonstriert. Das Programm bewirkt die Vergrößerung einer eingegebenen Zeichenkette.

Beispiel:
```
1Ø COLOR 1,1:CLS:WINDOW
2Ø INPUT A$:COLOR 7,1
3Ø FOR K=7 TO Ø STEP-1
4Ø PRESET Ø,24Ø+K
5Ø FOR I=Ø TO39
6Ø A=PTEST (16+I)
7Ø IF A=1 THEN PRINTAT (2Ø-K,I);CHR$ (91)
8Ø NEXT I
9Ø NEXT K
1ØØ END
```

Nach Eingabe einer Zeichenkette von maximal 5 Zeichen (z. B. HE DU oder HALLO) wird diese Zeichenkette vergrößert. Dabei werden durch Zeile 6Ø die X-Werte geprüft. Zeile 4Ø ist unbedingt erforderlich, denn dort werden die Y-Koordinaten eingestellt. Ohne diese Programmzeile wäre das Programm nicht lauffähig.

6.2.5. Geben Sie andere Zeichenketten ein! Durch Änderung des Zeichencodes in Zeile 7Ø können Sie auch andere Zeichen zur Vergrößerung nutzen (z. B. CHR$ (143).

6.2.6. Sehen Sie sich die Wirkung dieser Anweisung auch an Hand des Programms "Sonne" im BASIC-Handbuch des KC 85/3 an!

Alle bisherigen Programmbeispiele dieses Kapitels waren bereits einfache Computergrafiken. Doch um noch weiter in dieses Gebiet einzudringen, ist es notwendig, einige mathematische Zusammenhänge zu erkennen, mit deren Hilfe noch andere Figuren mit der Anweisung PSET konstruiert werden können.

In dem folgenden kleinen Exkurs sollen vor allem für Interessierte Hinweise zu weiteren Möglichkeiten auf dem Gebiet der Computergrafik gegeben werden.

Zeichnen von Kreisen

Es soll mit PSET ein Kreis gezeichnet werden. M sei sein Mittelpunkt mit den Koordinaten MX und MY, kurz: M(MX,MY). Der Radius des Kreises sei R. Bekanntlich ist R der Abstand jedes Punktes der Peripherie vom Mittelpunkt.

In Bild 11 ist ein Kreispunkt in ein Achsenkreuz xy gezeichnet worden, und es wurden einige Bezeichnungen für wichtige Strecken gewählt und eingetragen.

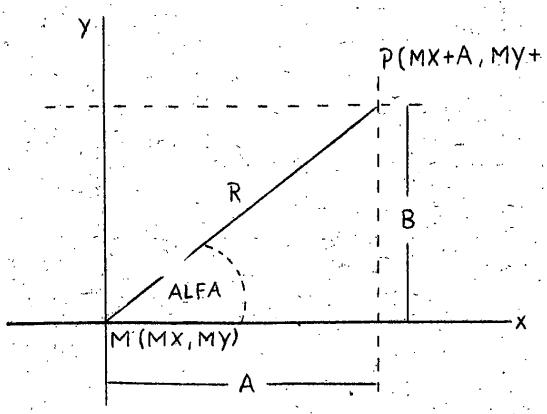


Bild 11: Beziehungen im Kreis

In Bild 11 bilden der Radius und die Achsenabschnitte A und B des Kreispunktes P ein rechtwinkliges Dreieck. Also gilt für sie der Satz des Pythagoras, nämlich:

$$A^2 + B^2 = R^2 \qquad (1)$$

Radiziert man diese Gleichung, d. h. zieht man die Wurzel aus der linken und rechten Seite der Gleichung, und vertauscht man die Seiten, so ergibt sich:
(in BASIC Schreibweise ist SQR(X) = Quadratwurzel aus X)

```
R = SQR (A*A + B*B)     (2)
```

In Bild 11 erkennt man auch die Ableitung der trigonometrischen Funktionen. Winkel ALFA ist eingezeichnet. Das Streckenverhältnis B zu R (geschrieben B/R) wird 'Sinus von ALFA' genannt und das Verhältnis A/R 'Cosinus von ALFA'. Als Formeln geschrieben:

B/R = SIN(ALFA). Mit R multipliziert:

```
B = R * SIN (ALFA)
```

Entsprechend:

```
A = R * COS (ALFA)      (3)
```

Auch hier wurde die BASIC-Schreibweise benutzt. Denn in BASIC gibt es die Funktionen SIN(X) und COS(X). Der Computer rechnet und schreibt also die Werte dafür aus. Bedingung ist, daß der Winkel ALFA in Bruchteilen von 2*PI, man sagt "im Bogenmaß" angegeben wird. Wenn der Radius die Länge 1 hat (R=1), ist 2*PI der Umfang des Kreises.

Der Bogen verhält sich zu 2*PI, wie sich der Winkel zu 360 Grad verhält. Auch das als Formel geschrieben ergibt:

Bogen / 2*PI = Winkel / 360 . Mit 2*PI multipliziert:

```
Bogen = Winkel * PI / 180     (4)
```

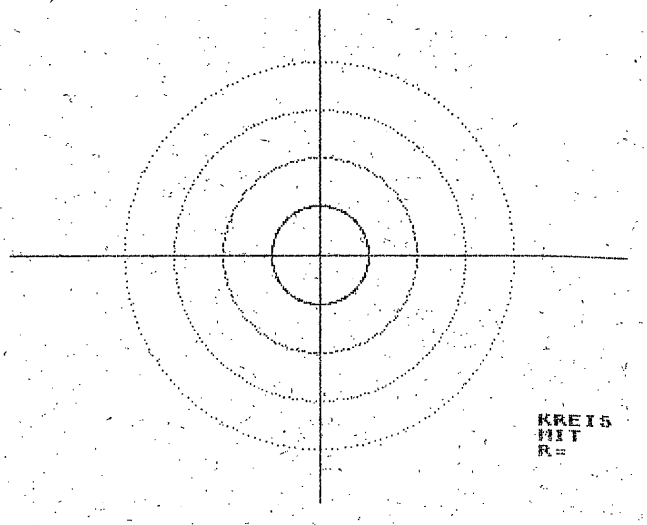
Doch zurück zu den beiden Gleichungen von (3). Sie sind fast alles, was man für das Kreisprogramm wissen muß. Läßt man nämlich ALFA alle Werte von 0 bis 2*PI durchlaufen, so erhält man alle Punkte der Peripherie.

Natürlich kann der Schritt der Schleife nicht 1 sein, sondern er muß einen kleinen Bruchteil von 2*PI betragen. Wird der zweihundertste Teil gewählt, so wird die Peripherie durch 200 Pixel markiert. Das genügt in den meisten Fällen. Also:

```
STEP PI / 100
```

Um wichtige Zusammenhänge deutlich zu machen, soll das Kreisprogramm auch das Koordinatenkreuz durch den Mittelpunkt des Kreises mitzeichnen. Der Mittelpunkt des Kreises soll mitten auf dem Bildschirm liegen, d. h. MX = 160, MY = 128. Der Radius soll frei wählbar sein. Das alles ist in dem Beispiel "Koordinatenkreuz und Kreis" realisiert.

```
1Ø ' Koordinatenkreuz und Kreis
2Ø WINDOWØ,31,Ø,39:COLOR7,1:CLS
3Ø MX=16Ø:MY=128
4Ø LINEØ,MY,319,MY,7:LINEMX,Ø,MX,255
5Ø WINDOW25,31,33,39:COLOR7,3
6Ø CLS:PRINT:PRINT"Kreis  mit":INPUT"R=";R
7Ø FOR ALFA=Ø TO 2*PI STEP PI/1ØØ
8Ø A=R*COS(ALFA):B=R*SIN(ALFA)
9Ø X=INT(MX+A+.5):Y=INT(MY+B+.5)
1ØØ IF X< Ø OR Y< Ø OR X>319 OR Y>255 GOTO12Ø
11Ø PSETX,Y
12Ø NEXT
13Ø GOTO 6Ø
14Ø END
```



In Zeile 4Ø wird das Achsenkreuz durch den Kreismittelpunkt gezeichnet,
in Zeile 6Ø wird mit INPUT die Möglichkeit gegeben, einen Radius beliebiger Länge einzugeben,
in Zeile 7Ø beginnt der Zeichenzyklus für den Kreis. Der Zyklus geht bis Zeile 12Ø.
Zeile 8Ø enthält die Formeln (3).
Aus A ergibt sich X, welches in PSET benötigt wird, dadurch, daß noch MX, der Abstand des Kreismittelpunktes vom linken Bildschirmrand, addiert wird.
Entsprechend ergibt sich Y aus B.
In Zeile 1ØØ erkennt man eine Fluchtfunktion. Sie schützt das PSET in Zeile 11Ø vor zu kleinen und zu großen Werten,
Zeile 13Ø wird dann erreicht, wenn der Kreis vollständig gepixelt wurde. Sie führt wieder auf Zeile 6Ø, d. h. daß nun ein anderer Radius eingegeben werden und damit ein Kreis in die vorhandene Zeichnung gezogen werden kann.

Hinweis:

In Zeile 9Ø wird nicht nur MX zu A und MY zu B addiert, sondern es werden auch mathematische Rundungen durchgeführt. Warum?

Die Werte X und Y in PSET müssen Integerzahlen sein. Hier werden sie dadurch gebildet, daß der Betrag von Ø,5 addiert und dann die Funktion INT(X) von BASIC auf den Ausdruck angewendet wird. INT schneidet den gebrochenen Teil der Zahl ab. Wegen der computer-internen Darstellung positiver und negativer Zahlen folgt:

aus +25.2 wird 25, aus -25.2 wird -26.

Addiert man nun aber zu den Zahlen jeweils +Ø,5 (in BASIC: .5), dann wird INT auf +25.7 und auf -24.7 angewendet und es ergibt sich einmal +25, das andere mal -25. Die Beträge sind gleich, und so muß es beim Kreis sein: Der Radius ist in allen Richtungen gleich lang. Es ist somit erforderlich, bei Benutzung der Anweisung PSET mathematisch zu runden.

Zeichnen von Geraden

Der Kreis entsteht dadurch, daß zunächst ein Punkt im Abstand R vom Mittelpunkt gesetzt wird, dann wird der Winkel etwas geändert und ein weiterer Punkt gesetzt usw.

Wenn ein Winkel gewählt und konstant gehalten wird und der Radius von Ø bis zu seiner Länge R geändert wird, dann muß die Spur des Radius, also ein Strahl vom Mittelpunkt zu einem Punkt der Peripherie entstehen.

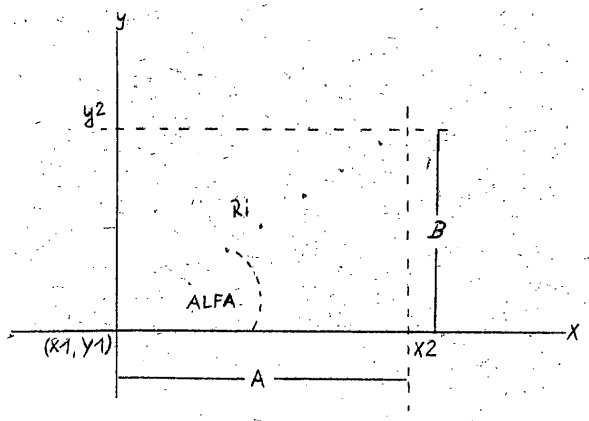


Bild 12: Entstehung eines Strahls (ALFA=const, R variiert)

Bild 12 soll das deutlich machen. Ri bezeichnet den veränderlichen Radius und die Punkte deuten gesetzte Pixel an.

Im Programm "Koordinatenkreuz und Kreis" müssen Radius R und Winkel ALFA die Rollen und die Plätze tauschen. Dazu werden die Zeilen 6Ø und 7Ø überschrieben. Sie sollen lauten:

```
6Ø CLS:PRINT:PRINT" Strahl mit " :INPUT"ALFA="; AF
65 ALFA = AF * PI/18Ø
7Ø FOR R=Ø TO 12Ø STEP 4
```

Zeile 65 wurde eingefügt, damit der Winkel in Grad eingegeben werden kann. Die Umrechnung wurde in Formel (4) erklärt.

Nun zeichnet das Programm beliebig gerichtete Strahlen. Wie man vom Strahl zur Geraden zwischen zwei Punkten gelangt, ist in Bild 11 abzulesen.

Es wird nur e i n Strahl betrachtet. Der Nullpunkt des Koordinatensystems ist sein Ausgangspunkt. Er sei P1(X1,Y1). Der Endpunkt sei P2(X2,Y2). Die Koordinaten X1, Y1, X2 und Y2 sind auf den Achsen in Bild 11 abgetragen. Man sieht:

```
A = X2 - X1 und
B = Y2 - Y1                (5)
```

Formel (2) liefert die Beziehung:

```
R = SQR(A*A + B*B)         (6)
```

Aus Formel (3) ergibt sich durch Umformung und Umbenennung:

```
COS(ALFA) = A/R , oder kurz: C = A/R
SIN(ALFA) = B/R              S = B/R      (7)
```

Werden diese Erkenntnisse berücksichtigt, so versteht man das Programm "Gerade von P1 nach P2".

```
 1Ø ! Gerade von P1 nach P2
 2Ø WINDOWØ,31,Ø,39:COLOR7,1:CLS
 3Ø LOCATEØ,Ø:INPUT" X1=";X1:INPUT" Y1=";Y1:PRINT
 4Ø INPUT" X2=";X2:INPUT" Y2=";Y2
 5Ø A=X2-X1:B=Y2-Y1
 6Ø IF A=Ø AND B=Ø GOTO15Ø:! Escape(/Ø)
 7Ø R=SQR(A^2+B^2)
 8Ø C=A/R:S=B/R
 9Ø FOR RS=Ø TO R STEP1
1ØØ AS=RS*C:BS=RS*S
11Ø X=INT(X1+AS+.5):Y=INT(Y1+BS+.5)
12Ø IF X>Ø AND Y>Ø AND X<32Ø AND Y<256 THEN PSET X,Y,7
13Ø NEXT
14Ø GOTO3Ø
15Ø PRINT:PRINTINK23;" P1=P2!":PAUSE3Ø
16Ø LOCATE CSRLIN(Ø)-1,1:PRINT"        ":GOTO3Ø
17Ø END
```

Wie die Anweisung LINE benötigt das Programm als Eingabeparameter die Koordinaten der zu verbindenden Punkte. In den Zeilen 3Ø und 4Ø sind sie zu übergeben.

In den Zeilen 5Ø bis 8Ø werden aus den Eingabewerten alle für die Geraden-Konstruktion benötigten konstanten Werte berechnet, und dann folgt in den Zeilen 9Ø bis 13Ø der Zyklus für die Pixelung der Geraden.

Zeile 6Ø ist eine Fluchtfunktion. Sie verhindert, daß in Zeile 8Ø durch Ø dividiert wird. Dieser Fall würde eintreten, wenn P1 und P2 identische Punkte wären. Tritt das auf, so zeigt Zeile 15Ø das blinkend auf dem Bildschirm an, und Zeile 16Ø besorgt, daß die blinkende Schrift wieder verschwindet.

Mit dem Programm "Achsenkreuz und Gerade" wird ein vervollkommnetes Programm zur Verfügung gestellt. Es enthält den Geradenzeichner als Unterprogramm in den Zeilen 5ØØ bis 59Ø. Diese Subroutine kann man isolieren und in andere Programme einfügen.

```
 1Ø ! Achsenkreuz und Gerade
 2Ø WINDOWØ,31,Ø,39:PAPER1:CLS
 3Ø WINDOW28,31,Ø,39:COLOR7,4
 4Ø PRINTAT(1,2);"Gerade P1 - P2"
 5Ø ! P1 mit Achsenkreuz
 6Ø CLS:INPUT"X1=";X1
 7Ø IF X1<Ø OR X1>319 GOTO6Ø:! Escape
 8Ø LOCATE1,Ø:INPUT"Y1=";Y1
 9Ø IF Y1<Ø OR Y1>255 GOTO8Ø:! Escape
1ØØ LINE Ø,Y1,319,Y1,7:LINE X1,Ø,X1,255
11Ø ! Gerade P1-P2 (über UP)
12Ø LOCATEØ,1Ø:INPUT"X2=";X2
13Ø LOCATE1,1Ø:INPUT"Y2=";Y2
14Ø IF X2>=Ø AND Y2>=Ø AND X2<32Ø AND Y2<256 THEN PSET X2,Y2,23
15Ø GOSUB5ØØ:! Sprung ins UP Linie
16Ø INPUT"";Q$
17Ø CLS:PRINT"P1 und P2 neu :  <1>"
18Ø     PRINT"nur    P2 neu :  <2>"
19Ø     PRINT"neue Zeichnung:  <3>"
2ØØ LOCATE1,22:INPUT" ? ";WL
21Ø IF WL<1 OR WL>3 GOTO2ØØ
22Ø ON WL GOTO6Ø,23Ø,2Ø
23Ø CLS:PRINT"X1=";X1:PRINT"Y1=";Y1:GOTO12Ø
24Ø END
```

```
5ØØ ! Subroutine "Linie P1-P2"                    (Parameter: X1,Y1,X2,Y2)
51Ø A=X2-X1:B=Y2-Y1:IF A=Ø AND B=Ø GOTO59Ø
52Ø R=INT(SQR(A*A+B*B)+.5)
53Ø C=A/R:S=B/R
54Ø FOR RS=Ø TO R STEP1
55Ø AS=RS*C:BS=RS*S
56Ø X=INT(X1+AS+.5):Y=INT(Y1+BS+.5)
57Ø IF X>=Ø AND Y>=Ø AND X<32Ø AND Y<256 THEN PSETX,Y,7
58Ø NEXT
59Ø RETURN
```

Variationen des Kreisprogramms

Durch geringfügige Änderungen in dem Programm "Koordinatenkreuz und Kreis" erreicht man neue Qualitäten bis zu völlig anderen Darstellungen. Einige sollen gezeigt werden, bezogen auf den Quelltext des genannten Programms.

