

TA alphaTronic PC

ALMANACH '85/86

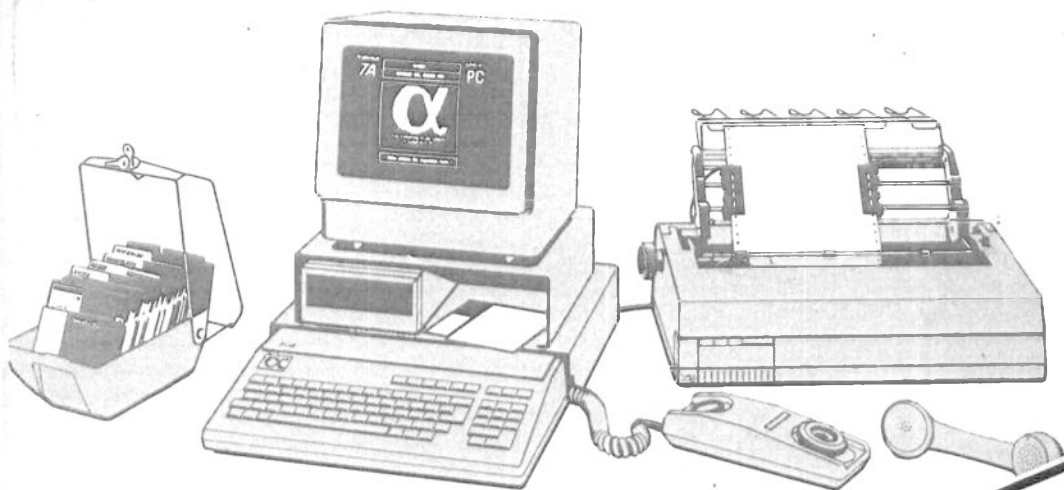
Programmierteil für

**BASIC,
TURBO-PASCAL und
Maschinensprache**

Datenübertragung

Grafik-Aufrüstung

Insider-Infos



H. Stöber

**Band 1
Programmierteil**

TA Alphatronic PC-ALMANACH '85/86, Band 1

Herausgeber: K.H. Stöber

Autoren : Norbert Griebinger, Siegfried Kerschbaum,
Peter Hagemann, Michael Titgemeyer,
Stefan Igelmann, Gerhard Müller-Dorn,
Dieter Reinhardt, Patrick V. Thomas,
Stephan Kruip, Johann Rödel-Krainz

Anmerkung: Alphatronic ist ein geschütztes Warenzeichen der
TRIUMPH-ADLER AG

Umschlaggestaltung: Venus-Werbung, Nürnberg
Satz, Gestaltung, Repro: H. Stöber
Gesamtherstellung: Druckerei Kuch, Nürnberg

Alle Rechte vorbehalten. Kein Teil dieses Werkes darf in irgendeiner Form (Druck, Fotokopie, Mikrofilm, Datenträger von Datenverarbeitungsanlagen) ohne schriftliche Genehmigung des Verlages reproduziert oder verbreitet werden.

Verkaufpreis im Fachhandel 58,-- DM - bei Direktbestellung (Verlag H. Stöber) 58,-- DM - plus Versandkosten. Vertrieb über Fachhandel und Universitätsbuchhandlung Korn & Berg, Hauptmarkt 9, 8500 Nürnberg

Copyright © Verlag Hildegard Stöber, PC-Sachbücher
Rudolf-Harbig-Str. 5, 8502 Zirndorf

ISBN 3-87432-103-7
1. Auflage 1985



Die Abbildung zeigt den Alphontronic PC in einer "typischen", praxisgerechten Ausstattung.

Angeschlossen sind:

- a) 1 Laufwerk (ausreichend für den Anfang, für den Einsatz vieler "professioneller" Softwarepakete sind jedoch zwei Laufwerke zu empfehlen, bzw. unumgänglich)
- b) Farbbildschirm (für viele Programme optisch schöner, für "Bürosoftware" jedoch nicht erforderlich)
- c) Typenrad-"Schönschrift"-Drucker (für Textverarbeitung ein Muß, bei Bedarf erweiterbar mit Endlosformular-Traktor)
- d) Akustikkoppler für Datenübertragung
- e) Metallgehäuse zur Aufnahme des Laufwerkes und als Monitorständer (verdeckt vor allem den "Kabelsalat" aller angeschlossenen Geräte).

Das microLAND[®] Patent

Deutsche Microcomputer-Programme mit eingebauter
Bedienungsanleitung und mit verständlicher »Bildschirm-Sprache«:

Überschriftenzeile

Hier oben wird Ihnen immer angezeigt, in welchem Programmteil Sie sich gerade befinden.

Aktionsfeld

Hier läuft das eigentliche Programm, d. h. hier
geben Sie Ihre Daten ein.

...zum Beispiel für

- * Die kleine Buchhaltung (E/Ü)
- * Brief- und Adreßprogramm
- * Rechnungsschreibung
- * Lagerbestandsführung
- * Hausverwaltung
- * Vereinsbuchhaltung

Hinweisfeld

Hier werden im Programm
Zusatzinformationen und
Hilfestellungen angezeigt.
Bei falschen oder unlogi-
schen Eingaben erscheint
im unteren Teil des Hinwei-
sfeldes ein Hilfsfenster und
ein Hinweis, welche Eingabe
vom Programm erwartet wird.

Hilfsfenster

Kommandozeile

Hier stehen jeweils Abfragen oder Aufforderungen,
wenn das Programm Eingaben von Ihnen erwartet.

microLAND Programme:

Neue einheitliche Bedienerführung.

Neuer einheitlicher Bildschirmaufbau. Hohe Bediener-Qualität.

Ohne Vorkenntnisse sofort einsetzbar.

microLAND-Programme gibt es für Commodore, Schneider, SVI,
TA PC/PC 16, MSX- und MS-DOS-Computer.

	Seite
Vorwort	7
Die Autoren	9
Der Umgang mit dem Alphatronic PC	13
- Installation.	13
- BASIC	19
- Arbeiten mit einem Diskettenlaufwerk.	50
Programmieren in BASIC - welche Möglichkeiten gibt es?	57
- Unterschiede zwischen den BASIC-Sprachen.	58
- Reparieren der Allocation Table	60
- Ein ROM-BASIC-Programm im Steckmodul.	71
- Adressen der BASIC-Pointer.	73
Der richtige Bildschirm	75
Probleme mit dem Drucker?	85
- Die Parallelschnittstelle	85
- Die serielle Schnittstelle.	86
- Konfigurieren von Textprogrammen.	87
Wie man die Tastaturbelegung ändert	91
Hochauflösende Grafik mit der BiCom-Nachrüstkarte	95
- Der Grafik-Bildspeicher	97
- Darstellung von Sprites (Assembler-beispielprogramm.	101
- Arbeiten mit frei definierten Zeichen (BASIC-Beispielprogramm)	105
Programmierung der BiCom-Grafik in BASIC und TURBO-PASCAL.	111
- Deklarationen in BASIC.	111
- Programm Circle - Satz des Pythagoras	113
- Freidefinierte Zeichen.	114
- Großflächige Zeichnungen.	116
- Kreisdarstellungen.	118
- Schraffuren - Balkendiagramme	122
- Prozeduren in TURBO-PASCAL.	126
- BiCom- Grafik und Textverarbeitung.	130

	Seite
- Hilbert-Kurven - Rekursion.	132
Datenübertragung mit dem Alphantronic PC	137
- Voraussetzungen	137
- Allgemeines zum Thema Mailbox	141
- Grundsätzliches zum Akustikkoppler.	143
- DATEX-P	145
Von BASIC zu TURBO-PASCAL	151
- Systemanpassung	151
- Die wichtigsten Unterschiede zw. BASIC und PASCAL	153
- Einfache Programme in PASCAL.	154
- Prozeduren richtig angewandt.	170
- Diskettenzugriff.	174
- Fehler abfangen	176
Einstieg in die Maschinensprache.	179
- Programm: Die Briefmaschine	
SW-Uhr.	245
- Dokumentiertes Programm für Übungen in Maschinensprache	
Terminal-Emulation (VT-52).	257
- Dokumentiertes Programm für Übungen in Maschinensprache	

Vorwort

Die Beschaffung, Speicherung, Verarbeitung und Aufbereitung von Informationen stellt ein Hauptproblem unserer Zeit dar.

Ohne die moderne Kommunikations- und Datentechnik mit ihren immer kürzer werdenden Innovationszyklen ist effektive menschliche Arbeit nicht mehr vorstellbar. Letztlich ist jeder in unserer Gesellschaft von den Auswirkungen der Computer- und Informationstechnik, der Computerisierung des Lebens, irgendwie betroffen.

Um mit der Entwicklung Schritt zu halten und um für den Beruf und die Zukunft gerüstet zu sein, ist permanente und schnelle Aktualisierung des Fachwissens notwendig. Das "life long learning" wird zur Devise des Computerzeitalters.

Der Intention entsprechend will dieser Almanach nicht in Konkurrenz zu konzipierten "Programmierlehrbüchern" treten; vielmehr fügt das vorliegende Jahrbuch ein Sammelsurium an Wissen und Erfahrung von PC-Spezialisten mosaikartig zusammen.

Trotz der Relevanz werden viele Einzelartikel nicht gedruckt, gleichwohl sie viele an Händler und Hersteller gestellte Fragen umfassend klären. Der Almanach soll auch zukünftig häufige Fragestellungen behandeln und mühsames Suchen ersparen.

Der vorliegende Almanach '85/86 richtet sich an alle **alphaTronic-PC-Benutzer**, die die Einsatzmöglichkeiten und Grenzen ihres Gerätes kennenlernen und dabei ihr Wissen reichlich vergrößern wollen.

Das speziell auf den **alphaTronic-PC** zugeschnittene Arbeits-/ Nachschlagebuch teilt sich in zwei Bände. Band I ist der Wissensvermittlung, Information und Lösung häufiger Anwenderfragen gewidmet. Der Software-Katalog (Band II) informiert, beschreibt und beurteilt das weite Spektrum der erhältlichen Programme. Detaillierte Leistungsbeschreibungen, Gegenüberstellungen bzw. Testberichte erleichtern eventuelle

Kaufabsichten. Wer schnell und billig "Büroprobleme" lösen will, sollte aber nicht nur den Software-Katalogteil studieren.

Viel Raum wurde im Band I dem Problemkreis **Programmierung** überlassen. Einsteiger, Kleingewerbetreibende, BASIC-Anfänger, -Köner und Freaks aber auch PASCAL- und Maschinensprache-Neulinge erhalten umfassende Einführung mit Erfahrungshinweisen. Quasi als Krönung gilt eine komplett dokumentierte Terminal-Emulation in Maschinensprache.

Ein Bild sagt oft mehr als tausend Worte. Wissenswertes über die Nachrüstung mit der hochauflösenden Pixel-Grafik von BICOM und anschauliche Programmierbeispiele zeigen den PC in einer neuen (grafischen) Dimension. Der kommerzielle Rechner wird zum Zeichner und Maler.

Wer Zugang zu Mailboxen und Datenbanken wünscht, bedarf eines Akustikkopplers. Wie "Originaldokumente" in Sekunden den Adressaten erreichen - also Daten(fern)übertragung stattfindet - wissen nur wenige PC-Benutzer. Anwendungsmöglichkeiten und wichtige Informationen über Akustikkoppler beweisen, daß Geschwindigkeit keine Hexerei ist.

Der **Peripherie**, insbesondere dem Thema Drucker, gilt ein weiterer Artikel. Die Nutzung der Schnittstellen durch verschiedenste Geräte zeigt die Vielseitigkeit des "Kleinsten von TA". Viele Insider-Informationen, die kaum ein Händler aufzubieten hat, runden den TA-alphaTronic-PC Almanach ab.

Konzipiert für etwa 400 Seiten incl. Software-Katalog wuchs das Material dank des großen Echos auf 600 Seiten an. Eine Aufbereitung des Materials in zwei Bänden wurde nötig. Dem Leser wird trotz erheblicher Druckkostensteigerung der erhöhte Umfang zum angekündigten Preis von DM 58,- zur Verfügung gestellt. Ab Auslieferung Januar '86 können beide Bände nur noch zum erhöhten Preis von DM 74,- abgegeben werden.

Wegen einer plötzlichen schweren Krankheit des Herausgebers verzögerte sich der Erscheinungstermin. Zudem kann zunächst nur der Band I des Almanachs fertiggestellt werden. Natürlich wird der Software-Katalogteil ohne Preisaufschlag an die Bezieher des ersten Bandes nachgeliefert (voraussichtlich Ende 1985).

Die Autoren

Karl-Heinz Stöber, Herausgeber,
Exportkaufmann, seit 12 Jahren im
Marketing im Produkt-Management-
Bereich bei Triumph-Adler tätig.



Siegfried Kerschbaum, Studium zum Diplom-Handelslehrer, seither im Berufsschulbereich und in der Erwachsenenbildung tätig. Schwerpunkte: Fragen der Methodik und pädagogischen Nutzung, Einsatzmöglichkeiten branchenspez. Software für Berufsschulen im Sinne eines fächerübergreifenden Unterrichts, Anwendung des Computers für den häuslichen Schülerverwaltungs- und Hinführungsprogrammieren.

Norbert Griebinger, Dipl.-Ing.(Univ.)
Nach dem Studium der Elektrotechnik in Erlangen seit Juli '83 bei der Firma Triumph-Adler als Systemanalytiker speziell für den Alphasatronic PC angestellt. Jetzt für Systemsoftware der Systeme P50 und P60 in der Abteilung Produktmanagement, Hard- und Software-Unterstützung zuständig.



Stephan Kruip, 20 Jahre alt, kam im Wahlfach Informatik am Gymnasium erstmals mit Computern in Kontakt. Seit Oktober '83 programmiert er auf dem Alphasatronic PC (BASIC, PASCAL). "Hauptberuflich" Student der Physik in Würzburg.



Dieter Reinhardt, geboren 1949 in Hannover, Dipl. Phys.. Seit 1979 als wissenschaftlicher Mitarbeiter der Universität Hannover mit der Entwicklung von meßtechnischen Systemen für die Biophysik befaßt. 1984 Mitgründer der BiCom Datensysteme GmbH, seitdem dort als Geschäftsführer tätig.

Stefan Igelmann, geboren 1962 in Osnabrück, verheiratet, Student der Sozialpädagogik. Einsatz des PC bei der Textverarbeitung im Studium, beim Bandmanagement und zur Programmierung von Lernprogrammen und Musikcomputer-Software.



Peter Hagemann, ehemaliger Hobbyelektroniker, wollte einmal das DCF-77 Zeitzeichensignal dekodieren. Als sich die ersten Teilschaltungen zu "TTL-Bergen" entwickelten, stand der Entschluß fest: Ein Computer muß her! Seitdem begann die Interessenverlagerung von der Hardware zur Software. Als erster "semiprofessioneller" Computer wurde eine TRS80-Anlage beschafft. Mittlerweile wird auf mehreren Geräten, so z.B. EPSON-, OSBORNE- und TA-Computer Software erstellt. Die Geräte dienen nun nicht mehr zur "Dekoration" des Hobbyzimmers, sondern gehören zum Inventar einer kleinen Softwarefirma.

Michael Titgemeyer, Jahrgang 1958. Nach einer Lehre als Büromaschinenmechaniker arbeitete er seit 1978 im Zentralkundendienst von Triumph-Adler in Nürnberg. Seit 1980 ist er dort in der Entwicklungsabteilung für Mikrocomputer tätig. Er arbeitet an der Implementierung der Betriebssysteme CP/M-80 und MS-DOS für die Alphantronic Mikrocomputer.



Johann Rödel. Fragen zur Person mag er nicht. Wir fanden heraus: Mathematiker (weshalb er sich selbst als Puristen und schlechten Programmierer bezeichnet). Zumindest letzteres widerlegt er mit seinem Beitrag in diesem Buch.



Patrick V. Thomas, Jahrgang 1965. Kam schon 15 Jahre später über einen PC 1211 (PC stand damals für Pocket Computer) zur Computerei. Mit der Zeit wuchsen die Ansprüche. Es folgten ein Colour Genie, ein Sinclair Spectrum und danach der TA PC. Seit Herbst 83 Studium der Informatik in München.

Gerhard Müller-Dorn, geboren 1953 in Langen, verheiratet, 2 Kinder, Dipl.-Ing. der Datentechnik. Seit 1979 zunächst angestellter Mitarbeiter in Systemhäusern, später selbständig. 1984 Mitgründer der BiCom Datensysteme GmbH, seitdem dort als Geschäftsführer tätig.

Auf Profis programmiert:



Mit mc kommen Sie jeden Monat auf neue Ideen, wenn Sie selbst programmieren oder Hardware bauen und verändern.



Durch Programme in mc werden Sie manches Problem überhaupt nicht mehr als Problem betrachten.



Nach mc-Bauanleitungen löten Sie vom einfachen Interface bis zum kompletten System, was an Hardware nur schwer zu kaufen ist.



Die Mikrocomputer-Zeitschrift

6.90 DM - 55 05 - 7 St. - September 1980

68008-Platine für Apple-II

Geknackter Macintosh

Kommunikation mit dem mc-68000-Computer

UCSD-Pascal unter MS-DOS

Erweitertes C-64-Grafikpaket



In mc-Fachaufsätzen geht's um neue Entwicklungen, um professionelle Hardware und Peripherie.



Natürlich testet mc Geräte und Programme. Die Ergebnisse werden aus der Sicht des professionellen Anwenders interpretiert.

Aktuelles aus der Branche zu Unternehmen, Produkten, Kongressen, Tagungen und Messen finden Sie jeden Monat in mc.

mc bringt Profis weiter.

Für DM 6,50 bekommen Sie mc an jeder größeren Zeitschriften-Verkaufsstelle.

Ein kostenloses Probeheft schicken wir Ihnen gerne.

Das ist Ihr Gutschein:

GUTSCHEIN für ein kostenloses Probeheft der mc

Bitte schicken Sie mir die neueste Ausgabe der mc kostenlos an meine folgende Anschrift:

Name

Beruf

Straße

PLZ/Ort

Coupon auf Postkarte kleben, mit 60 Pfennig freimachen und ab damit an

Franzis'

Franzis-Verlag, Abt. Service (PCA)
Postfach 37 01 20
8000 München 37

mc
Die Mikrocomputer-Zeitschrift

MICRO-TEXT PLUS

das professionelle Textverarbeitungs-/Adressverwaltungsprogramm für den

7A alphaTronic PC

Leistungsmerkmale:

- Steckmodul als Programmräger; für das Programm selbst wird kein Diskettenlaufwerk benötigt;
- extrem große Arbeitsgeschwindigkeit und hoher Benutzerkomfort durch komprimierte Assembler-Programmierung und Menü-Steuerung;
- kann gleichzeitig zwei verschiedene Drucker ansteuern;
- erstellt Standard CP/M-Dateien, die mit anderen CP/M-Programmen weiterbearbeitet werden können;
- Dienstprogramme für Gestaltung von Druckformularen, Konfiguration beliebiger Drucker, Datensicherung und Disketten-Initialisierung

Arbeits-Funktionen:

- Eingeben, Ändern, Löschen und Umbenennen von Texten und Textbausteinen bis zu 30 Seiten Länge;
- Zugriff auf 1150 Adressen (oder andere Daten) mit je 15 Adressfeldern;
- Kopieren, Löschen, Verschieben, Einfügen und Auslagern von Textblöcken bzw. Floskeln;
- Druck von Serienbriefen mit Adressen-Selektion; Direkt-druck-Programm;

- Suchen und Austauschen von Wörtern; Zentrieren von Zeilen;
- Blocksatz-Darstellung am Bildschirm und beim späteren Ausdrucken;
- Unterstreichung, Fettschrift, Kursivschrift u. a. Druckersteuerungen;
- Anzeige des Seitenumbruchs (im laufenden Text) und der Seitennummer;
- automatische Kopplung von Datei-Name und Formular; Hardcopy-Ausdruck;
- frei definierbare Formulare und Adress-Masken;
- Zeilenvorschub, Tabulatoren und Textänderer frei wählbar;
- Kopf- und Fußzeile; Etikettendruck; automatische Seiten-numerierung;
- einfache Bedienung durch jederzeit aufrufbares Hilfs-menü;
- Verarbeitung von Trennungsvorschlägen und Graphik-Zeichen.

Vertrieb über den TA-Fachhandel
empl. Verk.-Preis

DM 750,-



Victor Soft EDV-Beratung und Programmentwicklung

Halbmond 30 d · 2058 LAUENBURG/ELBE · Tel. 04153/2314

Neue Bücher für alphaTronic Computer aus dem *Helm*-Verlag

Heidelberger Landstr. 194 · 6100 Darmstadt 13 · Tel. 061 51/5 53 75



B-032
68,- DM



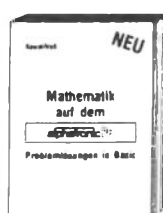
B-037
68,- DM



B-031
38,- DM



B-036
49,- DM



B-035
49,- DM



B-001
96,- DM



B-018
96,- DM



B-012
68,- DM



B-034
68,- DM



B-040
68,- DM

Der Umgang mit dem Alphonetic PC – die ersten Programmierschritte

S. Kerschbaum

Der folgende Beitrag richtet sich an blutige Anfänger bzw. Einsteiger ohne jegliche Bedienungs- und Programmierkenntnisse. Auch wer bereits durch Anwenderprogramme gewisse Vorkenntnisse hat, sollte die ersten Überlegungen überfliegen.

Sie werden nach dem Durcharbeiten der folgenden Seiten:

- den Umgang mit dem Computer und anschließbaren Zusatzgeräten lernen,
- mit fertigen Programmen umgehen können,
- einfache BASIC-Programme erstellen können,
- "Computerdeutsch"-Begriffe verstehen.

1.1 Vorbemerkungen

Mikrocomputer sind heute in fast alle Bereiche des Lebens eingedrungen. Obwohl unsere hochtechnisierte Gesellschaft auf die Computertechnologie angewiesen ist, und der Umgang mit Geräten der Elektronischen-Daten-Verarbeitung selbstverständlich sein müßte, gibt es noch immer in weiten Teilen der Bevölkerung viele Bedenken, Hemmnisse ja Aversionen.

Um die Einsatzmöglichkeiten und Grenzen Ihres alphaTronic PCs kennenzulernen, sollten Sie den Computer als Besitzer und/oder Anwender als ökonomisches und einfaches Werkzeug für Beruf, Haushalt, Hobby und Schule einsetzen bzw. testen.

1.2 Installation der Computeranlage

Bei jedem Computer ist zwischen **HARDWARE** (= alles, was man anfassen kann, Geräte, Elektronikteile) und **SOFTWARE** (=Programme) zu unterscheiden.

1.2.1 Hardware für den alphaTronic PC

Ein Computersystem (=Konfiguration) besteht in der Regel aus

mehreren einzelnen - voneinander abhängigen - Bausteinen. Die Einheiten arbeiten nach dem EVA-Prinzip zusammen (E = Eingabe - V = Verarbeitung - A = Ausgabe). Streng genommen erledigt der alphaTronic PC oder kurz PC alle EVA-Arbeiten alleine.

1.2.2 Tastaturcomputer alphaTronic PC

Das Aussehen des PCs erinnert an eine Schreibmaschine; man spricht deshalb von einem Tastaturcomputer. Der eigentliche Rechner (Fachausdruck Zentraleinheit - hier geschieht die Verarbeitung) mit der zur Eingabe, Verarbeitung und Ausgabe von Daten (z.B. Zahlen, Namen) erforderlichen Tastatur findet Platz im gleichen Gehäuse.

Neben der Schreibmaschinentastatur hat der PC separate Zusatz Tasten: den sogenannten Zehnerblock (ähnlich einem Taschenrechner Zahlen und Rechenoperationen), die Sonder- insbesondere Steuertasten und die orangefarbenen Funktions- tasten.

Im Erscheinungsbild fallen zwei übergroße Tasten mit abgeknicktem Pfeil auf. Diese wohl wichtigsten Tasten sind für die Eingabe von Daten erforderlich. Im Computer-Deutsch heißen sie RETURN-Tasten (engl. carriage return abgekürzt CR, was soviel wie Wagenrücklauf - Schalten bei der Schreibmaschine - bedeutet). Die RETURN-Taste veranlaßt den PC, die von Ihnen eingegebenen Anweisungen bzw. Informationen in den Speicher zu übernehmen. Nach jeder Eingabe muß die Eingabetaste gedrückt werden! Welche der beiden Tasten Sie wählen ist gleichgültig.

Die Tastatur bietet dem Benutzer die Möglichkeit, dem Computer Befehle, Daten oder ganze Programme zu übermitteln (einzugeben). Grundsätzlich gliedert sich die Tastatur in fünf Gruppen:

- **alphabetische Tasten**
große und kleine Buchstaben A-Z, nach Betätigen der SHIFT-Taste entstehen Großbuchstaben, Kleinbuchstaben ohne SHIFT-Taste (wie Schreibmaschine);
- **numerische Tasten**
Zahlen 0 - 9, wichtig! Vor- und Nachkommastellen werden nicht durch Dezimalkomma sondern durch Punkt getrennt z.B. nicht 5,12 sondern 5.12 ;

- **grafische Zeichentasten**

z.B. Q stellt ein Karo dar ; Darstellung durch GRAPH-Taste (gelbe Anzeigelampe leuchtet) und beliebige Taste;

- (wichtige) **Sondertasten**

CURSOR-Steuertasten (Pfeiltasten in Höhe der Leertaste)

bewegen den blinkenden Leuchtstrich in Pfeilrichtung

SHIFT - und beliebige Tasten ergibt Großbuchstaben

LOCK - erzeugt Großbuchstaben (Dauertaste!)

INS/DEL - Zeichen löschen

SHIFT + INS/DEL - Zeichen einfügen

BREAK - bricht laufendes Programm ab

TAB - Cursor springt acht Zeichen nach rechts;

- **RETURN-Taste**

oder Eingabetaste - schließt Dateneingabe ab, der Rechner übernimmt in den Speicher.

1.2.3 Sichtgerät

Ohne Sichtgerät ist Kontakt mit dem PC nur schlecht denkbar. Sowenig Sie selbst "Radiowellen" empfangen können, sind Sie in der Lage die Stromzustände im Inneren des Rechners bzw. an der "Fernsehbuchse" zu sehen. Damit der PC und seine Zusatzgeräte in Ihrem Sinne miteinander arbeiten (Sie wollen z.B. einen Brief drucken), müssen Sie ihm mitteilen, was er zu tun hat, welche Buchstaben, Abstände usw. Sie von ihm verlangen. Vergleichen Sie den Bildschirm mit dem Blatt Papier, das Sie normalerweise in eine Schreibmaschine einspannen. Erst das Sichtgerät (= Monitor) zeigt dem Benutzer - also Ihnen - wichtige Vorgänge des Computers (z. B. Fehlermeldungen, bei falscher Programmierung oder den nächsten von Ihnen verlangten Schritt in einem fertigen Programm).

Als preiswertestes Daten-Ausgabegerät empfiehlt sich der Anschluß eines handelsüblichen Fernsehempfängers. Natürlich sollten Sie ein Farbgerät wählen, da ihr PC farbliche Zeichen und Abbildungen liefert. Wenn Sie aber viel mit der im Geschäftsbereich üblichen 80-Zeichendarstellung arbeiten, ist ein TV-Gerät denkbar ungeeignet. Die TV-Auflösung der 80 Zeichen pro Zeile, wird vom menschlichen Auge als klein und dadurch anstrengend empfunden. Der Anschluß eines (Farb-)Monitors liefert randscharfe Zeichen im 80-Zeichen-Modus (siehe auch Beitrag über Bildschirme).

1.2.4 Drucker

Über Fernsehempfänger oder Monitor werden die gewünschten Daten sichtbar dargestellt. Wer ein Schriftstück mit dem PC erstellen will, braucht ein Gerät zur Dokumentation: den Drucker. Neben schnellen Nadeldruckern, die auch grafische Zeichen darstellen, sind günstige Typenradschreibmaschinen an den PC anschließbar (z.B. Gabriele 9009, Gabriele 8008, SE 310, usw.).

Schönes Schriftbild bei gleichzeitig möglicher Nutzung der Schreibmaschinenvorzüge sprechen für Typenradschreibmaschinen. Zu bedenken gilt, die 9009 bzw. 8008 benötigen eine gesonderte Schnittstellenbox (= Baustein, der zwischen verschiedenen Geräten Verbindung schafft -nicht im Preis der Schreibmaschine enthalten), Grafikzeichen werden nicht oder nur nach Typenradtausch gedruckt. Der schöne Druck bedeutet allerdings langsame Ausgabe - ca. 12 Zeichen pro Sekunde. Ob Sie jedoch den Brief direkt in eine Schreibmaschine eingegeben haben oder über ein Textverarbeitungsprogramm erstellt haben, ist absolut nicht zu erkennen. Der von Ihnen erstellte fehlerfreie Brief ist mit dem PC als quasi "denkendem" Partner keine Zukunftsmusik.

Eine weitere Alternative stellt der TRD 7020 dar, der kompatibel ist (=verträglich, d.h. er ist ohne Probleme auch an Computer anderer Hersteller anschließbar). Kleiner Preis und große Leistung (voll textverarbeitungsfähig, plottfähig, 20 Zeichen pro Sekunde, große Typenradauswahl) sprechen für eine Anschaffung dieses Typenrad-Schönschreibdruckers.

1.2.5 Speichermedium

Will man eigene Programme erstellen, gekaufte Software ablaufen lassen oder beliebige Daten aufbewahren, benötigt man eine Speichereinheit. Zu PC, Bildschirm und Drucker gesellt sich ein Tonbandkassettengerät oder eine Diskettenstation (F1 und eventuell F2).

Die billige Lösung liegt im Anschluß eines normalen Kassettenrecorders, der oft Probleme in der Übertragung bereitet. Bis man die optimale Lautstärke nach vielen Tests herausgefunden hat, ist oft schon beim ersten Kontakt mit dem Speichermedium die rechte Lust an der Computerei getrübt. Zu empfehlen ist daher ein spezifischer Datenrecorder.

Wer das oft umständliche Vor- und Zurückspulen satt hat, sollte an den Kauf einer Floppy (=Diskettenlaufwerk) denken. Trotz sekundenschnellem Zugriff auf Daten oder Programme wird höchste Sicherheit geboten.

Für den Kassettenrecorder-Betrieb sind herkömmliche Magnetbandkassetten erforderlich, die Floppy benötigt Disketten (=Magnet-scheiben - 5 1/4 Zoll s.u.).

Unsere Konfiguration besteht somit aus PC (Rechner), Bildschirm (Sichtgerät), Drucker (Dokumentationsgerät), Floppy und/oder Kassettenrecorder (Speichergerät).

1.3 Installation des Computersystems für eilige Anwender

Da wohl mindestens drei bzw. vier Stromverbraucher miteinander Daten austauschen, ist eine Mehrfachsteckdose mit integriertem Ein-/Ausschalter zweckmäßig (gleichzeitiges Ausschalten aller Geräte möglich). Hinweis: eine sogenannte "intelligente Steckdose" bietet z.B. die Firma MEZ über den Fachhandel an.

1.3.1 Fernsehanschluß

- mitgelieferten 8-poligen DIN-Stecker des TV-Adapters in RGB-Buchse stecken (seitlich links am PC),
- Antennenkabel (Koaxialkabel) mit TV-Adapter und Fernsehantenneneingang verbinden,
- TV und PC an Stromnetz anschließen - TV einschalten,
- PC durch Kipp-Schalter (seitlich rechts) einschalten - rote Kontroll-Lampe leuchtet, gleichzeitig ertönt ein "Piep",
- TV-Empfangsteil auf Empfangsbereich III (Kanal 5 oder 6) justieren und so lange suchen, bis klares Bild empfangen wird (evtl. Wahlschalter L und H des Adapters betätigen),
- Helligkeit, Kontrast bzw. Farbe optimieren.

1.3.2 Monitoranschluß

- Koaxialkabel des Monitors in BAS-Buchse stecken (seitlich links am PC),
- PC wie oben einschalten,
- Monitor einschalten (auf Betriebsbereitschaft achten!),
- Helligkeit, Kontrast bzw. Bildsynchronisation (bei TA-Monitoren auf Rückseite!) optimieren.

1.3.3 Floppyanschluß (für F1)

- Laufwerk mit Stromnetz verbinden,
- mehradriges Übertragungskabel in PC stecken (falls noch nicht geschehen Schutzmetall abschrauben),
- bei PC-Benutzung mit Laufwerk zuerst Floppy mit Kippschalter an der Rückseite einschalten (rote Kontroll-Leuchte zeigt Betriebsbereitschaft),
- DISK-BASIC-Systemdiskette vorsichtig bis zu einem spürbaren Widerstand in Laufwerk einlegen (wichtig! das Etikett liegt oben auf der Ihnen zugewandten Seite - viereckige Schreibe Schutzkerbe links (Ausnahme: keine Kerbe bei Original-DISK-BASIC-Diskette) - jede Berührung der Magnetscheibe im Bereich des Schreib-Lese-Ausschnittes (Langloch) ist zu vermeiden - Aufbewahrung in der mitgelieferten Schutzhülle, - Verschußhebel langsam in Pfeilrichtung bis zu einem hörbaren Klicken einrasten,
- Bildschirm einschalten, PC einschalten,
- rechteckige rote Betriebslampe an F1 leuchtet auf - Laufgeräusche zeigen an, daß PC und Floppy miteinander arbeiten (wichtig! solange die rechteckige Betriebslampe leuchtet, muß die Diskette im Laufwerk bleiben),
- Bildschirm fragt How many files? - auf Eingabetaste drücken,
- PC und Floppy sind betriebsbereit.

1.3.4 Floppyanschluß (für F2)

- Laufwerk F2 zur Stromversorgung mit vieradrigem, farbigem Kabel an F1 anschließen,
- Laufwerk F2 zum Datenaustausch mit mehradrigem Kabel an F1 anschließen,
- beide Laufwerke einschalten (rote Kontroll-Lampen leuchten).

1.3.5 Kassettenrecorder

- Voraussetzung an den Recorder:
Eingang für Ohrhörer oder Zweitlautsprecher und DIN-Stecker-Anschlußmöglichkeit (Mikrofoneingang)
- mitgeliefertes Kabel (drei Stecker) wie folgt einsetzen:
Steckerende mit nur einem Stecker in seitlich linke Anschlußbuchse des PCs (Symbol zweier verbundener Kreise)
Steckerende mit zwei Steckern - (kleiner) Klinkenstecker für Ohrhörerschluß, 5-poliger DIN-Stecker für Mikrofoneingang - Recorder ohne Ohrhörerbuchse könnten mit vorhandenem Zweitlautsprecher-Anschluß betrieben werden (Adapter im Fachhandel!).

1.3.6 Drucker (V.24)

- Drucker mit Stromnetz verbinden,
 - Drucker und PC durch mehradriges Kabel miteinander verbinden (sitzen Sie frontal zum PC, finden Sie Anschluß rückseitig links außen neben Anschluß für Laufwerk F1),
 - Drucker einschalten - (Papier einspannen),
 - PC einschalten,
 - Test durch LPRINT "AAA" - Eingabetaste drücken,
 - Drucker schreibt AAA
- (Hinweis: ergeben sich bei Ihnen Probleme mit diesem kleinen Test, lesen Sie den Beitrag über Drucker).

2. BASIC mit dem alphaTronic PC

Der PC verfügt über verschiedene Arten von **BASIC**. BASIC ist eine schnell erlernbare Sprache, die einen Dialog mit Ihrem PC ermöglicht (Beginners **All purpose Symbolic Instruction Code**= BASIC = symbolischer Allzweckbefehlscode für Anfänger). In der Grundausstattung (Betrieb ohne Floppy) ist der PC mit dem sog. ROM-BASIC ausgestattet. So oft Sie den Rechner auch ein-/aus-schalten, meldet sich der Festwertspeicher (Read-Only-Memory = ROM) mit seiner BASIC-Version wieder, er ist nicht löschar.

Arbeiten Sie mit einem Laufwerk F1, liest der PC das sog. **Disk-BASIC** von der mitgelieferten Systemdiskette. Wollen Sie BASIC mit dem Betriebssystem CP/M von Digital Research benutzen, benötigen Sie **MBASIC**, die dritte mögliche BASIC-Version auf Ihrem TA alphaTronic PC (Auf die Unterschiede der BASIC-Versionen wird detailliert in einem separaten Kapitel eingegangen). Jede dieser einzelnen BASIC-Versionen hat einen gleichen Grundwortschatz. Es ist daher nicht erforderlich, für jede Abart von BASIC die komplette Sprache neu zu lernen, vielmehr erweitert die jeweilige Variante die Einsatzmöglichkeiten.

Nahezu alle Personal-Computer sind heute seitens der Hersteller mit einer "Mini"-BASIC-Version ausgerüstet. Ihr PC hat im ROM ein bereits sehr ausgebautes BASIC.

2.1 Aktivieren des ROM-BASIC

Sind PC und Bildschirm aufeinander abgestimmt, wird nach dem Starten des Tastaturcomputers folgende Meldung ausgegeben:

Microsoft ROM BASIC Ver 5.11
Copyright (c) 1983 by Microsoft
28156 Bytes Free
OK

Eine Zeile tiefer blinkt links ein Markierungsstrich, der **CURSOR** (bewegliches Leuchtfeld, mit dem man Zeichen eingibt). Der Cursor kann an jede beschreibbare Stelle des Bildschirms gebracht werden. Die Steuerung des Cursors erfolgt über die Pfeiltasten neben der Leertaste. Ein Druck auf Pfeilrichtung rechts läßt den Leuchtstrich um ein Feld (=Spalte) nach rechts wandern.

Nur der praktische Umgang mit dem PC kann Ihnen schnelles Verständnis und Beherrschen bringen. Üben Sie! Dirigieren Sie zuerst den Cursor in die Mitte des Bildschirms. Steuern Sie die linke und rechte obere Ecke als Ihre nächsten Ziele an usw.

Nach dem "Start" hat Ihr PC pro Zeile 40 Spalten und 23 Zeilen. In der 24. Zeile sehen Sie sechs in negativer Schrift erscheinende Felder, die den sechs orangefarbenen **Funktionstasten** zugeordnet sind. An erster Stelle steht GOTO. Wenn Sie eine der Funktionstasten drücken, wird der zugehörige Text ab der Stelle der Cursorposition auf den Bildschirm geschrieben. Jede dieser orangefarbenen Tasten verkörpert einen BASIC-Begriff.

Sie werden sehr schnell den Vorzug einer Funktionstaste erkennen, wenn Sie selbst programmieren. Statt die erforderliche Programmanweisung mühsam Buchstabe für Buchstabe einzuhacken, genügt ein kurzer Druck auf die betreffende Taste - das komplette Wort steht da.

2.2 Arbeitsweise des alphaTronic PCs

Ohne Sie und Ihre Anweisungen ist Ihr PC strohduhm. Zum Beweis vorab eine kleine Übung:

Geben Sie folgende Aufgabe ein

5 + 2 = (Eingabe- bzw. RETURN-Taste drücken).

WICHTIG! : Um einen Befehl an den Computer weiterzugeben, ist stets die Return-Taste zu drücken - ohne RETURN geht nichts!

Was jeder billige Taschenrechner schnellstens errechnet, bereitet dem PC enormes Kopfzerbrechen - auch nach Stunden kein Ergebnis. In Wirklichkeit hat Ihr PC Ihre Anweisung schon lange ausgeführt. Geben Sie **LIST** (oder Funktionstaste 4 und Eingabetaste) ein, und Sie werden Ihr erstes "BASIC-Programm" wiederfinden $5 + 2 =$.

LIST : listet alle Programmzeilen, die sich im Speicher befinden auf - Funktion nur in Verbindung mit RETURN-Taste!

Ähnlich dem Menschen braucht jeder Computer zur Lösung von Problemen nicht nur Formeln, sondern er zwingt Sie gewisse Umgangsformen mit ihm einzuhalten. Der Ton macht die Musik - Regeln und Befehle ergeben ein Programm.

Bevor Sie weiterprogrammieren, müssen Sie wissen, daß Ihr PC prinzipiell drei Arbeitsarten unterscheidet:

DIREKT-Modus, **PROGRAMMIER-Modus** und **AUSFÜHRUNGS-Modus**.

Den **DIREKT-Modus** erkennt man an der **OK-Meldung** des Rechners. Der PC zeigt Ihnen durch diese Meldung Betriebsbereitschaft; er gibt Ihnen die Möglichkeit, Befehle einzugeben, die er sofort (direkt) ausführt. Der Befehl **LIST** (=liste alle Programmzeilen auf) wurde sofort bearbeitet - das Beispielrechenproblem hingegen nicht.

PRINT : (=schreibe/drucke aus) mit diesem Befehl lernt Ihr PC das Schreiben (und das Rechnen), Texte, Zahlenwerte oder Grafikzeichen, die nach **PRINT** stehen, werden auf den Bildschirm gebracht, **PRINT** alleine erzeugt Leerzeilen (Zeilenvorschub).

Geben Sie jetzt ein:

PRINT 5 + 2 (= Istgleich-Zeichen entfällt - Leerstelle nach **PRINT** einhalten!).

Der PC bringt das gewohnte Ergebnis (er führt Ihren Befehl aber erst aus, wenn die Eingabetaste gedrückt wurde). Halten Sie diese Regel ein, können Sie jede Grundrechenaufgabe mit Ihrem PC lösen, z.B. **PRINT 10 - 2** oder **PRINT 514 * 125**, usw.

Regelverstoß bedeutet stets eine **Fehlermeldung**. Die häufigste Meldung heißt **SYNTAX ERROR** (= Fehler in der Schreibweise). Ihr

PC ist ein sturer Lehrmeister. Jeden Fehler in seiner "Grammatik" quittiert er mit dem Abbruch der Bearbeitung Ihres Problems. Versuchen Sie z.B.:

PRIN 5 * 5 - (Sie vergaßen den Buchstaben T)

Ihr PC bringt wieder die Fehlermeldung SYNTAX ERROR und zeigt Eingabebereitschaft (OK).

Eine ein- oder mehrmalige Wiederholung unserer Beispielaufgabe läßt der Direktmodus nicht zu. Den Ergebniswert 6 hat der PC ähnlich einem Taschenrechner vergessen. Wer Programme mehrfach wiederholen will, muß im Indirekt- oder **PROGRAMMIER**-Modus arbeiten.

In BASIC setzt man vor jede Anweisung eine Zeilennummer. Unser Beispiel sieht nun folgendermaßen aus

10 PRINT 5 + 2 (Eingabetaste!).

Wichtig! : der Computer unterscheidet die Ziffer "0" (Null) und den Buchstaben O - (O mit Punkt = Null) -
gewöhnen Sie sich an, Zahlen mit dem Zehnerblock einzugeben - Falscheingabe bedeutet Fehlermeldung!

Ihre Programmzeile wird vom Speicher übernommen, d. h. der PC merkt sich diese Zeile, Sie können sie immer wieder aufrufen. Versuchen Sie **LIST**. Mit Erstaunen werden Sie zwei Programmzeilen bemerken (Zeile 5 + 2 = und 10 PRINT 5 + 2 =). Jetzt wird auch klar, warum der PC unsere Aufgabe im Direktmodus nicht bearbeitet hat. Gemäß seiner Grammatik besteht 5 + 2 = aus einer Zeilennummer 5 mit dem "Inhalt" (=Befehl) + 2 =.

Wir wollen unser kleines Programm "laufen" lassen. Dazu gibt es den Befehl **RUN** (Eingabetaste nicht vergessen! - benutzen Sie Funktionstaste 6. erübrigt sich die Eingabetaste). Der PC versteht + 2 nicht - Fehlermeldung. Sie können RUN mehrfach wiederholen, stets erhalten Sie SYNTAX ERROR. Jedesmal wenn Sie RUN eingeben, beginnt der PC ab der kleinsten Zeilennummer das Programm abzuarbeiten. Nur wenn Sie die Zeile 5 entfernen oder berichtigen, gelangt der Rechner zur Zeile 10.

Geben Sie die Zahl 5 ein (+Eingabetaste) und listen Sie das Programm (LIST). Zeile 5 ist entfernt, Ihr Programm ist jetzt lauffähig - so oft Sie wollen.

RUN : das im Speicher befindliche Programm wird beginnend mit der kleinsten Zeilennummer "gestartet".

Zeilennummer-Eingabe : die eingegebene Zeilennummer wird aus dem im Speicher befindlichen Programm gelöscht - falls eingegebene Zeilennummer nicht vorhanden, erfolgt Fehlermeldung (Undefined line number).

Sie befinden sich im **AUSFÜHRUNGS**-Modus, d.h. nach dem Befehl **RUN** sucht der PC nach einem Programm im Speicher, arbeitet es Zeile für Zeile ab und meldet sich nach Beendigung mit OK, er ist im Direkt-Modus. Während des Programmlaufs können keine weiteren Programmzeilen eingegeben werden. Ihr Ausgangsprogramm sollten Sie ergänzen mit weiteren Rechenoperationen.

Es empfiehlt sich die Zeilennummern in Zehnerschritten durchzunummerieren. Daß der PC zählen kann, erkennen Sie, wenn Sie z.B. eine Zeile 2 einfügen wollen, die er sofort an den rechten Platz setzt (Aufruf über LIST). Um sich die Mühe mit der Zeilennummervergabe zu ersparen, gibt es Funktionstaste **SHIFT + 5** (=AUTO, der PC produziert von sich aus Zeilennummern in Zehnerschritten).

Sicherlich ist Ihnen aufgefallen, daß die Bildschirmaufzeichnung von oben nach unten erfolgt. Je mehr Sie üben, desto "unübersichtlicher" wird Ihr Bildschirm. Sie sollten ihn säubern. Der **BASIC**-Befehl dazu heißt **CLS**.

CLS : clear screen (=Bildschirm löschen), alle Zeichen auf dem Bildschirm werden entfernt und der Cursor springt an den Bildanfang links oben - den selben Effekt erzielen Sie mit der Tastenkombination **SHIFT** und **CLEAR HOME** (gleichzeitig drücken!).

Wollen Sie Ihr Programm "verschwindenlassen", den Speicher gewissermaßen aufräumen bzw. löschen, brauchen Sie nicht jede Programmzeilen-Nummer einzugeben - ein Befehl genügt **NEW**.

NEW : der Programmspeicher wird gelöscht, d.h. nach Eingabe dieses Befehls ist Ihr Programm für immer verschwunden - vorher sichern (siehe unten).

Fassen wir kurz zusammen:

mit dem **CURSOR** kann die Stelle jedes Vertippens erreicht werden, einfaches Überschreiben durch das richtige Zeichen;

Funktionstasten erleichtern die Eingabe der Befehle und Programme;

jeden **Verstoß gegen die SYNTAX** ("BASIC-Grammatik") quittiert der PC mit Fehlermeldung - das Programm wird beendet;

im **Direkt-Modus** arbeitet der PC quasi als Taschenrechner;

im **Programmier-Modus** erlaubt er zeilenweises Eingeben von Programmschritten bzw. Befehlen;

im **Ausführungsmodus** führt er fertige Programme aus;

wichtige **BASIC-Vokabeln**

RUN : Programm wird gestartet

LIST : Programmzeilen werden aufgelistet

PRINT : Zeichen werden auf dem Bildschirm ausgedruckt

NEW : Inhalt des Programmspeichers wird gelöscht

CLS : Bildschirm wird gelöscht.

2.3 Programmieren mit dem alphaTronic PC in ROM-BASIC

Ebenso wie ein Sachbearbeiter Arbeitsanweisungen und Informationen benötigt, um gestellte Probleme zu lösen, braucht Ihr PC fortlaufende Anweisungen für ein Programm.

Da Ihr PC keinerlei Verstand besitzt, muß ihm jeder Handgriff vorher genau in einer ihm verständlichen Sprache vorgeschrieben werden - z.B. in ROM-BASIC.

Schreiben Sie Programme für sich, müssen Sie beachten, daß Ihr PC auf den richtigen Wortschatz, fehlerlose Anweisungen und logisch aufeinanderfolgende (eindeutige) Handlungsschritte angewiesen ist.

Ein **Befehl** oder eine Anweisung ist eine in einer Programmiersprache abgefaßte Arbeitsvorschrift. Sehr häufig verwendet man das Wort **Statement** als Synonymbegriff.

Jeder Befehl besteht aus:

Operationsteil (Befehlswort) und einem Operandenteil (Parameter z.B. Rechenaufgabe $5 + 2$). Der Operationsteil sagt dem PC, was zu tun ist (PRINT = drucke), der Operandenteil befiehlt, womit etwas zu tun ist (addiere 5 und 2).

Damit ist eine generelle Regelung für Programmzeilen in BASIC gefunden

10	PRINT	5 + 2
Zeilennummer	Operationsteil	Operandenteil
	B A S I C - A N W E I S U N G	

2.3.1 PRINT-Befehl

Fast jeder Computer-Fan mußte einmal zuerst "kleine Brötchen backen". Er begann das Computern in kleinsten Schritten zu lernen - so wie Sie.

Ein erstes Erfolgserlebnis löst der in Zukunft von Ihnen sehr häufig angewandte PRINT-Befehl aus. Er ermöglicht gespeicherte bzw. berechnete Werte (Variablen, Konstanten) auf dem Bildschirm auszugeben.

Der folgende kleine Exkurs erläutert, was in dem Zusammenhang unter Konstanten und Variablen zu verstehen ist. Die Konstante ist ein für ein gesamtes Programm festgelegter Wert, d.h. während des Programmlaufs ändert sich z.B. ein Text oder der Wert einer Zahl nicht. Im Computerdeutsch spricht man von numerischen (z.B. $A = 5$) oder von alphanumerischen Konstanten z.B. ($A\$ = \text{"BASIC-Kurs 5"}$). Es sind also Zahlen-, Text- und "Mischkonstanten" möglich. Zu beachten ist, daß Buchstaben oder Buchstaben und Zahlen ($A=5$ oder $A1=2$) sowie Buchstaben zusammen mit dem \$-Zeichen kombiniert werden können ($A\$ = \text{"PC"}$ - $A\$1 = \text{"PC"}$ ist falsch - $A1\$ = \text{"PC"}$ hingegen richtig). Texte (Zeichenketten = strings) stehen zwischen Anführungszeichen!

Variablen können im Laufe eines Programmes mehrfach ihren Wert bzw. Inhalt wechseln. Zu unterscheiden sind Name der Variablen (z.B. KUNDE\$) und der (momentane) Wert (z.B. Auer). Die ganze Variable heißt: KUNDE\$=Auer. Schon beim nächsten Programmlauf kann sich durch Eingabe oder Dateneinspeisung per Diskette ein neuer Kunde unter KUNDE\$ "verstecken".

Stellen Sie sich eine Variable als Schublade in Ihrem Rechner vor. Um die richtige Schublade wieder zu finden, geben Sie Ihr einen Namen (z.B. KUNDE\$). Das \$-Zeichen ist Bedingung, da Sie mit Hilfe der Tastatur Buchstaben in die Lade eingeben wollen z.B. Namen also Informationen. Für Zahlenwerte genügt ein Buchstabe als Variablenbezeichnung (z.B. A als Schublade für die lfd. Nummer). Heißt der Kunde Nummer 10 z.B. Huber, legt der PC unter KUNDE\$ nach der Eingabe den Wert Huber bzw. unter der Lade A den Wert 10 ab. Ersetzen Sie Huber durch Auer, wirft der Rechner Huber aus der Lade und legt Auer ab usw.

Die Namensgebung für Variablen und/oder Konstanten ist gleich; sie wird vom Programmierer -also Ihnen- festgelegt. Zu beachten ist, daß der Wert rechts vom Gleichheitszeichen in der Variablen links vom Gleichheitszeichen abgelegt wird. Beispiel:

SUMME = SUMME + 5

Der Wert der Variablen SUMME (Ladeninhalt) wird vom Rechner um 5 erhöht. Ist SUMME z.B. 0 ergibt sich für SUMME nach der Berechnung ein Wert von 5. Wiederholen Sie den Vorgang, lautet das Ergebnis 10, da in der Lade SUMME 5 abgelegt ist. Der PC holt sich also den alten Wert, sieht nach, was gespeichert ist, addiert 5 dazu und legt nun 10 in der Lade SUMME ab. Welcher Wert liegt wohl beim nächsten Durchlauf vor?

So ganz nebenbei haben Sie die LET-Anweisung gelernt.

LET : weist Variablen und/oder Konstanten Werte zu,
LET A=5 ist identisch mit A=5 - das eigentliche LET kann also entfallen (SUMME=SUMME+5 = LET SUMME=SUMME+5).

Welche praktischen Möglichkeiten der PRINT-Befehl bietet, sehen Sie in der folgenden Übersicht:

konstante Zahlen	10 PRINT 510
variable Zahlen	20 PRINT Z (noch ohne Wert = 0)
Rechenausdrücke	30 PRINT A * B
Zeichenketten (Strings)	40 PRINT "Name:"
Stringvariablen	50 PRINT A\$ (noch ohne Wert).

Natürlich sind die obigen Möglichkeiten beliebig kombinierbar. Versuchen Sie folgendes Beispiel:

10 PRINT "10 * 3 ergibt"10*3

Denken Sie daran, wenn Sie einen Text ausdrucken wollen, muß dieser in Hochkommata (Anführungszeichen) stehen. Zwischen PRINT und der Variablen, Konstanten bzw. dem Hochkomma muß eine Leerstelle eingegeben werden.

Auch daß Leerstellen und Leerzeilen den Bildschirm gestalten, verdanken wir PRINT. Eine Überschrift in die Mitte gesetzt, sieht nun einmal besser aus als gleich an den Zeilenanfang gedruckt z.B.

```
10 PRINT ".....Überschrift":PRINT (. symbolisiert  
                                      Leerstelle = blank)
```

Mit dem PC können mehrere Anweisungen in eine Zeile geschrieben werden, Trennung erfolgt durch Doppelpunkt. Ob Sie PRINT buchstabenweise im Groß- oder Kleinschreibmodus eintippen ist egal. Der Computer setzt Kleinbuchstaben bei Befehlen und Variablen automatisch in Großbuchstaben um.

2.3.2 Formatierte Ausgabe mit PRINT

Um Ausdrücke z.B. Zahlenwerte voneinander zu trennen, verwendet man PRINT in Verbindung mit Semikolon (;) und/oder Komma (,). Werden Strings (Zeichenketten, Texte) oder Stringvariablen (Textvariablen z.B. A\$="Otto") durch Semikolon getrennt, druckt sie der PC ohne Zwischenraum nacheinander aus z.B.:

```
10 A=5  
20 A$="Ihr":B$="Name":C$="lautet:"  
30 PRINT A$;B$;C$;A
```

Der Ausdruck auf dem Bildschirm hierzu:

IhrNameIautet: 5

Die Zeichenketten werden ohne Abstand gedruckt. Da die Zahl 5 für den PC ein positives Vorzeichen (+) hat, das er nicht ausdruckt (negative Vorzeichen werden vor der Zahl ausgegeben), entsteht eine Leerstelle nach dem Text. Auch nach der Zahl setzt der PC eine Leerstelle, was Sie leicht nachprüfen können, wenn Sie die Zeile 30 abändern:

```
30 PRINT A$;B$;C$;A;A$
```


Ersetzt man Semikolon durch Komma, ergibt sich ein vollkommen anderer Ausdruck! Ändern Sie Ihr Beispiel entsprechend ab. Zeile nach der Modifizierung lautet:

```
30 PRINT A$,B$,C$,A
```

Am schnellsten ändern Sie, wenn Sie sich mit LIST das komplette Programm aufzeigen lassen, den Cursor in die Zeile 30 bringen, dann nach rechts unter das 1. Semikolon setzen und durch das Komma überschreiben. Die anderen Zeichen ersetzen Sie in gleicher Weise. (Die Änderungen werden nur übernommen, wenn jeweils mit RETURN abgeschlossen wurde.)

Nach einem Programmlauf erhalten Sie folgenden Ausdruck auf dem Bildschirm:

```
01234567890123456 (Spalten)
Ihr               Name
lautet:           5
```

Zeichen bzw. Texte werden nach einer festgelegten Einteilung der Zeilen (Druckzonen) ausgegeben. Der erste Textteil beginnt in der ersten Spalte (= Spalte 0), der zweite Text jedoch in Spalte 14. Die Zahl 5 kommt eine Stelle weiter zu stehen (eigentlich Spalte 14 da aber + - Vorzeichen Spalte 15).

Beachten Sie: Nach jedem abgeschlossenen PRINT-Befehl springt der Cursor in die nächste Zeile (Zeilenvorschub).

Wer viel mit Kalkulationen arbeitet, liebt natürlich Tabellen, konstante Abstände zwischen den Zahlen, kurz einen sauberen Ausdruck. Das bedeutet, die Dezimalpunkte stehen exakt untereinander. Das Wissen um diese Spaltendruckweise Ihres PC wird später noch einmal anzusprechen sein (siehe unten PRINT-USING). Mit PRINT ist stets nur eine Ausgabe möglich. Verarbeitung von Daten hat aber zum Ziel, aus dem Dialog mit dem Computer aus vorhandenen Daten neue Daten zu gewinnen. Unser Befehlsvorrat muß ergänzt werden um eine Eingabeanweisung.

2.3.3 INPUT-Befehl

In der Regel sind die von Ihrem PC zu verarbeitenden Daten erst zum Zeitpunkt des Programmlaufs bekannt. Wer eine gleiche Aufgabe mehrfach wiederholend zu lösen hat z.B. Berechnung von Zinsen bei vorgegebenen Werten, verwendet für jeden Rechengang dieselbe Formel, aber stets mit neuen Werten.

Die INPUT-Anweisung erlaubt Daten während des laufenden Programms über die Tastatur einzugeben. Die bereitgestellten Daten werden schließlich entsprechend Ihrer Anweisung verarbeitet.

Jeder INPUT-Befehl stoppt das Programm. Auf dem Bildschirm erscheint ein Fragezeichen, der Computer wartet auf Eingabe von Daten. Mit der Dateneingabe werden den zugeordneten Variablen gleichzeitig die Eingabewerte zugewiesen.

Allgemein hat die INPUT-Anweisung folgendes Format:

10	INPUT	A	oder
10	INPUT "Texterklärung";	A	
Zeilennummer	Schlüsselwort	Variable	

Zum besseren Verständnis sollten Sie folgendes Programmbeispiel (Listing) eingeben. Löschen Sie zunächst den Programmspeicher mit NEW.

<u>Listing</u>	<u>Kommentar</u>
10 PRINT ".....Flächenberechnung"	(. = Leerstelle)
20 PRINT:PRINT	(Zwei Leerzeilen)
30 INPUT L	(Längeneingabe)
40 INPUT B	(Breiteneingabe)
50 PRINT	(Leerzeile)
60 F=L*B	(Formel)
70 PRINT "Die Fläche beträgt"F	(Ergebnisausdruck)

Nachdem Sie das Programm mit RUN gestartet haben, taucht nach der Überschrift und zwei Leerzeilen ein Fragezeichen auf. Der PC wartet auf Zahleneingabe - solange, bis Sie Daten eingeben. Der INPUT-Befehl erfüllt somit eine Warteaufgabe. Geben Sie 25 ein, weist der Rechner der Variablen L den Wert 25 zu - Zuweisungsaufgabe. Der PC arbeitet nun die folgende Programmzeile 40 ab. Das nächste Fragezeichen erinnert Sie, über die Tastatur Zahlen einzugeben z.B. 50 - in die Lade B wird 50 abgelegt. Eine Leerzeile, das Ergebnis wird ausgedruckt, der Computer ist im Direkt-Modus (OK-Meldung). Natürlich können Sie das Programm neu starten und neue Zahlen eingeben. Um die Zuweisungsaufgabe der INPUT-Anweisung nachzuvollziehen, sollten Sie jedoch zuerst die augenblicklichen Werte der Laden L, B und F untersuchen

PRINT L;B;F ergibt die zuletzt eingegebenen Zahlenwerte.

Zeile 30 und 40 hätten vereinfacht werden können, da INPUT erlaubt, mehrere Werte gleichzeitig einzugeben, wenn die Variablen durch Komma getrennt werden (z.B. INPUT L,B). Halten Sie die INPUT-Regeln nicht ein, geben Sie z.B. bei INPUT L,B nur einen Wert ein, meldet der Rechner einen Fehler:

?Redo from start

Selbst wenn Sie vergessen, die Werte durch Komma voneinander zu trennen, taucht die obige Fehlermeldung auf. Gleichzeitig erwartet der PC neue Daten.

Um das kleine Programm wiederverwenden zu können, sollten Sie einige Verbesserungen vornehmen. Vor jedem Programmlauf sollte der Bildschirm gelöscht werden (Zeile 5 CLS). Äußerst störend sind die zwei Fragezeichen nacheinander. Derartige Programmierung zeugt von schlechter Bedienerführung - oder wissen Sie nach einem Monat noch, welcher Wert zuerst eingegeben wird? Ergänzen Sie die Zeilen 30 und 40 mit Texten z.B.:

```
30 INPUT "Länge :";L
```

Ungünstig kann auch das Fragezeichen empfunden werden. Wird das Semikolon vor der Variablen L durch ein Komma ersetzt, wird das Fragezeichen unterdrückt, d.h. es taucht auf dem Bildschirm nicht mehr auf (30 INPUT "Länge :",L).

2.3.4 Rechenoperationen in BASIC

In Ihrem Flächenprogramm tauchte erstmals eine Rechenoperation in einem Programm auf. In BASIC gelten für die Bestimmung eines Rechenausdrucks die üblichen mathematischen Regeln (z.B. Punkt- vor Strichrechnung bzw. Hochrechnung vor Punktrechnung).

Beispielaufgabe : PRINT 5 * 7 - 2 + 5 ^ 2 / 2.5

Zu beachten ist die besondere Darstellung bestimmter Rechenzeichen. Statt : ("Geteilt durch") ist / (Schrägstrich) zu setzen. Potenziert wird mit ^ ("Dach"). Der Malpunkt aus der Multiplikation wird ein Stern (*). Aus Komma wird Punkt (Bsp.: 2.5).

In der Beispielaufgabe berechnet der PC zunächst die Hochrechnung (5^2). Er multipliziert ($5 * 7$) und dividiert 25 durch 2.5. Schließlich führt er die Strichrechnung $35 - 2 + 10$ aus.

Sie sollten auch den Umgang mit Klammern üben. Testen Sie z.B.

PRINT 8 + 2 * 6 - 3 bzw. PRINT (8 + 2) * (6 - 3)

Fassen wir kurz zusammen:

PRINT : druckt den rechts vom Befehl stehenden Ausdruck z.B. Variablen (Konstanten), Zeichenketten auf Bildschirm;
LET : weist einer Variablen Wert zu bzw. ermittelt den Wert eines rechts vom Zuweisungszeichen stehenden Ausdrucks
INPUT : wartet, bis Eingabe erfolgt und weist Eingabewert zu.

2.3.5 IF - Anweisung

Sie können mit dem Computer Rechenoperationen durchführen, sich mit ihm "verständigen". Da Sie bisher nur elementare Bausteine der Sprache BASIC kennenlernten, wissen Sie, daß Ihr PC wiederkehrende Aufgaben mit hoher Geschwindigkeit löst. Daß er Ihnen aber eine viel wichtigere Aufgabe abnimmt, nämlich die der Entscheidung bzw. Entscheidungsaufbereitung, indem er Bedingungen logisch prüft, soll am Ende dieses Teilkapitels klar sein.

Entscheidungen setzen Alternativen und Vergleiche voraus. Mit **IF - THEN** wird dem Rechner befohlen, den in Ihrer Verzweigungsbedingung stehenden Vergleich durchzuführen z.B. ist eine Person 65 Jahre alt (J/N). Ist die Bedingung erfüllt, hat der Computer z.B. alle Personen zu speichern oder auszudrucken. Zum besseren Verständnis dient ein Beispiel:

Bestellt ein Kunde mehr als 50 Stück eines Artikels, erhält er 20 % Mengenrabatt. Schreiben Sie ein Programm.

Listing

Kommentar

10 CLS	(Bildschirmlöschung)
20 PRINT "Rabattermittlung "	(Überschrift)
30 PRINT: PRINT	(2 Leerzeilen)
40 INPUT "Geben Sie Bestellmenge ein: ",M	(Werteingabe für M)
50 IF M > 50 THEN 70	(Bedingungsprüfung)
60 GOTO 10	(Sprung zu Zeile 10)
70 PRINT "Kunde erhält 20 % Rabatt!"	(Kunde hat Bedingung erfüllt, Ausdruck)

Die Überprüfung, ob der Kunde Rabatt erhält oder nicht, erfolgt in Zeile 50. Wenn (IF) der Wert für die Bestellmenge (M) größer als 50 ist, dann (THEN) muß der Rechner in die Zeile 70 springen. Hier wird er angewiesen auszudrucken. Falls für M ein Wert kleiner 50 oder 50 eingegeben wird, ist die Bedingung nicht erfüllt, der PC arbeitet die nächste Programmzeile ab - also Zeile 60. Hier taucht ein neuer BASIC-Befehl auf GOTO.

GOTO : "gehe nach"; der Rechner muß unbedingt zu der angegebenen Zeilennummer springen (verzweigen) und dort mit dem Programm fortfahren.

Im Programmlisting verzweigt der PC bei $M = 50$ zum Anfang, eine Neueingabe ist möglich. Ist der Wert größer 50, ist das Programm beendet (eventuell 80 GOTO 10 ergänzen!).

Eine Verzweigung erfolgt durch Vergleich von Daten, wozu Vergleichsoperatoren erforderlich sind. In BASIC stehen folgende Vergleichsoperatoren zur Verfügung:

<u>BASIC-Symbol</u>	<u>Beispiel</u>	<u>Bedeutung</u>
>	$M > 50$ (52)	größer als
<	$M < 50$ (48)	kleiner als
=	$M = 50$ (50)	gleich
> =	$M \geq 50$ (53)	größer als oder gleich
< =	$M \leq 50$ (42)	kleiner als oder gleich
<> oder ><	$M \neq 50$ (0)	ungleich

Natürlich ist über Verzweigungen auch die sogenannte Menutechnik möglich. Schreiben Sie ein Programm, das folgenden Bildschirm Ausdruck hat.

Überprüfen Sie sich im Kopfrechnen!

Addition	(1)
Subtraktion	(2)
Multiplikation	(3)
Division	(4)

Was möchten Sie üben?

Ähnlich einer Speisekarte wählen Sie aus vier Alternativen Ihren Wunsch. Wenn Sie eine Zahl von 1 bis 4 eingeben, stellt der PC Ihren Menüwunsch parat. Was er Sie nun abzufragen hat,

muß ihm aber vorher eingegeben sein. Die ersten sieben Zeilen werden mit PRINT gelöst (auch Leerzeilen). Die Abfrage erfolgt über INPUT "Was ..."; A - die Verzweigungen könnten Sie z.B. so gestalten:

```
100 IF A = 1 then 200
200 IF A = 2 then 300          usw.
```

In den Zeilen 200 bzw. 300 könnte geprüft werden

```
200 INPUT "Wieviel ist 35 + 65";L
```

Nach Ihrer Lösung muß der PC natürlich wieder eine (interne) Abfrage lösen. IF L = 100 THEN PRINT "Richtig!" - (hinter THEN könnte auch ein Text stehen). Überlegen Sie, was im Falle einer falschen Lösung zu programmieren wäre - ein Sprung zum Anfang sollte sich anschließen. Die komplette Lösung der kleinen Aufgabe finden Sie am Ende dieses Beitrags.

Sicher haben Sie bemerkt, daß GOTO ein Dauerprogramm ermöglicht; aus dem Teufelskreis kommen Sie nur heraus, wenn Sie das Programm mit der BREAK-Taste stoppen (Ausschalten des Gerätes sollten Sie meiden). Versuchen Sie z.B.:

```
10 PRINT "alphaTronic PC"
20 GOTO 10
```

Die Endlosschleife ist perfekt. Der Rechner springt zwischen Zeile 10 und 20 hin und her, der Bildschirm wird voll mit alphaTronic PC. Wer es leid ist, Kurzprogramme immer wieder mit RUN starten zu müssen, baut einfach ein GOTO mit Zeilennummer für die Anfangsadresse ein, das Problem ist beseitigt.

Wer tiefer in die Verzweigungstechnik einsteigen möchte, lernt bald Kombinationen mit IF - THEN kennen. Möglichkeiten wie AND OR und NOT erweitern ebenso wie IF - THEN - ELSE. Bei AND müssen beide Bedingungen, bei OR mindestens eine der beiden Bedingungen erfüllt sein. NOT verändert einen logischen Ausdruck in einer IF-Anweisung, d.h. es kehrt den Wert des Ausdrucks um.

Sie sehen, es handelt sich um recht komplizierte Bedingungen, weswegen einige Beispiele folgen:

<u>Verzweigungsbedingung</u>	<u>(R)ichtig/(F)alsch für</u>
wenn R größer 4 und R kleiner 10 IF R > 4 AND R < 10	R 5, 6, 7, 8, 9 F 3, 4, 10, 11, 12
wenn R kleiner 4 oder R größer 10 IF R < 4 OR R > 10	R 3, 11, 12 F 4, 5, 6, 7, 8, 9, 10
wenn C nicht gleich 3 drucke XX IF C <> 3 THEN PRINT "XX"	Bedingung ist wahr, da C nicht ungleich 3

Ein schweres Pensum, das der Anfänger nicht allzuoft braucht. Für IF - THEN - ELSE gilt diese Aussage nicht.

Stellen Sie sich ein Programmlisting mit 1000 Zeilen vor. Da man mit GOTO jede Programmzeile schnell und zielsicher erreicht, wurde der Befehl vierzigmal eingesetzt. Ein lustiges "Herumgehupfe" des PCs ist die Folge. Wehe, wenn Sie auch nur eine Zeile ändern müssen. Keine Sprungadresse stimmt mehr je mehr Sie ändern. Einen Ausweg bietet IF - THEN - ELSE.

IF - THEN - ELSE : ist die Bedingung erfüllt, wird die hinter THEN stehende Anweisung ausgeführt, ist die Bedingung nicht erfüllt, wird der Befehl hinter ELSE ausgeführt.

Ein Beispiel bringt auch hier Verständnis:

```
100 IF A=5 THEN SCHALTER=1 ELSE SCHALTER=0
300 IF A$="J" THEN PRINT "JA" ELSE PRINT "NEIN"
```

Wenn A den Wert 5 hat bzw. diese Zahl eingegeben wurde, dann hat die Variable SCHALTER den Wert 1 sonst den Wert 0. Geben Sie in Zeile 300 A\$=J ein, dann erscheint auf dem Bildschirm JA ansonsten NEIN.

2.3.6 REMARK (REM)

Zum Entspannen soll an dieser Stelle ein einfacher Befehl besprochen werden. Zwar wird er vom Rechner ignoriert, d.h. auf dem Bildschirm geschieht absolut nichts, jedoch hilft er Programme besser zu verstehen.

REM : (remark = Bemerkung); erläuternde Kommentare dienen der Beschreibung eines Programms; REM kann sowohl in einer eigenen Zeile stehen als auch hinter anderen Anweisungen (letzte Anweisung in der Zeile!); ersatzweise kann das Zeichen `*` stehen.

Beispiele:

```

10 REM *****
20 REM * Testprogramm                Autor:
30 REM *****
40 LET A = 5                        : * LET kann entfallen
50 *                               : Addition zweier Zahlen
60 SUMME = ZAHL1 + ZAHL2           : REM oder S=Z1+Z2
70 * ++++++ Sprung in Unterprogramm A ++++++

```

2.3.7 FOR ... NEXT - Anweisung

Mittlerweile werden Sie Ihre Schwellenangst in Sachen Computer abgebaut haben. Sie wissen, wenn Sie etwas durch falsche Bedienung zerstören, dann ist es in der Regel Ihr Programm. Ihr PC ist stabil gebaut und clever. Sprachfehler meldet er. Denkfehler führt er beliebig häufig aus - solange, bis Sie es "spannen". Probieren geht über Studieren, das gilt auch für die Computerei.

Wie schnell können Sie die Summe der Zahlen von 1 bis 500 ermitteln? Hand aufs Herz, nach 1 Minute sind Sie noch nicht sehr weit. Ihr PC schafft das in ca. 3 Sekunden! Dazu braucht er allerdings eine Schleife (= Reihe von Anweisungen, die wiederholt auszuführen sind), die er bis zur Zahl 500 durchläuft. Zwar läßt sich das obige Vorhaben auch mit IF - THEN -Programmierung erreichen; einfacher, übersichtlicher und schneller ist die Lösung mit FOR ... NEXT Schleife.

FOR ... NEXT : zählergesteuerte Schleife; FOR markiert Anfang
NEXT schließt zu wiederholenden Teil ab (Ende)

Allgemein hat die FOR ... NEXT Schleife das Format:

FOR	I	=	A	TO	E	:	NEXT	I
Schlüsselwort	Laufvariable		Anfangswert	Schlüsselwort	Endwert		Schlüsselwort	Laufvariable

Das Programm für die Addition der Zahlen 1 bis 500 sieht damit etwa so aus (eigentliches Programm):

Listing

Kommentar

30 FOR I=1 TO 500	Laufvariable I beginnt bei Zahl 1
40 SUMME=SUMME + I	Variable SUMME = 0, nimmt beim 1.
50 NEXT I	Durchlauf Wert 1 an, nächste Zahl
60 PRINT SUMME	wenn I=500, drucke SUMME aus

Der Computer muß also zuerst seine Aufgabe in den Zeilennummern 30 bis 50 abarbeiten, bis er in Zeile 50 ausdrückt. Beim ersten Durchlauf hat die Laufvariable I den Wert 1, beim zweiten 2 usw. bis sie schließlich den Wert 500 erreicht. Bei jedem Durchlauf nimmt die Variable SUMME den Wert der Laufvariablen I zum bestehenden Wert hinzu (addiert). Erst nachdem die Schleife beendet ist, druckt der PC das Ergebnis aus.

Am Beispiel für die ersten beiden Werte soll die Aufgabe des PCs verfolgt werden. Nachdem er auf Zeile 30 stößt, erkennt er eine FOR-Schleife und sucht sofort nach dem Ende der Schleife dem NEXT. Jetzt weist er der Variablen I den Wert 1 zu, da I ja mit 1 beginnen soll (1 bis 500).

In Zeile 40 hat der Rechner der Variablen SUMME, die noch den Wert 0 hat, den Wert für I (derzeit 1) zu addieren (Wert nach 1. Durchlauf Zeile 40 = 1). Zeile 50 weist den PC an, das nächste I aufzusuchen. Da I den Endwert 500 noch nicht erreicht hat, springt er wieder in Zeile 30. Für I nimmt er jetzt den nächsten Wert (I=2), addiert zur momentanen SUMME=1 den Wert 2 dazu (SUMME=3) und gelangt in Zeile 50. Das "Spiel" geht solange, bis I=500; erst dann ist die FOR-NEXT-Schleife durchlaufen, die nächste Programmzeile kann bearbeitet werden.

Wollen Sie eine andere Schrittweite als 1 - der PC hat ja I stets um eine Zahl erhöht (1,2,...500) - ergänzen Sie

FOR I=1 TO 500 STEP 10 (negatives STEP z.B. STEP -2)

STEP : bestimmt die Schrittweite in einer FOR ... NEXT Schleife; sowohl negative als auch positive Werte sind zulässig; fehlt STEP gilt Schrittweite + 1 .

Um Fehlermeldungen zu vermeiden, müssen Sie auf gleiche Laufvariable(n) achten, sie kann bei NEXT entfallen (NEXT I = NEXT). Die FOR .. NEXT-Anweisung kann nur komplett eingesetzt werden. Kein FOR ohne NEXT und umgekehrt!

Wenn Sie einen Punkt über den Bildschirm laufen lassen wollen, brauchen Sie FOR .. NEXT in Verbindung mit PRINT. Versuchen Sie z.B. das kleine Programm und schon kennen Sie ein wesentliches Element der bekannten "Abschuß-Jagdspiele":

```
10 FOR Z=0 TO 80: PRINT ". ";PRINT CHR$(29);PRINT " ";:NEXT
```

Kommentar: Eine FOR .. NEXT Schleife kann in eine Zeile gepackt werden. Erhöhen Sie den Endwert entsprechend (jetzt 80!), läuft der Punkt über den ganzen Schirm. Der Punkt ist durch jedes Zeichen ersetzbar (Taste GRAPH).

Wichtig sind die gesetzten Semikola, da PRINT ja stets einen Zeilenvorschub erzeugt (unerwünschter Sprung in nächste Zeile). Neu ist PRINT mit CHR\$(29), was bedeutet, daß der Cursor um eine Stelle nach links springen soll- für rechts CHR\$(28). Dadurch wird erreicht, daß der zuvor gesetzte Punkt in Verbindung mit PRINT " " (=Stelle löschen -nichts drucken) verschwindet, und der neue Punkt eine Stelle weiter (neben dem ursprünglichen Punkt) auftaucht.

Man meint, der Punkt wandert, dabei wird er nur gelöscht und neu gedruckt.

Mehrere Schleifen können auch ineinander verschachtelt werden, wobei die zuletzt eröffnete Schleife wieder als erste geschlossen werden muß. Beispiel Zeitsimulation (digital):

```
FOR Z=1 TO 60: FOR I=1 TO 50:LOCATE 15,12,0:PRINT Z;:NEXT: NEXT
```

Kommentar: Etwa in der Mitte des Bildschirms werden die Zahlen 1 bis 60 cirka im Sekundenabstand angezeigt. Für die Zahlen ist die erste Schleife verantwortlich (Laufvariable Z). Damit die Sekundenabstände in etwa eingehalten werden, muß eine zweite Schleife, eine Zeitschleife, eingebaut werden. Sie beschäftigt den Rechner nur, da er je Durchgang bis 50 zu zählen hat.

Um zu erreichen, daß die Zahlen quasi an einem Platz übereinander geschrieben werden (Digitaleffekt), benutzen wir den neuen BASIC-Befehl LOCATE.

LOCATE : ermöglicht den Cursor an eine genau bezeichnete Stelle des Bildschirms zu dirigieren; drei durch Komma getrennte Zahlen geben an Spalte, Zeile, Cursor Aus/EIN
z.B. LOCATE 15,12,0 Spalte 15, Zeile 12, Cursor Aus.

Wenn Sie das Lauf-Punkt-Beispiel eingegeben haben, suchen Sie jetzt sicherlich verzweifelt den Cursor.

Wichtig: LOCATE 12,15,1 schaltet den Cursor wieder ein (Spalten- und Zeilenwert sind beliebig).

Ebenfalls wichtig ist die Tatsache, daß Schleifen fast beliebig tief verschachtelt werden können. Eine Überlappung darf es aber nicht geben, d.h. zwei Schleifen dürfen sich nicht überschneiden. Außerdem kann der Rechner auch nicht von einem Punkt innerhalb der äußeren Schleife zu einem Punkt innerhalb der inneren Schleife springen.

2.3.8 GOSUB - RETURN

Immer wiederkehrende Programmteile in einem Programm können mehrfach eingebaut werden, d.h. ein gleicher Programmblock ist an verschiedenen Stellen eines Programms anzutreffen, muß also beim Programmieren mehrfach eingegeben werden.

Einmal vergeuden Sie damit viel Zeit, und dann "blähen" Sie damit ein Programm unnötig auf. Mit **GOSUB** schlagen Sie zwei Fliegen mit einer Klappe: der wiederkehrende Baustein wird einmal eingegeben; das Programm bleibt übersichtlich.

Das Menuebeispiel für die Grundrechenarten könnte damit so gestaltet werden, daß für jede Verzweigungsmöglichkeit (z.B. Addition) ein eigener Programmblock geschrieben wird, man spricht von Unterprogramm.

Innerhalb eines umfangreichen Programmlaufs hat ein in sich abgeschlossenes Unterprogramm den Vorteil, daß es für sich allein überprüfbar ist, das Gesamtprogramm übersichtlich bleibt und bei Vergrößerung des Gesamtkonzepts einfach ein Unterprogramm angehängt wird, d.h. eine zusätzliche Verzweigung angeboten wird (z.B. Potenzieren). Diese Art Programme aufzubauen nennt man modulare Programmierung.

Nach soviel Theorie ein praktisches Beispiel:

Listing

```
10 REM Kundendatei
20 PRINT "Kundendaten":PRINT
30 PRINT "ändern (1)"
40 PRINT "eingeben (2)"
50 PRINT "suchen (3)"
60 PRINT: INPUT "Wunsch:";A
70 IF A=1 THEN GOSUB 1000
80 IF A=2 THEN GOSUB 2000

1000 REM Unterprogramm Kundendaten ändern

1500 RETURN
2000 REM Unterprogramm Kundendaten eingeben

2500 RETURN
```

Der Rechner bietet Ihnen ein Menü. Sie programmieren also zuerst den Menüaufbau (Auswahlblock). In einer höheren Zeilennummer (Kommentar erleichtert die Arbeit!) schreiben Sie Ihr Änderungsprogramm - welche Daten ändern. Unabhängig von diesem Programmblock bauen Sie das Modul "Dateneingabe" auf. Da der PC diesen Block getrennt von Wahlmöglichkeit (1) anläuft, könnte in 1 noch ein zu beseitigender Fehler enthalten sein. Ihr Gesamtprogramm läuft zwar noch nicht, der Teil Eingabe ist aber lauffähig. Haben Sie z.B. den Baustein "Abspeichern" übersehen, kann dieser unproblematisch ergänzt werden (z. B. 55 PRINT "abspeichern (4)").

Was ist bei GOSUB .. RETURN zu beachten:

GOSUB : "gehe hinunter zu"; läßt Computer in die angegebene Zeilennummer verzweigen, wobei er sich die auf GOSUB folgende Zeilennummer als Rücksprungsadresse merkt;
kein GOSUB ohne RETURN !

RETURN : "kehre zurück"; beendet das Unterprogramm und verzweigt zu dem unter GOSUB gemerkten Rückkehradresse;
kein RETURN ohne GOSUB !

Setzen Sie dieses BASIC-Anweisungspaar ein, steht nur hinter GOSUB eine Zeilennummer, für RETURN wird keine Rückkehradresse angegeben. Damit der Rechner nach dem Rücksprung in das Hauptprogramm beim Abarbeiten aller Zeilennummern nicht in das erste Unterprogramm eindringen kann (Fehlermeldung RETURN WITHOUT

GOSUB), ist im Hauptblock ein Endekriterium zu setzen (für obiges Listing z.B. 200 END).

END : Ende; das abzuarbeitende Programm ist mit diesem Befehl beendet; Rücksprung in den Direkt-Modus (OK-Meldung); Neustart des Programmes möglich, da Programm noch im Speicher!

2.3.9 Speicherung von Programmen auf Kassette

Ihre Programme werden zunehmend wertvoller, so daß man daran denken sollte, sie zu sichern. Nach jedem Ausschalten des Rechners bleibt zwar der Festspeicher (ROM) mit dem Microsoft BASIC erhalten, der Programmspeicher (RAM = Random Access Memory = Schreib-Lese-Speicher, frei durch den Benutzer programmierbar) verliert bei Stromausfall jedoch alle Daten (Programmzeilen).

Ist Ihr Programm abgeschlossen, sollten Sie mehrere Probeläufe starten. Sie überprüfen quasi Ihre Arbeit auf Richtigkeit. Da wohl kaum ein Programm auf Anhieb fehlerfrei ist, kann nur die Fehlersuche Abhilfe schaffen.

Zu unterschieden sind generell zwei Fehlerarten:

- Fehler in der "Grammatik" = SYNTAX ERROR, bestimmte Regeln der Programmiersprache BASIC sind nicht eingehalten; z.B.: statt GOSUB tippten Sie GOOSUP ein, statt INPUT muß der PC INBUTT "lesen", das in seinem Wortschatz nicht vorkommt;
- Fehler in der Logik; Fehler dieser Art erkennt der PC natürlich nicht, da er Ihre Gedankengänge nicht nachvollziehen kann, z.B. statt zwei Werte zu addieren multipliziert der Rechner; logische Fehler behebt man, wenn man sich alle Variablen ausdrucken läßt und das Programmlisting durch den Drucker auflistet.

Angenommen Ihr Programm läuft einwandfrei, es soll abgespeichert werden. Überprüfen Sie, ob der Kassettenrecorder angeschlossen ist (siehe oben). Suchen Sie Platz auf Ihrer Kassette (eventuell Bandanfang - auf farbigen Vorspann achten, hier erfolgt keine Aufnahme!). Mit CSAVE wird das Programm auf Kassette "übertragen", d.h. der PC teilt dem Recorder Daten ähnlich einem Musikstück mit.

CSAVE : auf (C)assette speichern; PC bringt alle Daten zum Recorderausgang (leises Klicken), die Übertragung wird vorgenommen, leises Klicken und OK-Meldung zeigen Ende an.

Wie bei herkömmlichen Aufzeichnungen auf Band/Kassette ist der Recorder auf Aufnahme zu stellen (gleichzeitig PLAY/RECORD drücken), aber bevor Sie Daten mit CSAVE absenden. Damit Sie Ihr Programm auch wiederfinden, erhält es einen (beliebigen) Namen. Verwenden Sie den Namen der ersten REM-Zeile, in der Sie das Programm bezeichnen. Leider müssen Sie sich auf sechs Zeichen beschränken (z.B. CSAVE "ADDI" nicht aber CSAVE "ADDITION" der Rechner schneidet automatisch ab!).

Wer an seinem Recorder die Möglichkeit hat, einen Zählerstand festzuhalten, sollte sich eine Karteikarte anlegen, die das Inhaltsverzeichnis und den Zähleranfangs/endstand der Kassette enthält. Späteres mühevolleres Suchen erübrigt sich damit.

Nach der OK-Meldung stoppen Sie den Recorder. Sie müssen die Aufzeichnung auf Band überprüfen. Denken Sie an die vielen Fehlerquellen: Lautstärkeregelung vergessen, zu spät auf Aufnahme/Play gedrückt, Bandende, usw. Häufigstes Problem für mangelhafte Aufzeichnung liegt in der Abstimmung der Übertragungsrate (Baudrate) PC - Recorder.

Sehr oft kann diffiziles Einstellen der Lautstärke wahre Wunder bewirken; wer ohnehin keinen Recorder besitzt, sollte einen Datenrecorder anschaffen.

CLOAD? : nach Zurückspulen zum Aufnahmeanfang überprüft der PC die Übertragung auf Fehler; ist der Vergleichslauf in Ordnung, erfolgt OK-Meldung andernfalls Fehlermeldung; im letzten Fall Abspeichern wiederholen!

Haben Sie ein lauffähiges Programm abgespeichert, können Sie es jederzeit mit **CLOAD** in den Programmspeicher des PCs laden und ablaufen lassen.

CLOAD : von der Kassette laden; ähnlich CLOAD? zum Bandanfang bzw. Zählerwert spulen, auf PLAY-Taste des Recorders drücken, CLOAD (oder CLOAD "Name") eingeben; bei CLOAD wird das erste erkannte Programm geladen, bei CLOAD "" (mit Bezeichnung des Programms) gesucht, bis der Name mit dem eingegebenen Namen identisch ist (Programm gefunden=laden); Programme mit anderen Namen werden mit SKIP: Name gemeldet.

2.3.10 DIM - Anweisung

In den bisher behandelten Programmen war ein eingegebener Wert einer Variablen während eines Programmes nur so lange zur Verfügung, bis er durch die folgende Eingabe überschrieben wurde. Beispiel:

```
10 INPUT A
20 SUMME=A+5
30 PRINT SUMME
40 GOTO 10
```

Wird für A der Wert 10 eingegeben, wird als SUMME 15 ausgedruckt. In Zeile 40 springt der PC zurück zu Zeile 10, es erfolgt eine neue Eingabe für A (z.B. 50). In Zeile 20 wird nun der bestehende Wert 15 durch 55 ausgetauscht. Die Zahl 15 existiert nicht mehr.

Wollen Sie alle Eingabedaten des gesamten Laufes verfügbar halten, müssen Sie Speicherplatz reservieren. Sie schaffen sog. Felder, Bereiche, denen numerische oder alphanumerische Werte zugeordnet werden.

DIM : reserviert Speicherfächer für die Aufnahme von Variablen; nicht erforderlich, wenn nicht mehr als 10 Fächer benötigt werden; DIM T\$(30) bedeutet Platz für 31 Strings.

Im u.a. Beispiel werden 15 INPUTs in ein Feld aufgenommen. Der Inhalt des Feldes kann auf Wunsch jederzeit abgerufen werden.

```
10 DIM A(15)
20 FOR I=1 TO 15
30 INPUT A(I)
40 NEXT I
```

Wollen Sie Ihre eingegebenen Werte der Reihe nach wieder aus dem Speicherfeld lesen, geben Sie zusätzlich ein:

```
50 FOR I=1 TO 15
60 PRINT A(I)           : REM Ausdruck untereinander - ; versuchen
70 NEXT I
```

Kommentar: Zunächst sind 15 Zahlen durch Sie einzugeben (FOR... NEXT - Schleife), die der PC im Feld A ablegt (I=1 bis 15). Den Ablegevorgang bekommt man nicht zu sehen. Erst die Ergänzung der Zeilen 50 bis 70 läßt die eingegebenen Zahlen wieder sichtbar werden. Der Rechner läßt auch zweidimensionale Felder zu.

Werden zwei Indexvariablen (z.B. I und J) zur Eingabe verwendet und das Feld entsprechend dimensioniert, ergibt sich:

```
10 DIM A$(3,3)
20 FOR I=1 TO 3
30     FOR J = 1 TO 3
40 INPUT A$(I,J) : REM möglicher Ausdruck PRINT A$(I)
50 NEXT J,I
```

Nachdem I=1, und J die Werte 1 bis 3 durchlaufen hat, wird I um 1 erhöht (I=2 - J durchläuft wieder 1 bis 3), usw. Ergänzen Sie ähnlich dem obigen Beispiel den Ausdruck, bekommen Sie eine komplette Matrix gedruckt (experimentieren Sie mit diesem Beispiel). Sie erhalten insgesamt $3 \times 3 = 9$ Datenelemente.

Beachten Sie, daß im letzten Beispiel auch Texte als Namen eingegeben werden können. Noch vor dem ersten Zugriff auf das Feld muß die DIM-Anweisung ins Programm eingebracht sein. Daß Ihr PC mehrere DIMs zuläßt, ist selbstverständlich.

2.3.11 LEN, LEFT\$, RIGHT\$, MID\$, STRING\$, +, ASC, CHR\$, INSTR

Bereits mehrfach wurden Zeichenketten (strings) angesprochen. BASIC erlaubt die Bearbeitung dieser strings mit einer Reihe von Funktionen, auf die kurz eingegangen werden soll.

```
10 A$="Buch": B$="stabe"
```

Addiert man die Variablen A\$ und B\$ in einer PRINT-Anweisung (PRINT A\$+B\$), druckt der PC Buchstabe.

Mit ASC(A\$) wird der Zahlenwert des ersten Buchstaben der Zeichenkette "Buch" (also B = 60) ausgedruckt.

Die schon bekannte Funktion CHR\$(I) erlaubt fast den gesamten Zeichenvorrat des PCs auf den Bildschirm zu bringen. Beispiel:

```
10 FOR I=32 TO 255: PRINT CHR$(I):NEXT
```

Die Zahlen bis 32 sind Steuerzeichen. Setzen Sie einen Strichpunkt, werden alle Zeichen aneinandergedruckt. Setzen Sie eine weitere FOR...NEXT Schleife ein, können Sie dem Zeichen die zugehörige CHR\$-Zahl entlocken (siehe auch Handbuch).

Wer die Grafikausrüstung des PCs anzeigen möchte, gibt ein (CHR\$(223) ist für den PC ein Schalter!):

```
20 FOR I = 0 TO 255 : PRINT CHR$(223);CHR$(I);: NEXT
```

STRING\$(10,"-") zeichnet eine 10 Zeichenlinie bestehend aus dem Trennungsstrich. Erhöhen Sie die Zahl in der Klammer auf 39!

LEN (A\$) ermittelt die Länge des Wortes A\$, also Buch mit 4.

LEFT\$(A\$,2) druckt Bu aus; zwei Zeichen werden von links beginnend abgezählt und gedruckt.

RIGHT\$(A\$,3) druckt uch aus; von rechts zählend drei Buchstaben werden gedruckt.

MID\$(A\$,2) bringt als Ergebnis uch ; wahlweise steht **MID\$(A\$,2,1)** zur Verfügung -testen Sie selbst, das Ergebnis ist ein u , entferne zwei Zeichen von hinten (rechts) und ein Zeichen von links = u .

INSTR(A\$,"u") mit dieser Funktion sucht der PC, ob das "u" in A\$ vorhanden ist und an welcher Stelle :Ausdruck 2 (2 ter Buchstabe).

Natürlich sind Ergebnisse nur in Verbindung mit dem PRINT-Befehl auf dem Schirm zu sehen.

Ein kurzer Überblick läßt die Möglichkeiten erkennen:

Programm

Ausdruck

10 A\$="Buch":B\$="stabe"	
20 PRINT ASC(A\$)	60
30 PRINT LEN(A\$)	4
40 PRINT LEFT\$(A\$,2)	Bu
50 PRINT RIGHT\$(A\$,3)	uch
60 PRINT MID\$(A\$,2,1)	u
70 PRINT MID\$(A\$,2)	uch
80 PRINT INSTR(A\$,"u")	2
90 PRINT CHR\$(66)	B

2.3.12 READ DATA

Manche Problemstellungen erfordern die Ablage von Daten im Programm. Ein Vokabel-Lernprogramm oder ein Intelligenztest muß bereits abfragbare bzw. überprüfende Daten enthalten. Mit **READ DATA** liegt ein Anweisungspaar vor, das der Aufgabenstellung gerecht wird (Funktion nur in Verbindung miteinander).

READ : weist dem PC Werte zu, die in DATA-Zeilen abgelegt sind
Werte werden in einer READ zuzuordnenden Variablen untergebracht (READ A)

DATA : Ablagefach für Daten (Zahlen, Zeichen, Texte) in der Form DATA 12,...,TEXT, usw

DATAs können überall im Programm stehen. Ein Zeiger wird an das erste Datenfeld gesetzt. Kommt der Rechner zu einer READ-Anweisung, "öffnet" er das Fach, vor dem der Zeiger steht, der erste Wert wird der Variablen zugeordnet, der Zeiger wandert ein Fach weiter.

Beispiel einer kleinen Personendatei:

<u>Listing</u>	<u>Kommentar</u>
10 PRINT "Vorname","Name"	(Überschrift - Einteilung)
20 PRINT STRING\$(20," - ")	(Untersteichung)
30 FOR I=1 TO 3	(Schleife für 3 Durchläufe)
40 READ V\$,N\$	(suche und nehme aus NAMES)
50 PRINT V\$, N\$	(Druck der gelesenen Werte)
60 NEXT I	(Schleifenende)
70 DATA Hans, Huber	(Daten der Personen Huber.
80 DATA Horst, Meier	usw. - Zeiger beginnt in
90 DATA Gerda, Beyer	Zeile 70 - 1. Druck)

Sie erhalten eine Fehlermeldung (Out of DATA), wenn Sie z.B. die Laufvariable in der FOR-NEXT-Schleife auf 4 erhöhen. Der Zeiger läuft alle DATAs ab, Nummer 4 fehlt; eine Zeile müßte ergänzt oder auf der Zeiger auf 1 zurückgesetzt werden. Mit **RESTORE** erreicht man die letzte Überlegung.

RESTORE : der folgenden READ-Anweisung werden wieder alle DATAs zur Verfügung gestellt; Zeiger auf erster DATA-Zeile.

2.3.13 PRINT-USING, PRINT TAB(I), PRINT SPC(I)

Die PRINT-Anweisung hat den Nachteil, daß formatierte Ausgabe nur sehr bedingt möglich ist. Der Programmierer kann die von Ihnen eingegebenen Werte nicht im voraus auf dem Bildschirm testen; ein Druck einer Berechnung im E-Format (1.45E+..) ist durchaus denkbar. Sollen diese Werte gar noch untereinander zu stehen kommen, wird man schier wahnsinnig. PRINT-USING hilft aus der Patsche. Alle Arten von Variablen, Konstanten und arithmetrischen Ausdrücken lassen sich komfortabel untereinander drucken.

PRINT-USING : formatierter Ausdruck (Maske); #-Zeichen ist Platzhalter für aufzunehmende Zahlenwerte; Zahlen werden rechtsbündig mit gerundeten Werten aufgeführt (Komma als Orientierungshilfe)

PRINT TAB(X) : erzeugt soviele Leerstellen, bis die angegebene Position X erreicht ist

Auf ein gesondertes Beispiel wird verzichtet, da u.a. Listing die Problematik näher erläutert (selbsterklärend).

2.3.14 COLOR, CONSOLE

Eigentlich haben Sie sich einen "Farbcomputer" gekauft. Die obigen Beispiele haben bisher auf Farbwerte verzichtet. Für die eigentlichen Programmläufe spielt Farbe eine untergeordnete Rolle. Daß sie gestaltet, hervorhebt und dadurch die Merkfähigkeit erhöht, ist einsichtig. Insgesamt stehen sieben Farben zur Verfügung. Sie sollten durch Veränderung der Zahlenwerte die unterschiedlichen Kombinationen testen. COLOR hat das Format:

COLOR A,B,C : A steht für die Farbe der Zeichen 7=weiße Schrift
 B symbolisiert die Hintergrundfarbe 2=rot
 C läßt inverse oder blinkende Darstellung zu

Beispiel:

```
10 FOR I=0 TO 3:COLOR 2,7,i:PRINT "alphaTronic":NEXT:COLOR ,,0
20 FOR J=1 TO 7:COLOR 0,J:PRINT "TRIUMPH-ADLER"
30 FOR Z=1 TO 7:COLOR Z,0:PRINT "Personal-Computer"
```

CONSOLE : legt die Dimensionierung des Bildschirmfensters fest; läßt den Schirm "scrollen" (nach oben verschwindende Zeichen)

Testen Sie:

CONSOLE 0,24,1,1 : normaler Status des Bildschirms mit Anzeige der Funktionstasten

CONSOLE ,,0,1 : Anzeige der Funktionstasten verschwindet

CONSOLE ,15,1 : Fenster ab Zeile 1 bis 15 im Roll-Modus

Das folgende Listing einer Kalkulation zeigt das Zusammenspiel der bisher besprochenen Befehle, insbesondere PRINT-USING und TAB.

Bemerkung: Die optische Trennung der BASIC-Befehle in verschiedenen Zeilen (z.B. 220 oder 240) ist drucktechnisch bedingt. Natürlich kann nach dem Doppelpunkt angesetzt werden.

```

100 CLS : REM Bildschirmlöschung - Funktionstasten?
110 PRINT " Geben Sie im folgenden die Werte ein!":PRINT
120 LOCATE 0,8:INPUT " Einkaufspreis in DM :          ",E:PRINT
130 LOCATE 0,10: INPUT " Liefererrabatt in % :          ",L:PRINT
140 LOCATE 0,12: INPUT " Liefererskonto in % :          ",S:PRINT
150 LOCATE 0,14: INPUT " Bezugskosten in DM :          ",B:PRINT
160 LOCATE 0,16: INPUT " Selbstkosten in % :          ",K:PRINT
170 LOCATE 0,18: INPUT " Gewinn in % :          ",G
180 REM *****
190 '  * K A L K U L A T I O N - BERECHNUNG - A U S G A B E *
200 '  *****
210 PRINT :PRINT CHR$(12) : REM Bildschirmlöschung
220 PRINT " Listeneinkaufspr. DM " TAB(25):
    PRINT USING"#####.##";E
230 LET L1 = E*L/100 : LET kann entfallen
240 PRINT " - Lief.-Rabatt(%) DM " TAB(25):
    PRINT USING"#####.##";L1
250 PRINT STRING$(35," ")
260 LET P1 = E -L1
270 PRINT " Zieleinkaufspreis DM " TAB(25):
    PRINT USING"#####.##";P1
280 LET S1=P1*S/100
290 PRINT " - Lief.-skonto(%) DM " TAB(25):
    PRINT USING"#####.##";S1
300 PRINT STRING$(35," ")
310 LET P2=P1-S1

```

```

320 PRINT " Bareinkaufspreis  DM " TAB(25):
    PRINT USING"#####.##"; P2
330 PRINT " + Bezugskosten  DM " TAB(25):
    PRINT USING"#####.##";B
340 PRINT STRING$(35," - ")
350 LET P3 = P2 +B
360 PRINT " Bezugspreis      DM " TAB(25):
    PRINT USING"#####.##";P3
370 LET K1 =P3*K/100
380 PRINT " + Selbstkst.in(%) DM " TAB(25):
    PRINT USING"#####.##";K1
390 PRINT STRING$(35," - ")
400 LET P4=P3+K1
410 PRINT " Selbstkostenpr.  DM "TAB(25):
    PRINT USING"#####.##"; P4
420 LET G1=P4*G/100
430 PRINT " + Gewinn in (%)  DM " TAB(25):
    PRINT USING"#####.##";G1
440 PRINT STRING$(35," - ")
450 LET P5 = P4 + G1
460 COLOR 6,0
470 PRINT " Barverkaufspreis DM " TAB(25):
    PRINT USING"#####.##";P5
480 PRINT STRING$(35,"=")
490 COLOR 0,6 :PRINT "      Ende der Kalkulation "
500 LOCATE 38,21
510 BEEP
520 END :` Programmende für Kalkulation
530 K$="" : Beginn einer Unteroutine zur Eingabeüberprüfung
540 A$=INKEY$:IF A$="" THEN 540
550 IF ASC(A$)=13 THEN 600
560 IF (ASC(A$)<>&H2E AND ASC(A$)<&H30 OR ASC(A$)>&H39) THEN
    PRINT CHR$(7);:GOTO 540
570 K$=K$+A$
580 PRINT A$;
590 GOTO 540
600 RETURN :   bedenken Sie beim Test, es fehlt das GOSUB

```

Wesentliches in Kürze:

Jeder Befehl besteht aus einer Zeilennummer, einem Schlüsselwort und einem von Befehl zu Befehl verschiedenen Operandenteil.

Neben Systembefehlen wie RUN, NEW, LIST und CLOAD, werden in BASIC Kommandos verwendet:

PRINT	erlaubt den Ausdruck von Variablen und Texten
LET	weist Variablen/Konstanten zu = eigentliche Verarbeitung
INPUT	läßt Eingaben über die Tastatur zu
\$	Dollarzeichen für die Kennzeichnung von Textvariablen
IF THEN	bedingte Verzweigung, eine bestimmte Zeilennummer abzuarbeiten
GOTO	unbedingte Verzweigung die hinter dem Befehl stehende Zeilennummer anzusteuern
Ver-gleichs-operatoren	Zeichen, die zwei oder mehr Ausdrücke miteinander vergleichen; dienen insbesondere dem Vergleich für Verzweigungsbedingungen
REM	erläutert in einem Programm (Bedeutung, macht auf Änderungsmöglichkeiten aufmerksam usw.)
FOR..NEXT	Anweisungspaar zur Programmierung von zählergesteuerten Schleifen
LOCATE	Positionierungsmöglichkeit des Cursors in Spalten und Zeilenwerte; Cursorschaltung Ein/Aus
GOSUB	Anweisung zum Aufruf eines Unterprogrammes (in sich abgeschlossenes aber untergeordnetes Modul in einem Hauptprogramm)
RETURN	beendet Unterprogramm und kehrt in Hauptprogramm zurück
CLOAD	Laden von Programmen/Daten, die auf Magnetbandkassetten gespeichert sind
CSAVE	speichert Programme auf Kassetten - lagert aufbewahrungswerte Daten/Programme aus
DIM	legt Anzahl der aufzunehmenden Elemente eines Feldes fest
READ	liest Werte aus DATA-Zeilen und ordnet sie einer Variablen zu

DATA nimmt eine Reihe von Eingabewerten in Programmzeilen auf.

3. Arbeiten mit einem Diskettenlaufwerk

Schneller Datenaustausch und hohe Speicherkapazität, sind Kennzeichen der Floppy. Profi-Software bedingt mindestens ein Laufwerk (F1). Außerdem ist man schnell in der Lage, eigene Datenbänke aufzubauen. Ein Stück des Weges dorthin soll der Schluß dieses Beitrags führen.

Wenn Sie die Floppy wie eingangs beschrieben gestartet haben, erscheint nach kurzer Ladezeit (DISK-BASIC!):

How many files(0-15)?

Es genügt zunächst, die Eingabetaste zu drücken, worauf sich DISK-BASIC meldet (Copyright, Version, freier Speicher). Auch hier zeigt die OK-Meldung Eingabebereitschaft (blinkender Cursor).

3.1 Besonderheiten

Grundsätzlich funktioniert DISK-BASIC wie ROM-BASIC. Der Wortschatz ist lediglich um einige sehr nützliche Befehle erweitert. Bevor Sie Programme entwerfen, sollten Sie an die Speicherung denken. Sie brauchen eine formatierte Diskette (mit System).

Auf der mitgelieferten SYSTEM-DISC ist ein Programm FORMAT, mit dem Sie fabrikneue Disketten zunächst für den PC "lesbar" machen (Laden des Programms mit LOAD "FORMAT" - Schlußhochkomma kann entfallen). Rote Betriebslampe zeigt den Ladevorgang an - niemals währenddessen die Diskette herausnehmen!.

Ist das Programm geladen, wechseln Sie die Disketten (SYSTEM-DISK. aus F1, neue Disk einschieben), starten Sie FORMAT - die neue Diskette wird formatiert. Sie sollten auch ein SYSTEM auf die Diskette bringen (SYSCOPY in PC laden - starten mit inliegender SYSTEM-DISC, da PC SYSTEM davon, liest und auf neue DISK überträgt - auf Abfragen achten!).

Erst wenn Ihnen eine aufnahmebereite Diskette vorliegt, sollten Sie ans Programmieren gehen.

Den Inhalt jeder DISK-BASIC-Diskette können Sie mit FILES einsehen. Geben Sie LFILES ein, wird das Verzeichnis über einen angeschlossenen Drucker ausgegeben.

Programme, die Sie nicht mehr brauchen, sollten Sie löschen. Mit KILL (+ "Name des Programms") wird die betreffende Datei aus dem Verzeichnis entfernt und ist damit "verloren". Wollen Sie eine wichtige Datei schützen, ist dies möglich mit

SET "Name", "P" : schützt die mit "Name" bezeichnete Datei vor Überschreibungen oder Löschungen

SET "Name", " " : der o.a. Schreibschutz wird aufgehoben

SAVE "Name" : unter "Name" wird eine Datei angelegt, die Programm aufnimmt (abspeichern); Neustart möglich, da Programm im Speicher erhalten bleibt

TA als Hersteller Ihres Laufwerks empfiehlt zur Datenaufzeichnung Disketten vom Type DS/DD bzw. 2S/2D (=doppelte Schreib- und Aufzeichnungsdichte - Vorder- und Rückseite werden beschrieben). Natürlich ist es möglich, Disketten SS/DD (einfach beschichtet) zu verwenden. Sie gehen natürlich ein höheres Risiko von Fehlern ein. Bei Verwendung einseitig geprüfter Disketten (SS/DD) sind Fehlermeldungen wie "BAD SECTOR" oder "DISK-IO-ERROR" nicht auszuschließen. Der geringe Preisunterschied sollte die erhöhten Anforderungen an Datensicherheit Wert sein.

3.2 Einrichten einer Datei

Grundsätzlich wären zwei Arten von Dateien möglich: Sequentielle und Random-Dateien. Das Beispiel soll auf sequentiellen Zugriff (von 1 bis n nacheinander) begrenzt bleiben.

Eingabe-, Verarbeitungs- und Ausgabeprogrammteile sind aufzubauen. In Kürze wichtige BASIC-Befehle:

OPEN : öffnet eine Datei zur Ein/Ausgabe von Daten

FOR OUTPUT : jeder Ein/Ausgabevorgang beginnt am Anfang der Datei

CLOSE : schließt geöffnete Dateien

WRITE : schreibt Daten in eine sequentielle Datei

EOF : Endemarkierung einer Datei

Listing

Kommentar

10 OPEN "KUNDE" FOR OUTPUT AS #1	Datei KUNDE eröffnen
20 INPUT "Kunden-Nummer:",KNR	lfd. Kundennummer
30 IF KNR = 0 THEN CLOSE # 1:END	bei Eingabe 0 ENDE
40 INPUT "Name : ",NAM\$	Eingabewerte des
50 INPUT "Vorname : ",VOR\$	Kunden
60 INPUT "PLZ : ",PLZ\$	
70 INPUT "Ort : ",ORT\$	
80 INPUT "Straße - Nr. : ",STR\$	
90 WRITE #1, KNR,NAM\$,VOR\$,PLZ\$,ORT\$,STR\$	Daten auf DISK
100 GOTO 10	Rücksprung zum Anfang

Um die Daten auf der Diskette lesen zu können, ist ein weiteres Teilprogramm erforderlich. Hier ein mögliches Listing:

200 OPEN "KUNDE" FOR INPUT AS #1	öffnen der Datei mit
210 IF EOF(1) THEN CLOSE: END	Satz 1, Datei-Ende
220 INPUT #1,KNR,NAM\$,VOR\$,PLZ\$,ORT\$,STR\$	
230 PRINT "Kunden-Nummer:",KNR	
240 PRINT "Name : ",NAM\$	Datenanzeige des Kunden
250 PRINT "Vorname : ",VOR\$	
260 PRINT "PLZ : ",PLZ\$	
.	
.	
300 GOTO 200	Rücksprung bis alle Kunden angezeigt

Das Programm kann beliebig ausgebaut werden. Statt Kunden nimmt es bei kleinen Änderungen Schüler, Geburtstage usw. auf. Wer sich mit Dateien in der Selbstprogrammierung beschäftigt, wird bei manchem Problem Spezialliteratur benötigen. Letztlich sollte dieser kleine Ausblick ermutigen, eigene DISK-Programme mit Dateien zu erstellen. So manches "Profi-Programm" könnten Sie beim weiteren Studium selbst erstellen und damit teures Geld ersparen - nur wer weiß, was eine Maschine leisten könnte, wer Ihre Möglichkeiten richtig einschätzt im Hinblick auf Einsatzbereiche und Arbeitserleichterung, der profitiert von der neuen Technologie - dem Computer.

Nur ein Beispiel: wer einmal mit einem guten Textverarbeitungsprogramm gearbeitet hat (z.B. MICRO-TEXT-PLUS), kann schnell auf eine Schreibmaschine verzichten.

Auflösung des Beispiels (Kopfrechnen) :

```
10 REM *****
11 REM *   Konzept - Kopfrechenprogramm   *
12 REM *****
15 CLS:          REM Bildschirmlöschung
20 PRINT "  Überprüfen Sie sich im Kopfrechnen!"
30 PRINT :PRINT
40 PRINT "Addition      (1)"
50 PRINT "Subtraktion   (2)"
60 PRINT "Multiplikation (3)"
70 PRINT "Division      (4)"
75 PRINT
80 PRINT "Ende          (0)"
85 PRINT
90 INPUT "Was möchten Sie üben";A
100 IF A=1 THEN 200
110 IF A=2 THEN 300
120 IF A=3 THEN 400
130 IF A=4 THEN 500
140 IF A=0 THEN END
200 PRINT :INPUT "Wieviel ist 35 + 65";L
210 IF L=100 THEN PRINT "Richtig!" ELSE BEEP
220 FOR I=1 TO 250:NEXT:GOTO 10:'Zeitschleife
300 PRINT :INPUT "Wieviel ist 35 - 65";S
310 IF S=40 THEN PRINT "Richtig!" ELSE BEEP
320 FOR I=1 TO 250:NEXT:GOTO 10:'Zeitschleife
400 REM Multiplikation
500 REM Division
600 GOTO 10 : ENDLOSSCHLEIFE!
```

Zum Schluß sei noch auf Programme aus dem Hause TA hingewiesen, die sich mit dem Erlernen der Sprache BASIC im Dialog mit dem Anwender beschäftigen. Zugängliche Listings, Möglichkeiten des Einsatzes (Rechen- /Spielbereich) und Überprüfung der Lernfortschritte durch den Computer sollten Anreiz genug sein. Daß

sich das Erstellen von Programmen als sehr zeitaufwendig erweist, haben Sie sicher nach dem Durcharbeiten und Testen der aufgezeigten Beispiele erkannt. Fertige Programme können Ihnen bei der Selbstprogrammierung hilfreiche Dienste leisten z.B.:

- Programmteile mit MERGE (nur DISK-BASIC) in bestehende Eigenprogramme einbaubar
- Programmablauf nachvollziehbar
- Lernen nach (vorliegendem) Muster
- Überprüf-Unterprogramme (einsehbar)
- viele Tricks (Grafikanwendung - Tonvariationen)
- erweiterter Einblick in die Anwendung und Einbettung nicht genannter Befehle (INKEY\$, WHILE WEND, RND, usw.)

LERNEN MIT BASIC: Programmsammlung mit 11 Programmen aus dem Rechen-/Spielbereich (alle Listings sind in der Bedien.-Anleitung abgedruckt)

BASIC LEICHT GELERNT: Interaktiver Grundkurs zum Erlernen von BASIC (der PC reagiert auf Ihre Fehler)

PC-GRAFIK: Programmierübungen mit den Farb- und Semigrafik-Möglichkeiten des PC

SEQDAT: "Disketten-Unterricht" und Lernprogramm zum Erstellen sequent. Dateien

RANDAT: Was sind RANDOM-Dateien - wie geht man damit um - wie werden sie programmiert

LINEARE GLEICHUNGSSYSTEME:

BASIC und Mathematik: Eine umfassende Dokumentation erläutert den Lösungsweg anhand von kommentierten Nassi-Shneiderman-Struktogrammen

Als weiterführende Literatur zu dieser kleinen Einführung in BASIC eignet sich besonders das Buch

"BASIC mit dem TA alphatronic PC
Fortschrittliches Programmieren für die Praxis
Band 1"

von Prof. Dr. B. Bollow, N. Bollow, S. Lumma

aus dem Heim-Verlag, Darmstadt.

Die zahlreichen Programmierbeispiele, darunter z.B. eine Kundendatei oder eine Fakturation, werden auch mit Struktogrammen und Flußplänen dargestellt.

Hochauflösende Grafik für TA alphasatronic PC

Die BiCom-Grafikerweiterung wird direkt im PC installiert. Damit erhält Ihr PC eine professionelle Grafik mit 320x264 Bildpunkten in 8 Farben oder Helligkeitsstufen bei monochromen Monitoren. Löschen oder Scrollen des Textes bewirkt keine Beeinflussung der Grafik-Darstellung, da die Erweiterungskarte einen eigenen, unabhängigen Bildspeicher besitzt.

Das BiCom-Netzwerk für Ausbildung und Büro

Mit den MH-Systemen von BiCom und Arbeitsplatzrechnern von TA ergeben sich leistungsfähige und kostengünstige Netzwerksysteme für bis zu 15 Netzwerkteilnehmer. Das BiCom-Ausbildungssystem mit dem alphasatronic PC ist in vielen Schulen eingeführt. Durch seine vielfältigen Interaktionsmöglichkeiten zwischen Lehrerplatz und Schülerplätzen läßt es sich im Informatikunterricht ebenso wie in der beruflichen Ausbildung einsetzen. Alle Eigenschaften kommerzieller Systeme, wie Datei- und Satzschutz, Drucker-Spooler, freikonfigurierbare Peripherie-Anschlüsse, sind selbstverständlicher Standard.

Betriebssysteme und Standard-Software

BiCom hat Concurrent DOS von Digital Research, das derzeit wohl modernste Betriebssystem für Microcomputer, an TA P50/P60 angepaßt. Dabei kann ein Programm über einen alphasatronic PC als externe Konsole bedient werden, so daß damit eine besonders preisgünstige EDV-Lösung für kleine Betriebe zur Verfügung steht, bei denen größere Mehrplatz- oder Netzwerksysteme zu aufwendig wären. Daneben vertreibt BiCom ein umfangreiches Angebot von Standard-Software, angepaßt für die verschiedenen TA-Rechner.

Alle Produkte von BiCom erhalten Sie über den TA-Fachhandel. Informationen und Preislisten bekommen Sie bei Ihrem TA-Händler oder direkt von

BiCom Datensysteme GmbH
Geschäftsstelle Hannover
Bultstraße 25
3000 Hannover 1

Tel. (0511) 814075
Telex 922597 bicom d

BiCom Peripherie für Microcomputer von **TA**

Lernen Sie Ihren alphantronic PC besser kennen mit



Stefan Igelmanns
BASIC-

TRICK-

KISTE

- **DISK BASIC Nachschlagewerk**
Syntax-Grafiken, Erläuterungen zu jedem Befehl
- **Tabellen**
Tastaturbelegung, ASCII-Code-Tabelle, Port-Tabelle ...
- **Hochauflösende Grafik**
Erläuterungen zur BiCom-Grafik, Farb- und Liniendemos,
Testbild zum Monitor-Einstellen
- **Maschinennahe Programmierung/Programmiertricks**
Wichtige Systemadressen, Drucker- und Summeransteuerung,
Darstellung einer 25. Bildschirmzeile, Hardcopy...
- **Experimentierprogramme**
Bildschirm-Controller programmieren, Cursordarstellung verändern,
Attribut-RAM bitweise programmieren ...
- **Mikrocomputer-Blockdiagramm**
Veranschaulicht die Funktion eines Computers und stellt grafisch die
Spannungszustände an den Schnittstellen dar

Alle Informationen können innerhalb von Sekunden auf den Bildschirm geholt und in das eigene Programm eingefügt werden. Zeitraubendes Nachschlagen in Büchern und Abtippen von Programmzeilen wird weitgehend überflüssig.

Stefan Igelmanns BASIC-TRICK-KISTE wird auf Diskette im TA-DISK-BASIC-Format geliefert und kann über alle TA-Händler zum empfohlenen Verkaufspreis von DM 98,00 einschl. MWSt. bezogen werden.

Stefan Igelmanns BASIC-TRICK-KISTE © 1985 Stefan Igelmann, Osnabrück

Programmieren in BASIC – welche Möglichkeiten gibt es ?

N. Griebinger

Wie Sie bereits erfahren haben, ist man schon in der Grundausstattung, d.h. ohne Diskettenlaufwerk in der Lage, den PC in BASIC zu programmieren. Dies wird ermöglicht durch einen in ROM's (Read Only Memory) befindlichen Interpreter. Dieser von Microsoft stammende ROM-BASIC-Interpreter enthält einen im Vergleich zum Standard-BASIC erweiterten Sprachumfang, wodurch ein komfortableres Programmieren möglich wird. Kann man zwar mit diesem ROM-BASIC Programme auf Kassette abspeichern und wieder laden, so ist doch der Betrieb mit dem Kassettenrekorder recht zeitraubend und nur für den Einstieg zu empfehlen.

Hat man aber die Diskettenstation zur Verfügung, dann gibt es drei weitere BASIC-Programmierungsmöglichkeiten. Da ist zunächst das von Triumph-Adler mitgelieferte Disk-BASIC, das zum ROM-BASIC paßt, aber um die Befehle zur Dateiverarbeitung vergrößert ist. Dieser Interpreter wird nicht vom ROM, sondern von der Diskette in den Speicher des PC's geladen. Zwar kennt auch das ROM-BASIC die Namen der Diskettenbefehle, wird aber bei deren Ausführung immer die Meldung "Disk offline" anzeigen. Damit Programme, die unter ROM-BASIC erstellt wurden, später auch unter DISK-BASIC ohne Änderung ablaufen können, werden reservierte Diskettenbefehle unter ROM-BASIC auch nicht als Variablennamen zugelassen.

Unter dem Betriebssystem CP/M 2.2 von Digital Research sind schließlich die weiteren beiden Möglichkeiten zu finden. Einmal gibt es auch hier einen BASIC-Interpreter, das sogenannte MBASIC. Andererseits kann man ein z.B. mit einem Editor oder Textprogramm eingetipptes Programm mit dem BASIC-Compiler und Linker in ein dem Prozessor verständliches Programm übersetzen.

MBASIC und BASIC-Compiler sind bezüglich der Befehle fast gleich, enthalten einen BASIC-Grundwortschatz und sind auch auf vielen anderen Computern zu finden. Waren die ersten beiden Interpreter im Preis der Geräte enthalten, sind diese zusammen mit dem CP/M separat zu beziehen. Sicherlich gibt es unter CP/M noch mehr BASIC-Dialekte als hier genannt, diese sind allerdings nicht so weit verbreitet wie die von Microsoft.

Unterschiede zwischen den BASIC-Sprachen

Wie bereits oben erläutert, gehören ROM- und DISK-BASIC sowie MBASIC und BASIC-Compiler enger zusammen. Der Unterschied bei den Gruppen liegt zum einen im unterschiedlichen Sprachumfang, zum anderen in der unterschiedlichen Diskettenorganisation. Befehle wie CLS, COLOR, LOCATE usw. gibt es in MBASIC nicht und sind dort durch Steuersequenzen zu ersetzen. Beispielsweise erzielt man dort die gleiche Wirkung wie COLOR 5,0 durch die PRINT-Anweisung:

```
PRINT CHR$(27);CHR$(85);CHR$(53);CHR$(48)
```

Dabei ist das Zeichen CHR\$(27) (ESCAPE) als Steuerzeichen zu verstehen, das dem Computer sagt, daß die nachfolgenden Zeichen nicht als Zeichen, sondern als Bildschirmsteuerung zu interpretieren sind. Diese Form der Steuerung trifft im Wesentlichen auf alle Bildschirm- bzw. Cursorsteuerungen zu. Eine genaue Liste der für den PC geltenden ESCAPE-Sequenzen befindet sich im MBASIC- und Systemhandbuch von TA.

Der zweite Unterschied liegt in der Diskettenorganisation. So steht das Inhaltsverzeichnis mit der FAT (FILE ALLOCATION TABLE) bei Disk-BASIC auf Spur 18 Seite 1, bei CP/M dagegen auf Spur 2 Seite 0. Außerdem arbeitet Disk-BASIC mit 256 Byte-Sektoren, während CP/M mit 128 Byte-Record's arbeitet.

Hat man nun ein Programm in Disk-BASIC geschrieben, so ist es durchaus möglich, dieses auf CP/M-Format zu bringen. Dazu muß das Programm zunächst als ASCII-Datei auf der Diskette abgespeichert werden. Dann kann man mit dem auf der TA-UTILITY-Diskette befindlichen Programm "CONVERT" die Umsetzung vornehmen. Bleibt nur noch die Änderung bestimmter Befehle wie oben erläutert und die Erprobung des Programms. Dieser Weg ist immer dann leichter als das erneute Eintippen, wenn es sich um umfangreichere Programme handelt. Natürlich läßt sich auch dieser Vorgang durch ein Programm automatisieren. Als halbautomatisch könnte man die Verwendung eines Textprogrammes bezeichnen, das über einen Befehl "Ersetzen von durch" verfügt (PCTEXT oder Microtext).

Vorteile und Nachteile

Der große Vorteil von ROM- und DISK-BASIC liegt darin, daß hierfür kein zusätzliches Geld ausgegeben werden muß, weil diese Bestandteil der von TA gelieferten Anlage sind. Es ist auch kein eigenes Betriebssystem nötig, da auch diese Aufgaben

vom Interpreter mitübernommen werden.

Bezüglich der Größe der Programme hingegen ist man durch die Tatsache, daß der Interpreter immer im Speicher vorhanden sein muß, um BASIC dem Prozessor verständlich zu machen, doch etwas beschränkt (ca. 27 kB). Außerdem lassen sich die Programme nicht für den Benutzer schwer zugänglich machen, weil es kein Abspeichern mit Listschutz gibt, was speziell für Softwareanbieter nachteilig ist.

Der bereits oben erwähnte erweiterte Wortschatz ist sicherlich ein Vorteil, ebenso, wie die doppelte Rechengenauigkeit bei Funktionen wie Sinus usw. Dies ist übrigens auch der Grund, warum der PC bei den Benchmark-Tests mancher Computerzeitschriften anscheinend etwas schlechter abschneidet als Geräte, die preislich und von der Hardware her darunter liegen. Leider wird in solchen Berichten das Bild dann völlig verzerrt, weil die Bewertung dieser standardmäßig doppelten Genauigkeit sehr selten miteinbezogen wird und in der Regel nur Zeiten als Maß für die Schnelligkeit des Rechners aufgelistet werden.

Bei MBASIC gilt der gleiche Nachteil bezüglich des verbleibenden Speichers wie oben. Der Vorteil liegt aber hier in der sehr hohen Kompatibilität mit dem BASIC-Compiler. Man kann mit keinen oder wenigen Änderungen ein mit MBASIC erstelltes und getestetes Programm mit dem Compiler und Linker in ein ohne Zusatzprogramme laufendes Programm übersetzen. Mit MBASIC können also die Programme sofort getestet werden, bevor man sie übersetzt.

Verzichtet man darauf und erstellt die BASIC-Quelldatei z. B. mit einem Editor (ED, Microtext, PCTEXT), so muß man bei eventuell vom Compiler bemerkten Fehlern das Programm mit dem Editor zuerst ändern und dann immer wieder neu übersetzen. Dies ist zeitlich aufwendiger als mit MBASIC zu arbeiten, spart aber die Anschaffung von MBASIC und hat zudem den Vorteil, daß man größere Programme in einem Stück in den Speicher laden kann.

Ein weiterer Vorteil des Compilers liegt in einer Vergrößerung der Geschwindigkeit in der Ausführung und in der fast unmöglichen Veränderbarkeit der Programme, was bei Software-erstellern sehr erwünscht ist.

Mit dem Laufzeitmodul BRUN.COM lassen sich noch weitere Vorteile nennen. Doch wofür ist BRUN (sprich Bi-Rann) eigentlich nötig?

Wurde beim Interpreter jeder Befehl sofort interpretiert, so wird beim Compiler grob gesagt jeder Befehl durch Unterprogrammsprünge in eine LIBRARY (Bibliothek) realisiert. Dort

stehen dann Routinen, die für die Ausführung z. B. einer PRINT-Anweisung nötig sind. Das Dazubinden dieser Bibliothek zum vom Compiler übersetzten Programm übernimmt der Linker (Binder).

Hat man nun viele Programme auf einer Diskette, die mit dem Compiler übersetzt wurden, ist leicht einzusehen, daß durch das Hinzubinden der gleichen Bibliothek die gleichen Informationen bei jedem Programm vorhanden sind und somit eine Menge Platz überflüssig belegt wird.

Deshalb wurde die Möglichkeit geschaffen, beim Übersetzen des Programms mit dem Compiler zwei Optionen anzugeben, einmal mit der OBSLIB, die der umfangreichen Bibliothek entspricht, oder mit der BASLIB später linken zu wollen, was auch Standardeinstellung ist. Wurde mit der BASLIB gelinkt, so muß zur Ausführungszeit eines Programms auch das Laufzeitmodul BRUN vorhanden sein. Dieses Laufzeitmodul wird auch von allen anderen derartig gelinkten Programmen benutzt, womit eine ansehnliche Platzersparnis auf der Diskette verbunden ist.

Darüberhinaus wird mit der Verwendung von BRUN die COMMON-Deklaration von Variablen möglich, d. h. Variablen haben für mehrere Programme, die nachgeladen werden, ihre Gültigkeit.

Der Nachteil von BRUN liegt darin, daß hierfür eine eigene Lizenz zu entrichten ist. Im Prinzip läßt sich das oben Gesagte auch auf andere Programmiersprachen verallgemeinern, soweit es sich um Eigenschaften von Interpretern oder Compilern handelt.

Einen Vorteil von MBASIC kann man vielleicht noch darin sehen, daß dieser Interpreter auf sehr vielen Computersystemen unter CP/M verfügbar ist, sodaß man ASCII-abgespeicherte Programme, die ja auch der Compiler benötigt, auf andere Systeme übertragen (PCTRANS, CPMTRANS, MOVE-IT), anpassen (Bildschirmsteuerung etc.) und schließlich starten kann. In der 16-Bit-Welt heißt der zu MBASIC bezüglich des Wortschatzes sehr nah verwandte Interpreter BASIC86, womit eine Übertragungs- und Anpassungsmöglichkeit auch in dieser Richtung gewährleistet ist.

Einige Tips für ROM- und Disk-BASIC

Reparieren der Allocation-Table

Wer viel Programme unter Disk-BASIC geschrieben hat, hat vielleicht auch schon einmal die sehr schlimme Fehlermeldung "Bad allocation table" auf seinem Bildschirm lesen können.

Dies bedeutete in der Regel, daß mit dieser Diskette nicht mehr zu arbeiten ist bzw., daß die darauf enthaltenen Programme verloren waren.

Zunächst soll dargelegt werden, wie dieser Fehler überhaupt entsteht. Sie haben ein Programm in der Erstellungs- und Erprobungsphase, das eine oder auch mehrere OPEN-Anweisungen enthält, durch Betätigen der Break-Taste oder mit CTRL+C unterbrochen, bevor die offenen Dateien wieder geschlossen wurden. Dann tauschen Sie die Diskette und es wird mit großer Sicherheit die "Allocation Table" dieser Diskette beim nächsten Diskettenzugriff, sofern damit ein automatisches Schließen der noch offenen Dateien verbunden ist, zerstört.

Durch einen OPEN-Befehl wird die Lage der Dateien in den Speicher eingelesen, verändert und nach dem CLOSE wieder auf die Spur 18, Seite 1, Sektor 14 bis 16 der Diskette dreifach übertragen.

Durch einen Wechsel der Diskette wird dann die Dateiverteilung von der vorherigen Diskette auf die neu eingelegte Diskette geschrieben, wenn noch Dateien offen sind wie oben beschrieben. Weil jetzt das Inhaltsverzeichnis (Spur 18 Seite 1 Sektor 1 ff.), in dem die Dateinamen, Attribute und der erste von jeder Datei belegte Cluster steht, jetzt nicht mehr mit der Dateiverteilungstabelle übereinstimmt, wird die Fehlermeldung "Bad allocation table" ausgegeben.

Derartig unbenutzbar gewordene Disketten sind mit Mühe aber wieder zu restaurieren, wenn Geduld, Wissen und einige hilfreiche Programme geschrieben werden oder vorhanden sind. Eine wichtige Funktion hierbei spielt die Möglichkeit, unter Disk-BASIC direkt auf die Informationen der Diskette mit den Befehlen DSKI\$ bzw. DSKO\$ zugreifen zu können (vgl. Handbuch zu Disk-BASIC).

Mit diesen Anweisungen kann man zunächst den Anfangssektor eines jeden Clusters ansehen oder noch besser ausdrucken, um erkennen zu können, zu welcher Datei dieser gehört. Dazu muß man natürlich erst wissen, daß ein Cluster immer 8 aufeinanderfolgende Sektoren a 256 Byte ist. Diese Cluster werden durchnummeriert, beginnend bei Spur 0 Seite 0.

Cluster	00	bezeichnet die Sektoren 1-8 von Spur 0, Seite 0.							
"	01	"	"	"	9-16	"	0,	"	0.
"	02	"	"	"	1-8	"	0,	"	1.
"	03	"	"	"	9-16	"	0,	"	1.
"	04	"	"	"	1-8	"	1,	"	0.

usw.

Der Aufbau der FAT (File Allocation Table) ist so, daß auf Spur 18 Seite 1 Sektor 1 ff. Name usw. (sh. oben) steht und in den Sektoren 14-16 die weiteren Verweise, in welcher Clusternummer die Fortsetzung der Datei steht. Der Inhalt dieser Sektoren ist aufgebaut wie ein zweidimensionales Feld, dessen Elemente gemäß der Clusternummerierung gekennzeichnet werden.

Steht z.B. im Element 37(hex.) die Zahl 35(hex.), so heißt das, daß die Fortsetzung des Files, zu dem 37 gehört, in 35 zu finden ist. Ist weiter der Inhalt von 35 beispielsweise C7, so bedeutet dies, daß dieses Cluster (35) das letzte von diesem File belegte Cluster ist und zwar mit 7 benutzten Sektoren.

Ein sehr hilfreiches Programm ist das auf der UTILITY-Diskette befindliche Programm "FILLLOC", womit die Clusterverteilung am Bildschirm sehr schön zu sehen ist und mit dem außerdem die "Allocation table" geprüft und ein Blankoformular hierfür ausgedruckt werden kann. Die beiden folgenden Tabellen sind damit erstellt.

Dateiname/	Attr./	belegte	Cluster
FORMAT	80	48 47 3E	C4
BACKUP	00	49 46 41	C1
BSCOPY	00	45 44 42	C1
DUMP	00	4C 50	C1
PCKOPF	00	43	C6
SYSCOP	80	4D 4E 4F	C3
DISCOP	80	40 3F 3C	C3
PCDUMP	80	51 52	C4
PCKOP2	00	3D	C6
DISK	80	53	C5
SPIEL1	80	54 55 56	C7
SPIEL3	80	3B	C8
SPIEL4	80	3A 38 37 34 33 1A	C4
DATEI	80	39 36 35 32 30 2F 2C	C4
KOPF	00	57	C6
DATKOP	80	58 59 5A 5B 5C 5D 5E 75	C2
QFORM	80	31 2E 2B	C4
QSYSCO	80	5F 60 61	C4
QDISCO	80	62 63 64	C5
QDUMP	80	2D 2A	C5
ERRLOC	80	65	C3
SPIEL2	80	66 67 68 74	C2
SP4	80	29 28 27 24 23	C8
S1	80	69 6A 6B 6C	C2
S2	80	26 25 22 20	C2
S3	80	6D	C8

S4	80	6E 6F 70 71 72 73 C4
SPIEL5	80	21 1F 1C C3
SPIEL6	80	1E C4
S5	80	1D 1B 18 C3
CTOOL	80	76 C2
CDEMO	80	19 C7
TA1	80	77 78 C5
Aufn 2Bas	80	17 14 C5
CT	80	79 C6
CTOOL2	80	16 C6
PRIME	80	7A C3
erato	80	15 C2
QCLUST	80	7B 7C C6
CONVER	80	13 10 11 C7
ICDAT	80	7D 7E 7F 80 81 82 C7
IDCDAT2	80	12 83 84 85 86 87 C7
ICDAT3	80	88 89 8A 8B 8C 8D C7
BIO	80	8E 8F C8
CTOOLA	00	90 C2

D a t e i v e r z e i c h n i s i n O r d n u n g

Überprüfung einer FAT mit "FILLOC"

Desweiteren ist das Wissen von Nutzen, daß die Dateien auf einer leeren DISK-BASIC-Diskette so angelegt werden, daß sie gleichmäßig um die FAT herumgruppiert werden, wobei oberhalb in fortlaufender Folge die Custer belegt werden, unterhalb (Spur 18) dagegen in Sprüngen. Der folgende Ausdruck soll dies verdeutlichen:

Drive	(1/2, ESC, +/-):	1	P-rnt, W-rite, M-od (Cursor, TAB, Value, ESC)
Side	(0/1)	:	1
Track	(0-39)	:	18
Sector	(1-16)	:	14
00	01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F		0123456789ABCDEF
00	FE FE FE FE FE FE FE FE FE FE FE FE FE FE FE		00
10	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF		10
20	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF		20
30	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF		30
40	C4 3E 3F 40 C8 42 43 44 47 46 FE FE 4D C6 4F 50		40 .>\$.BCDGF..M.OP
50	C2 52 53 54 C6 FF FF FF FF FF FF FF FF FF FF		50 .RSI.....
60	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF		60
70	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF		70
80	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF		80
90	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF		90
A0	F6 EB 00 FF 00 00 6E 73 23 72 23 3A 55 6E 01 09		A0ns#r#:Un...
B0	01 B7 28 0A EB 09 EB 73 23 72 23 3D 20 F6 EB 09		B0 ..(....s#r#:= ...
C0	23 E5 3D 32 7A 6E 2A 56 6E 5E 23 56 21 09 00 19		C0 #.=2zn*Vn^#V!...
D0	22 5A 6E E1 23 22 CA 07 22 90 0A D1 7B 95 6F 7A		D0 "Zn.#" "...ä.oz
E0	9C 67 DA 4B 45 06 03 B7 7C 1F 67 7D 1F 6F 10 F7		E0 .g.KE....ö.gü.o...
F0	7C FF 02 38 03 21 00 02 7B 95 6F 7A 9C 67 DA 4B		F0 ö..8.!...ä.oz.g.K

Beispiel für das Ablegen von Dateien auf Diskette

Die Vorgehensweise bei der Restaurierung ist also so, daß man in ein Blankoformular der FAT die aus dem noch korrekten Inhaltsverzeichnis (18,1,1) die Namen der Dateien oberhalb der Clusternummer vermerkt und dann den Fortsetzungscluster unter Berücksichtigung obiger Hinweise sucht. Hat man diesen gefunden, wird dessen Clusternummer in das aus der FAT bekannte Feld geschrieben. Dies ist solange zu wiederholen, bis der letzte Cluster gefunden ist, in dessen Feld dann noch die Anzahl der benutzten Sektoren (C+Zahl von 1-8) einzutragen ist.

Wenn man noch weiß, wie groß einige Programme in etwa waren, kann man hieraus schließen, nach wieviel Cluster man suchen muß (1 Cluster = 2KB). Hat man dann dies auch für jeden File durchgeführt, dann kann man die so restaurierte FAT auf die Sektoren 14-16 identisch zurückschreiben. Sollte bei einigen Dateien dies nicht gänzlich gelingen, so sollte man in diese Cluster FF eintragen, sodaß diese zwar verloren sind, der Rest aber wieder in Ordnung ist. Dann ist aber auch der Eintrag in Sektor 1 zu löschen. Ein "FE" bedeutet übrigens, daß dieses Cluster nicht beschrieben werden darf. Dies ist bei den Stellen für das System und der FAT der Fall.

Achtung: Auf eine Disk-BASIC-Diskette, die bereits sehr voll ist und kein Betriebssystem enthält, sollte man nur dann das System nachträglich aufspielen, wenn man sicher ist, daß die Cluster 00-0F noch nicht von anderen Dateien belegt sind. Die Überprüfung kann z.B. mit "FILLOC" durchgeführt werden.

Drive	(1/2, ESC, +/-):	2	P-rnt, W-write, M-mod	(Cursor, TAB, Value, ESC)
Side	(0/1)	: 1	-----	-----
Track	(0-39)	: 18	-----	-----
Sector	(1-16)	: 14	-----	-----
00	01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F			0123456789ABCDEF
00	80 84 4C 04 CC 40 80 00 FE 72 7E 00 43 F0 00 72			..L..\$....r0.C..r
10	00 04 4C 04 CC 40 4F 00 00 72 4C 08 40 44 80 80			10 ..L..\$0..rL.\$D..
20	C0 80 80 80 CC 00 CE 72 7F E2 4C 80 80 0E 4C 4C			20r..L...LL
30	10 F8 4C 27 F0 06 00 20 00 00 21 88 00 90 3E 3F			30 ..L'....!...>?
40	00 00 78 02 00 18 00 19 C0 00 40 3F 30 FF 13 9E			40 ..x.....\$?0...
50	7F FC 80 20 00 00 21 88 00 90 45 22 55 0A EB 22			50!...E"U.."
60	C6 07 22 7A 0A F9 22 90 0A 2A CA 07 EB CD 51 45			60 .."z..".*....QE
70	B7 ED 52 28 2B E5 21 A9 72 CD ED 2D E1 CD CA 3F			70 ..R++..!..r..-...?
80	21 9D 72 CD ED 2D 21 ED 2D 22 92 0D CD 1A 55 21			80 !..r..-!..-".....U!
90	8C 22 02 00 C3 E6 65 48 6F 77 20 6D 61 6E 79			90 ..".....eHow many
A0	20 66 69 00 45 00 00 00 00 00 00 46 4F 52 4D			A0 fi.EE.....FORM
B0	41 54 20 20 20 80 48 FF FF FF FF 53 59 53 43			B0 AT ..H.....SYSC
C0	4F 50 20 20 20 80 49 FF FF FF FF 44 49 53 43			C0 OP ..I.....DISC
D0	4F 50 20 20 20 80 45 FF FF FF FF 55 44 55 45			D0 OP ..E.....PCDU
E0	4D 50 20 20 20 80 4C FF FF FF FF 46 45 48 4C			E0 MP ..L.....FEHL
F0	20 20 20 20 20 00 4E FF FF FF 4C 47 53 20			F0 ..N.....LGS

Beispiel für eine zerstörte FAT

Drive	(1/2, ESC, +/-):	2	P-rnt, W-write, M-od (Cursor, TAB, Value, ESC)	
Side	(0/1)	: 1	-----	
Track	(0-39)	: 18		
Sector	(1-16)	: 1		
00	01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F		0123456789ABCDEF	
00	46 4F 52 4D 41 54 20 20 20 20 20 20 20 20 20		00 FORMAT	.H.....
10	53 59 53 43 4F 50 20 20 20 20 20 20 20 20 20		10 SYSCOP	.I.....
20	44 49 53 43 4F 50 20 20 20 20 20 20 20 20 20		20 DISCOP	.E.....
30	43 44 55 4D 50 20 20 20 20 20 20 20 20 20 20		30 PCDUMP	.L.....
40	46 45 48 4C 20 20 20 20 20 20 20 20 20 20 20		40 FEHL	.N.....
50	47 53 20 20 20 20 20 20 20 20 20 20 20 20 20		50 LGS	.A.....
60	47 53 32 20 20 20 20 20 20 20 20 20 20 20 20		60 LGS2	.0.....
70	47 53 33 20 20 20 20 20 20 20 20 20 20 20 20		70 LGS3	.>.....
80	42 45 4E 45 4E 20 20 20 20 20 20 20 20 20 20		80 EBENEN	.I.....
90	5E 59 20 20 20 20 20 20 20 20 20 20 20 20 20		90 X^Y	.9.....
A0	52 49 4D 5A 41 20 20 20 20 20 20 20 20 20 20		A0 PRIMZA	.W.....
B0	27 28 78 29 20 20 20 20 20 20 20 20 20 20 20		B0 F'(x)	.6.....
C0	55 53 49 48 20 20 20 20 20 20 20 20 20 20 20		C0 MUSIK	.Z.....
D0	50 48 4C 45 20 20 20 20 20 20 20 20 20 20 20		D0 MÜHLE	.5.....
E0	49 43 54 41 43 20 20 20 20 20 20 20 20 20 20		E0 TICAC	.Ü.....
F0	4F 52 53 45 20 20 20 20 20 20 20 20 20 20 20		F0 MORSE	

Inhaltsverzeichnis der zerstörten Diskette (Sektor 1)

Drive Side	(1/2, ESC, +/-): 2	P-rnt, W-rite, M-od (Cursor, IAB, Value, ESC)	
Track	(0/1)	: 1	-----
Track	(0-39)	: 18	
Sector	(1-16)	: 2	
00	01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F		0123456789ABCDEF
00	48 41 4E 47 4D 41 20 20 20 20 80 2D FF FF FF FF		00 HANGMA
10	31 4D 41 4C 20 31 20 20 20 20 80 63 FF FF FF FF		10 1MAL 1
20	47 41 52 54 45 4E 20 20 20 20 80 66 FF FF FF FF		20 GARTEN
30	4D 32 32 37 38 34 20 20 20 20 80 29 FF FF FF FF		30 M22784
40	4D 4F 52 53 20 20 20 20 20 20 80 26 FF FF FF FF		40 MORS
50	4D 4F 52 53 50 20 20 20 20 20 80 6C FF FF FF FF		50 MORSP
60	FF 47 53 32 20 20 20 20 20 20 80 4F FF FF FF FF		60 .GS2
70	4C 47 53 33 20 20 20 20 20 20 80 3E FF FF FF FF		70 LGS3
80	45 42 45 4E 45 4E 20 20 20 20 80 54 FF FF FF FF		80 EBENEN
90	58 5E 59 20 20 20 20 20 20 20 80 39 FF FF FF FF		90 X^Y
A0	50 52 49 4D 5A 41 20 20 20 20 80 57 FF FF FF FF		A0 PRIMZA
B0	46 27 28 78 29 20 20 20 20 20 80 36 FF FF FF FF		B0 F'(x)
C0	4D 55 53 49 48 20 20 20 20 20 80 5A FF FF FF FF		C0 MUSIK
D0	4D 5D 48 4C 45 20 20 20 20 20 80 35 FF FF FF FF		D0 MÜHLE
E0	54 49 43 54 41 43 20 20 20 20 80 5C FF FF FF FF		E0 TICTAC
F0	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF		F0

Inhaltsverzeichnis der zerstörten Diskette (Sektor 2)

Ein ROM-BASIC-Programm im Steckmodul

Eine interessante Möglichkeit unter ROM-BASIC besteht darin, daß man dort erstellte Programme auch in ein Steckmodul, d.h. in EPROM's einbrennen und dann quasi mit einem Autoload durch zweimaliges Betätigen der Resettaste starten kann. Dazu benötigt man natürlich als Hardwarevoraussetzung ein EPROM-Programmiergerät, einige Programme und CP/M. Die Vorgehensweise soll im folgenden kurz beschrieben werden:

1. Zunächst wird das Programm unter ROM-BASIC oder Disk-BASIC erstellt, wobei natürlich keine Disketten-Befehle enthalten sein dürfen. Dann ist das Programm, falls es unter Disk-BASIC erstellt wurde, zunächst auf Kassette abzuspeichern und von dort unter ROM-BASIC wieder zu laden. Damit ist sichergestellt, daß der Anfang des Programms bei 6000H zu finden ist. Das Ende ist durch drei hintereinanderstehende Bytes 00 00 00 gekennzeichnet. Die Länge und die Endadresse werden später noch benötigt und sind deshalb am besten zu notieren.

Auf CP/M-Format kann das Programm gebracht werden, indem man unter ROM-BASIC im Monitorprogramm den Anfang und das Ende des Programms sucht und mit dem Move-Befehl das Programm im Speicher nach Adresse 0110H verschiebt. Damit wird zwar der Interpreter überschrieben, was aber nicht weiter schlimm ist. Nun wird das Diskettenlaufwerk eingeschaltet, eine CP/M-Systemdiskette eingelegt und mit der RESET-Taste das Betriebssystem geladen. Da dadurch der Inhalt des Speichers zumindest ab 0100H bis mindestens A000H (und sogar noch weiter) nicht gelöscht wird, kann man sofort nach der Bereitschaftsanzeige A> den Befehl SAVE 64 XXXXXXXX.YYY eingeben und mit Return starten.

Damit werden 64 Blöcke à 256 Bytes (64/4=16 kB) vom Speicherinhalt ab Adresse 0100H als Datei unter dem angegebenen Namen XXXXXXXX.YYY auf der Diskette abgespeichert. Bei kleineren Programmen ist entsprechend weniger anzugeben.

2. Mit einem Debugger oder auch einem "Diskettendoktorprogramm" ist das Programm nun so zu ergänzen, daß im Eprom folgendes steht:

```
Adresse
A000      JP   FLGCHK
A010      'Ab hier steht das ROM-BASIC-Programm
```

```

XXXX      NOP      'XXXX ist nächste Adresse nach dem Programm
FLGCHK:   LD      HL,09FFEH  'Überprüfen des Flags
          LD      A,(HL)     'in Adresse 9FFEH
          CP      41H        'und vergleichen mit 41H
          LD      (HL),41H
          INC     HL          'Adresse des nächsten Bytes
          JR      NZ,NEU     'Sprung, falls ungleich nach NEU
          LD      A,(HL)     'Nächstes Byte mit
          CP      42H        '42H vergleichen
          JR      Z,UMLAD    'Falls gleich, zur Umladeroutine
NEU:      LD      (HL),42H
          JP      0E023H     'Sprung in Systeminitialisierung
UMLAD:    LD      HL,0A010H  'Quelladresse ins HL-Reg.
          LD      DE,06000H  'Zieladresse ins DE-Reg.
          LD      BC,ENDE-0A010H 'Länge des umzulad. Programms
          LDIR                     'Und los geht's
          JP      ROMOFF-04010H 'Sprung auf das umgel. ROMOFF
ROMOFF:   LD      HL,0E468H  'In E468H ist Kopie des System-
          RES     6,(HL)     'port 10H, davon Bit 6 rücksetzen
          LD      A,(HL)     'damit wird ROM-PACK gesperrt
          OUT     (10H),A
          CALL    0EF74H     'CALL BASINT (BASIC-Interpreter)
          LD      HL,06001H  'Jetzt die Basic-Pointer für
          LD      (0D053H),HL 'Programmanfang und
          LD      HL,XXXX
          LD      (0D320H),HL 'Programmende setzen
          CALL    42B7H
          JP      0F8AH      'Zuguterletzt Einsprung in BASIC
ENDE:     NOP

```

Obige Assemblerprogrammteile sind vorher mit einem Macro-assembler oder von Hand in Z80-Code zu übersetzen und dem ROM-BASIC-Programm vorne bzw. hinten anzufügen.