1. Veränderung (Translationen):
Durch Verschiebung des Mittelpunktes kann der ganze Kreis, die ganze Konstruktion verschoben werden. Um das zu zeigen, wird Zeile 14Ø verändert und ergänzt. Zeile 13Ø muß gelöscht werden.

```
14Ø INPUT"M?(J)";Q$:IF Q$="J" THEN Q$="":GOTO 16Ø
15Ø GOTO 7Ø
16Ø CLS:INPUT"MX=";MX:IF MX<Ø OR MX>319 GOTO 16Ø
17Ø LOCATE 1,Ø:INPUT"MY=";MY:IF MY<Ø OR MY>255 GOTO 17Ø
18Ø GOTO 7Ø
```

Werden nun neue Werte für MX und MY eingegeben und die Eingaben für R= mit ENTER übergangen, so wird der zuvor gezeichnete Kreis verschoben darüber gezeichnet. Es ergibt sich damit eine Vorstellung, wie eine Translation durchzuführen ist.

2. Veränderung (Sinus-Funktion):
Zeile 8Ø wird überschrieben mit:

```
8Ø A=2Ø*ALFA:B=R*SIN(ALFA)
```

Das Programm zeichnet nun eine Sinuskurve. R ist die Amplitude. Die Funktionen Sinus und Cosinus heißen ja auch "Schwingungsfunktionen". Beim Zeichnen helfen sie, die Pixel zwischen zwei Extremwerten pendeln zu lassen.

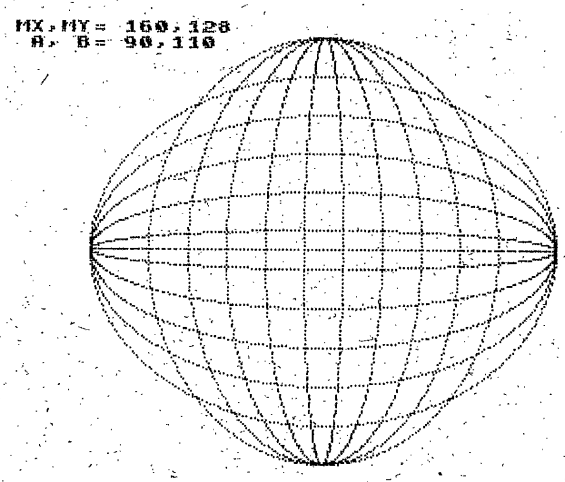
3. Veränderung (Ellipse):
Zeile 8Ø wird geändert in:

```
8Ø A=R*COS(ALFA):B=R*SIN(ALFA)*2/3
```

Das Programm zeichnet eine liegende Ellipse.

Die nahe Verwandtschaft von Kreis und Ellipse ist aus dem Geometrie-Unterricht bekannt. Statt eines Radius R sind die beiden Halbachsen A und B für die Konstruktion wichtig. Wenn A=B ist, wird aus der Ellipse ein Kreis. Das folgende Beispielprogramm für die Ellipsen-Konstruktion zeichnet ziemlich schnell. Das wird dadurch erreicht, daß die Berechnungen der Punktlagen auf das geringst mögliche Maß reduziert werden. Wegen der hohen Symmetrie der Ellipse ist es nämlich nur nötig, die Peripheriepunkte zwischen Ø und PI/4 zu berechnen, also nur für 1/8 des Umfangs. Aus jedem dieser Punkte ergeben sich 7 weitere durch Spiegelung an den Achsen (3) und Zentrosymmetrie (4). Im Programm "Ellipse" werden in den Zeilen 13Ø bis 25Ø diese Fakten programmtechnisch umgesetzt. Durch Kenntnis und Anwendung der Mathematik kann man schneller und besser zeichnen.

```
1Ø  ! ELLIPSE
2Ø  WINDOWØ,31,Ø,39:COLOR7,1:CLS
3Ø  P=.785399:ST=PI/18Ø:F=7
4Ø  ! Eingabe der Parameter
5Ø  LOCATE1,Ø:INPUT"MX,MY=";MX,MY
6Ø  IF MX<Ø OR MX>319 OR MY<Ø OR MY>255 GOTO5Ø
7Ø  LOCATE2,Ø:INPUT" A, B=";A,B
8Ø  IF A<Ø OR B<Ø GOTO7Ø
9Ø  ! Ellipsen-Berechnung
1ØØ FOR W=Ø TO P STEP ST
11Ø S=SIN(W):C=COS(W)
12Ø ! Nutzung der Symetrie,Rundung
13Ø AS=INT(A*S+.5):AC=INT(A*C+.5)
14Ø BS=INT(B*S+.5):BC=INT(B*C+.5)
15Ø X1=MX+AS:X2=MX-AS:X3=MX+AC:X4=MX-AC
16Ø Y1=MY+BC:Y2=MY-BC:Y3=MY+BS:Y4=MY-BS
17Ø ! Punktsetzen mit Escapefunktion
18Ø IFX1<32Ø AND Y1<256 THEN PSETX1,Y1,7
19Ø IFX2>Ø   AND Y1<256 THEN PSETX2,Y1
2ØØ IFX1<32Ø AND Y2>Ø   THEN PSETX1,Y2
21Ø IFX2>Ø   AND Y2>Ø   THEN PSETX2,Y2
22Ø IFX3<32Ø AND Y3<256 THEN PSETX3,Y3
23Ø IFX4>Ø   AND Y3<256 THEN PSETX4,Y3
24Ø IFX3<32Ø AND Y4>Ø   THEN PSETX3,Y4
25Ø IFX4>Ø   AND Y4>Ø   THEN PSETX4,Y4
26Ø NEXT:INPUT"";Q$
27Ø !Ueberzeichnen oder neu?
28Ø IF Q$<>""THEN Q$="":GOTO2Ø
29Ø GOTO5Ø
3ØØ END
```



4. Veränderung (Lissajou-Figuren):

Zeile 8Ø des Programms von Seite 63 wird geändert in:

```
8Ø A=R*SIN(2*ALFA):B=R*SIN(3*ALFA)
```

Der sogenannte Schmetterling entsteht, eine Lissajou'sche Figur.

5. Veränderung (Kreis- (und Ellipsen-) Bögen):

Das Programm erhält wieder die ursprüngliche Form, es sollen Kreise gezeichnet werden, genauer: Kreisbögen. Ein Programm soll entstehen, welches Kreisbögen mit dem Redius R vom Winkel A1 bis zum Winkel A2 zeichnet. Es soll gelten: A1< =A2 und die Winkel sollen in Grad angegeben werden. Diese Forderungen kann man erfüllen, wenn z. B. zwischen Zeile 6Ø und 7Ø einige Zeilen eingefügt werden und Zeile 7Ø geändert wird.

```
62 CLS:INPUT"A1=";A1:AA=A1*PI/18Ø
64 INPUT"A2=";A2:IF A2< A1 GOTO 72
66 AE=A2*PI/18Ø
7Ø FOR ALFA=AA TO AE STEP PI/1ØØ
```

Das so veränderte Programm erfüllt die gestellten Bedingungen. Gibt man ein:A1=Ø und A2=36Ø, so wird natürlich wieder ein Kreis gezeichnet.

6.2.7. Entwickeln Sie ein Programm zum Zeichnen von Ellipsenbögen!

Grafische Effekte durch Escape-Funktionen

Mit den folgenden beiden Programmen ergeben sich interessante grafische Effekte.

a)
```
1Ø ! Escape bei LINE
2Ø WINDOW Ø,31,Ø,39:CLS
3Ø FOR Y=-1ØØ TO 1ØØ STEP 1Ø
4Ø LINE Ø,Y,319,-Y,7
5Ø NEXT
```

b)
```
1Ø ! Escape bei CIRCLE
2Ø WINDOW Ø,31,Ø,39:CLS
3Ø FOR K=Ø TO 32Ø STEP 1Ø
4Ø CIRCLE K,-K,-K,7
5Ø NEXT
```

Die Maschinenprogramme für die Anweisungen LINE und CIRCLE enthalten natürlich auch Fluchtfunktionen. Allerdings andere, als in den Beispielprogrammen verwendet wurden. Doch auch sie sorgen dafür, daß es nicht zum Programmabsturz kommt, wenn X und Y außerhalb des Pixelbereiches liegen. Hat X den Bereich verlassen, so wird nicht gezeichnet. Doch wenn Y< 0 oder Y> 255 ist, so werden die Funktionen gespiegelt. Das zeigt, daß auch in den Fluchtfunktionen Möglichkeiten stecken, die man in Grafikprogrammen wirkungsvoll einsetzen kann.

6.3. Tonausgabe

Der KC 85/3 bzw. KC 85/2 besitzt einen Tongenerator. Damit ist die Erzeugung von Rechteckschwingungen unterschiedlicher Frequenz und Amplitude möglich. Mit zwei Kanälen lassen sich Töne von ca. 35 Hz bis ca. 6 KHz erzeugen. Die Tonsignale stehen zweikanalig aber stets mit maximaler Lautstärke an der am Grundgerät eingebauten Diodenbuchse zur Verfügung. Nach Mischung der Signale im Videointerface und einer 32stufigen Amplitudeneinstellung können die Töne mit definierter Lautstärke am RGB-Ausgang entnommen werden.

Soll ein bestimmter Ton erzeugt werden, muß dem Computer folgendes mitgeteilt werden:

auf welchem Kanal,
wie lange,
in welcher Tonhöhe und
in welcher Lautstärke

die Tonausgabe erfolgen soll.

In BASIC erfolgt diese Mitteilung an den Computer über die SOUND-Anweisung in der Form:

```
SOUND Z1, V1, Z2, V2, LS, TD
```

Dabei legen die Parameter Z1 und V1 die Tonhöhe des Kanals 1 und die Parameter Z2 und V2 die Tonhöhe des Kanals 2 fest. V1 und V2 können nur die Werte Ø und 1 annehmen. Schreibt man für Z1 und Z2 allgemein Z, so gilt hierfür

$Ø < Z < 256$ (ganze Zahlen).

Mit den Parametern LS und TD werden Lautstärke und Tondauer bestimmt. Es gelten folgende Grenzen:

$Ø < LS < 32$
$Ø < TD < 256$.

Hierfür sind folgende Sonderfälle zu beachten:

- ist LS=Ø, erfolgt keine Tonausgabe über Fernsehgerät, jedoch weiterhin über die Diodenbuchse;
- ist Z=Ø, wird kein Ton ausgegeben;
- werden LS und TD nicht angegeben, bleiben die vorherigen Werte erhalten.

Beispiel:

Mit dem folgenden kleinen Programm kann z. B. ein Ton mit auf- und absteigender Frequenz erzeugt werden.

```
1Ø ! TON MIT AUF- UND ABSTEIGENDER FREQUENZ
2Ø FOR I = 255 TO 2Ø STEP-1
3Ø SOUND I, Ø, I, Ø, 31, 1
4Ø NEXT
5Ø FOR I = 2Ø TO 255
6Ø SOUND I, Ø, I, Ø, 31, 1
7Ø NEXT
8Ø GOTO 2Ø
```

Natürlich können auch Musikstücke vom Computer gespielt werden, wenn die richtigen Tonhöhen und die den Tonwerten entsprechenden Tondauern programmiert werden. Der Tonumfang des Computers beinhaltet 7 Oktaven. Diese sind mit den dazugehörigen Werten für Z und V im Bild dargestellt.

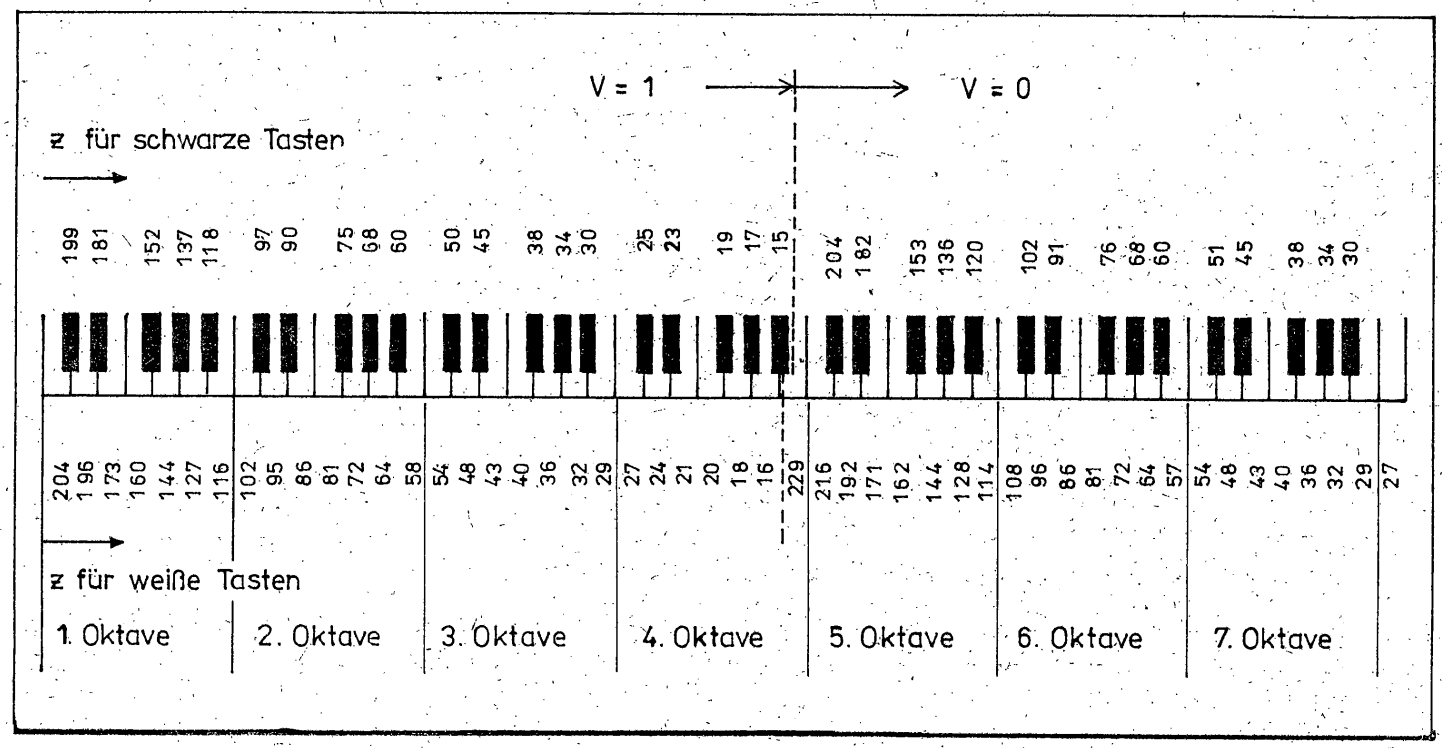


Bild 13: Richtwerte für den Parameter Z, um Frequenzen zu erreichen, die etwa den musikalischen Tonhöhen entsprechen.

6.3.1. Laden Sie das Programm MUSIK aus der Programmsammlung und arbeiten Sie die dazugehörige Aufgabenstellung ab!

Das folgende kleine BASIC-Programm veranlaßt den Computer, z. B. ein bekanntes Musikstück zu spielen. Über den Wert der Variablen D, der in Zeile 9Ø zugewiesen wird, kann das Tempo bestimmt werden.

Beispiel: ODE AN DIE FREUDE

```
 1Ø DATA 86,2
 2Ø DATA 86,2, 81, 2, 72, 2, 72, 2, 81, 2
 3Ø DATA 86, 2, 96, 2, 1Ø8, 2, 1Ø8, 2, 96, 2, 86, 2
 4Ø DATA 86, 3, 96, 1, 96, 4
 5Ø DATA 96, 3, 1Ø8, 1, 1Ø8, 3
 6Ø DATA 96, 2, 96, 2, 86, 2, 1Ø8, 2, 96, 2, 86, 1
 7Ø DATA 81, 1, 86, 2, 1Ø8, 2, 96, 2, 86, 1, 81, 1
 8Ø DATA 86, 2, 96, 2, 1Ø8, 2, 96, 2, 144, 2, 86, 4
 9Ø D = 1Ø
1ØØ RESTORE : FOR X = 1 TO 15: READ T, N : SOUND T, Ø, T, Ø, 31, D*N : NEXT
11Ø RESTORE : FOR X = 1 TO 12 : READ T, N : SOUND T, Ø, T, Ø, 31, D*N : NEXT
12Ø RESTORE 5Ø : FOR X = 1 TO 3 : READ T, N : SOUND T, Ø, T, Ø, 31, D*N : NEXT
13Ø SOUND Ø, Ø, Ø, Ø, 31, D*1
14Ø GOTO 19Ø
15Ø RESTORE 6Ø : FOR X = 1 TO 18 : READ T, N : SOUND T, Ø, T, Ø, 31, D*N : NEXT
16Ø RESTORE 2Ø : FOR X = 1 TO 11 : READ T, N : SOUND T, Ø, T, Ø, 31, D*N : NEXT
17Ø RESTORE 5Ø : FOR X = 1 TO 3 : READ  T, N : SOUND T, Ø, T, Ø, 31, D*N : NEXT
18Ø RETURN
19Ø GOSUB 15Ø : GOSUB 15Ø
2ØØ END
```

Neben der SOUND-Anweisung gibt es zur Tonausgabe noch die BEEP-Anweisung der Form:

BEEP N

Mit Hilfe dieser Anweisung werden akustische Signale mit festgelegter Tonhöhe und festgelegter Tonlänge erzeugt. Der Parameter N bestimmt die Anzahl der Signale, die nacheinander ausgegeben werden. Für N gelten die Grenzen:

Ø < N < 256 (ganze Zahlen).