3. Das so modifizierte Programm kann nun in ein oder zwei Eprom's vom Typ 2764 gebrannt und diese in die Platine eines Steckmoduls eingesetzt werden.

4. Der Start des Programms erfolgt dann nach Einstecken des Steckmoduls durch zweimaliges Betätigen der Resettaste oder nach dem Einschalten des Geräts durch nochmaliges Drücken der Resettaste, wodurch das Programm dann sofort gestartet wird.

Die Adressen der BASIC-Pointer

Für den Fall, daß man ein BASIC-Programm durch ein versehentliches Betätigen der RESET-Taste gelöscht hat, kann dieses durch Verändern der BASIC-Pointer durchaus wieder gerettet werden (nur in ROM-BASIC). Dazu sind die Zeiger, die auf den Programmanfang und das Programmende deuten auf die Adressen, die man mit Hilfe des Monitorprogramms herausfinden kann, nur auf die alten Werte zu setzen und schon ist das vermeintlich verlorene Programm wieder zum Leben erwacht. Aus der folgenden Aufstellung können die Adressen der Pointer entnommen werden:

	ROM- Vers.:5.26B	DISK-BASIC 5.26	
Programmanfangszeiger	D053	7D8	7CA
Programmendezeiger	D320	AAF	AA1
Variablenzeiger (Anfang	D322	AB1	AA3
(Ende des Variablenblocks)	D324	AB3	AA5
Höchste für BASIC benutzbare Adresse	D2D4	A63	A55
Stack	D04F	7D4	7C6

Zu beachten ist, daß in den oben angegebenen Speicherstellen das niederwertige Byte und in der nächsten das höherwertige Byte des Pointers steht.

Ein Beispiel hierzu:

Tippen Sie unter ROM-BASIC folgendes Miniprogramm ein:

```

10 PRINT "Dies ist ein Test"
20 PRINT "Dieses Programm wird trotz"
30 PRINT "RESET wieder zugänglich !"
40 GOTO 10
50 END

```

Nach dem Eintippen des Programms betätigen Sie die RESET-Taste und rufen sofort den Monitor mit mon auf. Jetzt können Sie zunächst mit d 6000,6070 sehen, daß das Programm mit Ausnahme des Programmanfangs noch komplett im Speicher steht.

Verändern Sie nun die folgenden Speicherstellen mit dem Change-Befehl:

D320: 03 zu 72
D322: 03 zu 72
D324: 03 zu 72
6001: 00 zu 18
6002: 00 zu 60

Die Änderung der Speicherstellen 6001 und 6002 ist deshalb nötig, weil die vorherigen Werte (Adresse der nächsten BASIC-zeile) durch die Programmendekennung 00 00 überschrieben wurde. Ein Verlassen des Monitors mit E und ein anschließendes LIST beweist, daß das vermeintlich verlorene Programm wieder da ist und auch wieder ausgeführt werden kann.

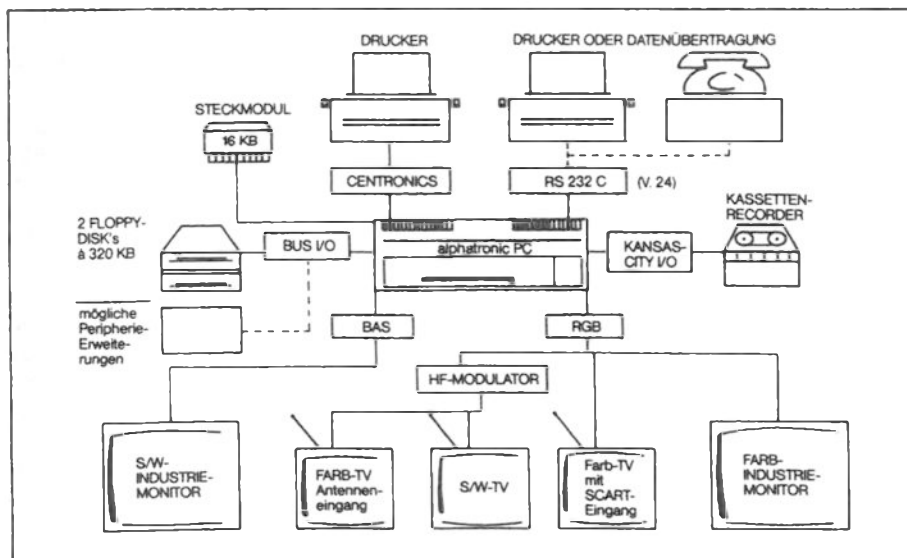
Diese Möglichkeit der Programmrückholung ist in Disk-BASIC nicht möglich, weil dort der Speicher gelöscht wird.

Der richtige Bildschirm

N. Griebinger

Einführung

Um mit einem Computer einen Dialog führen zu können, ist neben der Tastatur als Eingabeeinheit der Bildschirm als Ausgabeeinheit ein wichtiger Bestandteil der gesamten Anlage. Doch gerade im unteren Preisniveau werden die Computer als Keyboard-Modelle (gesamte Recheneinheit ist mit Tastatur in einem Gehäuse) oft ohne Bildschirm geliefert, weil es viele Möglichkeiten für den Anschluß eines Sichtgerätes gibt. Je nachdem wie strapazierfähig der Geldbeutel des Käufers ist, kann er vom normalen Fernsehgerät über einfarbige Monitore bis hin zu teuren Farbmonitoren beim Alphatronic PC alles anschließen. Mit dem Preisunterschied ist natürlich auch ein Qualitätsunterschied verbunden. In den folgenden Ausführungen sollen diese Unterschiede und Wissenswertes zu den Bildschirmanschlüssen näher erklärt werden.



Anschluß am Heimfernseher

Über einen Hochfrequenz-Modulator kann der Alphantronic PC an jeden Farbfernseher an der Antennenbuchse, die ja immer vorhanden ist, angeschlossen werden. Dazu ist lediglich der richtige Empfangskanal am Fernsehgerät z.B. auf einer nicht benutzten Programmtaste einzustellen, nachdem zuvor der PC mit dem HF-Adapter mit dem Fernseher verbunden wurde. Dies ist zweifellos die preislich günstigste Lösung, weil einerseits in fast jedem Haushalt ein TV-Gerät mindestens einmal vorhanden ist und andererseits der HF-Adapter zur Grundausstattung des PC's gehört. Diese Lösung ist demnach auch von der Bildqualität her diejenige, die am wenigsten für ein häufigeres Arbeiten besonders bei 80 Zeichen pro Zeile zu empfehlen ist. Dies kann man auch daran erkennen, daß die Farbfähigkeit des Computers nur bei 40 Zeichen und dann oft unscharf sichtbar wird.

Anders ist das allerdings, wenn der Farbfernseher über eine sogenannte SCART-Buchse (auch Peritel- oder Euronorm-Buchse genannt) verfügt. Dann kann die Qualität des Bildes wesentlich verbessert werden und es können 80 Zeichen pro Zeile auch in Farbe dargestellt werden. Für den Anschluß des PC am SCART-Eingang ist ein spezielles Kabel erforderlich, worauf später noch eingegangen wird. Bei häufigem Arbeiten am Computer oder, wenn um die Nutzungsart des Fernsehers (Krimi oder Textprogramm) keine Einigkeit besteht, sollte die Anschaffung eines Monitors in Erwägung gezogen werden.

Anschluß an einem einfarbigen Monitor

Für professionelle Arbeiten wie Textverarbeitung, Kalkulation und ähnliches ist ein monochromer Monitor sehr gut geeignet, weil es hier weniger auf Farbe als auf ein gut lesbares und scharfes Bild ankommt, das auch die Augen weniger belastet. Solche Geräte gibt es heute schon zwischen 200 und 800 DM und darüber. Man kann auch zwischen grünen, bernsteinfarbenen und weißen Bildröhren wählen und das Anschlußsignal dieser Monitore ist überwiegend ein BAS-Signal. Dieses Signal steht beim PC ebenfalls zur Verfügung, sodaß man bei der Kaufentscheidung nur noch auf die Bildwiederholfrequenz von 50 Hertz und auf den persönlichen Bildeindruck achten muß. Ein Vergleich mehrerer Bildschirme nebeneinander beim Fachhändler hilft hier mehr als die oft verwirrenden Zahlen von Grenz-Frequenz, Auflösung und Ähnlichem, weil die monochromen Monitore von da her dem Computer in der Regel überlegen sind. Dies ist nicht mehr so bei Farbmonitoren.

Anschluß an einem Farbmonitor

Um mit einer Bildröhre alle möglichen Farben erzeugen zu können, bestehen die Bildpunkte einer Farbröhre aus 3 Einzelpunkten (oder Schlitzzen bei sog. Schlitzmasken) mit den Farben Rot, Grün und Blau. Durch Variation der Intensitäten dieser drei Grundfarben kann man alle Farbtöne erzeugen. Daraus läßt sich sofort der Schluß ziehen, daß ein Farbmonitor nie so scharf sein kann wie ein einfarbiger Monitor, weil die Mischfarben erst durch Verschmelzung der drei Einzelfarben im Auge aufgrund der Entfernung entstehen.

Weiter kann man folgern, daß ein Bild umso schärfer wird, je mehr Löcher in einer Maske sind, d.h. je kleiner der Abstand zwischen den Löchern ist. Wenn der Abstand und die Löcher aber klein sind, so passen wesentlich mehr Bildpunkte auf die gleiche Fläche und es steigt damit die Auflösung. Als Maß hierfür wird oft der Loch- oder Spaltenabstand in mm angegeben, wobei eine Lochmaske einer Schlitzmaske überlegen ist. Werte um 0,3 mm kennzeichnen eine hochauflösende Röhre, während eine mittlere um 0,45 mm und eine normale um 0,6 mm liegt.

Hat ein Computer eine hochauflösende Grafik (z.B. die Grafik der Fa. BICOM für den Alphatronc PC), so sollte der Bildschirm mindestens 20% bis 30% mehr Bildpunkte am Monitor zur Verfügung haben. Ist keine Grafik vorhanden, kann man aus der Zeichenmatrix (Anzahl der Punkte, die für die Darstellung eines Zeichens benutzt wird), multipliziert mit der Anzahl der darstellbaren Zeichen pro Zeile und den möglichen Zeilen, die Mindestanforderung errechnen. Für den PC errechnet sich hier mit der 8x10 Matrix und 80 Zeichen und 24 Zeilen ein Wert von 153 600 plus 20%.

Daß die Grafik meistens eine niedrigere Auflösung hat liegt daran, daß man sehr viel Video-RAM benötigt, wenn man die Farbinformation eines jeden Bildpunktes einzeln zu speichern hat. Bei der BICOM-Grafik mit 320x248 Bildpunkten ergibt sich ein Wert von 79 360. Es werden hier nämlich die Informationen zweier horizontal benachbarter Punkte zusammengefaßt und sind somit auch nur gemeinsam ansprechbar. Der Monitor muß also die von der Zeichenmatrix her berechnete Punktzahl mindestens aufweisen. Ein visueller Vergleich ist aber auch hier anzuraten. Was beim einfarbigen Monitor kaum ins Gewicht fällt, nämlich die Art des Signals, ist beim Farbmonitor von großer Bedeutung. Hier gibt es nämlich 2 bzw. sogar 3 verschiedene, die nicht oder nur mit Anpassung zu einem anderen Computersignal passen. Sie werden mit FBAS-Signal, mit analogem RGB-Signal

und mit digitalem RGB-Signal bezeichnet.

Die Charakteristika dieser Signale werden später noch erläutert. Eine Verbindung von RGB-Ausgang und FBAS-Eingang bzw. umgekehrt ist ohne sehr großen Aufwand nicht möglich. Stärker als bei den einfarbigen Bildschirmen tritt hier das Problem der Vielfalt der verschiedenen Anschlußstecker zu Tage. Desweiteren kann nicht davon ausgegangen werden (ausgen. SCART-Buchse), daß Geräte mit gleicher Steckverbindung auch gleiche Belegung der Stifte aufweisen. Hier ist, wenn nicht ein Kabel vom Händler bereits mitgeliefert wird, die Belegung der Stecker in der jeweiligen Beschreibung zu entnehmen. Die RGB-Buchsenbelegung des Alphantronic PC ist im Bedienerhandbuch, Teil A beschrieben. Außerdem finden Sie am Ende dieses Kapitels eine Übersicht gängiger Stecker.

Ein weiteres Qualitätsmerkmal eines Farbmonitors ist die Bandbreite oder Grenzfrequenz in Megahertz. Kann ein Farbfernseher beim Fernsehbetrieb maximal 4,5 MHz aufweisen, liegen die Werte für Monitore zwischen 15 und 25 MHz. Je höher der Wert ist, umso besser ist die Auflösung.

Nun noch einige Worte zu den Preisen. Beginnend um die 1000 DM für mittelauflösende Geräte, über 2000 DM für hochauflösende gibt es darüber Spitzenmonitore bis 10 000 DM. Das Hauptfeld mit guter Qualität liegt zwischen 1800 und 3000 DM. Hier sind dann noch Unterschiede zu finden, ob man am Monitor zwischen Signalarten hin und herschalten kann, ob mehrere verschiedene Eingangsbuchsen vorhanden sind oder ob ein Schwenkfuß dabei ist. In der Regel weisen die Bildschirme in dieser Preislage alle eine entspiegelte Bildröhre auf, auf die man auch bei billigeren Modellen achten sollte.

Die verschiedenen Anschlußsignale

Wie bereits oben erwähnt gibt es im Wesentlichen 5 Signalarten, die für den Anschluß eines Bildschirms am Computer benutzt werden. Für einfarbige Monitore ist es das BAS-Signal. BAS ist die Abkürzung für Bild-, Austast- und Synchron-Signal. Hier sind alle notwendigen Signale für die Ansteuerung eines Bildschirms zu einem Signal zusammengemischt und dieses wird dann über ein Koaxialkabel zum Monitor übertragen. Als Steckverbindungen sind entweder die sogenannten Cinchstecker oder BNC-Stecker üblich.

Bei Farbwiedergabe wird das Bild eigentlich aus 3 Farbbildern zusammengesetzt. Die Bildröhre benötigt hierzu also 3

Farbsignale und - damit das Bild auch ruhig und am richtigen Platz steht - Synchronisationssignale für die vertikale und horizontale Richtung. Gibt man diese von der Bildröhre verarbeitbaren Signale direkt am Computer aus, so ist leicht einzusehen, daß damit ein qualitativ hochwertiges Bild erzeugt werden kann, weil ja kaum Bauteile dazwischen sind, die das Signal durch Störungen oder Begrenzung negativ beeinflussen. In diesem Fall erhält man ein RGB-Signal wie es der Alphatronic PC ausgibt. Dieses RGB-Signal kann man nun noch in digitaler Form oder in analoger Form an den Monitor übergeben.

Wie oben schon dargelegt, werden andere Farben als rot, grün oder blau durch Veränderung der Leuchtstärken einer jeden Grundfarbe erzeugt. Soll diese Veränderung kontinuierlich vorgenommen werden, so muß man ein analoges Signal verwenden. Hat man dagegen sowieso nur die Absicht, die Grundfarben entweder leuchten zu lassen oder nicht, woraus insgesamt 8 verschiedene Farben entstehen, so kann man hierzu ein digitales RGB-Signal benutzen. Um dann doch noch mehr Farben zu erzeugen, wird heute oft mit einem zusätzlichen Signal, der Intensität, gearbeitet, womit die Leuchtstärke noch auf die Hälfte abgestuft werden kann, sodaß dann 16 Farben möglich sind. Die folgende Tabelle zeigt die Zusammensetzung der acht Farben aus den Grundfarben:

Farbe:	ROT	GRÜN	BLAU
weiß	1	1	1
gelb	1	1	0
hellblau	0	1	1
purpur	1	0	1
blau	0	0	1
grün	0	1	0
rot	1	0	0
schwarz	0	0	0

Gibt es jede dieser Farben noch mit halber Leuchtstärke, ergeben sich daraus die 16 von oben. Bei einem analogen Signal ist die Zahl der möglichen Farben unbegrenzt. Dieses Signal wird z.B. bei Bildschirmtext und damit auch beim SCART-Eingang benutzt.

Die elektrische Spezifikation der beiden RGB-Signaltypen sieht wie folgt aus:

	analoges Signal	digitales Signal
Pegel	1 Vss (=1 Volt Spitze Spitze an 75 Ohm)	5 V (TTL pos./ neg.Logik)
Farbsignale	R,G,B	R,G,B
Synchr.Sign.	Composite SYNC	HSYNC,VSYNC

Aus obiger Tabelle ist zu entnehmen, daß die elektrischen Unterschiede der beiden RGB-Signale im Pegel und im Synchronisierungssignal liegen. Ein digitales Signal kann durch Anpassung der Pegel und durch EXOR-Verknüpfung der SYNC-Signale ohne Schwierigkeiten an einen analogen Eingang angepaßt werden. Dieses Verfahren ist beim Anschluß des PC an den SCART-Eingang eines Fernsehers nötig. Die hierfür nötigen Bauelemente können dabei untergebracht werden.

War das RGB-Signal der kürzeste Weg vom Computer zur Bildröhre, so hat das FBAS-Signal schon einen "weiteren Weg" zurückzulegen. Hier müssen im Computer die Farb- und Synchronisationsinformationen zuerst vereinfacht gesagt zusammen gemischt werden, ehe sie über ein Koaxialkabel zum Monitor gelangen, wo sie wiederum in Einzelsignale zerlegt werden, bevor sie die Bildröhre steuern können. Dieses Signal ist bereits von den Videorecordern her auch als "Videosignal" bekannt. Es kann auch mit dem monochromen BAS-Signal verglichen werden, wobei noch die Farbinformationen hinzukommen.

Bezüglich der Bandbreite ist dieses Signal durchaus mit dem RGB-Signal zu vergleichen, jedoch können beim Mischen und späteren Zerlegen Störungen durch die Toleranzen der Bauelemente auftreten, sodaß die Qualität des Bildes darunter leidet. Dieses Signal wird deshalb auch nur noch bei Homecomputern unterer Preisklassen benutzt, wo die Fähigkeiten des Rechners in Bezug auf Auflösung bzw. Zeichenmatrix und Anzahl der Zeichen pro Zeile (meist nur 40) sowieso beschränkt sind. Eine Erzeugung dieses Signals aus einem RGB-Signal ist zu aufwendig und nicht sinnvoll, da es ja keine Verbesserung darstellt. Dieses Signal steht beim PC nicht zur Verfügung.

Den schließlich "weitesten Weg" vom Computer zur Bildröhre hat das HF-Signal zurückzulegen. Dieses ist ein FBAS-Signal, das noch mit einem Modulator auf Hochfrequenz aufmoduliert wird. Im Fernseher muß dann zunächst eine Demodulation stattfinden, bevor die vom Videosignal her bekannte Trennung durchgeführt werden kann. Zu den bereits erwähnten Störungen beim FBAS-Signal kommt hier eine beträchtliche Beschränkung der Band-

breite auf max. 4,5 MHz. Hierin liegt der Grund dafür, daß 80 Zeichen/Zeile bei einer 8x10 Matrix nicht mehr in Farbe übertragen werden können.

Weitere Möglichkeiten beim Alphanetronic PC

Mit Ausnahme des Anschlusses über ein FBAS-Signal waren beim PC alle anderen Möglichkeiten gegeben, sogar über die SCART-Buchse. Eine weitere interessante Möglichkeit besteht darin, das digitale RGB-Signal durch eine kleine elektronische Schaltung zu einem BAS-Signal zusammenzufassen, sodaß auf dem einfarbigen Monitor die 8 verschiedenen Farben als Stufen verschiedener Helligkeit erscheinen. Dies wird erreicht durch Dazwischenschalten von Widerständen in abgestufter Größe in die Leitungen der Grundfarben, bevor sie zu einem Signal vereint werden. Diese Methode hat den Vorteil, daß Programme wie PC-SCHACH, die normalerweise für einen Monochrom-Monitor nicht geeignet sind, weil man dort die Spielfiguren nicht unterscheiden kann, jetzt durchaus nutzbar sind. Gleichzeitig ist eine deutliche Verbesserung der Bildschirmdarstellung zu beobachten. Schaltungsbeispiele dafür wurden bereits in Fachzeitschriften, u.a. c't vorgestellt. Je nach Geschmack können die Widerstandswerte zur Dämpfung der einzelnen Farbpegel etwas variiert werden. Für Nichtbastler gibt es ein fertiges Kabel von der Fa. TRC, Nürnberg, über den Fachhandel zu kaufen. Das oben erwähnte SCART-Kabel ist bei der Fa. EVA, Sonnenstr.4, in 8520 Erlangen erhältlich.

Besonders für die Textverarbeitung sollte nicht an den verhältnismäßig geringen Kosten für z.B. das TRC-Kabel gespart werden, denn der schärfere Kontrast und die gleichmäßige Leuchtintensität aller Bildpunkte eines Zeichens wirken sich doch sehr wohltuend für das Auge aus. In der "normalen" Bildschirmdarstellung des PC's erscheinen ja die horizontal aneinandergereihten Bildpunkte intensiver als die vertikalen. Daß dieser Schönheitsfehler nicht von TA bereits im Gerät durch Einbau einiger Widerstände beseitigt wurde, liegt in dem technischen Kompromiß begründet, daß sowohl TV-Geräte als auch RGB-Monitore über den gleichen RGB-Ausgang versorgt werden müssen.

Bildschirmtest und Störstrahlung

Zusätzlich zu den bereits angeführten Eigenschaften, die ein zum Computer passender Monitor haben muß, kann man diesen durch eine Reihe kleiner Tests hauptsächlich auf die Qualität der Röhre hin vor Ort untersuchen. Dazu prüft man insbesondere die

Ränder, indem man in der 1. und letzten Zeile eine Linie mit dem Unterstrich und links und rechts mit dem Ausrufezeichen schreibt, sodaß ein Rechteck entsteht. Hier kann man dann manchmal Kurven statt Linien sehen, was auf schlechte Justierung der Ablenkspule oder billige Ausführung schließen läßt. Durch Vergleichen von ein und demselben Buchstaben in den Ecken und der Mitte des Schirmes erkennt man Randunschärfen, wenn diese nicht alle gleich aussehen.

Die Entspiegelung prüft man sehr einfach, wenn man seitlich auf die Bildröhre sieht und darin Gegenstände des hinteren Raumes nicht schemenhaft, sondern mit deutlichen Konturen erkennt.

Die am Monitor befindlichen Einstellregler für Helligkeit und Kontrast sollten auch einen weiten Einstellbereich besitzen und nicht bei vollem Ausschlag ein nur mäßig helles Bild liefern. Die heute üblichen Bildschirmgrößen liegen bei 12 bzw. 14 Zoll. Größere sind nur bei genügendem Abstand vom Monitor zu empfehlen. Kleinere sind vielleicht niedlich, aber wegen der kleinen Schrift nicht gerade gut für die Augen (besonders bei längerem Arbeiten).

Bezüglich der Störstrahlung eines Monitors ist zu sagen, daß hier auf ein Funkschutzzeichen geachtet werden sollte. In jedem Fall sollten Magnetspeichermedien wie Diskettenlaufwerke oder Festplattenlaufwerke möglichst nicht in unmittelbarer Nähe der Monitore betrieben werden, weil durch die elektromagnetischen Felder der Bildröhre Störungen der Laufwerke verursacht werden können. Deshalb sollten auch keine Disketten auf Monitore gelegt werden, auch wegen der dort entstehenden Wärme. Schließlich beeinflussen sich zwei nebeneinanderstehende Bildschirme derart, daß deutliche Streifen über das Bild laufen.

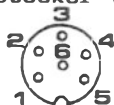
Einige Anschlußbuchsen von Farbbildschirmen

Die im Folgenden aufgezeichneten Anschlußbuchsen erheben keinen Anspruch auf Vollständigkeit, insbesondere kann die Belegung der Stifte von Hersteller zu Hersteller verschieden sein und sollte deshalb vor dem Selbstlöten eines Kabels anhand der Monitorbeschreibung genau überprüft werden.

Die wichtigsten Steckverbindungen und Kontaktstiftbelegungen

DIN-Stecker (nach 45482)

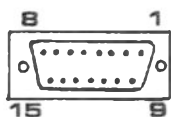
- 1 Schaltspannung + 12 V
- 2 Video-Eingang (BAS)
- 3 Masse



- 4 Ton-Eingang
- 5 Versorg.-spanng. 12 V

RGB-Stecker (15polig)

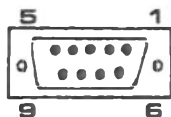
- 1 Masse
- 2 NC
- 3 Rot
- 4 Grün
- 5 Blau
- 6 NC
- 7 NC
- 8 Horizontal bzw. H/V-Mischsynchronsignal



- 9 Vertikalsynchronsignal
- 10 NC
- 11 Masse
- 12 Ton-Frequenz
- 13 Rot
- 14 Grün
- 15 Blau

RGB-Stecker (9polig)

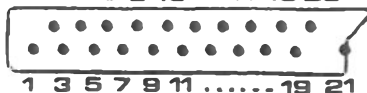
- 1 Horizontal bzw. H/V-Mischsynchronsignal
- 2 Rot
- 3 Grün
- 4 Blau



- 5 NC
- 6 Masse
- 7 NC
- 8 Vertikalsynchronsignal
- 9 Y (NC)

SCART-Buchse

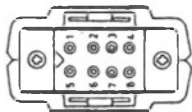
2 4 6 8 10 18 20



- 1 Audio-Ausgang B, Stereo-Kanal R
- 2 Audio-Eingang, Stereo-Kanal R
- 3 Audio-Ausgang A, Stereo-Kanal L
- 4 Audio-Masse
- 5 RGB Blau, Masse
- 6 Audio-Eingang A, Stereo-Kanal L
- 7 RGB Blau, Signal
- 8 Schaltspannung
- 9 RGB Grün, Masse
- 10 Datenleitung 2
- 11 RGB Grün, Signal
- 12 Datenleitung 1
- 13 RGB Rot, Masse
- 14 Reserve
- 15 RGB Rot, Signal
- 16 Austastsignal
- 17 Video, Masse
- 18 Austastsignal, Masse
- 19 Video-Ausgang
- 20 Video-Eingang
- 21 Schirmung/Masse

8-Pin-Buchse

- 1 NC
- 2 Rot
- 3 Grün
- 4 Blau



- 5 Masse
- 6 Masse
- 7 Horiz.-Synchronsignal
- 8 Vertikal-Synch.-sign.



Software für alphantronic

alphantronic PC

Micro-Text-Plus
Turbo-Fakt
Fakturierung/Lager
Einnahmen-/Überschußrechnung
Finanzbuchhaltung
Baustellenabrechnung
Vereinsverwaltung
Computer-Kalender
PC-Systembox

alphantronic P3-P40

Lohn-/Gehalt
Textil-Handelsvertreter
Fakturierung/Lager
Einnahmen-/Überschußrechnung
Finanzbuchhaltung
Baustellenabrechnung
Arztprogramm
Steuerberater-Kanzlei
Auftragsbearbeitung Handwerk
Auftragsbearbeitung Tiefkühlkost

alphantronic P50-P60

Fakturierung/Lager
Einnahmen-/Überschußrechnung
Auftragsbearbeitung Handwerk
Auftragsbearbeitung Tiefkühlkost
Literaturdatenbank
Unikommerz-ML
Versicherungs-Agentur

Wir sind iN

iN - BERLIN
PLZ 1

Commercial Software
Birkbuschstr. 59
1000 Berlin 41
Tel.: 030/7715214

iN - DÜSSELDORF
PLZ 4

Peter Software
Kundgensweg 18
4000 Düsseldorf
Tel.: 0211/222822

iN - KÖLN
PLZ 5

Saak Electronic
Pantaleonswall 26
5000 Köln 1
Tel.: 0221/319130

iN - WIESBADEN
PLZ 4 + 6

Softwarehaus Lange
Münzenbergstr. 22
6200 Wiesbaden
Tel.: 06122/4572

iN - KARLSRUHE
PLZ 7

Pfeifle Systemtechnik
Kolberger Str. 20d
7500 Karlsruhe
Tel.: 0721/689396

iN - NÜRNBERG
PLZ 8

SP Prem
Singenthalstr. 1
8430 Neumarkt/Opl.
Tel.: 09181/32132

S P O R T . . . S P I E L . . . S P A N N U N G . . .



mit dem S P O R T - S P I E G E L von c r d

- * Verwaltung von Sportligen mit bis zu 20 Mannschaften
- * Berechnung der Spieltage und der Spiele/Spieltag
- * automatische und manuelle Spielplanerstellung
- * Verwaltung der Spiel- und Ausweichtermine
- * Tabellenerstellung mit Heim- und Auswärtsspielwertung
- * Wahl verschiedener Sportarten (Fußball, Volleyball usw.)
- * Druckerausgabe
- * Blätterfunktion (Spieltage)
- * Fenstertechnik
- * Help-Funktion
- * Index-Funktion für Datendisketten
- * ausführliches Handbuch

178, -- DM

c r d - s o f t - 4630 bochum - kellermannsweg 5 - 0234-46 00 23

Probleme mit dem Drucker ?

N. Griebinger

Zum Anschluß eines Druckers gibt es beim Alphatronic PC gleich 2 Schnittstellen, nämlich eine serielle (RS 232C) und eine parallele (Centronic) Schnittstelle. Der Unterschied liegt darin, daß einmal die Daten bitweise nacheinander über eine Leitung mit im einfachsten Fall 3 Drähten (1 Sendedraht, 1 Empfangsdraht und Masse) zum angeschlossenen Gerät übertragen werden, im anderen Fall dagegen werden die Daten byteweise, d. h. 8 Bit auf einmal, über 8 Leitungen + Steuerleitungen übertragen. Ist die Schnittstelle des anzuschließenden Druckers bekannt, so kommt es doch gerade mit dem seriellen Anschluß zu Problemen, zu deren Beseitigung im Folgenden beigetragen werden soll.

Die Parallelschnittstelle

Als Schwierigkeit beim Anschluß an die Parallelschnittstelle stellt sich oft lediglich heraus, daß es eigentlich keine Norm für Centronics-Stecker gibt und daß der Stecker am PC nicht der häufig übliche Centronic-Stecker mit 36 Stiften ist, sondern ein Flachbandkabelstecker mit 34 Stiften. Wenn man weiß, daß die Zählweise der Stifte beider Stecker unterschiedlich ist, aber die physikalische Lage der Signale beider Stecker übereinstimmt, so ist es überhaupt kein Problem mehr, ein Kabel selbst zu löten. Stecker und Flachbandkabel gibt es heute schon in fast jedem bessersortierten Elektronikladen oder über den Elektronikversand (z.B. CONRAD Elektronik, Postfach 1180, 8452 Hirschau). Sollten auf dem Flachbandstecker keine Nummern angegeben sein, so kann man sich an einem Pfeil orientieren, der immer auf Stift 1 zeigt. Alle Stifte dieser Reihe erhalten dann die ungeraden Ziffern, die der unteren Reihe dagegen die geraden Ziffern. Beim Centronic-Stecker dagegen werden die Stifte in einer Reihe fortlaufend gezählt und dann in der nächsten fortgesetzt. Das folgende Schema zeigt die richtige Zuordnung der Stiftnummern beider Stecker:

Fl.b.St.:	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	
Cent.St.:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

!	!	!	!	!	!	!	!	!	!	!	!	!	!	!	!	!	!

!	!	!	!	!	!	!	!	!	!	!	!	!	!	!	!	!	!

Cent.St.:	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
Fl.b.St.:	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	

Für diejenigen, die nicht selbst ein Kabel anfertigen wollen, gibt es fertige Kabel über den TA-Kundendienst bzw. den TA-Fachhandel zu beziehen. Mit einem richtigen Kabel sollte dann der Drucker ohne weitere Probleme funktionieren.

Die serielle Schnittstelle

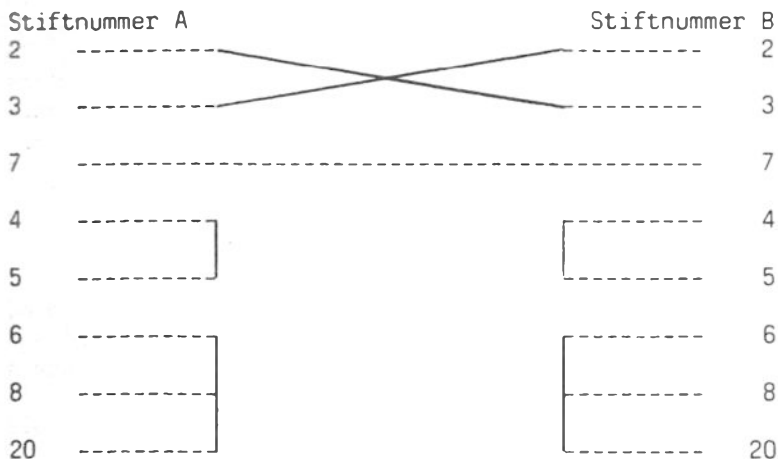
Im Gegensatz zum Parallelanschluß sind beim seriellen Druckeranschluß neben der richtigen Verkabelung noch einige weitere richtige Einstellungen nötig, damit der Drucker auch das ausdruckt, was er soll, und nicht irgendwelchen Unsinn. Ist der Drucker mit einem fertigen Kabel am PC angeschlossen und auf Online geschaltet, so werden nur dann die richtigen Zeichen erscheinen, wenn die Einstellung des Druckers bezüglich der folgenden Punkte mit der des PC übereinstimmt:

a) Die Übertragungsgeschwindigkeit (Baudrate) ist in beiden Geräten gleich eingestellt. Diese beträgt beim PC durch werkseitige Voreinstellungen **4800 Baud**. Zur Überprüfung ist ev. der PC zu öffnen und der im Bedienerhandbuch im Kap. 8.5.3 beschriebene Rangiersteckverbinder zu versetzen. Die Einstellung des Druckers geschieht meist über kleine Schalter und ist anhand der Druckerbedienungsanleitung zu überprüfen und gegebenenfalls zu ändern.

b) Das Übertragungsprotokoll muß übereinstimmen. Hierunter ist zu verstehen, mit wieviel Startbits, Datenbits, Stopbits ein Byte übertragen wird, ob ein Prüfbit mit übertragen wird und ob, falls ja, dieses im Falle einer geraden oder ungeraden Prüfsumme gesetzt wird. Der PC gibt normalerweise immer 1 Startbit und 2 Stopbits aus. Die anderen Bits können je nach verwendetem Programm variieren. Dabei sind die beiden am meisten gebrauchten Einstellungen entweder 7 Datenbits mit EVEN-Parity-Check oder 8 Datenbits ohne Parity-Check.

Was tun, wenn's nicht geht?

Wurde ein Drucker am PC über die serielle Schnittstelle angeschlossen und es tut sich gar nichts, so liegt das in der überwiegenden Zahl der Fälle an den Steuerleitungen bzw. dem Kabel generell. Das heißt im Klartext, daß der Drucker oder der PC nur dann Daten verarbeitet bzw. sendet, wenn das Signal einer bestimmten Steuerleitung gesetzt ist. Weiß man nicht genau, welche Steuerleitungen benutzt werden, so kann man mit einem kleinen Trick das Problem wie folgt umgehen. Durch Kurzschließen der entsprechenden Sende- mit der Empfangssteuerleitung kann man dem PC oder Drucker vortäuschen, daß das Signal von jeweiligen Partner käme. Das sieht dann in der Verdrahtung etwa so wie im folgenden Schema aus:



Sollte es mit einem derart modifizierten Kabel trotzdem noch nicht klappen, ist entweder das Gerät defekt oder der Drucker benötigt noch andere Signale. Als Literaturhilfe kann das Buch "V24/RS-232 Kommunikation" von Joe Campbell empfohlen werden, das im SYBEX-Verlag (ISBN 3-88745-075-2), erschienen ist.

Kommen dagegen am Drucker Zeichen an, die aber mit dem richtigen Text nicht übereinstimmen, so liegt dies entweder an einer falschen Baudrate oder am falschen Datenformat. Als erstes ist zu überprüfen, ob an beiden Geräten die Übertragungsgeschwindigkeit gleich eingestellt ist (beim PC wurde werksseitig 4800 voreingestellt). Ist dies der Fall, so ist zu prüfen, ob das Datenformat übereinstimmt (7 Datenbit, Even-Parity oder 8 Datenbit ohne Parity). Hier können zwischen dem BASIC-Interpreter und Textprogrammen (Microtext) Unterschiede sein! Bei manchen Druckern (z.B. TRD 7020) ist zu beachten, daß eine Änderung der Schalterstellungen erst nach einem erneuten Einschalten des Druckers übernommen und damit wirksam wird.

Werden die Zeichen mit Ausnahme der Umlaute, des "ß" und einigen Sonderzeichen richtig gedruckt, so ist das auf eine falsch eingestellte Ländervariante oder ein falsches Typenrad (nur bei Typenradruckern) zurückzuführen. Aus der Druckerbeschreibung kann die richtige Einstellung entnommen werden.

Konfigurieren von Textprogrammen

Für Textprogramme wie PCTEXT, die die Möglichkeit einer individuellen Konfiguration des angeschlossenen Druckers erlauben, ist für ein korrektes Arbeiten des Druckers das richtige Eintragen der entsprechenden Werte in Tabellen erforderlich.

Dies geschieht dort z.B. im Untermenü Dienstprogramme durch folgende Schritte:

a) Auswahl des Programmteils "A. Konfiguration Drucker".

b) Mit "A. Einstellen Druckertyp" kann man feststellen, welche Drucker bereits vorhanden sind und welche Buchstaben dafür verwendet wurden. Für einen noch nicht vorhandenen Drucker ist ein neuer Kennbuchstabe festzulegen. Bei einem bereits vorhandenen Drucker ist bei abweichendem Kennbuchstaben für "aktueller Typ" der richtige Buchstabe als "gewünschter Typ" einzutragen und damit ist bereits alles erledigt.

c) für einen neu zu konfigurierenden Drucker ist zuerst im Menü DRUCKER-KONFIGURATION der Punkt "B. Definieren Druckertyp" auszuwählen. Da sich die Konfigurationsinformation des Druckers nicht in Form einer separaten Datei auf der Diskette befindet, ist im nächsten Menü "B. Frei programmierbarer Drucker" die richtige Wahl.

d) In der nun erscheinenden Maske wird zuerst nach dem Kennbuchstaben des Druckers gefragt. Bei einem bereits vorhandenen Drucker (Ändern) wird sofort die Druckerbezeichnung für ev. Änderungen angezeigt. Im Falle eines neuen Druckers ist die Bezeichnung einzugeben. Nach Bestätigung mit der RETURN-Taste gelangt man nun an den Anfang des Feldes "Drucker initialisieren:". Hier ist einzutragen, was dem Drucker vor Beginn eines jeden Textes geschickt werden soll, um ihn z.B. auf eine bestimmte Schriftart oder einen festen linken Rand oder nur auf die Grundstellung einzustellen (falls vorher mit dem Drucker in einem anderen Programm gearbeitet wurde und der Druckerkopf z.B. sich noch nicht am Blattanfang befindet).

Um den Drucker in einen definierten Anfangszustand zu setzen, ist das Eintragen von entsprechenden Steuerzeichen (ESCAPE-Sequenzen oder ASCII-Codes unter 32) empfehlenswert, aber nicht unbedingt erforderlich. Für Rückschritt ist für fast alle Drucker der ASCII-Code 08 gültig ebenso wie 0D für Wagenrücklauf und 0A für Zeilenvorschub. Bei Druckern, die keine eigene Steuersequenz für "Text halbe Zeile tiefstellen" haben, läßt sich bei der Initialisierung z.B. auf einen Zeilenvorschub von einer halben Zeile einstellen und man trägt bei Zeilenvorschub statt einem dann zwei 0A ein, wodurch durch ein 0A bei "Text halbe Zeile tiefstellen" doch diese Funktion ermöglicht wird. Das Ausfüllen der Zeile "Wagenrücklauf + Zeilenvorschub" ist nur dann erforderlich, wenn beim Drucker die getrennte Ausführung der Einzelsequenzen nicht möglich ist (Autolinefeed). Dann sind aber die beiden darüberliegenden Zeilen leer. Die Zeile "Text halbe Zeile hochstellen" wird für Quadratformeln oder ähnliches benötigt und ist mit der Sequenz aus der

Druckeranleitung auszufüllen. Gleiches gilt für die nächste Zeile, die aber auch für das Drucken von eineinhalbzeiligem Text ausgefüllt sein muß. Die Einträge für Fettdruck und Unterstreichung sind ebenfalls der Druckeranleitung zu entnehmen. Falls es keine Sequenzen hierfür gibt, kann durch die Angabe von 5F 08 im Feld Präfix der Drucker dazu gebracht werden, daß er im Unterstreichfall zuerst der Unterstrich, dann einen Rückschritt und dann den eigentlichen Buchstaben druckt. Analog hierzu ist das Feld Präfix bei Fettdruck zu benutzen. Schließlich sind nur noch die Sequenzen für 10, 12 und 15 Zeichen pro Zoll aus der Druckeranleitung einzutragen und mit Verlassen der Maske durch die Cursor-nach-unten- bzw. RETURN--Taste werden die Eingaben fest abgespeichert.

e) Bevor auf den neuen Drucker umgestellt wird, ist noch die Druckertabelle zu definieren. Der Sinn dieser Tabelle ist es, eine sogenannte Konvertiertabelle zu haben, die es ermöglicht, den im Textprogramm verwendeten Zeichencode auf einen Code des Druckers zu konvertieren. Damit ließe sich z.B. für ein Zeichen ein anderes am Drucker erzeugen oder zusammen mit der Sequenztafel eine ganze Reihe von Sequenzen mit einem Zeichen auslösen. Normalerweise wird die Tabelle jedoch eins zu eins ausgefüllt. Das heißt, daß an der Position 00 auch 00, 01 auch 01 usw. eingetragen wird, was einer direkten Druckeransteuerung ohne Umsetzung irgendwelcher Zeichen gleichkommt. Würde man z.B. an der Position 24 (\$-Zeichen) eine 00 eintragen, so ist dies die Markierung für das Programm, daß es in der Sequenztafel nachsehen soll und daraus die Ersatzsequenz an den Drucker schicken soll. Nur wenn also in der Druckertabelle an einer Stelle (außer 00 selbst) ein 00 steht, ist in der Sequenztafel überhaupt ein Eintrag erforderlich, wobei vor dem Doppelpunkt der zu ersetzende Zeichencode (im obigen Beispiel also 23) und danach die Ersatzsequenz steht. Hiermit lassen sich Dinge wie Umschalten in Kursivschrift durch Eingabe eines einzigen \$-Zeichens oder ähnliches auslösen, sofern der Drucker dies unterstützt.

f) nach dem Ausfüllen der Druckertabelle (und Sequenztafel) kann dann auf den Drucker umgestellt und damit gearbeitet werden.

Programme wie MICROTTEXT-PLUS arbeiten ähnlich, allerdings ohne Druckertabelle.

**Zufriedene Kunden kommen wieder! Auf Wiedersehen
bei Ihrem TA alphasonic-Spezialisten ganz in Ihrer Nähe.**

Computerberatung Koller
Beratung · Verkauf · Service
Schulung · Programm · Zubehör
Individuelle Software-Entwicklung

Schumacherring 30
8505 Röthenbach
Tel. 0911/507164



WALTER KOLLER



ALPHASOFT NÜRNBERG

Norbert Kauschinger
Welser Straße 70
8500 Nürnberg 20
Telefon: 0911-511115

DIE ALPHATRONICSPEZIALISTEN

Wir bieten Hardware, Software und Peripherie aus einer Hand. Wir kümmern uns in allen drei Bereichen um Ihre persönlichen Anforderungen. Unser Lieferprogramm umfasst z.B. komplette Lösungen in den Bereichen Dateiverwaltung, Warenwirtschaft, FIBU und DFÜ, sowie Utilities für Programmierer. Fragen Sie uns - die Spezialisten

Fakturierung für den alphaTronic PC

- * Kundenadressen speichern
- * Artikel speichern
- * Rechnungen schreiben
- * Rechnungsausgangsjournal abrufen

alphaTronic P3 bis P60 :
Auftragsverwaltung (Aufzüge)
Hausfinanzierung

Programmierung * Beratung * Verkauf

Michael Dielen

EDV
Handel und Dienstleistungen
Nernstweg 10
4000 Düsseldorf 13

C-COMPILER

**MI-C für CP/M,
CP/M 86, MS DOS**

**Nutzen Sie die Vorteile von C
auch für den**

TA Alphasonic PC

**MI-C vereint hohen Bedienungskomfort
mit hervorragender Leistung**

- Vollständige Version mit 13stelliger BCD-Arithmetik für Gleitkommazahlen
 - Erzeugt kurze und schnelle Programme, die auch in ein ROM gebraucht werden können
 - Ausgabe in 8086- oder Z80- 8080-Assemblercode
 - Kompatibel zu MAC80/L80 (MASM/LINK) v. Microsoft
 - Fehlerverfolgung mittels Trace möglich
 - Umfangreiche Bibliothek
 - UNIX-kompatibel
 - Deutsche oder englische Version lieferbar
- | | |
|-------------------------------------|----------|
| 8"-/5,25"-/3,5"-Disk + dt. Handbuch | |
| MI-C für CP/M | 445.- DM |
| MI-C für CP/M 86 oder MS-DOS | 575.- DM |
| MI-C Cross (Zielprozessor 8080/Z80) | 745.- DM |

Herbert Rose, Bogenstraße 32, 4390 Gladbeck,
Telefon 0 20 43/2 49 12 und 4 35 97

Vertrieb in Österreich:
Dr. Willibald Kraml, Microcomputer-Software,
Degengasse 27/16, A-1160 Wien

S. Kruip

Für viele Anwendungen und Programme ist es wünschenswert, die Belegung der Tastatur mit bestimmten Zeichen beeinflussen zu können. Dadurch können auch zusätzliche Sonderzeichen oder Control-Funktionen leicht zugänglich gemacht werden, da manche Tasten des PC nur einfach belegt sind. Dazu muß man wissen: Der Computer hat in seinem Arbeitsspeicher eine Tabelle, wo für jede Taste der Tastatur eine Zahl festgelegt ist, die als Zeichen auf den Bildschirm geschrieben werden soll. Diese Tabelle befindet sich im PC ab der Speicherstelle 58589 (hexadezimal: HE4DD). Folgende Abbildung veranschaulicht die zuständigen Speicherstellen. Der Zeichencode der abgebildeten Taste steht in der Speicherstelle 58589 + Zahl (&HE4DD- Zahl) unter dem Zeichen. Für die Zeichen, die in Verbindung mit der Shifttaste erscheinen, ist jeweils noch der Wert 96 zu addieren (hexadezimal &H60 abzuziehen).

																1	2	3	4	5	6				+	-	=
																95	94	93	92	91	90				12	13	14
ESC	1	2	3	4	5	6	7	8	9	0	ß	BRK			GRAPH	7	8	9									
79	49	50	51	52	53	54	55	56	57	58	48	59	87	85		07	08	09									
CTRL	q	w	e	r	t	z	u	i	o	p	ü	+	Hm			4	5	6									
78	33	39	21	34	36	41	37	25	31	32	60	61	86	I		04	05	06									
														I													
LOCK	a	s	d	f	g	h	j	k	l	ö	ä	#	<-			1	2	3									
77	17	35	20	22	23	24	26	27	28	71	63	62	84			01	02	03									
SHIFT	<	y	x	c	v	b	n	m	,	.	-	SHIFT			TAB	0											
82	67	42	40	19	38	18	30	29	68	69	70	82	83			00		I									
																		I									
c	^	<	SPACE				>				v	INS				.	<-										
76	75	74	66				73				72	81				10	84										

Die Shift-, Lock-, Graph- und Control-Tasten lassen sich nicht undefinieren. Den Speicherstellen 11, 15, 16, 34-47, 64, 65, 80, 88 und 89 sind keine Tasten zugeordnet und enthalten den Wert 255.

Die Speicherstellen kann man nun mit einer Zahl zwischen Null und 255 belegen. Meist handelt es sich dabei um ein ASCII-Zeichen. Aber auch Controlcodes lassen sich von der Tastatur aus aufrufen.

Die folgende Übersicht zeigt sämtliche unter ROM- und Disk-Basic verfügbaren Funktionen: ganz links die Codenummer dezimal und hexadezimal, dann die Tasten auf der Tastatur, die in Verbindung mit der CTRL-Taste die Funktion auslösen sowie die Bezeichnung und eine Beschreibung. Am rechten Rand ist angegeben, wie Sie die Funktion in einem Basic-Programm verwenden können.

Liste aller Control-Befehle des alphontronic-PC

ASC-Wert

Dez.	Hex.	Tasten	Bez.	Beschreibung	im Pgm:
0	00	\$	*	NUL -	-
1	01	A	!	-	-
2	02	B	"	BCK back: C.auf Anfang des 1.Wortes	-
3	03	C	#	BRK break: wie Breaktaste	STOP
4	04	D	\$	-	-
5	05	E	%	DLE löscht Zeile von C. bis Ende: CHR\$(27);"K"	-
6	06	F	&	FWW forward: C.auf nächstes Wort	-
7	07	G	'	BEL gibt Ton aus (BEEP)	CHR\$(7)
8	08	H	(	BS Zeile ab C. nach links schieben	-
9	09	I	)	HT Horizontaltabulator	CHR\$(9)
10	0A	J	*	LF Line feed: C. in nächste Zeile	CHR\$(10)
11	0B	K	+	HME home: C.in linke, obere Ecke	CHR\$(11)
12	0C	L	,	CRL Bildschirmfenster löschen	CHR\$(12)
13	0D	M	-	CR wie Return	CHR\$(13)
14	0E	N	.	EOL C auf Ende der Zeile	-
...				...	
18	12	R	2	INS wie INS-Taste: einsetzen	-
19	13	S	3	WT hält Ablauf an, wartet auf Taste	-
20	14	T	4	-	-
21	15	U	5	DLL löscht Zeile	CHR\$(27);"F"
...				...	
27	1B	Ä	;	ESC wie Escape-Taste	CHR\$(27)
28	1C	Ö	<	> Cursor nach rechts	CHR\$(28)
29	1D	Ü	=	< Cursor nach links	CHR\$(29)
30	1E	^	ß	^ Cursor nach oben	CHR\$(30)
31	1F	_	?	v Cursor nach unten	CHR\$(31)
...				...	
127	7F			DEL wie Delete-Taste	-
129	81	<1>	1	Funktionstastenbelegungen	-
bis	bis	bis	bis	ausgeben	-
140	8C	<12>	12		-
...				...	

```
153 99 - IVON Zeichendarstellung invers      CHR$(153)
154 9A - IVOF Zeichendarstellung normal     CHR$(154)
155 9B - c wie C-Taste                      CHR$(155)
- - - DBE löscht Schirm ab C.nach unten: CHR$(27);"J"
```

Das folgende Programmlisting enthält ein paar Beispiele für mögliche Tastaturänderungen. Müssen Sie zum Beispiel viele DATA-Zeilen eingeben, so ist es nützlich, wenn statt des Dezimalpunktes ein Komma auf dem Zahlenblock zur Verfügung steht. Sie können Ihre Programme gegen unbeabsichtigtes Unterbrechen schützen, indem Sie die Breaktaste sperren. Auch die frei belegbaren Funktionen können verschoben werden, etwa die Funktion (6):RUN auf die Escape-Taste. Genauso können die Funktionstasten als normale Eingabetasten benutzt werden. Statt der Tabulatorfunktion, die nicht oft benutzt wird, kann man dort von Normal nach Invers und zurück umschalten. Die Zahlen am oberen Rand der Tastatur sind entbehrlich, da ein Zahlenblock vorhanden ist. Diese Tasten können also oft benötigte Sonderzeichen, etwa die mathematischen, zur Verfügung stellen. Eine ASCII- oder länderspezifische Tastatur können Sie ebenfalls schnell definieren. Bei so vielen Möglichkeiten ist allerdings zu bedenken: An jede Tastaturbelegung muß sich der Benutzer erst gewöhnen, sie sollte also nicht zu oft geändert werden. Bei Programmen besteht die Gefahr, inkompatibel zu anderen PC's zu werden, zum Beispiel bei der Tastaturabfrage.

Beispielprogramm:

```
10 ' Änderung der Tastaturbelegung:
20 ' REM-Zeilen können weggelassen werden.
30 ' Laderoutine -----
40 READ TASTE, ZEICHEN
50 IF TASTE = 256 THEN END
60 POKE 58589!+TASTE, ZEICHEN
70 GOTO 40: ' -----
80 ' 1: Komma statt Punkt auf Zahlenblock
90 ' z.B. für Data-Eingabe
100 DATA 10,&H2C
110 ' 2: Break-Schutz
120 ' Break nur noch mit Shift '='
130 ' gilt nicht für Laden und Speichern
140 DATA 87,&H07,183,&H07,19,&HD7,110,&H03
150 ' 3: Inhalt von Funktionstaste 6
160 ' auf Escape-Taste:(RUN +Return)
170 DATA 79,&H86
180 ' 4: Invers-on auf TAB-Taste
190 ' Invers-off auf Shift-TAB
200 DATA 83,&H99,179,&H9A
```

210 ' 5: Sonderzeichen auf obere

220 ' Zahlenreihe:

230 DATA 49,&HAB,50,&HAD,51,&HDB,52,&HDC,53,&HAE,54,&HB3

240 DATA 55,&HDO,56,&HB5,57,&HB6,48,&HD3, 256,256:' Ende

Hochauflösende Grafik mit der BiCom-Nachrüstkarte

D. Reinhardt

Die BiCom-Graphikkarte ist mit derzeit (Ende 1985) etwa 1000 verkauften Exemplaren wohl das populärste Zubehör für den TAlphatronic PC. Wir erhalten des öfteren Anfragen nach näheren Details und auch viele Beispielprogramme von Anwendern. Leider können wir nicht auf alle Wünsche und Anregungen eingehen - wir haben einfach nicht die nötige Zeit dazu. Wir bemühen uns aber, zumindest alle Anfragen aus dem Ausbildungsbereich zu beantworten und dabei auch interessante Beispielprogramme (z.B. zur Verwendung der Graphik-Karte unter Turbo-PASCAL) weiterzugeben.

Dieses Buch bietet nun eine Gelegenheit, näher auf technische Details der Graphik-Karte einzugehen. Ich hoffe, daß viele Anwender daraus Anregungen für Ihre Programme gewinnen können. Damit es auch Spaß macht, habe ich (extra für dieses Buch!) etwas neues 'erfunden' - Sprite-Graphik mit der BiCom-Karte. Wie es funktioniert, wird später gezeigt.

Grundsatzfragen

Bevor ich die Funktion der BiCom-Graphikkarte beschreibe und ein Programmbeispiel für den direkten Zugriff auf den Graphik-Bildspeicher zeige, möchte ich erst einmal erläutern, welche Überlegungen zur Entwicklung der Graphik-Karte geführt haben.

Am Anfang stand die Forderung, eine hochauflösende Farbgraphik zu schaffen, die besonders für den Einsatz im Schulbereich geeignet sein sollte. Daher wurde eine Darstellung gewählt, die gleichen Bildpunktabstand in horizontaler und vertikaler Richtung ergibt. Auf diese Weise ist ohne Umrechnungsfaktoren eine verzerrungsfreie Darstellung von mathematischen Kurven auf dem Bildschirm möglich. Leider ist der Ausdruck auf einem Drucker nicht in jedem Fall verzerrungsfrei, aber zu diesem Punkt kommen wir noch später. Um den gewünschten Bildpunktabstand zu erreichen, wurde die horizontale Pixelfrequenz (beziehungsweise die Frequenz des Bildschirm-Taktgenerators) gegenüber dem ursprünglichen Zustand um etwa 10 % verringert, außerdem wurde

die Zeilenzahl um 10 % erhöht und damit die Lesbarkeit der Zeichen verbessert.

Einige Fernseher oder Video-Monitore schneiden bei der nun verbesserten Bildschirm-Ausnutzung seitlich oder (häufiger) oben bzw. unten einen Teil des Bildes ab. Das ist in jedem Fall ein Zeichen dafür, daß der Monitor nicht normgerecht eingestellt ist! Abhilfe: Das Programm 'RAHMEN.BAS' (siehe die folgenden Programmbeispiele) laden und danach Bildhöhe und Bildlage einstellen, bis das Bild vollständig angezeigt wird. Die äußeren Linien bilden ein Rechteck, welches das gesamte Bildfeld umschreibt. Die inneren Linien sollten ein Quadrat ergeben. (Zur exakten Einstellung mit einem Lineal nachmessen, maximale Abweichung bei hochwertigen Monitoren etwa 2 bis 5 %).

Aber zurück zur Technik der Graphik-Karte. Selbstverständlich kam für uns von Anfang an nur eine Vollgraphik in Frage, bei der sich also jeder beliebige Bildpunkt in jeder möglichen Farbe setzen läßt. Das erfordert einen recht großen Bildspeicher. Wenn wir 8 Farben darstellen wollen (schwarz, weiß und 6 Buntfarben), dann benötigen wir zur Codierung eines Bildpunktes 3 Bit. Diese 3 Bit entsprechen den 3 Grundfarben rot, grün und blau eines Farbmonitors. Bei 320 horizontalen und 264 vertikalen Bildpunkten ergibt das einen Speicherbedarf von $320 * 264 * 3$ Bit, das sind 253440 Bit. Bei der üblichen Speicherorganisation von 8 Bit pro Speicherplatz entspricht das also einem Speicherbedarf von 31680 Byte, oder etwa 32 KByte. Damit ist klar, daß ein besonderer Graphik-Bildspeicher erforderlich wird, unabhängig vom Programmspeicher des Rechners.

Eine weitere Schwierigkeit ergibt sich aus der großen Geschwindigkeit, mit der die Information aus dem Bildschirmspeicher auf den Bildschirm gebracht werden muß. Bekanntlich wird das Bild auf dem Fernseh-Monitor dadurch erzeugt, daß ein Elektronenstrahl sehr schnell Zeile für Zeile das Bild auf den Leuchtschirm schreibt - bei Personal Computern entsprechend der deutschen Video-Norm insgesamt 50 komplette Bilder pro Sekunde. Daraus läßt sich errechnen, daß bei 320 Bildpunkten pro Zeile nur etwa 140 Mikrosekunden pro Bildpunkt bleiben. Das ist zu schnell für die üblicherweise verwendeten Speicher-ICs. Man greift daher zu einem kleinen 'Verpackungs-Trick', um auch mit langsameren RAM-ICs einen schnellen Graphik-Speicher zu realisieren. In unserem Fall haben wir in ein Speicherwort die Daten für zwei Bildpunkte 'gepackt'. Dabei muß dann zur Darstellung von zwei Bildpunkten nur ein Speicherzugriff erfolgen, die Geschwindigkeit des Speichers braucht nur noch halb so hoch sein. Da wir pro Bildpunkt 3 Byte benötigen, und weil Speicher-

ICs mit 64 KBit üblich sind, haben wir also einen Bildschirmspeicher mit einer Kapazität von $64K * 6$ Bit verwendet. Das sind insgesamt $64 * 1024 * 6$ Bit (das große 'K' steht für die Zahl $2^{10} = 1024$) = 393216 Bit oder etwa das 1,5-fache dessen, was für unseren Graphik-Bildschirm mindestens benötigt wird.

Der Graphik-Bildspeicher

Nach diesen Vorüberlegungen sollten wir uns das Blockschaltbild des Graphik-Adapters einmal genauer ansehen.

Das Bild, das wir gerne auf dem Monitor darstellen möchten, wird in dem schon erwähnten RAM mit $64 K * 6$ Bit gespeichert. Auf dieses RAM wird auf drei Wegen zugegriffen:

Erstens wird das RAM kontinuierlich ausgelesen, um das Bild auf dem Monitor zu erzeugen. Die dazu benötigte Speicheradresse wird in einem Zähler erzeugt. Dieser Zähler wird durch die Synchronsignale des IA PC synchronisiert, so daß sich ein stehendes Bild gleichzeitig mit der Text-Darstellung ergibt. Weil immer zwei Bildpunkte (Pixel) mit einem Speicherzugriff ausgelesen werden, müssen diese noch mit einem Multiplexer nacheinander dem Video-Ausgang zugeführt werden.

Zweitens müssen wir einen Inhalt in den Bildschirmspeicher schreiben können. Dafür sind zwei Adress-Register vorhanden, die zusammen eine 16-Bit-Adresse ergeben. Bei einem Schreib-Zugriff des Rechners auf den Bildspeicher wird infolgedessen ein Byte (bzw. die davon benötigten 6 Bit für zwei Bildpunkte) an die Speicheradresse geschrieben, die durch die beiden Register adressiert wird.

Drittens wollen wir auch den Inhalt des Bildschirmspeichers wieder auslesen können. Dabei werden wieder die beiden Adressregister verwendet, und der Speicherinhalt wird über ein Treiber-IC auf den Datenbus ausgegeben.

Der Adress-Zähler läuft ohne Unterbrechung und adressiert damit kontinuierlich das Graphik-RAM. Das ist notwendig, weil die dynamischen RAM-ICs des Graphik-Speichers ein dauerndes 'Refresh' benötigen. Dynamische RAMs speichern die Information auf Kondensatoren sehr kleiner Kapazität. Diese Kondensatoren verlieren ihre Ladung, wenn nicht innerhalb von etwa 2 Millisekunden ein Speicherzugriff erfolgt. Bei jedem Zugriff auf einen Speicherplatz wird die Ladung des entsprechenden Kondensators aufgefrischt. Wegen der internen Organisation des RAM-ICs muß

die Zeilenzahl um 10 % erhöht und damit die Lesbarkeit der Zeichen verbessert.

Einige Fernseher oder Video-Monitore schneiden bei der nun verbesserten Bildschirm-Ausnutzung seitlich oder (häufiger) oben bzw. unten einen Teil des Bildes ab. Das ist in jedem Fall ein Zeichen dafür, daß der Monitor nicht normgerecht eingestellt ist! Abhilfe: Das Programm 'RAHMEN.BAS' (siehe die folgenden Programmbeispiele) laden und danach Bildhöhe und Bildlage einstellen, bis das Bild vollständig angezeigt wird. Die äußeren Linien bilden ein Rechteck, welches das gesamte Bildfeld umschreibt. Die inneren Linien sollten ein Quadrat ergeben. (Zur exakten Einstellung mit einem Lineal nachmessen, maximale Abweichung bei hochwertigen Monitoren etwa 2 bis 5 %).

Aber zurück zur Technik der Graphik-Karte. Selbstverständlich kam für uns von Anfang an nur eine Vollgraphik in Frage, bei der sich also jeder beliebige Bildpunkt in jeder möglichen Farbe setzen läßt. Das erfordert einen recht großen Bildspeicher. Wenn wir 8 Farben darstellen wollen (schwarz, weiß und 6 Buntfarben), dann benötigen wir zur Codierung eines Bildpunktes 3 Bit. Diese 3 Bit entsprechen den 3 Grundfarben rot, grün und blau eines Farbmonitors. Bei 320 horizontalen und 264 vertikalen Bildpunkten ergibt das einen Speicherbedarf von $320 * 264 * 3$ Bit, das sind 253440 Bit. Bei der üblichen Speicherorganisation von 8 Bit pro Speicherplatz entspricht das also einem Speicherbedarf von 31680 Byte, oder etwa 32 KByte. Damit ist klar, daß ein besonderer Graphik-Bildspeicher erforderlich wird, unabhängig vom Programmspeicher des Rechners.

Eine weitere Schwierigkeit ergibt sich aus der großen Geschwindigkeit, mit der die Information aus dem Bildschirmspeicher auf den Bildschirm gebracht werden muß. Bekanntlich wird das Bild auf dem Fernseh-Monitor dadurch erzeugt, daß ein Elektronenstrahl sehr schnell Zeile für Zeile das Bild auf den Leuchtschirm schreibt - bei Personal Computern entsprechend der deutschen Video-Norm insgesamt 50 komplette Bilder pro Sekunde. Daraus läßt sich errechnen, daß bei 320 Bildpunkten pro Zeile nur etwa 140 Mikrosekunden pro Bildpunkt bleiben. Das ist zu schnell für die üblicherweise verwendeten Speicher-ICs. Man greift daher zu einem kleinen 'Verpackungs-Trick', um auch mit langsameren RAM-ICs einen schnellen Graphik-Speicher zu realisieren. In unserem Fall haben wir in ein Speicherwort die Daten für zwei Bildpunkte 'gepackt'. Dabei muß dann zur Darstellung von zwei Bildpunkten nur ein Speicherzugriff erfolgen, die Geschwindigkeit des Speichers braucht nur noch halb so hoch sein. Da wir pro Bildpunkt 3 Byte benötigen, und weil Speicher-

ICs mit 64 KBit üblich sind, haben wir also einen Bildschirmspeicher mit einer Kapazität von $64K * 6$ Bit verwendet. Das sind insgesamt $64 * 1024 * 6$ Bit (das große 'K' steht für die Zahl $2^{10} = 1024$) = 393216 Bit oder etwa das 1,5-fache dessen, was für unseren Graphik-Bildschirm mindestens benötigt wird.

Der Graphik-Bildspeicher

Nach diesen Vorüberlegungen sollten wir uns das Blockschaltbild des Graphik-Adapters einmal genauer ansehen.

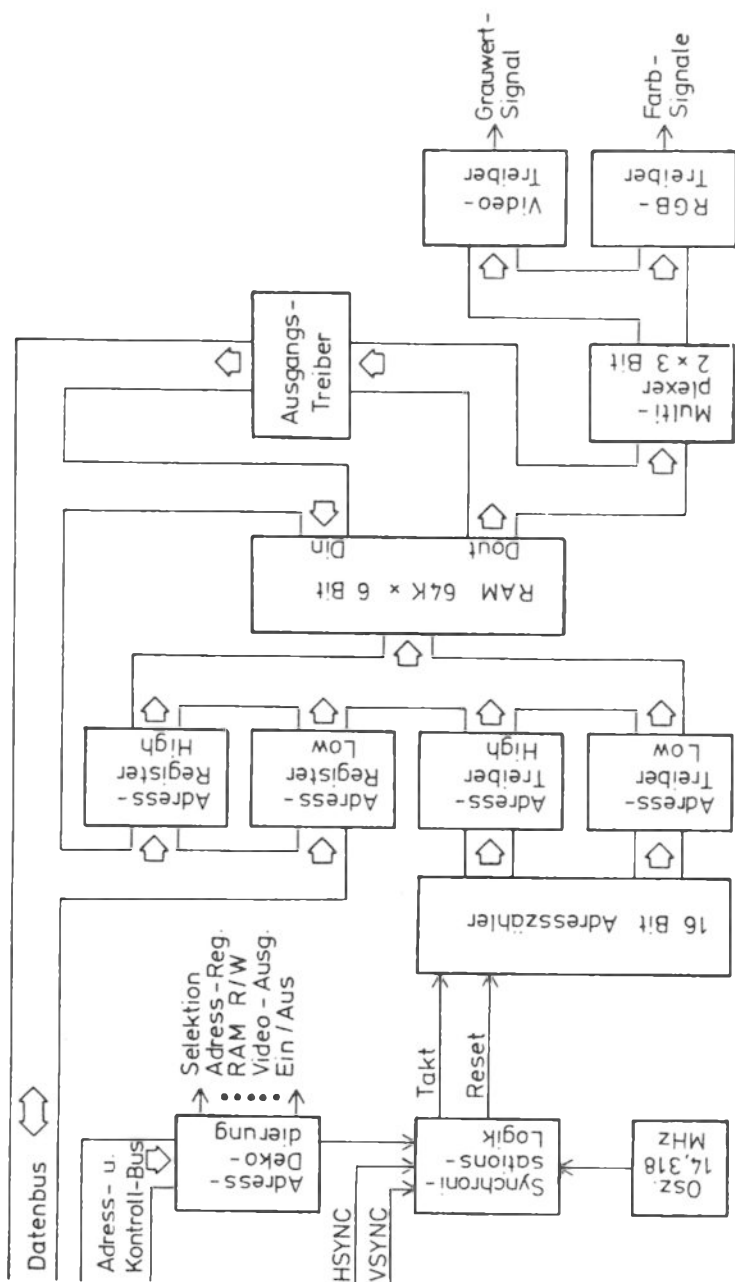
Das Bild, das wir gerne auf dem Monitor darstellen möchten, wird in dem schon erwähnten RAM mit $64 K * 6$ Bit gespeichert. Auf dieses RAM wird auf drei Wegen zugegriffen:

Erstens wird das RAM kontinuierlich ausgelesen, um das Bild auf dem Monitor zu erzeugen. Die dazu benötigte Speicheradresse wird in einem Zähler erzeugt. Dieser Zähler wird durch die Synchronsignale des IA PC synchronisiert, so daß sich ein stehendes Bild gleichzeitig mit der Text-Darstellung ergibt. Weil immer zwei Bildpunkte (Pixel) mit einem Speicherzugriff ausgelesen werden, müssen diese noch mit einem Multiplexer nacheinander dem Video-Ausgang zugeführt werden.

Zweitens müssen wir einen Inhalt in den Bildschirmspeicher schreiben können. Dafür sind zwei Adress-Register vorhanden, die zusammen eine 16-Bit-Adresse ergeben. Bei einem Schreib-Zugriff des Rechners auf den Bildspeicher wird infolgedessen ein Byte (bzw. die davon benötigten 6 Bit für zwei Bildpunkte) an die Speicheradresse geschrieben, die durch die beiden Register adressiert wird.

Drittens wollen wir auch den Inhalt des Bildschirmspeichers wieder auslesen können. Dabei werden wieder die beiden Adressregister verwendet, und der Speicherinhalt wird über ein Treiber-IC auf den Datenbus ausgegeben.

Der Adress-Zähler läuft ohne Unterbrechung und adressiert damit kontinuierlich das Graphik-RAM. Das ist notwendig, weil die dynamischen RAM-ICs des Graphik-Speichers ein dauerndes 'Refresh' benötigen. Dynamische RAMs speichern die Information auf Kondensatoren sehr kleiner Kapazität. Diese Kondensatoren verlieren ihre Ladung, wenn nicht innerhalb von etwa 2 Millisekunden ein Speicherzugriff erfolgt. Bei jedem Zugriff auf einen Speicherplatz wird die Ladung des entsprechenden Kondensators aufgefrischt. Wegen der internen Organisation des RAM-ICs muß



nun aber nicht jede einzelne Speicherzelle innerhalb von 2 Millisekunden einmal adressiert werden. Es genügt vielmehr, wenn innerhalb dieser Zeit alle 128 'Row-Adressen' des Speichers erzeugt werden. Um dieses zu gewährleisten, läuft der Adress-Zähler kontinuierlich und wird nur beim gleichzeitigen Auftreten von Horizontal- und Vertikal-Bildimpulsen auf die Adresse Null zurückgesetzt. Damit ist die Synchronität mit dem Text-Bildschirm gewährleistet. Gleichzeitig ergibt sich daraus auch, daß der erste dargestellte Punkt auf dem Bildschirm (also in der linken oberen Ecke) nicht etwa bei der Adresse Null abgespeichert ist. Die Speicheradresse wurde ja bereits während des Bild-Synchronimpulses auf Null gesetzt, und bis zur Anzeige des ersten sichtbaren Bildpunktes hat der Adresszähler weitergezählt. Da der Adresszähler auch während der Dunkelpause am Ende einer Bildschirmzeile weiterzählt, ist die Differenz der Anfangsadressen zweier aufeinanderfolgender Bildschirmzeilen auch größer als die Zahl der dargestellten Punkte-Paare.

Die tatsächliche Adress-Zuordnung geht aus dem folgenden Schema hervor:

Y = 000	1D91	1D92	1D93	...	1E31	1E32	1E33
	res.	H	L	H	L	H	L
X =		000	001	002		317	318
						319	

Y = 001	1E75	1E76	1E77	...	1F15	1F16	1F17
	res.	H	L	H	L	H	L
X =		000	001	002		317	318
						319	

Y = 263	07CD	07CE	07CF	...	086D	086E	086F
	res.	H	L	H	L	H	L
X =		000	001	002		317	318
						319	

Daraus ergibt sich, daß der erste dargestellte Bildpunkt mit der (X,Y)-Bildschirmadresse (000,000) auf dem Speicherplatz 1D92 gespeichert ist, und zwar in den oberen (High) Bits. Der folgende Speicherplatz mit der Adresse 1D93 enthält die Information für den Bildpunkt (001,000) in den unteren (Low) Bits und die Information für den Bildpunkt (002,000) in den oberen (High) Bits. Dieses Schema setzt sich fort, bis der Adresszähler nach der Adresse FFFF die Adresse 0000 ausgibt. Dabei wird

die Bildschirmausgabe fortgesetzt, bis in der letzten Zeile mit dem Bildpunkt (319,263) das Low-Nybble (die unteren Bits) des Speicherplatzes 086F erreicht ist. Der Adresszähler zählt weiter, wird mit dem Bildsynchronimpuls auf 0000 gesetzt und hat wieder die Adresse 1D92 erreicht, wenn der erste Bildpunkt ausgegeben werden soll. Links und rechts von den angezeigten Speicherbereichen ist noch jeweils ein Byte als 'reserviert' gekennzeichnet. Diese Speicherstelle wird beim Löschen des Bildschirmspeichers mit auf den Hintergrund-Code gesetzt.

Dieses Schema erscheint auf den ersten Blick kompliziert. Normalerweise muß sich der Anwender jedoch damit nicht auseinandersetzen, da die Routinen zur Adreßberechnung in den im System-PROM vordefinierten Funktionen ja bereits enthalten sind. Anders wird es aber, wenn wir direkt auf den Bildschirmspeicher zugreifen wollen, um z.B. eine Sonderfunktion zu realisieren, die nicht vordefiniert ist. Zur Erläuterung, wie man's macht, soll das folgende Sprite-Programm dienen. Damit lassen sich beliebige Zeichen definieren und auf dem Bildschirm anzeigen - allerdings mit einer kleinen Einschränkung: die Zeichen müssen eine Breite von 2 Pixeln oder Vielfachen davon haben, und sie können nur auf ungeraden X-Adressen beginnen. Diese Einschränkung wurde in Kauf genommen, um jeweils mit einem Speicherzugriff die Bildinformation für zwei nebeneinanderliegende Pixel schreiben zu können. Dabei wurde die X-Adresszuordnung so gewählt, daß beim Wert 0 das Zeichen auf den X-Adressen 1 und 2 beginnt, beim Wert 1 beginnt es auf den X-Adressen 3 und 4, usw. Die Y-Adressen können wie üblich vom 0 bis 263 gewählt werden. Das Programm überprüft die Adressen nicht auf Zulässigkeit, zu große Werte führen also dazu, daß das Zeichen 'irgendwo' auf dem Bildschirm erscheint.

Das Bitmuster für das darzustellende Zeichen wird in einer Zeichenkette übergeben. Die ersten beiden Zeichen dieses Strings enthalten die Zahl der Pixel-Paare in X-Richtung und die Zahl der Zeilen des Zeichens in Y-Richtung. Danach folgt, Zeile für Zeile, die Farbinformation für jeweils zwei Pixel pro Byte. Dabei ist zu beachten, daß die Information für das jeweils linke Pixel (mit der niedrigeren X-Adresse) im niederwertigen Nybble steht. Die Farbfolge blau-rot-grün-weiß, die mit den Farbcodes 1-2-4-7 verschlüsselt wird, steht also in zwei aufeinanderfolgenden Bytes mit den Werten 21 und 74 (jeweils hexadezimal). Auch hier gilt, daß von der Routine keine Plausibilitätskontrolle durchgeführt wird, falsche Parameter können also zu beliebig zufälligen Farbeffekten auf dem Bildschirm führen.

Da ein String maximal 255 Zeichen enthalten kann und 2 Zeichen davon für X- und Y- Zeichengröße benötigt werden, können noch 506 Bildpunkte in einem String codiert werden. Damit können also Zeichen mit mehr als 20 * 20 Bildpunkten definiert werden.

Ein Assembler-Programm zur Darstellung von Sprites

```
;*****  
;  
; SPRITE.MAC      (c) 1985 BiCom Datensysteme GmbH  
;  
; 09.09.85  D. Reinhardt  
;  
; Subroutine für BiCom-Graphik  
; zur Ausgabe von freidefinierten Zeichen  
;  
; Der Aufruf dieser Routine ist von jedem  
; Microsoft-Basic-Programm möglich mit:  
;  
; CALL SPRITE(IX,IY,A$)  
;  
; wobei folgende Variablen übergeben werden:  
;  
; IX      horizontale Position des oberen linken Pixels,  
;          in Vielfachen von 2 Bildpunkten (0 ... 158)  
; IY      vertikale Position des oberen linken Pixels  
;          (0 ... 263)  
; A$      String zur Definition des Zeichens, Aufbau:  
;          1. Byte:  Zahl der horizontalen Bildpunkte,  
;                   in Vielfachen von 2 Bildpunkten  
;          2. Byte:  Zahl der vertikalen Bildpunkte  
;          folgende: Farbcode für jeweils 2 Bildpunkte  
;                   pro Byte  
;                   (linkes Pixel im Low Nybble)  
;  
; Die übergebene Farbinformation wird pixelweise mit  
; der jeweils im Bildspeicher vorhandenen Farbinfor-  
; mation Exklusiv-Oder-verknüpft.  
;  
;*****
```

.z80

```

0000'      sprite::
0000'      D5          push    de      ; Pointer auf IY auf
                                ; Stack merken
0001'      5E          ld      e,(hl)
0002'      16 00      ld      d,0      ; IX in (de)
0004'      E1          pop     hl      ; Pointer auf IY jetzt
                                ; in (hl)
0005'      D5          push    de      ; IX auf Stack merken
0006'      5E          ld      e,(hl)
0007'      23          inc     hl
0008'      56          ld      d,(hl)  ; IY in (de)
0009'      EB          ex      de,hl   ; IY jetzt in (hl)
000A'      E5          push    hl
000B'      29          add     hl,hl
000C'      29          add     hl,hl
000D'      29          add     hl,hl   ; 8*IY
000E'      E5          push    hl
000F'      29          add     hl,hl   ; 16*IY
0010'      E5          push    hl
0011'      29          add     hl,hl   ; 32*IY
0012'      D1          pop     de
0013'      19          add     hl,de   ; 48*IY
0014'      D1          pop     de
0015'      19          add     hl,de   ; 56*IY
0016'      D1          pop     de
0017'      19          add     hl,de   ; 57*IY
0018'      29          add     hl,hl
0019'      29          add     hl,hl   ; 228*IY
001A'      D1          pop     de      ; IX wieder in (de)
001B'      19          add     hl,de   ; IX+228*IY
001C'      11 1D93    ld      de,ld93h; 1. Pixelpaar auf
                                ; Bildschirm
001F'      19          add     hl,de   ; 1D93(hex)+IX+228*IY

; (hl) enthält die Speicheradresse des ersten Pixel-
; paares (oben links) des auszugebenden Zeichens

0020'      E5          push    hl
0021'      03          inc     bc      ; String-Länge
                                ; überlesen
0022'      0A          ld      a,(bc)
0023'      6F          ld      l,a
0024'      03          inc     bc
0025'      0A          ld      a,(bc)

```

```

0026' 67          ld      h,a      ; (hl) : Anfangsadresse
                                ; des Strings
0027' 4E          ld      c,(hl)   ; Zahl der Pixelpaare
                                ; in X-Richtung
0028' 23          inc     hl
0029' 46          ld      b,(hl)   ; Größe des Zeichens
                                ; in Y-Richtung
002A' 23          inc     hl
002B' EB          ex      de,hl    ; Adresse erstes
                                ; Zeichenbyte in (de)
002C' E1          pop     hl       ; 1. Speicheradresse
                                ; wieder in (hl)

002D' C5          yloop: push    bc ; Schleifenzähler auf
                                ; Stack merken
002E' E5          push    hl       ; Anfangsadresse merken

002F' 7C          xloop: ld      a,h
0030' D3 02       out      (2),a   ; High Byte der
                                ; Adresse ausgeben
0032' 7D          ld      a,l
0033' D3 01       out      (1),a   ; Low Byte ausgeben
0035' E5          push    hl       ; Adresse auf Stack
0036' 1A          ld      a,(de)   ; ein Byte (=2 Pixel)
                                ; des Zeichens
0037' 67          ld      h,a      ; in Register h merken

0038' DB 10       wait1: in      a,(10h)
003A' 17          rla              ; disptime-bit
                                ; in carry
003B' 38 FB       jr      c,wait1  ; Warten auf
                                ; Anzeigezyklus

003D' DB 10       wait2: in      a,(10h)
003F' 17          rla
0040' 30 FB       jr      nc,wait2; Warten auf Ende des
                                ; Anzeigezyklus

; Jetzt ist soeben die Anzeige einer Bildschirmzeile
; beendet worden. Während der folgenden Dunkelpause
; kann auf den Bildschirmspeicher zugegriffen werden.

0042' DB 00       in      a,(0)   ; vorhandenen Farbcode
                                ; lesen
0044' AC          xor      h       ; EXOR-Verknüpfung mit
                                ; neuem Wert

```

```

0045'   D3 00           out      (0),a    ; in Graphik-Speicher
                                           ; schreiben

0047'   E1             pop      hl        ; Speicheradresse
                                           ; zurückladen
0048'   23             inc      hl        ; und inkrementieren
0049'   13             inc      de        ; Adresse des Zeichen-
                                           ; codes erhöhen
004A'   0D             dec      c         ; X-Zähler dekremen-
                                           ; tieren
004B'   20 E2          jr        nz,xloop

004D'   78             ld       a,b       ; Y-Zähler merken
004E'   E1             pop      hl        ; Anfangsadresse
                                           ; zurückladen
004F'   01 00E4        ld       bc,228   ; Adress-Differenz zur
                                           ; nächsten Zeile
0052'   09             add      hl,bc
0053'   C1             pop      bc        ; X-Schleifenzähler
                                           ; wiederherstellen
0054'   47             ld       b,a       ; Y-Zähler wieder
                                           ; in Register b
0055'   05             dec      b         ; Y-Zähler dekremen-
                                           ; tieren
0056'   20 D5          jr        nz,yloop

0058'   C9             ret              ; zurück zum
                                           ; aufrufenden Programm

                                end

```

Noch eine Erläuterung dazu: Um ein unschönes Flackern beim direkten Zugriff auf den Graphik-Speicher zu vermeiden, sollte nur während der Bildschirm-Dunkelpause in den Speicher geschrieben werden. Die Adresse kann schon vorher in die Adress-Register geschrieben werden. Die Dunkelpause wird durch Auswerten des DISPTIME-Statusbits festgestellt. Wenn man aber einfach während der Dunkelpause schreiben würde, dann könnte es sein, daß der Speicherzugriff erst gegen Ende der Dunkelpause erfolgt und die Bildschirmanzeige beginnt, bevor der Speicherzugriff beendet ist. Deshalb muß immer erst auf den Beginn einer Dunkelpause gewartet werden, damit genügend Zeit zur Verfügung steht.

Dieses Programm kann mit dem M80-Assembler assembliert werden und dann z.B. zu einem compilierten BASIC-Programm gelinkt

werden. Wer aber nur über einen Interpreter verfügt oder nicht gern im Compiler-BASIC programmiert, da dabei das Testen so langwierig ist, kann die Sprite-Routine auch in ein Interpreter-Programm einbinden. Das geschieht beispielsweise nach dem Muster des folgenden Programms.

Ein BASIC-Beispielprogramm für das Arbeiten mit freidefinierten Zeichen

```
10 ' TATA.BAS (C) 1985 BiCom Datensysteme GmbH
20 ' 09.09.85 D. Reinhardt
30 ' Beispiel f. freidefinierbare Zeichen mit BiCom-Graphik
40 '
50 DEFINT I
60 DEFSTR A
70 CLRGRA=&HEF20
80 SETPIX=&HEF24
90 GETPIX=&HEF28
100 DRAWLINE=&HEF2C
110 PRTLINE=&HEF30
120 PRTINIT=&HEF34
130 '
140 ' SUBROUTINE 'SPRITE' IN STRING VERPACKEN
150 '
160 AS=""
170 RESTORE 300
180 FOR I=1 TO 89
190 READ A1
200 I1=VAL("&H"+A1)
210 AS=AS+CHR$(I1)
220 NEXT I
230 S=VARPTR(AS)
240 IF S<0 THEN S=S+65536!
250 SPRITE=PEEK(S+1)+256*PEEK(S+2)
260 IF SPRITE>32767 THEN SPRITE=SPRITE-65536!
270 '
280 ' CODE FÜR SUBROUTINE 'SPRITE'
290 '
300 DATA D5,5E,16,00,E1,D5,5E,23,56,EB,E5,29,29,29,E5,29
310 DATA E5,29,D1,19,D1,19,D1,19,29,29,D1,19,11,93,10,19
320 DATA E5,03,0A,6F,03,0A,67,4E,23,46,23,EB,E1,C5,E5,7C
330 DATA D3,02,7D,D3,01,E5,1A,67,DB,10,17,38,FB,DB,10,17
340 DATA 30,FB,DB,00,AC,D3,00,E1,23,13,0D,20,E2,78,E1,01
350 DATA E4,00,09,C1,47,05,20,D5,C9
360 '

```



```
370 ' BYTEMUSTER FÜR FREIDEFINIERBARES ZEICHEN ERZEUGEN
380 '
390 ATA=""
400 RESTORE 870
410 FOR I=1 TO 9*12+2
420 READ A1
430 I1=VAL("&H"+A1)
440 ATA=ATA+CHR$(I1)
450 NEXT I
460 '
470 ' BYTEMUSTER FÜR FARBWECHELSEL
480 '
490 AB(1)=CHR$(9)+CHR$(12)+STRING$(108,&H11)
500 AB(2)=CHR$(9)+CHR$(12)+STRING$(108,&H22)
510 AB(3)=CHR$(9)+CHR$(12)+STRING$(108,&H33)
520 AB(4)=CHR$(9)+CHR$(12)+STRING$(108,&H44)
530 AB(5)=CHR$(9)+CHR$(12)+STRING$(108,&H55)
540 AB(6)=CHR$(9)+CHR$(12)+STRING$(108,&H66)
550 AB(7)=CHR$(9)+CHR$(12)+STRING$(108,&H77)
560 '
570 ' BILDSCHIRM LÖSCHEN
580 '
590 IB=0
600 CALL CLRGRA(IB) ' GRAPHIK LÖSCHEN UND EINSCHALTEN
610 PRINT CHR$(12) ' TEXT LÖSCHEN
620 '
630 ' BILDSCHIRM MIT 'TA' FÜLLEN
640 '
650 FOR IY=0 TO 264-12 STEP 12
660 FOR IX=0 TO 158-9 STEP 9
670 CALL SPRITE(IX,IY,ATA)
680 NEXT IX
690 NEXT IY
700 '
710 ' FARBWECHELSEL DURCH EXOR-VERKNÜPFUNG
720 '
730 I=INT(6*RND+1.5)
740 GOSUB 770
750 GOTO 730
760 '
770 FOR IY=0 TO 264-12 STEP 12
780 FOR IX=0 TO 158-9 STEP 9
790 CALL SPRITE(IX,IY,AB(I))
800 NEXT IX
810 NEXT IY
820 RETURN
830 '

```

```

840 ' DATEN FÜR ZEICHEN 'IA'
850 ' (9 Byte horizontal, 12 Byte vertikal)
860 '
870 DATA 09,0C
880 DATA 77,77,77,77,77,77,77,77,77
890 DATA 77,22,22,22,22,22,77,22,72
900 DATA 27,22,22,22,22,72,27,22,72
910 DATA 77,77,27,22,77,77,22,22,72
920 DATA 77,77,27,22,77,77,22,22,72
930 DATA 77,77,22,72,77,27,22,27,72
940 DATA 77,77,22,72,77,22,72,27,72
950 DATA 77,27,22,77,27,22,22,22,72
960 DATA 77,22,72,77,22,22,22,22,72
970 DATA 27,22,77,27,22,77,77,27,72
980 DATA 27,22,77,27,22,77,77,27,72
990 DATA 77,77,77,77,77,77,77,77,77

```

Anregungen zum Selbermachen

Natürlich kann man mit den Sprites noch mehr machen, als nur selbstdefinierte Zeichen darzustellen. Wegen der EXOR-Verknüpfung des neuen Bildinhaltes mit dem alten Inhalt (der zu den Farbeffekten beim Überschreiben mit einem andersfarbigen Feld führt, siehe die Wirkung von AB(1) bis AB(7)), läßt sich damit auch bewegte Graphik realisieren. Wird nämlich ein Symbol (z.B. ein Fahrzeug) mit sich selbst überschrieben, so wird das Symbol durch die EXOR-Verknüpfung gelöscht. Es kann jetzt - geringfügig versetzt - neu geschrieben werden und hat sich so scheinbar bewegt.

Dieser Effekt läßt sich mit Hilfe der Verknüpfungstabelle für die Exklusiv-Oder-Funktion noch genauer untersuchen:

A	B	A EXOR B
0	0	0
0	1	1
1	0	1
1	1	0

Aus der Tabelle sieht man, daß die EXOR-Verknüpfung zweier logischer (boolescher) Variablen genau dann den Wert 1 ergibt, wenn entweder die erste oder die zweite Variable den Wert 1 besitzt. Sind beide 0, oder sind beide 1, dann erhält man als Ergebnis der Verknüpfung den Wert 0.

Nehmen wir als Beispiel an, ein Bildpunkt habe die Farbe gelb. Der zugehörige Farbcode ist 6, oder in Binär-Schreibweise die Zahl 110 ($1*4 + 1*2 + 0*1$). Das Zeichen, das wir auf den Bildschirm schreiben wollen, soll beispielsweise an dieser Stelle die Farbe violett haben, mit dem Farbcode 3 bzw. 011 in Binär-Schreibweise. Die bitweise EXOR-Verknüpfung der beiden Farbcodes ergibt den Wert 101, das ist die Farbe 5 bzw. türkis. Wenn wir den gelben Bildpunkt wieder mit gelb überschreiben, also 110 mit 110 Exklusiv-Oder-verknüpfen, erhalten wir den Farbcode 000 - das Zeichen ist wieder gelöscht.

Bei dem gezeigten BASIC-Programm wird die Sprite-Subroutine in einem String 'verpackt'. Dem aufmerksamen Leser, der eine ältere Graphik-Karte besitzt, wird der Name einer weiteren Subroutine aufgefallen sein, nämlich PRTINIT. Diese Routine wurde neu in das Graphik-PROM aufgenommen. Die Routine wird mit einem Integer-Parameter aufgerufen und dient zum Umschalten des Drucker-Modus nach folgender Tabelle:

CALL PRTINIT(IMODE) mit Integer-Wert IMODE ergibt:

IMODE Funktion

- | | |
|---|---|
| 0 | Ausgabe über seriellen Drucker |
| 1 | Ausgabe über parallelen Drucker |
| 2 | Graphik-Ausdruck EPSON-kompatibel |
| 3 | wie 2, mit einem Punkt Zwischenraum zwischen Pixeln |
| 4 | Graphik-Ausdruck ITOH-kompatibel |
| 5 | wie 4, mit einem Punkt Zwischenraum zwischen Pixeln |

Standard-Einstellung beim Einschalten des Rechners ist: EPSON-kompatibler Graphik-Ausdruck (Mode 2), Drucker durch DIP-Schalter des PC gewählt.

Steht z.B. der DIP-Schalter auf 'serielle Druckerausgabe', und es soll mit einem ITOH-kompatiblen Drucker über die parallele Schnittstelle ausgegeben werden, dann muß PRTINIT zweimal aufgerufen werden, einmal mit dem Parameter 1 und ein weiteres Mal mit dem Parameter 4 oder 5.

Wer über einen ITOH-kompatiblen Drucker verfügt und noch das alte PROM besitzt, sollte sich direkt an BiCom wenden. Für eine kleine Austauschgebühr ist dort das geänderte PROM erhältlich.

Da manche Drucker den Bildschirminhalt nicht 1:1 ausgeben, sondern in X-Richtung mehr oder weniger stauchen (je nach Druckermodell und gewählter Zeichendichte), wurden die Modes 3

und 5 eingeführt. Dadurch wird der Ausdruck in X-Richtung um den Faktor 2 gestreckt. Es empfiehlt sich, mit verschiedenen Modes und verschiedenen Zeichendichten zu experimentieren, um den bestmöglichen Ausdruck zu erreichen. Dabei (und bei der richtigen Einstellung des Video-Monitors) kann das folgende BASIC-Programm helfen.

Ein Hilfsprogramm zum Bildschirm- und Drucker-Test

```
10 ' RAHMEN.BAS      Testprogramm für Monitor und Drucker
20 ' Zeichnen einer Bildschirm-Umrandung mit eingeschrie-
30 ' benem Quadrat
40 '
50 DEFINT I
60 CLRGRA=&HEF20
70 SETPIX=&HEF24
80 DRAWLINE=&HEF2C
90 PRTLINE=&HEF30
100 PRTINIT=&HEF34
110 '
120 ITYPE=5
130 CALL PRTINIT(ITYPE)
140 IBACK=0
150 CALL CLRGRA(IBACK)
160 PRINT CHR$(12)
170 '
180 ICOL=6 : ILIN=0
190 IX=0 : IY=0
200 CALL SETPIX(IX,IY,ICOL)
210 IX=319
220 CALL DRAWLINE(IX,IY,ILIN)
230 IY=263
240 CALL DRAWLINE(IX,IY,ILIN)
250 IX=0
260 CALL DRAWLINE(IX,IY,ILIN)
270 IY=0
280 CALL DRAWLINE(IX,IY,ILIN)
290 IX=28 : IY=0
300 CALL SETPIX(IX,IY,ICOL)
310 IY=263
320 CALL DRAWLINE(IX,IY,ILIN)
330 IX=291 : IY=0
340 CALL SETPIX(IX,IY,ICOL)
350 IY=263
360 CALL DRAWLINE(IX,IY,ILIN)
370 PRINT
```

' ITOH-Mode, mit Leerzeichen
' zwischen Pixeln

' Graphik-Bildschirm löschen
' Text-Bildschirm löschen

' gelb, durchgezogene Linien

```
380 INPUT "          Ausdrucken ? (J/N) ";A$
390 IF A$="j" OR A$="J" THEN GOTO 410
400 PRINT : END
410 INUM=24          ' 24 Graphik-Zeilen ausgeben,
420 ISTART=0        ' ab Zeile 0
430 CALL PRTLINE(ISTART,INUM)
440 INUM=0          ' Zurückschalten auf Zeilen-
450 CALL PRTLINE(ISTART,INUM) ' abstand für Textdruck
460 PRINT : END
```

Bei diesem Programm wurde PRTINIT mit dem Parameter 5 aufgerufen, also Graphik-Ausdruck für ITOH-Drucker mit einem Zwischenraum zwischen den ausgedruckten Bildpunkten gewählt. Bei EPSON-kompatiblen Druckern (z.B. beim DRH 136) entfällt dieser Aufruf.

Der Drucker muß vorher für den gewünschten Zeichenabstand (bei ITOH 10, 12 oder 17 Zeichen pro Zoll) initialisiert werden. Da die Graphik-Zeilen einen geringeren Abstand als normale Textzeilen besitzen, muß außerdem eine eventuell gewählte Seitenlänge entsprechend geändert werden. Hier hilft am besten ein bißchen probieren - wie überhaupt dieses Kapitel hoffentlich Anlaß zu eigenen Software-Experimenten ist.

Eine Anregung soll zum Schluß noch gegeben werden: wir erinnern uns, daß auf der Graphik-Karte die Speicherbereiche von Adresse 1E34 bis 1E74, von Adresse 1F18 bis 1F58 usw. nicht benutzt werden, ebenso der große Bereich von Adresse 0870 bis 1D90. Hier kann man gut die Daten für Sprite-Graphik abspeichern und dann durch eine entsprechend geänderte Sprite-Routine auf die gewünschten Adressen des Bildschirmspeichers umkopieren. Der freie Speicherbereich reicht aus, um mehrere freidefinierte Zeichensätze verfügbar zu halten.

S. Igelmann

Durch den Einbau der BiCom-Grafik-Platine erhält der alphantronic PC eine hochauflösende Farbgrafik, die vollständig unabhängig von der Textdarstellung arbeitet. Löschen oder Scrollen des Textes bewirkt keine Beeinflussung der Grafik-Darstellung. Die Auflösung beträgt 320*264 Punkte. Alle 84480 Punkte sind in 8 Farben (einschl. schwarz und weiß) darstellbar. Die Abstände der Bildpunkte sind horizontal und vertikal gleich, sodaß z.B Kreise auch tatsächlich als Kreise und nicht als Ellipsen dargestellt werden. Als Farben stehen zur Verfügung:

0	schwarz	4	grün
1	blau	5	türkis
2	rot	6	gelb
3	violett	7	weiß

Im folgenden soll nun die Programmierung dieser Grafik anhand einiger ausgewählter Beispiele in BASIC und Turbo Pascal erläutert werden. Weil aber immer wieder Schwierigkeiten beim Aufruf der Grafikroutinen auftreten, wird am Anfang noch einmal der Aufruf der Funktionen von BASIC und Pascal ausführlich erklärt.

Grundlegende Deklarationen in BASIC

Die BiCom-Grafik-Platine wird intern über einige Grafikroutinen im Systemmonitor angesprochen. Um nun diese Grafikroutinen von BASIC aus nutzen zu können, ist es notwendig, sich die Syntax des BASIC-Kommandos CALL zu vergegenwärtigen. CALL ermöglicht den Aufruf dieser Routinen. Die Syntax ist:

```
CALL SETPIX(IX,IY,ICOL)
```

Dieses Kommando ruft ein Maschinenprogramm auf, welches einen Punkt auf dem Grafikbildschirm mit der Farbe ICOL zeichnet. Die Startadresse (&HEF24) muß vorher deklariert werden. Ebenso muß jeder Variablen der Parameterliste zuvor ein Wert zugewiesen

worden sein. BASIC lädt nämlich in die Register HL, DE und BC des Z80-Prozessors die Speicheradressen dieser Variablen zur weiteren Bearbeitung durch das Maschinenprogramm.

HL : Adresse des ersten Parameters
 (in diesem Falle die X-Koordinate)
 DE : Adresse des zweiten Parameters
 (in diesem Falle die Y-Koordinate)
 BC : Adresse des dritten Parameters bzw. eines Daten-
 blocks mit weiteren Parameteradressen (in diesem
 Fall als dritter Parameter die Farbe ICOL)

Nachfolgend ein Programmkopf, der in den Zeilen 20 bis 60 einschließlich die notwendigen Deklarationen für alle in diesem Kapitel behandelten BASIC-Programme enthält.

Deklarationen

10	DEFINT I	'alle Grafikvariablen als INTEGER
20	CLRGRA =&HEF20	'Aufruf mit: CALL CLRGRA(IBACK)
30	SETPIX =&HEF24	'CALL SETPIX(IX,IY,ICOL)
40	GETPIX =&HEF28	'CALL GETPIX(IX,IY,ICOL)
50	DRAWLINE=&HEF2C	'CALL DRAWLINE(IX,IY,ILIN)
60	PRTLIN =&HEF30	'CALL PRTLIN(IPS,IPN)
70	ILIN =0	'durchgezogene Linien
72	IPS =0	'Anfangszeile für Druck (Textzeilen)
74	IPN =24	'Anzahl zu druckender Zeilen
76	ICOL =7	'Vordergrundfarbe weiß
80	IBACK=0	'Hintergrundfarbe schwarz
90	CALL CLRGRA(IBACK)	'Grafikschirm löschen/ Farbe schwarz

Die hochauflösende Grafik kann auch ein- und ausgeschaltet werden:

OUT 3,0	'Grafik aus
OUT 3,1	'Grafik ein

Nachdem wir nun die grundlegenden Deklarationen kennengelernt haben, soll ihre Anwendung in zwei kleinen Programmbeispielen gezeigt werden. Beide Programme zeichnen einen Kreis, jedoch kommen jeweils unterschiedliche Verfahren zur Anwendung.

Zunächst ein CIRCLE-Algorithmus, der mit dem Satz des Pythagoras arbeitet. Ein Kreis wird dabei in acht Kreisabschnitte zerlegt. Jeder Punkt auf diesem Kreisabschnitt ist gleichzeitig

Eckpunkt eines rechtwinkligen Dreiecks, das aus Radius, der Laufvariablen I und der zu berechnenden Seite X besteht. (Satz des Pythagoras: $I^2 + X^2 = \text{Radius}^2$) In den Zeilen 180 bis 190 wird diese einmalige Berechnung von X auf alle Kreisabschnitte übertragen.

Der Vorteil dieses Verfahrens ist, daß unabhängig vom Durchmesser des Kreises immer eine geschlossene Kreislinie gezeichnet wird. Ein weiterer Vorteil ist die Geschwindigkeit, mit der dies vonstatten geht. Eine Wurzelberechnung wird nämlich erheblich schneller ausgeführt als Sinus- und Cosinusberechnungen.

Programm Circle - Satz des Pythagoras

```

100 'Schneller Circle Algorithmus - Satz des Pythagoras
110 DEF SNG R
120 MX=160 : MY=132                'Koordinaten Mittelpunkt
130 INPUT "Farbe ",ICOL
140 INPUT "Radius",RADIUS : RQ=RADIUS*RADIUS
150 GR=RADIUS/1.414
160 FOR I=0 TO GR
170     X= SQR(RQ-I*I)
180     X1=MX-X : X2=MX+X : X3=MX-I : X4=MX+I
190     Y1=MY-I : Y2=MY+I : Y3=MY-X : Y4=MY+X
200     CALL SETPIX(X1,Y1,ICOL)    'Kreisabschnitte zeichnen
210     CALL SETPIX(X1,Y2,ICOL)
220     CALL SETPIX(X2,Y1,ICOL)
230     CALL SETPIX(X2,Y2,ICOL)
240     CALL SETPIX(X3,Y3,ICOL)
250     CALL SETPIX(X3,Y4,ICOL)
260     CALL SETPIX(X4,Y3,ICOL)
270     CALL SETPIX(X4,Y4,ICOL)
280 NEXT
    
```

Besonders geeignet für das schnelle Zeichnen größerer Kreise (Radien größer als 30) ist die folgende Routine. Sie zeichnet eigentlich strenggenommen ein Polygon. Jedoch ist die Kantenlänge so klein gewählt, daß praktisch ein Kreis gezeichnet wird. Um trotz relativ großer Schrittweite eine geschlossene Kreislinie zu erhalten, werden die Punkte auf der Kreislinie mit DRAWLINE verbunden. In den Zeilen 150-170 wird der Anfangspunkt berechnet und gesetzt. Ab Zeile 180 wird dann in einer FOR - NEXT Schleife von -PI bis PI fortlaufend mit einer bestimmten Schrittweite der Kreis gezeichnet. Durch Abändern des Nenners in der STEP-Angabe auf Zeile 180 können andere Schrittweiten gewählt werden.

Programm Circle - Polygon

```

100 DEFSNG R
110 MX=160 : MY=132                'Koordinaten Mittelpunkt
120 INPUT "Farbe ",ICOL
130 INPUT "Radius",RADIUS
140 PI=3.141592#
150 IX=MX+RADIUS*SIN(-PI)          'Anfangspunkt berechnen
160 IY=MY+RADIUS*COS(-PI)
170 CALL SETPIX(X, IY, ICOL)      'Anfangspunkt zeichnen
180   FOR L=-PI TO PI STEP PI/24  'Schrittweite
190     IX=MX+RADIUS*SIN(L)
200     IY=MY+RADIUS*COS(L)
210     CALL DRAWLINE(X, IY, ILIN) 'Kreislinie zeichnen
220   NEXT
230 IX=MX+RADIUS*SIN(-PI)          'Kreis schließen
240 IY=MY+RADIUS*COS(-PI)
250 CALL DRAWLINE(X, IY, ILIN)

```

Wir werden diese Routine später zur Erstellung einer Tortengrafik verwenden.

Freidefinierte Zeichen mit der BiCom-Grafik

Der alphantronic PC hat leider nicht wie manche andere Computer einen frei vom Anwender definierbaren Zeichensatz. Vielmehr sind die Zeichen als Punktmatrix im Zeichengenerator-PROM fest gespeichert und so für den Programmierer nicht zugänglich. Oft wünscht man es sich jedoch, in eine hochauflösende Grafik auch Buchstaben, wissenschaftliche oder mathematische Zeichen einfügen zu können. Auf dem Bildschirm ist dies kein Problem. Schwierig wird es jedoch, wenn der hochauflösende Grafikbildschirm ausgedruckt werden soll. Warum also nicht selbst Zeichen mit hochauflösender Grafik erstellen? Im folgende Programmbeispiel werden die Grundlagen dazu erläutert.

Das Zeichen wird als Punktmatrix in DATA-Zeilen gespeichert. In diesem Falle wurde eine relativ große Matrix von 11*11 Punkten gewählt, um ein Männchen zeichnen zu können. Für Buchstaben wäre z.B. eine Matrix von 4*11 Punkten ausreichend gewesen. In dieser Matrix werden die Farben durch Zahlen von 1-7 festgelegt. Andere Zeichen werden als nicht gesetzt interpretiert. Der Computer speichert zunächst die DATA-Zeilen als Stringvariablen in einem zweidimensionalen Feld ab. Dann werden diese Strings durchsucht und gegebenenfalls gesetzte Pixel zur Anzeige gebracht. Bei nicht gesetzten Pixeln wird das CALL SETPIX

Kommando übersprungen und dadurch die Hintergrundfarbe beibehalten.

Der Vorteil dieses Verfahrens ist, daß das Programm gut lesbar und auch nach Wochen noch verständlich ist. Ein weiterer Vorteil ist, daß die Breite der Zeichen und die Anzahl der Rasterzeilen frei variiert werden kann. Dadurch sind auch größere Grafikobjekte einfach zu realisieren. Der Nachteil ist der relativ hohe Speicherbedarf. Auch die Geschwindigkeit läßt für Spielanwendungen zu wünschen übrig. Wie man, allerdings unter Zuhilfenahme eines Assembler-Unterprogramms, eine schnelle Sprite-Graphik programmieren kann, wird an anderer Stelle in diesem Buch beschrieben.

Mit freidefinierten Zeichen in BASIC oder mit schneller Sprite-Graphik ergeben sich viele Möglichkeiten. Ganz eifrige Programmierer könnten auf dieser Grundlage z.B. den kompletten Zeichensatz des PC für die hochauflösende Grafik verfügbar machen, so daß Grafiken mit Text gemischt ausgedruckt werden können. Die Möglichkeiten der BiCom-Grafik würden dadurch um ein vielfaches erweitert. Man könnte Schaltbilder mit den verschiedenen vorher definierten Schaltungssymbolen erstellen.

Mögliche Programmerweiterungen:

Objekte können, indem man die Variablen IX bzw. IY mit bestimmten Streckungsfaktoren multipliziert, vergrößert oder verkleinert werden. Auch eine Drehung der Grafikobjekte ist programmtechnisch leicht zu bewerkstelligen, indem man x- und y-Koordinaten vertauscht.

Programm Freidefinierte Zeichen

100 XPOS=160 : YPOS=132	'Position d. Zeichens
110 RZ=10	'11 Rasterzeilen (0-10)
115 RS=11	'11 Rasterspalten
120 FOR I=0 TO RZ	
130 READ A(I)	'DATA einlesen
140 NEXT	
150 FOR I=0 TO RZ	
160 IY=I+YPOS	'Y Koordinate
170 FOR L=1 TO RS	'String durchsuchen
180 A=MID\$(A(I),L,1)	
190 ICOL=VAL(A)	'Farbe festlegen
200 IF ICOL=0 THEN 230	'Bei Farbe=0 überspringen
210 IX=L+XPOS	'X Koordinate berechnen

```
220          CALL SETPIX(IX,IY,ICOL)      'Punkt zeichnen
230      NEXT
240 NEXT
250 END
260 'Ab hier DATA für Zeichen (buntes Männchen)
270 DATA ---444---
280 DATA ---47--4---
290 DATA ----444----
300 DATA -----4-----
310 DATA ----333----
320 DATA ---3-5-3---
330 DATA --3--5-3---
340 DATA ---3333----
350 DATA ----3-3----
360 DATA ---3--3----
370 DATA -33--33----
```

Erstellung von großflächigen Zeichnungen

Für viele Programme wünscht man sich eine schöne Grafik als Titelbild. Es ist relativ einfach, eine geometrische Figur mit einer Funktion zu erstellen. Vor dem Erstellen z.B. einer Zeichnung schrecken jedoch viele Programmierer zurück, weil die Übertragung der Vorlage in SETPIX- und DRAWLINE- Befehle sehr mühselig ist.

Das folgende Programmbeispiel bietet jedoch eine Möglichkeit, mit relativ geringem Programmieraufwand großflächige, fein strukturierte Zeichnungen zu erstellen, die noch dazu mit großer Geschwindigkeit auf dem Bildschirm dargestellt werden.

Das Programm INDIANER zeichnet einen auf einem Baum sitzenden, spähenden Indianer. Die Koordinaten wurden folgendermaßen gewonnen:

Zuerst wurde die Zeichnung mit Durchschlagpapier auf Millimeterpapier übertragen, dann wurde neben die Zeichnung ein Koordinatenkreuz mit einer geeigneten Auflösung gelegt, um die X- und Y-Werte leicht ablesen zu können. Die abgelesenen Werte wurden schließlich als DATA-Zeilen im Programm gespeichert.

Beim Programmdurchlauf werden diese Punkte in den DATA-Zeilen der Reihe nach eingelesen (Zeile 300) und mit CALL DRAWLINE (Zeile 400) verbunden. Wenn zwei Punkte nicht miteinander verbunden werden sollen, dann sind die Koordinaten mit negativen Vorzeichen angegeben. Das Kommando CALL DRAWLINE auf Zeile 400

wird dann übersprungen. Damit können dann auch "Inseln" leicht gezeichnet werden. Die Grafik wird mit den Streckungsfaktoren SX und SY (Zeilen 100,110) in X- und Y-Richtung auf die volle Bildschirmgröße gestreckt. Selbstverständlich können auch andere Streckungsfaktoren eingesetzt werden.

Das vorliegende Beispielprogramm ermöglicht es, um das Prinzip anschaulich zu machen, die Geschwindigkeit der Grafik herabzusetzen. Auch kann man sich fortlaufend in der Bildschirmhälfte unten rechts die Koordinaten anzeigen lassen.

Aber auch der Nachteil dieses Verfahrens soll nicht verschwiegen werden: Hat man bei der Eingabe Fehler gemacht, dann ist die Fehlersuche in den DATA-Zeilen außerordentlich schwierig. Deshalb sollte man für jedes neue Bildelement eine neue DATA-Zeile schreiben und diese ausführlich mit REM dokumentieren.

Programm Indianer

```

100
110 SX=4                                'Streckungsfaktoren
120 SY=4
130 LOCATE 24,0
140 WRITE "I N D I A N E R G R A F I K"
150 LOCATE 0,3
160 INPUT "Koordinatenanzeige (EIN=1 AUS=0):",M
170 IF M<>0 AND M<>1 THEN 150
180 LOCATE 0,4
190 INPUT "Warteschleife (0-999)           :",PAUSE
200 IF PAUSE<0 OR PAUSE>999 THEN 180
210 LOCATE 0,5
220 INPUT "Farbe (0-7)                     :",ICOL
230 IF ICOL<0 OR ICOL>7 THEN 210
240 LOCATE 0,6
250 INPUT "Linienart (0-15)                :",ILIN
260 IF ILIN<0 OR ICOL>15 THEN 240
270 FOR D=0 TO 117
280   FOR P=0 TO PAUSE                    'Warteschleife
290   NEXT
300   READ RY,RX                          'DATA einlesen
310   IX= ABS(RX)*SX                      'Streckung x
320   IY=264-ABS(RY)*SY                  'Streckung y
330   IF M=0 THEN 390
340   '---- Nur wenn Anzeige der Koordinaten gewünscht! ----
350   LOCATE 55,20 : PRINT " X Y "
360   LOCATE 55,21 : PRINT USING " ### ### - Position";IX,IY

```

```

370      LOCATE 55,22 : PRINT USING " ###  ## - DATA  ";RX,RX
380      '---- Bei negativen Koordinaten DRAWLINE überspringen--
390      IF RY<0 THEN 410
400      CALL DRAWLINE(IX,IY,ILIN)
410      CALL SETPIX (IX,IY,ICOL)
420 NEXT
430 RESTORE
440 GOTO 130
450 '----- Ab hier DATA -----
460 '
470 DATA 66,00,60,03,50,07,40,08,30,10,27,8,32,0,28,0,20,10,10
480 DATA 12,0,14,0,30,5,30,1,32,0,35,2,37,2,40,5,40,6,37,7,42
490 DATA 8,43,9,45,10,60,15,70,17,80,25,80,18,64,14,55,15,54
500 DATA -15,-51,15,49,13,48,13,42,15,43,19,43,23,46,21,46,20
510 DATA 48,15,51,-15,-54,16,53,23,50,27,47,27,49,25,50,27,49
520 DATA 25,51,27,50,24,56,27,59,30,60,35,61,36,60,38,61,39,60
530 DATA 40,56,39,57,39,58,37,59,35,59,27,57,28,53,29,53,28,54
540 DATA 28,56,31,56,30,55,31,56,32,55,33,55,33,58,35,58,37,56
550 DATA 40,55,-38,-54,38,53,39,53,38,54
560 DATA -40,-55,41,54,41,50,40,48,41,46,48,46
570 DATA 52,47,54,52,55,49,50,45,40,45,40,44,37,46,33,45,40,40
580 DATA 50,41,51,40,30,39,41,40,36,43,30,39,30,37,40,37,35,36
590 DATA 35,35,40,35,35,34,40,34,30,34,25,36,25,35,23,30,16,29
600 DATA 50,21,66,30,66,0,0,0,0,80,66,80,66,0

```

Kreisdarstellungen

Eine der wesentlichen Errungenschaften des Computers und speziell der Computergrafik ist die Möglichkeit menschengerechter Aufbereitung von Zahlen, Daten und Fakten. Mit Diagrammen lassen sich Zusammenhänge leichter verdeutlichen als mit Zahlenkolonnen. Grafik wird im professionellen Einsatz hauptsächlich zur Lösung derartiger Probleme eingesetzt.

Aus der Vielzahl möglicher Darstellungsformen soll hier die Kreisdarstellung herausgegriffen werden. Die Kreis- oder auch Sektordarstellung eignet sich besonders für die Abbildung prozentualer Häufigkeiten. Die Gesamtfläche des Kreises entspricht dabei 100 Prozent. Die einzelnen Sektoren erhalten eine Größe, die den darzustellenden Prozentanteilen entspricht.

Das folgende Programm ermöglicht die Darstellung verschiedener Werte in einem Kreisdiagramm. Der Computer errechnet die Prozentwerte und trägt diese Anteile als Zentriwinkel in einem Kreis an. Die Anzahl der Werte ist dabei auf maximal zwanzig begrenzt. Eine größere Anzahl wäre zwar möglich, die Übersicht-

lichkeit der Darstellung würde jedoch erheblich darunter leiden.

Das Programm gliedert sich in einen Programmkopf mit den notwendigen Felddimensionierungen, Deklarationen etc. (hier sind die Zeilen 10-90 mit den BiCom-Grafik-Deklarationen zu ergänzen), ein Hauptprogramm und verschiedene Unterprogramme.

In der Eingaberoutine (Zeile 520-630) werden die Stammdaten eingegeben und in einem Feld 'WRT(20)' gespeichert. Eingabe eines <CR> (Carriage Return) ohne Wert beendet die Eingaberoutine vorzeitig. Der Computer bildet im Anschluß daran aus diesen Werten die Summe, und speichert die Prozentwerte in einem Feld 'PRW(20)' (Zeilen 640-740).

Die Kreisabschnitte werden in den Zeilen 420-490 errechnet und im Unterprogramm (Zeilen 930-1080) in den Kreis eingetragen. Wahlweise kann sich der Benutzer entweder die Prozentwerte oder die Stammdaten anzeigen lassen. Die PRINT USING Anweisungen auf den Zeilen 1050 und 1060 legen das Format dieser Anzeige auf bis zu drei Stellen vor und zwei Stellen hinter dem Komma fest.

Mit einem Trick wurde erreicht, daß vom Benutzer bei der INKEY\$ Abfrage nur Großbuchstaben getippt werden können. Der Befehl POKE &HE46F,1 bewirkt nämlich das Gleiche wie das Betätigen der <LOCK> Taste, mit dem Unterschied, daß die Lampe dunkel bleibt. Natürlich läßt sich dieses Programm leicht erweitern. Mit der schon zuvor erläuterten Möglichkeit, eigene Zeichen in hochauflösender Grafik zu definieren ließen sich die Zahlenwerte in die Grafik einfügen, damit sie zusammen mit dem Kreisdiagramm ausgedruckt werden können! Weiterhin wäre es denkbar, die einzelnen Kreisabschnitte mit Schraffuren zu versehen oder die Darstellung räumlich zu gestalten. Ändert man z.B. die Zeile 1120 folgendermaßen,

$$IY=MY*RADIUS*COS(L)/3$$

dann entsteht der Eindruck einer räumlichen Darstellung. Es ist allerdings zu beachten, daß durch diese Änderung keine perspektivische Umrechnung bewirkt wird.

Weil dieses Programm ein Demonstrationsprogramm ist, hat es auch seine Grenzen. Werden z.B. viele kleine Werte angegeben, dann überschreibt PRINT USING möglicherweise schon dargestellte Werte. In einem solchen Falle kann evtl. eine Formatänderung (Zeilen 1050 und 1060) Abhilfe schaffen. Trotzdem, bei allzu großer Kategorienzahl wird die Kreisdarstellung unübersicht-

lich. Man sollte sich dann Gedanken machen, ob die Daten nicht in weniger Nominalklassen unterteilt werden können.

Programm Kreisdarstellung

```

100 '
110 '----- Definitionen -----
120 '
130 DEFINT M
140 DEFDBL R
150 DIM WRT(20)
160 DIM PRD(20)
170 DIM PRW(20)
180 WIDTH 80
190 CONSOLE 0,24,0,1
200 '
210 '----- Hauptprogramm -----
220 '
230 LOCATE 25,10
240 PRINT "K R E I S D A R S T E L L U N G"
250 LOCATE 25,11
260 PRINT STRING$(31,"=")
270 LOCATE 0,23 : PRINT "Taste drücken...";
280 GOSUB 1140                                ' Tastaturabfrage
290 GOSUB 510                                ' Eingaberoutine
300 CLS
310 LOCATE 25,0
320 PRINT "K R E I S D A R S T E L L U N G"
330 GOSUB 740                                ' Kreis zeichnen
340 POKE &HE46F,1                            ' <SHIFT>+<LOCK>
350 LOCATE 0,23
360 PRINT "<W>--Werte, <P>--Prozente, <D>--Druck, <S>--Start, <
E>--Ende ?";
370 DW$=INKEY$ : IF DW$="" THEN 370 ELSE PRINT DW$;
380 IF DW$="D" THEN GOSUB 1220 : BEEP
390 IF DW$="E" THEN END
400 IF DW$="S" THEN RUN
410 IF DW$="W" OR DW$="P" THEN 420 ELSE 340
420 FOR I=1 TO IMAX
430   FOR ID=1 TO I
440     PRD(I)=PRD(I)+PRW(ID)
450   NEXT
460   PROZENT=PRD(I)
470   GOSUB 920
480   PRD(I)=0
490 NEXT

```

```

500 GOTO 340
510 '
520 '----- Eingaberoutine -----
530 '
540 CLS
550 PRINT "E I N G A B E :      Maximal 20 Werte/ <CR>--- Abbruch
560 PRINT STRING$(80,"=");
570 FOR I=1 TO 20
580     LOCATE 0,I+2
590     PRINT USING "Wert ## : ";I; : INPUT "",WRT(I)
600     IF WRT(I)=0 THEN IMAX=I-1 : I=20 : GOTO 630
610 NEXT
620 IMAX=20
630 '
640 '----- Summe/ Prozentwerte -----
650 '
660 SUMME=0
670 FOR I=1 TO IMAX
680     SUMME=SUMME+WRT(I)
690 NEXT
700 FOR I=1 TO IMAX
710     PRW(I)=WRT(I)/SUMME*100
720 NEXT
730 RETURN
740 '
750 '----- Unterprogramm: Kreis zeichnen -----
760 '
770 MX=160 : MY=132
780 RADIUS=100
790 ICOL=7
800 PI=3.141592#
810 L=-PI
820 GOSUB 1090
830 CALL SETPIX(IX,IY,ICOL)
840 FOR L=PI TO -PI STEP -(PI/24)
850     GOSUB 1090
860     CALL DRAWLINE(IX,IY,ILIN)
870 NEXT
880 L=-PI
890 GOSUB 1090
900 CALL DRAWLINE(IX,IY,ILIN)
910 RETURN
920 '
930 '----- Unterprogramm: Prozentwerte eintragen -----
940 '
950 ICOL=5
960 RADIUS=100

```



```

970 L=-PI/50*PROZENT+PI
980 CALL SETPIX(MX,MY,ICOL)
990 GOSUB 1090
1000 CALL DRAWLINE(IX,IY,ILIN)
1010 L=-PI/50*(PROZENT-PRW(I)/2)+PI
1020 RADIUS=120
1030 GOSUB 1090
1040 LOCATE IX/4-3,IY/11
1050 IF DW$="W" THEN PRINT USING "###.## ";WRT(I)
1060 IF DW$="P" THEN PRINT USING "###.##%";PRW(I)
1070 RETURN
1080 '
1090 '----- Unterprogramm: X,Y Koordinaten errechnen -----
1100 '
1110 IX=MX+RADIUS*SIN(L)
1120 IY=MY+RADIUS*COS(L)
1130 RETURN
1140 '
1150 '----- Unterprogramm: Pause - Tastaturabfrage -----
1160 '
1170 FOR PAUSE=0 TO 1500
1180 IF INKEY$>"" THEN PAUSE=1500
1190 NEXT
1200 RETURN
1210 '
1220 '----- Unterprogramm: Druck -----
1230 '
1240 IPS=0 : IPN=24
1250 CALL PRTLINE(IPS,IPN)
1260 RETURN

```

Schraffuren - Balkendiagramme

Balkendiagramme oder auch Histogramme eignen sich sehr gut für die Darstellung absoluter Häufigkeiten. Die Meßwertklassen werden auf der Abszisse (Merkmalsachse) und deren Häufigkeiten auf der Ordinatenachse angetragen.

Das Programm 'Balkendiagramme' ermöglicht die Darstellung von bis zu zwanzig Meßwertklassen in einer Grafik. Die Ordinatenachse ist in Einer-Schritten von 0 bis 100 skaliert, wodurch sich eine hohe Ablesegenauigkeit ergibt. Das Ablesen sehr weit rechts stehender Werte wird durch zehn horizontale Hilfslinien erleichtert. Zudem werden die Zahlenwerte noch einmal oben auf jedem Balken angezeigt. Die Balken erhalten jeweils eine andere Farbe.

Das Interessante an diesem Programm ist jedoch, daß die Balken unterschiedlich schraffiert werden. Es stehen insgesamt drei verschiedene Schraffurarten zur Verfügung (unten links- oben rechts, oben rechts- unten links, gekreuzt). Die Dichte der einzelnen Schraffurlinien läßt sich variieren, ebenso die Steigung der Linien. Der Vorteil von Schraffurlinien ist, daß sich mit ihnen beim Ausdruck auf einem Matrixdrucker recht ansprechende Grafiken erstellen lassen.

Das vorliegende Programm kann so nur Werte von 0-100 verarbeiten. Es wäre denkbar es so zu verändern, daß es beliebige -auch negative- Werte in umgekehrter Balkendarstellung anzeigen kann. Die Eingaberoutine könnte sicher noch komfortabler gestaltet werden. Grundsätzlich ist es auch möglich, aus diesem Programm einen Polygonzug (Häufigkeitskurve) abzuleiten.

Nun das Listing! Es ist so ausführlich dokumentiert, daß sich eine weitere Beschreibung wohl erübrigt.

Programm Balkendiagramm

```

100 '----- Definitionen -----'
110 '
120 DEFINT X
130 CONSOLE 1,22,0,0
140 DIM WRT(20)
150 '
160 '----- Hauptprogramm -----'
170 '
180 LOCATE 0,0
190 PRINT ">>>>>>>>>>>>>>>>" B A L K E N D I A G R A M
M <<<<<<<<<<<<<<<<
200 LOCATE 0,2
210 GOSUB 350
220 LOCATE 0,2,0
230 GOSUB 470
240 GOSUB 850
250 GOSUB 1110
260 LOCATE 0,23,0
270 PRINT ">>>>>>>>>" <S>--- Neustart <D>--- Druck
<E>--- Ende <<<<<<<<<<
280 GOSUB 1450
290 IF DW$="S" THEN RUN
300 IF DW$="D" THEN GOSUB 1510
310 IF DW$="E" THEN 330
320 GOTO 280
```

```

330 CONSOLE 0,24,0,1: WIDTH 80 : OUT 3,0 : END
340 '
350 '----- Eingaberoutine -----
360 '
370 LOCATE 0,1,1
380 PRINT CHR$(&HC)
390 PRINT "Eingabe:   Max. 20 Werte (0-100)      <CR>--- Abbruch
400 FOR I=1 TO 20
410     PRINT USING "Wert ## :";I; : INPUT "",WRT(I)
420     IF WRT(I)=0 THEN IMAX=I-1 : I=20 : GOTO 450
430 NEXT
440 IMAX=20
450 RETURN
460 '
470 '----- Abszisse und Ordinate zeichnen -----
480 '
490 PRINT CHR$(&HC)
500 '----- Abszisse
510 ICOL=7
520 ILIN=0
530 IX1=16 : IY1=242
540 IX2=319 : IY2=242
550 GOSUB 1390
560 '----- Ordinate
570 IX1=16 : IY1=242
580 IX2=16 : IY2=22
590 GOSUB 1390
600 '----- Skalierung
610 FOR I=100 TO 0 STEP -1
620     ICOL=7
630     ILIN=0
640     IY1=(1+10)*2.2
650     IY2=IY1
660     IX1=16
670     IX2=14
680     '----- Jede fünfte Einheit Strich=3
690     IF I MOD 5 =0 THEN IX2=13
700     '----- Jede zehnte Einheit Strich=4
710     IF I MOD 10=0 THEN IX2=12
720     GOSUB 1390
730     '----- Mit blauer Farbe Hilfslinien zeichnen
740     IX1=17
750     IX2=319
760     ILIN=1
770     ICOL=1
780     IF I<100 AND I MOD 5=0 THEN GOSUB 1390
790     LOCATE 0,(100-I)/5+2

```

```

800      '----- Skalenwerte antragen
810      IF (I MOD 10)=0 THEN PRINT USING "###";I
820 NEXT
830 RETURN
840 '
850 '----- Balken zeichnen -----
860 '
870 ILIN=0
880 FOR I=1 TO IMAX
890      '----- Für jeden Wert eine andere Farbe
900      ICOL=I MOD 6+1
910      '----- Koordinaten Fußpunkt
920      IX1=4+I*15 : IY1=241
930      IX2=IX1
940      '----- Koordinate für Balkenhöhe
950      IY2=242-WRT(I)*2.2
960      GOSUB 1390
970      '----- Wert auf Balken
980      LOCATE IX1/4+1,IY2/11-1
990      PRINT USING "###";WRT(I)
1000     '----- Wertenummer in Zeile 22
1010     LOCATE IX1/4+1,22
1020     PRINT USING "## ";I
1030     '----- relative Koordinaten/ Rest des Balkens
1040     IX2=IX1+12
1050     GOSUB 1420
1060     IY2=IY1
1070     GOSUB 1420
1080 NEXT
1090 RETURN
1100 '
1110 '----- Schraffuren -----
1120 '
1130 ILIN=1
1140 ICOL=5
1150 FOR I=1 TO 20
1160     '----- Art der Schraffur:
1170     '          X=0 - links unten -> rechts oben
1180     '          X=1 - links oben  -> rechts unten
1190     '          X=2 - kreuzweise schraffieren
1200     X=I MOD 3
1210     '----- Schraffurdichte
1220     DICHT=I MOD 6+1
1230     ICOL=DICHT
1240     '----- Laufvariable für Schraffur
1250     FOR IYS=1 TO WRT(I)*2.2 STEP DICHT
1260         IX1=4+I*15 : IY1=IYS
    
```

```

1270      IX2=IX1+12 : IY2=IY1+DICHTE
1280      '----- Koordinaten für Ecke oben rechts
1290      IF IY2>WRT(I)*2.2 THEN 1300 ELSE 1320
1300      IY2=WRT(I)*2.2
1310      IX2=(12/DICHTE)*(IY2-IY1)+IX1
1320      IY1=242-IY1 : IY2=242-IY2
1330      IF X=0 THEN GOSUB 1390
1340      IF X=1 THEN          SWAP IX1,IX2 : GOSUB 1390
1350      IF X=2 THEN GOSUB 1390 : SWAP IX1,IX2 : GOSUB 1390
1360      NEXT
1370 NEXT
1380 RETURN
1390 '----- Linie zeichnen -----
1400 '
1410 CALL SETPIX(IX1,IY1,ICOL)
1420 CALL DRAWLINE(IX2,IY2,ILIN)
1430 RETURN
1440 '
1450 '----- Tastaturabfrage -----
1460 '
1470 DW$=INKEY$
1480 IF DW$="" THEN 1450
1490 RETURN
1500 '
1510 '----- Druckroutine -----
1520 '
1530 IPS=0 : IPN=24
1540 CALL PRTLINE(IPS,IPN)
1550 RETURN
1560 '

```

Grundlegende Prozeduren in Turbo Pascal

In diesem Abschnitt sollen einige Grundlagen der Programmierung der BiCom-Grafik mit Turbo-Pascal erläutert werden. Wir haben schon gesehen, wie die Grafik in BASIC mit dem CALL-Kommando programmiert wird. Wie schon zuvor erwähnt, werden die Parameter über eine Liste übergeben. BASIC schreibt die Adressen der übergebenen Parameter automatisch der Reihe nach in die Register HL, DE und BC der Z80-CPU, so daß sie vom Maschinenprogramm bearbeitet werden können. In Turbo Pascal müssen hierzu Prozeduren mit INLINE Maschinencode formuliert werden. Die Syntax der Prozeduren sowie die Variablennamen wurden der BASIC-Syntax angeglichen.

Bei den INLINE-Maschinenprogrammen ist zu beachten, daß am Anfang der maskierbare Interrupt mit dem Assembler Befehl DI gesperrt werden sollte. Bei gesperrtem Interrupt reagiert die CPU nicht auf entsprechende Anforderungen des Interruptcontrollers. Würde dies unterlassen, könnte der Computer bei Aufruf einer Grafikprozedur außer Kontrolle geraten. Der gleichzeitige Betrieb der BiCom-Grafik und z.B. einer Softwareuhr wäre dann nicht möglich. Dieses Problem tritt nur bei Interrupt-Routinen auf, die in vermeintlich 'freie' Bereiche im oberen Teil des RAM geladen wurden. Hier stehen nämlich auch die Graphik-Routinen im System-PROM, und ein Interrupt während eines Graphik-Befehls würde nicht die gewünschte Interrupt-Routine aktivieren, sondern statt dessen unvorhersehbare Effekte in den Graphik-Routinen auslösen.

Des weiteren muß der Inhalt der Registerpaare AF,BC,DE,HL,IX und IY mit PUSH auf den Stack gelegt werden, und am Ende des Maschinenprogrammes mit POP wiederhergestellt werden. Zum Schluß wird dann der maskierbare Interrupt mit EI wieder freigegeben und wieder in das Hauptprogramm zurückgekehrt. Diese Verfahrensweise ist bei den folgenden grundlegenden Prozeduren CLRGRA, GRAFIKEIN, GRAFIKAUS, SETPIX, GETPIX, DRAWLINE und PRTLINe immer gleich. Parameter bzw. die Adressen der Parameter werden mit den entsprechenden 16-Bit Ladebefehlen an die Register der CPU übergeben. Nachdem alle benötigten Parameter übergeben sind, wird mit dem Assembler-Befehl CALL nn auf die jeweilige Grafikroutine im Systemmonitor verzweigt.

Nachfolgend nun die grundlegenden Grafik-Prozeduren. Die Prozedur CLRGRA ist mit ausführlichen Kommentaren versehen, damit das Prinzip verständlich wird. Sie löscht den Grafikbildschirm und füllt ihn mit der Hintergrundfarbe IBACK.

PROCEDURE CLRGRA(VAR IBACK:INTEGER);

```
begin
  inline
    ($f3/      (* DI          ; Interrupt sperren          *)
    $f5/      (* PUSH AF     ; Registerinhalte auf den      *)
    $c5/      (* PUSH BC     ; Stapel legen                  *)
    $d5/      (* PUSH DE     ;                               *)
    $e5/      (* PUSH HL     ;                               *)
    $dd/$e5/  (* PUSH IX     ;                               *)
    $fd/$e5/  (* PUSH IY     ;                               *)
    $2a/IBACK/ (* LDA HL,IBACK ; lade Register HL mit IBACK *)
    $cd/$20/$ef/ (* CALL EF20H ; call clrgra                *)
```

```

$fd/$el/      (* POP IY      ; Registerinhalte vom      *)
$dd/$el/      (* POP IX      ; Stapel holen              *)
$el/          (* POP HL      ;                          *)
$dl/          (* POP DE      ;                          *)
$cl/          (* POP BC      ;                          *)
$f1/          (* POP AF      ;                          *)
$fb);         (* EI          ; Interrupt freigeben      *)
end;

```

Die Prozedur SETPIX(IX,IY,ICOL) setzt einen Grafikpunkt mit der durch ICOL spezifizierten Farbe.

```
PROCEDURE SETPIX(VAR IX,IY,ICOL:INTEGER);
```

```

begin
  inline
    ($f3/$f5/$c5/$d5/$e5/$dd/$e5/
    $fd/$e5/
    $2a/IX/      (* LD HL,(IX)   ; lade Register HL mit (IX) *)
    $ed/$5b/IY/  (* LD DE,(IY)   ; lade Register DE mit (IY) *)
    $ed/$4b/ICOL/(* LD BC,(ICOL) ; lade Register BC mit (ICOL)*)
    $cd/$24/$ef/ (* CALL ef24H   ; call setpix ef24H          *)
    $fd/$el/
    $dd/$el/$el/$dl/$cl/$f1/$fb);
end;

```

Die Prozedur GETPIX(IX,IY,ICOL) liest vom Grafikpunkt IX,IY den entsprechenden Farbwert. ICOL wird entsprechend gesetzt.

```
PROCEDURE GETPIX(VAR IX,IY,ICOL: INTEGER);
```

```

begin
  inline
    ($f3/$f5/$c5/$d5/$e5/$dd/$e5/
    $fd/$e5/
    $2a/IX/      (* LD HL,(IX)   ; lade Register HL mit (IX) *)
    $ed/$5b/IY/  (* LD DE,(IY)   ; lade Register DE mit (IY) *)
    $ed/$4b/ICOL/(* LD BC,(ICOL) ; lade Register BC mit (ICOL)*)
    $cd/$28/$ef/ (* CALL ef28H   ; call getpix ef28H          *)
    $fd/$el/
    $dd/$el/$el/$dl/$cl/$f1/$fb);
end;

```

Die Prozedur DRAWLINE(IX,IY,ILIN) zeichnet eine Linie zum Zielpunkt (IX,IY). Der Anfangspunkt wird vom vorhergehenden SETPIX bzw. DRAWLINE übernommen. Die Linienart wird mit ILIN festgelegt. Sie kann von durchgehend bis grob gestrichelt (0-15) variiert werden.

```
PROCEDURE DRAWLINE(VAR IX,IY,ILIN :INTEGER);
```

```
begin
  inline
    ($f3/$f5/$c5/$d5/$e5/$dd/$e5/
    $fd/$e5/
    $2a/IX/      (* LD HL,(IX)   ; lade Register HL mit (IX) *)
    $ed/$5b/IY/  (* LD DE,(IY)   ; lade Register DE mit (IY) *)
    $ed/$4b/ILIN/(* LD BC,(ILIN) ; lade Register BC mit(ILIN) *)
    $cd/$2c/$ef/ (* CALL ef2cH   ; call Unterprogramm ef2cH  *)
    $fd/$el/
    $dd/$el/$el/$dl/$cl/$fl/$fb);
end;
```

Die Prozedur PRTLINE(IPS,IPN) erstellt eine Hardcopy des Grafikbildschirms. Durch die Variable IPS wird die Anfangszeile, durch IPN die Anzahl der auszudruckenden Zeilen, jeweils gerechnet in Textzeilen, festgelegt. Wird IPS=0 und IPN=24 gesetzt, dann wird der ganze Grafikbildschirm ausgedruckt.

```
PROCEDURE PRTLINE(VAR IPS,IPN :INTEGER);
```

```
begin
  inline
    ($f3/$f5/$c5/$d5/$e5/$dd/$e5/
    $fd/$e5/
    $2a/IPS/      (* LD HL,(IPS)  ; lade Register HL mit (IPS) *)
    $ed/$5b/IPN/  (* LD DE,(IPN)  ; lade Register DE mit (IPN) *)
    $cd/$30/$ef/  (* CALL ef30H   ; call Unterprogramm ef30H  *)
    $fd/$el/
    $dd/$el/$el/$dl/$cl/$fl/$fb);
end;
```

Mit den Prozeduren GRAFIKEIN und GRAFIKAUS läßt sich der Grafikbildschirm ein- und ausschalten, indem an die Schnittstelle 03H der Wert 01H (Grafik ein) bzw. 00H (Grafik aus) ausgegeben wird. Der Inhalt des Grafikbildschirms bleibt unbeeinflußt.


```
PROCEDURE GRAFIKEIN;
```

```
begin
  inline
    ($f3/$f5/$c5/$d5/$e5/$dd/$e5/
     $fd/$e5/
     $3e/$01/      (* LDA A,01H      ; lade Register A mit 01H      *)
     $d3/$03/      (* OUT A,03H      ; Ausgabe an Adresse 03H      *)
     $fd/$e1/
     $dd/$e1/$e1/$d1/$c1/$f1/$fb);
end;
```

```
PROCEDURE GRAFIKAUS;
```

```
begin
  inline
    ($f3/$f5/$c5/$d5/$e5/$dd/$e5/
     $fd/$e5/
     $3e/$00/      (* LDA A,00H      ; lade Register A mit 00H      *)
     $d3/$03/      (* OUT A,03H      ; Ausgabe an Adresse 03H      *)
     $fd/$e1/
     $dd/$e1/$e1/$d1/$c1/$f1/$fb);
end;
```

BiCom-Grafik und Textverarbeitung

Nachdem im vorhergehenden Abschnitt die Grundlagen für die Programmierung der BiCom-Grafik in TURBO PASCAL beschrieben worden sind, soll hier eine einfache, aber sehr nützliche Anwendung vorgestellt werden. Bei der Textverarbeitung wünscht man es sich häufig, einen besseren Überblick über die Textgestaltung zu haben. Jedoch fehlen auf dem Bildschirm die Anhaltspunkte. Das Auge muß ständig Schwerarbeit leisten und das Gehirn hat es schwer, die auf dem Bildschirm dargebotenen Informationen zu strukturieren. Doch dem kann leicht mit einem kleinen Turbo Pascal-Programm abgeholfen werden.

Das Programm SCREEN unterlegt dem Textbildschirm eine Grafik mit feinen Hilfslinien, ähnlich einem linierten Schreibbogen. Nach einer kurzen Eingewöhnungszeit möchte man diese Hilfslinien nicht mehr missen. Man sollte sich auch noch vertikale Hilfslinien als Randbegrenzer programmieren. Da die Lage solcher Randbegrenzungen jedoch vom voreingestellten Wert des Textverarbeitungsprogrammes abhängig ist, wurde in dem Beispielprogramm darauf verzichtet.

Das Programm verwendet die Prozeduren CLRGRA, SETPIX und DRAWLINE (Sie sind im Programmlisting zu ergänzen). Als zusätzliche Prozedur ist die Prozedur CURSOR hinzugekommen. Sie programmiert den Cursor als schnell blinkendes Feld mit neun Rasterzeilen:

```
3e 0a    LDA 0aH    ;Lade Akkumulator mit Registernr.
d3 50    OUT 50H    ;Ausgabe an CRT Controller
3e 40    LDA 40H    ;Lade Akkumulator mit 40H
                (Cursor: schnell blinkend, Startzeile=0)
d3 51    OUT 51H    ;Ausgabe an CRT Controller
3e 0b    LDA 0bH    ;Lade Akkumulator mit Registernummer
d3 50    OUT 50H    ;Ausgabe an CRT Controller
3e 09    LDA 09H    ;Lade Akkumulator mit 09H
                (Cursor: Endzeile=9 )
d3 51    OUT 51H    ;Ausgabe
```

```
PROGRAM SCREEN;
```

```
var I,IX,IY,ICOL,ILIN,IBACK:integer;
```

```
Procedure Cursor;
```

```
begin
```

```
    inline
```

```
    ($F3/$F5/$c5/$d5/$e5/$dd/$e5/$fd/$e5/
     $3e/$0a/$d3/$50/$3e/$40/$d3/$51/
     $3e/$0b/$d3/$50/$3e/$09/$d3/$51/
     $fd/$el/$dd/$el/$el/$dl/$cl/$fl/$fb/$c9);
```

```
end;
```

```
(* Procedure clrgra(var iback:Integer);          *)
```

```
(* Procedure setpix(var ix,iy,icol:Integer);      *)
```

```
(* Procedure drawline(var ix,iy,ilin:Integer);    *)
```

```
begin
```

```
    ilin :=1;          (* Linienart: fein punktiert          *)
    iback:=0;          (* Hintergrund schwarz          *)
    clrgra(iback);
    cursor;            (* Cursor als blinkendes Vollfeld *)
    iy  :=0;
    icol :=3;          (* Linienfarbe violett          *)
    for i:=0 to 23 do  (* Schleife für 24 Linien      *)
```

```

begin
    iy:=I*11+9;          (* Y Koordinaten      *)
    ix:=0;                (* X Koordinate  *)
    setpix(ix,iy,icol);   (* Anfangspunkt  *)
    ix:=319;
    drawline(ix,iy,ilin); (* Endpunkt      *)
end;
end.

```

Hilbert-Kurven - Rekursion

Zum Schluß noch ein Turbo Pascal-Programm für die Freunde rekursiver (verschachtelter) Prozeduren. Das Programm Hilbert-Kurven zeichnet auf dem Bildschirm ein Muster von Linien, die sich in immer feineren Strukturen, ähnlich einem Kristall, umlagern. Der rekursive Algorithmus, der diesen Kurven zugrunde liegt, ist nach dem deutschen Mathematiker und Logiker David Hilbert (1862- 1943) benannt. Es werden mit verblüffender Geschwindigkeit insgesamt sieben Überlagerungen ($n:=7$) in verschiedenen Farben gezeichnet. Eine achte Überlagerung wäre schon von so feiner Struktur, daß die Grafikauflösung für die Darstellung nicht mehr ausreichen würde!

PROGRAM HILBERTKURVEN;

```

(*$A-*) (* COMPILER-OPTION fuer REKURSION *)
(* Ueberlagerung der Hilbertkurven H(1)-H(n) nach N.Wirth *)
Const
    h0:=256;
Var
    n,i,h,x,y,x0,y0,lin,Farbe: integer;
    farbig : boolean;
    c : char;

Procedure Setpix(Var X,Y,ICOL : Integer);
Begin
    Inline ($f3/$2A/X/$ED/$5B/Y/$ED/$4B/ICOL/$CD/$24/$EF/$fb);
End;

Procedure DrawLine(Var X,Y,ILIN : Integer);
Begin
    Inline ($F3/$2A/X/$ED/$5B/Y/$ED/$4B/ILIN/$CD/$2C/$EF/$FB);
End;

```

```

Procedure Clrgra(Var IBACK : Integer );
Begin
  Inline ($f3/$2A/IBACK/$CD/$20/$EF/$fb);
End;

```

```

Procedure Grafikaus;
Begin
  Inline ($f3/$3E/$00/$D3/$03/$fb);
End;

```

```

Procedure ProcB( i:integer);forward;
Procedure ProcC( i:integer);forward;
Procedure ProcD( i:integer);forward;

```

```

Procedure ProcA(i:integer);
Begin
  If i>0 Then
    Begin
      ProcD(i-1); x:=x-h; DrawLine(x,y,lin);
      ProcA(i-1); y:=y-h; DrawLine(x,y,lin);
      ProcA(i-1); x:=x+h; DrawLine(x,y,lin);
      ProcB(i-1)
    End
  End;

```

```

Procedure ProcB;
Begin
  If i>0 Then
    Begin
      ProcC(i-1); y:=y+h; DrawLine(x,y,lin);
      ProcB(i-1); x:=x+h; DrawLine(x,y,lin);
      ProcB(i-1); y:=y-h; DrawLine(x,y,lin);
      ProcA(i-1)
    End
  End;

```

```

Procedure ProcC;
Begin
  If i>0 then
    Begin
      ProcB(i-1); x:=x+h; DrawLine(x,y,lin);
      ProcC(i-1); y:=y+h; DrawLine(x,y,lin);
      ProcC(i-1); x:=x-h; DrawLine(x,y,lin);
      ProcD(i-1)
    End
  End;

```

```

Procedure ProcD;
Begin
  If i>0 Then
    Begin
      ProcA(i-1); y:=y-h; DrawLine(x,y,lin);
      ProcD(i-1); x:=x-h; DrawLine(x,y,lin);
      ProcD(i-1); y:=y+h; DrawLine(x,y,lin);
      ProcC(i-1);
    End;
  End;

Begin
  farbig:=false;
  lin:=0;
  writeln(chr(12), 'Ueberlagerung der Hilbertkurven H(1)-H(n)');
  n:=7;
  writeln;writeln;
  write('Mehrfarbig ? <J/N> ');
  repeat read(kbd,c);c:=upcase(c) until c in (.'J','N'.);
  If c = 'N'
  Then
    Begin
      writeln;writeln;
      write('Invers (d.h. dunkel auf hell) ? <J/N> ');
      Repeat read(kbd,c) ;c:=UPCASE(c) Until c in (.'J','N'.);
      If c = 'N'
      Then Begin Farbe:=0;Clrgra(Farbe);Farbe:=6 End
           else Begin Farbe:=6;Clrgra(Farbe);Farbe:=0 End;
      End
    else
      Begin
        Farbe:=0;Clrgra(Farbe) ;farbig:=true
      End;

      ClrScr;
      i:=0;
      h:=h0;
      x0:=h div 2;
      y0:=h div 2;

      Repeat
        i:=i+1;
        gotoxy(72,1);write('H(',i,')');
        If farbig
        Then
          If i<=4
          Then Farbe:=5 else Farbe:=9-i;

```

```

h:=h div 2;
x0:=x0+(h div 2);
y0:=y0+h div 2;
x:=x0;y:=y0;
Setpix(x,y,Farbe);
ProcA(i);
Until i=n;
gotoxy(70,24);
write('<RETURN>');
readln;
Grafikaus;
ClrScr
End. (*$A+*)

```

Ausblick - Möglichkeiten - benutzte Quellen

Es ist wohl deutlich geworden, daß die BiCom-Grafik eine Vielzahl von Möglichkeiten bietet, Informationen besser und übersichtlicher auf dem Bildschirm darzustellen.

Einige weitere Anwendungsmöglichkeiten sind noch nicht angesprochen worden. Die BiCom-Grafik läßt sich nämlich auch als zusätzlicher Speicher verwenden! In den Speicherbereichen der Grafik-Karte, die während der Dunkelpausen auf dem Bildschirm nicht angezeigt werden, liegen ca. 16 kByte Speicherplatz brach. Da dieser Speicher nicht durch NEW oder CLEAR gelöscht wird, eignet er sich besonders für die Zwischenspeicherung von Daten, die von mehreren verschiedenen Programmen verwendet werden sollen. Die Nutzung dieses Speichers ist jedoch nicht ganz einfach. Interessenten sollten deshalb im Kapitel über die Funktion der BiCom-Grafikplatine nachlesen. Dabei muß besonders beachtet werden, daß (bei Übergabe von 8-Bit-Werten) nur jeweils 6 Bit gespeichert werden. Man kann also nicht einfach Datenbytes im Grafik-Speicher ablegen, sondern muß vorher eine Umcodierung vornehmen. Ohne Umcodierung kann man selbstverständlich die Daten für freidefinierte Zeichen in den unbenutzten Bereichen des Bildschirmspeichers ablegen.