Wird kein Parameterwert angegeben, so gilt N=1. Die BEEP-Anweisung läßt sich z. B. sehr gut als akustische Kennzeichnung einer Eingabeanforderung im Programm verwenden.

6.3.2. Verwenden Sie die Anweisung BEEP in einem von Ihnen ausgewähltem Programm zur Aufforderung einer Eingabe!

6.4. Anweisungen zur Programmierung von einfachen Bewegungen

Mit dem KC 85/3 bzw. KC 85/2 ist auch die Programmierung von einfachen Bewegungen möglich. Dazu ist es erforderlich, daß einzelne Bewegungsphasen ähnlich wie bei einem Zeichentrickfilm in zeitlich kurzen Abständen auf dem Bildschirm dargestellt werden. Für einfache Bewegungen reichen zwei bis drei Bewegungsphasen aus. Für kontinuierlichere Bewegungsphasen (z. B. laufende Figuren) sollten jedoch 1Ø bis 15 Bewegungsphasen entwickelt werden.

Die zu bewegenden Figuren müssen mit Hilfe der Grafik des Computers aus einzelnen Zeichen zusammengesetzt werden.

Die Grafik des KC 85/3 bzw. KC 85/2 ermöglicht, beliebige Sonderzeichen in einer 8 x 8 - Pixel - Matrix neu zu generieren, die dann in einer Zeichenbildtabelle abgelegt werden müssen. Der Computer nutzt nach dem Einschalten seine internen Zeichenbildtabellen z. B. für Großbuchstaben, Kleinbuchstaben, Sonderzeichen usw. Soll der Computer jedoch mit der neu generierten Zeichenbildtabelle arbeiten, muß in eine der Systemarbeitszellen (CCTL 1 bis CCTL 3) die entsprechende Anfangsadresse als Tabellenzeiger eingetragen werden.

Beispiel:

Mit dem nachfolgenden Programm "VOGEL" wird die Realisierung einer einfachen Bewegung auf dem Bildschirm gezeigt.

1. Erzeugen der Sonderzeichen für die Bewegungsphasen

Pixel - Matrix (8*8)	Hexa-Code	Dezimal-Code
128 64 32 16 8 4 2 1		
▫▫▫▫▫▫▫▫	ØØ	Ø
▫▫▫▫▫▫▫▫	ØØ	Ø
▫▫▪▫▫▪▫▫	24	36
▫▪▫▪▪▫▪▫	5A	9Ø
▪▫▫▪▪▫▫▪	99	153
▫▫▫▪▪▫▫▫	18	24
▫▫▫▫▫▫▫▫	ØØ	Ø
▫▫▫▫▫▫▫▫	ØØ	Ø

(Berechnung des Dezimal-Codes am Beispiel der 4. Zeile:
Ø*128+1*64+Ø*32+1*16+1*8+Ø*4+1*2+Ø*1=9Ø)

▫▫▫▫▫▫▫▫	ØØ	Ø
▫▫▫▫▫▫▫▫	ØØ	Ø
▪▪▪▫▫▪▪▪	E7	231
▫▫▫▪▪▫▫▫	18	24
▫▫▫▪▪▫▫▫	18	24
▫▫▫▪▪▫▫▫	18	24
▫▫▫▫▫▫▫▫	ØØ	Ø
▫▫▫▫▫▫▫▫	ØØ	Ø

▪▫▫▫▫▫▫▪	81	129
▫▪▫▫▫▫▪▫	42	66
▫▫▪▫▫▪▫▫	24	36
▫▫▫▪▪▫▫▫	18	24
▫▫▫▪▪▫▫▫	18	24
▫▫▫▪▪▫▫▫	18	24
▫▫▫▫▫▫▫▫	ØØ	Ø
▫▫▫▫▫▫▫▫	ØØ	Ø

2. BASIC-Programm

```
 1Ø CLS
 2Ø A=VPEEK(15626)
 3Ø IF A=36 THEN 9Ø  :ELSE 4Ø
 4Ø AD=15624
 5Ø FOR I=Ø TO 23
 6Ø READ P
 7Ø VPOKE AD+I, P
 8Ø NEXT I
 9Ø VPOKE 14249, 188
1ØØ FOR B=3Ø TO 1 STEP-1
11Ø PRINT AT (B,B); CHR$(97):GOSUB 21Ø
12Ø PRINT AT (B,B); CHR$(98):GOSUB 21Ø
13Ø PRINT AT (B,B); CHR$(99):GOSUB 21Ø
14Ø PRINT AT (B,B); " "
15Ø NEXT B
16Ø VPOKE 14249, 254
17Ø DATA Ø, Ø, 36, 9Ø, 153, 24, Ø, Ø
18Ø DATA Ø, Ø, 231, 24, 24, 24, Ø, Ø
19Ø DATA 129, 66, 36, 24, 24, 24, Ø, Ø
2ØØ END
21Ø FOR K=Ø TO 2Ø:NEXT K:RETURN
```

In den Zeilen 17Ø bis 19Ø befinden sich die Daten für die drei Sonderzeichen. Die Zeilen 4Ø bis 8Ø bewirken das Abspeichern der Daten ab Adresse $BDØ8_H$ (Hexadezimal). Da die Anweisung VPOKE angewendet wird, muß von $BDØ8_H$ $8ØØØ_H$ subtrahiert werden ($3DØ8_H$=15624). In Zeile 9Ø wird der Zeiger der Arbeitszelle CCTL1 von FE_H=254 auf BC_H=188 geändert. Dadurch liegen die drei Sonderzeichen für die Bewegungsphasen auf den Kleinbuchstaben a, b und c (CHR$(97) usw.). In Zeile 16Ø wird der Normalfall wieder hergestellt. Das ist wichtig, um die normale Tastenbelegung für die Arbeit mit anderen Programmen wieder zu sichern.

Mit dem Unterprogramm in Zeile 21Ø kann die Pause zwischen den einzelnen Bewegungsphasen beeinflußt werden. In Zeile 3Ø wird geprüft, ob die Daten für die Sonderzeichen eingelesen wurden. Wenn ja, wird in Zeile 9Ø fortgesetzt.

Erläuterung der im Beispiel verwendeten Anweisungen:

VPEEK (I)

Mit dieser Anweisung kann der Inhalt der Adresse I im Bildwiederholspeicher gelesen werden. Das Ergebnis wird als ein Byte dezimal ausgegeben.

z. B. PRINT VPEEK(Ø) oder PRINT VPEEK(14249)

So wird der Inhalt der Adressen $8ØØØ_H$=32768 und $B7A9_H$=47Ø17. Von beiden Adressen muß 32768 subtrahiert werden, um auf I zu kommen.

VPOKE I,K

Durch diese Anweisung können die Adressen I im Bildwiederholspeicher mit einem Byte beschrieben werden (Ø<=K<=255).

Beispiel: VPOKE 15626,36

Die Adresse $BDØA_H$ wurde mit 24_H belegt.

Weitere Anweisungen zur Arbeit mit dem Speicher:

PEEK (I)

Die Anweisung ermöglicht ein Lesen des Inhaltes der Adressen von $ØØØØ_H$ bis $7FFF_H$.

Beispiel: PRINT PEEK(5Ø9)

Es wird der Inhalt der Adresse $Ø1FD_H$ gelesen.

POKE I,K

Durch diese Anweisung können die Adressen von $ØØØØ_H$ bis $7FFF_H$ mit einem Byte beschrieben werden (Ø<=K<=255).

Beispiel: POKE 8Ø,255

DEEK(I)

Mit dieser Anweisung kann die Adresse I und I+1 im Bereich $ØØØØ_H$ bis $7FFF_H$ gelesen werden.

Beispiel: PRINT DEEK(5Ø2)

DOKE I,K

Diese Anweisung ermöglicht ein Beschreiben der Adressen I und I+1 mit je einem Byte (-32768<=K<=32767).

Beispiel: DOKE Ø, 14249

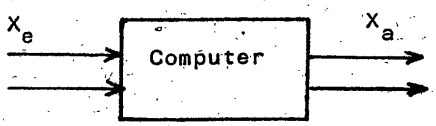
Es wurde die Adresse $37A9_H$ abgespeichert, wobei der niederwertige Teil (A9) auf I und der höherwertige Teil (37) auf I+1 liegt.

7. Prozeßautomatisierung mit dem Computer

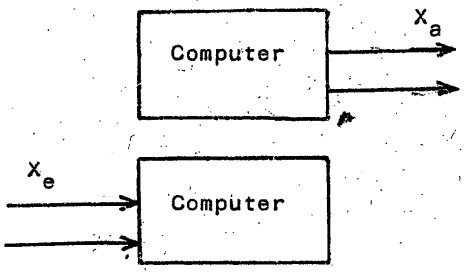
7.1. Der Computer als flexible Steuereinrichtung

In diesem Abschnitt wird dargestellt, wie der Computer zur Steuerung von Prozessen genutzt werden kann. Er übernimmt dabei die Funktion der Steuereinrichtung (vgl. auch Abschnitt 5 Lb GdA). Der Computer kann prinzipiell alle Steueraufgaben erfüllen. In der Praxis kann es lediglich zu Einschränkungen kommen, wenn der vorhandene Speicherplatz nicht ausreicht oder die Operationsgeschwindigkeit des Computers nicht hoch genug ist. Im letzteren Fall muß es sich aber schon um die Beherrschung sehr schneller Vorgänge handeln, denn der Computer reagiert in Bruchteilen von tausendstel Sekunden. Das wird allerdings mit der Programmiersprache BASIC nicht erreicht. Für Echtzeitsteuerungen (z. B. Regelung der Beleuchtungsstärke) reagiert der Computer zu langsam. Es wären dann Programme in Maschinensprache oder in einer speziellen Sprache für automatische Steuerungen erforderlich. Da der Computer für die Steuerung unterschiedlicher Prozesse - nur unter Nutzung verschiedener Programme - eingesetzt wird, kann er als flexible Steuereinrichtung betrachtet werden.

Dabei gilt für den Informationsaustausch mit dem zu steuernden Prozeß folgendes allgemeine Schema (wie für jede Steuereinrichtung):



Bei manchen Aufgaben vereinfacht sich der Zusammenhang:



nur Ausgabe
(z. B. Zeitplansteuerungen)

nur Eingabe, Verarbeitung und Anzeige
(z. B. Messung, Kontrolle)

7.2. Interface

Zur Steuerung von Prozessen mit dem Experimentiersatz durch den KC 85/3 bzw. KC 85/2 sind zwei Interface-Bausteine erforderlich (Modul M ØØ1, Koppelbaustein).

Modul M ØØ1 für den KC 85/3 bzw. KC 85/2

Dieser Modul kann in einem freien Steckplatz des Rechners untergebracht werden. Er ermöglicht über einen Stecker den Informationsaustausch mit dem zu steuernden Prozeß. Er enthält einen Zähler-Zeitgeber-Schaltkreis (CTC) und einen Parallel-Input-Output-Schaltkreis (PIO). Beide Schaltkreise sind sehr variabel einsetzbar. Um ein bestimmtes Betriebsverhalten zu erzeugen, sind entsprechende Steuerwörter in Register dieser Schaltkreise zu laden. Wie das praktisch realisiert werden kann, wird in diesem Abschnitt am Beispiel ausgewählter Betriebsweisen des PIO-Schaltkreises dargestellt.

Der CTC-Schaltkreis besitzt 4 Kanäle, die sowohl einzeln als auch kombiniert genutzt werden können. Ist der CTC-Schaltkreis als Zähler programmiert, so zählt und speichert er die Anzahl von Impulsen, die vom Prozeß an den Eingang TRG (Trigger) gelangen. Entsprechend der Programmierung können nach Erreichen einer bestimmten Impulszahl Programmroutinen ausgelöst werden. Es kann auch der Zählerstand durch die CPU abgerufen und im Programm mit verarbeitet werden. In der Betriebsart Zeitgeber gibt der Schaltkreis in (durch Programmierung festgelegten) Zeitabständen Impulse ab. Diese Impulse können für den Aufruf einer Programmroutine genutzt werden oder über den Ausgang ZC/T an den Prozeß abgegeben werden.

Der PIO-Schaltkreis hat folgende Struktur (Übergabesignale sind weggelassen):

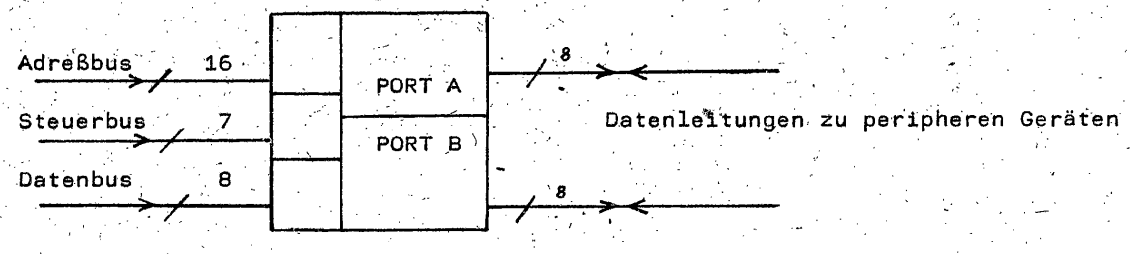


Bild 14: PIO - Schaltkreis

Es können über den Datenbus Informationen zum PIO-Schaltkreis gebracht werden. Die entsprechenden Signale werden über Port A oder Port B an die Peripherie ausgegeben. Die Signalpegel liegen hier solange an, bis neue Daten ausgegeben werden.

Gleichfalls können Informationen, die von der Peripherie kommen, über Port A oder Port B auf den Datenbus gelangen und stehen so dem Computer zur Verarbeitung zur Verfügung. Die Steuerung dieser Vorgänge erfolgt mittels Adreß- und Steuerbus. Die Betriebsart eines jeden Ports wird festgelegt, indem ein oder mehrere Steuerworte in Register des PIO-Schaltkreises geschrieben werden. Diese Register sind beim KC 85/3 bzw. KC 85/2 unter folgenden Adressen erreichbar:

Adresse für Steuerworte	
Port A	6
Port B	7

Wir nutzen bei den folgenden Experimenten die Betriebsarten

- Bit - Ein - Ausgabe
- Byte - Eingabe

Die Betriebsart Bit- Ein- Ausgabe kann gewählt werden, wenn jedes Bit einzeln ausgewertet werden soll. Z. B. könnte durch ein Bit signalisiert werden, ob ein Temperaturgrenzwert überschritten ist oder nicht, ob eine Lichtschranke unterbrochen ist oder nicht usw. Auch bei der Ausgabe hat jedes Bit seine eigene Bedeutung. Z. B. könnte durch 1 Bit ein Motor ein- und ausgeschaltet werden.

Das Steuerwort für diese Betriebsart ist CF (hexadezimal) oder 2Ø7 (dezimal, wie es in BASIC genutzt wird). Durch ein zweites Steuerwort wird festgelegt, welche der Bit als Eingänge und welche als Ausgänge wirken sollen.

Dabei gilt folgende Zuordnung:

Ø - Ausgang
1 - Eingang

Sollen beispielsweise BØ ... B5 Eingabebit und B6 und B7 Ausgabebit sein, so ergibt sich das Steuerwort

B7	B6	B5	B4	B3	B2	B1	BØ
Ø	Ø	1	1	1	1	1	1

Diese Dualzahl ist dezimal verschlüsselt einzugeben. So wird das Steuerwort zu 63.

Für die Ausgabe eines Bytes über einen Kanal wird in BASIC die Anweisung

```
OUT Kanaladresse, Byte
```

verwendet.

Für das Einlesen über einen Kanal gibt es die Anweisung

```
INP (Kanaladresse).
```

Es ergibt sich folgendes Programmstück zur PIO-Programmierung (Port B für den KC 85/3 bzw. KC 85/2 entsprechend vorstehender Forderungen):

```
1Ø OUT 7, 2Ø7
2Ø OUT 7, 63
```

Die PIO-Programmierung wird meist zu Beginn eines Programms oder Programmabschnitts einmalig vorgenommen. Bei Steuerungen werden dann oft in schneller Folge Daten über den PIO-Schaltkreis ausgetauscht. Dafür gelten die Adressen 4 (Port A) und 5 (Port B).