Die Anregung für den schnellen CIRCLE-Algorithmus stammt von Gerd Benischek, Hamburg. Er hat auch das Programm 'Hilbert-Kurven' (nach Wirth) an den alphasonic PC angepaßt. Ich habe eine modifizierte Version dieses Programmes übernommen. Die Variablennamen und die Syntax der Prozeduren habe ich, um eine einheitliche Syntax unter DISK BASIC und PASCAL zu erhalten, entsprechend abgeändert. Co-Autorin dieses Kapitels war Claudia Sanner, Osnabrück.

Datenübertragung mit dem Alphontronic PC

Michael Titgemeyer

Voraussetzungen für die Datenübertragung

Für die Datenübertragung mit Ihrem PC benötigen Sie neben einem Telefon, einem geeigneten Kommunikationsprogramm noch einen Akustikkoppler. Diese werden seit einiger Zeit sehr preiswert auf dem Markt angeboten, nachdem sie noch vor ein bis zwei Jahren um tausend Mark kosteten.

Ich benutze den Typ dataphon s2ld, der sowohl mit Batterie, externem Netzteil und über den Schnittstellenstecker mit Spannung versorgt werden kann. Das dataphon s2ld erfüllt alle Voraussetzungen für die Datenübertragung (incl. FIZ-Nummer!) und ist das z.Zt. preiswerteste Gerät am Markt. Vor dem Kauf eines Akustikkopplers sollten Sie sich allerdings vergewissern, daß der Telefonhörer Ihres Telefons auch in die Gummimuscheln des Akustikkopplers paßt. Einige Telefonmodelle der Post und anderer Hersteller sind nämlich nicht geeignet, weil der Hörer nicht in den Gummimuscheln hält. In diesem Fall wenden Sie sich am besten an das für Sie zuständige Fernmeldeamt, um ggf. den Telefonapparat austauschen zu lassen. Beachten Sie auch die Gebrauchsanweisung, die Ihrem Akustikkoppler beiliegt.

Was beim Anschluß an den alphontronic PC zu beachten ist

Der Akustikkoppler wird mit der V.24 Schnittstelle des PC's über ein geeignetes Kabel verbunden. Wichtig bei dem Kabel ist, daß die Leitungen 2,3 (Sende- und Empfangsdaten); 4,5 (RTS und CTS) und 6,20 (DSR und DTR) nicht wie für einen Drucker mit V.24 Schnittstelle üblich, gekreuzt sind, sondern eins zu eins miteinander verbunden sind. Das Druckerkabel ist daher für diesen Zweck nicht geeignet!

Beachten Sie auch, daß im PC die Schalter für die Baudratenerzeugung auf 300 Baud (= bits/Sekunde) gesteckt sind. Falls Sie diese Umschaltung noch nicht vorgenommen haben, wird die Datenübertragung nicht funktionieren. Denken Sie ebenfalls daran, daß durch das Umschalten der Baudrate der Drucker, falls er an der V.24 Schnittstelle angeschlossen ist, ebenfalls auf 1200 Baud umgeschaltet werden muß. (Siehe auch Bedienungsanleitung bzw. Systemhandbuch).

Auch Software wird benötigt

Ein geeignetes Programm zur Kommunikation finden Sie am Ende des Buches abgedruckt. Sie können es entweder abtippen oder bereits fertig auf einer Diskette beziehen. Eine Anregung: Wenn Sie inzwischen der Assemblersprache des PC's mächtig sind, können Sie ja versuchen, das Programm soweit zu verändern, daß Sie damit auch die empfangenen Daten auf Diskette speichern können, um diese dann bei Bedarf wieder parat zu haben. Auch der umgekehrte Weg ist durchaus sinnvoll, nämlich Daten von der Diskette zu senden. Diese Funktion ist dann nützlich, wenn Sie bereits mit einem Texteditor vorbereitete Texte haben und diese an eine Mailbox (= (elektronischer) Briefkasten) weiterschicken wollen. Auch ist es denkbar, über diesen Weg Programme oder Daten mit einem anderen PC-Benutzer auszutauschen. Dazu kommen wir aber noch.

Frisch an's Werk

Was kann man nun mit einem Akustikkoppler anfangen? Am besten, Sie probieren es gleich mal aus. Sie wählen die Nummer einer öffentlichen Mailbox, die sich in der Nähe Ihres Wohnortes befindet (siehe nachfolgende Liste). Nachdem Sie (hoffentlich) das Freizeichen im Telefonhörer vernehmen, folgt daran anschließend ein hoher Ton, auf dessen Bedeutung wir später noch kommen. Legen Sie nun den Telefonhörer in die Gummimuscheln des

Akustikkopplers (möglichst richtig herum, sonst passiert nämlich gar nichts). An allen Akustikkopplern befinden sich deshalb Hinweise, wie der Telefonhörer in die Gummimuscheln eingelegt werden muß.

Auf Ihrem PC sollten Sie natürlich **vorher** das Kommunikationsprogramm gestartet haben, damit ab jetzt die empfangenen Daten auf dem Bildschirm angezeigt werden und die Tastatureingaben auf der anderen Seite "verstanden" werden können. Der Computer auf der anderen Seite des Telefons wird nun beginnen, sich und sein System vorzustellen. Die meisten Systeme zeigen dann noch an, der wievielte Anrufer Sie gerade sind. (Bei einigen Mailboxen sind das inzwischen bestimmt schon an die 10.000 Anrufer).

Und nun kann's losgehen. Es ist überhaupt nicht schwer, sich in solchen Mailboxen zurechtzufinden. In der Hauptsache tippt man nur einige Ziffern für ein Menü ein und schon hat man die entsprechende Information. Beachten sollten Sie allerdings auch, das Ihr Gebührenzähler im Telefon kräftig rotiert, wenn Sie tagsüber außerhalb Ihres Ortsnetzes anrufen. Ruft man abends oder am Wochenende (Billigtarif) an, ist meistens der Anschluß besetzt.

Liste einiger Mailboxen in Deutschland

(Stand: Juni 1985)

0201/274625	E.M.S. (20-13 Uhr)
0202/559350	Toelleturm
0203/782497	M.M.S. (20-8 Uhr)
0209/271666	Vollrath Mailbox
0211/414579	Software Express
0211/593453	Epson
02151/801339	K.I.S (C=64 Krefeld)
02161/200928	Symic
02162/58457	Johnny Walker (20-6 Uhr)
02202/50033	Computer Center
0221/371076	WDR-Computer-Club
0221/394976	P-M-S
02234/58603	F.I.S. (20-6 Uhr)
0231/170414	Dortmunder Mailbox
0231/179414	Dortmunder Mailbox II
02331/16401	Kobra Box
02364/13826	H.I.B.
02373/66877	Ueding Electronic
02383/50866	I.G.S.

0241/870555	A.I.S. (18-6 Uhr)
02841/57325	M.H.B. (20-6 Uhr)
030/3052635	Berliner Mailbox(18-9Uhr)
030/4652439	COCO-BOX (20-8 Uhr)
030/6818679	IBB (22-10 Uhr)
040/2512371	MCS
040/4916117	H.I.S.
040/5277016	Tornado-Box
040/5593129	VMS
040/6323517	C.L.I.N.C.H.
040/6788783	H.O.M.
040/6936657	M.A.G.
040/7540598	C=64 Harburg
040/8802383	RAMses' Box
04101/200543	PEKA (22-6 Uhr)
04101/23789	Wang-Info (20-6 Uhr)
04131/82593	M.M.M.
0421/402844	B.M.S.
0421/428667	BAM 1000 (18-7 Uhr)
04348/7513	N.C.S.
0461/93727	Wiking Mail Flensburg
04683/554	COMAL
05121/45792	Aquila (20-8 Uhr)
06081/9677	Taunus Mailbox
06128/5117	Infosystem (21.30-9 Uhr)
06154/51433	Decates
06181/48884	Otis
06187/25828	Thor-Mb.
06434/6291	CCCCamberg
069/6638191	COMBO
069/816787	Tecos(20-7 Uhr)
07031/26166	Elias
0711/519008	Norsak
0721/556468	MBS-Karlsruhe
0721/685010	M.C.S. Karlsruhe
08362/7647	B.O.S. (20-6 Uhr)
089/164959	Info-Control
089/392289	HIGH TECH jr.
089/596422-3	Tedas 1 und 2
089/7931332	Phoenix
0911/574160	Smurf-Box

Ohne Gewähr und Anspruch auf Vollständigkeit. (es können inzwischen einige neue Anbieter dazugekommen sein bzw. einige könnten ihren Betrieb eingestellt haben).

Wie Sie feststellen werden, gibt es in einigen Gebieten regelrechte Häufungen solcher Mailboxen, dagegen scheinen manche Regionen wahre "Entwicklungsländer" auf diesem Gebiet zu sein. Aber das kann sich ja noch in der nächsten Zeit ändern.

Allgemeines zum Thema Mailbox

Die Struktur einer solchen Mailbox ist durchweg ähnlich: An den "Schwarzen Brettern" können Angebote, Gesuche und Kontaktwünsche "ausgehängt" werden. Benutzer können untereinander Nachrichten austauschen, die nur sie selbst betreffen. Bei einigen Systemen ist das ohne weiters möglich, bei anderen muß man erst eine "Erlaubnis" dafür haben.

Auch können Programme abgefragt werden. Viele Betreiber lassen dieses entweder gegen Bezahlung oder nach vorheriger "Zusendung" eines eigenen Programmes zu. Der Betreiber einer solchen Mailbox benutzt sie meistens auch für eigene Werbezwecke oder bietet "Werbefläche" für andere Anbieter an.

Wenn Sie, ohne selbst zu probieren, sehen möchten, was bei einem Anruf passiert, hier ein kurzes Beispiel:

D=Deutsch; E=English >D

(das System kann beides)

```
#####  
## TORNADO BULLETIN BOARD SYSTEM ##  
## Programm vl.7 (C) Schewe 1985 ##  
## Hamburg (040) 527 70 16 ##  
## Täglich 24 Stunden geöffnet ##  
#####
```

Guten Abend !

Willkommen bei TORNADO BBS

Heute ist Mittwoch, 17.07.1985

Es ist 22:42:48 Uhr

> 11 Minuten Zeitlimit im System (je nach Tageszeit)
> 60 Sekunden Timeout bei Eingaben (damit keiner einschläft)

Ihr Name (max.25) >MITI
 Sind Sie eingetr. User? J/N >
 Ihr Passwort (max.9) >----- (braucht nicht jeder zu sehen)
 Hallo MITI aus NUERNBERG (wird beim ersten Mal abgefragt)
 dies ist Ihr 8. Anruf
 verbr.Zeit bisher: 1:35:16 (Schwarz-Schilling freut sich)
 Zuletzt im System: 24.04.1985 14:51:22

H A U P T M E N U E

- 1 Informationen ueber TORNADO-BOX
- 2 Die HOMECOMPUTER-Ecken
- 3 Oeffentliche Box (schwarzes Brett)
- 4 Private Box (elektron. Briefkasten)
- 5 Programm - Box
- 6 DFUe - Ecke
- 7 Benutzerliste / Statistiken
- 8 Parameter aendern
- 9 Sysop rufen / Mail an Sysop
- ? Gibt diese Liste aus
- Restzeit = 10:30 (Galgenfrist)

(1-9, *, ?, 0=LOGOFF)

Ihre Eingabe, MITI >*6 (sehr persönlich)

DFUe - Ecke (dies ist ein Untermenü)

- 1 Mailboxen in Deutschland
- 2 Mailboxen im Ausland
- 3 DATEX-P
- 4 Die TELEBOX der Post
- 5 Besuche bei auslaendischen Mailboxen
- * Zurueck zum Hauptmenue
- ? Gibt diese Liste aus
- Restzeit = 10:18

.....

.... und so weiter, aber probieren Sie doch selbst, es ist nicht schwierig. Andere Mailboxen haben andere Menüs und sind ebenso einfach (oder auch nicht) zu bedienen. Einige Anbieter verlangen für ihre "Dienste" allerdings auch bare Münze, nämlich dann, wenn Sie bestimmte Informationen oder Programme ansehen wollen. Hier sollten Sie dann selbst abwägen, ob das für Sie persönlich in Frage kommt.

Grundsätzliches zum Akustikkoppler

Beschäftigen wir uns nun mit dem Akustikkoppler etwas näher. Er hat die Aufgabe, die "Nullen" und "Einsen", die der PC über seine V.24 Schnittstelle aussendet, in ein analoges Signal umzusetzen, das dann über die ja nur für Sprache ausgelegte Telefonleitung zu der Gegenstelle geschickt werden kann. Der Name Akustikkoppler sagt auch schon etwas über die Art aus, in der die analogen Signale in das Telefonnetz eingespeist werden, nämlich akustisch über den Telefonhörer.

Im Gegensatz dazu gibt es noch die galvanische oder auch direkt gekoppelten Modems, die aber nur von der Post direkt vermietet werden. Da ein solches Modem von der Post monatlich ca. 50 DM kostet, hat man mit der Investition für einen Akustikkoppler (ca. 300 DM) diese schon nach knapp einem halben Jahr wieder 'raus. Ein direkt gekoppeltes Modem bietet allerdings den Vorteil, daß bei der Übertragung weniger Fehler durch Nebengeräusche entstehen können, als bei einem Akustikkoppler. Ein Postmodem kann ankommende Rufe beantworten, es arbeitet dann im Answermode. Eines ist an dieser Stelle allerdings wichtig zu wissen: Der PC ist von der Post für den Betrieb eines solchen Modems nicht zugelassen!

Für technisch Interessierte ...

"Wie funktioniert denn die Übertragung genau?", werden Sie sich fragen. Nun, das ist im Prinzip ganz einfach. Man ordnet den zwei logischen Pegeln (0 und 1) der V.24 Schnittstelle einfach zwei Frequenzen zu, 980 Hz für log. 1 und 1180 Hz für log. 0. Die Elektronik des Akustikkopplers erzeugt entsprechend der Eingangssignale an der Schnittstelle den entsprechenden Ton und steuert damit einen kleinen Lautsprecher an, der der Sprechkapsel des Telefonhörers gegenüberliegt. Die umgekehrte Richtung, sprich Empfang, erfolgt über ein kleines Mikrofon, das sich gegenüber der Hörkapsel des Telefonhörers befindet.

Der Empfänger des Akustikkopplers reagiert aber nicht auf die selben Frequenzen wie die des Senders. Es würde zu einem furchtbaren Durcheinander führen, da das Telefon die Eigenschaft hat, das gesprochene Wort ebenfalls wieder hörbar zu machen. Die "Tonlage" des Empfängers ist deswegen etwas höher, nämlich 1650 Hz für log. 1 und 1850 Hz für log. 0. Nachdem das Mikrofonsignal verstärkt wurde, durchläuft es Filter, die nur diese zwei Frequenzen durchlassen und, so gut es geht, alle Störgeräusche ausfiltern. Das Ergebnis sind wieder zwei logi-

sche Zustände, die die Empfangsdaten der V.24 Schnittstelle darstellen. Die empfangenen Daten werden dann im USART des PC von der seriellen Darstellung in die 8 bit parallele Darstellung zurückgewandelt und durch das Kommunikationsprogramm auf dem Bildschirm angezeigt.

Answer- und Originatemode

Viele Akustikkoppler haben einen Umschalter für **Answer-** und **Originate Mode** (answer = Antwort; originate = Ursprung). Damit wird das jeweilige Frequenzpaar für Senden und Empfangen festgelegt. Das dataphon s2ld kennt zusätzlich noch einen Auto-Mode, bei dem abwechselnd zwischen Answer- und Originatemode umgeschaltet wird. Wird ein Frequenzpaar richtig erkannt, "rastet" der Akustikkoppler ein und die Übertragung kann beginnen.

Datenübertragung zwischen zwei PC's

Wollen Sie Daten mit einem anderen PC austauschen, muß der eine Akustikkoppler im Answer-, der andere im Originatemode arbeiten. Achten Sie deshalb beim Kauf Ihres Akustikkopplers auf diese Umschaltmöglichkeit. Ohne sie kann es Probleme beim gegenseitigen Datenaustausch geben. Vor der eigentlichen Datenübertragung sollten sich die zwei Partner darauf einigen, wer seinen Akustikkoppler wie einstellt. Am Besten ist es, wenn einer der beiden Partner die Ablaufsteuerung der Übertragung übernimmt und so Mißverständnisse und unnötige Kosten vermieden werden.

Voll- und Halbduplex

Einige Akustikkoppler haben einen Schalter für Halb- und Vollduplexumschaltung. Unter Vollduplexübertragung versteht man die Möglichkeit, Daten **gleichzeitig** in beiden Richtungen zu übertragen (vergleichbar mit dem Telefon). Halbduplexbetrieb läßt die Übertragung der Daten in eine Richtung zu einer Zeit zu (vergleichbar mit CB-Funk). Grundsätzlich ist die Vollduplexübertragung mit Ihrem Akustikkoppler möglich, daher ist die Bezeichnung des Schalters verwirrend. Er hat lediglich die Funktion, die von der Schnittstelle empfangenen Daten sofort wieder zurückzuschicken. Dieses bezeichnet man auch als "local echo" im Gegensatz zum "remote echo", bei dem der angeschlossene Rechner diese Aufgabe übernimmt. Da es nur ganz selten vorkommt, daß ein Rechner kein "remote echo" erzeugt, ist ein sol-

cher Umschalter nicht unbedingt nötig. Außerdem läßt sich das Problem auch per Software lösen, eine Aufgabe für den versierten Assemblerprogrammierer.

Der Antwortton – gewußt warum

Alle Mailboxsysteme (und sonstigen Computer), die man anwählen kann, arbeiten im sog. Antwortmode, das heißt, sie antworten auf einen Anruf. Der Akustikkoppler des Anrufers steht dagegen im Originatemode. Das angerufene Modem sendet vor Beginn der eigentlichen Datenübertragung einen sog. Antwortton (2250 Hz) aus, der alle auf der Übertragungsstrecke befindlichen Echosperrern ausschaltet. Diese Echosperrern verhindern die bei längeren Signallaufzeiten auftretenden Echos. Sind in einer Übertragungsstrecke, meist in Weitverkehrs-Vermittlungsstellen, solche Echosperrern eingebaut, so würden sie die Übertragung zu einer Zeit nur in eine Richtung zulassen (Simplex-Betrieb). Dadurch wird der bei allen 300 Baud Modems mögliche Vollduplexbetrieb verhindert. Im Nahbereich werden aber keine Echosperrern eingesetzt.

Für den professionellen Anwender – die Datenbankabfrage

Neben der Möglichkeit, die DÜ-Fähigkeit Ihres PC's für Zwecke des Hobbies zu verwenden, (Mailbox) gibt es auch eine durchaus sinnvolle Anwendung: Viel Wissen ist heutzutage in Datenbanken gespeichert und kann von dort abgerufen werden (gegen Bezahlung, versteht sich). Es gibt für nahezu alle Gebiete, Wissenschaft, Medizin, Wirtschaft usw., spezielle Datenbanken, mit deren Hilfe Sie sich viel Zeit ersparen können, wenn Sie nach etwas suchen. Mit ihrem PC, einem Telefon und einem Akustikkoppler haben sie Zugriff auf eine enorme Menge von Daten und Informationen, nicht nur in Deutschland, sondern weltweit.

Eine Möglichkeit – DATEX-P

Datenbanken bzw. deren Rechner sind über festgeschaltete Leitungen miteinander verbunden. Auf diesen Verbindungswegen sind allerdings Übertragungsgeschwindigkeiten von einigen tausend Baud angesagt, bei denen unser verhältnismäßig langsame PC seinen Dienst versagt. Eine Abhilfe hat die Deutsche Bundespost geschaffen, als sie, neben anderen DATEX-Diensten, den DATEX-P20F Dienst einrichtete. Dadurch ist es möglich, über das öffentliche Telefonnetz und einen Umsetzer (PAD=Packet

Assembly/Disassembly Facility), Rechner mit höheren Übertragungsgeschwindigkeiten, zu erreichen.

Das "P" von DATEX-P, übrigens eine Zusammensetzung von DATA EXchange, sagt aus, daß auf den Verbindungswegen sog. Datenpakete verschickt werden. Diese Pakete haben, genau wie bei der "gelben Post", Ziel- und Absenderadressen. Über die jeweilige DATEX-P Rufnummer wird der Zielrechner adressiert und kann so mit dem PAD bzw. dem angeschlossenen System kommunizieren.

Durch die Verwendung von Datenpaketen ist es möglich, viele Datenpakete, zeitlich versetzt, zu verschicken. Dadurch können die Übertragungswege optimal ausgelastet werden. Der PAD übernimmt seinerseits die Aufgabe, ankommende Datenpakete "auszupacken" (disassembly) bzw. über das Telefonnetz empfangene Daten zu "verpacken" (assembly) und sie, wenn das Paket voll ist (oder eine bestimmte Zeit abgelaufen ist), an die Zieladresse zu schicken.

Ich meine, das reicht für's erste. Wenn Sie mehr über DATEX-P wissen wollen, wenden Sie sich entweder an das nächste Fernmeldeamt oder an einen Telefonladen. Wo sich ein PAD befindet, zeigt die nachfolgende Übersicht. Nachdem Sie den Antwortton hören, legen Sie den Telefonhörer in den Akustikkoppler. Danach "." und <RETURN> eingeben (ggf. wiederholen). Danach meldet sich der PAD mit DATEX-P: 44 XXXX XXXXX. Anschließend "identifizieren" Sie sich durch Ihre Teilnehmerkennung und stellen die Verbindung zum Zielrechner durch eingeben der DATEX-P Rufnummer her.

Rufnummern der DATEX-P PADs in Deutschland

DATEX-P PAD	Vorw	300Bd	1200Bd	1200/75
Augsburg	0821	36791	36761	36761
Berlin	030	240001	240081	240061
Bielefeld	0521	59011	59021	59041
Bremen	0421	170131	14291	15077
Dortmund	0231	57011	52011	52081
Düsseldorf	0211	329318	329249	320748
Essen	0201	787051	791021	793003
Frankfurt	069	20281	20291	20201
Hamburg	040	441231	441261	441281
Hannover	0511	326651	327481	327591
Karlsruhe	0721	60241	60381	60581
Köln	0221	2911	2931	2951
Mannheim	0621	409085	39941	39951

München	089	228730	228630	228758
Nürnberg	0911	20571	20541	20501
Saarbrücken	0681	810011	810031	810061
Stuttgart	0711	299171	299061	299261

Nichts ist umsonst ...

Wie alle Dinge kostet auch dieser Dienst der Post Geld. Nach erfolgter Antragstellung für die "Zuteilung einer Teilnehmererkennung DATEX-P" kommen auf Sie monatliche Gebühren, ähnlich wie beim Telefon, zu. Da ist einmal die Grundgebühr von 15 DM plus die anfallenden "Verbindungsgebühren". Ohne hier weiter ins Detail zu gehen muß gesagt werden, daß Rechnerverbindungen über DATEX-P im Vergleich zu Verbindungen über's Telefonnetz in den meisten Fällen günstiger sind. Das richtet sich allerdings auch danach, wo Sie wohnen und wo der Zielrechner steht.

Mit der Teilnehmererkennung können Sie dann über das DATEX-P Netz Verbindungen in die weite Welt herstellen. Im konkreten Fall werden Sie den DATEX-P Zugang hauptsächlich für Datenbankabfragen benutzen, da es hier keine öffentlichen Mailboxen gibt, die umsonst zugänglich sind.

Kommerzielle Nutzung der Mailboxidee

Eine Möglichkeit der Mailboxen, den persönliche Nachrichtenaustausch, haben verschiedene Rechnerbetreiber im DATEX-P Netz kommerziell nutzbar gemacht. (Die Möglichkeit, öffentliche Mitteilungen am schwarzen Brett auszuhängen, spielt eher eine untergeordnete Rolle).

Als Beispiel der TELEBOX-Dienst der Deutschen Bundespost, der sich an dem bereits in USA und anderswo bestehenden DIALCOM System orientiert, und die IMCA-Mailbox (Betreiber: **IMCA-Microcomputer GmbH, Hannetal/Stärklos**), bei der es z.B. möglich ist, eine virtuelle Mailbox für einen geschlossenen Benutzerkreis zu mieten. Hier gibt es neben den "öffentlichen" Nachrichten auch die Möglichkeit, Nachrichten untereinander auszutauschen und zwar schneller, als mit der "gelben" Post, sofern die Benutzer einen regelmäßigen Abfragemodus vereinbaren.

Die TELEBOX der Bundespost

Informationen über den TELEBOX-Dienst können Sie entweder über DATEX-P oder den Zugang per Telefon bekommen. Wenn Sie Besitzer einer DATEX-P Teilnehmerkennung sind, wählen Sie **45621040000** oder über Telefon **0621/413091**. Nach der Aufforderung, sich vorzustellen, geben Sie **ID INF100 TELEBOX** ein, alles weitere erklärt Ihnen das System selbst. Die Post hat diesen Zugang für potentielle Interessenten an ihrem neuen Dienst geschaffen.

Die IMCA-Mailbox

Die Benutzer der IMCA Mailbox haben fast die selben Möglichkeiten, wie die der Telebox. Darüber hinaus bietet IMCA die Möglichkeit, Telexe zu senden und zu empfangen. Damit haben selbst kleine Unternehmen die Möglichkeit, die ihnen sonst wegen der hohen Anschaffungskosten für einen Telexer verschlossen sind. Ebenfalls können Nicht-Inhaber einer Box dem Inhaber eine Nachricht übermitteln, sofern er ihm das entsprechende Passwort nennt. (Es ist übrigens nicht das selbe Passwort, wie das für seine eigene Box).

Persönliche Erfahrungen

Meine ersten Erfahrungen mit der Datenübertragung begannen damit, daß ich mir einen Akustikkoppler von einem Bekannten auslieh, da es zu der Zeit noch keine preiswerteren Geräte gab. Es war ein Gefühl wie an Weihnachten, als ich die Nummer von TEDAS in München anwählte. Ich war der so und sovielte Anrufer ..., naja, und dann habe ich halt so in den diversen Menüs herumgelesen, hauptsächlich in den Rubriken Suche, Biete und Kontakte, weil ich mir davon versprach, Kontakt zu anderen Computeranwendern zu bekommen.

Gleichzeitig wünschte ich mir auch meine Kenntnisse auf dem Gebiet der Computerei einer größeren Masse zur Verfügung zu stellen. Nachdem ich mit einigen Leuten aus der TEDAS Mailbox in Kontakt gekommen war, (zwei sind dann zu Besuch von München gekommen), erfuhr ich auch die Telefonnummern von einigen anderen Mailboxen in Deutschland, z.B. DECATES etc.

Jemand erzählte mir auch von Mailboxen in England, Finnland und Südafrika und von den 1600 Gebühreneinheiten, die er kürzlich zusammengebracht hatte, als er, nur so aus Spaß an der Freud', eine Nummer in Schweden angerufen hatte. Diesen Versuch habe ich allerdings tunlichst vermieden, dafür war mir mein Geld nun

doch zu schade. Es steht Ihnen allerdings frei, diese Erfahrungen selbst zu machen.

Was sich in so einer Mailbox abspielt, ist fast immer das selbe. Es gibt schwarze Bretter, an denen die Anrufer ihre Sorgen, Probleme und sonstiges loswerden können. Dann haben einige Anbieter einen elektronischen Briefkasten eingerichtet, in denen die, bei einigen Mailboxen privilegierten Benutzer, untereinander Nachrichten austauschen können. Viele Mailboxen zielen mit ihren Computerecken auf einen bestimmten Kreis von Computermarken ab, z.B. Commodore 64, Sinclair, Atari etc. Daraus resultiert dann auch, das man sehr häufig die neuesten PEEK's und POKE's für die entsprechenden Computer erfährt. Die angebotenen Programme in den Programmboxen sind deswegen auch meistens nur auf den entsprechenden Rechnern lauffähig. Sucht man dagegen Informationen über den eigenen Rechner, so muß man oft nach langem Suchen in den verschiedensten Brettern wieder erfolglos auflegen. Selbst Informationen über CP/M und sonstige Informationen für den professionellen Anwender fehlen häufig. In der SMURF-BOX in Nürnberg hat sich allerdings eine alphanet PC Ecke etabliert, die allerdings nur mit einem bestimmten Passwort zugänglich ist. Wenn Sie dort eine Nachricht hinterlassen, wird es Ihnen einer der Benutzer freundlicherweise mitteilen.

Immer diese hohen Telefonrechnungen ...

Fast ein Jahr habe ich in den verschiedensten Mailboxen in Deutschland herumgestöbert, was sich allerdings sehr stark auf meine monatlichen Telefonrechnungen auswirkte. Selbst in einer Großstadt wie Nürnberg gibt es erst seit kurzem eine private Mailbox, die zwar auf einem Commodore 64 läuft, dafür aber zum Ortstarif erreichbar ist.

DATEX-P, eine Kostenalternative ?!

Der Zeitpunkt war gekommen, sich auf die Suche nach einer preiswerteren Alternative zu den vielen teuren und auch teilweise erfolglosen Telefonanrufen bei Mailboxen, kam ich auf DATEX-P. Leider gibt es hier nicht so viele Anbieter, um nicht zu sagen keine, wie über das öffentliche Telefonnetz erreichbare.

Viele Benutzer schrecken einerseits vor den 15 Mark Grundgebühr pro Monat und den entsprechenden Verbindungsgebühren zurück, sollten andererseits aber eine Gegenüberstellung der Kosten einer

Verbindung über DATEX-P und öffentlichem Fernsprechnetzen machen. Eine Alternative ist, mehrere Interessenten für eine Teilnehmerkennung zu finden. Jede weitere Teilnehmerkennung kostet dann nur noch 5 DM Grundgebühr und man kann diese Gebühr auf alle Teilnehmer verteilen. Jeder kann auf seine Kosten Verbindungen herstellen und bekommt am Monatsende seine Rechnung zugeschickt. Schlecht sind allerdings diejenigen dran, die keinen PAD in ihrem Ortsnetz haben. Es bleibt abzuwarten, ob sich Anbieter finden, die Kosten für einen Rechner und den entsprechenden DATEX-P Hauptanschluß übernehmen und ihr System kostenlos der Allgemeinheit zur Verfügung stellen.

Dreht euch nicht um, der Passwortklau geht um!

So lustig sich das auch anhört, es hat einen ernsten Hintergrund. Einige Mitglieder eines Hamburger Computer Clubs verschafften sich Zugang zu TELEBOX über eine, meiner Ansicht nach, lustige Art. Sie riefen einen nicht näher genannten Herrn vom FIZ in Darmstadt an und gaben sich als Techniker des Rechenzentrums in Mannheim aus. Unter einem Vorwand wurde ihnen das derzeit gültige Passwort des Herrn genannt und dann ...

Nun ja, das kann jedem mal passieren, sollte aber nicht. Passwörter sind nicht für die Öffentlichkeit gedacht, ebenso wenig wie die Geheimnummer, die man zur Benutzung der Geldautomaten mit einer Scheckkarte benötigt. Für die Benutzung einer Datenbank, dem DATEX-P Netz, TELEBOX usw. muß man ja bares Geld für jede Minute Anschaltzeit bezahlen. Die Abrechnung der anfallenden Gebühren erfolgt also, ähnlich wie beim Telefon, über entsprechende "Zähler". Da ist es dann nicht besonders erfreulich, wenn man am Monatsende feststellen muß, daß da außer einem selbst noch jemand anders "telefoniert" hat.

Deshalb sollten Sie sich von Anfang an angewöhnen, Passwörter nirgendwo schriftlich zu hinterlegen und sie sich so einprägen, wie den eigenen Geburtstag. Sollte jemand auf irgendeine Art und Weise versuchen, Passwörter zu erfahren, sollten Sie sehr misstrauisch werden, denn an der Sache ist etwas oberfaul.

Von BASIC nach TURBO-PASCAL

Peter Hagemann

Nachdem man "alle" Programmier-Probleme in BASIC gelöst hat, wächst bei vielen das Interesse an einer anderen Programmiersprache. Sei es aus Wissensdurst oder weil es da doch ein "paar" Probleme gab, die sich in BASIC nicht oder nur schlecht lösen ließen - ein neues System muß her. Und damit es nicht "koste, was es wolle", bietet sich dazu TURBO-PASCAL an, ein Compiler der durch die "Lobgesänge" der Testberichte auf sich aufmerksam gemacht hat.

TURBO-PASCAL wird z.Zt. für MS-DOS- und für CP/M-Computer mit Z80-Prozessor angeboten. Eine Diskette im PC-Format ist mit einem deutschsprachigen Handbuch lieferbar. Obwohl das Handbuch alle Möglichkeiten und Besonderheiten von TURBO-PASCAL beschreibt, ist es als alleinige Unterlage für "Autodidakten" nicht ausreichend geeignet. Dieser Artikel soll daher bei den Anfängen behilflich sein.

1. Systemanpassung

Kopieren Sie alle Files auf eine CP/M-Diskette! Sie können jetzt einmal TURBO-PASCAL durch Eingabe von TURBO<CR> starten. Nach einer kurzen Ladezeit meldet sich TURBO und gibt den Hinweis: "No Terminal selected". Wenn Sie jetzt weiter arbeiten, erscheinen nur unsinnige Zeichen auf dem Bildschirm. -- Warum? Da TURBO-PASCAL für alle Z80-CP/M-Rechner geeignet ist, muß es vor der ersten Inbetriebnahme zunächst einmalig auf dem PC "installiert" werden. Darunter versteht man das Anpassen der Tastaturcodes und der Bildschirmkommandos.

Was sich hier so kompliziert liest, geht in Wirklichkeit sehr einfach. Auf der Diskette befindet sich nämlich ein besonderes "Installations-Programm", mit dem man genau diese Arbeiten im Dialog mit dem PC erledigen kann. Da verschiedene Werte für die richtige Installation im PC-Handbuch nicht genannt sind, wird an dieser Stelle eine Minimal-Installation beschrieben:

Starten Sie das Programm durch Eingabe von TINST<CR>. Sie werden gefragt ob der <S>creen oder die <C>ommands installiert werden sollen.

1.1. COMMAND-INSTALLATION

Mit diesem Programmteil kann die Tastatur angepaßt werden. Zusammen mit den Blockbefehlen bietet der Editor von TURBO PASCAL insgesamt 45(!) Funktionen an. Leider liefern die Funktions- und Pfeiltasten der PC-Tastatur nur ASCII-Werte oberhalb von 127. Diese Werte werden vom Editor nicht verstanden. Sie sollten sich daher zunächst auf die im Handbuch vorgeschlagenen, WORDSTAR-ähnlichen Control-Sequenzen festlegen.

1.2. SCREEN-INSTALLATION

Es erscheint ein Auswahlmenü mit Rechnern, deren Daten bereits vorhanden sind. Der PC ist leider nicht dabei, und Sie müssen "None of the above" anwählen. Jetzt läßt sich der PC durch Druck auf die angezeigten Tasten installieren:

```

Send an initialization string to the terminal?      <N>
Send a reset string to the terminal?               <N>
CURSOR LEAD-IN command                            <ESC><Y><CR>
CURSOR POSITIONING CMD to send between line & column: <CR>
CURSOR POSITIONING CMD to send after both line & column: <CR>
Column first                                       <N>
OFFSET to add to line:                            <3><2><CR>
OFFSET to add to column:                          <3><2><CR>
Binary adress                                     <Y>
CLEAR SCREEN command                              <CTRL-L><CR>
Does CLEAR SCREEN also HOME cursor                <Y>
DELETE LINE command                               <CR>
ERASE TO END OF LINE command                      <ESC><K><CR>
START HIGHLIGHTING command                       <CTRL-R><CR>
END HIGHLIGHTING command                         <CTRL-Ö><CR>
Number of rows (lines) on your screen:            <2><4><CR>
Number of columns on your screen:                 <8><0><CR>
Delay after CURSOR ADDRESS (0-255 ms):             <CR>
Delay after CLEAR, DELETE and INSERT (0-255 ms)   <CR>
Delay after ERASE TO END OF LINE and HIGHLIGHT (0-255ms) <CR>
Is this definition korrekt?                        <Y>
Operating frequency of your microprocessor in MHz <4><CR>

```

Nach der Installation werden auch die PC-8 Daten aufgenommen und können bei einer eventuell notwendigen Neuinstallation aus dem Menü abgerufen werden. TURBO-PASCAL ist nun einsatzbereit.

1.3. SCHWIERIGKEITEN mit dem PC und dem Editor von TURBO-PASCAL (Die hier gemachten Angaben beziehen sich auf die Version TURBO-PASCAL 2.0. Möglicherweise wird sich die in Kürze er-

scheinende Version 3.0 anders verhalten.)

Es können nicht alle PC Escape-Sequenzen installiert werden. Z.B. arbeiten die Funktionen DELETE-Line oder INSERT-Line nicht ordnungsgemäß. TURBO-PASCAL bildet diese Möglichkeiten allerdings softwaremäßig nach, so daß der Anwender diesen Mangel kaum bemerkt. Auch beim Scrollen ergeben sich in der untersten Bildschirmzeile gelegentlich Schwierigkeiten. Wichtig ist aber, daß trotz der kleinen Fehler ein ordentliches Arbeiten mit TURBO-PASCAL möglich ist.

Anmerkung des Herausgebers: Der Verfasser arbeitete mit einer alten, 1984 vom Generalimporteur noch nicht an den TA-PC angepaßten Version. Die Software-Distributoren COSTEC in Kassel und TRC in Nürnberg bieten jedoch speziell an den TA-PC angepaßte und makellos laufende Versionen an, teils auch mit deutschen Fehlermeldungen (siehe SOFTWARE-KATALOG). Auch die Fa. Heimsoeth versichert, daß inzwischen nur noch angepaßte Versionen ausgeliefert werden.

2. Die wichtigsten Unterschiede zwischen BASIC und TURBO-PASCAL

Wenn Sie Ihren BASIC-Interpreter starten, lassen sich sofort im sogenannten "Direkt-Modus" Befehle ausführen. Man kann z.B. eine Addition durch Eingabe von: "PRINT 20 + 10<CR>" durchführen. Auch Wertzuweisungen an Variable sind möglich: "PI=3.14159<CR>". Ebenso lassen sich laufende Programme unterbrechen und deren Variablenwerte abfragen: "PRINT A\$<CR>". Das alles ist möglich, weil die Anweisungen der Reihe nach "interpretiert"; d.h. in für das Computersystem ausführbare Anweisungen übersetzt werden. Für das Austesten von Programmen können diese Eigenarten sehr nützlich sein. Grundsätzlich aber lassen sich mit Interpretern keine zeitkritischen Probleme lösen, da der ständig erforderliche Übersetzungsvorgang relativ viel Zeit in Anspruch nimmt.

Ganz anders läuft die Programmerstellung und -verarbeitung in TURBO-PASCAL ab. Wenn Sie TURBO starten, befinden Sie sich zunächst im Editor. Mit diesem sehr leistungsfähigen Software-Werkzeug läßt sich der Programmtext erstellen. Die Bedienung ist dabei ähnlich dem bekannten WORD-STAR. Es lassen sich mit dem Editor aber genauso gut auch beliebige andere Texte verfassen. Ebenso könnten Sie den PASCAL-Programmtext auch mit einem beliebigen anderen Textprogramm schreiben; z.B. mit MICRO-TEXT. Der Texteditor ist nämlich vom eigentlichen Programm völlig unabhängig. Wenn der Programmtext fertig geschrieben ist, kann daraus vom Compiler ein ablauffähiges

Maschinenspracheprogramm angefertigt werden. Das erzeugte Programm (ein COM-File) kann dann völlig unabhängig vom Programmtext und TURBO-PASCAL gestartet werden. Eine Unterbrechung des Programms, um z.B. eine Variablenabfrage im Direktmodus durchzuführen, ist daher natürlich nicht mehr möglich. Alle so erstellten Programme laufen mit einer deutlich höheren Geschwindigkeit ab als vergleichbare BASIC-Programme. Der Nachteil dabei ist, daß jede Änderung im Programmtext einen neuen Compilerlauf erforderlich macht. TURBO-PASCAL unterstützt den Anwender dabei in hervorragender Weise: Nach dem Verlassen des Edit-Modus kann ohne Nachladen direkt der Compiler gestartet werden. Bei kleineren Programmen ist es sogar möglich direkt im RAM zu compilieren und anschließend die Programmausführung zu starten. Der Editor, der Compiler und das erstellte Programm befinden sich dann gleichzeitig im RAM. Die gesamte Datenverwaltung wird automatisch von TURBO übernommen.

3. Einfache Programme in PASCAL

Während bei BASIC ein ziemlich legerer Programmstil erlaubt ist, herrscht bei PASCAL "Zucht und Ordnung". Diese "Kleinigkeitskrämerei" schreckt manchen BASIC-Programmierer bei seinen ersten PASCAL-Kontakten ab. Aber gerade durch die erzwungene Ordnung in PASCAL-Programmen ergeben sich große Vorteile. Die Programme bleiben übersichtlich und verständlich. Wenn nun im Folgenden ein kleiner Streifzug durch PASCAL erfolgt, dann können und sollen diese Angaben nur für TURBO-PASCAL gelten. Andere PASCAL-Versionen (z.B. UCSD) arbeiten anders und erfordern u.U. andere Anweisungen. Außerdem wurde bewußt auf eine allzu "wissenschaftliche" Ausdrucksweise verzichtet, da es für den BASIC-kundigen Leser einfacher ist, sich durch kleine Programmbeispiele mit PASCAL vertraut zu machen.

3.1. Das erste "AHA-Erlebnis"

Um mal eben schnell seinen BASIC-Interpreter "auszuprobieren", genügte bereits die Eingabe von z.B.: PRINT "alphaTronic"<CR>, und schon konnte man das Ergebnis auf dem Bildschirm bewundern. Für PRINT existiert in PASCAL der Befehl WRITE. Mit WRITE können Werte, die in einer nachfolgenden Klammer stehen, ausgegeben werden. Zeichen oder Zeichenketten werden in PASCAL mit einem Apostroph abgegrenzt. Also wird die BASIC-Anweisung durch: WRITE('alphaTronic'); ersetzt. Das Semikolon zum Schluß teilt dem Compiler mit, daß die Anweisung hier beendet ist. Es dürfen auch mehrere Anweisungen in einer Zeile stehen. Es ist kein trennender Doppelpunkt wie im BASIC erforderlich, da hier ja das Semikolon ein Anweisungsende kennzeichnet. Innerhalb der "WRITE-Klammer" können auch mehrere auszugebende Werte stehen.

Die Trennung erfolgt durch Kommata. Z.B. ergeben folgende Anweisung alle das gleiche Ergebnis :

```
A:  write('alphaTronic');  
    write(' ');  
    write('PC-8');  
  
B:  write('alphaTronic');  write(' ');  write('PC-8');  
  
C:  write('alphaTronic PC-8');  
  
D:  write('alphaTronic',' ','PC-8');
```

Beachten Sie auch das WRITE, im Gegensatz zu PRINT (bei BASIC) keinen automatischen Zeilenvorschub erzeugt. Während Sie bei BASIC den Zeilenvorschub unterbinden mußten (durch ein Semikolon), wird bei PASCAL ein Zeilenvorschub nur durchgeführt wenn Sie es ausdrücklich wünschen. Der entsprechende Befehl lautet WRITELN. Ansonsten verhält sich der Befehl WRITE und WRITELN völlig gleich.

Geben Sie als Beispiel: WRITE('alphaTronic'); ein und schließen Sie mit <CR> ab. Es geschieht hier im Gegensatz zu BASIC nichts! Warum? -- Sie haben ja nur mit dem Editor einen Text geschrieben, mehr nicht ! Der Text muß erst vom Compiler einmalig in ein Maschinenspracheprogramm umgewandelt werden. Verlassen Sie dazu den Editor mit <CTRL-K> <CTRL-D> und starten Sie den Compilerlauf mit <C>. Nach einer kurzen Zeit erscheint auf dem Bildschirm anstatt der erwarteten String-Ausgabe eine Fehlermeldung: "BEGIN expected". Der Compiler vermißt das Wort "BEGIN". Und damit fängt es mit der oben erwähnten Genauigkeit bereits an. Man muß dem Compiler ausdrücklich mitteilen, daß der Programmtext anfängt. Das geschieht durch das "Reservierte" Wort BEGIN. Unter "Reservierten Wörtern" versteht man in PASCAL Wörter mit besonderen Bedeutungen. Es gibt in TURBO-PASCAL verschiedene reservierte Wörter. Sie sind im Handbuch zu einer Tabelle zusammengefaßt.

Also schnell den Editor mit <E> aktiviert und den Text geändert:

```
BEGIN  
WRITE('alphaTronic');
```

Ein neuer Compilerlauf ergibt immer noch nicht das gewünschte Ergebnis, sondern eine neue Fehlermeldung: "Unexpected end of source". Der Compiler hat jetzt zwar den Textanfang erkannt, aber er vermißt den Hinweis zum Programmtext-Ende. Auch das muß in PASCAL mitgeteilt werden und zwar mit dem reservierten Wort

END, gefolgt von einem Punkt. Wenn der Text nun so geändert wird:

```
BEGIN  
WRITE('alphaTronic');  
END.
```

kann endlich auch ein fehlerfreier Compilerlauf gestartet werden. Die Übersetzung geschieht hier natürlich sehr schnell, mit Wartezeiten müssen Sie erst bei deutlich größeren Programmen rechnen. Jetzt endlich kann das Programm mit <R> gestartet werden. Sie dürfen eine kurze Pause machen: Ihr erstes PASCAL-Programm läuft.

Obwohl die Beschreibung schon sehr ausführlich war, gibt es noch weitere wissenswerte Dinge zu unserem kleinen Programm. Nach dem erfolgreichen compilieren können Sie ohne weiteren Compilerlauf das Programm beliebig oft starten. Die Übersetzungsarbeit muß nämlich nur einmal durchgeführt werden. Um das Programm abzusichern, wird im Optionen-Menü der COM-File angewählt und ein weiterer Compilerlauf gestartet. Jetzt wird das Ergebnis nicht in den RAM, sondern auf die Diskette geschrieben. Der File erhält den von Ihnen gewählten Namen des Workfile mit der Extension ".COM". Sie können das Programm dann ohne TURBO-PASCAL direkt von CP/M aus starten.

Lassen Sie sich die Größe des COM-Files einmal mit STAT anzeigen. Erschreckt werden Sie feststellen, daß der generierte File, trotz superkurzem Programm, 8k belegt. Sind das die kurzen Maschinenspracheprogramme? -- Nein! Bei TURBO-PASCAL muß sich unabhängig von der Größe, in jedem COM-File die PASCAL-"runtime library" befinden. Es handelt sich dabei um eine Routinen-Sammlung, die das Arbeiten erst ermöglicht. Der Routinenteil belegt unabhängig von der eigenen Programmgröße bereit fast 8k. Obwohl in unserem kleinen Beispielprogramm nur ein Teil dieser Routinen genutzt wird, schreibt der Compiler immer die gesamte runtime-library in einen COM-File.

Wenn Sie den Text unseres Beispiels nicht auf dem Bildschirm, sondern auf einen Drucker ausgeben möchten, müssen Sie den Inhalt der WRITE-Klammer ändern:

```
WRITE('alphaTronic');      -->      WRITE(LST,'alphaTronic');
```

Durch das Einfügen von LST wird die Ausgabe zum Drucker geleitet. Bei der Bildschirmausgabe könnten Sie auch: WRITE(CON,'alphaTronic') schreiben (CON für CONSOLE; siehe

CP/M). Bei TURBO ist diese Angabe aber optional. Man kann allerdings einen Nutzen daraus ziehen. Wenn Sie ein Programm mit Druckerausgaben zunächst auf dem Bildschirm testen möchten, lassen Sie den Editor einfach alle LST in CON umwandeln. Das geht sehr einfach, denn für den Austausch von Wörtern verfügt der Editor über besondere Befehle.

3.2. Eingabe-Anweisungen & Variable

Nachdem unser "Ausgabe-Programm" nun läuft, soll sich hier mit der Dateneingabe auseinandergesetzt werden. Für den aus BASIC bekannten INPUT-Befehl gibt es in PASCAL die READ-Anweisung. Wie bei WRITE und WRITELN erfolgt auch hier nach READ kein Zeilenvorschub. Wird ein solcher erwünscht, kann die Anweisung READLN benutzt werden. Die von der Tastatur übernommenen Werte werden an Variable übergeben. Diese Variablen müssen in der READ-Klammer genannt werden :

```
BEGIN
WRITE('Drei Zahlen, durch je ein Leerzeichen getrennt,
     eingeben: ');
READLN(Zahl1,Zahl2,Zahl3);
WRITELN(Zahl1,Zahl2,Zahl3);
END.
```

Das Programm soll drei Zahlen von der Tastatur einlesen und anschließend auf dem Bildschirm ausgeben. Geben Sie den Text ein und starten Sie den Compiler. Ergebnis: Schon wieder eine Fehlermeldung! "Unknown identifier or syntax error"; der Compiler "kennt" diese Variablennamen nicht. BASIC würde in diesem Fall die drei Variablen anlegen und deren Werte auf Null setzen. PASCAL macht nichts, was Sie nicht ausdrücklich anordnen. Ein "aus Versehen" falsch eingegebener Variablenname wird dadurch sofort entlarvt! Damit ist eine gefürchtete BASIC-Fehlermöglichkeit ausgeschlossen.

Wie wird's gemacht?

Sie müssen vor dem Programmblock die Variablen definieren. Das geschieht durch das reservierte Wort "VAR", gefolgt vom Variablennamen und -typ. Damit Sie keine erklärenden Namen "ausknobeln" müssen, sind bei TURBO alle Zeichen signifikant. Die Groß- und Kleinschreibung bleibt unbeachtet, was ebenfalls zu besserer Lesbarkeit beiträgt. Die Bezeichnung "AngestelltenPersonalNummer" läßt sich eben leichter lesen als "ANGESTELLTENPERSONALNUMMER". Auch der Unterstrich ist erlaubt: "Angestellten_Personal_Nummer". Sie sollten ruhig lange und erklärende Namen benutzen. Nach dem Compilieren ergibt sich in

der Programmgröße kein Unterschied. Sie konnten in BASIC mit verschiedenen Variablentypen arbeiten: Integer, reelle Zahlen mit einfacher Genauigkeit, reelle Zahlen mit doppelter Genauigkeit und Strings. In PASCAL ist noch viel mehr möglich:

Typ	Art	Speicherbedarf
BYTE	: ganze Zahlen (0 bis 255)	1 Byte
INTEGER	: ganze Zahlen (-32768 bis 32767)	2 Bytes
REAL	: reelle Zahlen (1E-38 bis 1E+38) 11 signifikante Stellen	6 Bytes
BOOLEAN	: Bool'sche Wahrheit (TRUE/FALSE)	1 Byte
CHAR	: ein ASCII-Zeichen	1 Byte
STRING	: Zeichenkette bis 255 Zeichen	1 Byte + Länge
nnnnnnn	: eigene definierte Typen	

Ändern Sie also das Beispiel um :

```
VAR
    Zahl1,Zahl2,Zahl3 : INTEGER;

BEGIN
    WRITE('Drei Zahlen, durch je ein Leerzeichen getrennt,
    eingeben: ');
    READLN(Zahl1,Zahl2,Zahl3);
    WRITELN(Zahl1,Zahl2,Zahl3);
END.
```

Jetzt kann der Text kompiliert und das Programm ausgeführt werden. Beachten Sie, daß in PASCAL definierte Variable zufällige Werte annehmen. Wenn Sie also die Variablen vor ihrer ersten Zuweisung anzeigen, können unsinnige Zeichen auf dem Bildschirm erscheinen.

```
VAR
    Zeichenkette : STRINGÄ100Ü;

BEGIN
    WRITELN(Zeichenkette);
END.
```

Berechnungen würden verfälscht! Wenn nicht gewährleistet ist, daß das Programm eine Zuweisung durchführt, müssen Sie dafür sorgen, daß die Variablen "vernünftige" Werte erhalten. Der Zuweisungsoperator ist in PASCAL der Doppelpunkt, gefolgt von einem Gleichheitszeichen.

```

VAR
    Zahl          : REAL;
    Zeichenkette  : STRING$200Ü; (* Die Länge eines Strings *)
                                (* muß in eckigen Klammern *)
                                (* angegeben werden. Bei der *)
                                (* PC-Tastatur entspricht *)
                                (* das dem Ä und Ü *)

BEGIN
    Zahl := 1;      Zeichenkette := ``; (* Unsinn verhindern *)
    WRITELN(Zahl, `` ,Zeichenkette);
END.

```

Um das Programm verständlich zu machen, können beliebig viele Kommentare eingefügt werden. Kommentare müssen mit `(*` oder `ä` begonnen und mit `*)` oder `ü` abgeschlossen werden. Der Compiler "überliest" diese Hinweise beim Übersetzen.

3.3. Fehlerhafte Eingaben

Bei TURBO gibt es eine interessante Möglichkeit, die Anzahl der bei einer Eingabe zulässigen Zeichen zu begrenzen. Während normalerweise bis zu 127 Zeichen (Bufferlänge) angenommen werden, kann dieser Wert durch Zuweisung an die automatisch vordefinierte Variable "BufLen" geändert werden:

```

VAR
    Satz : STRING$50Ü;

BEGIN
    BufLen := 50;      (* braucht nicht mit VAR deklariert werden *)
    WRITE('Sie können bis zu 50 Zeichen eingeben : ');
    READLN(Satz);
    WRITELN(Satz);
END.

```

Damit ist es im Gegensatz zu BASIC hier nun leicht möglich dafür zu sorgen, daß Bildschirmmasken nicht "kaputt" geschrieben werden können. Die Anweisung "BufLen:= x" hat nur für eine einzige READ- oder READLN-Anweisung Gültigkeit. Nach der Eingabe wird BufLen automatisch wieder auf 127 gesetzt. Die Bufferbegrenzung ist eine Möglichkeit, Eingabefehler zu verhindern. TURBO ist nämlich sehr "streng"! Verkehrte Eingaben, z.B. ein String statt einer Zahl, führen unweigerlich zu einem Programmabbruch. Wie man solche Fehler abfangen kann, wird später beschrieben.

3.4. Ausgabeformatierung

Wenn Sie das zuletzt beschriebene Programm laufen lassen, wird

die Zahl "Eins" als: 1.0000000000E+00 ausgegeben. Diese Darstellung wird nur in den wenigsten Fällen erwünscht sein. TURBO bietet Ihnen zwar die Möglichkeit die Ausgabe zu formatieren, aber so leistungsfähig wie der PRINT USING-Befehl in BASIC ist die Anweisung nicht. Sie müssen in der WRITE-Klammer durch Doppelpunkt getrennt die minimal gewünschte Gesamt-Ausgabelänge angeben. Eine weitere Zahl nach einem weiteren Doppelpunkt definiert die Anzahl der Nachkommastellen. Wenn der Wert also in einem 7-stelligen Feld mit 3 Nachkommastellen angezeigt werden soll, ergibt sich folgende Schreibweise:

```
WRITELN(Zahl:7:3);
```

Ähnliches gilt für Zeichenketten. Die Anweisung:

```
WRITELN('Test':20);
```

setzt den String rechtsbündig in ein 20-Spalten-Feld.

3.5. Konstante und Typen

Ähnlich wie bei den Variablen, lassen sich vor dem Programmblock auch Konstante und Typen definieren. Es werden dazu die reservierten Wörter "CONST" bzw. "TYPE" benutzt. Unter "Konstante" versteht man Werte, die im Programmverlauf nicht verändert werden. Zuweisungen an eine Konstante führen zu einer Fehlermeldung. Die Definition von Konstanten kann ebenfalls helfen, Programme lesbarer zu gestalten. Der Programmierer kann außerdem Programme viel leichter ändern. Er braucht ggf. nur an einer Stelle (bei der Konstanten-Definition) Werte zu wechseln. Es erübrigt sich, im gesamten Programmtext nach den zu ändernden Angaben zu suchen.

Durch TYPE-Festlegungen können Sie neue Variablentypen einführen oder vorhandene Typen umdefinieren. Bevor Sie sich in PASCAL nicht "sicher bewegen", sollte man diese Möglichkeiten nicht unbedingt benutzen.

```
CONST
```

```
    MehrwertSteuer = 14.0;
```

```
    MaxBufferSize  = 10;
```

```
TYPE
```

```
    Str10 = STRINGÄ10Ü;
```

```
VAR
```

```
    Produkt      : Str10;
```

```
    NettoPreis,  (* Es dürfen mehrere Variablennamen *)
```

```
BruttoPreis : REAL; (* durch Komma getrennt, genannt
werden! *)
```

```
BEGIN (* --- Hauptprogramm --- *)
WRITE('Geben Sie die Produktbezeichnung ein (10 Zeichen) : ');
BufLen := MaxBufferSize;  READLN(Produkt);
WRITELN;
WRITE('Geben Sie den Netto-Preis ein : ');
READLN(NettoPreis);
WRITELN;  WRITELN;  WRITELN;  (* 3 Leerzeilen Abstand *)
BruttoPreis := NettoPreis + ((NettoPreis / 100) * Mehrwert
Steuer);
WRITE(Produkt, ' kostet ', BruttoPreis:7:2, ' DM. ');
END.
```

4. Wiederholungen und Entscheidungen

Aus BASIC sind Ihnen die FOR/NEXT- und die WHILE/WEND-Schleifen bekannt. Diese und weitere Möglichkeiten sind auch in PASCAL vorhanden.

4.1. Die FOR-Anweisung

Wenn z.B. 80 Unterstriche ausgegeben werden sollen, ist das wie folgt möglich:

```
VAR
    i : BYTE;

BEGIN
    FOR i:= 1 TO 80 DO      WRITE('_ ');
END.
```

Beachten Sie, daß nach dem "DO" kein Semikolon gesetzt werden darf. Das Semikolon signalisiert hier das Ende der Schleifenanweisung. Es ist in PASCAL aber auch möglich, beliebig viele Befehle in die Schleife aufzunehmen. Die Anweisungen müssen einfach mit einem weiteren BEGIN und END "geklammert" werden. Sie dürfen übrigens im Programmtext beliebig viele Blöcke mit BEGIN und END klammern. Auch wenn Sie es z.B. "nur" der Übersichtlichkeit wegen wollen.

```
CONST
    Add = 3.1;

VAR
    i      : BYTE;
    Summe : REAL;
```



```

BEGIN
Summe:= 0;
FOR i:= 1 TO 80 DO
  BEGIN
    WRITE(' ');          (* Erste Anweisung in der Schleife *)
    Summe:= Summe + Add;  (* Letzte Anweisung in der Schleife *)
  END; (* FOR i *)
WRITELN('Summe : ',Summe:10:1);
END.

```

Jetzt beziehen sich die Schleifendurchläufe auf alle Anweisungen innerhalb des inneren BEGIN/END-Blocks. Wichtig ist, daß nur nach dem letzten END ein Punkt folgen darf, da dieser Punkt dem Compiler ja das Ende vom Programmtext anzeigt. Die Schleifenvariable kann auch abwärts gezählt werden. In diesem Fall ist das TO durch ein DOWNT0 zu ersetzen.

```

FOR i:= 80 DOWNT0 1 DO      BEGIN  Anweisungen END;

```

Das BASIC-bekannte "STEP" ist leider nicht möglich, da diese Schleife bei TURBO nicht mit REAL-Zahlen kontrolliert werden kann. Innerhalb der Schleife darf mit der Zählvariablen gerechnet werden. Zuweisungen sind im Gegensatz zu BASIC aber nicht statthaft. Auch der Anfangs- oder der Endwert der Schleife darf nicht geändert werden. Das liegt daran, daß in BASIC die Schleife jedesmal NEU interpretiert wird. In TURBO hat der Compiler diese Arbeit bereits einmalig vor dem Programmstart erledigt.

4.2. Eine Schleife mit WHILE

Wie in BASIC wird auch in TURBO hinter WHILE die Bedingung definiert. Ein WEND ist nicht erforderlich, da der Anweisungsblock mit BEGIN und END geklammert wird.

```

VAR
  Eingabe : STRING80Ü;

BEGIN
  Eingabe:= '';
  WHILE Eingabe<> 'ENDE' DO
    BEGIN
      Buflen:= 80;
      WRITE('Ihr Eingabetext : ');  READLN(Eingabe);
    END;
  END.

```

Wenn bereits zu Beginn die Schleifenbedingung unwahr (FALSE)

ist, wird die WHILE-Schleife nicht ausgeführt.

4.3. Noch eine Möglichkeit : REPEAT..UNTIL

Hierbei brauchen die Anweisungen in der Schleife nicht mit BEGIN und END geklammert werden, da dieser Effekt bereits durch REPEAT und UNTIL selbst erreicht wird:

VAR

Eingabe : STRING\$80;

BEGIN

Eingabe:= '';

REPEAT

 Buflen:= 80;

 WRITE('Ihr Eingabetext : '); READLN(Eingabe);

UNTIL Eingabe= 'ENDE';

END.

Im Unterschied zur WHILE-Schleife wird hier nicht die Start-, sondern die Abbruch-Bedingung geprüft. Also wird die REPEAT/UNTIL-Schleife auf jeden Fall mindestens einmal durchlaufen.

4.4. IF..THEN..ELSE

Diese Entscheidungs-Anweisung ist dem BASIC-Programmierer bekannt. In TURBO sind die gleichen Befehle möglich. Der Vorteil von PASCAL zeigt sich dadurch, daß eine IF- Anweisung nicht auf eine Programmzeile beschränkt ist. Die Befehlssequenzen bei einer erfüllten, oder auch nichterfüllten Bedingung dürfen beliebig lang sein. Es müssen nur die jetzt schon bekannten BEGIN..END-Klammern gesetzt werden:

IF A=B THEN BEGIN

 Anweisung 1;

 Anweisung 2;

 Anweisung n;

 END

(* 1. END *)

ELSE BEGIN

 Anweisung 1;

 Anweisung 2;

 Anweisung n;

 END;

(* 2. END *)

Durch dieses Beispiel wird die Schreibweise erkennbar. Beachten Sie, daß nach dem 1.END kein Semikolon stehen darf. Das Semikolon nach dem zweiten END teilt dem Compiler mit, daß hier die IF..THEN..ELSE-Anweisung beendet ist.

4.5. Die Auswahlanweisung CASE OF

Ähnlich dem aus BASIC bekannten ON X GOSUB gibt es auch in PASCAL eine Mehrfach-Auswahlmöglichkeit. Durch Abfrage von einem Ausdruck wird ein bestimmtes Programmverhalten erreicht. Es können wahlweise direkte Anweisungen ausgeführt, oder verschiedene Unterprogramme aufgerufen werden (siehe hierzu das nächste Kapitel). Ebenfalls ist es möglich, anstatt eines Einzelbefehls einen mit BEGIN..END geklammerten Anweisungsblock einzusetzen.

VAR

DiskettenAnzahl : INTEGER;

BEGIN

WRITE('Wieviele Disketten besitzen Sie : ');

READLN(DiskettenAnzahl);

CASE DiskettenAnzahl OF

1..9 : WRITELN('Sie besitzen nicht einmal eine Packung !');

10..99 : BEGIN

WRITELN('Sie haben schon eine ganze Menge.');

WRITELN; WRITELN;

END;

END; (* CASE *)

WRITE('Die CASE-Anweisung ist beendet.');

END.

Die beiden Punkte zwischen den Ziffern (z.B. bei 1..9) bedeuten, daß ein beliebiger Wert innerhalb der Grenzwerte (einschließlich) als logisch wahr (TRUE) akzeptiert wird. Trifft bei CASE keine der angegebenen Möglichkeiten zu, so wird die dem END von CASE unmittelbar folgende Anweisung ausgeführt. Man hat allerdings die Möglichkeit, auch darauf zu reagieren. Es muß einfach eine ELSE-Anweisung in die CASE-Klammer eingefügt werden:

VAR

DiskettenAnzahl : INTEGER;

BEGIN

WRITE('Wieviele Disketten besitzen Sie : ');

READLN(DiskettenAnzahl);

CASE DiskettenAnzahl OF

1..9 : WRITELN('Sie besitzen nicht einmal eine Packung !');

10..99 : BEGIN

WRITELN('Sie haben schon eine ganze Menge.');

WRITELN; WRITELN;

END;

```
ELSE   WRITELN('Ihre Antwort war nicht vorgesehen !');
END;   (* CASE *)
WRITE('Die CASE-Anweisung ist beendet. ');
END.
```

4.6. Der unbedingte Sprung

Auch ein GOTO-Befehl ist in TURBO-PASCAL vorhanden. Von seinen BASIC-Programmen weiß man ja, daß viele GOTO-Anweisungen ein Programm unübersichtlich und schwer verständlich machen. Darum sollte man insbesondere in PASCAL auf diese Anweisung möglichst ganz verzichten. Schließlich bietet TURBO durch seine mögliche Programmierung in Blöcken andere Techniken als BASIC. Sollte ein GOTO doch einmal unabänderlich sein, so darf in TURBO mit GOTO eine Markierung (Label) angesprungen werden. Alle Markierungen müssen vor dem Programmblock durch das reservierte Wort LABEL deklariert werden.

```
LABEL
    Stop;

VAR
    i : INTEGER;

BEGIN
REPEAT
    BufLen:= 4;
    WRITE('Wert eingeben : ');  READLN(i);
    IF i< 0 THEN GOTO STOP;
    WRITELN(i);
UNTIL i= 999;
Stop :
END.
```

Die Markierung muß im Programm mit einem nachfolgenden Doppelpunkt dargestellt werden. Das kleine Beispielprogramm würde in leicht abgeänderter Form auch ohne GOTO-Anweisung das gleiche Ergebnis liefern:

```
VAR
    i : INTEGER;

BEGIN
REPEAT
    BufLen:= 4;
    WRITE('Wert eingeben : ');  READLN(i);
    IF i>= 0 THEN WRITELN(i);
```

```
UNTIL (i= 999) or (i< 0);
END.
```

5. Prozeduren und Funktionen

BASIC-Programmierer sind es gewohnt, öfter benötigte Programmteile zusammenzufassen. Diese "Unterprogramme" werden dann mit GOSUB nnnn angesprungen und mit RETURN wieder verlassen. PASCAL bietet mit seinen "Prozeduren" und "Funktionen" noch viel leistungsfähigere Programmier-Möglichkeiten an.

Prozeduren und Funktionen sind genau wie die bisher besprochenen Programmteile aufgebaut. Alle dort erklärten Anweisungen haben auch hier Gültigkeit. Lediglich nach dem letzten END einer Prozedur bzw. Funktion darf kein Punkt (PROGRAMM-Ende) stehen, sondern hier wird mit dem Semikolon das Prozedurende angezeigt.

5.1. Einfache Prozeduren als Beispiel;

Nehmen wir an, Sie wollten bei einem Eingabeprogramm jeden Begriff durch eine durchgezogene Line abteilen. Das Programm könnte etwa so aussehen:

```
CONST
    Strich      = '-';
    DoppelStrich = '=';
    MaxDat      = 99;
    MaxLen      = 70;

TYPE
    Str70 = STRING$MaxLenÜ;

VAR
    Zaehler,Eingabe      : byte;
    Ch                   : Char;
    NameUndTelefon,Anschrift : ARRAY$1..MaxDatÜ OF Str70;
                                (* So werden in TURBO Felder definiert *)

BEGIN
    Eingabe:= 0;
    FOR Zaehler:= 1 TO MaxDat DO
        BEGIN
            NameUndTelefon$ZaehlerÜ := '';
            Anschrift$ZaehlerÜ      := '';
        END; (* FOR Zaehler *)
    REPEAT
        Eingabe:= Eingabe + 1;
        WRITELN('Geben Sie den ',Eingabe:2,'. Namen mit Tel-Nr.
```

```

ein :');
  Buflen:= MaxLen;
  READLN(NameUndTelefonÄEingabeÜ);
  FOR Zaehler:= 1 TO 80 DO      WRITE(Strich);
  WRITELN('Geben Sie die ',Eingabe:2,'. Anschrift ein :');
  Buflen:= MaxLen;
  READLN(AnschriftÄEingabeÜ);
  FOR Zaehler:= 1 TO 80 DO      WRITE(DoppelStrich);
UNTIL (Eingabe = MaxDat) or (NameUndTelefonÄEingabeÜ = '');
WRITELN('*** Eingabe beendet ***');
FOR Zaehler:= 1 TO 80 DO      WRITE(DoppelStrich);
WRITE('Nach einem Tastendruck werden die Daten angezeigt : ');

READ(KBD,CH);  (* liest ein Zeichen von der Tastatur *)
               (* entspricht dem "Ch$= INPUT$(1)" aus BASIC *)
WRITELN;
Eingabe:= 1;
WHILE (NameUndTelefonÄEingabeÜ > '') AND (Eingabe <= MaxDat) DO
  BEGIN
    WRITELN(NameUndTelefonÄEingabeÜ);
    FOR Zaehler:= 1 TO 80 DO      WRITE(Strich);
    WRITELN(AnschriftÄEingabeÜ);
    FOR Zaehler:= 1 TO 80 DO      WRITE(DoppelStrich);
    Eingabe:= Eingabe + 1;
  END;  (* WHILE *)
WRITELN('*** Programm beendet ***');
END.

```

Das Programm läßt sich durch die Verwendung von Prozeduren wesentlich übersichtlicher gestalten:

```

CONST
  Strich      = '-';
  DoppelStrich = '=';
  MaxDat      = 99;
  MaxLen      = 70;

TYPE
  Str70 = STRINGÄMaxLenÜ;

VAR
  Zaehler,Eingabe      : byte;
  Ch                   : Char;
  NameUndTelefon,Anschrift : ARRAYÄ1..MaxDatÜ OF Str70;
                        (* So werden in TURBO Felder definiert *)

```

Procedure Initialisieren;

```
BEGIN
  Eingabe:= 0;
  FOR Zaehler:= 1 TO MaxDat DO
    BEGIN
      NameUndTelefonÄZaehlerÜ := '';
      AnschriftÄZaehlerÜ      := '';
    END; (* FOR Zaehler *)
  END; (* Initialisieren *)
```

Procedure EStrich;

```
BEGIN
  FOR Zaehler:= 1 TO 80 DO    WRITE(Strich);
  END; (* EStrich *)
```

Procedure DStrich;

```
BEGIN
  FOR Zaehler:= 1 TO 80 DO    WRITE(DoppelStrich);
  END; (* DStrich *)
```

Procedure DatenEingabe;

```
BEGIN
  REPEAT
    Eingabe:= Eingabe + 1;
    WRITELN('Geben Sie den ',Eingabe:2,'. Namen mit Tel-Nr.
              ein :');
    Buflen:= MaxLen;
    READLN(NameUndTelefonÄEingabeÜ);
    EStrich;
    WRITELN('Geben Sie die ',Eingabe:2,'. Anschrift ein :');
    Buflen:= MaxLen;
    READLN(AnschriftÄEingabeÜ);
    DStrich;
  UNTIL (Eingabe = MaxDat) or (NameUndTelefonÄEingabeÜ = '');
  WRITELN('*** Eingabe beendet ***');
  END; (* Eingabe *)
```

Procedure AufEinZeichenWarten;

```
BEGIN
  WRITE('Nach einem Tastendruck werden die Daten
        angezeigt : ');

  READ(KBD,Ch); (* liest ein Zeichen von der Tastatur *)
                (* entspricht dem "Ch$= INPUT$(1)" aus
                BASIC *)

  WRITELN;
  END; (* AufEinZeichenWarten *)
```

```

Procedure Ausgabe;
BEGIN
  Eingabe:= 1;
  WHILE (NameUndTelefonÄEingabeÜ > '') AND (Eingabe <= MaxDat)
    DO
      BEGIN
        WRITELN(NameUndTelefonÄEingabeÜ);
        EStrich;
        WRITELN(AnschriftÄEingabeÜ);
        DStrich;
        Eingabe:= Eingabe + 1;
      END; (* WHILE *)
    END; (* Ausgabe *)

BEGIN      (* --- Hauptprogramm --- *)
  Initialisieren;
  DatenEingabe;
  AufEinZeichenWarten;
  Ausgabe;
  WRITELN('*** Programm beendet ***');
END.

```

Prozeduren können durch einfache Nennung ihres Namen beliebig oft aufgerufen werden. Das Hauptprogramm läßt sich nun wesentlich einfacher lesen. Die groben Abläufe werden sofort verstanden. Die einzelnen Prozeduren lassen sich einfacher ändern oder erweitern. Damit sind alle Teile eines TURBO-PASCAL-Programms angesprochen. Der schematische Programmaufbau sollte so aussehen:

```

+-----+
! LABEL                                !
! CONST                               !
! TYPE                               !
! VAR                                !
! Procedure/Function                 !
+-----+
! Hauptprogramm                       !
+-----+

```

Obwohl TURBO die Reihenfolge im Deklarationsteil nicht vorschreibt, sollte man einen gleichbleibend systematischen Aufbau wählen.

5.2. Die Prozeduren richtig angewandt

Das letzte Programm aus 5.1. ist lauffähig. Die Deklarationen und Definitionen am Anfang haben auch für die Prozeduren Gültigkeit. Man nennt diese Eigenschaft "global". Eine "globale Variable" kann im gesamten Programm geändert und angezeigt werden. Das ist für einen BASIC-Programmierer nichts Neues. BASIC-Variable sind bei MBASIC oder DISKBASIC immer global. Trotzdem ist diese Eigenschaft u.U. sehr hinderlich. Der Programmierer hat immer darauf zu achten, daß ja kein Wert geändert wird, der von einem anderen Programmteil noch benötigt wird. Außerdem lassen sich schlecht "Universal-Routinen" schreiben und bei den Programm-Erstellungen einbinden, da man nicht weiß ob es einen Konflikt mit den gewählten Variablenamen gibt.