Wir setzen obiges Beispiel fort. Nach der PIO-Programmierung sollen Daten von Port B eingelesen werden.

```
3Ø X = INP (5)
4Ø X = X AND 63
```

In Zeile 3Ø werden alle 8 Bit von Port B gelesen. So wird also auch eine 1 von B6 und B7 gelesen, die in einem vorhergehenden Programmschritt ausgegeben wurde. Deshalb wird in Zeile 4Ø eine Maske gesetzt:

Ø Ø 1 1 1 1 1 1

Jedes eingelesene Bit wird mit dem entsprechenden Bit der Maske durch die logische Funktion UND verknüpft. Somit werden die gelesenen B6 und B7 zu Null, BØ ... B5 bleiben erhalten. Die dezimale Verschlüsselung der Maske ist 63.

Für die Byte-Eingabe gilt das Steuerwort 4F (hexadezimal) bzw. 79 (dezimal). Bei der Byte-Eingabe wirken 8 Anschlüsse des Kanals gemeinsam als Eingang (vgl. auch 7.4.).

Zusammenstellung einiger Steuerwörter und Adressen:

Bit-E/A CF_H bzw. $2Ø7_D$
Defin. E/A Eingang:1; Ausgang:Ø
Byte-Eingabe $4F_H$ bzw. 79_D
PIO-Adressen
Steuerregister für Port A: 6
Steuerregister für Port B: 7
Datenregister für Port A: 4
Datenregister für Port B: 5

7.2.1. Port A eines PIO-Schaltkreises soll zum Einlesen von 8 einzelnen Bit, Port B zur Ausgabe von 8 einzelnen Bit genutzt werden.

a) Welche Steuerwörter sind auf welche Adressen auszugeben?
OUT 6, 2Ø7 : OUT 6, 256
OUT 7, 2Ø7 : OUT 7, Ø

b) Schreiben Sie ein Programmstück zur PIO-Programmierung.

c) Ergänzen Sie das Programm so, daß ein Datenwort eingelesen und auf dem Bildschirm ausgegeben werden kann.

Koppelbaustein Rechner-Grundgerät

Es ist jetzt noch notwendig, die Signale zwischen Ausgang des PIO-Schaltkreises und Grundgerät zu übertragen. Dazu ist weiterhin zu beachten, daß von dem PIO-Schaltkreis Signale mit dem Pegel 5V abgegeben werden, das Experimentiergerät aber nur Signale von 7 ... 12V als 1-Signal auswertet. Umgekehrt ist der Pegel von 12V des Experimentiergerätes auf die für den Rechner notwendigen 5V herabzusetzen.

Für die Verwendung im Fach "Grundlagen der Automatisierung" wird ein Koppelbaustein geliefert, der neben dieser Pegelumsetzung auch den Schutz des PIO-Schaltkreises bei Fehlbedienung sichert. Für die Experimente in der Berufsausbildung genügt es, daß dieser Koppelbaustein nur einige Anschlüsse des PIO-Schaltkreises mit dem Experimentiergerät verbindet.

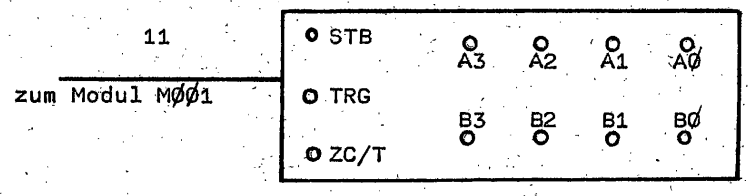


Bild 15: Koppelbaustein zum Experimentiergerät

Es werden weiterhin einige Einschränkungen vorgenommen. Die Anschlüsse AØ ... A3 sind 4 Bit, die als Eingang genutzt werden können. Sie sind mit Bit Ø ... Bit 3 von Port A verbunden. Die Anschlüsse BØ ... B3 sind Ausgänge. Sie sind mit Bit Ø ... Bit 3 von Port B verbunden. Gemeinsam mit dem Koppelbaustein muß deshalb immer Port A zur Eingabe, Port B zur Ausgabe genutzt werden.

Der Zweck der drei weiteren Anschlüsse wurde im Zusammenhang mit dem CTC-Schaltkreis genannt bzw. wird beim Byte-Einlesen noch erläutert.

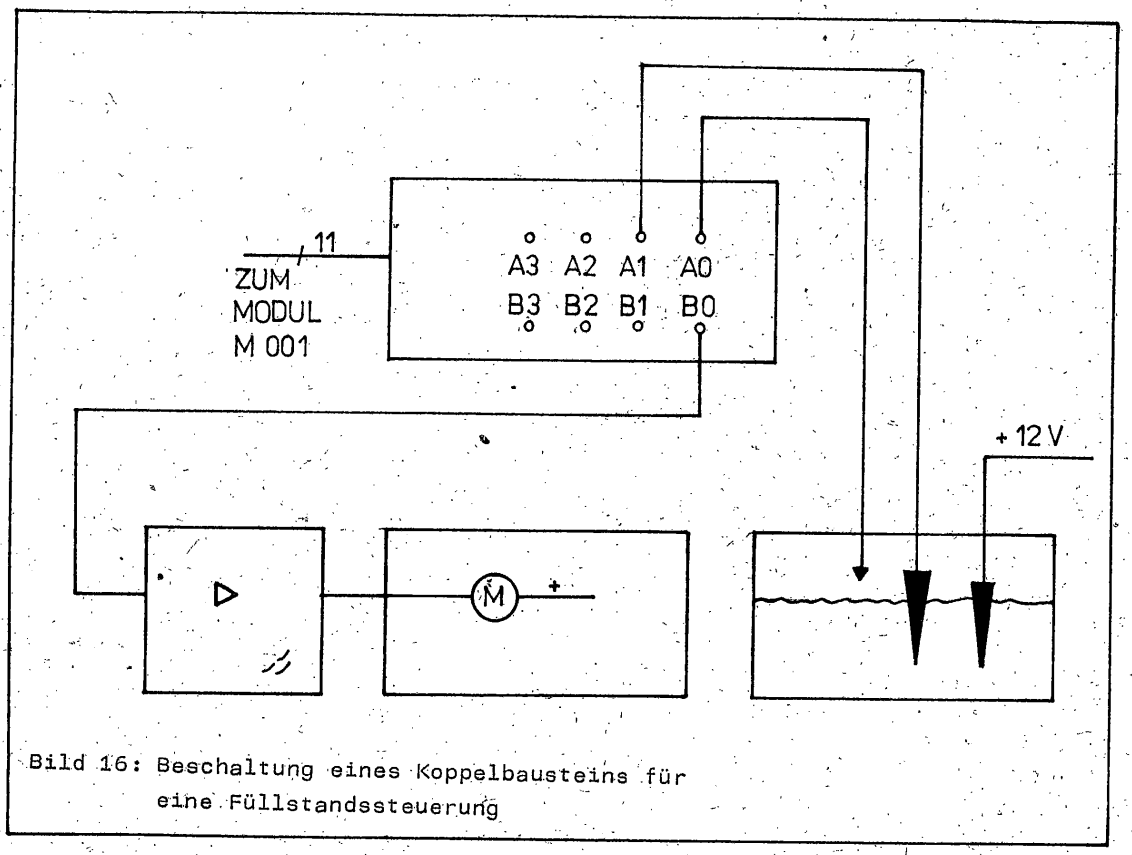


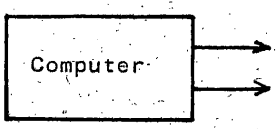
Bild 16: Beschaltung eines Koppelbausteins für eine Füllstandssteuerung

7.3. Steuerungen mit binären Signalen

Zeitplansteuerungen

Wir wenden uns zunächst dem Sonderfall zu, daß der Rechner in Abhängigkeit von der Zeit Signale zur Steuerung ausgibt (Zeitplansteuerung).

Es sind also nur Ausgänge zu berücksichtigen, Typ:



Beispiel: Ampelsteuerung

Die Lampen der Ampel seien folgendermaßen an Port B angeschlossen:

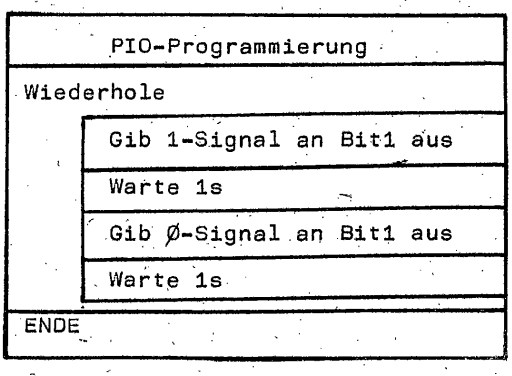
BØ : Rot, B1 : Gelb, B2 : Grün

a) Es ist ständiges Blinken von Gelb zu programmieren (nach je 1s umschalten)

b) Es ist folgende Schaltfolge zu realisieren:

Rot (9s), Rot-Gelb (2s), Gelb (2s), Grün (7s), Grün-Gelb (2s); ständige Wiederholung

Zu a)




```
1Ø OUT 7, 2Ø7 : OUT 7, Ø
2Ø OUT 5, 2
3Ø PAUSE 1Ø
4Ø OUT 5, Ø
5Ø PAUSE 1Ø
6Ø GOTO 2Ø
7Ø END
```

In Zeile 1Ø wird Port B für Bit E/A programmiert. Alle Anschlüsse werden als Ausgang festgelegt.

7.3.1. Realisieren Sie das Blinklicht im Experiment! Erhöhen Sie die Frequenz durch Programmänderung!

Zu b)

Bei dieser Zeitplansteuerung sind mehrere Bit gleichzeitig zu berücksichtigen. Es empfiehlt sich deshalb, zunächst ein Schaltfolgediagramm aufzustellen und die jeweilige dezimale Verschlüsselung des Bitmusters an Port B zu ermitteln.

Schaltfolgediagramm

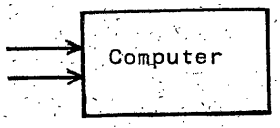
	B2 2^2 Grün	B1 2^1 Gelb	BØ $2^Ø$ Rot	Verschlüsselung
1. Schaltschritt Pause 9s	Ø	Ø	1	1
2. Schaltschritt Pause 2s	Ø	1	1	3
3. Schaltschritt Pause 2s	Ø	1	Ø	2
4. Schaltschritt Pause 7s	1	Ø	Ø	4
5. Schaltschritt Pause 2s	1	1	Ø	6

6. = 1. Schaltschritt

7.3.2. Führen Sie die Programmierung selbständig aus, und erproben Sie die Lösung im Experiment!

Zustandskontrollen durch Computer

Im folgenden Beispiel nimmt der Computer über den PIO-Schaltkreis nur Informationen aus dem Prozeß auf und verarbeitet sie. Es handelt sich also um den folgenden Anwendungstyp:



Hierzu gehören Messungen und Funktionskontrollen.

Beispiel:

Es soll der Zustand von 2 Lampen geprüft und auf dem Bildschirm mittels Schrift dargestellt werden. Die Prüfung und Textausgabe erfolge nach jeweils 1s.

Zum Einlesen wird Port A verwendet und alle Bit als Eingang programmiert. Es wird folgende Anordnung gewählt:

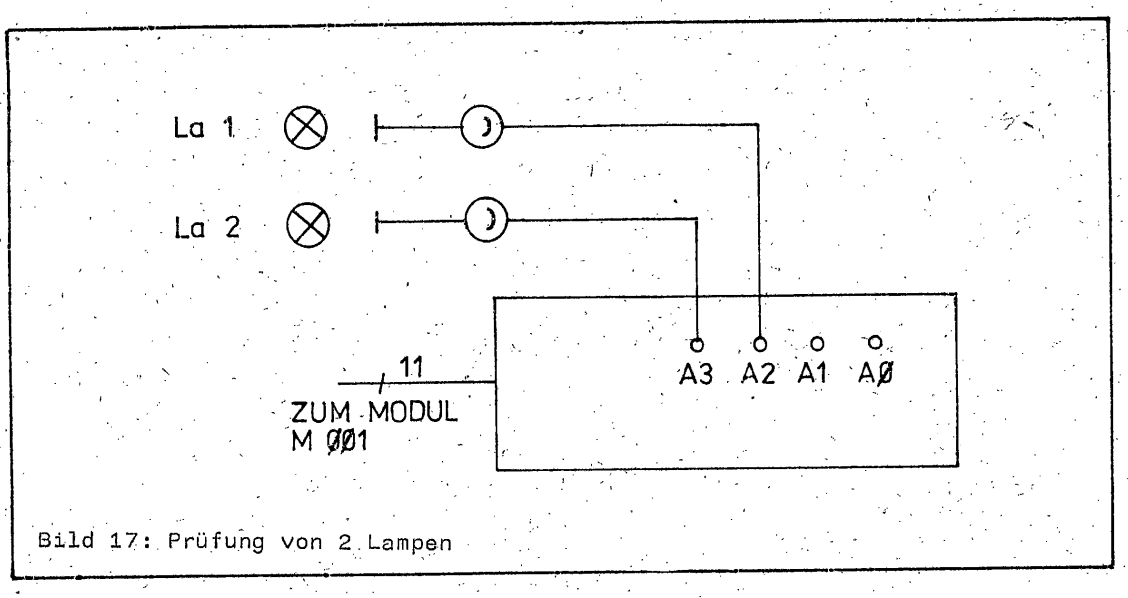
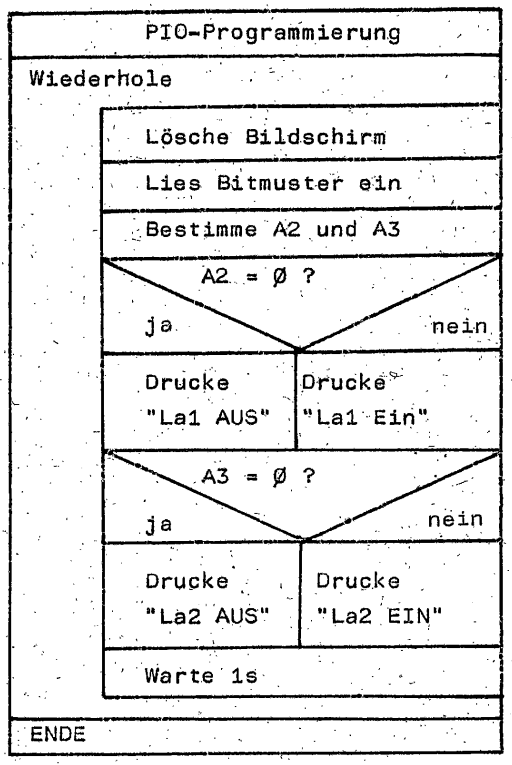


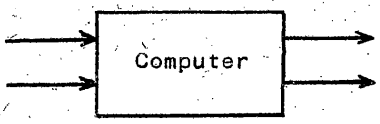
Bild 17: Prüfung von 2 Lampen




```
1Ø OUT 6, 2Ø7 : OUT 6, 15
2Ø CLS
3Ø A = INP (4)
4Ø A2 = A AND 4
5Ø A3 = A AND 8
6Ø IF A2 = Ø THEN PRINT
"Lampe 1 AUS": ELSE PRINT
"Lampe 1 EIN"
7Ø IF A3 = Ø THEN PRINT
"Lampe 2 AUS" : ELSE PRINT
"Lampe 2 EIN"
8Ø Pause 1Ø
9Ø GOTO 2Ø
1ØØ END
```

Computer zur Prozeßsteuerung mit Ein- und Ausgaben

Wir wenden uns jetzt dem Fall zu, daß der Computer Informationen vom Prozeß aufnimmt und Informationen an den Prozeß abgibt:



Als Beispiel wird eine Steuerung für eine Fußgängerampel gewählt. Für die Ausgabe soll gelten (an Port B):

BØ = 1 : Fahrbahn, Rot
B1 = 1 : Fahrbahn, Gelb
B2 = 1 : Fahrbahn, Grün
B3 = 1 : Fußgänger, Rot
B3 = Ø : Fußgänger, Grün

Für die Eingabe soll gelten: Signalgeber an AØ (Port A).

Die Ampel soll für Fußgänger ständig Rot zeigen und für die Fahrbahn Grün, nach Betätigung des Signalgebers für 3s Gelb zuschalten, 2s danach auf Rot für Fahrzeuge umschalten. Nach weiteren 2s erhalten die Fußgänger für 5s Grün. Weitere 5s danach erhalten die Fahrzeuge für 3s die Gelbzuschaltung und dann wieder Grün. Es soll gesichert sein, daß die Fahrzeuge mindestens 1Øs Grün erhalten, bevor ein Fußgänger die Ampel wieder umschalten kann.

Lösung:

Es werden alle Bit von Port A als Eingang, alle Bit von Port B als Ausgang definiert.

PIO-Programmierung
Gib an B2 und B3 1-Signal aus
Warte 1Øs
Wiederhole
Lies Bitmuster an Port A ein; A =
bis A > Ø
Programm nach Zeitplan

```
1Ø OUT 6, 2Ø7 : OUT 6, 15
2Ø OUT 7, 2Ø7 : OUT 7, Ø
3Ø OUT 5, 12
4Ø PAUSE 1ØØ
5Ø A = INP (4) AND 1
6Ø IF A = Ø THEN 5Ø
7Ø
 .
 .
```

7.3.3. Ergänzen Sie das Programm Fußgängerampel, und realisieren Sie die Steuerung im Experiment!

Das vorliegende Programm hat noch folgenden praktischen Mangel:

Betätigt der Fußgänger kurzzeitig den Signalgeber während der Pause von 1Øs (Zeile 4Ø), so wird der Tastendruck ignoriert. Man könnte das Programm ändern, so daß auch in der Pause die Signalgeberbetätigung ständig abgefragt wird.

```
35 B = Ø
4Ø FOR I = 1 TO 1ØØ
45 PAUSE 1 : A = INP (4) AND 1
47 IF A = 1 THEN B = 1
48 NEXT I
49 IF B = 1 THEN 7Ø
5Ø A = INP (4) AND 1
6Ø IF A = Ø THEN 5Ø
```

Dieses Programm könnte durch Beachtung weiterer Bedingungen ergänzt werden. Es kann z. B. durch Abfrage einer Lichtschranke ermittelt werden, ob sich Fahrzeuge im Stauraum befinden. In diesem Fall könnte erst nach 5s die Umschaltung erfolgen.

7.3.4. Es ist die Beleuchtungsstärke zu regeln. Ein lichtelektrischer Geber wird von einer Lampe beleuchtet. Das Ausgangssignal des Gebers wird vom Rechner eingelesen. Bei Beleuchtung (hohes Potential, 1-Signal) soll die Lampe über den Rechner ausgeschaltet, im anderen Fall eingeschaltet werden.

Bemerkung:

Durch schnelles Ein- und Ausschalten stellt sich eine mittlere Helligkeit ein. Wird in BASIC programmiert, so ergibt sich ein Flackern der Lampe. BASIC ist für diese Echtzeitanwendung zu langsam.

7.3.5. Es ist eine Füllstandssteuerung (vgl. Anordnung in 7.2.) zu programmieren. Der Motor wird eingeschaltet, wenn die Flüssigkeit die untere Sonde nicht berührt. Er wird ausgeschaltet, wenn die Flüssigkeit die obere Sonde erreicht hat.

Nutzung von Standardprogrammen für binäre Steuerungen

Sind zwei oder mehr Bit des eingelesenen Wortes getrennt durch das Programm auszuwerten, so sind zunächst erst einmal die einzelnen Bit jedes für sich darzustellen. Dafür kann ein Standardprogramm genutzt werden, das auch gleichzeitig die PIO-Programmierung und das Einlesen realisiert. Es wird zum Einlesen Port A gewählt. Die einzelnen Bit können nun - wie bisher praktiziert - durch Setzen von Masken ermittelt werden. (Das Programm wird hier nur für die Bit ausgeführt, die auch am Koppelbaustein vorhanden sind.)

```
1Ø OUT 6, 2Ø7 : OUT 6, 15
2Ø A = INP (4)
3Ø AØ = A AND 1
4Ø A1 = (A AND 2)/2
5Ø A2 = (A AND 4)/4
6Ø A3 = (A AND 8)/8
7Ø END
```

Auch für die Ausgabe kann ein Standardprogramm eingesetzt werden. Es ist Port B zu programmieren, aus den einzelnen Bit das Ausgabewort zusammenzustellen, und es ist dann schließlich noch auszugeben.

Zunächst wird jedem Bit der Zahlenwert zugeordnet, den es repräsentiert, und dann werden diese Werte addiert. Das Ergebnis wird ausgegeben.

```
1ØØØ OUT 7, 2Ø7 : OUT 7, Ø
1Ø1Ø B = BØ + 2 * B1 + 4 * B2 + 8 * B3
1Ø2Ø OUT 5, B
```

Es werden also immer 4 Bit gemeinsam eingelesen, die zum Zwecke der Verarbeitung einzeln bekannt sein müssen. Nach der Verknüpfung zu einzelnen Ausgangsbit sind diese zum Zwecke der Ausgabe zu einem Bitmuster zusammenzufassen.

Beispiele für die Nutzung der Standardprogramme:

7.3.6. . Eine Lampe an B2 soll leuchten, wenn am Eingang folgende Belegung auftritt: AØ,A1,A3 = 1, A2 = Ø und alle anderen Bit beliebig.

. Weiterhin soll ein Summer ertönen bei der Eingangsbelegung AØ = Ø, A1 = 1, alle weiteren Bit beliebig. Entwerfen Sie das Programm!

7.3.7. . Ein Motor wird nach Zeitplan gesteuert. Die Steuerung wird unterbrochen und der Motor in den HALT-Zustand versetzt, solange mindestens 2 von 3 Lichtschranken unterbrochen sind (Sicherheitsschaltung). Entwerfen Sie die Experimentierschaltung und schreiben Sie das Programm!

Hinweis:

Man realisiere die Pausen in der Zeitplansteuerung durch Aufruf eines Unterprogramms, das gleichzeitig auch die Unterbrechnungsbedingung prüft und die Unterbrechung realisiert.

7.4. Experimente mit digitalen Signalen

Unter Nutzung der neuen Bausteine des Experimentiersatzes sind auch Experimente mit digitalen Signalen möglich. Es können beispielsweise Signale des Analog-Digital-Wandlers oder des Lochstreifenlesers genutzt werden. Zum Einlesen kann wieder Port A genutzt werden. Da die digitalen Signale im Experimentiersatz nur vier Informationsparameter enthalten, genügen die 4 Bit AØ ... A3.

Jetzt muß aber eine andere Form des Einlesens der Daten angewendet werden. Die Daten dürfen erst vom PIO-Schaltkreis übernommen werden, wenn an allen Bit der richtige Pegel anliegt. Deshalb wird beispielsweise beim Lochbandlesen noch ein zusätzliches Signal (Ladesignal) geliefert, wenn alle Löcher richtig positioniert sind. Dieses Signal (Pegel : ØV) gelangt an den Eingang STROBE von Port A und veranlaßt die Datenübernahme. Port A ist auf Byte-Eingabe zu programmieren. Schon mit einfachsten Hilfsmitteln kann die Funktion folgendermaßen demonstriert werden:

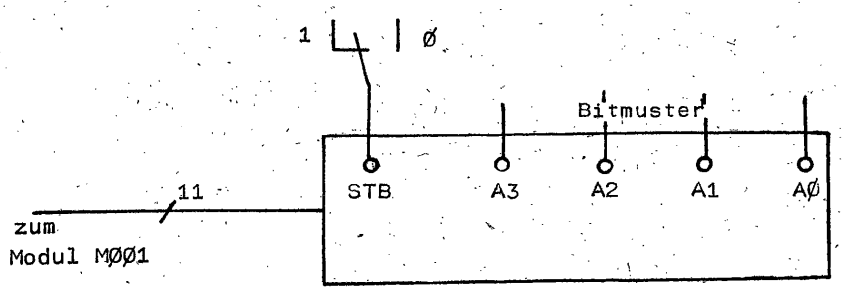


Bild 18: Schaltung zur Betriebsart Byte-Einlesen

An die Bit AØ ... A3 wird ein Bitmuster angelegt und das Einlesen programmiert (die restlichen Bit sind durch eine Maske auf Ø zu setzen).

STB wird auf 1-Signal gelegt. Jetzt kann am Bitmuster beliebig geändert werden, ohne daß der PIO-Schaltkreis Daten übernimmt. Erst wenn STB kurz Ø-Signal erhält, übernimmt er das aktuelle Bitmuster.

```
1Ø OUT 6, 79
2Ø CLS
3Ø X = INP (4) AND 15
4Ø PRINT X;
5Ø PAUSE 2Ø
6Ø GOTO 3Ø
```

7.4.1. Es ist das Ausgangspotential des Temperaturgebers (B.9.5.) in Abhängigkeit von der Zeit durch den Rechner aufzunehmen. Dazu ist zunächst das analoge Signal in ein digitales zu wandeln. Mit dem Start des Programms ist die Heizung am Baustein von Hand einzuschalten. Steigt die Temperatur nicht weiter an, wird die Heizung wieder ausgeschaltet.

a) Entwerfen Sie die Experimentieranordnung!

Hinweis:

- Beschalten Sie den Strobeeingang so, daß der PIO-Schaltkreis ständig die anliegende Information übernimmt.
- Empfehlenswert ist, die Pause im vorgegebenen Programm auf 8 Sekunden zu erhöhen.

b) Realisieren Sie die Anordnung, und führen Sie das Experiment durch!

c) Stellen Sie die auf dem Bildschirm ausgegebenen Werte auf Papier grafisch dar! Interpolieren Sie beim Zeichnen des Kurvenzuges.

8. Hinweise zur Arbeit mit komplexen Anwenderprogrammen

Beim Einsatz von Computern zur Lösung beruflicher Aufgaben werden verschiedene Arten von Software genutzt. Prinzipiell wird dabei zwischen Basissoftware, die auf den entsprechenden Computer orientiert ist, und Anwendersoftware, die auf ein konkretes Problem bezogen ist, unterschieden.

Die Basissoftware ist zur vielseitigen Nutzung anwenderunabhängiger gerätespezifischer Funktionen eines Computers erforderlich. Sie wird in der Regel vom Gerätehersteller entwickelt und bereitgestellt. Dazu gehören:

- Betriebssysteme zur Steuerung und Überwachung des Arbeitsablaufes im Computer und der Zusammenarbeit mit den peripheren Geräten.
- Programmiersprachen zur Formulierung von Programmen.
- Compiler oder Interpreter (Übersetzungsprogramme) zur Übersetzung der in einer Programmiersprache geschriebenen Programme in die interne Sprache des Computers.

- Kommunikationssoftware zur Kopplung mit anderen Computern und Terminals sowie zum Anschluß an Datennetze.

- Standardsoftware zur Unterstützung der Lösung typischer Aufgabenkomplexe der Informationsverarbeitung wie z. B. Textverarbeitungssysteme oder grafische Standardprogramme.

Anwendersoftware wird zur Lösung spezieller Probleme des Anwenders erarbeitet. Sie wird meist durch den Anwender selbst oder durch zentrale EDV-Projektierungseinrichtungen entwickelt.

Für den KC 85/3 bzw. KC 85/2 wurden komplexe Anwenderprogramme entwickelt, mit denen wesentliche Funktionen einiger Standardprogramme, die an Personalcomputern Anwendung finden, realisiert werden können.

Mit dem Programm UNIDATEI können Aufgaben der Datenerfassung, der Sortierung von Daten und der Suche von Daten gelöst werden. Das entspricht wesentlichen Funktionen, wie sie auch durch das relationale Datenbanksystem REDABAS realisiert werden. Damit können Daten erfaßt, gespeichert und ausgewertet werden.

Mit dem Programm UNIDATEI kann eine Datei nach eigenen Vorstellungen angelegt und gestaltet werden. Die Arbeit mit diesem Programm wird durch übersichtliche Menüs sehr erleichtert.

Nach dem Laden des Programms von Kassette kann zwischen dem Einlesen einer Datei vom Band und dem Einrichten einer neuen Datei gewählt werden. Damit ist bereits gesagt, daß eingegebene Daten auf Band gespeichert und wieder geladen werden können. Soll eine neue Datei eingerichtet werden, so ist zunächst der Name der Datei einzugeben:

z. B. Dateiname : Lit.-Verzeichnis (Abschluß mit ENTER)

Daran anschließend muß die Datei eingerichtet werden. Es ist die Anzahl der Felder pro Datensatz einzugeben. Das bedeutet: Ein Feld entspricht einem bestimmten Merkmal des einzugebenden Datensatzes (vergleichbar mit Merkmalen einer Karteikarte. Satz = Karteikarte; Feld = Merkmal). Bei der Einrichtung einer Datei für ein Literaturverzeichnis könnten z. B. folgende Felder von Bedeutung sein:

Feld 1: Autor	z. B. Autorenkollektiv
Feld 2: Titel des Buches	- Stoffsammlung Informatik
Feld 3: Verlag	Staatsverlag der DDR
Feld 4: Erscheinungsjahr	1988

Die Anzahl der Felder wäre in diesem Fall 4. Die Feldnamen sind dann entsprechend einzugeben. Nach diesen Eingaben erscheint das Hauptmenü auf dem Bildschirm und es können folgende Funktionen ausgewählt werden:

(1) Neue Daten eingeben
(2) Daten suchen oder löschen
(3) Daten ändern
(4) Daten ausgeben
(5) Daten auf Band speichern
(6) Speicherplatz prüfen
(7) Ende

Hinweise:

- Es ist zu beachten, daß beim Anlegen der Datei der vorhandene Speicherumfang nicht überschritten wird. Dazu ist der Speichertest (Kennziffer 6) zu nutzen.

- Wenn mit einem 64 K-RAM gearbeitet wird, kann der erweiterte Speicherumfang nur genutzt werden, wenn in Zeile 240 der Parameter von CLEAR 14000 erhöht wird. Es ist zu beachten, daß abgespeicherte Dateien nicht zwischen den einzelnen Speicherkonfigurationen (Grundausrüstung, Speichererweiterung mit 16 K Byte und Speichererweiterung mit 64 K Byte) austauschbar sind.

8.1. Auf der Kassette befindet sich direkt hinter dem Programm "UNIDATEI" ein Beispiel für eine Datei (nur ohne Speichererweiterung nutzbar). Laden Sie diese Datei! Die Datei wird in zwei Files eingelesen.

8.2. Arbeiten Sie mit der Datei"SOFTWARE" entsprechend den Kennziffern 1, 2, 3, 4 und 6 des Menüs. Beachten Sie beim Ändern (KZ 3), daß bei Korrekturen stets das gesamte Feld eingegeben wird. Das Suchen von Daten ist bereits durch Eingabe eines Teils des Strings möglich.

8.3. Starten Sie das Programm neu,und legen Sie eine Kartei mit selbstgewählten Merkmalen (Feldern) an! Achten Sie nach Möglichkeit auf kurze Feldnamen und darauf, daß die Daten für ein Feld nicht über eine Bildschirmzeile hinausgehen. Ein übersichtliches Schriftbild können Sie erreichen, wenn die einzelnen Feldnamen rechtsbündig abgeschlossen werden (z. B. durch Leerzeichen, Punkte, Sterne).

8.4. Speichern Sie Ihre Datei auf Kassette. Die erforderlichen Arbeiten werden im Dialog mit dem Computer realisiert. Da auf ein Korrekturlesen (VERIFY) verzichtet wurde, sollten Sie sich eine Sicherheitskopie anfertigen.

Mit dem Kalkulationsprogramm UNICALC (CALC-BA) können häufig wiederkehrende Berechnungen und Variantenvergleiche durchgeführt werden. Das Programm entspricht in wesentlichen Funktionen dem Kalkulationsprogramm an Personal-, Büro- und Arbeitsplatzcomputern. Die Arbeit mit dem Programm UNICALC erfolgt wahlweise über ein Grund bzw. Funktionsmenü.

Nach dem Laden und dem Start des Programms erscheint das Arbeitsblatt mit dem Grundmenü (vgl. Bild 19).

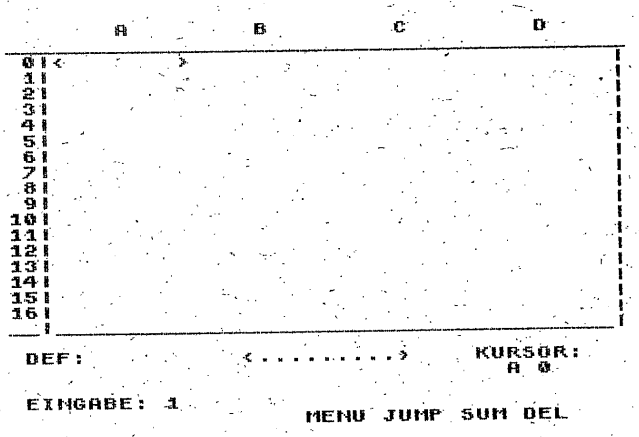


Bild 19: Arbeitsblatt und Grundmenü

Das Arbeitsblatt besteht aus 17 Zeilen (Ø - 16) und 26 Spalten (A - Z), von denen jedoch nur die Spalten A bis D auf dem Bildschirm sichtbar sind. Alle weiteren Spalten lassen sich je nach Bedarf aufrufen.

Eine weitere Größe des Arbeitsblattes sind die Felder. Sie lassen sich eindeutig durch die Angabe der entsprechenden Zeile und Spalte kennzeichnen (z. B. B3 : Feld im Schnittpunkt der Spalte B und der Zeile 3). Die Länge eines Feldes beträgt 9 Zeichen.

Mehrere Felder können zu einem Block zusammengefaßt werden. Ein Block entspricht einer Matrix bestehend aus Zeilen und Spalten. Er wird gekennzeichnet durch die Angabe des linken oberen und rechten unteren Eckfeldes (z. B. in einem Block A2 : C5 sind alle Felder von A2 bis A5, B2 bis B5 und C2 bis C5 enthalten).

Für die Arbeit mit dem Programm UNICALC sind neben Feld-, Spalten-, Zeilen- und Blockangaben auch Bereichsangaben erforderlich. Ein Bereich kann umfassen:

- eine Spalte
- eine Zeile,
- ein Feld,
- mehrere zusammenhängende Felder einer Spalte,
- mehrere zusammenhängende Felder einer Zeile,
- einen Block.

Beachte: Der Begriff Feld wurde in Anlehnung an Kalkulationsprogramme des PC gewählt. Er ist aus mathematischer Sicht nicht exakt und entspricht auch nicht entsprechenden BASIC-Anweisungen.