PASCAL bietet daher die Möglichkeit, "LOKALE" Variable zu benutzen. Die Prozedur EStrich aus 5.1. könnte also geändert werden:

```

Procedure EStrich;
  CONST
    Strich = '-';
  VAR
    b : BYTE;
  BEGIN
    FOR b:= 1 TO 80 DO
      WRITE(Strich);
    END; (* EStrich *)

```

"Strich" und "b" haben hier nur lokale Bedeutung. Ihr Gültigkeitsbereich ist in diesem Fall die Prozedur. Die Variable "b" ist außerhalb der Prozedur nicht bekannt. Ein Aufruf würde zu einer Fehlermeldung führen. Die Konstante "Strich" existiert außerhalb der Prozedur ebenfalls, da sie auch dort definiert ist. Da "Strich" aber auch innerhalb der Prozedur definiert ist, kann der Wert völlig unterschiedlich sein. Eine innere Definition hat Vorrang. Strich-Außen und Strich-Innen können also wie verschiedene Namen behandelt werden. Die Möglichkeit lokale Variable zu deklarieren, ist sicherlich eine der schönsten und zweckmäßigsten Annehmlichkeiten von PASCAL. Sie können z.B. die Prozedur "EStrich" in der modifizierten Form in jedem weiteren Programm einbinden, ohne das Schwierigkeiten mit dem anderen Programmtext auftauchen. Beachten Sie, das auch Prozeduren eigene Prozeduren beeinhalteln können, deren Bedeutung dann ebenfalls lokal zur Hauptprozedur ist. Oft benötigte Prozeduren können mit einem eigenen Namen versehen auf die Diskette geschrieben werden. In beliebigen Programmtexten lassen sie sich als "INCLUDE"-Dateien wieder einbinden. Das be-

deutet, daß der Compiler bei einer INCLUDE-Anweisung vom RAM zur Disk verzweigt und anschließend nach der INCLUDE-Anweisung weiterarbeitet. Wenn z.B. der INCLUDE-File den Namen: "EINGABE.PAS" trägt, dann lautet die entsprechende Anweisung: ä\$I EINGABE.PAS ü. Lesen Sie hierzu Ihr TURBO-PASCAL-Handbuch.

Wenn man nun die beiden modifizierten Prozeduren EStrich und DStrich miteinander vergleicht, dann fällt sofort auf, daß alle Anweisungen gleich sind. Lediglich die Konstante ist unterschiedlich.

-----+-----	
<pre> Procedure EStrich; CONST Strich = '-'; VAR b : BYTE; BEGIN FOR b:= 1 TO 80 DO WRITE(Strich); END; (* EStrich *) </pre>	<pre> Procedure DStrich; CONST Strich = '='; VAR b : BYTE; BEGIN FOR b:= 1 TO 80 DO WRITE(Strich); END; (* DStrich *) </pre>
-----+-----	

Wenn man nun eine Möglichkeit hätte, der Prozedur einen Wert zu übergeben, dann könnten von diesem Programmteil beliebige Zeichen ausgedruckt werden. Auch diese Möglichkeit ist in PASCAL vorgesehen. Man muß einfach die gewünschten Parameter im Prozedurkopf erwähnen. Dabei ist auch die Nennung des Parametertyps erforderlich:

```

VAR
  Zeichen : CHAR;

Procedure ZeichenKetteAusgeben ( Ch : CHAR );
  VAR
    b : BYTE;
  BEGIN
    FOR b:= 1 TO 80 DO
      WRITE(Ch);
    END; (* ZeichenKetteAusgeben *)

BEGIN (* --- Hauptprogramm --- *)
  ZeichenKetteAusgeben(' ');
  Zeichen := '-';
  ZeichenKetteAusgeben(Zeichen);
END.

```

Sie können jetzt beliebige Zeichen ausgeben. Die Auswahl geschieht entweder direkt ('+'), oder durch eine Variable (Zeichen). Zum Zeitpunkt des Aufrufs wird der Variableninhalt in die Prozedur-Variable kopiert; d.h. in unserem Beispiel hat sich der Wert von "Zeichen" nicht geändert. Man nennt diese Art der Wertübergabe : "call by value".

Wenn wir jetzt auch noch die Anzahl der auszugebenden Zeichen übergeben, haben wir eine Prozedur programmiert, die dem STRING\$-Befehl von BASIC entspricht :

VAR

Zeichen : CHAR;

Procedure ZeichenKetteAusgeben (Anzahl : BYTE; Ch : CHAR);

VAR

b : BYTE;

BEGIN

FOR b:= 1 TO Anzahl DO

WRITE(Ch);

END; (* ZeichenKetteAusgeben *)

BEGIN (* --- Hauptprogramm --- *)

ZeichenKetteAusgeben(40,'+');

Writeln;

Zeichen := '+';

ZeichenKetteAusgeben(50,Zeichen);

END.

Damit haben wir bereits ein brauchbares Unterprogramm. Einen äquivalenten Befehl gibt es nämlich in TURBO nicht. Die festgelegte Prozedur kann im Programm wie eine Anweisung benutzt werden. Wenn Parameter "call by value" übergeben werden, ändert sich der Variablenwert im Hauptprogramm nicht. Manchmal wäre es aber von Nutzen, wenn die Prozedur Variablenwerte im Hauptprogramm ändern könnte. Darum gibt es eine weitere Möglichkeit, Parameter zu übergeben. Die Angaben im Prozedurkopf müssen ausdrücklich als variabel erklärt werden:

VAR

x : REAL;

Procedure Quadrat (VAR Zahl : REAL);

BEGIN

Zahl:= Zahl * Zahl;

END; (* Quadrat *)

BEGIN (* --- Hauptprogramm --- *)

```
x:= 5.1;
Quadrat(x);
WRITELN(x:7:2);
x:= 30.6;
Quadrat(x);
WRITELN(x);
END.
```

Diese Art der Parameterübergabe heißt "call by reference". Hierbei wird die Information über den Speicherplatz der Variablen übergeben. Die Prozedur ändert den Wert, und das Hauptprogramm arbeitet mit den neuen Werten weiter. Der Vollständigkeit halber sei hier erwähnt, daß auch Prozeduren oder Funktion als Parameter übergeben werden können. In TURBO-PASCAL sind bereits viele "Standard-Prozeduren" enthalten. Über diese Unterprogramme kann der Programmierer sofort verfügen.

5.3. Funktionen

Sinngemäß läßt sich das für Prozeduren gesagte auf Funktionen übertragen. Während eine Prozedur wie eine Anweisung gehandhabt werden kann, entspricht eine Funktion eher einer Variablen. Im Funktionskopf wird daher der Datentyp der Funktion angegeben:

```
VAR
  x : REAL;

Function Quadrat ( Zahl : REAL ) : REAL;
  BEGIN
    Quadrat:= Zahl * Zahl;
  END; (* Quadrat *)

BEGIN (* --- Hauptprogramm --- *)
  x:= 5.1;
  ClrScr;          (* Standard-Prozedur : löscht den Bildschirm *)
  WRITELN(Quadrat(x));
  x:= 30.6;
  WRITELN(Quadrat(x):5:2);
END.
```

Außerdem muß dem Funktionsnamen ein Wert zugewiesen werden. Diesen Wert liefert die Funktion dann an das Hauptprogramm. In TURBO-PASCAL sind verschiedene Standard-Funktion implementiert, über die der Programmierer sofort verfügen kann.

6. Diskettenzugriff

TURBO-PASCAL ermöglicht im Gegensatz zu anderen PASCAL-Versionen nicht nur den sequentiellen, sondern auch den direkten Zugriff auf Diskettendaten. Da die Behandlung von RANDOM-Dateien Kenntnisse über "RECORD's" und andere Strukturen voraussetzt, wird im Folgenden nur der Umgang mit sequentiellen Dateien beschrieben.

Sequentielle Dateien verhalten sich in TURBO wie "normale" ASCII-Texte; d.h. jeder Eintrag (jede Zeile) ist mit einem RETURN und einem LINEFEED abgeschlossen. Bevor eine solche Datei benutzt werden kann, muß eine entsprechende "File-variable" deklariert werden. Anschließend kann der Filevariablen der Name des Diskettenfiles zugeordnet werden. Auf die Datei kann mit READ und WRITE zugegriffen werden. Bevor der Zugriff aber möglich ist, muß die Datei entweder mit RESET für Lesevorgänge, oder mit REWRITE für Schreibvorgänge eröffnet werden. Das kleine Beispielprogramm zeigt die Möglichkeiten Dateien zu eröffnen, zu beschreiben und zu lesen :

```
LABEL
```

```
    Stop;
```

```
CONST
```

```
    Max = 79;
```

```
VAR
```

```
    WorkFile : TEXT;           (* Filevariable *)
```

```
    Daten    : STRINGAMaxÜ;
```

```
    FileName : STRINGA14Ü;
```

```
    b        : BYTE;
```

```
Procedure Beep;
```

```
    BEGIN      WRITE(#7)      END;    (* erzeugt einen Ton *)
```

```
Procedure OpenOutput;
```

```
    BEGIN
```

```
        REWRITE(WorkFile);      (* Öffnet die Datei und bereitet *)  
                                (* zum Schreiben vor. ACHTUNG : *)  
                                (* Evt. vorhandene Daten gehen *)  
                                (* verloren ! *)
```

```
        GOTOXY(26,1);           (* Standard-Prozedur zur CURSOR-  
                                Positionierung *)
```

```
        WRITE('Datei zum Schreiben geöffnet');
```

```
    END; (* OpenOutput *)
```

```
Procedure OpenInput;
```

```
    BEGIN
```

```
RESET(WorkFile);          (* Öffnet die Datei und bereitet *)
                           (* zum Lesen vor ! *)
GOTOXY(28,1);
WRITE('Datei zum Lesen geöffnet');
ClrEol;                    (* Standard-Prozedur : löscht alle Zeichen *)
                           (* bis zum Zeilenende *)
END; (* OpenInput *)

Procedure DateiSchliessen;
BEGIN
  CLOSE(WorkFile); (* Die Datei wird geschlossen *)
  GOTOXY(26,1);   ClrEol;
  END; (* DateiSchliessen *)

BEGIN (* --- Hauptprogramm --- *)
  ClrScr;
  Daten:= '';
  GOTOXY(1,12);
  WRITE('Geben Sie eine gültige Dateibezeichnung ein : ');
  BufLen:= 14;   READLN(FileName);
  IF Length(FileName) < 1 THEN GOTO STOP; (* Programm-Abbruch *)
  ASSIGN(WorkFile,FileName);             (* Name wird übergeben *)
  OpenOutput;
  GOTOXY(1,12);   ClrEol;
  WRITELN('Dateneingabe kann mit ENDE beendet werden :');
  WHILE Daten<> 'ENDE' DO
    BEGIN
      GOTOXY(1,13);   ClrEol;
      BufLen:= Max;
      READLN(Daten);
      If Daten = '' THEN Beep ELSE
        IF Daten<> 'ENDE' THEN WRITELN(WorkFile,Daten);
      END; (* WHILE *)
    DateiSchliessen;
  FOR b:= 12 TO 13 DO
    BEGIN
      GOTOXY(1,b);
      ClrEol;
    END; (* FOR b *)
  OpenInput;
  GOTOXY(1,3);
  WHILE NOT EOF(WorkFile) DO (* Die Standard-Funktion EOF *)
    (* testet ob das File-Ende *)
    (* erreicht ist. *)
    BEGIN
      READLN(WorkFile,Daten);
      WRITELN(Daten);
    END; (* WHILE *)
```



```
        WRITELN(' >>> FEHLER <<<');  
        END;  
UNTIL NOT Fehler;  
WRITE('Die Eingabe war fehlerfrei !');  
END.
```

Im Beispiel soll eine Zahl eingegeben werden. In einem Programm ohne Fehlerroutine würde eine Stringeingabe zum Programmabbruch führen. Hier wurde aber vor der READLN-Anweisung die Fehlerüberwachung ausgeschaltet. Erfolgt nun eine Fehleingabe, liefert IOResult einmalig einen Wert über Null. Bei einem weiteren IOResult-Aufruf würde wieder der Wert Null geliefert! Darum wurde der Fehler in einer Variablen gespeichert. Solange eine Fehlermeldung vorliegt, verlangt das Programm eine neue Eingabe.

8. Schlußbemerkung

Wenn Sie alle Kapitel sorgfältig durchgearbeitet haben, sollten Sie jetzt in der Lage sein, Ihre BASIC-Kenntnisse in PASCAL anzuwenden. Sie dürfen dabei aber nicht vergessen, daß die meisten Annehmlichkeiten, die von PASCAL ausgehen, noch gar nicht behandelt wurden. Die Anwendungen von z.B. : RECORDS, MENGEN und ZEIGERN stellen mächtige Programmiermöglichkeiten dar. Eine Einführung in diese Techniken hätte allerdings den Rahmen dieses Artikels gesprengt. Nach dem Motto "WENIGER kann MEHR sein" sollten Sie erst einmal die hier beschriebenen Übungen verstanden haben. Anschließend werden Ihnen die vielen Bücher, die zum Thema PASCAL erschienen sind, sicherlich weiterhelfen, die Feinheiten von PASCAL - und insbesondere von TURBO-PASCAL - zu erlernen.

Das Profi-Software-Konzept von TA: Qualität vor Quantität.

TA bietet Ihnen für seine alphanetronic Personal Computer leistungsfähige, erprobte und auf die Anforderungen deutscher Anwender zugeschnittene Software. Programme, die sofort und problemlos laufen. Fordern Sie Qualität in Hard- und Software.

TA Triumph-Adler AG
Fürther Straße 212 · D-8500 Nürnberg 80
Tel. (0911) 3 22-0 · Telex 6-23 295
Teletex 9118 203 TA tele

TA TRIUMPH-ADLER



Einstieg in die Maschinensprache

Patrick V. Thomas

EINLEITUNG

Ein Personal-Computer, wie der TA-PC, ist im Grunde ein Alleskönner. Um seine zahlreichen Anwendungsmöglichkeiten auch tatsächlich zu nutzen, bedarf es oft einer zeitaufwendigen Programmierung, entweder durch den Benutzer selbst oder durch unabhängige System- und Software-Häuser.

Die Freude am "Selbst-Programmieren" kann allerdings ein jähes Ende erfahren, sobald man mit seinem Computer-Sprach-Latein an die Grenzen des Machbaren gelangt ist. In dieser Hinsicht ist der reine BASIC-Anwender sehr gefährdet. Dies hängt damit zusammen, daß 100% BASIC-Programme sehr langsam und zusätzlich noch speicherplatzfressend (durch ihre Länge) sind. An diesem Punkt beginnt man sich umzuhören, der Begriff der Maschinensprache taucht auf. Ab und zu schielt man zwar zu ihr hinüber, scheint aber meist wenig geneigt zu sein, den Schritt in das Zahlenabenteuer zu wagen.

Nun, diesen ersten Schritt einer Einführung werden wir wagen und gleichzeitig, mehr oder weniger nebenbei, ein sinnvolles BASIC-/Maschinensprache-Programm erstellen.

Unter den vielfältigen Peripherie-Geräten, die man heutzutage an den eigenen Home-/Personal-Computer anschließen kann, steht der Drucker ganz oben auf der Wunschliste vieler PC-Anwender. Insbesondere für Textverarbeitungsaufgaben kommt man um die Anschaffung eines Matrix- oder Typenrad-Druckers nicht herum.

Der TA-PC, zur Textverarbeitung wie geschaffen, hat in dieser Hinsicht einen würdigen Partner, den Typenrad-Drucker TRD 7020, bekommen.

Dieser, an fast alle PC's anschließbare, von TA entwickelte Typenraddrucker sticht vor allem aufgrund seiner vielfältigen Gestaltungs- und Kontrollmöglichkeiten aus dem Heer der Konkurrenten hervor.

Das von uns als Ergebnis der Maschinensprache-Einführung angestrebte Anwendungsprogramm in BASIC/MASCHINENSPRACHE wird die "Features" des TRD 7020 weitgehend zu nutzen wissen. In Anlehnung an seine Aufgabe haben wir es BRIEFMASCHINE genannt. Es wird deutlich, daß es sich um "irgendein Druckprogramm" handeln muß, doch lassen Sie sich überraschen.

Zunächst müssen Sie sich nämlich durch die Grundzüge der TA-PC-Maschinensprache arbeiten und zu einer grundlegenden, abstrakteren Denkweise bereit sein. Der insgesamt höhere Programmieraufwand wird mit kürzestmöglichen Ausführungszeiten und extremer Speicherplatz-Ökonomie belohnt. Am besten, Sie gehen einfach mit der realistischen Vorstellung, "ich lerne eine - im Grunde simple - neue Programmiersprache", in die ganze Angelegenheit.

1. GRUNDLAGEN

Der wichtigste Baustein im TA-PC, wie in jedem anderen Computer auch, ist sein Mikroprozessor. Der im PC verwendete "Z-80A" von der Firma ZILOG wird, in Lizenz, auch von anderen Halbleiter-Herstellern gebaut und erfreut sich einer hohen Popularität. Um seine Arbeitsweise zu verstehen und ihn schließlich auch selbst zu programmieren, müssen wir die nun folgende Theorie über uns ergehen lassen.

Betrachten wir zunächst die grundlegenden Aufgaben eines Computer-Systems: es erlaubt INFORMATIONEN zu speichern, auszugeben und zu verarbeiten. Die Informations-Speicherung geschieht in RAM- oder ROM-Bausteinen. Diese Chips unterscheiden sich in einem wesentlichen Punkt. Während ein RAM-Baustein (RAM = Random Access Memory ... wahlfreier Zugriff) vom Anwender abgefragt - und mit Informationen beschrieben werden kann, dient der ROM-IC (ROM = Read Only Memory ... Nur-Lese-Speicher) als reiner Abfragespeicher, d.h. die in ihm enthaltenen Informationen können vom Anwender nur abgefragt ("gelesen") und nicht verändert werden.

In einem Computer wird jede Information codiert gespeichert. Zwei Codierungsarten, die sich allerdings stark von einander unterscheiden, werden wir nun kennenlernen: den Binär-Code und, auf höherer Ebene, den ASCII-Code. Der ASCII-Code ist ein Zahlen-Code, der computerintern als Binär-Code abgespeichert ist, d.h.: jede beliebige Zahl wird auf der Basis des Binär-Codes in eine für die Bausteine des Computer-Systems verständliche Form gebracht und derart auch gespeichert.

Das BIT. Ein Bit stellt die kleinste Informationseinheit in der EDV dar. Das bedeutet: unabhängig davon, was wir dem Computer an Informationen zuführen - es wird letztendlich in eine Kombination mehrerer Bits umgewandelt. Ein solches Bit muß man sich wie einen Schalter vorstellen, den man entweder auf AN oder AUS stellen kann oder anders gesagt: der nur 2 Zustände kennt.

In Zukunft werden wir diese zwei Zustände eines Bits mit 0 und 1 bezeichnen. Entscheidend für unsere Überlegungen ist dabei, daß jeder Speicherbaustein, ob RAM oder ROM, auf solch einer Bit-Struktur aufbaut, d.h. aus einer Vielzahl derartiger Schalter bzw. Bits besteht. Nun stellt sich natürlich die Frage, wie man mit derlei Bits und Bit-Kombinationen Informationen codieren bzw. darstellen kann.

Vergleichen wir beispielsweise eine einfache Rot/grün-Ampel mit einem BIT und legen folgende Zuordnung fest:

Das Bit ist "0" = Ampel rot

Das Bit ist "1" = Ampel grün

Anders ausgedrückt: wenn das Bit "0" ist, darf ich nicht fahren und erst wenn das Bit "1" ist, habe ich freie Fahrt. Dadurch, daß wir den Bit-Zuständen eine bestimmte Bedeutung zuordnen, können wir mit diesem Bit Informationen anzeigen und speichern.

Interessant wird es aber erst, wenn wir mehrere Bits zusammennehmen und als Gruppe betrachten. Mit mehr als einem Bit können wir natürlich auch, mehr als nur 2 Zustände bzw. Informationen speichern, der Weg zur Zahlen-Codierung durch Bits wird frei.

Bei 2 Bits zum Beispiel ergeben sich schon 4 Kombinations-Möglichkeiten, nämlich:

	Bit Nr.1	Bit Nr.2
1.Kombination	0	0
2.Kombination	0	1
3.Kombination	1	0
4.Kombination	1	1

Dies kann man nun mit beliebig vielen Bits fortführen und dadurch immer größere Kombinations-Möglichkeiten erreichen, die gleichzeitig auch mehr speicherbare Informationen bedeuten.

Das BYTE. Ein Byte ist der Oberbegriff für eine Gruppe von acht Bits.

Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit
Nr.7	Nr.6	Nr.5	Nr.4	Nr.3	Nr.2	Nr.1	Nr.0
0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1

Die 8 Bits eines Bytes erhalten jeweils eine "Platznummer", von 7 bis 0. Das Bit Nr.0 wird niederwertigstes Bit und dementsprechend Bit Nr.7 höchstwertigstes Bit genannt. Ein Byte, das sei vorweggenommen, kann 256 verschiedene Informationen (z.B. die Zahlen von 0 bis 255) darstellen, d.h. speichern. Wenn man sich die Mühe machte und jede mögliche Kombination von 8 Bits zählen würde, dann käme man tatsächlich auf die 256:

Bit-Nr	7	6	5	4	3	2	1	0
0.	0	0	0	0	0	0	0	0
1.	0	0	0	0	0	0	0	1
2.	0	0	0	0	0	0	1	0
3.	0	0	0	0	0	0	1	1
4.	0	0	0	0	0	1	0	0
.								.
.								.
.								.
254.	1	1	1	1	1	1	1	0
255.	1	1	1	1	1	1	1	1

Genau 256 mögliche Kombinationen !

Um diese Bit-Kombinationen auch zur Zahlen-Codierung und damit zur Zahlenspeicherung verwenden zu können, benötigen wir eine Regel bzw. Umrechnungsart, sodaß jede Kombination von zunächst einmal 8 Bits genau einer Zahl (zwischen 0 bis 255) entspricht.

Der Binär-Code baut auf der Tatsache auf, daß jedem Bit, sei es in einem oder auch in mehreren Bytes, aufgrund seiner Platznummer ein bestimmter Wert zugesprochen wird. Um schließlich von der Binär-Darstellung zur natürlichen Zahl zu kommen, summiert man die Werte der Bits, deren Zustand "1" ist, und läßt die "0"-Bits unbeachtet. Das mag zunächst etwas verwirrend klingen,

ist es aber, sobald man die Umrechnung zwei-/ dreimal am Beispiel vollzogen hat, bald nicht mehr: schließlich steckt eine recht simple Logik dahinter, die von folgenden Werten ausgeht:

Bit-Nr	7	6	5	4	3	2	1	0
Wert	128	64	32	16	8	4	2	1

Wären z.B. die Bits 2 und 5 eines Bytes auf "1", dann entspräche das einem Zahlen-Wert von $4 + 32 = 36$. Das Byte in Binär-Code:

Binär 0 0 1 0 0 1 0 0 = 36 Dezimal

Würde der PC von uns die Information "36" erhalten, dann stünde diese Zahl in der o.a. codierten Form in den Speicherbausteinen des Systems. Bei einem POKE <Adresse> 36 in Basic passiert genau das gleiche: das Byte Nr. <Adresse> enthält den Wert 36 als

0 0 1 0 0 1 0 0 Bit-Kombination.

Auf diese Art und Weise wurden die 256 möglichen 8-Bit-Kombinationen zur Darstellung der Zahlen 0 bis 255 verwendet. Noch ein paar Beispiele zum Üben (immer von Bit 7 bis Bit 0 zu lesen):

0 0 0 0 0 0 0 0 = 0

0 0 0 1 0 0 0 0 = 16

1 0 0 0 0 0 0 0 = 128

0 0 0 0 1 1 1 0 = 14

1 0 0 0 0 0 0 1 = 129

0 1 1 1 1 1 1 1 = 127

1 1 1 1 1 1 1 1 = 255

Umgekehrt, d.h. von einer Zahl zu ihrem Binär-Code, versucht man vom höchsten zum niedrigsten Bit-Wert vorzugehen. Die Zahl 94 z.B. soll in ihren Binär-Code überführt werden.

Hierzu ziehen wir den höchstmöglichen Binär-Code, die 64, ab und verfahren mit dem Rest in gleicher Weise - bis wir bei 0 angelangt sind.

1: 94 - 64 (Bit 6) = 30	BYTE : 0 1 0 0 0 0 0 0
2: 30 - 16 (Bit 4) = 14	BYTE : 0 1 0 1 0 0 0 0
3: 14 - 8 (Bit 3) = 6	BYTE : 0 1 0 1 1 0 0 0
4: 6 - 4 (Bit 2) = 2	BYTE : 0 1 0 1 1 1 0 0
5: 2 - 2 (Bit 1) = 0 !	BYTE : 0 1 0 1 1 1 1 0

Ergebnis : die Zahl 94 entspricht der Bit-Kombination ...

0 1 0 1 1 1 1 0

Dieses Verfahren läßt sich in beiden Richtungen natürlich auch auf größere Bit-Kombinationen anwenden. War die Anzahl der möglichen 8-Bit-Kombinationen = 2 hoch 8 = 256, so steigt der Wert bei z.B. 2 Bytes (16 Bits) schon auf 2 hoch 16 = 65536! Mit zwei Bytes lassen sich demnach bis zu 65536 Zahlen, z.B. von 0 bis 65535 darstellen, mit drei Bytes schon... und so weiter. Der Z-80 Mikroprozessor arbeitet meist mit ein oder zwei Bytes auf einmal.

Die Speicherkapazität wird in der Regel mit ? KByte angegeben, beim TA-PC z.B. : 64 KByte. Dieses K als Einheit bedeutet nicht Kilo (1000), sondern 1024. Wenn von "64 K" die Rede ist, handelt es sich zunächst mal von einem unpräzisen, verwechselbaren Ausdruck. Ist von "64 KBYTE" die Rede, sind also nicht 64 000, sondern 64 * 1024 = 65536 Bytes gemeint. Bei Speicherbausteinen gilt es allerdings weiter zu unterscheiden. Dort werden nicht Bytes, sondern Bits angegeben. Die Bezeichnung 256K-Speicherbaustein bedeutet nicht 262.144 Bytes, sondern Bits, d.h. 32.768 Bytes, die in diesem Baustein gespeichert werden können.

Der augenscheinliche Zusammenhang zwischen dem 64 KByte-Speicherplatz des TA-PC und den 65536 möglichen Kombinationen in 16 Bit (= 2 Bytes) bringt uns auf die Tatsache, daß der Mikroprozessor genau 16 Bit dazu verwendet, seine demzufolge maximale Speicherkapazität von 65536 Bytes zu verwalten.

Die Speicherstelle. Eine Speicherstelle ist genau 1 Byte dieser Gesamtspeicherkapazität. Um den ganzen Haufen Speicherstellen auch unterscheiden zu können, erhält jede Speicherstelle eine Zahl, die sog. Adresse. Beim TA-PC sind es die Adressen 0 bis 65535, die auch tatsächlich je einem Byte entsprechen. Wenn wir uns eine Speicherstelle als Schublade vorstellen, dann ergeben die 65536 Speicherstellen des TA-PC eine riesige Kommode. Von 0 bis 65535 durchnummeriert, liegt eine Schublade über der anderen.

Der ASCII-Code. Wie jeder andere Computer auch, kann der TA-PC nur Zahlen verarbeiten. Um ihn dazu zu bringen, auch Texte oder Sonderzeichen zu verarbeiten, d.h. auch zu speichern, wurde ein Code geschaffen, der die Eigenheiten eines Computers mit der Notwendigkeit, Informationen zu standardisieren, verbindet. Der sog. ASCII-Code steht zunächst einmal für die Zahlen 0 bis 255. Jede Zahl entspricht einem bestimmten Zeichen bzw. einer bestimmten Funktion. Der ASCII-Code enthält neben dem Alphabet (Groß- und Klein-Buchstaben), den Ziffern 0 bis 9, noch einige Sonderzeichen, wie z.B. das Fragezeichen, die Klammern "()" .., etc. Eine Tabelle, die den gesamten Zeichen-Vorrat des TA-PC auflistet, finden Sie in den BASIC- und Disk-BASIC Handbüchern. Der Vorteil des Ganzen steckt in der Tatsache, daß dieser Code ein Standard ist und damit auf praktisch allen gängigen PCs (fast) gleich ist. Daraus ergibt sich erst die Möglichkeit, z.B. Texte auf einem Computer-System einzutippen und diese auf ein anderes System zu übertragen, dort wieder lesbar zu machen und weiter zu verarbeiten.

Allerdings beschränkt sich diese Austauschmöglichkeit auf die "reinen" ASCII-Codes: die Zahlen von 0 bis 127. Am Beispiel des TA-PC läßt sich zum Beispiel erkennen, daß die Zahlen ab 128 im wesentlichen für Semi-Grafik und weitere Sonder-Zeichen reserviert sind, die natürlich nicht auf jedem Computer vorhanden sind. Auf einem Sirius I entsprächen die Zahlen von 128 bis 255 z.B. ganz anderen Sonderzeichen. Der Bereich von 0 bis 127 ist deshalb meist gleich, weil dort einige Standard-Kommandos (von 0 bis 31), die Ziffern von 0 bis 9 (48 bis 57), das große und das kleine Alphabet (von 65 bis 126) sowie Sonderzeichen (wie Interpunktions-Zeichen, etc.), Platz finden.

Die Information, die Sie dem Computer eintippen, hat also zwei Hürden zu nehmen. Ihre Eingabe wird einmal als ASCII-Code angenommen und intern als Binär-Code verarbeitet bzw. gespeichert. Ein Beispiel aus der Praxis: Bei der einfachsten Aktion - "Sie tippen etwas in die Tastatur ein und es erscheint auf dem Bildschirm" - passiert folgendes. Nehmen wir an, der Cursor steht ganz links oben auf dem Bildschirm und Sie tippen das Wörtchen "test" ein. Der Computer hat jeden Tastendruck registriert und entsprechend den getippten Zeichen, die zugehörigen ASCII-Codes in die Speicherstellen geschrieben. Welche Speicherstellen ?

Der Video-RAM-Bereich. Alles was Sie auf dem Bildschirm sehen, jedes Zeichen und jede Bildschirmstelle überhaupt, entspricht einem Byte im Gesamtspeicher des TA-PC. Dieser ganz spezielle Speicherbereich wird Video-RAM-Bereich genannt und ist genau 40 mal 24 bzw. 80 mal 24 Zeichen groß. Auf dem Bildschirm er-

scheint nur das, was an ASCII-Informationen (Text, Zahlen..) im VIDEO-RAM-BEREICH steht. Die erste Speicherstelle des VIDEO-RAM ist bei Adresse 61400, d.h. stünde in 61440 der ASCII-Code z.B. eines Fragezeichens (Code = 63), dann würde in der ersten Bildschirmstelle ganz links oben ein Fragezeichen erscheinen.

Doch zurück zu unserem "test". Der Beweis dafür, daß die ASCII-Codes des Wörtchens "test", d.h. für t - e - s - und t auch wirklich in den ersten Speicherstellen des VIDEO-RAM abgelegt wurden, ist leicht zu liefern. Wir brauchen uns nur den Inhalt der Speicherstellen 61440 bis 61443 ausgeben zu lassen (PEEK) und der PC listet uns die Codes auf:

Speicherstelle	Zeichen	ASCII-Code
61440	"t"	116
61441	"e"	101
61442	"s"	115
61443	"t"	116

Unsere Aussage, daß jede Information codiert gespeichert wird, wird so langsam greifbarer. Dieser Text zum Beispiel wurde mit einem Textverarbeitungs-Programm erstellt und ist auf eine Diskette gespeichert. Auf dieser Diskette ist jedes Zeichen, jeder Buchstabe in je einem Byte gespeichert - als Zahl bzw. ASCII-Code. Aus dem, was der ASCII-Code an Zeichen zur Verfügung stellt, setzt sich jedes Schriftstück, jede Kalkulation...etc zusammen.

Die Maschinensprache. Die Programmierung in Maschinensprache bringt uns auf die tiefste Ebene der Systemprogrammierung überhaupt. Es sind dabei jedoch zwei Begriffe auseinanderzuhalten: die MASCHINENSPRACHE und der MASCHINENCODE. Die Maschinensprache (auch ASSEMBLER genannt) entspricht einer "normalen" Programmiersprache mit etwas "grundlegenderen" Befehlen. D.h. im Klartext: die Befehle der Programmiersprache ASSEMBLER sind nicht derart umfassend, wie z.B. der BASIC-Befehl PRINT USING oder die FOR-NEXT-Konstruktionen, die im Grunde aus lauter Einzelschritten bestehen, sondern erlauben es gerade, diese Einzelschritte in eigener Regie und nach eigenen Vorstellungen zu programmieren. Die ASSEMBLER-Befehle beziehen sich z.B. oft direkt auf bestimmte Speicherstellen oder sogar auf einzelne Bits. Obwohl die Befehle der ASSEMBLER-Sprache eher Kürzel sind, kann man doch von einer "vollwertigen Programmiersprache der niedrigsten Stufe" sprechen.

Wie jede andere Programmiersprache auch, muß ASSEMBLER in den

MASCHINENCODE "übersetzt" werden. Für höhere Programmiersprachen existiert ein Übersetzerprogramm, das die im ASCII Format - bzw. beim TA-BASIC in einer Kurzform als sog. Tokens, auf die wir an dieser Stelle nicht näher eingehen müssen - vorliegenden Programmbefehle in vom Mikroprozessor ausführbaren Zahlencode übersetzt. Beim eingebauten TA-BASIC wird dies von einem sog. INTERPRETER getan. Bei ASSEMBLER-Programmen tut dies eben ein "ASSEMBLER".

Der Maschinencode ist der Zahlen-Code, der letztendlich vom Mikroprozessor ausgeführt wird. Solange der PC in Betrieb ist, geht der Mikroprozessor systematisch eine Speicherstelle nach der anderen durch und führt, je nachdem was für Zahlen er vorfindet, bestimmte Befehle aus. Ein ASSEMBLER-Befehl wird also zunächst in einen Zahlen-Code umgewandelt, der dann im Speicher abgelegt und vom Computer abgearbeitet wird.

Diese Codierungs-Arbeit nimmt uns der ASSEMBLER ab. Prinzipiell gilt das auch für die Interpreter und Compiler, nur auf einer sehr viel höheren Ebene.

In der Praxis gilt: Kleinere Z-80-Routinen kann man von Hand codieren, bei größeren Programmen ist ein ASSEMBLER unerlässlich. Schließlich will man nicht in einem Zahlen-, Codier und Berechnungswust untergehen.

Die Maschinensprache des Z-80-Mikroprozessors ist im 8-Bit-Prozessorenbereich eine der umfang- und befehlsreichsten überhaupt.

Der Datenbus. Der Z-80 ist ein 8-Bit Prozessor. Das heißt, daß sein Datenbus, über den er Daten z.B. mit seinen Speicherbausteinen austauscht, 8-Bit breit ist: es werden 8 Bits auf einmal verarbeitet. Einen solchen 8 Bit breiten "Bus" muß man sich wie 8 parallele Leitungen ("Leiterbahnen") vorstellen, über die 8 Bits bzw. Bit-Informationen gleichzeitig auf die Reise geschickt werden können. Der Datenbus ist die Verbindung zum Mikroprozessor und wird bei jedem Befehl, der in irgendeiner Weise Daten berührt, als "Haupt-Daten-Verkehrsader" genutzt. Um auch zu wissen, wo ("aus welcher Speicherstelle") er sich als nächstes Daten herholen soll, besitzt der Z-80 Chip eine zweite Bus-Einrichtung: den ADRESSBUS.

Der Adressbus. "Wie schon erwähnt, werden vom Z-80 16 Bit zur Speicherverwaltung benutzt". Nun, dies sind die 16 Bits: der sog. Adressbus. Das einzige, was dieser Bus zu transportieren hat, sind Adressen von bestimmten Speicherstellen, die als

weitere Datenquellen ausersehen sind. Beide Bus-Systeme sind interna des Prozessors und brauchen nur einmal registriert zu werden.

Um Daten nicht nur auszutauschen, sondern auch verarbeiten zu können, stellt der Z80 dem Programmierer Pseudo-Variablen, sog. REGISTER, zur Verfügung. Ein Register ist 8 Bit groß. Es können aber auch bestimmte Register gemeinsam zur Aufnahme von 16 Bit angesprochen werden (Beispiel: HL, BC, DE). Das Register kann entweder Zahlen von 0 bis 255 oder von 0 bis 65535 speichern und verarbeiten. Die Bezeichnung jedes einzelnen Registers (z.B. A, B, L oder iX ...) ist festgelegt (von ZILOG und durch die ASSEMBLER) und unveränderlich. Diese zwei Punkte machen den Unterschied zu den gewohnten BASIC-Variablen deutlich.

Wenn man sich das so vor Augen führt, ...Speicherstellen, Befehle, Register..., dann scheint dieses ASSEMBLER tatsächlich eine ganz normale Programmiersprache zu sein. Nun, im Prinzip schon, allerdings ist die ASSEMBLER-Sprache immer an einen bestimmten Mikroprozessor gebunden, repräsentiert genau dessen Befehle. Nicht mehr und nicht weniger. Praktisch bedeutet das, daß ein in Z-80-Maschinensprache geschriebenes Programm nicht auf einem anderen Prozessor laufen kann. Der Vorteil höherer Programmiersprachen, die Kompatibilität bzw. Übertragbarkeit auf andere Rechner, fällt weg.

Andererseits kann man nur in ASSEMBLER alle Möglichkeiten eines Computer-Systems optimal ausnutzen. Assembler-Programme sind nicht nur extrem schnell, sondern meist auch um ein Vielfaches kürzer, als z.B. vergleichbare BASIC-Programme.

Wenn wir uns in dieser Einführung mit den Befehlen des Z-80-Mikroprozessors auseinandersetzen und hier und da, wo es meist auch nicht nötig ist, der Problemstellung mal nicht ganz, ganz tief auf den Grund gehen sollten, bedenken Sie bitte, daß das quasi Standard-Werk zur Z-80-Programmierung von Rodney Zaks ca. 600 Seiten umfaßt.

Die Hilfsmittel. Neben den Registern, die keinen Speicherplatz beanspruchen, sondern eigentlich "extra-Schublädchen" sind, kann der Programmierer noch auf eine Vielzahl von Hilfen des Z80 zurückgreifen: die FLAGS.

Die Flags. Ein Flag ist ein Zeichen, eine Anzeige für ein bestimmtes Ereignis. Neben den Registern, die wir im Folgenden noch ausführlicher behandeln werden, sind die Flags der

wichtigste Prozessor-Bestandteil. Zum besseren Verständnis der Flag-Benutzung sei das Beispiel eines einfachen Zähl-Programmes herangezogen, einmal als BASIC- und einmal als ASSEMBLER-Programm.

* BASIC *

```
10 A = 100
20 A = A - 1
30 IF A <> 0 THEN GOTO 20
40 STOP oder END oder RETURN (oder gar nichts)
```

In der ersten Zeile definieren wir eine Variable, dann vermindern wir sie (Zeile 20) und tun dies solange, bis der Wert der Variable die Null erreicht hat (Zeilen 20 und 30). Zuletzt wird z.B. wie aus einem Unterprogramm mit RETURN zurückgesprungen. In ASSEMBLER benutzen wir anstatt einer Variablen ein 8-Bit-Register, genannt A (für "Akkumulator").

```
LD  A, 100      ; das Register A wird mit dem Wert 100 geladen
DEC A          ; und wird dann um 1 vermindert ("DECREMENT")
JR  NZ, -3      ; falls A nicht null wird zu DEC A gesprungen
RET            ; wie BASIC-RETURN, aus Unterprogramm zurück
```

Das ist schon eine komplette, wenn auch bescheidene, Maschinensprach-Routine. Die Kürzel LD, DEC, JR und RET stehen jeweils für einen ASSEMBLER- bzw. Maschinensprachbefehl. Zeilennummern gibt es bei der ASSEMBLER-Programmierung nicht.

Beide Unterprogramme machen das gleiche: sie zählen von 100 bis 0. Der "Befehlswortlaut" der ASSEMBLER-Kürzel lautet:

"LOAD Register A mit 100"

"DECREMENT Register A"

"JUMP RELATIVE IF NOT ZERO -3 Bytes"

"RETURN"

Von Interesse ist für uns zunächst nur das "IF NOT ZERO" (falls nicht null). An dieser Stelle wird nämlich ein FLAG bemüht, das sog. ZERO-Flag. Ein Flag ist 1 Bit, das auf bestimmte Ereignisse reagiert und diese durch Zustands-Änderung (0 bzw 1) anzeigt. Das ZERO-FLAG reagiert z.B. darauf, ob der Wert NULL bei der letzten Befehls-Ausführung aufgetreten ist. Falls ja, wird das ZERO-FLAG bzw. BIT auf "1" gesetzt: es wird ein ZERO angezeigt, wohingegen meist ein NOT ZERO vorherrscht, d.h. ZERO-

Bit = "0". Es gibt nicht nur das ZERO-Flag. Da jedes Flag nur ein Bit benötigt, wurde ein spezielles 8-Bit-Register (6 FLAGS) als FLAG-Speicher eingebaut, das FLAG-Register F. Von den 8 Bits werden 5 als Flags für den Programmierer reserviert, wovon wir 4 vorstellen werden. Die 8 Bits des Flag-Registers "F" bedeuten:

Bit-Nr	Funktion
0	Das CARRY-Flag "C", es zeigt einen "Übertrag" an
1	- (nur Prozessor-intern benutzt)
2	Das PARITÄTS- Flag (PE/PO) hat mehrere Funktionen
3	- (wird nicht genutzt)
4	Das Halb-Übertrags-Flag "H", wird nicht besprochen
5	- (wird nicht genutzt)
6	Das ZERO-Flag "Z", meistbenutzt neben dem C-Flag
7	Das SIGNUM-Flag (Vorzeichen) "S", Minus/Plus-Flag

Das ZERO-FLAG wird, wie alle anderen auch, in Verbindung mit bestimmten Befehlen - Sprungbefehlen - abgefragt und entscheidet bei derartigen "bedingten" Sprungbefehlen darüber, ob der Befehl ausgeführt oder übergangen werden soll. Die schon benutzte Abkürzung "NZ" steht für NOT ZERO, ein einfaches "Z" für ZERO.

Das CARRY-FLAG. Das C-Flag zeigt einen "Übertrag" an. Ein Übertrag tritt auf, wenn z.B. bei einer Rechenoperation ein Wert größer als 255 entsteht und nur ein 8-Bit-Register zur Speicherung vorgesehen war. Die über die 255 hinausgehenden Bits fallen weg und das C-Flag wird gesetzt, d.h. CARRY= "1". Außerdem existieren spezielle Befehle zur Manipulation einzelner Bits z.B. eines Registers, die ebenfalls Einfluß auf das Carry-Flag haben. Die Verwendung als Universalflag durch den Programmierer wird durch besondere ASSEMBLER-Befehle unterstützt, die ein direktes setzen von "1" bzw. zurücksetzen auf "0" des C-Flags gestatten. Das Kürzel "C" steht für CARRY, ein "NC" für NOT CARRY.

Das SIGNUM-FLAG, auf deutsch Vorzeichen-Flag, zeigt an, ob das

Ergebnis der letzten Rechenoperation positiv oder negativ ist. Maßgebend für die Unterscheidung in Plus und Minus ist das Bit Nr. 7 eines Byte-Wertes. Um keine Mißverständnisse aufkommen zu lassen: an der internen Bit-Struktur ändert sich nichts. Das höchstwertigste Bit wird nur als "Hilfe" genommen, um ein Plus oder Minus zu "simulieren". Dies schränkt natürlich den Wertebereich ein, da von 8 Bits nur 7 zur Werte-Codierung genutzt werden können: von -127 bis +127. Die Abkürzungen sind "P" für Plus und "M" für Minus.

Das PARITÄTS-FLAG. Die Parität bestimmt man, indem die Anzahl der "1er" in einem Byte gezählt wird. Eine gerade Anzahl wird gerade Parität genannt, "PARITY EVEN", und bei einer ungeraden Anzahl herrscht eine ungerade Parität, "PARITY ODD". Das P-Bit ist "1" bei PARITY EVEN und "0" bei PARITY ODD. Das Paritäts-Bit kennt auch den Namen Prüf-Bit. Dies rührt daher, daß "einfache" Übertragungsfehler durch die Parität erkannt werden können. Ein Byte, das auf die Reise geschickt wird, erhält als Begleiter ein derartiges "Prüf-Bit". Tritt nun bei der Übertragung, z.B. über die Telefon-Leitung, ein Fehler auf ändert sich mit hoher Wahrscheinlichkeit das Verhältnis der "1" und "0" dieses Bytes. Ein Vergleich mit dem Paritäts-Bit führt zur Fehlererkennung. Entsprechend den Bezeichnungen ODD und EVEN werden als Kürzel "PO bzw. "PE" angegeben.

Die Flag-Kürzel können in Verbindung mit JR ("JUMP RELATIVE"), JP ("JUMP"), CALL (Vergleich: GOSUB) und RET ("RETURN")- Befehlen verwendet werden. Zum Beispiel RET Z ("RETURN IF ZERO")....

Alle Flag-Kürzel nochmals auf einen Blick :

FLAG	KÜRZEL	BEDEUTUNG	BIT
ZERO	Z	"falls Null aufgetreten"	Z=1
ZERO	NZ	"falls nicht Null aufgetreten"	Z=0
CARRY	C	"falls Carry-Bit gesetzt"	C=1
CARRY	NC	"falls kein Carry-Bit gesetzt"	C=0
PARITY	PE	"gerade Parität"	P=1
PARITY	PO	"ungerade Parität"	P=0
SIGNUM	P	"Plus"	S=0
SIGNUM	M	"Minus"	S=1

Achtung: das FLAG-REGISTER kann vom Programmierer als REGISTER nicht beeinflußt (abgesehen vom CARRY-Bit) und schon gar nicht als Datenspeicher genutzt werden. Es sind nur die Flags, die bei der Programmierung beachtet werden müssen.

Die MNEMONICS. Dieser Zungenbrecher ist der Fachbegriff für das, was wir bisher als Befehlskürzel kennengelernt haben, z.B. JR, RET oder CALL.

Die REGISTER sind das A und O des Z-80-Mikroprozessors. Der Z-80 kennt folgende Register : A, B, C, D, E, H, L, F, ix, iy, SP, PC, I und R. Darüberhinaus stellt der Mikroprozessor bei bestimmten Operationen einen sonst "unsichtbaren" zweiten Register-Satz A',F',B',C',D',E',H' und L' zur Verfügung. Auf diesen Register-Satz hat der Programmierer ausschließlich durch sog. Austausch-Befehle Einfluß. Doch zunächst die Universalen.

Das REGISTER A. Das "A" steht für AKKUMULATOR und stellt ein 8-Bit-Register dar. Der weitaus größte Teil aller Rechen-, Vergleichs- und Logik-Befehle "läuft" über den Akkumulator und endet auch dort mit einem Ergebnis. Er ist, so gesehen, der universalste aller 8-Bit-Universal-Register.

Die 8-BIT-REGISTER. Als Universal-8-Bit-Register stehen weiterhin zur Verfügung: B, C, D, E, H und L. Sie können beliebige Daten beinhalten und, paarweise zusammengefaßt, als 16-Bit Register agieren (BC, DE und HL). Dabei nimmt das Registerpaar HL eine ähnlich zentrale Stellung ein wie der Akkumulator im 8-Bit-Bereich. Der linke Buchstabe gibt immer das "höherwertige" Byte an, d.h. B (BC), D (DE) bzw. H (HL). Die Buchstaben H / L helfen da als kleine Eselsbrücke: H wie HIGH und L wie LOW. In der Praxis ist diese Unterscheidung von großer Bedeutung.

MSB und LSB. Eine Adresse, durch 16-Bits dargestellt, besteht aus zwei BYTES: dem höherwertigen und dem niederwertigen Byte. Das MSB (More Significant Byte) stellt die Bits mit den höheren Platznummern und das LSB (Less Significant Byte), die mit den niedrigeren Platznummern. Bei der Adressenspeicherung gilt:

"Zuerst das LSB und in die nächste Speicherstelle das MSB"

Eine Adresse (z.B. 2049), in den Speicherstellen 3000 und 3001 gespeichert, wäre folgendermaßen codiert worden:

Adresse	Byte	Bedeutung
3000	1	LSB der Adresse 2049
3001	8	MSB der Adresse 2049

P.S: Adresse bzw. 16-Bit-Wert = $LSB + 256 * MSB$ (hier: $1+8*256$)

Ein LD HL, (3000) "lade HL aus Speicherstelle 3000,3001" hätte

zur Folge, daß im L-Register eine 1 stünde und im H-Register eine 8. Zusammen ergibt dies den 16-Bit-Wert: 2049. Nach diesem kleinen Ausflug ins Innere des Z-80-Lebens wenden wir uns nun wieder den Registern zu.

Die INDEX-REGISTER. Die zwei Index-Register, iX und iY genannt, sind 16-Bit-Register. Sie lassen sich (offiziell) nicht in Einzelregister, ähnlich wie HL zu H und L, aufspalten. iX und iY werden im allgemeinen als Adressspeicher und weniger zur Datenspeicherung bzw. -verarbeitung verwendet. Das Wörtchen INDEX deutet eine Besonderheit der Index-Register an: die Möglichkeit der "indizierten" Adressierung. Diese ergibt sich aus der Tatsache, daß die Index-Registerbefehle des Mikroprozessors einen zusätzlichen 8-Bit-Wert verlangen, der zum Index-Registerwert addiert wird. Das iX-Register enthält beispielsweise den Wert einer Adresse, den Beginn einer speicherinternen Datentabelle, die es abzuarbeiten bzw. zu durchforsten gilt. Die Daten sind geordnet, d.h. die Datenstruktur wiederholt sich in unserem Fall alle X-Bytes (Name, Vorname, Wohnort, Name, Vorname, -Wohnort..). Die Index-Registerbefehle erlauben z.B. ein "lade die Daten aus der Daten-Adresse + 2 Bytes" (z.B. Wohnort). Für jeden Datensatz würde dieser Befehl auf den WOHNORT zielen. Ein derartiger Hinweis auf eine bestimmte Adresse wird POINTER genannt. Zwar sind die Anwendungsmöglichkeiten der Index-Register durch einen beschränkten Befehlssatz eingeengt, doch sollten sie deshalb, weil sie nicht so universell sind, nicht vergessen werden. Das sind meist die Zwischenspeicher, die man gerade braucht.

Im Unterschied zum 6502 (das ist auch ein 8-Bit Mikroprozessor) - das sei an dieser Stelle für diejenigen erwähnt, die 6502-Anwendern schon über die Schulter geschaut haben - arbeitet der Z-80 eher registerbezogen. Deshalb sollte man auch jedes Register zwanglos in die Programmiererei mit einbeziehen.

Das SP-REGISTER. Eine sehr nützliche Einrichtung der meisten Mikroprozessoren sind die STACKs. Ein STACK ist einfach ein bestimmter Speicherbereich (bzw. zunächst nur einige Speicherstellen), der als solcher bestimmt und genutzt wird. Im Hinblick auf unser Bemühen, Daten schnell und einfach zwischenspeichern zu können, kommt ein STACK wie gelegen. Zwei Z-80 Befehle sind für den STACK entscheidend: PUSH und POP. Der PUSH-Befehl gestattet es, nahezu alle 16-Bit-Register in zwei definierte Speicherstellen zu kopieren und damit zu "retten". Der POP-Befehl veranlaßt genau das Gegenteil: die Daten werden wieder in das 16-Bit-Register geschrieben. Das SP-Register enthält die Adressen dieser bewußten zwei Speicherstellen, die bei Be-

darf zur Datenspeicherung genutzt werden. Der Programmierer hat die Möglichkeit, den Wert des SP-Registers (SP = STACK POINTER) frei zu bestimmen: jede Adresse zwischen 0 und 65535 kann gewählt werden. Zweckmäßig wäre es jedoch, einen möglichst unbenutzten Speicherbereich als STACK zu wählen, das u.a. deshalb, weil der Stack "wächst".

Der Stack wächst von "oben nach unten", das SP-Register zeigt auf das augenblickliche Ende des Stack-Bereiches. Wir wählen zum Beispiel die 49000 als Start-Adresse des Stack, d.h. SP= 49000. Der erste PUSH-Befehl bewirkt die Abspeicherung zweier Bytes in die Adressen 49000 und 48999. Ein nochmaliges PUSH wirkt dann auf die Adressen 48998 und 48997, d.h. SP wird bei jedem PUSH um 2 vermindert, damit bei nächsten Mal nicht die Daten des vorherigen PUSH-Befehls überschrieben werden. Analog bewirkt ein POP genau daß Gegenteil: SP wird um 2 erhöht, zeigt somit auf die zuletzt abgespeicherten Bytes und übergibt diese an das entsprechende Register. Der Fachausdruck für ein derartiges Daten-Handling": LIFO (Last In First Out)... "wer zuletzt kommt, mahlt zuerst".

Vorsicht ist geboten bei Unterprogrammen. Bei einem CALL-Sprung in ein Unterprogramm benutzt der Z-80 den Stack als Zwischenspeicher für seine augenblickliche Speicheradresse, zu der er bei einem RET zurückkehren soll. Ein POP zuviel, bzw. mehr POPs als PUSHes im Unterprogramm und schon haben Sie sich die Rückkehradresse des Prozessors geschnappt. Beim RET-Befehl steht dann irgendein Wert im Stack und Ihr Programm ist so gut wie weg: "abgestürzt"!

Das PC-REGISTER. Der "PC" (Program Counter) ist ein reines 16-Bit-Adress-Register. Im PC steht die Adresse des Befehls, der als nächster dran ist - da wo der Z-80 sich im Programmablauf gerade befindet. Die PC-Adresse, bzw. der PC-Wert ist auch der, der bei Bedarf auf den STACK gerettet wird. Der Programmierer kann den PC als Register nicht einsetzen.

Das I-REGISTER tritt bei sog. INTERRUPTS in Aktion, auf die wir aber nicht näher eingehen werden.

Das R-REGISTER dient zur Steuerung der REFRESH-Aktivitäten des Mikroprozessors, die bei sog. dynamischen RAM-Bausteinen nötig sind. Für den Anwender bedingt interessant.

Die AUSTAUSCH-REGISTER. Der Name sagt es schon. Die Register A' bis L' stellen einen zweiten unsichtbaren Registersatz dar, der komplett und nur komplett zum Daten-Austausch mit den regulären

Register-Partner A bis L zur Verfügung steht. Neben dem direkten Register/Registertausch gibt es auch die Möglichkeit, Daten mit dem Stack auszutauschen. Hierzu später mehr.

Ein Z-80 ist nicht gern allein. Er kann natürlich nicht nur Daten verarbeiten, sondern auch mit der "Außenwelt" kommunizieren, d.h. Daten im weitesten Sinne senden und empfangen. Dies geschieht über sogenannte I/O-Ports (I/O= INPUT/OUTPUT). Ein Port ist eine Adresse eines Bausteines, die die Verbindung des Z-80 z.B. mit der Tastatur schafft, eine andere mit dem Video-Controller-IC ...etc. Der Mikroprozessor kann 256 solcher Ports, bzw. deren Adressen verwalten, was darauf zurückzuführen ist, daß ein 8-Bit-Wert zur "Port-Adressierung" verwendet wird. Die Ein-/ausgabe der 8-Bit-Informationen geschieht über zwei ASSEMBLER-Befehle: IN und OUT. An dieser Stelle sei der Hinweis gestattet, daß beide Befehle auch im TA-PC-BASIC "implementiert", d.h. eingebaut sind. Wir können deren Wirkungsweise dadurch sehr bequem im BASIC-Level ausloten. (Das Z-80-IN wird in Basic zu INP). Beim TA-PC sind einige Ports "belegt" und können durch den Programmierer genutzt werden. All dies erlaubt eine grundlegendere Kontrolle der Datenein-/ausgabe und sogar die Erstellung eigener I/O-Routinen.

Wenn Sie einen Port abfragen, dann liefert er Ihnen einen 8-Bit Wert. Jedes Bit dieses Bytes hat meist eine bestimmte Funktion bzw. Bedeutung. Andersherum hat jedes Bit eines von "uns" gesendeten Bytes auch eine bestimmte Bedeutung. Wir können damit spezielle Aktionen auslösen, z.B. Beeper an oder LED an... etc. Da jeder Port abgefragt wie auch beschrieben werden kann (auf die Ausnahmen müssen wir hier nicht eingehen), werden jeweils die IN- und dann die OUT-Bedeutungen der Bits 0 bis 7 vorgestellt.

*** PORT 16 ***

IN (16) :

Bit-Nr 0 "0" = Floppylaufwerk ist nicht ansprechbar
 "1" = Floppylaufwerk ist ansprechbereit

Bit-Nr 1 "0" = Graphik-Board (?) ist nicht verfügbar
 "1" = Graphik-Board (?) verfügbar

Bit-Nr 2..3..4 Auswahl des Tastatur - Zeichensatzes nach folgendem Code :

0 0 0 International

0	0	1	Deutsch
0	1	0	USA
0	1	1	Französisch
1	0	0	Englisch
1	0	1	Italienisch
1	1	0	Spanisch
1	1	1	<reserviert>

Bit-Nr 5 "0" = Druckerdaten-Ausgabe über Centronics
 "1" = Druckerdaten-Ausgabe über V24

Bit-Nr 6 "0" = PAL TV-Norm
 "1" = NTSC TV-Norm

Bit-Nr 7 "0" = Der CRTC schreibt gerade auf das TV
 "1" = sog. "Austastlücke"

Zu Bit 7: Diese Information wird benötigt, um das "Bildschirm-Flimmern" zu unterdrücken. CRTC steht für Cathode Ray Tube Controller (Video-Controller). Dieser Baustein ist für die Bilddarstellung auf Ihrem Monitor oder TV Gerät verantwortlich und darf "in bestimmten Phasen" seiner Arbeit nicht gestört werden, sonst kommt es zu dem Phänomen "Bildschirm-Flimmern".

Diese Daten über den PC erhält man z.B. durch den BASIC-Befehl A = INP (16). Je nach Einstellung der DIP-Schalter und anderer Faktoren ändert sich der INP-Wert. Jedes einzelne Bit des abgefragten Wertes hat, wie Sie sehen, seine besondere Funktion.

*** PORT 16 ***

OUT (16):

Bit-Nr 0 "0" = Bildschirmdarst.: 40 x 24 Zeichen/Zeile
 "1" = Bildschirmdarst.: 80 x 24 Zeichen/Zeile

Bit-Nr 1 "0" = Bildschirmdarstellung: EIN
 "1" = Bildschirmdarstellung: AUS

Bit-Nr 2 "0" = Cassetten-Recorder
 "1" = V.24-Schnittstelle

Bit-Nr 3 "0" = Cassetten-Motor: AUS
 "1" = Cassetten-Motor: EIN

Bit-Nr. 4 "0" = Summer (Piepser): AUS
 "1" = Summer (Piepser): EIN

Bit-Nr. 5 ist immer, bzw. muß immer "0" sein!

Bit-Nr. 6 "0" = ROM-Pack gesperrt ("disabled")
 "1" = ROM-Pack freigegeben ("enabled")
 (Bedingung: Bit 7 muß "0" sein)

Bit-Nr. 7 "0" = ROM-Zugriff freigegeben
 "1" = ROM-Zugriff gesperrt, d.h., die gesamten
 64 KBYTE des PC sind RAM.

Der Programmierer kann durch einfache OUT-Befehle die Daten-ein-/ausgabe des PC beliebig steuern. der Port 16, der mit OUT angesprochen wird, ist offensichtlich ein anderer, als der mit IN angesprochene. Um die Zustände der Bits 0 - 7 bei einem OUT (16) zu kennen (schließlich muß ich, auch wenn nur ein Bit verändert werden soll, die anderen mit berücksichtigen), wird der Port-Wert automatisch vom System in die Speicherstelle 58472 geschrieben. Wollten wir beispielsweise den Summer einschalten, wären drei Schritte einzuleiten:

1. Port- bzw. Status-Wert aus Adresse 58472 abfragen und retten
2. Status-Wert + 16 (weil Bit-Nr. 4 zu "1" wird) nehmen
3. Neuen Status-Wert mittels OUT-Befehl ausgeben

In BASIC: OUT (16), PEEK(58472) + 16

In ASSEMBLER:

```
LD  A , ( 58472 ) ; Status-Wert aus Adresse 58472 in A
SET 4 , A         ; Bit-Nr. 4 wird auf "1" gesetzt (+16)
OUT (16) , A      ; Neuen Status-Wert über Port 16 ausgeben
```

*** PORT 32 ***

OUT (32):

- Bit-Nr. 0 "0" = CRTC nicht zurücksetzen
 "1" = CRTC zurücksetzen
- Bit-Nr. 1 "0" = Centronics nicht zurücksetzen
 "1" = Centronics zurücksetzen
- Bit-Nr. 2 "0" = Centronics-Strobe-Signal inaktiv
 "1" = Centronics-Strobe-Signal aktiv
- Bit-Nr. 3 "0" = Monitor-ROM angewählt
 "1" = Video-RAM angewählt
- Bit-Nr. 4 "0" = GRAPH-LED aus
 "1" = GRAPH-LED an
- Bit-Nr. 5 "0" = LOCK-LED aus
 "1" = LOCK-LED an
- Bit-Nr. 6 "0" = Monitor-ROM Freigabe: höhere 4KBYTE
 "1" = Monitor-ROM Freigabe: niedrige 4KBYTE
- Bit-Nr. 7 nicht genutzt!

Die Zustände der Bits 0 - 7 werden vom System in die Speicherstelle 58473 geschrieben. Um z.B. das GRAPH-LED anzuschalten, wäre Adresse 58473 abzufragen, Bit-Nr 4 auf "1" zu setzen und der Gesamtwert über Port 32 auszugeben.

Die Port-Adressen 32 bis 43 werden zur Tastaturabfrage genutzt. Die Abfrage erfolgt über den IN-Befehl und ergibt einen Wert, dessen Bits für jeweils eine Taste der TA-PC Tastatur stehen. Um die Funktions-Tasten (1) bis (6) abzufragen, auch unabhängig von den SHIFT-Tasten, genügt z.B. ein IN (43), dessen Bits von 7 bis 2 die Funktionstasten (6) bis (1) repräsentieren.

Der CRTC. Bisher nur kurz erwähnt, werden wir die Aufgaben des CRTC im PC und den entsprechenden CRTC-Port näher beschreiben. Ein CRTC stellt mehr oder weniger eine Brücke zwischen dem eigentlichen Computer und dem Sichtgerät, sei es nun ein Monitor oder auch ein einfacher Fernseher, dar. Die im VIDEO-RAM in ASCII Form gespeicherten Informationen sind es, die auf dem Bildschirm erscheinen. Portmäßig interessant wird dieser

Baustein besonders dadurch, daß er sehr variabel ("programmierbar") ist.

Zunächst sein Name: 6845SP. Der 6845 kann bis zu max 16K Video-Ram verwalten, d.h. abfragen und die TV-Signale erzeugen. Belegt sind beim TA-PC max., d.h. im 80-Zeichen-Modus, ca 4K: Adressen von 61440 bis 65407. Was stimmt hier nicht? 80 x 24 sind 1920 Zeichen - woher kommen die 4K? Nun, der 6845 geht nicht nur die ASCII-Zeichen durch und setzt diese in Signale um, sondern tut genau das gleiche auch für die Attribute: Informationen über Farbe, Hintergrundfarbe, Blinken...etc. Wir kommen demnach auf 1920 x 2, immerhin "fast" 4K, es bleibt ein bißchen Luft zum rumexperimentieren.

Und das geht so. Wir bedienen uns der "Register" des 6845 und verändern damit z.B. die Anzahl der Zeichen/Zeile, stellen einen anderen "Bildfang" ein, bringen das Bild zum wackeln... etc.

Jedes Register hat eine bestimmte Bedeutung. Ein CRTC-Register stellt praktisch eine Variable dar, die entweder gelesen, beschrieben oder gelesen und beschrieben werden kann. Über Port 80 steht der TA-PC mit seinem CRTC in Kontakt. Die Funktion der einzelnen Register sei hier ansatzweise beschrieben.

- R(0) - Dieses Register ist mit bis zu 255 programmierbar und steht für die Gesamtzahl der möglichen Zeichen in einer Text-Zeile minus 1.
- R(1) - In diesem Register steht die Anzahl der tatsächlich angezeigten Zeichen/Zeile. Hiermit läßt sich die TA-PC Anzeige z.B. in die Breite strecken.
- R(2) - Mit diesem Register läßt sich die Anzeige / das Bild horizontal verschieben. Für Effekte sehr nützlich.
- R(3) - Länge des Zeilensynchronpulses
- R(4) - In diesem Register steht die Gesamtzahl der möglichen Zeilen pro Bild.
- R(5) - Die Programmierung dieses Registers entspricht dem Bildfang beim Fernseher.
- R(6) - Dieses Register bestimmt die Anzahl der auf dem Bildschirm angezeigten Text-Zeilen pro Bild. Man kann dadurch das Bild vertikal vergrößern oder verkleinern.

- R(7) - Siehe auch R(2) für die Horizontale. Mit diesem Register können Sie das Bild nach oben oder unten versetzen.
- R(8) - Interlace und Skew
- R(9) - Dieses Register bestimmt die maximale Linienzahl pro Zeichen minus 1.
- R(10,11)- Die sog. Cursor-Register, mit denen man die Gestalt des Bildschirm-Cursors verändern kann.
- R(12,13)- Mit diesem Register legen wir fest, aus welcher Stelle der CRTC das erste Zeichen ausliest, d.h. wir legen den VIDEO-RAM-Bereich neu fest.
- R(14,15)- Dieses Register bestimmt, an welcher Stelle der Cursor auf dem Bildschirm erscheint.

Die eigentliche Programmierung des 6845 geschieht über ein Register namens AR. Dieses Zuweisungs- bzw. Adress-Register bestimmt nämlich, an welches der 16 Register die Daten geleitet werden. Der PC ist mit seinem CRTC durch zwei Port-Leitungen verbunden:

Port-Nr. 80 für das AR-Register

Port-Nr. 81 für Daten, die hin- oder hergehen.

Um ein CRTC-Register zu lesen oder zu beschreiben, sind demnach immer 2 Schritte notwendig: dem AR-Register mitteilen, welches Register gemeint ist und dann die Daten entweder empfangen oder senden.

Leicht kann es Ihnen nun passieren, daß beim Probieren die Anzeige verloren geht. Damit Sie auch wieder zum normalen Bild zurückfinden, hier nochmal die Ursprungswerte (beim Einschalten) der Register 0 bis 15:

Reg:	0	1	2	3	4	5	6	7	8	9	10	11	12/13	14/15
40 :	70	40	52	86	30	2	24	27	0	9	73	9	0	0
80 :	126	80	95	92	30	4	24	27	0	9	73	9	0	0

Die erste Reihe enthält die Werte für den 40-Zeichen-Modus und die zweite Reihe die Werte für den 80-Zeichen-Modus, die intern

vorgegeben sind.

Ein Beispiel: der TA-PC befindet sich im Width-40-Modus. Um die Bildweite (jetzt noch 40 Zeich./Zeile) zu strecken, beschreiben wir das Register R1 z.B. mit 45 (ergibt 45 Zeichen/Zeile)

OUT 80 , 1 ; das AR-Register wird mit 1 (R1) geladen.

OUT 81 , 45 ; es werden nun 45 Zeichen/Zeile angezeigt

... und schon ist das Bild gestreckt. Wir können es natürlich auch komprimieren (z.B. auf 20 Zeichen/Zeile):

OUT 80, 1 : OUT 81, 20

Die Effekte sind natürlich nur zu sehen, wenn auch etwas auf dem Bildschirm steht.

Wir haben bisher im Schnellverfahren die wesentlichen Züge der Maschinensprache, des Mikroprozessors und dessen Zusammenwirken mit seiner "Außenwelt" besprochen. Bevor wir uns nun den eigentlichen Assembler-Befehlen zuwenden, bietet sich nachfolgend die Möglichkeit, ein Blick in das HEXADEZIMALE-Zahlensystem zu tun. Diese Abhandlung wurde bewußt an das Ende dieses Kapitels gestellt, weil Sie zwar Kenntnisse zum sog. HEX-Code früher oder später ohnehin benötigen werden, diese aber für die noch folgenden Kapitel unerheblich sind. Schon der im TA-PC eingebaute Monitor arbeitet rein im HEX-Betrieb, d.h. nimmt nur HEXADEZIMALE Zahlen an. Es bleibt demnach Ihnen überlassen, zu welchem Zeitpunkt Sie sich damit "auch noch" befassen.

Das HEX-SYSTEM. Maschinensprache-Programmierer ziehen normalerweise das sog. HEXADEZIMALE-Zahlensystem dem BINÄREN (zur Basis 2) und unserem "normalen" DEZIMALEN (zur Basis 10) vor. Das HEX System hat die 16 zur Basis, was bedeutet, daß die Ziffern dieses Zahlen-Systems die Zahlen von 0 bis 15 darstellen können. Der wesentliche Vorteil dieser Zählweise liegt darin, daß die Zahlen von 0 bis 15 mit genau 4 Bits ausgedrückt werden können, so daß jede 8-Bit-Zahl (d.h. jedes BYTE) mit nur zwei HEX-Ziffern dargestellt werden kann. (anstatt z.B. 245 "dezimal" = 3 Ziffern). Das Problem liegt darin, daß die Zahlen von 10 bis 15 nicht als solche auch im Hex-System dargestellt werden können, als wären sie zur Basis 10. Deshalb werden sie durch die Buchstaben A bis F repräsentiert. Das bedeutet, daß C in Hex einer 12 entspricht und F dementsprechend einer 15. Damit können wir jede BYTE-große Zahl in HEX als zwischen 00 (0) und FF (255) darstellen.

BINÄR	DEZIMAL	HEX
0000	0	00
0001	1	01
0010	2	02
0011	3	03
0100	4	04
0101	5	05
0110	6	06
0111	7	07
1000	8	08
1001	9	09
1010	10	0A
1011	11	0B
1100	12	0C
1101	13	0D
1110	14	0E
1111	15	0F

Wie man sieht, tritt der Überlauf nicht wie im Dezimalen bei jeder 10er, sondern bei jeder 16er Potenz (16, 256, 4096 ...) auf. Das bedeutet: zwei Ziffern für 256 verschiedene Zahlen, d.h. von 0 bis 255, nur vier Ziffern für die Zahlen von 0 bis 65535, z.B. Adressen...etc. Nehmen wir als Beispiel die Adresse 01FA Hex:

Zur Umrechnung in das Dezimale System bedienen wir uns des folgenden Schlüssels:

4 1 E A (HEX)

1. Ziffer "A" wird immer mal 1 genommen = 1 * 10 = 10
2. Ziffer "E" wird immer mal 16 genommen = 16 * 14 = 224
3. Ziffer "1" wird immer mal 256 genommen = 256 * 1 = 256
4. Ziffer "4" wird immer mal 4096 genommen = 4096 * 4 = 16384

Das ergibt $10 + 224 + 256 + 16384 = 16874$ Dezimal. Die Ziffern EA entsprechen dem LSB der Adressen und die 41 dem MSB. Am besten wir üben noch ein bißchen:

$$\begin{aligned}
 0C &= 0 * 16 + 12 * 1 &= 12 \text{ Dez.} \\
 DA &= 13 * 16 + 10 * 1 &= 218 \text{ Dez.} \\
 A3 \text{ } 0F &= 10 * 4096 + 3 * 256 + 0 * 16 + 15 * 1 = 41743 \text{ Dez.} \\
 FF \text{ } FF &= 15 * 4096 + 15 * 256 + 15 * 16 + 15 * 1 = 65535 \text{ Dez.}
 \end{aligned}$$

Nun ist es aber Zeit, das Kapitel abzuschließen und uns eingehender mit den Assembler-Befehlen zu befassen.

2. Z-80-BEFEHLE

Der Z-80 kennt mehr als 600 Befehle. Dieser sehr umfangreiche Befehlssatz hat seinen Ursprung in der Konzeption bzw. Philosophie des Z-80-Mikroprozessors. Im Gegensatz zum 6502, erinnern Sie sich noch, arbeitet er eher "registerbezogen" und da er nun einmal so viele davon hat, und jedes Register auch möglichst vielseitig verwendbar sein sollte, entstand dieser "Mammut"-Befehlssatz. Allerdings, nicht alle Z80-Befehle werden gebraucht. Einige Befehle sind so gesehen sehr speziell und äußerst rar in der Anwendung.

Wollten wir alle Z80-Befehle in einem Byte codieren, dann wären nur 256 verschiedene Befehle möglich. Um aber den gesamten vorhandenen Befehlssatz zu erfassen, sind einige Befehle sog. 2- bzw. sogar 3-Byte Befehle. Das erste Byte dient dann praktisch als Hinweis darauf, daß es sich um eine andere Befehlsart handelt.

Auf einige Z-80-Befehle folgt ein 8- bzw. 16-Bit-Wert, so daß ein Assembler-Befehl bis zu 4 Bytes lang werden kann. Möglich sind:

1. Ein 8-Bit Wert als solcher (Daten)
2. Ein 8-Bit Wert als Sprung-Distanz (in Bytes)
3. Ein 16-Bit Wert als solcher (Daten)
4. Ein 16-Bit Wert als ADRESS-Wert

Beispielsweise benötigt die schon erwähnte Anweisung JR NZ,+/-X ein Byte für den Sprung-Befehl selbst und ein weiteres Byte für den Wert der Sprung-Distanz + bzw. - X. Der Befehl LD A,100 belegt ebenfalls zwei Bytes: für den Befehl und für den Wert 100. Die Schreibweise der Assembler-Befehle folgt einigen Konventionen:

- a. Einzelne Register werden durch ihren Buchstaben benannt.

Beispiel: A für Akkumulator, D für Register D, IX, HL...etc

- b. Das Ziel eines Ergebnisses wird immer zuerst angegeben.

Beispiel: ADD A , C

"addiere den Wert aus C zum Akkumulator und schreibe das Ergebnis in den Akkumulator (Ziel)"

- c. Ein Wert in Klammern deutet auf seine "indirekte" Verwendung hin.

Beispiel: LD HL, (33000)

"lade HL aus den Speicherstellen 33000 und 33001"

...aber: LD HL, 33000

"lade HL mit dem Wert 33000"

Steht HL in Klammern (HL), ist die durch HL adressierte Speicherstelle gemeint, HL ohne Klammern bezieht sich auf den in HL gespeicherten Wert.

- d. Eine Sprungdistanz wird zwischen 0 und 127 als positiv angesehen, zwischen 128 und 255 als negativ. Der Sprung wird bei derartigen Befehlen erst hinter dem Sprung-Befehl durchgeführt, es müssen demnach immer 2 Bytes subtrahiert werden.