Die Bezeichnung eines Bereiches erfolgt durch die Angabe des ersten und letzten Feldes, getrennt durch Doppelpunkt (z. B. B1 : B5 oder A2 : C5).

Nach dem Start des Programms steht der Feldkursor (< >) im Feld A1. Er kann durch die Kursorsteuertasten in jedes beliebige Feld bewegt werden. Dieses Feld wird dadurch zum aktiven Feld und übernimmt nach Betätigung der Taste ENTER die im Eingabefeld stehenden Zahlen bzw. Strings sowie die vorgegebenen mathematischen Definitionen (vgl. OP im Funktionsmenü).

Unterhalb des Arbeitsblattes befinden sich das Grundmenü (MENU, JUMP, SUM, DEL) sowie Möglichkeiten zur Darstellung von Informationen über Formeleingaben (DEF), den Feldinhalt (<----->), die Kursorposition sowie die aktuelle Eingabe.

Über MENU (M) gelangt man in das Funktionsmenü, das weitere Kommandos zur Arbeit mit dem Kalkulationsprogramm enthält. Nach Ausführung der gewählten Eingabe geht das Programm wieder in das Grundmenü zurück. Wird die Ausführung eines gewählten Kommandos nicht gewünscht, gelangt man auch durch ENTER wieder zurück.

Übersicht über die Kommandos des Programms

Bei den einzelnen Kommandos wird der erste Buchstabe eingegeben. Bei DEL wird jedoch die entsprechende Taste betätigt.

- Kommandos des Grundmenü

Eingabemodus: Der Computer befindet sich nach dem Start des Programms im Eingabemodus für Zahlen und Strings.

Zahlen werden direkt eingegeben und auf dem Bildschirm rechtsbündig mit 2 Dezimalstellen dargestellt. Sie können bis zu 9 Ziffern lang sein. Bei der Eingabe von Strings wird ein Apostroph "'" vorangestellt. Die Strings werden linksbündig dargestellt und können bis zu 36 Zeichen betragen.

Eine Korrektur der Eingabe ist durch die Betätigung der DEL-Taste möglich. Der Eingabekursor " — " rückt dann eine Stelle nach links.

MENU Zum Aufruf des Funktionsmenüs, durch das weitere Kommandos angeboten werden, Rückkehr durch Abarbeitung des gewählten Kommandos bzw. durch Betätigung der ENTER-Taste.

JUMP Zur Auswahl von Spalten, die auf dem Bildschirm sichtbar werden sollen. Die bei JUMP eingegebene Spalte erscheint am linken Bildschirmrand. Die Lage des Kursors auf dem Bildschirm verändert sich dabei nicht. Im Kursorfeld wird die aktuelle Position angezeigt. Neben der Verschiebung des"Spaltenfensters" auf dem Bildschirm kann über JUMP auch ein schnelles Positionieren des Kursors erreicht und somit effektiv Felder am Tabellenende aktiviert werden.

SUM Zum Bilden von Summen für Bereiche. Das Ergebnis wird in das durch den Feldkursor gekennzeichnete aktive Feld geschrieben und zusammen mit der dazugehörenden Formel gespeichert.

DEL Zum Löschen von Bereichen.
Beachte: Taste DEL betätigen. Mit DEL gelöschte Felder besitzen nicht den Wert Ø. Um spätere Kalkulationen richtig durchzuführen, muß in alle Felder, die in mathematische Operationen (vgl. OP) einbezogen werden, eine Ø eingetragen werden. Felder, in denen Formeln gespeichert sind, werden exakt gelöscht.

Kommandos des Funktionsmenüs

Der Aufruf des Funktionsmenüs erfolgt durch MENU

KALK Werden im Arbeitsblatt Veränderungen vorgenommen, kann mit KALK erneut durchgerechnet werden. Dabei werden die neuen Ergebnisse durch die ebenfalls abgespeicherten Formeln (vgl. SUM und OP) errechnet und das Arbeitsblatt korrigiert. Der zu kalkulierende Bereich ist anzugeben.

SAVE Speichern des Inhalts eines Arbeitsblattes, einschließlich aller Formeln auf Kassette. Der Name des File (CAL) wird durch das Programm vorgegeben und kann nicht frei gewählt werden.

LOAD Laden eines File von Kassette. Beim Pfeifton ist die Taste S (START) zu drücken.

PRINT Der Inhalt des gesamten Arbeitsblattes bzw. von Bereichen kann über einen Drucker ausgegeben werden.

TITEL Zum Feststellen einer "Titelspalte". Die festgelegte Spalte wird durch das Programm an den linken Rand des Arbeitsblattes gelegt und durch zwei Sterne (*) gekennzeichnet. Dabei wird die Spalte A überschrieben, bleibt aber intern erhalten und kann jederzeit sichtbar gemacht werden. Manipulationen sind mit der Titelspalte nicht möglich. Sie kann aufgehoben werden, wenn "T" ohne Parameter (nur ENTER) betätigt wird.

COPY Zum Kopieren eines Feldes, mehrerer Felder einer Spalte bzw. der kompletten Spalte in eine andere Spalte. Es sind zwei Angaben erforderlich, die beide mit ENTER abzuschließen sind ("von : bis" und "nach" : z. B. "AZ : A7" "0").

REPEAT Der Inhalt des aktiven Feldes kann über mehrere Spalten kopiert werden. Beispielsweise bei Unterstreichungen von Tabellen.

DEZ Beim Start des Programms werden Zahlen mit zwei Dezimalstellen dargestellt. Durch DEZ können die Dezimalstellen zwischen Ø und 4 verändert werden. Die interne Speicherung der Zahlen erfolgt jedoch trotzdem in voller Länge. Bei Aufruf von DEZ ist grundsätzlich eine Zahl einzugeben. Es ist nicht möglich, nur mit ENTER ins Grundmenü zurückzukehren.

OP Durch den Aufruf von OP stehen die folgenden mathematischen Operationen zur Auswahl:

- Addieren +
- Subtrahieren -
- Multiplizieren *
- Dividieren /
- Potenzieren ^

Es können miteinander verknüpft werden:

- eine Spalte mit einer Spalte
- eine Spalte mit einer Konstanten
- eine Konstante mit einer Spalte

Nach der Eingabe der gewünschten Operation bei "DEFINITION:" muß bei der Frage "von : bis" noch der Teil einer Spalte angegeben werden, in dem das Ergebnis gespeichert werden soll (z. B. DEFINITION : A * 2, die Felder von AØ bis A7 sollen mit 2 multipliziert werden. Das Ergebnis kann in jede beliebige freie Spalte, jedoch nur von Zeile Ø bis 7 abgespeichert werden).

Beachte: Operationseingaben werden durch das Programm nicht geprüft! Wird bei DEFINITION eine im Programm nicht enthaltene Operation bzw. eine unsinnige Eingabe (z. B. A % B) eingegeben, so wird das Programm mit FC-ERROR unterbrochen und der Computer befindet sich im Kommandomodus. Damit die aktuellen Speicherinhalte erhalten bleiben, kann durch die Betätigung der Taste F1 ein Warmstart durchgeführt werden.

Sind Operationen mit mehr als zwei Operanden durchzuführen, so sind "Hilfsspalten" einzurichten in denen die Zwischenergebnisse gespeichert werden können.

Beispiel:

$$R = \frac{\rho \cdot l}{A}$$

Spalte A = Rho
" B = L
" C = A
" D = Zwischenergebnis = A*B
" E = Ergebnis = D/C

Zur Arbeit mit UNICALC:

8.5. Starten Sie das Programm, und machen Sie sich als erstes mit dem Grundmenü sowie den Kommandos DEZ, OP, TITEL, COPY und REPEAT des Funktionsmenüs vertraut.

8.6. Wählen Sie im Funktionsmenü das Kommando LOAD, und laden Sie das 1. Beispiel (Kreis) von der Kassette (das Beispiel befindet sich auf der Kassette direkt hinter dem Programm "CALC-BA")!

1. Geben Sie für die Anzahl der Dezimalstellen den Wert Ø ein. Gehen Sie mit dem Kursor auf die Felder D4 bis D8 und vergleichen Sie die Eintragung im Arbeitsblatt mit der in der Feldanzeige (<...>). Gleichzeitig erkennen Sie bei "DEF:" die Operation, mit der der Wert in dem entsprechenden Feld errechnet wurde!

2. Verändern Sie die Radien in den Feldern A4 bis A6 in 1ØØ, 1Ø und 5!

3. Lassen Sie sich die Spalten B bis E anzeigen, indem Sie JUMP B eingeben!

4. Da die Ergebnisse in den Spalten B bis E noch für die ersten Radien 3, 5, 7 zutreffen, müssen Sie die Tabelle neu durchrechnen lassen. Geben Sie -KALK - AØ : E 9 - ein, und beobachten Sie die Wirkung des Kommandos!

5. Unterbrechen Sie das Programm mit BRK, und beobachten Sie die Wirkung der Funktionstaste F1. Unterbrechen Sie nochmals das Programm, und starten Sie jetzt mit RUN!

8.7. Laden Sie das 2. Beispiel (Betrieb) von der Kassette!

1. Entfernen Sie die Betriebsabteilung 3 mit allen Angaben aus der Tabelle, und vergleichen Sie die Spalte E13 vor und nach der Kalkulation!

2. Geben Sie für die Anzahl der Dezimalstellen den Wert Ø ein!

3. Entfernen Sie auch die Abteilungen 1 und 2, und stellen Sie jetzt nach dem vorgegebenen Angaben im Arbeitsblatt selbst ein Beispiel zusammen. Speichern Sie anschließend das Beispiel auf Kassette.

Für den KC 85/3 bzw. KC 85/2 wird vom Hersteller ein TEXOR-Modul bereitgestellt. Dieser Modul enthält ein Programm, mit dem wesentliche Operationen der Textverarbeitung möglich sind. Bisher wurden Texte immer auf Schreibmaschinen geschrieben. Dabei mußten Tippfehler geschickt auf dem Papier korrigiert werden bzw. war die ganze Seite neu zu schreiben. Außerdem war es nicht möglich, mehr als 5 Durchschläge sauber bereitzustellen, ohne den Text nicht mehrfach einzutippen. Mit der Anwendung des Textverarbeitungsprogramms ist es möglich, wesentlich mehr Text zu speichern, als es schon moderne elektronische Schreibmaschinen zulassen. Ein weiterer Vorteil besteht darin, daß beim Bearbeiten des Textes dieser auf dem Bildschirm erscheint und alle Korrekturen und Veränderungen im direkten Dialog mit dem Computer vorgenommen werden können. Der fertige Text kann abgespeichert und beliebig oft ausgedruckt werden.

Die Beschreibung der Arbeit mit dem TEXOR-Modul ist dem Handbuch, welches zum TEXOR-Modul mitgeliefert wird, zu entnehmen:

Kleincomputer KC 85/3
Beschreibung zu MO12 TEXOR
Staatsverlag der DDR

Mit diesem TEXOR-Modul können mit dem Kleincomputer wesentliche Funktionen realisiert werden, wie sie auch an Personalcomputern mit einem Textprozessor möglich sind.

Für die Lösung beruflicher Aufgaben mit einem Computer ist es erforderlich, sich mit dem jeweils anzuwendenden Standardprogramm oder einem speziellen Anwenderprogramm eingehend vertraut zu machen. Nur durch genaue Kenntnis der verschiedenen Computerfunktionen ist ein effektiver Einsatz der Technik möglich.

9. Entwicklungstendenzen des Computereinsatzes

Für die umfassende Nutzung moderner Informationsverarbeitungstechnologien wurden in unserer Republik in den vergangenen Jahren entscheidende Grundlagen geschaffen. In relativ kurzer Zeit vollzogen sich in fast allen Bereichen unserer Volkswirtschaft durch den Einsatz moderner Technik gravierende technisch-technologische Veränderungen. In der DDR gibt es gegenwärtig schon mehr als 24 700 CAD/CAM-Arbeitsstationen und bis 1990 sind etwa 85 000 bis 90 000 geplant. Insgesamt sind über 35 000 Büro- und Personalcomputer im Einsatz, 1990 werden es zwischen 160 000 und 170 000 sein. Durch einen effektiven Einsatz dieser Technik ist es möglich, eine Steigerung der Arbeitsproduktivität auf das 5- bis 6fache und eine Senkung der Kosten um 15 bis 20 % zu erreichen. Dabei erfolgt der Einsatz der Computer vor allem zur Übernahme der unmittelbaren Steuerung von Fertigungsprozessen, zur Übernahme formalisierbarer informationsverarbeitender Leistungen und zur Automatisierung vielfältiger routinemäßiger Tätigkeiten. Durch diese Einsatzbreite der Computer werden immer mehr Werktätige bei der Ausübung ihres Berufes mit dieser Technik konfrontiert und damit zugleich mit der Notwendigkeit, sich mit dieser Technik vertraut zu machen, sich zu qualifizieren. Es vollziehen sich somit nicht nur technische, sondern auch tiefgreifende soziale Veränderungen. Schwerpunkt und Ziel des Einsatzes der modernen Technik in unserem Land ist die weitere Erhöhung des materiellen und kulturellen Lebensniveaus aller Menschen. Der Einsatz von Computern zum Wohle des Menschen heißt aber auch, Schaffung von optimalen Bedingungen für die Entwicklung und Entfaltung der Fähigkeiten jedes einzelnen. Das jedoch ist nur möglich, wenn der Einsatz der modernen Technik nicht durch Profit- und Machtstreben beeinflußt wird. Besonders auf diesem in der ökonomischen Auseinandersetzung der beiden Gesellschaftssysteme so wichtigem Gebiet kommt es darauf an, den wissenschaftlich-technischen Fortschritt eng mit den Vorzügen des Sozialismus zu verbinden.

Im Zusammenhang mit der Mikroelektronik und Rechentechnik hat die Informatik bereits in vielen Bereichen des gesellschaftlichen Lebens Einzug gehalten. Der gesellschaftliche Anwendungsbereich liegt besonders

- in der Forschung und Entwicklung bis zum Entwurf von Produkten und Technologien,
- in der Produktion bzw. Fertigung von Erzeugnissen,
- in der Produktions- und Fertigungsplanung sowie
- in der Leitung und Planung im Büro, in der Verwaltung und im Bildungs- und Dienstleistungsbereich.

Aber auch im privaten bzw. Konsumgüterbereich ist die Mikroelektronik anzutreffen, z. B. in der Steuerung von Waschmaschinen, in der Unterhaltungselektronik, in der optimalen Regelung der Kraftstoffzuführung in Kraftfahrzeugen u. a.[1] Die Informatik mit ihren zahlreichen Teilgebieten spielt in der weiteren Entwicklung einer effektiven Gestaltung der informationsverarbeitenden Prozesse eine wesentliche Rolle. Schwerpunktaufgaben der nächsten Jahre sind vor allem:

- Beschleunigung des Entwurfs höchstintegrierter Schaltkreise,
- Erschließung neuer Rechnerarchitekturen und Softwaretechnologien,
- breite Anwendung der Methoden der künstlichen Intelligenz,
- Vernetzung von Computern im Land und über die Landesgrenzen hinaus,
- verstärkte Anwendung des rechnergestützten Entwurfs und der rechnergestützten Fertigung (CAD/CAM-Technologie) bis hin zur rechnerintegrierten Fertigung (CIM-Technologie),

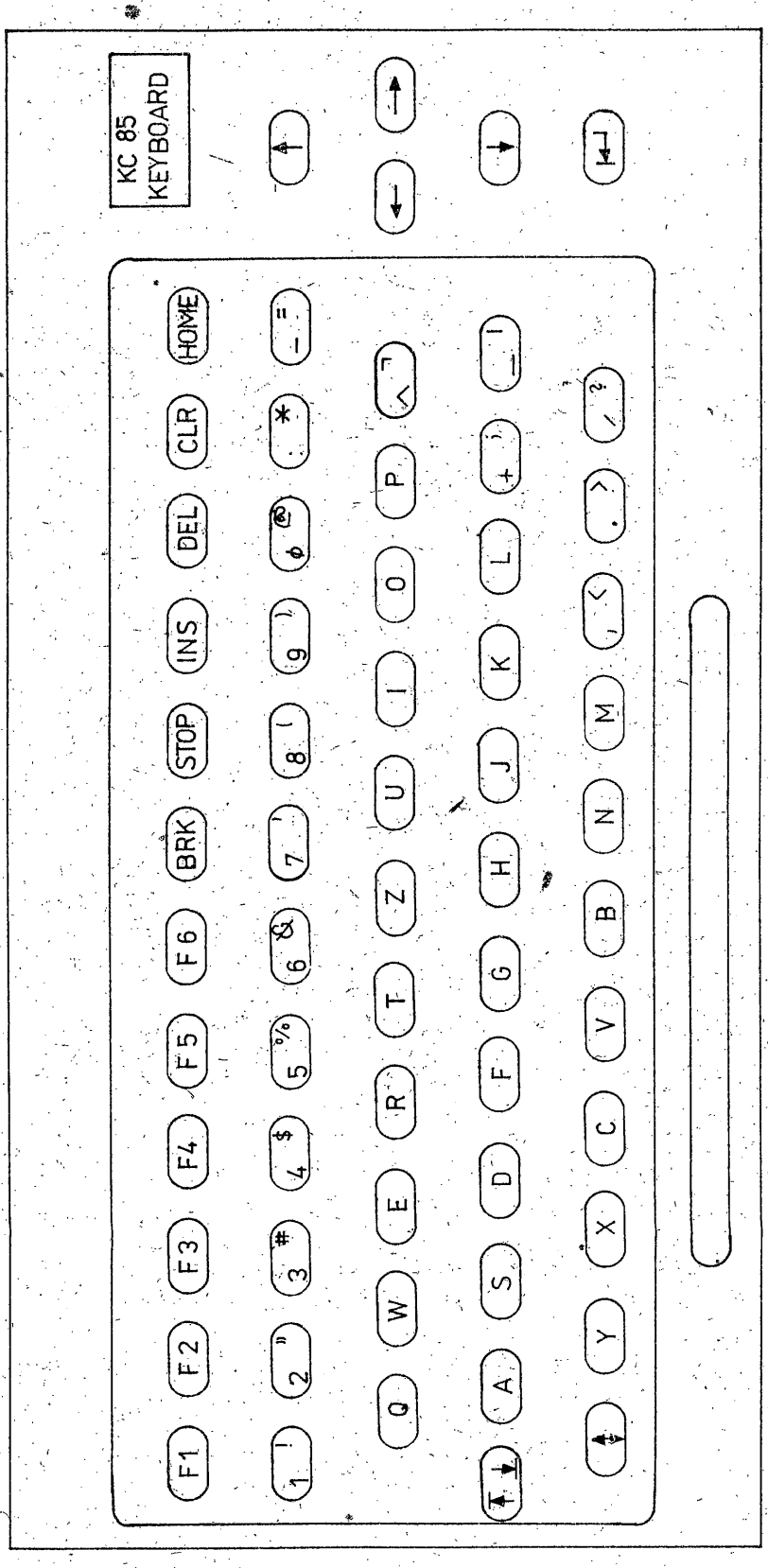
1 Vergleiche Prof. Dr. sc. D. Hammer: Informatik - Stand und Perspektiven. In: URANIA-Extra, Sonderheft 1987

- Automatisierung der Büro- und Verwaltungsarbeit einschließlich der Prozesse der Leitung und Planung,
- Weiterentwicklung der Kommunikationstechnologien durch digitale hochleistungsfähige Nachrichtenübertragungsnetze.

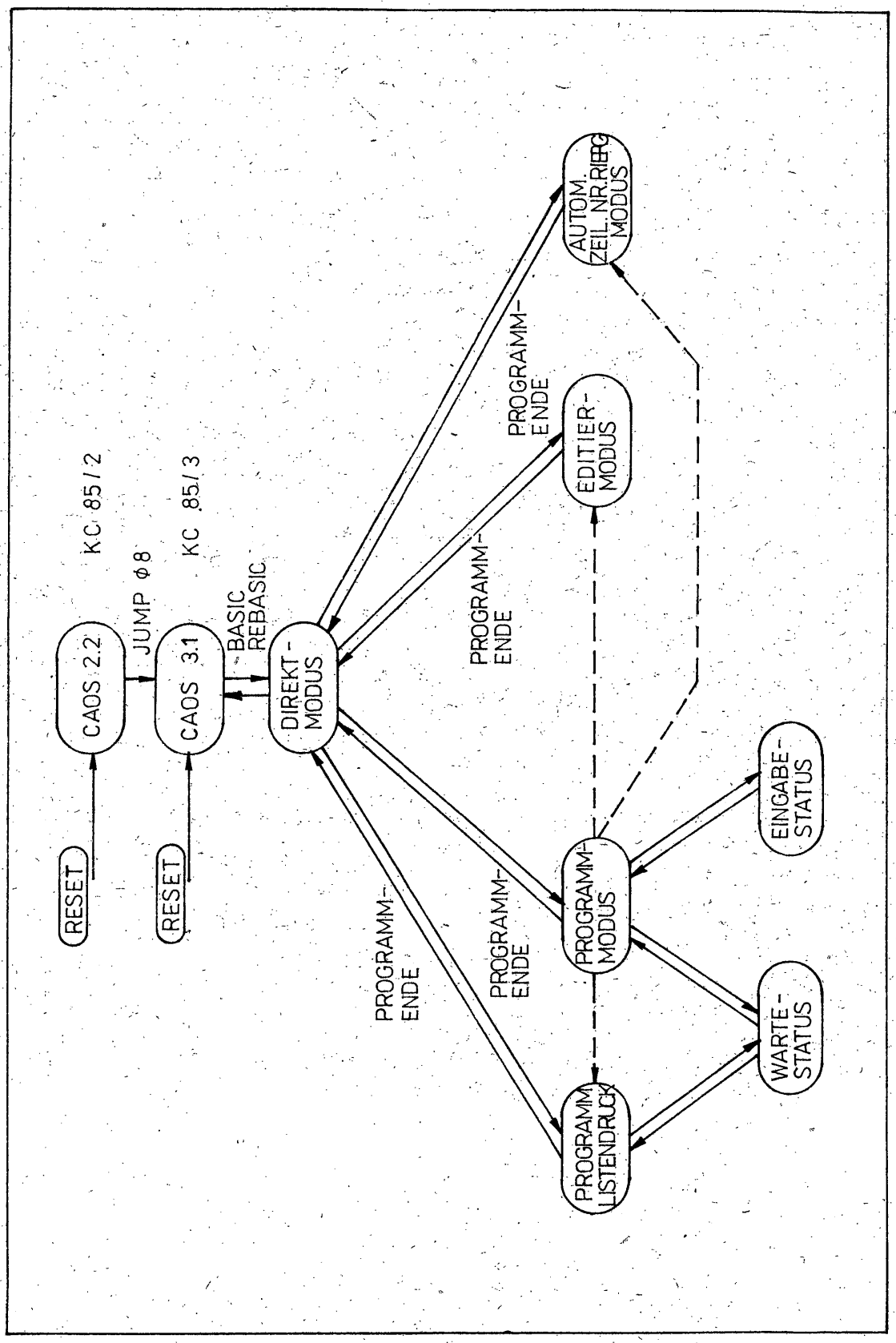
Die Vielzahl der weiter zu erforschenden Probleme des Einsatzes moderner Informationsverarbeitungstechnik zeigt, daß wir mit dieser Entwicklung noch am Anfang stehen. Es eröffnen sich noch ungeahnte Möglichkeiten, diese Technik zum Wohle des Menschen zu nutzen.

Zugleich ergibt sich für jeden, ob Lehrling, Facharbeiter, Meister oder Ingenieur, die Notwendigkeit der Aneignung von Kenntnissen auf diesem Gebiet. Das betrifft das aufmerksame Verfolgen der weiteren Entwicklung des Einsatzes von Computern im speziellen Berufsgebiet, der Weiterentwicklung der Gerätetechnik insgesamt sowie das Erlernen des Umgangs mit der Computertechnik, mit speziellen Anwenderprogrammen (Benutzersystemen) bis hin zum Erlernen spezieller Programmiersprachen.

Anlage 1: Übersicht zur Tastatur des KC 85/3 bzw. KC 85/2
(Arbeitsblatt)



Anlage 2: Übersicht über Arbeitsmodi des Computers und BASIC-Interpreters
(Arbeitsblatt)



Anlage 3: Standardfunktionen und mathematische Operationen

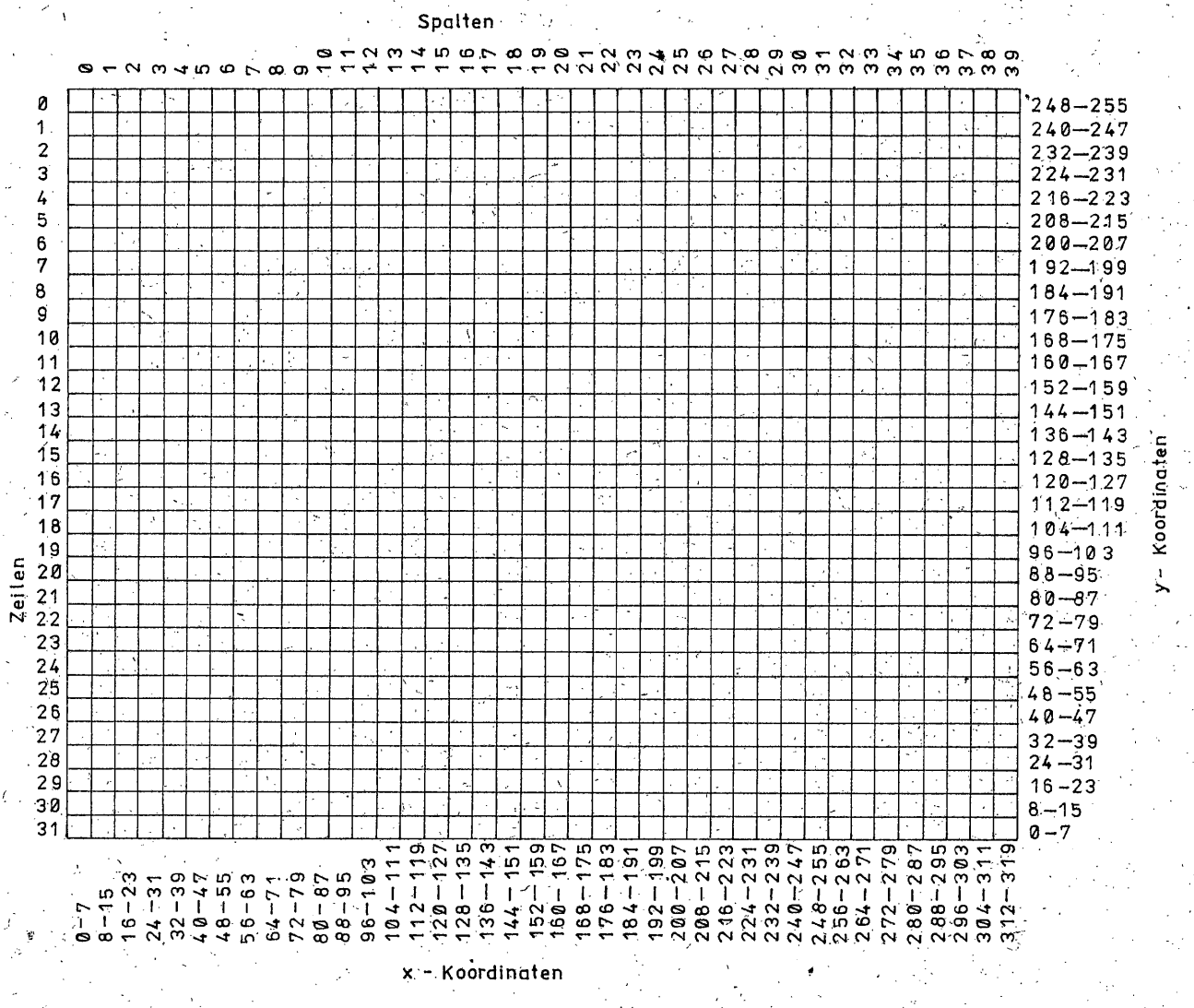
ABS (X)	$\|X\|$	Betrag von X
ATN (X)	arctan x	Resultat im Bogenmaß
COS (X)	cos x	x im Bogenmaß
EXP (X)	e^x	x <= 87,3365
INT (X)	ganzer Teil von x	
LN (X)	ln x	x > Ø Logarithmus von x
PI	π	= 3,14159
SGN (X)	Signumfunktion	-1 für x < Ø = Ø für x = Ø 1 für x > Ø
SIN (X)	sin x	x im Bogenmaß
SQR (X)	$\sqrt{x}$	x> 0
TAN (X)	tan x	x im Bogenmaß
Klammer	( )	
Exponenten	^	
negative Vorzeichen		
/,* (Division, Multiplikation)		arithm. Operatoren
+, - (Addition, Subtraktion)		
Vergleichsoperatoren: =, <, >, <= , >=, <> (ungleich)		
NOT		logische Operatoren
AND		
OR		

Mit der Anweisung DEF FN Name (verwendetes Argument) = Ausdruck können weitere Funktionen definiert werden.

Anlage 4: Fehlermeldungen

BS	Feldelement außerhalb des dimensionierten Bereiches aufgerufen
DD	Feld mehrfach dimensioniert
FC	Unzulässiger Funktionsaufruf
ID	Fehlerhafte Eingabe
MO	Anweisung unvollständig
NF	Variablen von NEXT und FOR passen nicht zusammen
OD	Es fehlten Daten in der DATA-Anweisung
ON	Speicherplatz unzureichend
OV	Zu großes Ergebnis der Berechnung
SN	Syntaktischer Fehler
RG	RETURN ohne GOSUB
UL	Es wurde eine nicht existente Zeilennummer angegeben
Ø	Division durch Null
CN	Programm kann nicht mit CONT fortgesetzt werden
LS	String länger als 255 Zeichen
OS	Vereinbarter Speicherplatz für String reicht nicht aus
ST	String zu lang oder zu komplex
TM	Variablen einer Gleichung indizieren verschiedene Typen, z. B. Zahl und String
UF	Funktion noch nicht definiert
IO	Falscher Name beim Programmladen

Anlage 5: Koordinatensysteme des Bildschirms



Anlage 6: Farbcodetabelle

Vordergrundfarbe F	Nummer	Hintergrundfarbe H
Schwarz	Ø	Schwarz
Blau	1	Blau
Rot	2	Rot
Purpur	3	Purpur
Grün	4	Grün
Türkis	5	Türkis
Gelb	6	Gelb
Weiß	7	Grau
Schwarz	8	
Violett	9	
Orange	1Ø	
Purpur	11	
Grünblau	12	
Blaugrün	13	
Gelbgrün	14	
Weiß	15	

Der Farbcode für die Vordergrundfarbe V errechnet sich wie folgt:

V = 16 * B + F

Es gilt B = 1 für Blinken und B = Ø für Nicht-Blinken.

F = Farbcodenummer für Vordergrundfarbe

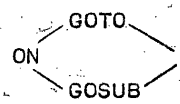
Beispiel:

V = 16 * 1 + 2 — Vordergrund rot blinkend
V = 18

V = 16 * 0 + 2 — Vordergrund rot nichtblinkend
V = 2

Anlage 7: BASIC-Anweisungen des KC 85/3 und KC 85/2 mit BASIC-Modul

Anweisung	Format	Bemerkung
AUTO	AUTO (Zeilennummer), (Schrittgröße)	Selbständige Zeilennummerierung
BLOAD	BLOAD	Ruft Betriebssystem-LOAD zum Einlesen von Maschinenprogrammen
BYE	BYE	Rückgabe der Steuerung an das Betriebssystem
CLEAR	CLEAR	Löscht den Variablenspeicher (ist auch in RUN enthalten)
CONT	CONT	Setzt unterbrochene Programme fort
CSAVE	CSAVE "NAME"	Abspeichern eines Programms auf Kassette
DELETE	DELETE Anfangszeile, Endzeile	Löscht Zeilen
EDIT	EDIT Zeilennummer	Programmkorrektur
LINES	LINES Zahl	Legt anzuzeigende Zeilenanzahl bei LIST fest
LIST	LIST oder LIST Zeilennummer	Listet Programme
NULL	NULL Zahl	Legt die Anzahl der auszugebenden Dummyzeichen am Zeilenende fest
RENUMBER	RENUMBER (Ab-alte-Zn.), (Bis-alte-Zn.), (Ab-neue Zn.), (Schrittgröße)	Neunummerierung der Programmzeilen
RUN	RUN oder RUN Zeilennummer	Startet das Programm und führt es aus (CLEAR ist enthalten)
TROFF	TROFF	Schaltet den Kontrollmodus aus
TRON	TRON	Schaltet den Kontrollmodus ein
WIDTH	WIDTH Zeichenanzahl	Legt die Länge der Ausgabezeilen fest
BEEP	BEEP (N)	Erzeugt einen Ton mit festgelegtem Parameter (1<= N<= 255)
CALL	CALL I	Aufruf eines Maschinenprogramms mit Startadresse I
COLOR	COLOR V,H	Stellt Vorder- und Hintergrundfarbe ein
DATA	DATA Konstantenliste	Folge der durch READ zu lesenden Werte
DEEK	DEEK (I)	Inhalt der Speicherplätze I und I+1
DEF FN	DEF FN Name (Verwendetes Argument) = Ausdruck	Definiert eine Funktion
DIM	DIM Variable (Index, ...)	Dimensioniert Variablenfelder
DOKE I, J	DOKE I, J	Schreibt den Wert J in die Speicherplätze I und I+1
END	END	Beendet das Programm
FOR TO STEP	FOR Variable = Anfangswert TO Endwert STEP Schrittweite	Festlegung einer Programmschleife
GOSUB	GOSUB Zeilennummer	Unterprogrammaufruf
GOTO	GOTO Zeilennummer	Sprunganweisung
IF THEN ELSE	IF Ausdruck THEN Anweisung: ELSE Anweisung	Programmverzweigung
INK	INK V	Stellt Vordergrundfarbe ein
INKEY $	INKEY $	Tastaturabfrage
INP I	INP I	Liefert das aus dem Port I gelesene Byte
INPUT	INPUT Variable (,Variable..) oder INPUT "String"; Variable (,Variable...)	Eingabeanweisung
LET	LET Variable = Ausdruck	Wertzuweisung
LOAD n	LOAD n "Programmname"	Lädt LIST-gespeicherte-Programme
LOCATE	LOCATE Z, S	Positioniert Cursor an bestimmte Stelle (Z = Zeile, S = Spalte)
NEW	NEW	Löscht den Programm- und Variablenspeicher
NEXT	NEXT Variable	Ende der Programmschleife