Wir werden nachfolgend die wichtigsten Z-80-Befehle vorstellen. Von besonderer Bedeutung ist immer die Wirkung des Befehls auf die Flags, insbesondere ZERO- und CARRY-Flag. Eine detaillierte Befehlsbeschreibung würde allerdings den Rahmen sprengen. Deshalb beschränken wir uns auf eine zum besseren Verständnis der in Zukunft verwendeten Assemblerbefehle notwendige Darstellung.

ADC

Sprich "add with carry". Mit diesem Befehl kann man eine Addition durchführen lassen, die - im Unterschied zum ADD-Befehl - auch das Carry-Flag mit einbezieht. Die Addition erfolgt entweder zum Akkumulator oder zu HL.

ADD

Sprich "add". Ähnlich dem ADC, nur daß auf die zusätzliche Addition des Carrys verzichtet wird. Addition zu A, HL, IX oder IY.

AND

Sprich "and" bzw. "und". Dieser Befehl gehört zur Gruppe der Logik-Befehle, so wie auch OR und XOR. Die "logische" Verknüpfung erfolgt zum Akkumulator.

BIT

Sprich "bit Nr. von". Ein spezielles Bit wird geprüft und das ZERO-Flag entsprechend gesetzt.

CALL

Sprich "call ADRESSE". Entspricht dem GOSUB und ruft ein Unterprogramm ab der Speicherstelle ADRESSE auf. Rückkehr mit RET.

CP

Sprich "compare". In Verbindung mit dem Akkumulator wird ein Wert geprüft und die Flags entsprechend gesetzt.

DEC

Sprich "decrement". Dekrementieren bedeutet um 1 vermindern bzw. "minus 1 nehmen". Der DEC-Befehl kann sich auf Register, Speicherstellen und Registerpaare beziehen.

DJNZ

Sprich "decrement and jump if not zero". Ein sehr spezieller Befehl, der sich ausschließlich auf das Register B bezieht und gleich drei Dinge auf einmal bewerkstelligt: das Register B wird dekrementiert, dessen Wert geprüft ($= 0$?) und falls dieser "nicht null" ist wird gesprungen, z.B. zurück in eine Schleife. Ideal zur Verwendung des Registers B als Zähler.

EX und EXX

Sprich "exchange". Mit diesen Befehlen wird ein Wertetausch zwischen Registern und Zweit-Registern angeregt.

IN

Sprich "in (put)". Dieser Befehl dient zur ABFRAGE der PORTs des Mikroprozessors.

INC

Sprich "increment". Das Gegenstück zum DEC-Befehl: das angesprochene Register, Registerpaar bzw. Speicherstelle wird um 1 erhöht, "plus 1 genommen".

JP

Sprich "jump". Es wird ein Sprung zu einer bestimmten Adresse angeregt. Dies kann auch in Abhängigkeit bestimmter Flag-Zustände erfolgen.

JR

Sprich "jump relative". Es wird "relativ" zur augenblicklichen Position um X Bytes vor- oder zurückgesprungen. Dies kann auch in Abhängigkeit bestimmter Flag-Zustände erfolgen.

LD

Sprich "load". Die LD-Befehlsgruppe ist die mächtigste und umfangreichste aller Z-80-Befehlsgruppen. Alle Datenbewegungen

erfolgen mittels solcher load-Befehle. Geladen wird aus Registern, in Register, aus Speicherstellen, in Speicherstellen, etc.

NOP

Sprich "no operation". Dieser Befehl tut nichts. Der Code ist 0.

OR

Sprich "or" bzw. "oder". Einer der logischen Befehle. Der Akkumulator wird "logisch OR" mit einem anderen Register bzw. Wert verknüpft.

OUT

Sprich "out (put)". Dieser Befehl dient zur AUSGABE von Daten über einen der Z-80-PORTs. OUT ist das Gegenstück zu IN.

POP

Sprich "pop". Mit diesem Befehl wird der letzte 16-Bit-Wert, der auf dem STACK abgelegt wurde, wieder zurück in ein 16-Bit-Register geschrieben.

PUSH

Sprich "push". PUSH ist das Gegenstück zu POP. Der PUSH-Befehl bewirkt, daß der Wert eines 16-Bit-Registers auf dem STACK abgelegt wird.

RES

Sprich "reset bit". Ein bestimmtes Bit kann durch RES zurückgesetzt werden.

RET

Sprich "return". Wie in BASIC bewirkt dieser Befehl den Rücksprung aus einem Unterprogramm. Dies kann auch in Abhängigkeit bestimmter Flag-Zustände erfolgen.

RST

Sprich "restart". Die RST-Befehle belegen genau 1 Byte und bewirken einen direkten Sprung zu einer von 8 festgelegten Adressen.

SBC

Sprich "subtract on carry". Mit diesem Befehl können Register/Speicherstellen (indirekt) voneinander subtrahiert werden. Das Carry-Flag wird in die Subtraktion miteinbezogen.

SET

Sprich "set bit". Mit diesem Befehl kann ein bestimmtes Bit ge-

setzt werden.

SUB

Sprich "subtract". Ein Befehl, der sich ausschließlich auf den Akkumulator bezieht. Ein 8-Bit-Wert (Register, Daten...) wird vom Akkumulator subtrahiert.

XOR

Sprich "xor". Einer der drei logischen Befehle. Der Akkumulator und ein 8-Bit-Wert werden "logisch xor" verknüpft.

Die LOGIK. Die Angaben zu den Logik-Befehlen kann man nicht so ohne weiteres stehen lassen, deshalb an dieser Stelle noch eine kurze Ergänzung. Zu jedem Logik-Befehl gibt es eine sog. "Wahrheitstabelle". Diese Tabelle enthält die Regeln zur Behandlung der aufeinandertreffenden Bits, z.B. zweier Register. Der Akkumulator enthält immer das Ergebnis der Operation.

AND	OR	XOR
0 AND 0 = 0	0 OR 0 = 0	0 XOR 0 = 0
0 AND 1 = 0	0 OR 1 = 1	0 XOR 1 = 1
1 AND 0 = 0	1 OR 0 = 1	1 XOR 0 = 1
1 AND 1 = 1	1 OR 1 = 1	1 XOR 1 = 0

Bei der Verknüpfung AND 200 z.B. wird jedes einzelne Bit des Akkumulators mit jedem einzelnen Bit des Wertes 200 nach o.a. AND-Tabelle verknüpft und das Ergebnis im Akkumulator abgelegt.

Nicht besprochen wurden bisher sog. Blocklade-Befehle, Rotations-Befehle und einige ganz spezielle Einzelbefehle, z.B. zur Interrupt-Steuerung.

Für eine tiefergehendere Darstellung des Z-80-Prozessors sollte man auf einschlägige Literatur zurückgreifen. Die "Bibel" zum Z-80 nennt sich "Programmierung des Z-80" (von Rodnay Zaks) und erscheint im Sybex-Verlag.

Unser nächstes Problem wird sein, unsere Maschinensprache-Routinen in entsprechende BASIC-Programme zu integrieren.

3. Z-80 PRAXIS

Obwohl es mehrere Methoden gibt, Maschinensprache-Routinen in BASIC-Programme zu integrieren, wollen wir uns auf eine konzentrieren und eine weitere am Rande erwähnen. Ausgangsbasis ist eine kleine Beispiel-Routine: PLUS. Die Routine PLUS tut nichts anderes, als das VIDEO-RAM durchzugehen und den Wert jedes Byte um 1 zu erhöhen. Dadurch können wir uns am Bildschirm von der korrekten Funktion unserer kleinen Routine vergewissern.

In BASIC könnte man z.B. einer FOR-NEXT-Schleife diesen Auftrag geben (WIDTH 40 vorausgesetzt):

```
10 FOR VIDEO = 61440 TO 62463
20 POKE VIDEO, PEEK ( VIDEO ) + 1
30 NEXT VIDEO
```

Die Umsetzung in Maschinensprache scheint nicht allzu schwer. Tatsächlich brauchen wir zunächst nur eine Schleife, die allerdings groß genug sein muß (1024 Zeichen zu durchlaufen), d.h. von einem 16-Bit-Register gesteuert wird (8-Bit = max. 256). Nehmen wir mal das Registerpaar BC. BC "läuft" demnach von 1024 auf 0 herunter. Ein zweites 16-Bit-Register benötigen wir als "Pointer", als Zeiger auf die zu "inkrementierende" Speicherstelle. Hierzu bestimmen wir HL. In Abhängigkeit des Zählers BC stehen in HL nacheinander die Adressen 61440 bis 62463 des Video-RAM-Bereiches.

Damit das BC-Registerpaar auch wirklich bei 0 aufhört zu zählen, muß eine Kontrolle eingebaut werden. Leider hat der DEC-Befehl bei 16-Bit-Registern (hier:DEC BC) keinen Einfluß auf die Flags, so daß eine einfache ZERO-FLAG-Kontrolle nicht möglich ist.

Wir müssen also die zwei 8-Bit-Register B und C von BC einzeln prüfen und erst wenn beide 0 sind, die Prozedur abbrechen. Dafür gibt es eine recht elegante Methode, die sich die Eigenschaften des OR-Befehls zunutze macht, denn: (laut OR-Tabelle) wenn auch nur ein Bit auf "1" ist, kann das Ergebnis schon nicht mehr 0 sein, d.h., nur wenn alle Bits von B und alle Bits von C "0" sind, dann wäre auch das Ergebnis von B OR C oder C OR B gleich 0. Und auf den OR-Befehl reagieren die Flags, die wir dann ja prüfen können.

Zum Abschluß der ganzen Routine erscheint, damit der TA-PC auch wieder ins BASIC zurückfindet, ein RET. Wir wollen nun in einem ersten Schritt die Assembler-Befehlsfolge von PLUS fest-

legen und anschließend die PLUS-Routine codieren.

Die Routine PLUS :

```
LD    HL , 61440      ; HL zeigt auf 1.Byte des VIDEO-RAM
LD    BC , 1024       ; BC, der Zähler, wird initialisiert
INC   (HL)            ; Hier fängt die Schleife an, das durch
INC   HL              ; (HL) adressierte VIDEO-Byte wird: + 1,
DEC   BC              ; HL + 1 und BC - 1 genommen. Jetzt wird
LD    A , B           ; BC auf 0 geprüft. Da es OR B,C nicht
OR    C               ; gibt, wird Akku mit B geladen. Falls
JR    NZ , - 7        ; NOT ZERO wird um 7 Bytes zurückgeJUMPt,
RET                                ; zu INC (HL). Sonst zurück zum BASIC.
```

Die Codierung solch kleiner Routinen kann von Hand erfolgen. In diesem Fall greifen wir, mangels einer Z-80-Code-Tabelle, etwas vor. Der Befehl LD HL, ? hat den Code 33 und die Adresse 61440 wird in die zwei Bytes 0 (LSB) und 240 (MSB) codiert. ($61440 = 0 + 240 * 256!$). Der Befehl LD BC, ? hat den Code 1 und die Adresse 1024 die Bytes 0 und 4. Unsere ersten zwei Assembler-Zeilen werden demnach als:

33, 0, 240, 1, 0, 4 codiert.

Die gesamte Routine PLUS sieht codiert so aus:

33, 0, 240, 1, 0, 4, 52, 35, 11, 120, 177, 32, 249, 201

Jetzt kommt es darauf an, das Ganze in ein BASIC-Programm zu integrieren und zwar mit

DATAs

Die Codes werden als DATAs in einer BASIC-Zeile gespeichert und anschließend durch eine Programm - Schleife in den auserwählten Speicherbereich gePOKEt. Dieser "Integrations-Methode" sind hinsichtlich der Länge der Z-80 Routinen kaum Grenzen gesetzt, bis auf den begrenzten BASIC-Speicherbereich natürlich. Bei einer großen Anzahl dieser Datas dauert es außerdem ein Weilchen bis alle in ihren Bytes gespeichert sind. Unsere PLUS-Routine könnte man auf folgende Art und Weise aktivieren:

```
10 DATA 33,0,240,1,0,4,52,35,11,120,177,32,249,201
20 FOR SPEICHER = 50000 TO 50013
30 READ WERT : POKE SPEICHER , WERT
40 NEXT SPEICHER
50 PLUS = 50000
```



```
60 CALL PLUS
70 GOTO 60
```

In Zeile 10 steht die PLUS-Routine codiert als DATAs. In den Zeilen 20 bis 40 wird die Routine, werden die Z-80-Codes aus der Data-Zeile ausgelesen und in die Speicherstellen 50000 bis 50013 geschrieben. Dort kann PLUS jederzeit aufgerufen und ausgeführt werden. Dies geschieht in den Zeilen 50 bis 70. Eine Variable, sinnigerweise PLUS genannt, wird mit dem Wert 50000 geladen, damit die Routine über einen CALL-Befehl ausgerufen werden kann (Zeile 60). In Zeile 70 wird dann immer wieder zu Zeile 60 zurückgesprungen. Der Bildschirm verändert sich demnach dauernd und ohne Ende. Mit BREAK läßt sich der Spuk stoppen.

Auf diese Art und Weise werden wir auch die Routinen zu unserem noch folgenden Programm "Die Briefmaschine" speichern und aktivieren. Als Anregung sollte folgender kleiner Hinweis dienen:

REM-Zeilen werden in BASIC beim Programm-Durchlauf übergangen. Das prädestiniert sie geradezu zur Speicherung von z.B. Z-80 - Routinen. Anstatt jede einzelne Ziffer eines Codes, wie in den DATA-Zeilen, zu speichern, wird das Code-Byte direkt in die REM-Zeile geschrieben und aus dieser auch wieder herausgelesen. Bei einem LIST erscheinen dann zwar lauter "wilde" Zeichen in der REM-Zeile, doch hat dies keinen negativen Effekt auf das Programm oder dessen Ausführungsgeschwindigkeit.

Einen Haken hat die ganze Sache allerdings doch. In die REM-Zeile darf man keine Bytes zwischen 0 und 31 schreiben. Die Zahlen 0 bis 31 stehen für Sonder-Sequenzen des eingebauten BASIC-Interpreters, was leicht zu Unannehmlichkeiten führen kann. Da man andererseits nicht auf die Z-80-Befehle mit den Codes 0 bis 31 verzichten kann, muß man sich einen eigenen ESCAPE-Code ausdenken, d.h. eine Zahl, deren Auftreten signalisiert: "der folgende Code ist im Grunde ein 0 bis 31-Code".

Der Code 1 für LD BC, ? z.B. würde zunächst + 32 genommen. Dem 33er Code wird nun unser ESCAPE-Code vorangestellt und das ganze in der REM-Zeile abgespeichert. Beim darauffolgenden "Wiederauslesen" wird der ESCAPE-Code erkannt, übergangen und die 33 wieder - 32 genommen, so daß der Original-Code 1 letztendlich in den Speicher gePOKEt wird. Weiterexperimentieren empfohlen!

4. DIE BRIEFMASCHINE

EINLEITUNG

Die BRIEFMASCHINE ist teils in Maschinensprache, teils in BASIC geschrieben und dient zur direkten Ansteuerung des TRD 7020-Typenraddruckers von Triumph-Adler. Nachfolgend werden wir die einzelnen Programmabschnitte der Briefmaschine kennenlernen und praktisch "im Lesen programmieren". In einem ersten Schritt wird das BASIC-Programm der Briefmaschine als Ganzes und anschließend die vier Z-80-Routinen im einzelnen vorgestellt. Wo immer nötig, sind Anmerkungen für PC-Besitzer mit neuen ROMs hinzugefügt. Entsprechende Modifikationen am Programm sind daher leicht durchzuführen. Die Anpassung der Briefmaschine an andere Drucker dürfte ebenfalls keine Probleme bieten, da sämtliche Druckeransteuerungs-Routinen im BASIC-Teil untergebracht und kommentiert sind. Es müssen lediglich die auszugebenden Steuer-Codes angepaßt werden. Beim Programm-Start sollte der angeschlossene Drucker schon im ONLINE-Modus sein, sonst erfolgt ein Hinweis.

Das Programm BRIEFMASCHINE, (C) Copyright 1985 P.V.Thomas, kann jeder Leser frei verwenden. Bitte, übernehmen Sie dabei jedoch den Urhebervermerk.

DAS PROGRAMM

Die ersten 36 Programm-Zeilen enthalten die DATAs der Z-80-Routinen und einen Urhebervermerk.

```

1 REM * BRIEFMASCHINE (C) Copyright 1985 Patrick V. Thomas *
1000 ' *****
1010 ' *
1020 ' *          DIE BRIEFMASCHINE          *
1030 ' *
1040 ' *****
1050 '
1060 ' ### Maschinen-Code Routinen als DATA's ###
1070 '
1080 DATA 33,0,240,17,1,240,1,0,16,54,0,237,176,201
1090 DATA 94,35,86,235,17,0,248,25,6,0,205,88,234,203
1100 DATA 254,35,16,248,201
1110 DATA 33,238,244,6,80,205,88,234,54,32,35,16,248,201
1120 DATA 120,50,244,194,33,24,235,34,170,231,201,33,0,0
1130 DATA 34,170,231,34,244,194,58,236,194,79,6,1,219,41
1140 DATA 203,119,32,224,4,203,65,32,21,203,95,32,215,4
1150 DATA 203,71,32,210,4,203,87,32,205,4,203,79,32,200

```

```

1160 DATA 24,29,6,2,219,33,203,71,194,127,195,219,32,4
1170 DATA 203,87,194,127,195,4,203,103,194,127,195,4,203
1180 DATA 119,194,127,195,4,219,42,203,103,194,127,195,4
1190 DATA 219,43,203,127,194,127,195,4,203,119,194,127
1200 DATA 195,4,203,111,194,127,195,4,203,103,194,127
1210 DATA 195,4,203,95,194,127,195,4,203,87,194,127,195
1220 DATA 4,219,41,203,127,194,127,195,4,219,42,203,95
1230 DATA 194,127,195,4,219,32,203,71,40,7,219,42,203,87
1240 DATA 194,127,195,203,73,40,15,4,219,33,203,119,194
1250 DATA 127,195,219,42,203,79,194,140,196,205,148,231
1260 DATA 183,202,151,195,254,127,210,151,195,254,32,218
1270 DATA 151,195,203,65,40,11,254,57,48,7,254,50,56,3
1280 DATA 195,151,195,203,73,40,5,254,61,202,151,195,42
1290 DATA 237,194,17,62,245,71,205,166,196,202,151,195
1300 DATA 120,205,87,234,112,35,54,46,34,237,194,203,73
1310 DATA 194,151,195,219,16,203,111,14,64,32,2,14,48
1320 DATA 237,65,6,0,195,127,195,42,237,194,17,239,244
1330 DATA 205,166,196,218,151,195,205,87,234,54,32,43
1340 DATA 54,46,34,237,194,195,172,196
1350 DATA 124,146,192,125,147,201,33,0,99,205,161,230
1360 DATA 195,151,195

```

Mittels einer FOR - NEXT - Schleife werden die Routinen BLANKO, DIVERSE, CLEAZE und TASTATUR in die Speicherstellen 50000 bis 50356 geschrieben. Die entsprechenden Variablen werden mit den jeweiligen START-ADRESSEN initialisiert:

```

BLANKO   = 50000 ... zum Bildschirm-Löschen
DIVERSE  = 50014 ... zum Text hervorheben
CLEAZE   = 50033 ... Eingabezeile löschen
TASTATUR = 50058 ... zur Tastaturabfrage

```

```

1370 `
1380 ` ### Mit FOR-NEXT Schleife einlesen ###
1390 `
1400 FOR SPEICHER= 50000 TO 50000+356
1410 READ DATEN: POKE SPEICHER, DATEN
1420 NEXT SPEICHER: BLANKO= 50000: DIVERSE= 50014
1430 CLEAZE= 50033: TASTATUR= 50058

```

In den folgenden Zeilen wird ein Trick angewandt um zu prüfen, ob ein betriebsfertiger Drucker ONLINE bereitsteht. Zunächst erscheint folgende Meldung:

" SCHALTEN SIE DEN ANGESCHLOSSENEN DRUCKER AUF O N L I N E..."

Dann werden zwei LPRINT-Anweisungen ausgegeben, die bei nicht betriebsbereitem Drucker bewirken, daß der PC solange wartet, bis er die Meldungen absetzen kann, d.h. der Drucker betriebsbereit ist. Ist der Drucker andererseits schon betriebsbereit, dann wird die Meldung durch WIDTH 80 sofort gelöscht.

```
1440 `
1450 ` ### Meldung, falls Drucker nicht bereit ###
1460 `
1470 WIDTH 80: COLOR 7,0,2
1480 PRINT"      SCHALTEN SIE DEN ANGESCHLOSSENEN DRUCKER ";
1490 PRINT"AUF  O N L I N E  ...."
1500 LPRINT CHR$(27)CHR$(13)CHR$(80): LPRINT CHR$(0);
1510 COLOR 7,0,0: WIDTH 80
```

Nun werden die wesentlichen Programm-Parameter festgelegt, das Bildschirm-Format und der Cursor geändert. Eine wichtige Variable, FLAG genannt, tritt in Erscheinung. Sie ist identisch mit dem Inhalt der Speicherstelle 49900, die als STATUS-Byte dient. Diese Speicherstelle baut sich folgendermaßen auf:

Bit-Nr 0 "0" = Druckkopf-Steuerung über PFEILTASTEN
 "1" = Druckkopf-Steuerung über 2,4,6 und 8

Bit-Nr 1 "0" = DIREKT-Modus
 "1" = ZEILEN-Modus

Bit-Nr 2 "0" = GROB-Einstellung
 "1" = FEIN-Einstellung

Bit-Nr 3 bis 7 keine Bedeutung, d.h. immer "0".

Wir haben uns praktisch unser eigenes FLAG-BYTE geschaffen, das einerseits von BASIC aus als Variable FLAG ansteuerbar ist und andererseits in Maschinensprache als Speicherstelle 49900. Weiterhin als Variablen werden die SPALTEN-, ZEILEN-Zähler, LRAND für den linken Tabulator, SCHRIFT für die eingestellte Schrift-Art, TEILORT und ABSTAND für die eingestellte Schrift-Teilung deklariert. Die LPRINT-Anweisungen in den Zeilen 1620,1630 bewirken die Löschung der AUTO-CR und AUTO-LF Funktionen des TRD 7020, damit nicht nach jeder LPRINT-Anweisung automatisch ein Wagenrücklauf und Zeilenvorschub erfolgt, bzw. stellen 1/12inch als Zeichenteilung ein. Die Speicherstellen 49901 und 49902 enthalten die Adresse des Cursors in der Eingabezeile, so daß auch die Maschinensprache-Routinen Text direkt anzeigen können.

```

1520 `
1530 ` ### Programm - Initialisierung ###
1540 `
1550 OUT 80,11:OUT 81,8:POKE &HE462,84
1560 CALL BLANKO: FLAG= 0: POKE 49900, FLAG
1570 POKE 49902, 244: POKE 49901, 238
1580 OUT 80,1: OUT 81,84: OUT 80,2: OUT 81,102: OUT 80,6
1590 OUT 81,24: OUT 80,10: OUT 81,32
1600 ZEILE1= 0: SPALTE1= 0: SPALTE2= 0
1610 SCHRIFT= 86: LRAND= 0: TEILORT= 326: ABSTAND= 11
1620 LPRINT CHR$(27)CHR$(98)CHR$(27)CHR$(100);
1630 LPRINT CHR$(27)CHR$(31)CHR$(11);

```

Erst jetzt wird die Arbeits-Maske der Briefmaschine gePRINTet. Die mit W beginnenden Variablen werden als INTEGER-Typ deklariert, um die Übergabe von Variablen-Werten an Z-80-Routinen so einfach wie möglich zu gestalten (siehe Z-80 Routinen).

Die Graphik-Zeichen erhalten Sie mittels der GRAPH-Taste des PC (siehe Bediener-Handbuch). Sie sind zwar auf dem Bildschirm gemäß dem abgedruckten Listing darstellbar, da mit einem Typenraddrucker wie dem TRD 7020 jedoch keine Graphik-Zeichen druckbar sind, mußte das Listing "getürkt" werden, d.h., die Zeichen wurden von Hand eingetragen.

Die Variable TEXT\$ wird mit 80 Leerzeichen gefüllt. Näheres dazu im Zeilendruck-Unterprogramm.

```

1640 `
1650 ` ### Bildschirm-Maske Printen ###
1660 `
1670 DEFINT W: TEXT$= SPACES$ (80)
1680 PRINT"
1690 PRINT"| NORMAL-SCHRIFT  DOPPEL-SCHRIFT || Modus .... (F5)
    || Ausdruck  ||SchriftTeilung|";
1700 PRINT"|
    ||
1710 PRINT"| UNTERSTREICHEN  SCHATTENSCHRFT || DIREKT-AUSDRUCK
    || SERIELLE  || 10 12 15 |";
1720 PRINT"|
    ||
1730 PRINT"| TRD 7020 Schriften ..... (F2) || ZEILEN-SPEICHER
    || PARALLEL  || (F4) |";
1740 PRINT"

```

```

1750 PRINT"          ||          ||
      ||
1760 PRINT"          ||          || ";
      ||          ||          ||
1770 PRINT"  <-->  || Zeilen  || Spalte  || LRand  ||
Reset  ||  || GROB  ||  || ";
1780 PRINT"          ||          ||          ||
      ||          ||          ||
1790 PRINT"  2468  ||  ||  ||  ||  ||
mit    ||  || FEIN  ||  ||  ||
1800 PRINT"          ||          ||          ||
      ||          ||          ||
1810 PRINT"  (CTRL) ||  || (F1)          || (TAB)
(ESC)   ||  || (F6)   ||  || ";
1820 PRINT"          ||
      ||          ||
1830 PRINT"          ||
      ||          ||
1840 PRINT"          ||
      ||          ||
1850 PRINT"Zeile : LÖSCHEN = (F3)      DRUCKEN = (=)      BACKSPA
CE = (DEL)      ENDE = SHIFT+(O) ";

```

Um wichtige Text-Abschnitte INVERSE darzustellen, wird nun eine DATA-Zeile mit den Angaben LÄNGE DES TEXTES und ORT DES TEXTES für die insgesamt 23 hervorgehobenen Stellen generiert. Eine kleine Routine liest jeweils 2 DATAs (Länge, Ort) aus der Zeile und ruft die Z-80-Routine DIVERSE (von DE-INVERSE) auf, die den Text hervorhebt. Danach wird anhand von Port 16 getestet, ob die PARALLELE oder SERIELLE Schnittstelle zur Daten-Ausgabe genutzt wird und der entsprechende Hinweis "INVERTIERT".

```

1860 `
1870 ` ### Daten für Ort und Länge der Textstellen ###
1880 `
1890 DATA 14,86,4,449,15,289,4,132,8,309,4,326,4
1900 DATA 494,4,760,6,772,6,784,5,798
1910 DATA 5,809,4,824,6,1095,4,1109,5,1134,5,1145,4
1920 DATA 1160,80,1262,7,1437,7,1456,9,1474,4,1496
1930 `
1940 ` ### Stellen mittels Z-80-Routine hervorheben ###
1950 `
1960 FOR T= 1 TO 23: READ ANZAHL, WERT
1970 GOSUB 3160: NEXT T
1980 A= INP(16): IF (A AND 32)= 32 THEN 2030
1990 ANZAHL= 8: WERT= 309: GOSUB 3210: WERT= 477: GOSUB 3160

```

Der nächste Programm-Abschnitt ist der Kern der Briefmaschine, der Hauptteil. Hier werden die Anzeigen für LRand, Spalte und Zeile aktualisiert, die TASTATUR-Routine wird aufgerufen und je nachdem auch zu den Funktions-Unterprogrammen gesprungen.

```

2000 '
2010 '   ###  Haupt - Teil  ###
2020 '
2030 LOCATE 42,11: PRINT LRAND" ";
2040 LOCATE 17,11: PRINT ZEILE1" ";
2050 LOCATE 27,11: PRINT SPALTE1" ";
2060 LOCATE 31,11: PRINT"." SPALTE2;
2070 CALL TASTATUR
2080 IF PEEK(49908)= 0 THEN SPALTE1= SPALTE1+1: GOTO 2050
2090 ON PEEK(49908) GOSUB 2110,2180,2250,2320,2370,2420,2490,
      2540,2610,2670,2770,2830,2890,2980,3030,3070
2100 GOTO 2030

```

Jetzt folgt die Zeit der Unterprogramme. In Zeile 2090 werden die nun folgenden Sub-Routinen angesprungen, nach einem Code, der von der TASTATUR-Routine in Speicherstelle 49908 geschrieben wird. Dieser geht von 0 bis 16 und umfaßt folgende Funktionen:

```

0 ..... kein Unterprogramm, sondern schon erfolgte Texteingabe

1 ..... CURSOR - WAHL , Pfeiltasten oder 2,4,6,8
2 ..... DRUCKKOPF hoch
3 ..... DRUCKKOPF runter
4 ..... DRUCKKOPF links
5 ..... DRUCKKOPF rechts
6 ..... WAGENRÜCKLAUF und ZEILENVORSCHUB
7 ..... ZEILENANZEIGE AUF NULL STELLEN
8 ..... SCHRIFTTYP WÄHLEN
9 ..... EINGABEZEILE LÖSCHEN
10 ..... SCHRIFTTEILUNG WÄHLEN
11 ..... SCHREIBMODUS WÄHLEN
12 ..... GROB/FEIN BEWEGUNG WÄHLEN
13 ..... DRUCKER-RESET AUSLÖSEN
14 ..... LINKEN RAND SETZEN
15 ..... PROGRAMMENDE
16 ..... EINGABEZEILE DRUCKEN

```

Als erstes Unterprogramm begegnen wir der CURSOR-WAHL. Um den Druckkopf zu bewegen, kann man zwischen zwei möglichen Cursor-Tasten-Sets wählen: den regulären Pfeiltasten, links und rechts der Space-Taste und den Tasten 2,4,6 und 8 des 10er Blocks. Die FLAG-Variable und die Cursor-Anzeige der Bildschirmmaske werden entsprechend verändert.

```
2110 `
2120 ` ### Cursor-Wahl durch CTRL  ###
2130 `
2140 ANZAHL= 4: WERT= 760: A= FLAG AND 1
2150 IF A THEN FLAG= FLAG-1: GOSUB 3160: WERT= WERT+168:
      GOSUB 3210 ELSE FLAG= FLAG+1: GOSUB 3210:
      WERT= WERT+168: GOSUB 3160
2160 POKE 49900, FLAG
2170 RETURN
```

Die nächsten vier Unterprogramme bewegen den Druckkopf in eine von 4 Richtungen, unabhängig von der Cursor-Tastenwahl und ob GROB oder FEIN bewegt werden soll. Die Bewegungssequenz bleibt in beiden Fällen gleich bis auf den Unterschied, daß für FEIN in den GRAPHIK-Modus umgeschaltet wird.

```
2180 `
2190 ` ### Move UP mit Pfeil oder (8)  ###
2200 `
2210 IF (FLAG AND 4)= 4 THEN GOSUB 3390: GOTO 2230
2220 IF ZEILE1 > 0 THEN ZEILE1= ZEILE1-1
2230 LPRINT CHR$(27)CHR$(10);
2240 GOSUB 3430: RETURN
```

```
2250 `
2260 ` ### Move DOWN mit Pfeil oder (2)  ###
2270 `
2280 IF (FLAG AND 4)= 4 THEN GOSUB 3390: GOTO 2300
2290 IF ZEILE1 < 999 THEN ZEILE1= ZEILE1+1
2300 LPRINT CHR$(10);
2310 GOSUB 3430: RETURN
```

```
2320 `
2330 ` ### Move LEFT mit Pfeil oder (4)  ###
2340 `
2350 GOSUB 3470: B= 8: GOSUB 3520
2360 RETURN
```



```
2370 `
2380 ` ### Move RIGHT mit Pfeil oder (6) ###
2390 `
2400 GOSUB 3470: B= 32: GOSUB 3610
2410 RETURN
```

Das folgende Unterprogramm führt den WAGENRÜCKLAUF und ZEILEN-VORSCHUB durch. Der Druckkopf bewegt sich bis zum linken Rand nach links und das Papier wird um eine Zeile vorgeschoben.

```
2420 `
2430 ` ### CR und Line Feed ( Drucker ) ###
2440 `
2450 LPRINT CHR$(13);: SPALTE1= LRAND: SPALTE2= 0
2460 IF ZEILE1 < 999 THEN ZEILE1= ZEILE1+1
2470 LPRINT CHR$(10);: GOSUB 2750: GOSUB 3350
2480 RETURN
```

Weiter geht es mit dem NULLSTELLER. Der ZEILENZÄHLER wird 0.

```
2490 `
2500 ` ### Nullsteller ###
2510 `
2520 ZEILE1= 0
2530 RETURN
```

Das folgende Unterprogramm dient zur SCHRIFTTYP Umschaltung und hebt die entsprechende Schriftbezeichnung in der Maske hervor. Die Variable SCHRIFT kommt zur Anwendung. Sie speichert die Adresse des Ortes der augenblicklich gültigen Schrifttyp-Bezeichnung.

```
2540 `
2550 ` ### Schrifttyp wählen ###
2560 `
2570 ANZAHL= 14: WERT= SCHRIFT: GOSUB 3210: GOSUB 3330
2580 IF SCHRIFT= 254 THEN SCHRIFT= 103 ELSE IF SCHRIFT= 271
    THEN SCHRIFT= 86 ELSE SCHRIFT= SCHRIFT+168
2590 WERT= SCHRIFT: GOSUB 3160
2600 RETURN
```

Mit folgendem Unterprogramm wird die Eingabezeile gelöscht. Die Routine CLEAZE übernimmt das eigentliche Löschen. Anschließend wird der Cursor-Pointer (Speicherstellen 49901,2) regeneriert und das Cursor-Punktchen "." neu gesetzt.

```
2610 `
2620 `   ### Zeile löschen   ###
2630 `
2640 CALL CLEAZE: POKE 49902,244: POKE 49901,238
2650 POKE 62702,46: POKE 62782,32
2660 RETURN
```

Das folgende Unterprogramm ist verantwortlich für die Wahl der Zeichenteilung von entweder 1/10, 1/12 oder 1/15 Inch. Die Variable TEILORT enthält die Adresse der "augenblicklich" aktiven Zeichenteilungsanzeige innerhalb der Briefmaschinenmaske. In ABSTAND steht der an den Drucker auszugebende Funktionswert. An dieser Stelle sei erwähnt, daß die als 1/15" bezeichnete Zeichenteilung im Grunde eine 7/120 (anstatt 8/120) Inch-Teilung ist (siehe Zeile 2730).

```
2670 `
2680 `   ### Schriftteilung   ###
2690 `
2700 ANZAHL= 4: WERT= TEILORT: GOSUB 3210
2710 IF TEILORT= 330 THEN TEILORT= 322: ABSTAND= 13: GOTO 2740
2720 TEILORT= TEILORT+4: ABSTAND= ABSTAND-2
2730 IF ABSTAND= 9 THEN ABSTAND= 8
2740 WERT= TEILORT: GOSUB 3160
2750 LPRINT CHR$(27)CHR$(31)CHR$(ABSTAND);
2760 RETURN
```

Die Umschaltung zwischen den zwei Programm-Modi erfolgt in den Zeilen 2800 bis 2820 . Im wesentlichen wird nur FLAG geändert, die Anzeige erneuert und die Eingabezeile gelöscht.

```
2770 `
2780 `   ### Schreib-Modus wählen   ###
2790 `
2800 ANZAHL= 15: WERT= 289: A= FLAG AND 2
2810 IF A= 2 THEN FLAG= FLAG-2: GOSUB 3160: WERT= WERT+168:
      GOSUB 3210 ELSE FLAG= FLAG+2: GOSUB 3210:
      WERT= WERT+168: GOSUB 3160
2820 POKE 49900, FLAG: GOSUB 2610: RETURN
```

Folgende Unterroutine schaltet zwischen GROB und FEIN um. Das FLAG und die Anzeige werden erneuert. Anhand des FLAGs können die Bewegungsroutinen entscheiden, ob der DRUCKKOPF in feinen oder "ganzen" Schritten bewegt werden soll.

```
2830 `
2840 `   ###   Grob/fein wählen   ###
2850 `
2860 ANZAHL= 4: WERT= 824: A= FLAG AND 4
2870 IF A= 4 THEN FLAG= FLAG-4: GOSUB 3160: WERT= WERT+168:
      GOSUB 3210 ELSE FLAG= FLAG+4: GOSUB 3210:
      WERT= WERT+168: GOSUB 3160
2880 POKE 49900, FLAG: RETURN
```

Ein DRUCKER-RESET bewirkt, daß der TRD 7020 in seinen Einschalt-Zustand, inkl. den entsprechenden Parametern, versetzt wird. Die Variablen LRAND und SPALTE werden mit neuen Werten geladen. Die Zeichenteilung 1/12" und die NORMAL-Schrift werden aktiviert.

```
2890 `
2900 `   ###   Reset auslösen   ###
2910 `
2920 LPRINT CHR$(27)CHR$(13)CHR$(80);
2930 WERT= SCHRIFT: ANZAHL= 14: GOSUB 3210: WERT= 86:
      SCHRIFT= WERT: GOSUB 3160
2940 LRAND= 0: SPALTE1= 0: SPALTE2= 0
2950 ANZAHL= 4: WERT= TEILORT: GOSUB 3210
2960 TEILORT= 322: ABSTAND= 13: GOTO 2720
2970 RETURN
```

Den "linken Rand" am Drucker festzusetzen bedarf es lediglich eines LPRINT-Befehls. Die "augenblickliche" Druckkopf-Position wird als Randwert übernommen und in LRAND gespeichert.

```
2980 `
2990 `   ###   Linken Rand setzen   ###
3000 `
3010 LRAND= SPALTE1: LPRINT CHR$(27)CHR$(57);
3020 RETURN
```

An dieser Stelle wird aus dem Programm "herausgesprungen".

```
3030 `
3040 `   ###   BREAK   ###
3050 `
3060 WIDTH 80: END
```

Um die EINGABEZEILE auszudrucken, bedienen wir uns der Variable TEXT\$ und des VARPTR-Befehls. VARPTR liefert bei String-Varia-

blen die Adresse einer in zwei Bytes gespeicherten Adresse, die auf den Beginn des Variablen-Textes weist bzw "pointet". Da wir TEXT\$ mit 80 Zeichen Länge definiert haben, brauchen wir ihr nur noch weiszumachen, daß der Variablen-Text bei der Eingabezeile beginnt. Ergebnis: die Variable TEXT\$ entspricht dann der Eingabezeile und läßt sich mit einem LPRINT-Befehl als "Ganzes" ausgeben. Um zu verhindern, daß - bei nicht ganz vollgeschriebener Eingabezeile - unnötige Leerzeichen nach dem Text mit ausgegeben werden, begrenzen wir den auszugebenden Teil der Variablen mittels LEFT\$ (TEXT\$). Zum Abschluß des Unterprogramms wird die Eingabezeile für neue Eingaben gelöscht.

```
3070 `
3080 ` ### Zeile drucken ###
3090 `
3100 A= VARPTR (TEXT$): POKE A+2, 244: POKE A+1, 238
3110 A= PEEK (49901)+ 256 * PEEK (49902): POKE A, 32
3120 LPRINT LEFT$( TEXT$, A-62702 );: GOSUB 2610: RETURN
```

Dies waren die 16 Unterprogramme der Briefmaschinen-Sonderfunktionen. Nun folgen einige Hilfsunterprogramme, die von o.g. Routinen aufgerufen werden.

Die Routine DIVERSE läßt sich zum INVERTIEREN wie auch zum DE-INVERTIEREN von Bildschirmstellen verwenden. Umgeschaltet wird durch einen Befehl mitten in der Routine:

```
SET 7 , (HL) ..... Invertieren
RES 7 , (HL) ..... De-Invertieren
```

Als Befehls-Code wird je nach Aufgabe SET oder RES in die Routine gePOKEt. Die Länge des Textes wird ebenfalls direkt in die Routine gePOKEt. Um letztlich noch zu wissen, wo der Text steht, wird der Routine vom BASIC-Programm aus ein ADRESS-PARAMETER (als Variable WERT) übergeben. Siehe auch Zeile 1670.

```
3130 `
3140 ` ### Textstellen INVERTIEREN ###
3150 `
3160 POKE 50028, 254: POKE 50023, ANZAHL: CALL DIVERSE (WERT)
3170 RETURN
```

```
3180 `
3190 ` ### Textstellen DE-INVERTIEREN ###
3200 `
3210 POKE 50028, 190: POKE 50023, ANZAHL: CALL DIVERSE (WERT)
3220 RETURN
```

Um bei der Schriftumschaltung auf eine Reihe von IF-THEN-ELSE zu verzichten, sind die Daten zur Bildschirmadresse der augenblicklich aktivierten Schriftart, zum Löschen derselben und zur Aktivierung der folgenden in DATA-Zeilen gespeichert. Diese werden nacheinander ausgelesen, bis die Subroutine die richtige Schriftart findet. In einem LPRINT-Befehl wird dann die alte Schriftart gelöscht und die neue aktiviert.

```
3230 `
3240 ` ### DATAs für Schriftumschaltung ###
3250 `
3260 DATA 86,0,0,27,69
3270 DATA 254,27,82,27,79
3280 DATA 103,27,38,27,87
3290 DATA 271,27,38,0,0
3300 `
3310 ` ### Schrifttype umschalten ###
3320 `
3330 RESTORE 3260
3340 READ A1,A2,A3,A4,A5: IF A1<>SCHRIFT THEN 3340
3350 LPRINT CHR$(A2)CHR$(A3)CHR$(A4)CHR$(A5);:RETURN
```

Die folgenden zwei Subroutinen werden von den Druckkopf-Bewegungs-Routinen bei FEIN-Einstellung aufgerufen. Sie schalten den GRAPHIK-Modus des Druckers an und aus.

```
3360 `
3370 ` ### Graphik-Modus an ###
3380 `
3390 LPRINT CHR$(27)CHR$(51);: RETURN

3400 `
3410 ` ### Graphik-Modus aus ###
3420 `
3430 LPRINT CHR$(27)CHR$(52);: RETURN
```

In den folgenden Programm-Zeilen wird ermittelt, ob HORIZONTAL in GROBen oder FEINen Schritten bewegt werden soll (anhand der Variable FLAG und deren Bit-Nr. 2) und dementsprechend eine Va-

riable A mit einem bestimmten Wert geladen. Für FEIN-Bewegungen wird ein LPRINT-Befehl ausgegeben, der den "horizontalen Bewegungs-Index" (Abstände zwischen den Zeichen) des TRD 7020 derart niedrig ansetzt, daß 10tel-Millimeter-Schritte möglich sind.

```
3440 `
3450 ` ### Horizontal-Bewegung ermitteln  ###
3460 `
3470 IF (FLAG AND 4)<>4 OR ABSTAND<>11 THEN A=0 : RETURN
3480 A=1: LPRINT CHR$(27)CHR$(31)CHR$(2);: RETURN
```

Der Spaltenzähler wird aus zwei Variablen gebildet:

SPALTE1 für GROBE bzw. "ganze" Schritte

SPALTE2 für FEIN-Bewegungen

Um die Werte der beiden SPALTE-Variablen bei Druckkopfbewegungen kümmern sich die letzten beiden Subroutinen.

```
3490 `
3500 ` ### Spaltenzähler setzen (LINKS)  ###
3510 `
3520 IF SPALTE1= 0 AND SPALTE2= 0 THEN RETURN
3530 IF SPALTE1>0 AND A=0 THEN SPALTE1= SPALTE1-1: GOTO 3570
3540 IF A=0 THEN RETURN ELSE SPALTE2= SPALTE2-1
3550 IF SPALTE2<0 AND SPALTE1= 0 THEN RETURN
3560 IF SPALTE2<0 THEN SPALTE1= SPALTE1-1: SPALTE2= 9
3570 LPRINT CHR$(B)CHR$(27)CHR$(31)CHR$(11);: RETURN
```

```
3580 `
3590 ` ### Spaltenzähler setzen (RECHTS)  ###
3600 `
3610 IF SPALTE1>135 THEN RETURN
3620 IF A=0 THEN SPALTE1= SPALTE1+1: GOTO 3570
3630 SPALTE2= SPALTE2+1
3640 IF SPALTE2>9 THEN SPALTE1= SPALTE1+1: SPALTE2= 0
3650 GOTO 3570
```

Bevor Sie die Briefmaschine starten, speichern Sie sie erst mal ab. Sicher ist sicher. Besonders bei den DATA-Zeilen der Z-80-Routinen, die wir nachfolgend vorstellen werden, kann es leicht zu Tippfehlern kommen.

Die Anpassung der Briefmaschine an "neue ROM PC's" liegt im Bereich der Z-80-Routinen. Die Änderungen sind daher in den DATA-Zeilen dieser Routinen vorzunehmen.

5. Z-80-ROUTINEN

BLANKO

Die Routine BLANKO löscht den von der Briefmaschine erweiterten Bildschirm inkl. Attribute. Hierzu wird ein sog. Blocklade-Befehl, LDIR, verwendet, der folgendes bewirkt: der durch (HL) adressierte Wert wird in die von DE adressierte Speicherstelle geladen, HL und DE inkrementiert und BC dekrementiert. Dies geschieht solange bis BC = 0 ist. BLANKO setzt BC auf 4096 (4K), HL auf die erste VIDEO-RAM Speicherstelle 61440 und DE auf die folgende (61441). Anschließend wird dafür gesorgt, daß 61440 mit 0 geladen wird. Der LDIR-Befehl überträgt so 4096 mal den Wert 0 von einer Speicherstelle in die folgende.

BLANKO: ab Adresse 50000

```
50000 LD   HL, 61440    ; 1.VIDEO-BYTE in HL
50003 LD   DE, 61441    ; 2.VIDEO-BYTE in DE
50006 LD   BC, 4096     ; Zähler (4096) in BC
50009 LD   (HL), 0      ; 1.VIDEO-BYTE löschen
50011 LDIR                ; 4096mal, 0 von HL nach DE übertragen
50013 RET                ; Rückkehr zu BASIC
```

DIVERSE

Die Routine DIVERSE dient zum hervorheben von Textstellen auf dem Bildschirm und gleichzeitig zur Aufhebung dieses "INVERTIERENS". Drei Dinge muß die Routine wissen: 1) Soll invertiert/oder de-invertiert werden, 2) wo (Adresse) steht der jeweilige Text und 3), wie lang ist dieser Text?

Da die Routine nur in einem Befehl geändert werden muß, um entweder zu invertieren oder zu de-invertieren, wird jeweils nur die entsprechende Speicherstelle mit dem gewünschten Befehls-Code bePOKEt (POKE 50028 in Zeilen 3160 bzw. 3210). Die Adresse des Textes, in der Variablen WERT gespeichert, wird durch den BASIC-Interpreter übergeben. DIVERSE (HL) erhält einen Adress-Wert, der auf zwei Speicherstellen zeigt, die die eigentliche Bildschirmadresse enthalten. Die Länge des Textes wird direkt in die Routine gePOKEt (Adresse 50023 bei LD B,...). Damit bei

der Invertiererei kein Flimmern entsteht, benützen wir eine ROM Routine, im folgenden NOFLIM genannt, zu dessen Unterdrückung. Als Zähler wird das Register B verwendet, das über einen Spezial-Befehl verfügt: DJNZ (siehe Z-80-Befehle). Da Bit-Nr 7 der Attribute für die INVERS-Darstellung zuständig ist, wird dieses Bit bei jedem Schleifendurchgang entweder geSETzt oder zurückgeRESetzt. Als Pointer wird HL verwendet. Der Attribute-Bereich fängt bei Adresse 63488 an. Da WERT einen INTEGER-Wert enthalten muß, wird WERT mit Werten zwischen 0 und 2048 übergeben, wozu dann von der Z-80-Routine 63488 addiert werden.

DIVERSE: ab Adresse 50014

```

50014 LD   E, (HL)      ; LSB der Pointer-Adresse in E
50015 INC  HL           ; HL zeigt auf MSB
50016 LD   D, (HL)      ; MSB der Pointer-Adresse in D
50017 EX   DE, HL       ; Pointer-Adresse von DE nach HL
50018 LD   DE, 63488    ; Attribute-Start in DE zur Addition
50021 ADD  HL, DE        ; WERT + Attribute-Start in HL
50022 LD   B, 0          ; Zähler in B, von BASIC gePOKEt
50024 CALL NOFLIM       ; Flimmer-Unterdrückung rufen
50027 SET  7, (HL)      ; Bit-Nr 7 von Attribut SET oder RES
50029 INC  HL           ; HL zeigt auf nächstes Attribut
50030 DJNZ 50024        ; Sprung in Schleife solange B > 0
50032 RET               ; Zurück zu BASIC

```

CLEAZE

Die Routine CLEAZE dient zur schnellen Löschung der Eingabezeile, die insgesamt 80 Zeichen umfaßt. Auch hier wird NOFLIM verwendet, um ein Flimmern zu unterdrücken. HL zeigt nacheinander auf die einzelnen Speicherstellen der Eingabezeile und beschreibt diese mit dem Wert 32 (SPACE). Register B fungiert als Zähler von 80 bis 0.

CLEAZE: ab Adresse 50033

```

50033 LD   HL, 62702    ; Startadresse der Eingabezeile
50036 LD   B, 80        ; Zähler auf 80 (...Zeichen)
50038 CALL NOFLIM       ; Flimmerunterdrückung rufen
50041 LD   (HL), 32     ; Speicherstelle löschen
50043 INC  HL           ; HL zeigt auf nächste Speicherstelle
50044 DJNZ 50038        ; Zurück in die Schleife falls B > 0
50046 RET               ; Rückkehr zu BASIC

```


TASTATUR

Die TASTATUR-Routine ist die weitaus wichtigste der vier Z-80-Routinen. Hier werden alle Tastaturabfragen und Unterprogramm-Code-Zuweisungen gemanagt. Die getippten Zeichen werden in der Eingabezeile sichtbar und die Funktion BACKSPACE (Zeichen löschen) wird innerhalb der TASTATUR-Routine ausgeführt. Die Schnittstelle zum BASIC-Teil, der hauptsächlich die Unterprogramme ansteuert, besteht in der Speicherstelle 49908, die den GOSUB-Code enthält.

ZUR ANPASSUNG AN NEUE ROMs: Zu Beginn der Routine wird eine ROM Routine zur Tastaturabfrage leicht geändert, die bei neuen ROM PC's adressmäßig woanders liegt. Hinzu kommt eine DELAY-Routine, die ebenfalls in neuen ROMs woanders liegt. Die schon zweimal verwendete NOFLIM-Routine (DIVERSE & CLEAZE) ändert ebenfalls ihren Sitz bei den neuen ROMs.

Wie die Anpassung vorzunehmen ist, wird im Anschluß an die Beschreibung der Z-80-Routinen gezeigt.

Der Einsprung in die Routine erfolgt bei Adresse 50058. Die Befehle der Bytes 50047 bis 50057 werden zum Rücksprung ins BASIC dadurch genutzt, daß hier der GOSUB-Code in Speicherstelle 49908 geladen und die ROM-Routine zur Tastaturabfrage wiederhergestellt wird.

TASTATUR: ab Adresse 50058

```

50047 LD   A, B           ; GOSUB-Code von B nach A
50048 LD   (49908), A     ; Code für BASIC in Adresse 49908
50051 LD   HL, 60184      ; Daten um ROM-Routine zu regenerieren
50054 LD   (59306), HL    ; ROM-Routine durch HL-Bytes erneuern
50057 RET                ; Rückkehr zum BASIC

50058 LD   HL, 0          ; EINSPRUNG VON BASIC AUS, HL wird 0
50061 LD   (59306), HL    ; ROM-Routine verändern
50064 LD   (49908), HL    ; GOSUB-Code Bytes löschen
50067 LD   A, (49900)     ; STATUS-Byte in A
50070 LD   C, A           ; STATUS-Wert von A nach C
50071 LD   B, 1           ; B = 1, CURSOR-WAHL - Code
50073 IN   A, (41)        ; TASTEN-CHECK
50075 BIT  6, A           ; Test ob: CTRL - Taste ?
50077 JR   NZ, 50047      ; wenn ja -> 50047 (BASIC)
50079 INC  B              ; B = 2, MOVE UP - Code
50080 BIT  0, C           ; welche Cursor-Tasten aktiviert ?
50082 JR   NZ, 50105      ; wenn 2,4,6,8 -> 50105

```

```
50084 BIT 3, A ; Test ob: PFEIL HOCH - Taste ?
50086 JR NZ, 50047 ; wenn ja -> 50047 (BASIC)
50088 INC B ; B = 3, MOVE DOWN - Code
50089 BIT 0, A ; Test ob: PFEIL RUNTER - Taste ?
50091 JR NZ, 50047 ; wenn ja -> 50047 (BASIC)
50093 INC B ; B = 4, MOVE LEFT - Code
50094 BIT 2, A ; Test ob: PFEIL LINKS - Taste ?
50096 JR NZ, 50047 ; wenn ja -> 50047 (BASIC)
50098 INC B ; B = 5, MOVE RIGHT - Code
50099 BIT 1, A ; Test ob: PFEIL RECHTS - Taste ?
50101 JR NZ, 50047 ; wenn ja -> 50047 (BASIC)
50103 JR 50134 ; überspringe 2,4,6,8 - Abfrage ->
50105 LD B, 2 ; B = 2, MOVE UP - Code
50107 IN A, (33) ; TASTEN-CHECK
50109 BIT 0, A ; Test ob: "8" - Taste (10er Block) ?
50111 JP NZ, 50047 ; wenn ja -> 50047 (BASIC)
50114 IN A, (32) ; TASTEN-CHECK
50116 INC B ; B = 3, MOVE DOWN - Code
50117 BIT 2, A ; Test ob: "2" - Taste (10er Block) ?
50119 JP NZ, 50047 ; wenn ja -> 50047 (BASIC)
50122 INC B ; B = 4, MOVE LEFT - Code
50123 BIT 4, A ; Test ob: "4" - Taste (10er Block) ?
50125 JP NZ, 50047 ; wenn ja -> 50047 (BASIC)
50128 INC B ; B = 5, MOVE RIGHT - Code
50129 BIT 6, A ; Test ob: "6" - Taste (10er Block) ?
50131 JP NZ, 50047 ; wenn ja -> 50047 (BASIC)
50134 INC B ; B = 6, CR + LF - Code
50135 IN A, (42) ; TASTEN-CHECK
50137 BIT 4, A ; Test ob: "RETURN" - Taste ?
50139 JP NZ, 50047 ; wenn ja -> 50047 (BASIC)
50142 INC B ; B = 7, NULLSTELLER - Code
50143 IN A, (43) ; TASTEN-CHECK
50145 BIT 7, A ; Test ob: (F1) - Taste ?
50147 JP NZ, 50047 ; wenn ja -> 50047 (BASIC)
50150 INC B ; B = 8, SCHRIFT-TYP - Code
50151 BIT 6, A ; Test ob: (F2) - Taste ?
50153 JP NZ, 50047 ; wenn ja -> 50047 (BASIC)
50156 INC B ; B = 9, ZEILEN-LÖSCH - Code
50157 BIT 5, A ; Test ob: (F3) - Taste ?
50159 JP NZ, 50047 ; wenn ja -> 50047 (BASIC)
50162 INC B ; B = 10, SCHRIFT-TEILUNG - Code
50163 BIT 4, A ; Test ob: (F4) - Taste ?
50165 JP NZ, 50047 ; wenn ja -> 50047 (BASIC)
50168 INC B ; B = 11, SCHREIB-MODUS - Code
50169 BIT 3, A ; Test ob: (F5) - Taste ?
50171 JP NZ, 50047 ; wenn ja -> 50047 (BASIC)
50174 INC B ; B = 12, GROB/FEIN - Code
```

```

50175 BIT 2, A ; Test ob: (F6) - Taste ?
50177 JP NZ, 50047 ; wenn ja -> 50047 (BASIC)
50180 INC B ; B = 13, DRUCKER-RESET - Code
50181 IN A, (41) ; TASTEN-CHECK
50183 BIT 7, A ; Test ob: "ESC" - Taste ?
50185 JP NZ, 50047 ; wenn ja -> 50047 (BASIC)
50188 INC B ; B = 14, LINKER-RAND - Code
50189 IN A, (42) ; TASTEN-CHECK
50191 BIT 3, A ; Test ob: "TAB" - Taste ?
50193 JP NZ, 50047 ; wenn ja -> 50047 (BASIC)
50196 INC B ; B = 15, BREAK - Code
50197 IN A, (32) ; TASTEN-CHECK
50199 BIT 0, A ; Test ob: "0" - Taste (10er Block) ?
50201 JR Z, 50210 ; wenn nein -> 50210, kein BREAK
50203 IN A, (42) ; TASTEN-CHECK
50205 BIT 2, A ; Test ob: "SHIFT" - Taste ?
50207 JP NZ, 50047 ; wenn ja -> 50047 (BASIC) und ENDE
50210 BIT 1, C ; welcher Schreib-Modus ?
50212 JR Z, 50229 ; wenn DIREKT -> 50229
50214 INC B ; B = 16, ZEILEN-DRUCK - Code
50215 IN A, (33) ; TASTEN-CHECK
50217 BIT 6, A ; Test ob: "=" - Taste (10er Block) ?
50219 JP NZ, 50047 ; wenn ja -> 50047 (BASIC)
50222 IN A, (42) ; TASTEN-CHECK
50224 BIT 1, A ; Test ob: "INS/DEL" - Taste ?
50226 JP NZ, 50316 ; wenn ja -> 50316 zu BACKSPACE

50229 CALL 59284 ; TASTATUR-ABFRAGE mit ASCII-Wert in A
50232 OR A ; ist A = 0 (...kein Zeichen) ?
50233 JP Z, 50071 ; wenn ja -> 50071 (CHECK-Schleife)
50236 CP 127 ; ASCII - Wert - Vergleich mit 127
50238 JP NC, 50071 ; A > 126 ? -> 50071 (CHECK-Schleife)
50241 CP 32 ; ASCII - Wert - Vergleich mit 32
50243 JP C, 50071 ; A < 32 ? -> 50071 (CHECK-Schleife)
50246 BIT 0, C ; welche Cursor-Tasten sind aktiv ?
50248 JR Z, 50261 ; bei Cursor-Pfeil-Tasten -> 50261
50250 CP 57 ; Vergleich ob ASCII zwischen 50 und 56
50252 JR NC, 50261 ; bei 2468-Cursor-Tasten, > 56 -> 50261
50254 CP 50 ; Vergleich ob ASCII < 50
50256 JR C, 50261 ; ASCII < 50 -> 50261, O.K.... sonst,
50258 JP 50071 ; -> Zurück zur CHECK-Schleife
50261 BIT 1, C ; welcher Schreib-Modus aktiv ?
50263 JR Z, 50270 ; wenn DIREKT -> 50270 (...überspringen)
50265 CP 61 ; Vergleich mit "=" bei ZEILEN-Modus
50267 JP Z, 50071 ; wenn ja -> 50071 (CHECK-Schleife)
50270 LD HL, (49901) ; Cursor-Adresse nach HL um zu checken,
50273 LD DE, 62782 ; ob letztes Zeichen schon erreicht...

```

```

50276 LD    B, A      ; ASCII-Wert nach B retten
50277 CALL 50342      ; -> COMPARE, HL und DE vergleichen
50280 JP    Z, 50071   ; wenn letzte Stelle erreicht -> 50071
50283 LD    A, B      ; ASCII-Wert wieder nach A
50284 CALL NOFLIM     ; -> NOFLIM, Flimmern unterdrücken
50287 LD    (HL), B    ; Zeichen an Cursor-Stelle anzeigen
50288 INC   HL        ; HL eine Stelle weiterrücken
50289 LD    (HL), 46    ; ... und dort Cursor "." anzeigen
50291 LD    (49901), HL ; neue Cursor-Adresse in 49901 / 49902
50294 BIT   1, C       ; welcher Schreib-Modus aktiv ?
50296 JP    NZ, 50071  ; wenn ZEILEN-Modus -> 50071
50299 IN    A, (16)    ; PORT 16 des PC abfragen
50301 BIT   5, A       ; SERIELLE oder PARALLELE Ausgabe ?
50303 LD    C, 64      ; C = aktiver Printer-Port, 64 = S.
50305 JR    NZ, 50309  ; wenn SERIELL -> 50309 (zu OUT)
50307 LD    C, 48      ; C = 48 = PARALLELE - Ausgabe
50309 OUT   (C), B     ; ASCII - Wert an Drucker ausgeben
50311 LD    B, 0       ; B = 0 für "kein GOSUB" in BASIC
50313 JP    50047      ; -> 50047, zurück zum BASIC

50316 LD    HL, (49901) ; "BACKSPACE", HL = Cursor-Adresse
50319 LD    DE, 62703   ; DE mit erster EINGABE-ZEILEN-Adresse
50322 CALL 50342      ; COMPARE, vergleicht HL und DE
50325 JP    C, 50071   ; Cursor schon ganz Links -> 50071
50328 CALL NOFLIM     ; NOFLIM, Flimmern unterdrücken
50331 LD    (HL), 32    ; Cursor löschen
50333 DEC   HL        ; HL eine Stelle zurück
50334 LD    (HL), 46    ; Zeichen mit Cursor überschreiben
50336 LD    (49901), HL ; neue Cursor-Adresse nach 49901,49902
50339 JP    50348      ; -> DELAY, zur Warteschleife

50342 LD    A, H       ; "COMPARE", H nach A zum MSB-Vergleich
50343 SUB   D          ; H und D werden verglichen
50344 RET   NZ         ; wenn H <> D -> Zurück zu Routine
50345 LD    A, L       ; sonst L nach A zum LSB-Vergleich
50346 SUB   E          ; L und E werden verglichen
50347 RET             ; -> RETURN, Flag's sind gesetzt

50348 LD    HL, 25344   ; "DELAY", HL mit Schleifen-Wert laden
50351 CALL 59041      ; Warteschleife (ROM) aufrufen
50354 JP    50071      ; -> 50071 (CHECK-Schleife)

```

Damit hätten wir alle vier Routinen erläutert. Die Anpassung der verschiedenen ROM-Routinen-Aufrufe geschieht durch Austausch bestimmter DATAs, nämlich der, die einer Routinen-Adresse entsprechen. Folgende Zeilen müssen geändert werden:

1090 DATA 94,35,86,235,17,0,248,25,6,0,205,40, 234, 203
1110 DATA 33,238,244,6,80,205,40, 234, 54,32,35,16,248,201
1120 DATA 120,50,244,194,33,24,235,34,120, 231, 201,33,0,0
1130 DATA 34, 120, 231, 34,244,194,58,236,194,79,6,1,219,41
1250 DATA 127,195,219,42,203,79,194,140,196,205, 98, 231
1300 DATA 120,205, 39, 234, 112,35,54,46,34,237,194,203,73
1330 DATA 205,166,196,218,151,195,205, 39, 234, 54,32,43
1350 DATA 124,146,192,125,147,201,33,0,99,205, 124, 230

6. BEDIENUNGSANLEITUNG

Einführung:

Das Programm "BRIEFMASCHINE" dient zur Erstellung kleinerer Schriftstücke, dem Ausfüllen von Formularen und ähnlichen Aufgaben, die eine direkte Kontrolle des verwendeten Druckers verlangen.

Die BM ist ein TA-PC-Programm, daß speziell auf den Typenrad drucker TRD 7020 von Triumph-Adler zugeschnitten worden ist und seine zahlreichen Sonderfunktionen weitgehend unterstützt.

Programmstart:

Die BRIEFMASCHINE kann sowohl unter ROM- als auch Disk-BASIC "laufen". Unabhängig davon, ob Sie sich noch mit Cassetten rumplagen oder schon Disketten benutzen: erst laden dann starten. Legen Sie die Programm-Diskette in das Floppy-Laufwerk und starten Sie die BRIEFMASCHINE mit RUN "brief". Die BM erwartet einen TRD 7020 oder kompatiblen, der sich im ONLINE-Modus befindet. Sollte der angeschlossene Drucker noch nicht betriebsbereit sein, meldet sich die BM mit der Aufforderung:

" Schalten Sie den angeschlossenen Drucker auf O N L I N E "

Sobald diese Hürde genommen ist, wird die Arbeitsmaske der BM aufgebaut: Der Bildschirm wird in mehrere Felder eingeteilt. Bevor wir uns eingehender mit dem Maskenaufbau und den damit zusammenhängenden Funktionen beschäftigen sei erwähnt, daß dem Anwender zwei Arbeits- bzw. Betriebs-Modi von der BM zur Verfügung gestellt werden.

Im sog. DIREKT-MODUS verhält sich die BM wie eine Schreibmaschine, d.h., wir benutzen den TA-PC praktisch nur als Tastatur. Jedes in die Tastatur getippte Zeichen erscheint nicht nur in der sog. Eingabezeile im Mittelfeld des Bildschirms, sondern wird auch direkt am Drucker ausgegeben.

In diesem Modus befindet sich die BM nach dem Programm-Start. Dieser Schreibmaschinen-Modus wird besonders bei der Formular-Bearbeitung bzw. beim Beschriften und Erstellen kurzer Notizen Verwendung finden.

Der ZEILEN-MODUS versetzt den Anwender in die Lage, den Text zunächst in die Eingabezeile einzugeben - bis zu max. 80 Zeichen finden darin Platz - und erst danach, z.B. nach ev. Korrektur, ausdrucken zu lassen. Dieses Verhalten entspricht mehr dem einer Speicherschreibmaschine und wird bei längeren Texten bevorzugt werden.

Die Maske:

Die Maske enthält diverse Anzeigen und die schon erwähnte EINGABE-ZEILE. Der Eingabebereich ist INVERS hervorgehoben und genau 80 Zeichen breit. Sie können demnach 80 Zeichen Text auf einmal behandeln. Einige der dazugehörigen Funktionen, wie z.B. Zeile DRUCKEN, LÖSCHEN oder die BACKSPACE-TASTE, sind direkt unter der EINGABE-ZEILE als Hilfs-Texte nochmals aufgeführt.

Wenden wir uns nun den Anzeigen zu und beginnen links oben in der Bildschirm-Ecke. Das erste Anzeigenfeld ist das SCHRIFT-Feld. In diesem Bereich zeigt die BM an, in welcher Schriftart gerade gedruckt wird. Als Schriftarten stehen zur Verfügung:

1. Die Normalschrift
2. Die unterstrichene Normalschrift
3. Die doppelt gedruckte Normalschrift
4. und die Schattenschrift (Fettdruck)

Beim Programm-Start befindet sich die BM im NORMALSCHRIFT-Modus. Dies wird, wie jede andere aktivierte Funktion auch, in einem invers dargestellten Schriftfeld kenntlich gemacht.

Von links nach rechts wandernd erblicken wir das MODUS-Feld. An dieser Stelle zeigt die BM an, in welchem Arbeits-Modus sie sich gerade befindet (siehe 1.2).

Die dritte große Anzeige signalisiert über welchen PORT die Daten zum Drucker gelangen: den SERIELLEN oder den PARALLELEN.

Die letzte Anzeige auf dieser Höhe zeigt an, welche ZEICHEN-TEILUNG gewählt wurde. Wählbar sind 1/10, 1/12 oder 1/15 Inch (Zoll) als Zeichenabstand, d.h., es passen z.B. 10 oder 12 Anschläge auf 1 Zoll (Schreibweise auch: 10").

Die zweite Anzeigen-Gruppe liegt eine Ebene tiefer, direkt über der EINGABEZEILE. Von links beginnend wird zunächst die CURSOR-WAHL angezeigt. Zur Kontrolle des Druckkopfes stehen dem Anwender zwei Steuertasten-Sets zur Verfügung. Neben den gewöhnlichen CURSOR-(Pfeil)-TASTEN, rechts und links von der Space-Taste, können auch über die Tasten 2,4,6 und 8 des 10er-Blocks Druckkopf-Bewegungen veranlaßt werden.

Die folgenden 3 Anzeigen ZEILE, SPALTE und LRAND sind als Orientierungs-Hilfe gedacht und haben die Funktion einer Positions-Anzeige des DRUCKKOPFES. Die LRAND-Anzeige enthält die Angabe, an welcher Spalten-Position der LINKE RAND festgelegt wurde.

Das RESET-Feld ist als Erinnerungshilfe gedacht und sagt Ihnen, mit welcher Taste ein DRUCKER-RESET, d.h., praktisch ein Warm-Start des Druckers, ausgelöst werden kann.

Das letzte Anzeigen-Feld, GROB/FEIN, bezieht sich auf die Kontrolle des Druckkopfes. Es ist nämlich möglich, den Druckkopf entweder in "groben" oder in "feinen" Schritten zu bewegen. Ein grober Schritt nach rechts entspricht einer Horizontal-Bewegung um ein ganzes Zeichen und ein feiner Schritt nach rechts bewegt den Druckkopf nur ein paar 10tel Millimeter. Auf diese Art und Weise kann der Druckkopf in der Vertikalen und in der Horizontalen sehr genau positioniert werden.

Um bei der Einarbeitung in die Möglichkeiten der BM nicht immer wieder diese Anleitung zu Rate ziehen zu müssen, sind entweder in oder an jedem Anzeigenfeld die der Funktion entsprechenden Tasten bzw. Tastenkombinationen angegeben.

Die Angabe (F6) unterhalb des GROB/FEIN-Feldes gibt z.B an, daß eine Umschaltung mit der Funktionstaste Nr. 6 möglich ist. Eine Übersicht über die BM-Funktionen und ihren Auslösern finden Sie am Schluß dieser Anleitung.

Die Funktionen:

Cursor-Wahl

Der Druckkopf läßt sich mit den Cursor-Tasten horizontal und vertikal bewegen. Unabhängig davon, ob dies in feinen oder groben Schritten erfolgt, läßt sich jedoch eine zweite Kontrollmöglichkeit wählen: über die Tasten 2, 4, 6 und 8 des

10er- Blocks.

Die Anzeige unterscheidet dabei zwischen "<-->" und "2468". Die Umschaltung von der einen Cursor-Steuerung zur anderen erfolgt mittels der CTRL-TASTE (auf der linken Seite der Tastatur).

CR+LF

Die Kürzel CR (für Carriage Return = Wagen zurück) und LF (für Line Feed = Zeilen-Vorschub) stehen für die <ENTER> / <RETURN> Funktion, die bewirkt, daß der Druckkopf zurück auf den eingestellten linken Rand fährt und das Papier um eine Zeile vorgerückt wird (RETURN-TASTE).

NULLSTELLER

Der sog. Nullsteller (denken Sie an Ihr Cassetten-Gerät) dient dazu, die ZEILENANZEIGE bei Bedarf auf Null zurückzustellen. Dies geschieht mit der FUNKTIONSTASTE Nr. 1.

SCHRIFTTYP

Um zwischen den 4 wählbaren Schriften umzuschalten, genügt die Betätigung der FUNKTIONSTASTE Nr. 2.

ZEILE LÖSCHEN

Mit dieser Funktion schaffen Sie wieder Platz in der Eingabezeile. Die gesamte Zeile wird gelöscht und der Eingabe-Cursor "." steht wieder auf der linken Seite der Zeile. Betätigen Sie hierzu die FUNKTIONSTASTE Nr. 3.

SCHRIFTTEILUNG

Es sind drei Schriftteilungen wählbar: 1/10, 1/12 oder 1/15tel Zoll. Für bestimmte Schrift-Arten ist der richtige Abstand unerlässlich, da sonst alles etwas bedrückt bzw. ausgeweitet aussehen kann. Beachten Sie bitte, daß bei Verwendung der 1/10 bzw. 1/15-Teilung keine Feinbewegungen des Druckkopfes in der Horizontalen möglich sind. Die Vertikale ist davon nicht betroffen, so daß einem Hoch- bzw. Tiefstellen nichts im Wege steht. Im 1/12-Modus, der beim Programmstart gewählt wird, sind in dieser Hinsicht keine Einschränkungen gegeben. Zur Umschaltung betätigen Sie bitte FUNKTIONSTASTE Nr. 4.

ARBEITS-MODUS

Die Umschaltung zwischen den zwei schon erwähnten Arbeits-Modi

der BM erfolgt mittels der FUNKTIONSTASTE Nr. 5.

GROB/FEIN

Diese Funktion erlaubt eine Umschaltung zwischen GROB und FEIN im Hinblick auf die Positionierung des Druckkopfes. Die feine Bewegung ist in der Horizontalen nur mit der 1/12er-Zeichenteilung möglich. Dies hängt mit der zweiten SPALTEN-Anzeige zusammen, die von 0 bis 9 zehn Schritte kennt und nur bei einer 1/12-Teilung funktioniert. Die Umschaltung erfolgt mittels der FUNKTIONSTASTE Nr. 6.

DRUCKER-RESET

Mit der Taste ESC veranlassen Sie einen Drucker-Warmstart, d.h., alle Voreinstellungen werden aufgelöst (Schrift, Linker Rand) und der Druckkopf auf seine Ursprungs-Position gebracht. Die gewählte Zeichenteilung bleibt jedoch.

LINKER RAND

Ein LINKER RAND kann mittels der TAB-TASTE an der augenblicklichen Druckkopf-Position (SPALTE) eingestellt wrden. Bei CR + LF wird dann nicht mehr ganz links an den Rand gefahren, sondern nur bis zur LRAND-SPALTEN-Position. Dies ist besonders für Tabellen-Eingaben nützlich.

ENDE

Mit der Tasten-Kombination SHIFT + "0" können Sie das Programm BM beenden und in den BASIC-MODUS zurückkehren. Die BM wird dabei nicht gelöscht und kann jederzeit neu gestartet werden.

ZEILE DRUCKEN

Um den im ZEILEN-Modus eingegebenen Text auch auszudrucken, bestätigen Sie bitte das "=" - Zeichen rechts über dem 10er-Block. Die Eingabezeile wird danach für neue Eingaben gelöscht.

BACKSPACE

Diese Funktion des ZEILEN-MODUS erlaubt ein Überschreiben des Zeichens links neben dem Eingabe-Cursor. Die Taste INS/DEL ist der Auslöser. Damit können in der Eingabe-Zeile Korrekturen ausgeführt werden.