ON GOTO ON GOSUB	 Liste von Zeilennummern	Mehrfach-Programmverzweigung
OUT I, J	OUT I, J	Gibt das Byte J aus dem Port I aus (Ø<= I, J<= 255)
PAPER	PAPER H	Stellt Hintergrundfarbe ein
PAUSE	PAUSE (T)	Unterbricht Programmabarbeitung (1<= T<= 255)
PEEK(I)	PEEK(I)	Inhalt des Speicherplatzes I
POKE I, J	POKE I, J	Schreibt das Byte J in den Speicherplatz I
PRESET	PRESET X, Y	Löscht Bildpunkte
PRINT	PRINT (Ausdruck) (Trennzeichen) ...	Ausgabe auf dem Bildschirm
PSET	PSET X, Y, F	Setzt Bildpunkte
READ	READ Variablenliste	Ordnet den hinter READ stehenden Variablen die hinter DATA stehenden Werte zu
REM	REM Kommentar	Kommentarkennzeichnung
RESTORE	RESTORE Zeilennummer	Setzt den DATA-Zeiger auf den Anfang der Liste
RETURN	RETURN	Ende des Unterprogramms
SOUND	SOUND Z1, V1, Z2, V2, LS, TD	Erzeugt Töne
STOP	STOP	Stoppt ein Programm
VPEEK	VPEEK(AD)	Ermöglicht Lesen des Inhalts einer Speicherzelle im Bildwiederholspeicher
VPOKE	VPOKE AD, W	Ermöglicht Beschreiben einer Speicherzelle im Bildwiederholspeicher
WAIT	WAIT I, J, (K)	Programm wartet bis ein bestimmtes Bitmuster an einem bestimmten Port erscheint
WINDOW	WINDOW ZA, ZE, SA, SE	Festlegung eines Fensters

Erweiterung der Anweisungen bei Anwendung eines BASIC-Moduls und beim KC 85/3:

RANDOMIZE		Zufallsgenerator wird mit einem zufälligen Anfangswert gestartet
KEY	KEY n	Funktionstastenbelegung
KEYLIST		Auflisten der Funktionstastenbelegung
LINE	LINE XA, YA, XE, YE (,F)	Zeichnen einer Linie
CIRCLE	CIRCLE XM, YM, R, (,F)	Zeichnen eines Kreises
CSRLIN	CSRLIN (n)	Liefert die Nummer der Zeile, in der sich der Cursor gerade befindet
SWITCH	SWITCH mm, kk	Schalten von Modulen innerhalb eines Programms
OPEN	OPEN r # n "Name"	Eröffnet eine Kanaloperation für dateifähige Geräte
CLOSE	CLOSE r # n	Schließt den Kanal nach Datenein- oder Datenausgabe
PRINT #	PRINT # n Daten	Ermöglicht die Ausgabe von Daten (z. B. auf Drucker oder Recorder)
INPUT #	INPUT # n Daten	Ermöglicht die Eingabe der mit PRINT ausgegebenen Daten
LIST #	LIST # n "Programmname"	Ausgabe eines Programmlistings
LOAD #	LOAD # n "Programmname"	Einlesen der mit LIST ausgegebenen Programme
PTEST	PTEST (x)	Test, ob ein Bildpunkt gesetzt ist

Anlage 8: Hinweise zur Programmoptimierung[1]

Bei der Optimierung von Programmen gibt es drei wesentliche Anforderungen:

- minimaler Speicherplatz für das Programm und dessen Abarbeitung,
- schnellstmögliche Abarbeitung des Programms
- gute Selbstdokumentation im Listing.

Diese Anforderungen gelten nicht nur in BASIC. Sie sind jedoch nicht gleichzeitig zu realisieren. Ein sich selbst gut dokumentierendes Programm benötigt z. B. mehr Speicherplatz, als ein Programm ohne Erläuterungen. Deshalb muß man sich für die Erfüllung einer der sich z. T. widersprechenden Anforderungen entscheiden.

Im folgenden sollen einige generelle Regeln für die Optimierung von Programmen genannt werden.

1. Minimierung des Speicherplatzes

. Vielfachanweisungen je Zeile sparen je Anweisung 4 Byte
. Weglassen von Leertasten
. Weglassen von REM-Texten
. Mehrfachverwendung von Variablen
. NEXT ohne Argument (wenn zulässig)
. GOSUB für mehrfach verwendete Programmteile
. Nullelemente von Arrays verwenden (DIM-Anweisung)
. LET weglassen
. Bei vielen Variablen (ab etwa 5 bis 7 Elementen) sind Arrays vorteilhafter
. Mehrfach verwendete Konstanten und Strings in Variablen bzw. DATA-Zeilen ablegen
. In Argumenten, z. B. bei PRINT, TO, STEP usw., Verknüpfungen zulassen
. Für mehrfach verwendete Verknüpfungen neue Variablen einmal definieren bzw. DEF FN verwenden

2. Beschleunigung des Programmablaufes

. Alle REM entfernen
. Variablen statt Zahlen verwenden
. Häufig verwendete Variablen zu Beginn des Programmablaufes initiieren
. Zeilen für GOSUB- und GOTO-Sprünge möglichst weit vorn im Programm anlegen
. GOSUB ist mit Rücksprung zusammen wesentlich schneller als zweimal GOTO
. FOR-NEXT ist wesentlich schneller als GOTO oder GOSUB
. Bei NEXT Variable weglassen (abhängig vom BASIC-Dialekt)
. FOR-NEXT-Schleifen, vor allem innere (bei mehrfachen) einfach gestalten, u. a.
  - keine Variablen initiieren
  - überflüssige Rechnungen vermeiden
  - kein REM benutzen
  - keine GOTO verwenden
. LET beschleunigt in vielen Dialekten
. Arithmetik einfach gestalten, z. B.
  2 * A durch A + A ersetzen
  A ^ 2 durch A * A ersetzen
  Multiplizieren statt dividieren
. Keine unnötigen Wiederholungen von Rechnungen
. DEF FN statt GOSUB verwenden
. Logik aufgliedern
. Konstanten verwenden
. Feldvariablen bei häufiger Benutzung übertragen
. Wertetabelle verwenden

1 Vgl. auch H. Grote, H. Völz: BASIC-Einmaleins des Programmierens in URANIA-Sonderhaft 1987

3. Gute Lesbarkeit der Programme

- Gut strukturieren, z. B. mit Subroutinen
- REM zur Erklärung der Funktion des Programmes und von Programmteilen
- REM zum Hervorheben von Programmteilen
- Keine Variablen doppelt verwenden
- Strukturierung durch Leertasten (Space) und Doppelpunkte
- Tiefe der FOR-NEXT-Schleifen durch Zurücksetzen der Zeilen anzeigen

Anlage 9: Literaturhinweise

Bueckner, U.: Kleincomputer leichtverständlich, Leipzig, Fachbuchverlag, 1986

Grote, U.; Völz, H.: BASIC Einmaleins des Programmierens, URANIA EXTRA, Berlin 1987

Gutzer, H.: Das kann der Mikrocomputer, Urania Verlag, Leipzig 1986

Heblik, P.: Wissensspeicher BASIC, Volk und Wissen, Volkseigener Verlag, Berlin 1986

Hopfer, R.; Müller, R.: BASIC-Einführung in das Programmieren, Fachbuchverlag, Leipzig 1987

Intensivierung - 100 Begriffe zur ökonomischen Strategie. In: Blickpunkt Wirtschaft, Verlag Die Wirtschaft Berlin 1986

Müller, S.: Programmieren mit BASIC, Verlag Technik, Berlin 1985

Hopfer, R.: Mikrorechentechnik - allgemeinverständlich, VEB Fachbuchverlag, Leipzig 1987

Anlage 10: Lösungen der Übungsaufgaben

4.2.4. Programmvorschlag:

```
10 REM SCHLEIFENDARSTELLUNG
20 CLS
30 FOR I = 1 TO 3
40 PRINT:PRINT I;". AEUSSERE SCHLEIFE":PRINT
50 FOR J = 1 TO 5
60 PRINT SPC(6)J;". INNERE SCHLEIFE"
70 NEXT J
80 NEXT I
90 END
```

5.1.2. b, c, e

5.2.1. b, c, d

5.2.11. Programmzeile 70 ist zu löschen.

5.3.5. Programmvorschlag

```
10 CLS:PRINT
20 PRINT "Das Programm berechnet nach"
30 PRINT "Eingabe des Radius"
40 PRINT "den Umfang, die Oberflaeche und"
50 PRINT "das Volumen einer Kugel"
60 PRINT " ------------------------":PRINT:PRINT
70 PRINT "UMFANG", "OBERFLAECHE", "VOLUMEN"
80 INPUT "RA";R:PRINT
90 U = 2 * PI * R : O = 4 * PI * R * R : V = 4/3 * PI * R * R * R
100 PRINT U, O, V : PRINT : GOTO 80
110 END
```

5.3.6. Programmvorschlag

```
10 REM STUNDENPLAN
20 CLS
30 ST$=  "I----I"
40 PRINT ST$;
50 FOR I=1 TO 6
60 PRINT  "----I";
70 NEXT I
80 PRINT
90 PRINT "ISTD I";
100 FOR I=1 TO 6
110 READ TA$
120 PRINT TAB (5*I+2)TA$;" I";
130 NEXT I
140 PRINT
150 PRINT ST$;
160 FOR K=1 TO 6
170 PRINT "----I";
180 NEXT K
190 PRINT
200 FOR I=1 TO 6
210 PRINT "I";1;
220 FOR K=1 TO 7
230 PRINT TAB (K*5) "I";
240 NEXT K
250 PRINT
260 PRINT ST$;
270 FOR K=1 TO 6
280 PRINT "----I";
290 NEXT K
300 PRINT
310 NEXT I
320 DATA MO, DI, MI, DO, FR, SA
330 END
```

5.4.5. Programmvorschlag

```
10 CLS
20 FOR X=1 TO 5
30 FOR Y=1 TO 20
40 PRINT "*";
50 NEXT Y : PRINT
60 NEXT X
70 END
```

5.6.1. Programmvorschlag

```
10 REM ANWESENHEITSLISTE
20 I = 20 : J = 2
30 DIM A$ (I,J)
40 WINDOW 0, 31, 0, 39 : CLS
50 PRINT "NAME", "VORNAME", "EINRICHTUNG"
60 PRINT " ----------------------------------------"
70 WINDOW 3, 31, 0, 39
80 FOR I=0 TO 20
90 LOCATE I+3, 0 : INPUT "";A$(I,0)
100 LOCATE I+3, 12 : INPUT"";A$(I,1)
110 LOCATE I+3, 24 : INPUT"";A$(I,2)
120 NEXT I
130 CLS
140 FOR I=0 TO 20
150 PRINT A$ (I,0),A$(I,1),A$(I,2)
160 NEXT I
170 END
```

6.1.1. Programmvorschlag

```
10 REM FENSTER
20 WINDOW 0, 31, 0, 39 : CLS
30 WINDOW 0, 15, 0, 19 : COLOR 1, 2 : CLS
40 PRINT AT (7,4); "1. Fenster"
50 WINDOW 0, 15, 20, 39 : COLOR 2, 1 : CLS
60 PRINT AT (7, 23); "2. Fenster"
70 WINDOW 16, 31, 0, 19 : COLOR 7, 0 : CLS
80 PRINT AT (23, 4); "3. Fenster"
90 WINDOW 16, 31, 20, 39 : COLOR 0, 7 : CLS
100 PRINT AT (23, 23); "4. Fenster"
110 LOCATE 14, 17
120 END
```

6.1.2. 32mal soviel, also 80k Byte.

Anlage 11: Sachwortverzeichnis

ASCII 53
AUTO 24,94
Algorithmus,Lösungs- 26,31
Alternative,vollständige-,unvollständige 32
Anwendersoftware 10,82
Arbeitsweise des Computers 13
Ausdrücke,arithmetische-,logische- 39
Auswahl 32
Automatische Steuerung 6
BASIC 19,36,94
BEEP 70
BEY 20
BRK-Taste 23
Basissoftware 82
Befehlszyklus 14
Betriebssystem 10,18
Bildschirmeinteilung,-gestaltung 46,54
Binäre Signale 77
Block,-nummer 22,25
Bus,-system 14
CAOS 18
CHR$ 53
CIRCLE 59,95
CLEAR 44,94
CLOAD 22,94
CLS 22
COLOR 18,56,94
CONT 23,94
CPU 15
CSAVE 25,94
CTC 74
Compiler 19
Computerarbeitsplatz 9,17
Computergrafik 58
Computertechnik,Einsatz der- 4,6,88
Cursor 18
DATA 41,94
DEL-Taste 24
DELETE 25,94
DIM 52,94
DISPLAY 18
Digitale Signale 81
Direktmodus 21
EDIT,editieren,Editiermodus 24,94
ELSE 49
END 51,94
ENTER-Taste 18
EPROM 12
ERROR,IO-,OM-,SN-,OV- 23,92
Escape 68
FOR TO STEP NEXT 47,94
Fallunterscheidung 34,50
Farbgestaltung 56,93
Fehler,syntaktische-,logische- 30
Feld 52,83
Fenster 55
Festkommazahl 37
Fluchtfunktion 30,68
GOSUB 51,94
GOTO 48,94
Grundmenü 18
Grundstrukturen,algorithmische- 31
HOME-Taste 22
Hardware 9
IF THEN 49,94
INK 57,94
INKEY$ 44,94
INP 75,94
INPUT 43,94
INS-Taste 24
INSTR 40
INT 63
IRM 13
Informatik 4
Interface 74
Interpreter 19
JUMP 18
KEY 18,95
KEYLIST 18
Kaltstart 20
Keyboard 17
Konstanten,numerische-,String- 36
Koordinatensystem,Zeichen-,Pixel- 46,54,93
Koorekturmöglichkeiten 24
Koppelbaustein 76
LEN 40
LET 41,94
LINE 58,95
LIST 24,94
LOAD 18,20,94
LOCATE 55,94
Laden von Programmen 22
Laufvariable 35
Lineare Programme 32
MEMORY END 20
MENU 18,50
MID$ 40
MODIFY 18
Maschinensprache 19
Matrix 52
Mikroprozessor 10,19
Mikrorechner 9
Modul,-schacht 19,74,87
NASSI-SHNEIDERMAN-Diagramm 31
ON GOSUB 52,95
ON GOTO 50,95
OUT 75,95
Operatoren,mathematische-,logische- 21,92
PAPER 57,95
PIO 12,74
PRESET 58,95
PRINT,-AT,-TAB,-SPC 21,45,95
PROM 12
PSET 58,95
PTEST 61
Problemanalyse 26,28
Programm,-ablaufplan,-testung,-dokumentation 27
Programm,-modus,-eingabe,-entwurf 23,26,28
Programm,-unterbrechung,-zeile 23
Programmiersprache 19,36
Programmierung,strukturierte-,Stufen der- 26,31
Programmoptimierung 96
Programmverzweigung 47
Prozeßsteuerung 74,79
RAM 11
READ DATA 41,95
REBASIC 20
REM 44,95
RESET 20
RESTORE 42,95
RETURN 51,95
RIGHT$ 40
ROM 11
RUN 23,94
Register 16
Reihung 31
SAVE 18
SHIFT-Taste 22
SIO 12
SOUND 68,95
STOP-Taste 23
STR$ 40
STRING$ 40
SWITCH 18,95
Schleifen,abweisende-,nichtabweisende 34,47
Schrittweite 47
Software,-wartung 10,30
Speicher 12
Speichern von Programmen 25
Speicherplatz,-adresse 20,72
Sprung,bedingt,unbedingt 48
Standardfunktionen 38
Standardsoftware 10,81
String,-operationen,-funktionen 39,40
Struktogramm 27,31
Strukturblöcke 31
TEXOR 87

Für die Facharbeiterausbildung ab September 1988 bestätigt.

Staatssekretariat für Berufsbildung

Die Stoffsammlung wurde von einem Autorenkollektiv unter Leitung von Dr. R. Neubert (ZIB) ausgearbeitet.

Autoren:

Bähr, Siegfried	Humboldt Universität zu Berlin
Gatsche, Hans-Jürgen	Betriebsschule "Friedrich Engels" Döbeln
Dr. Müller, Bernhard	VEB Mikroelektronik "Wilhelm Pieck" Mühlhausen
Dr. Neubert, Renate	Zentralinstitut für Berufsbildung der DDR, Berlin
Nichterwitz, Manfred	Zentralinstitut für Berufsbildung der DDR, Berlin
Saegert, Harald	Bezirkskabinett für Weiterbildung der Kader der Berufsbildung Potsdam
Dr. Tuschke, Siegfried	Zentralinstitut für Berufsbildung der DDR, Berlin
Dr. sc. Wolfram, Peter	Technische Universität Dresden

weitere Beiträge lieferten:

Dr. Adamski, Inge	Technische Universität Dresden
Berndt, Annegret	Bezirkskabinett für Weiterbildung der Kader der Berufsbildung Neubrandenburg
Dr. Höpfner, Hans-Dieter	Zentralinstitut für Berufsbildung der DDR, Berlin
Lühe, Gundula	Zentralinstitut für Berufsbildung der DDR, Berlin