Zum Schluß:

Wie immer macht die Übung den Meister. Für die BM lassen sich sicher vielerlei Einsatzmöglichkeiten finden und man wird sie bestimmt immer wieder für kleine Druckarbeiten einsetzen. Zum Schluß noch eine Gesamtübersicht der BM-Features und Tasten:

CURSOR-WAHL	CTRL-TASTE
DRUCKKOPF HOCH	Pfeil / (8)
DRUCKKOPF RUNTER	Pfeil / (2)
DRUCKKOPF LINKS	Pfeil / (4)
DRUCKKOPF RECHTS	Pfeil / (6)
CR und LF	RETURN-TASTE
NULLSTELLER	(F1)
SCHRIFTTYP	(F2)
ZEILE LÖSCHEN	(F3)
SCHRIFTEILUNG	(F4)
ARBEITS-MODUS	(F5)
GROB / FEIN	(F6)
DRUCKER-RESET	ESC-TASTE
LINKER RAND	TAB-TASTE
PROGRAMMENDE	SHIFT + "0"
ZEILE DRUCKEN	"=" des 10er Blocks
BACKSPACE	INS/DEL-TASTE

Programm-Listing

Die Briefmaschine

In Anschluß an diesen Einführungskurs in die Maschinensprache des Z-80 listen wir hier nochmals das Programm als Zusammenfassung auf.

Damit wird denjenigen Lesern, die sich nicht selbst mit der Programmierung befassen wollen, andererseits gerne das Programm für Formularbeschriftungen (Schreiben von Schecks u.a.m) nutzen möchten, um so die Möglichkeiten des TRD 7020 besser auszunutzen, das Abtippen und Kontrollieren erleichtert.

Beachten Sie dabei jedoch bitte: Dieses Listing ist lauffähig auf PC's mit "alter" PROM-Version (Gerätenummern bis 5000). Haben Sie einen PC mit höherer Seriennummer, ändern Sie die DATA-Zeilen 1090 bis 1350 (siehe Seite 230).

Die Zeilen für die Bildschirmmaske (1680 bis 1850) haben wir mit "Hilfszeichen" versehen, an deren Stelle die "richtigen" Grafik-Zeichen einzugeben sind (siehe Seiten 214/215). Ein Typenraddrucker wie der TRD 7020 hätte sonst an dieser Stelle einen konfus erscheinenden "Buchstabensalat" ausgedruckt.

```

1 REM * BRIEFMASCHINE (C) Copyright 1985 Patrick V. Thomas *
1000 ' *****
1010 ' *
1020 ' *          DIE BRIEFMASCHINE          *
1030 ' *
1040 ' *****
1050 '
1060 ' ### Maschinen-Code Routinen als DATA's ###
1070 '
1080 DATA 33,0,240,17,1,240,1,0,16,54,0,237,176,201
1090 DATA 94,35,86,235,17,0,248,25,6,0,205,88,234,203
1100 DATA 254,35,16,248,201
1110 DATA 33,238,244,6,80,205,88,234,54,32,35,16,248,201
1120 DATA 120,50,244,194,33,24,235,34,170,231,201,33,0,0
1130 DATA 34,170,231,34,244,194,58,236,194,79,6,1,219,41
1140 DATA 203,119,32,224,4,203,65,32,21,203,95,32,215,4
1150 DATA 203,71,32,210,4,203,87,32,205,4,203,79,32,200
1160 DATA 24,29,6,2,219,33,203,71,194,127,195,219,32,4
1170 DATA 203,87,194,127,195,4,203,103,194,127,195,4,203
1180 DATA 119,194,127,195,4,219,42,203,103,194,127,195,4
1190 DATA 219,43,203,127,194,127,195,4,203,119,194,127

```

```

1200 DATA 195,4,203,111,194,127,195,4,203,103,194,127
1210 DATA 195,4,203,95,194,127,195,4,203,87,194,127,195
1220 DATA 4,219,41,203,127,194,127,195,4,219,42,203,95
1230 DATA 194,127,195,4,219,32,203,71,40,7,219,42,203,87
1240 DATA 194,127,195,203,73,40,15,4,219,33,203,119,194
1250 DATA 127,195,219,42,203,79,194,140,196,205,148,231
1260 DATA 183,202,151,195,254,127,210,151,195,254,32,218
1270 DATA 151,195,203,65,40,11,254,57,48,7,254,50,56,3
1280 DATA 195,151,195,203,73,40,5,254,61,202,151,195,42
1290 DATA 237,194,17,62,245,71,205,166,196,202,151,195
1300 DATA 120,205,87,234,112,35,54,46,34,237,194,203,73
1310 DATA 194,151,195,219,16,203,111,14,64,32,2,14,48
1320 DATA 237,65,6,0,195,127,195,42,237,194,17,239,244
1330 DATA 205,166,196,218,151,195,205,87,234,54,32,43
1340 DATA 54,46,34,237,194,195,172,196
1350 DATA 124,146,192,125,147,201,33,0,99,205,161,230
1360 DATA 195,151,195
1370 '
1380 ' ### Mit FOR-NEXT Schleife einlesen ###
1390 '
1400 FOR SPEICHER= 50000! TO 50000!+356
1410 READ DATEN: POKE SPEICHER,DATEN
1420 NEXT SPEICHER:BLANKO=50000!:DIVERSE=50014!
1430 CLEAZE= 50033!:TASTATUR=50058!
1440 '
1450 ' ### Meldung, falls Drucker nicht bereit ###
1460 '
1470 WIDTH 80:COLOR 7,0,2
1480 PRINT" SCHALTEN SIE DEN ANGESCHLOSSENEN DRUCKER ";
1490 PRINT"AUF O N L I N E ....."
1500 LPRINT CHR$(27)CHR$(13)CHR$(80):LPRINT CHR$(0);
1510 COLOR 7,0,0:WIDTH 80
1520 '
1530 ' ### Programm - Initialisierung ###
1540 '
1550 OUT 80,11:OUT 81,8:POKE &HE462,84
1560 CALL BLANKO:FLAG=0:POKE 49900!,FLAG
1570 POKE 49902!,244:POKE 49901!,238
1580 OUT 80,1:OUT 81,84:OUT 80,2:OUT 81,102:OUT 80,6
1590 OUT 81,24: OUT 80,10: OUT 81,32
1600 ZEILE1= 0: SPALTE1= 0: SPALTE2= 0
1610 SCHRIFT= 86: LRAND= 0: TEILORT= 326: ABSTAND= 11
1620 LPRINT CHR$(27)CHR$(98)CHR$(27)CHR$(100);
1630 LPRINT CHR$(27)CHR$(31)CHR$(11);
1640 '

```

```

1650 ` ### Bildschirm-Maske Printen ###
1660 `
1670 DEFINT W: TEXT$= SPACE$ (80)
1680 PRINT"-----
      ....."
1690 PRINT"! NORMAL-SCHRIFT  DOPPEL-SCHRIFT !! Modus .... (F5)
      !! Ausdruck    !!SchriftTeilung!";
1700 PRINT"!                                     !!
      !!             !!             !";
1710 PRINT"! UNTERSTREICHEN  SCHATTENSCHRFT !! DIREKT-AUSDRUCK
      !! SERIELLE    !! 10 12 15 !";
1720 PRINT"!                                     !!
      !!             !!             !";
1730 PRINT"! TRD 7020 Schriften ..... (F2) !! ZEILEN-SPEICHER
      !! PARALLEL    !!      (F4)  !";
1740 PRINT"-----
      ....."
1750 PRINT"      !!                                     !!
      !!             !!             ";
1760 PRINT"-----!!-----!!-----!!-----
      ....."
1770 PRINT"! <--> ~~~~ Zeilen ! ! Spalte ~~~~ LRand ! !
Reset ! ! ! GROB ! ! ! ";
1780 PRINT"! ~~~~ ! ! ~~~~ ! !
      ~~~~ ! ! ~~~~ ! ! ";
1790 PRINT"! 2468 !!! ! ! ! ! ! ! !
mit ~..! ! FEIN ~..! ! ! ! ! !
1800 PRINT" ~~~~!! ~~~~ ~~~~ ~~~~
      ~~~~!! ~~~~ ! ! ";
1810 PRINT" (CTRL) !! (F1)                                     !! (TAB)

(ESC)    !! (F6)    !! ";
1820 PRINT"-----
      ....."
1830 PRINT"! .
      !";
1840 PRINT"-----
      ....."
1850 PRINT"Zeile : LÖSCHEN = (F3)          DRUCKEN = (=)          BACKSPA
CE = (DEL)      ENDE = SHIFT+(0) ";
1860 `
1870 ` ### Daten für Ort und Länge der Textstellen ###
1880 `
1890 DATA 14,86,4,449,15,289,4,132,8,309,4,326,4
1900 DATA 494,4,760,6,772,6,784,5,798
1910 DATA 5,809,4,824,6,1095,4,1109,5,1134,5,1145,4

```

```
1920 DATA 1160,80,1262,7,1437,7,1456,9,1474,4,1496
1930 '
1940 '   ### Stellen mittels Z-80-Routine hervorheben   ###
1950 '
1960 FOR T= 1 TO 23: READ ANZAHL, WERT
1970 GOSUB 3160: NEXT T
1980 A= INP(16): IF (A AND 32)= 32 THEN 2030
1990 ANZAHL= 8: WERT= 309: GOSUB 3210: WERT= 477: GOSUB 3160
2000 '
2010 '   ### Haupt - Teil   ###
2020 '
2030 LOCATE 42,11: PRINT LRAND" ";
2040 LOCATE 17,11: PRINT ZEILE1" ";
2050 LOCATE 27,11: PRINT SPALTE1" ";
2060 LOCATE 31,11: PRINT"." SPALTE2;
2070 CALL TASTATUR
2080 IF PEEK(49908)= 0 THEN SPALTE1= SPALTE1+1: GOTO 2050
2090 ON PEEK(49908) GOSUB 2110,2180,2250,2320,2370,2420,2490,
2540,2610,2670,2770,2830,2890,2980,3030,3070
2100 GOTO 2030
2110 '
2120 '   ### Cursor-Wahl durch CTRL   ###
2130 '
2140 ANZAHL= 4: WERT= 760: A= FLAG AND 1
2150 IF A THEN FLAG= FLAG-1: GOSUB 3160: WERT= WERT+168:
GOSUB 3210 ELSE FLAG= FLAG+1: GOSUB 3210:
WERT= WERT+168: GOSUB 3160
2160 POKE 49900, FLAG
2170 RETURN
2180 '
2190 '   ### Move UP mit Pfeil oder (8)   ###
2200 '
2210 IF (FLAG AND 4)= 4 THEN GOSUB 3390: GOTO 2230
2220 IF ZEILE1 > 0 THEN ZEILE1= ZEILE1-1
2230 LPRINT CHR$(27)CHR$(10);
2240 GOSUB 3430: RETURN
2250 '
2260 '   ### Move DOWN mit Pfeil oder (2)   ###
2270 '
2280 IF (FLAG AND 4)= 4 THEN GOSUB 3390: GOTO 2300
2290 IF ZEILE1 < 999 THEN ZEILE1= ZEILE1+1
2300 LPRINT CHR$(10);
2310 GOSUB 3430: RETURN
2320 '
2330 '   ### Move LEFT mit Pfeil oder (4)   ###
2340 '
2350 GOSUB 3470: B= 8: GOSUB 3520
```

```
2360 RETURN
2370 '
2380 ' ### Move RIGHT mit Pfeil oder (6) ###
2390 '
2400 GOSUB 3470: B= 32: GOSUB 3610
2410 RETURN
2420 '
2430 ' ### CR und Line Feed ( Drucker ) ###
2440 '
2450 LPRINT CHR$(13);: SPALTE1= LRAND: SPALTE2= 0
2460 IF ZEILE1 < 999 THEN ZEILE1= ZEILE1+1
2470 LPRINT CHR$(10);: GOSUB 2750: GOSUB 3350
2480 RETURN
2490 '
2500 ' ### Nullsteller ###
2510 '
2520 ZEILE1= 0
2530 RETURN
2540 '
2550 ' ### Schrifttyp wählen ###
2560 '
2570 ANZAHL= 14: WERT= SCHRIFT: GOSUB 3210: GOSUB 3330
2580 IF SCHRIFT= 254 THEN SCHRIFT= 103 ELSE IF SCHRIFT= 271
    THEN SCHRIFT= 86 ELSE SCHRIFT= SCHRIFT+168
2590 WERT= SCHRIFT: GOSUB 3160
2600 RETURN
2610 '
2620 ' ### Zeile löschen ###
2630 '
2640 CALL CLEAZE: POKE 49902,244: POKE 49901,238
2650 POKE 62702,46: POKE 62782,32
2660 RETURN
2670 '
2680 ' ### Schriftteilung ###
2690 '
2700 ANZAHL= 4: WERT= TEILORT: GOSUB 3210
2710 IF TEILORT= 330 THEN TEILORT= 322: ABSTAND= 13: GOTO 2740
2720 TEILORT= TEILORT+4: ABSTAND= ABSTAND-2
2730 IF ABSTAND= 9 THEN ABSTAND= 8
2740 WERT=TEILORT:GOSUB 3160
2750 LPRINT CHR$(27)CHR$(31)CHR$(ABSTAND);
2760 RETURN
2770 '
2780 ' ### Schreib-Modus wählen ###
2790 '
2800 ANZAHL=15:WERT=289:A=FLAG AND 2
```



```
2810 IF A=2 THEN FLAG=FLAG-2:GOSUB 3160:WERT=WERT+168:
      GOSUB 3210 ELSE FLAG=FLAG+2:GOSUB 3210
      WERT=WERT+168:GOSUB 3160
2820 POKE 49900!,FLAG:GOSUB 2610:RETURN
2830 '
2840 ' ### Grob / Fein wählen ###
2850 '
2860 ANZAHL= 4: WERT= 824: A= FLAG AND 4
2870 IF A= 4 THEN FLAG= FLAG-4: GOSUB 3160: WERT= WERT+168:
      GOSUB 3210 ELSE FLAG= FLAG+4: GOSUB 3210:
      WERT= WERT+168: GOSUB 3160
2880 POKE 49900, FLAG: RETURN
2890 '
2900 ' ### Reset auslösen ###
2910 '
2920 LPRINT CHR$(27)CHR$(13)CHR$(80);
2930 WERT= SCHRIFT: ANZAHL= 14: GOSUB 3210: WERT= 86:
      SCHRIFT= WERT: GOSUB 3160
2940 LRAND= 0: SPALTE1= 0: SPALTE2= 0
2950 ANZAHL= 4: WERT= TEILORT: GOSUB 3210
2960 TEILORT= 322: ABSTAND= 13: GOTO 2720
2970 RETURN
2980 '
2990 ' ### Linken Rand setzen ###
3000 '
3010 LRAND= SPALTE1: LPRINT CHR$(27)CHR$(57);
3020 RETURN
3030 '
3040 ' ### BREAK ###
3050 '
3060 WIDTH 80: END
3070 '
3080 ' ### Zeile drucken ###
3090 '
3100 A= VARPTR (TEXT$): POKE A+2, 244: POKE A+1, 238
3110 A= PEEK (49901)+ 256 * PEEK (49902): POKE A, 32
3120 LPRINT LEFT$( TEXT$, A-62702 );: GOSUB 2610: RETURN
3130 '
3140 ' ### Textstellen INVERTIEREN ###
3150 '
3160 POKE 50028, 254: POKE 50023, ANZAHL: CALL DIVERSE (WERT)
3170 RETURN
3180 '
3190 ' ### Textstellen DE-INVERTIEREN ###
3200 '
3210 POKE 50028, 190: POKE 50023, ANZAHL: CALL DIVERSE (WERT)
3220 RETURN
```

```
3230 `
3240 `   ###   Data's für Schrift-Umschaltung   ###
3250 `
3260 DATA 86,0,0,27,69
3270 DATA 254,27,82,27,79
3280 DATA 103,27,38,27,87
3290 DATA 271,27,38,0,0
3300 `
3310 `   ###   Schrifttype-Umschalten   ###
3320 `
3330 RESTORE 3260
3340 READ A1,A2,A3,A4,A5: IF A1<>SCHRIFT THEN 3340
3350 LPRINT CHR$(A2)CHR$(A3)CHR$(A4)CHR$(A5);:RETURN
3360 `
3370 `   ###   Graphik-Modus an   ###
3380 `
3390 LPRINT CHR$(27)CHR$(51);: RETURN
3400 `
3410 `   ###   Graphik-Modus aus   ###
3420 `
3430 LPRINT CHR$(27)CHR$(52);: RETURN
3440 `
3450 `   ###   Horizontal-Bewegung ermitteln   ###
3460 `
3470 IF (FLAG AND 4)<>4 OR ABSTAND<>11 THEN A=0 : RETURN
3480 A=1: LPRINT CHR$(27)CHR$(31)CHR$(2);: RETURN
3490 `
3500 `   ###   Spalten-Zähler setzen (LINKS)   ###
3510 `
3520 IF SPALTE1= 0 AND SPALTE2= 0 THEN RETURN
3530 IF SPALTE1>0 AND A=0 THEN SPALTE1= SPALTE1-1: GOTO 3570
3540 IF A=0 THEN RETURN ELSE SPALTE2= SPALTE2-1
3550 IF SPALTE2<0 AND SPALTE1= 0 THEN RETURN
3560 IF SPALTE2<0 THEN SPALTE1= SPALTE1-1: SPALTE2= 9
3570 LPRINT CHR$(8)CHR$(27)CHR$(31)CHR$(11);: RETURN

3580 `
3590 `   ###   Spalten-Zähler setzen (RECHTS)   ###
3600 `
3610 IF SPALTE1>135 THEN RETURN
3620 IF A=0 THEN SPALTE1= SPALTE1+1: GOTO 3570
3630 SPALTE2= SPALTE2+1
3640 IF SPALTE2>9 THEN SPALTE1= SPALTE1+1: SPALTE2= 0
3650 GOTO 3570
```

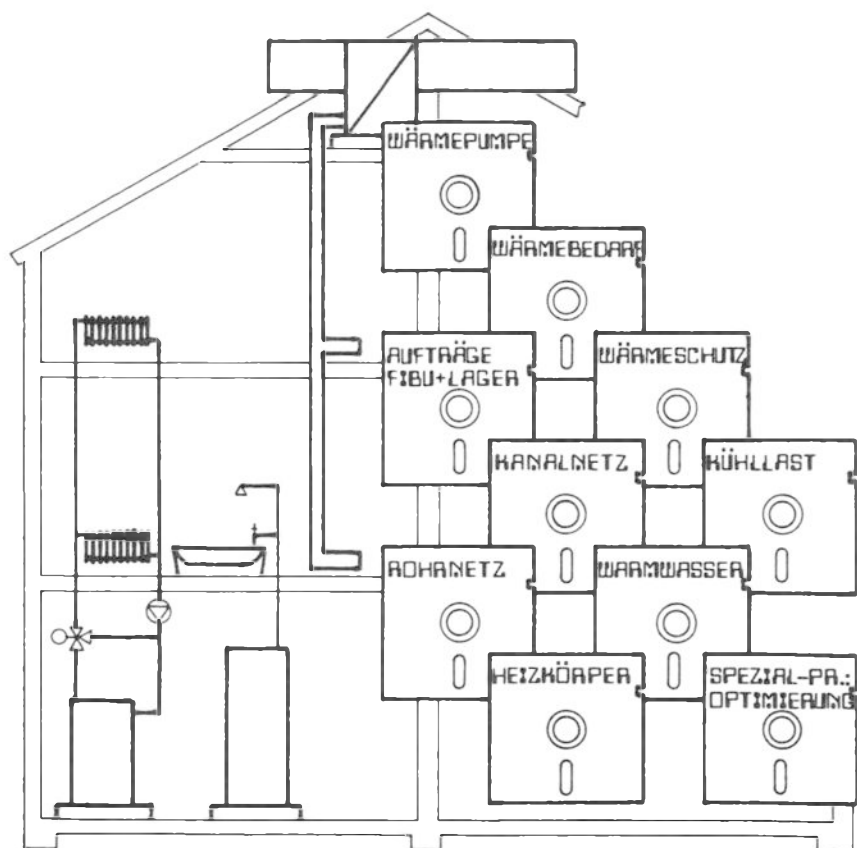
Software für Gebäudetechnik

ZUVERLÄSSIG-BEDIENUNGSFREUNDLICH

UNIVERSELL EINSETZBAR-PREISWERT



MASTERSOFTWARE®



ANFRAGEN :  MASTERSOFTWARE

LINDENSTRASSE 26 • 6148 HEPPENHEIM

Programmier-Herausforderung ECHTZEITUHR

Mikros der gehobenen Preisklasse verfügen häufig schon in der Grundausstattung über eine integrierte, batteriegepufferte Uhr, auf die sich programmtechnisch zugreifen läßt und die sich für eine Reihe von Anwendungen, besonders im technisch-wissenschaftlichen Bereich häufig als nützlich, manchmal als unentbehrlich erweist.

Bei Personal Computern im unteren Preisbereich, wie dem Alphatronic PC, die über ein solches "Hardware-Feature" nicht verfügen, ist es deshalb interessant, eine Echtzeituhr softwaretechnisch zu realisieren. Dies setzt allerdings nicht nur fortgeschrittene Kenntnisse der Maschinensprache voraus, sondern auch Experimente mit den Hardware-Gegebenheiten, wie z.B. dem Interrupt-Controller. Letztlich bleiben solche auf Interrupt-Routinen beruhende Experimente auch nicht problemlos, d.h., Konflikte mit ebenfalls auf hardware-nahen Ebenen angelegten und Interrupts nutzenden Programmen sind nicht ausgeschlossen. Zeitkritische Schnittstellenprogramme z.B. können teilweise durch die Interrupt-Steuerung der mitlaufenden Echtzeituhr gestört werden.

Diese Echtzeituhr, auf die wir bei der Suche nach "Assembler-Experimentierprogrammen", die auf unseren Einführungskurs aufbauen könnten, stießen, erhebt deshalb auch keinen Anspruch auf Perfektion. Dies wird auch ausdrücklich von der Software-Support-Abteilung der TA-AG betont, die uns dieses Programm freundlicherweise zur Verfügung stellte.

Das Programm wurde auch nicht aus der Absicht kommerzieller Vermarktung heraus entwickelt, sondern im "TA-Support" gewissermaßen als "Lockerungsübung" geschrieben, um einen besseren Überblick über die Arbeitsweise des Interrupt-Controllers des Alphatronic PC zu gewinnen und sich mit den notwendigen Routinen vertraut zu machen.

TA hat diese Echtzeituhr auch bereits dem Alphatronic PC-User-Klub in Barsinghausen zur Verfügung gestellt, wo es sofort zu einem "Renner" unter den experimentierfreudigen Assembler-Programmierern wurde. Nachdem viele tausend PC-Anwender keine Verbindung zu User-Clubs pflegen, gleichwohl das Interesse an der Vertiefung von Maschinensprache-Kenntnissen und Tricks besteht, sind wir sicher, daß die Echtzeituhr auch den Lesern dieses Buches neue Anstöße zur Kreativität und Weiterentwicklung des Programmes bietet.

Echtzeituhr

Johann Rödel-Krainz

Das hier vorgestellte Programm stellt eine Echtzeituhr dar, die weder den freien Anwenderspeicher einschränkt, noch zu großen zeitlichen Belastungen führt. Das Programm besteht im wesentlichen aus vier Teilen:

- 1) Der erste Teil besteht aus einer gewöhnlichen Laderoutine, welche die Interrupt- und Anzeigeroutine an die im Speicher vorgesehenen Stellen schiebt und die Datenfelder vorbelegt.
- 2) Anschließend wird der Interrupt-Controller initialisiert und die Sprungtabelle aktualisiert. Das Programm ist damit beendet, der Interrupt-Controller unterbricht alle 20 Millisekunden die Verarbeitung eines laufenden Programmes. Der Interrupt wird dann in der Interrupt-Routine bearbeitet.
- 3) Die erste der beiden Routinen ist die Interrupt-Routine, die nichts anderes zu tun hat, als einen Interrupt-Zähler sowie Sekunden-, Minuten- und Stundenzähler zu erhöhen. Bei jedem Aufruf, der eine Änderung der angezeigten Zeit bewirkt, wird die Anzeigeroutine aufgerufen. Die Interrupt-Routine bleibt bis zum Ausschalten des Gerätes, bzw. einem Reset im Speicher und in den oberen 48 Byte des Bildwiederholungspeichers.
- 4) Die zweite Routine dient der Auswertung und Anzeige der Zähler. Die in DCB-Darstellung vorhandenen Zeitwerte werden konvertiert und an eine andere Stelle gespeichert. Die Anzeigeroutine liegt im oberen Ende des Attributspeichers und bleibt ebenfalls bis zum nächsten Reset erhalten.

Die wesentlichen Eigenschaften der Echtzeituhr bestehen in den folgenden Punkten:

- Nach dem Start des Programmes kann unter CP/M ganz normal gearbeitet werden. Es besteht keine Einschränkung in Bezug auf den freien Anwenderspeicher (TPA). Im Normalfall startet die Uhr kurz mit der Zeit 00.00.00.
- Die Uhrzeit kann jederzeit auf einfache Weise geändert werden (<= 3 Pokes).
- Die automatische Anzeige der Uhrzeit kann unterdrückt werden (1 Poke).

- Das Ziel der Anzeige kann verändert werden (2 Pokes).
- Die Anzeige der Uhrzeit kann von einem Programm aus angefordert werden (1 Call, 2 Pokes).
- Es ist ein Stundengong eingebaut, der auch abgeschaltet werden kann (1 Poke).

Wie jedes andere Programm, unterliegt auch dieses einigen Einschränkungen, die jedoch alle, soweit bekannt, auf verschiedene Monitorstände des PC und auf das Sichern der Register zurückzuführen sind:

- In der abgedruckten Version wird der alternative Registersatz des Z-80-Prozessors von der Interruptroutine benutzt. Verwendet auch ein Anwenderprogramm diesen Registersatz, gibt's Probleme, die sich aber umgehen lassen. Das Sichern der Register des laufenden Anwenderprogrammes kann auch in der Interruptroutine durch die Befehle PUSH und POP erfolgen. Diese Art der Registersicherung benötigt jedoch im Fall der Interruptroutine 2 Bytes mehr als die Verwendung des alternativen Registersatzes, so daß man in diesem Fall auf eine der Optionen, automatische Anzeige oder Stundengong, verzichten muß. Von den zur Verfügung stehenden 48 Bytes stehen dann je nach Reduzierung der Optionen zwischen einem und vier Byte für sinnvolle Anwendungen zur Verfügung.
- Zum Zwecke der Sicherung von Registern wird von beiden Routinen ein Stack strapaziert, der nicht von den Routinen selbst eingerichtet und verwaltet wird. Es wird erwartet, daß zu jeder Zeit genügend Platz auf dem Stack zur Verfügung steht. Es werden benötigt:
 - in der Anzeigeroutine:
8 Bytes für die Sicherung von vier Registern (Standard)
 - in der Interruptroutine:
8 Bytes von dem Stundengong (Aufruf + Piepser-Routine, auf die auch verzichtet werden)
6 Bytes für die Sicherung von drei Registerpaaren, falls der Stack benutzt wird.
2 Bytes für den Aufruf der Anzeigeroutine, nicht änderbar.

In der vorliegenden Version des Programms werden also 10 Bytes Stack verwendet. Der minimale Verbrauch an Stackbereich liegt bei 0, wobei der Aufruf der Anzeigeroutine nicht stattfinden

darf (bitte nicht von BASIC aus). Der maximale Bedarf liegt bei 16 Byte (10 Byte für die Anzeigeroutine, 6 für die Registersicherung - der Gong läuft nicht innerhalb der Anzeigeroutine).

- Die Piepserroutine für den Stundengong liegt im alten Monitor an einer anderen Adresse: EAAB. Wird diese Adresse nicht geändert, läuft die Uhr unter dem alten Monitor höchstens eine Stunde richtig.

Die wichtigsten der bereits angesprochenen Änderungen werden jetzt, soweit möglich, anhand von BASIC-Beispielen erklärt.

- automatische Anzeige ausschalten:
 - Der NOP (&H00) an der Stelle &HhFFD0) muß in einen RETURN geändert werden: POKE &HFFD0, &HC9
- Stundengong ausschalten:
 - Der bedingte CALL-Aufruf (&HCD) an der Stelle &HF7EE muß in einen anderen geändert werden, wobei die Bedingung nicht erfüllt ist. Ein Überschreiben durch NOP's ist nicht zu empfehlen, da dabei zwei Bytes geändert werden müssen. Vor der Änderung des zweiten Bytes kann ein Interrupt unabsehbare Folgen haben: POKE &HF7EE, &HD4
- setzen der Uhrzeit:

Die Zeitangaben sind in den Datenfeldern in Binary Coded Dezimal-Darstellung gespeichert. Eine Änderung wird am besten durch ein Beispiel erklärt. Es soll die Uhrzeit auf 5:32:16 gesetzt werden:

POKE &HFFFE,&H05: POKE &HFFFD,&H32: POKE &HFFFC,&H16
- Anzeige von einem Programm aufrufen:

Soll die Anzeige aus einem Programm heraus aufgerufen werden, sind mehrere Schritte notwendig:

 - a) zuerst muß die automatische Anzeige der Uhrzeit unterdrückt werden. (siehe oben)
 - b) in der Anzeigeroutine sind zwei Bytes freigehalten, die für Interrupt-disable bzw. enable vorgesehen sind. Diese Interrupt-Befehle dürfen in der Standardversion (automatische Anzeige) nicht in der Anzeigeroutine vorhanden sein. Diese Befehle werden wie folgt gepatched:

POKE &HFFD5,&HF3: POKE &HFFD4,&HFB
 - c) die Anzeige kann dann im Programm mit einem

einfachen CALL-Aufruf angefordert werden:
CALL &HFFD1

- Position des Ausgabefeldes für die Uhrzeit ändern:
Für die Anzeigeroutine ist die Adresse des Bytes hinter dem Anzeigefeld von Interesse. Im folgenden Beispiel soll die Anzeige in der 24sten Zeile ab Spalte 70 erfolgen. Die zu patchende Adresse ergibt sich aus: $ADR = \&HF000 + 24 \cdot 80 + 70 + 9$, wobei &Hf000 die Anfangsadresse des Bildwiederholungspeichers ist. Die so erhaltene Adresse muß noch konvertiert werden:
 $ADRHIGH = INT(ADR/256)$; $ADRLow = ADR - ADRHIGH \cdot 256$, bevor sie in der Anzeigeroutine geändert werden kann:
POKE &HFFDA,ADRLow: POKE &HFFDB,ADRHIGH
- laden der Uhrzeit in eine Stringvariable:
10 REM Uhrzeit in eine Variable laden
20 REM
30 REM Vorbesetzen der Variablen
40 DISPLADR = &HFFD0 : DISPLRET = &HC9
50 ENABLEINT = &HFB: DISABLEINT = &HF3
60 ENABLEADR = &HFFF4 : DISABLEADR = &HFFD5
70 DISPLROUT = &HFFD1
80 OUTADRL = &HFFDA : OUTADRH = &HFFDB
90 REM Variable ZEIT\$ lang genug machen
100 ZEIT\$ = " "
110 REM Adressrechnung
120 ADR = PEEK(VARPTR(ZEIT\$) + 1) +
 PEEK(VARPTR(ZEIT\$) + 2)*256 + 9
130 ADRH = INT(ADR / 256) : ADRL = ADR - ADRH * 256
140 REM automatische Anzeige unterdrücken
150 POKE DISPLADR,DISPLRET
160 REM Ausgabeadresse patchen
170 POKE OUTADRL,ADRL : POKE OUTADRH,ADRH
180 REM Interruptbefehle ändern
190 POKE ENABLEADR,ENABLEINT: POKE DISABLEADR,DISABLEINT
200 REM Alles bis hierhin wird in Zukunft nicht mehr
210 REM benötigt. Das Laden der Uhrzeit in die Variable
220 REM ZEIT\$ erfolgt mit:
230 CALL DISPLROUT
240 REM Die Variable ZEIT\$ muß beim Aufruf wie in Zeile
250 REM 100 vorbesetzt sein
260 PRINT ZEIT\$

Das Programm wurde mit dem MACRO-Assembler M80 erstellt, das Listing für den Ausdruck an dieser Stelle komprimiert. Auch wurde die Kommentierung dem Druckformat angepaßt.


```

0000      ASEG      .Z80
F048      TVRAM      equ      0f048h
           ; Adresse hinter dem Ausgabefeld
F7D0      ADRES1     EQU      0F7D0H
           ; Anfang der Interrupt Routine
FFD0      ADRES2     EQU      0FFD0H
           ; Anfang der Anzeige Routine
FFFB      ADRES3     EQU      0FFFBH
           ; Anfang des Datenbereiches
EAYE      BIEBSA     EQU      0EA7EH
           ; fuer alten Monitor: 0EAC3H !!!
0032      CVALUE     EQU      50
           ; Anzahl Interrupts pro Sekunde
0070      INTCIN     EQU      70H
           ; Portadresse Int.-Controller
0071      INTCOP     EQU      71H
           ; Fuer ICw2, ICw3, OCw1
001E      ICW1       EQU      1EH
           ; LOWBYTE-ADDRESS INT-JUMPTABLE
00EF      ICW2       EQU      0EFH
           ; HIGHBYTE-ADDRESS INT-JUMPTABLE
007F      OCW1       EQU      07FH
           ; Maske fuer Interrupt level 7
0020      OCW2       EQU      20H
           ; NOT SPECIFIED END OF INTERRUPT
EF1C      JMPTB0     EQU      0EF1CH
           ; INT.-Sprungadresse fuer intr7

ORG      100H

```

; die Routine MOVEP schiebt die Interrupt- und die Anzeigeroutine an die
; vorgesehene Stelle

```

0100          MOVEP:
0100          21 014F          LD      HL,INT1ANF
                                ; Quelladresse Interruptroutine
0103          11 F7D0          LD      DE,ADRES1
                                ; Zieladresse
0106          01 0030          LD      BC,INT1END-INT1ANF
                                ; Laenge der I.-Routine
0109          ED B0            LDIR
                                ; laden

010B          LD      HL,INT2ANF
010E          11 FF00          LD      DE,ADRES2
0111          01 002A          LD      bc,INT2END-INT2ANF
                                ; verschieben der Anzeigeroutine
0114          ED B0            LDIR
                                ; HL,DAT1ANF
0116          21 01A9          LD      DE,ADRES3
0119          11 FFFB          LD      bc,DAT1END-DAT1ANF
011C          01 0005          LD      DE,ADRES3
011F          ED B0            LDIR
                                ; und des Datenbereiches
0121          MVEND:

```

```

0121          SETIME::
0121          F3              DI      ; DISABLE CPU-INTERRUPT
0122          3E 00          XOR     A      ; A löschen
0124          21 FFFB          LD      HL,counte
0127          77              LD      (HL),A      ; daten loeschen
0128          23              INC     HL
0129          77              LD      (HL),A
012A          23              INC     HL

```

```

012B 77 LD (HL),A
012C 23 INC HL
012D 77 LD (HL),A

; Die Interrupt-Sprungtabelle muss mit geändert werden: ein JUMP (C3H),
; der Adresse der Interruptroutine und einem RETURN (C9)

012E 01 C9C3 LD BD,0C9C3H ; JUMP und RETURN
0131 21 F7D0 LD HL,INTROUT ; Adr. Int.-routine
0134 ED 43 EF1 LD (JMPTB0),bc ; JUMP laden
0138 ED 43 EF1 LD (JMPTB0+2),bc ; RETURN laden
013C 22 EF1D LD (JMPTB0+1),HL ; Adresse laden

; Initialisierung des INTERRUPT-Controllers
013F 3E 1E LD A,ICW1
0141 D3 70 OUT (INTCIN),A
0143 3E EF LD A,ICW2
0145 D3 71 OUT (INTCOP),A
0147 3E 7F LD A,OCW1
0149 D3 71 OUT (INTCOP),A

014B ED 46 IM 0 ; INTERRUPT MODE 0
014D FB EI
014E C9 RET

014F INT1ANF: .PHASE ADRES1

```

```

; Interruptroutine

F7D0      F3      INTROUT:
F7D0      08      DI      ; keine Interrupts mehr
F7D1      D9      EX      ; alternativen Registersatz
F7D2      01      EXX     ; verwenden
F7D3      0350H   LD      BC,0350H ; B: Zähler für
                                ; Interrupt, Sekunde, Minute
                                ; C: Interruptfaktor
                                ; ADR. Interruptzähler

F7D6      21      LD      HL,COUNT
F7D9      7E      loop1:
F7D9      C6 01   LD      A,(HL) ; lesen
F7DA      27      ADD     A,1      ; erhöhen
F7DC      77      DAA      ; dezimal konvertieren
F7DD      91      LD      (HL),A   ; speichern
F7DE      13      SUB     C        ; gleich 50, 60 oder 24
                                ; Int. Sek., Min, Stunde
                                ; nein, Rücksprung

F7DF      20 13   JR      NZ,EXIT
F7E1      37      SCF

; falls der aktuelle Zähler den Grenzwert erreicht hat, muß er zurückgesetzt
; und der nächste Zähler um 1 erhöht werden usw.

F7E2      77      LD      (HL),A   ; Zähler rücksetzen
F7E3      0E 60   LD      C,60H    ; fuer Sekunden, Minuten
F7E5      23      INC     HL        ; nächster Zähler
F7E6      C8 7E   BIT     7,(HL)   ; am Ende der Tabelle??
F7E8      20 0A   JR      NZ,EXIT  ; aufhören
F7EA      10 ED   DJNZ    LOOP1     ; nächster Vergleich
F7EC      0E 24   LD      C,24h    ; Stundenwechsel
F7EE      CD EA7E CALL    BIEBSA    ; Stundengang
F7F1      37      SCF              ; wegen eines OR A im BIEBSA
F7F2      18 E5   JR      LOOP1    ; Stunden !!!

```



```

FFD6 21 FFFC      LD      HL,SECOND      ; zuerst Sekunden
FFD9 11 F049      LD      DE,IVRAM + 1 ; ein Byte hinter der
                                ; letzten Stelle der Ausgabe
FFDC 06 03        LD      B,03H        ; Zähler Sek, Min, Std

; in dieser Schleife findet die Konvertierung von BCD-Format auf ASCII
; statt, der RRD muß dreimal ausgeführt werden, um den ursprünglichen Stand
; zu erhalten
loop5:
FFDE 1B           DEC      DE           ; Ausgabeadresse
FFDF 3E 30        LD      A,30H        ; A vorbesetzen
FFE1 ED 67        RRD              ; den niederen Nibble drauf
FFE3 12           LD      (DE),A      ; Ausgabe
FFE4 1B           DEC      DE         ; nächste Stelle
FFE5 ED 67        RRD              ; original der hohe Nibble
FFE7 12           LD      (DE),A      ;
FFE8 1B           DEC      DE         ;
FFE9 ED 67        RRD              ; alles wie es war
FFEB 3E 3A        LD      A,3AH       ; Ausgabe :
FFED 12           LD      (DE),A      ;
FFEE 23           INC      HL         ; nächster Wert
FFEF 10 ED        DJNZ     LOOP5      ;
FFF1 3E 20        LD      A,20H      ; ein Leerzeichen
FFF3 12           LD      (DE),A      ;
FFF4 00           NOP                ; Platz fuer den "EI", falls von
                                ; Programm aufgerufen wird
FFF5 E1           POP      HL         ; alte Register wieder
FFF6 D1           POP      DE         ; herholen
FFF7 C1           POP      BC
FFF8 F1           POP      AF
FFF9 C9           RET                ; Rücksprung

```

```

01A9                                .DEPHASE
INT2END:

; Und ganz zum Schluß der Datenbereich

01A9 DAT1ANF:
        .PHASE OFFFBH
        ; Datenfelder
COUNT: db 00H
SECOND: db 00H
        ; Interruptzähler
        ; Sekunden,
        ; Minuten und
        ; Stunden
        db 00H
        db 0FFH

01AE                                .DEPHASE
dat1END:

```

Macros:

Symbols:

ADRES1	F7D0	ADRES3	FFFB	BIEBSA	EA7E
COUNTE	FFFB	DAT1AN	01A9	DAT1EN	01AE
DISPL	FFD0	ICW1	001E	ICW2	00EF
INT1AN	014F	INT2AN	017F	INT2EN	01A9
INTCIN	0070	INTR0U	F7D0	JMPTB0	EF1C
LOOP1	F7D9	MOVEP	0100	MVEND	0121
OCW1	007F	SECOND	FFFC	SETIME	0121I
TVRAM	F048				

Programmbeschreibung

Mit dem hier beschriebenen Terminalprogramm wird Ihr Alphatronic PC zu einem Terminal mit der Emulation VT-52 von digital equipment corporation. Diese Terminalemulation ist zu einem weit verbreiteten Standard geworden, der auch in der Microcomputerfamilie von TA teilweise zur Anwendung kommt. Ein Teil der Bildschirmsteuerzeichensequenzen, die zur Programmierung des PC verwendet werden, sind dieser Emulation entnommen. Das vorliegende Programm realisiert nahezu alle Funktionen des VT-52 mit Ausnahme der Semigrafikzeichen, die jedoch jederzeit eingefügt werden könnten.

Aufgabe des Programms ist es, jedes Zeichen, das über die serielle Schnittstelle empfangen wird, auf dem Bildschirm des PC darzustellen und jedes Zeichen, das über die Tastatur eingegeben wird, über die serielle Schnittstelle zu senden.

Das Terminalprogramm besteht im Wesentlichen aus drei Teilen. Dies sind

- die Interruptserviceroutine zum Empfang der Zeichen,
- der Teil zur Interpretation und Bearbeitung der Steuerzeichen
- sowie die Bildschirmroutinen zur Darstellung der Zeichen.

Für den Empfang der Zeichen über die serielle Schnittstelle wird eine Interruptserviceroutine verwendet. Der programmierbare Interrupt Controller (PIC) des PC wird so initialisiert, daß er jeweils für ein ankommendes Zeichen einen Interrupt auslöst. Die Serviceroutine holt das Zeichen aus dem USART ab und schreibt es in einen Ringpuffer, der im vorliegenden Fall 2 KByte groß ist. Dadurch kann mit höheren Übertragungsgeschwindigkeiten gearbeitet werden, ohne daß ein Überlauf entsteht, auch wenn die Gegenseite nicht über einen Hardwarehandshake verfügt (z.B. Modem). Sollte ein Softwarehandshake (XON/XOFF) benötigt werden, so läßt sich dies sehr leicht in das Programm einfügen.

Ein weiterer Vorteil ergibt sich durch die Verwendung einer Interruptroutine. Dadurch, daß der PC nicht über einen Tastaturpuffer verfügt, ist es schwierig, zwei Eingabequellen, dies sind die Tastatur und die serielle Schnittstelle, so häufig abzufragen, daß bei keiner von beiden ein Überlauf entsteht, ohne

jedesmal beim Abfragen der Tastatur das Signal RTS der seriellen Schnittstelle zu deaktivieren. Durch die Verwendung der Interruptroutine kann nun jederzeit die Tastatur abgefragt werden und dennoch können Zeichen von der seriellen Schnittstelle entgegengenommen werden. Durch diese Vorgehensweise ist es gelungen, einen Überlauf der Tastatur weitestgehend zu vermeiden.

Der Vollständigkeit halber sollte erwähnt werden, daß auch die Tastatur über Interrupt abgefragt werden könnte. Dies bedarf jedoch einigen Aufwands und führt uns über den Rahmen dieses Beitrags weit hinaus.

Jedes Zeichen, das über die Schnittstelle empfangen wird, wird vom Programm untersucht, ob es zu einer Steuerzeichensequenz gehört oder nicht. Ist eine Steuerzeichensequenz erkannt, wird diese bearbeitet und auf das nächste Zeichen gewartet. Gehört ein Zeichen nicht zu einer Steuerzeichensequenz wird es auf dem Bildschirm ausgegeben.

Die Programmteile, die die unterschiedlichen Bildschirmfunktionen realisieren, sollen hier nicht im Einzelnen beschrieben werden, sondern können der nachfolgenden Programmliste entnommen werden.

Zu den verschiedenen Betriebsarten des Programms und zur Bedienung des Bildschirms jedoch noch einige Erläuterungen.

Das Programm kann im ON-LINE und im LOCAL Modus arbeiten. Im ON-LINE Modus werden alle Zeichen, die über die Tastatur eingegeben wurden, über die serielle Schnittstelle gesendet. Im LOCAL Modus werden diese Zeichen nicht über die Schnittstelle gesendet, sondern unmittelbar auf dem Bildschirm dargestellt. In dieser Betriebsart werden alle Funktionstasten verarbeitet und es entsteht ein einfacher Bildschirmeditor, der als Grundlage zur Realisierung eines Block Mode Terminals verwendet werden kann.

Eine hilfreiche Testfunktion stellt der sogenannte TRANSPARENT Modus dar. In dieser Betriebsart werden nicht die empfangenen Zeichen als ASCII-Zeichen dargestellt, sondern das jeweilige Hexadezimaläquivalent wird als zwei ASCII Ziffern auf dem Bildschirm dargestellt. Sowohl im ON-LINE als auch im LOCAL Modus erfolgt die transparente Darstellung der Zeichen, die von der Tastatur eingegeben werden, in REVERSE Darstellung (unterlegt), um sie von den empfangenen Zeichen zu unterscheiden.

Eine weitere Funktionstaste erlaubt das Ein- bzw. Ausschalten

des automatischen LINE WRAP (Übergang auf die neue Zeile am Zeilenende).

Im PAGE Modus wird bei Überschreiten des unteren Bildschirmrandes der Bildschirm gelöscht und die Ausgabe der nachfolgenden Zeichen erfolgt wieder ab der HOME Position (links oben). Der PAGE Modus wird ebenfalls mit einer Funktionstaste ein- bzw. ausgeschaltet.

Eine weitere Sondertaste ist die NOSCROLL Taste, die die Zeichenausgabe auf den Bildschirm anhält bzw. freigibt.

Als Funktionstasten zum Ein-/Ausschalten der hier beschriebenen Betriebsarten, werden die drei Tasten oberhalb des Ziffernblocks verwendet, die zu Beginn des Programms mit neuen Codes belegt werden, so daß sie von den Tasten der Haupttastatur, die den gleichen Funktionscode liefern, zu unterscheiden sind. Die genaue Aufteilung der Tasten entnehme der Leser der Programmlieste.

Nun zur Bildschirmverwaltung. Das Terminalprogramm verwendet einen eigenen, internen Bildspeicher. Dies wurde so gewählt, um einen größeren Bildausschnitt einfach zu verwalten. Der größere Bildausschnitt dient dazu, bereits aus dem darstellbaren Bereich hinausgerollte Informationen, im LOCAL Modus zurückzuholen. Weiterhin braucht dadurch beim Verschieben des Bildspeichers nicht auf die Bildschirmsynchronisation geachtet werden. Dies ist erforderlich, um eine flimmerfreie Ein-/Ausgabe auf dem Bildschirm zu realisieren.

Je nach Funktion werden die Bildmanipulationen erst im internen Bildpuffer vorgenommen und dieser wird danach neu in das eigentliche VIDEO RAM (sichtbarer Bildspeicher) übertragen oder bei schnell ablaufenden Funktionen werden die Einzeloperationen sowohl mit dem internen als auch mit dem VIDEO RAM ausgeführt.

Bei jedem Zugriff auf das VIDEO RAM muß darauf geachtet werden, daß Schreib- oder Leseoperationen nur dann ausgeführt werden, wenn sich der Schreibstrahl der Bildröhre in einer Rücklaufphase befindet, um ein flimmerfreies Bild zu erhalten. Die dazu nötige Information liefert der Baustein 6845 (Videocontroller) und diese steht über ein SYSTEPORT abrufbar zur Verfügung. Man beachte, daß durch einen Interrupt an einigen Stellen des Programms das Zeitverhalten der Routinen beeinflußt werden könnte. Dieser Fall tritt jedoch so selten ein, daß es nicht gerechtfertigt ist, die Ausgabegeschwindigkeit zu Gunsten der Bildqualität weiter herabzusetzen.

Der Vollständigkeit halber muß auch hier gesagt werden, daß es für die Bedienung des VIDEO RAMs, speziell zum SCROLL des Bildschirms, Alternativen gibt. Man kann dem 6845 mitteilen, ab welcher Adresse er mit der Bilddarstellung beginnen soll, so daß sich ein sehr schneller SCROLL realisieren ließe, dies hat jedoch zur Folge, daß alle Routinen, die Zeichen innerhalb des VIDEO RAMs verschieben, über relative Zeiger arbeiten müssen und damit wesentlich aufwendiger werden. Aus Gründen der Übersichtlichkeit und ohne wesentliche Geschwindigkeitseinbußen - wovon man sich leicht überzeugen kann - wurde der vorliegende Weg gewählt.

Der interne Bildspeicher wurde nur für Textcodes wesentlich vergrößert. Der interne Attributspeicher entspricht in seiner Größe dem darstellbaren Bereich, so daß Attributinformationen beim SCROLL über die Bildschirmgrenzen hinaus verlorengehen. Eine Erweiterung des Attributspeichers ist jederzeit möglich, da noch genügend Speicherplatz zur Verfügung steht.

Da dieses Programm für die Verwendung mit Akustikkopplern bestens geeignet ist, sei dem Leser empfohlen, einen Programmteil zum Schreiben empfangener Daten auf Diskette und umgekehrt selbst zu ergänzen. Bei der Verwendung von Akustikkopplern mit einer Übertragungsrate von 300 bit/sec könnte es nützlich sein, einen Tastaturpuffer nachzubilden, um einen Zeilenüberlauf an der Tastatur zu vermeiden, da diese geringe Übertragungsgeschwindigkeit dazu führen kann, daß der Bediener schneller eingibt, als die Daten gesendet werden können. Dieses Problem läßt sich durch eine Interruptroutine für die Ausgabe auf der seriellen Schnittstelle leicht lösen. Der Interruptvektor für die Ausgabe steht auf der Adresse EINGABE + 4 und die Maske zum Aktivieren des PIC für den Sendeinterrupt ergibt sich durch shiften (links) der Maske für den Empfangsinterrupt.

Viel Erfolg.

Dipl.Ing. G. Müller-Dorn

BiCom Datensysteme GmbH

Terminalprogramm zur Emulation eines VT-52 Terminals auf dem
----- TA-PC. -----

Promversion : Setze "PROM" = TRUE
Diskversion : Setze "PROM" = FALSE

(c) 1985 BiCom Datensysteme GmbH Ringelnatzstr. 22
 6073 Egelsbach 2
 Tel: 06103 /44157

Geschäftsstelle Hannover: Bultstraße 25
 3000 Hannover 1

Verfasser: Dipl.Ing. G. Müller-Dorn
 BiCom Datensysteme GmbH

Schutzbestimmungen: Dieses Programm wurde zur Dokumentation der
Funktion des alphaTronic PC zur Verfügung
gestellt. Jeder Leser kann dieses Programm
frei verwenden, wird jedoch gebeten, den
Urhebervermerk zu übernehmen, auch wenn nur
Teile aus diesem Programm entnommen werden.

Zur Programmerstellung wurden verwendet:

WORDSTAR (r) Micropro
M80 (r) Microsoft
L80 (r) Microsoft

Um ein kommentiertes Assembler-Listing im Hochformat
in eine buchgerechte Form zu bringen, müßte es der-
artig stark verkleinert werden, daß es eine Zumutung
für jedes Leserauge wäre. Wir haben uns deshalb ent-
schlossen, das Listing in der Original-Schriftgröße
zu belassen und es dafür querformatig abzdrukken.

.Z80				; Assembleranweisung fuer Z80 - Mnemonics			
F800	ARAMAD	EQU	0F800H	; ADRESSE VIDEO - RAM	ATTRIBUTE		
F000	VRAMAD	EQU	0F000H	; ADRESSE VIDEO - RAM	ZEICHEN		
E40C	CRTST	EQU	0E40CH	; CONSOLE STATUS			
E40F	CRTIN	EQU	0E40FH	; CONSOLE IN - KEYBOARD			
E412	CRTOUT	EQU	0E412H	; CONSOLE OUT			
E469	SFF1	EQU	0E469H	; SYSTEM FLAG 1			
0020	SPOPT2	EQU	20H	; SYSTEMPORT 2			
E468	SFF0	EQU	0E468H	; SYSTEM FLAG 0			
0010	SPORT	EQU	10H	; FUER ABSCHALTEN PROM			
0070	PIC70	EQU	70H	; INTERRUPTCONTROLLER AO=0			
0071	PIC71	EQU	71H	; INTERRUPTCONTROLLER AO=1			
00FD	RXMSK	EQU	0FDH	; ENABLE RECEIVER INTERRUPT			
0100	LOWLIM	EQU	0100H	; LOWLIMIT FUER PROGRAMM (PROM UND DISK)			
FFFF	TRUE	EQU	-1	; DIES IST GANZ NÜTZLICH			
0000	FALSE	EQU	0	; DIES AUCH			
0018	ZMAX	EQU	24	; MAXIMALE ZEILENZAHL			
0050	SPMAX	EQU	80	; MAXIMALE SPALTENZAHL			
0780	SCLNG	EQU	1920	; ANZAHL ZEICHEN			
0080	REVMSK	EQU	80H	; REVERSE BIT			
0040	BLKMSK	EQU	40H	; BLINK BIT			

; EINIGE NÜTZLICHE GLEICHSETZUNGEN							

0005	ENQ	EQU	5	; ENQUIRY			
0006	ACK	EQU	6	; ACKNOWLEDGE			
0015	NACK	EQU	15H	; NOT ACK			
000A	LF	EQU	10	; LINE FEED --- ZEILENVORSCHUB			
000D	CR	EQU	13	; CARRIAGE RETURN -- WAGENRÜCKLAUF			
001B	ESC	EQU	27	; ESCAPE			

```

0040  USARTD EQU 40H      ; DATENPORT DES USART
0041  USARTC EQU 41H      ; CONTROLLPORT DES USART
00F2  F6 EQU 0F2H       ; CODE FÜR FUNKTIONSTASTE F6
0080  F1 EQU 080H       ; CODE FÜR FUNKTIONSTASTE F1

FFFF  PROM           ; WENN PROM DANN TRUE
      IF           ; WENN PROMVERSION
      ORG          ; DANN MUSS AUF NULL GEBUNDEN WERDEN
      C3 A042      ; PROGRAMMSTART FUER ROM - PACK 1

0000  JP 0A042H       ; MUSS ALS ERSTES IM PROM STEHEN

0007  ORG 0007H       ; KEINER WEISS ES WIRKLICH GENAU
0008  DB 0FAH        ; WARUM DIES HIER STEHEN MUSS,
      DB 07EH        ; ABER ES MUSS SEIN, DAMIT PROM
                       ; BEARBEITET WIRD
003B  ORG 003BH       ; EINSPRUNG NACH INITIALISIERUNG
      LD HL,078H
003E  JR A00K         ; STARTADRESSE
      ORG 042H

      ; UMLADEN PROGRAMM
      ;
0042  A00K: LD HL,0A100H ; START DES PROGRAMMCODES
0045  LD DE,LOWLIM      ; ADRESSE FUER ABLAUF PROGRAMM
0048  LD BC,02000H      ; LAENGE PROGRAMM ( ZUR SICHERHEIT GANZES
                       ; PROM, DANN MUSS MAN NICHT VIEL DENKEN )
004B  LDIR              ; DER Z80 IST JA SO PRAKTISCH
004D  LD HL,0A0F0H      ; ROUTINE ZUM ABSCHALTEN DER PROMS
0050  LD DE,9FA0H       ; AUS ROM-PACK 1 AN FREIES RAM LADEN
0053  LD BC,10H         ; SIE IST NICHT ALLZU LANG

```



```

0088` C3 9FA0 JP 9FA0H ; MANCHMAL IST ES NÜTZLICH AUCH ABSOLUTE
                                ; ADRESSEN ZU VERWENDEN. ES SPART ZEIT.
00F0` 0F0H
CPNOSBU:
;*****
; ROM PACK AUSSCHALTEN
;*****
00F0` 21 E468 LD HL,SFF0
00F3` CB FE SET 7,(HL) ; ALLE ROMS AUS NOCHMAL ZUR SICHERHEIT
00F5` 7E LD A,(HL)
00F6` D3 10 OUT (SPORT),A
00F8` C3 0100 JP LOWLIM ; STARTE ROGRAMM
                                ENDIF
;*****
; INIT 8251
; Die Übertragungsgeschwindigkeit der seriellen Schnittstelle wird durch
; ein Jumper Feld auf der Printkarte eingestellt. Trotzdem muss man dem
; 8251 noch ein paar schlaue Sprüche sagen.
; Es wird empfohlen aus dem nachfolgenden Spruch kein Verhohnepipelung
; des Chips zu machen. Er könnte es krumm nehmen.
;*****
0100` 0100H ORG 0100H ; JETZT GEHT ES WIRKLICH GLEICH LOS
CPADDR: IF PROM ; HIER IST DAS EIGENTLICH ÜBERFLÜSSIG

```



```

0120 00      ; UND KOMM ICH
0121 00      ; HEUTE NICHT,
0122 00      ; DANN KOMM ICH MORGEN
0123 03 41   OUT      (USARTC),A      ; ABER WIRKLICH ENDLICH.
-----
; INTERRUPTVECTOR FUER RECEIVE NACH "EF04". DORT GEHÖRT ER HIN, AUCH WENN
; ANDERE LEUTE VON INTERRUPTEN AUF DEM PC NICHT VIEL HALTEN; ES WURDE
; VORGESEHEN.
-----
0125 11 EF04 LD      DE,0EF04H      ; ADRESSE ÄNDERT SICH HOFFENTLICH NICHT
0128 21 02E4 LD      HL,RECJMP      ; UND WIRD MIT EINEM JUMP ZUR INTERRUPT-
012B 01 0003 LD      BC,3           ; SERVICEROUTINE VERSORGT.
012E ED 80   LDIR     ; EIN JUMP IST 3 BYTE LANG
*****
; INITIALISIEREN PROGRAMMABLE INTERRUPTCONTROLLER FUER RX-INT
; RX-INT IST INTERRUPT IR1 DES 8259
*****
; INITIALISIERUNGSWORT 1
; HIGH BYTE VECTOR INITIALISIERUNGSWORT 2
; MASKE FUER RX INT ALLEINE. ER KANN AUCH
; NOCH MEHR UND WER SPASS HAT, KANN EINEN
; OUTPUT INTERRUPT VERWENDEN. ICH HATTE
; JEDENFALLS KEINE ZEIT MEHR.
; INTERRUPT OK.
; MAL NEN ANSTÄNDIGEN "
; STAPELZEIGERSPEICHERPLATZ

0130 3E 16   LD      A,16H
0132 D3 70   OUT     (PIC70),A
0134 3E EF   LD      A,0EFH
0136 D3 71   OUT     (PIC71),A
0138 3E FD   LD      A,RXMSK
013A D3 71   OUT     (PIC71),A

013C FB      EI
013D 31 47CA LD      SP,PROGST

```



```

0155      23      INC      HL      ; UND DIE RECHTE
0156      3E F2    LD       A,0F2H ; MOECHTE F2
0158      77      LD       (HL),A  ; SENDEN
0159      E1      POP      HL      ; EINE SHIFT TABELLE GIBTS AUCH
015A      11 006C LD       DE,06CH ; UND HEX RECHNEN KANN DER
015D      19      ADD      HL,DE   ; Z80 BESSER ALS ICH
015E      3E F3    LD       A,0F3H ; F3 IST DIE LOGISCHE
0160      77      LD       (HL),A  ; FORTSETZUNG VON LINKS
0161      23      INC      HL      ; UND FÜR DIE MITTE
0162      3E F4    LD       A,0F4H ; F4 BEI SHIFT
0164      77      LD       (HL),A  ; EINTRAGEN
0165      23      INC      HL      ; UND EIN TASTE (DIE RECHTE) ZUM
0166      3E F2    LD       A,0F2H ; KLOPFEN, EGAL OB NON SHIFT
0168      77      LD       (HL),A  ; ODER SHIFT
0169      CD 02B6  CALL     MESANF  ; JEZT SOLL DER MENSCH AM BILDSCHIRM
                                ; SEHEN, DASS ES UNS GIBT UND AUCH ZEIT
                                ; ZUM LESEN HABEN. UNTERPROGRAMM WARTET
                                ; FREUNDLICHERWEISE AUF ANTWORT
                                ; ANZEIGE EIGENES VIDEOAM. IST LEER.
                                *****
016C      CD 0346  CALL     VIDDIS *****
                                ;
                                ; HAUPTPROGRAMMSCHLEIFE. DIE KNOCHENARBEIT WIRD WOANDERS GEMACHT
                                ;
                                ;
                                ; *****
                                ; UND ALLE WEGE FÜHREN NACH ROM. SO AUCH ALLE TEILE DIESES PROGRAMMS
                                ; ZU DIESEM PUNKT.
                                *****
                                -----
016F      2A 1342  INCHR:  LD      HL,(NETP04) ; ZWEI ZEIGER IN EINEN RINGPUFFER
0172      ED 5B 13 LD      DE,(NETPI4) ; FÜR EIN UND RAUS
0176      B7      OR      A      ; DIFFERENZ BITTE OHNE CARRY
0177      ED 52   SBC      HL,DE  ; WENN VERSCHIEDEN NULL IST ZEICHEN DA
0179      20 1B   JR       NZ,SCREEN ; UND SOLL AUF DEN EIGENEN BILDSCHIRM

```

```

017B 3A 0345          LD      A, (RTSFLG)      ; WENN DER BUFFER VOLL WAR
017E B7              OR      A                ; WURDE HANDSCHAKE FLAGE GESETZT UND
017F 28 09          JR      Z, INSTAT         ; ES KANN NUN, DA NICHTS MEHR DRIN IST
0181 3E 00          LD      A, 0             ; ZURÜCKGESETZT WERDEN.
0183 32 0345        LD      (RTSFLG), A      ; DURCH RÜCKSETZEN DIESER FLAGS WIRD
0186 3E 25          LD      A, 00100101B     ; DAS SIGNAL "RTS" NICHT JEDESMAL
0188 D3 41          OUT     (USARTC), A      ; NEU GESETZT.

018A INSTAT:
018A CD 02D8        CD      CONST           ; KEYBOARD INPUT FRAGEN OB
018D B7              OR      A                ; ZEICHEN DA IST
018E 20 22          JR      NZ, COMOUT       ; BEI AKKU = 0 IST KEIN ZEICHEN DA
0190 CD 0384        CD      CURAK           ; UND WIR HABEN ZEIT DEN CURSOR ZU
                                           ; AKTUALISIEREN
0193 C3 016F        JP      INCHR           ; UND DANN WIEDER NACH ROM ZU GEHEN
                                           ;
                                           ;-----
                                           ; ZEICHEN VON SERIELLER SCHNITTSTELLE EMPFANGEN UND NUN AUF DEM
                                           ; BILDSCHIRM DARSTELLEN.
                                           ;-----
SCREEN:
0196 2A 1342        LD      HL, (NETP04)     ; ZEIGER FUER ZEICHEN ZUM LESEN
0199 7E              LD      A, (HL)         ; UND HOLEN DES ZEICHENS
019A 23              INC     HL              ; ZEIGER VERBIEGEN
019B 22 1342        LD      (NETP04), HL    ; UND AUCH MERKEN
019E 11 1B46        LD      DE, NETEND      ; DANN NACHSCHAUEN
01A1 B7              OR      A                ; OHNE CARRY
01A2 ED 52          SBC     HL, DE          ; OB BUFFERENDE ERREICHT IST.
01A4 20 06          JR      NZ, SCCHR       ; WENN JA
01A6 21 1346        LD      HL, NETANF      ; DANN ZEIGER AUF BUFFERANFANG
01A9 22 1342        LD      (NETP04), HL    ; SETZEN
01AC CD 03B7        CD      CONCHR          ; UND ZEICHEN IN AKKU VERARBEITEN
01AF C3 018A        JP      INSTAT          ; VON HIER GEHTS NUR BIS KURZ VOR ROM

```

```

: ZEICHEN ÜBER TASTATUR EINGEGEBEN UND NUN UEBER DIE SERIELLE SCHNITTSTELLE
: AUSGEBEN.
:

```

01B2
01B2
01B5

COMOUT:

CD 02DC
F5

```

      CALL      CONIN      ; HOLE ZEICHEN VON TASTATUR AB
      PUSH      AF         ; MERKEN AUF DEM STACK UND SEHEN
                           ; WAS ZU TUN IST

```

```

: WENN LOCAL DANN VERSCHIEDENE TASTENCODES ALS ESCAPE SEQUENZEN
: BEHADELN. DIE TASTEN INS,DEL,HOME... USW. WERDEN INTERPRETIERT.
: DIES KANN ALS GRUNDLAGE FUER DIE ERWEITERUNG ZUM BLOCK-MODE VERWENDET
: WERDEN.
:

```

01B6 3A 03B4
01B9 B7
01BA CA 022D
01BD F1
01BE 21 018A
01C1 E5
01C2 FE 8B
01C4 CA 053E
01C7 FE 82
01C9 CA 0544
01CC FE 08
01CE CA 055E
01D1 FE 0C
01D3 CA 0581
01D6 FE 8F
01D8 CA 058A
01DB FE 89
01DD CA 0598

```

      LD        A,(LOCFLG) ; WURDE IRGENDWANN MAL LOCAL-MODE VERLANGT
      OR        A          ; BEI LOCAL-MODE WIRD DIES DURCH EINE
      JP        Z,NOSPK   ; EINS IM FLAGBYTE ANGEZEIGT
      POP       AF         ; ZEICHEN WIEDER NACH AKKU
      LD        HL,INSTAT  ; RÜCKSPRUNGADRESSE KURZ VOR ROM
      PUSH      HL         ; FÜR ALLE KNOCHENARBEITER
      CP        08BH       ; CURSOR DOWN
      JP        Z,COCUDO   ; TUE ALS OB LINEFEED
      CP        82H        ; CURSOR RIGHT
      JP        Z,COCURG   ; TUE ALS OB ZEICHEN EINGEGEBEN
      CP        08H        ; CURSOR LEFT
      JP        Z,COCULE   ; DAS IST WAS NEUES
      CP        0CH        ; ERASE SCREEN
      JP        Z,COERAS   ; WENN ER WEISS WAS ER TUT DANN BITTE
      CP        08FH       ; CURSOR HOME
      JP        Z,COCUHO   ; MY HOME IS MY CASTLE
      CP        089H       ; REVERSE INDEX (SCROLL UP)
      JP        Z,COREVI   ; DAS IST WAS HÜBSCHES, WAS DEM PC SONST FEHLT

```

01E0	FE 85	CP	085H	;	ERASE TO END OF SCREEN
01E2	CA 05AE	JP	Z,COERES	;	WIR MACHEN AUCH HALBE SACHEN
01E5	FE 86	CP	086H	;	ERASE TO END OF LINE
01E7	CA 05E4	JP	Z,COEREL	;	MEIN RADIERGUMMI IST ALLE
01EA	FE 87	CP	087H	;	ERASE FROM BEGINNING OF SCREEN
01EC	CA 0624	JP	Z,COERBS	;	HOFFENTLICH AUCH NICHT MEHR
01EF	FE 88	CP	088H	;	ERASE ENTIRE LINE
01F1	CA 0644	JP	Z,COERWL	;	DAS IST JA NOCH GANZ NUETZLICH
01F4	FE 8A	CP	08AH	;	ERASE FROM BEGINNING OF LINE
01F6	CA 068B	JP	Z,COERBL	;	ABER LANGSAM WIRD ES DOCH LÄSTIG
01F9	FE 95	CP	095H	;	INSERT BLANK LINE
01FB	CA 06D1	JP	Z,COINSL	;	ICH WÜNSCHTE ICH HÄTTE DAS FUER WORDSTAR
01FE	FE 96	CP	096H	;	DELETE LINE
0200	CA 0725	JP	Z,CODELL	;	UND DAS WÄRE SCHÖN FÜR TURBO PASCAL
0203	FE 8C	CP	08CH	;	DELETE ONE CHARACTER
0205	CA 077B	JP	Z,CODELC	;	JETZWIRD ER BESCHIEDEN
0208	FE 90	CP	090H	;	REVERSE ON
020A	CA 07DE	JP	Z,CORVON	;	UND JETZT ANSPRUCHSVOLL
020D	FE 91	CP	091H	;	REVERSE OFF
020F	CA 07E9	JP	Z,CORVOF	;	NA JA OK
0212	FE 92	CP	092H	;	BLINK ON
0214	CA 080E	JP	Z,COBLON	;	JETZ WIRD ER AFFIG
0217	FE 93	CP	093H	;	BLINK OFF
0219	CA 0819	JP	Z,COBL0F	;	ER HATS GEMERKT
021C	FE 94	CP	094H	;	LINE WRAP ON
021E	CA 0826	JP	Z,COWRON	;	DAS TUN WIR DOCH SOWIESO
0221	FE 80	CP	080H	;	LINE WRAP OFF
0223	CA 082E	JP	Z,COWROF	;	DA FINDET EINER KEIN ENDE
0226	FE 03	CP	03H	;	RESTORE ATTRIBUT
0228	CA 0836	JP	Z,COATRR	;	UND DER WEISS NICHT MEHR WAS ER GESAGT HAT
022B	E1	POP	HL	;	WURDE NICHT GEBRAUCHT, DA SICH EINER

[illegible]


```

025D 32 0384 LD (LOCFLG),A ; WAS ICH SAGE"
0260 C3 018A JP INSTAT ; DER SPRUCH MIT ROM WIRD LANGSAM LANGWEILIG
0263 FE F2 CP OF2H ; IST DIE HAMMERTASTE GEDRÜCKT ?
0265 20 11 JR NZ,TXTST ; NEIN
;+++++++
0267 CD 02DC COMMS2: CALL CONIN ; NOSROLL GEDRÜCKT
026A FE F2 CP OF2H ; MIT EINER TASTE IST ES DOCH SCHÖNER
026C 20 03 JR NZ,COMMONOW ; UND MAN VERRENKT SICH NICHT DIE HAND
026E C3 018A JP INSTAT ; BEIM ANHALTEN DER BILDSCHIRMAUSGABE
; HOFFENTLICH HAT SIE NICHT GEPRELT
; FÜR KLAVIERSPIELER HABEN
; WIR NUR EIN MÜDES HUPEN ÜBER
; WARTEN BIS SCROLL ENABLED
; ENDE FUNKTIONSTASTEN KEYPAD
; WENN EINER GESCHAFFT HAT UNS ZU ÜBERLISTEN
; SOLL ER SEINE FREUDE HABEN
; WIE WAR DAS MIT DEN KARTOFFELN
; MAL NACH ALLEN SONDERWÜNSCHEN SEHEN
; BEVOR WIR TRIVIALEN TEXT BEHANDELN
; ER WILL DIE ZEICHEN WIRKLICH SEHEN
; DIE ZEICHEN, DIE VON DER TASTATUR
; EINGEGEBEN WURDEN, WERDEN MIT
; REVERSE ATTRIBUT IN HEX-ASCII (XX-)
; DARGESTELLT, DAMIT DIE EINGEGEBENEN
; VON DEN EMPFANGENEN UNTERSCHIEDBAR SIND.
; DAS IST DER KNECHT FÜR ZEICHENAUSGABE
; REVERSE WIEDER ZURÜCKSETZEN, AUCH
; WENN DER BEDIENER NICHT WILL,
; DENN IM TRANSPARENTMODE SAGEN WIR WANN.

0271 0E 07 COMMONOW: LD C,07
0273 CD 02E0 CALL CONOUT
0276 18 EF JR COMMSC
;+++++++
0278 F5 TXTST: PUSH AF
0279 CHKLOC:
0279 3A 0385 LD A,(TRAFGL)
027C B7 OR A
027D 28 1E JR Z,NOLCOT
027F 3A 03AE LD A,(ATTRB)
0282 F6 80 OR REVMASK
0284 32 03AE LD (ATTRB),A
0287 F1 POP POP
0288 F5 PUSH AF
0289 CD 03B7 CALL CONCHR
028C 3A 03AE LD A,(ATTRB)
028F E6 7F AND NOT REVMASK
0291 32 03AE LD (ATTRB),A

```

```

0294 3A 03B4 LD A,(LOCFLG) ; SOLL NUN AUCH DAS ZEICHEN GESENDET WERDEN
0297 B7 OR A ; JA HIER DEBUGGED EINER BILDSCHIRMROUTINEN
0298 28 10 JR Z,TXTRD ; IST ES NICHT NÜTZLICH ?
029A C3 018A JP INSTAT ; VENI VIDI VICI
      NOLOCT: ; DIESER BESCHIEDENE MENSCH MÖCHTE DAS
029D LD A,(LOCFLG) ; EINGEGEBEN ZEICHEN OHNE SCHNICKSCHNACK
02A0 B7 OR A ; GESENDET HABEN
02A1 28 07 JR Z,TXTRD ; ODER DOCH ERST HÖREN
02A3 F1 POP AF ; WAS ER SAGT
02A4 CD 03B7 CALL CONCHR ; UM ZU SEHEN WAS ER DENKT
02A7 C3 018A JP INSTAT ; BELLA ITALIA
      TXTRD: IN A,(USARTC) ; IST DER USART SO GNÄDIG
02AA DB 41 AND 1 ; (TXRDY ?) EIN ZEICHEN AUFZUNEHMEN
02AC E6 01 JR Z,TXTRD ; FAST WIE BEI DER POST
02AE 28 FA POP AF ; NACH VIER TAGEN IST ER SCHON DA
02B0 F1 OUT (USARTD),A ; AUCH WENN ER PÜNKTLICH IM KASTEN WAR.
02B1 D3 40 JP INSTAT ; MIR GEHEN LANGSAM DIE KOMMENTARE AUS
02B3 C3 018A ; *****
; *****
; ; WER BIS HIERHER GEFOLGT IST, KENNT DIE LOGIK DES PROGRAMMS. FÜR DIE,
; ; DIE ES GENAU WISSEN WOLLEN, KÖNNTE ES SICH LOHNEN NOCH WEITER BEI
; ; DER STANGE ZU BLEIBEN.
; *****
; *****
      MESANF: LD HL,MELANF ; FORTSETZUNGSMELDUNG
0286 21 0918 CALL MESAUS ; AUSGEBEN
0289 CD 02CC MESLES: CALL CRTIN ; UND WARTEN BIS ZEICHEN KOMMT
02BC CD E40F CP '?' ; DER WILL ES GENAU WISSEN
02BF FE 3F RET NZ ; DER NICHT
02C1 C0 LD HL,MELSEQ ; ALSO SAGEN WIR, WAS WIR
02C2 21 0EEE

```

02C5	CD 02CC	CALL	MESAS		
02C8	CD E40F	CALL	CRTIN		
02C8	C9	RET			
02CC	7E	MESAS: LD	A, (HL)		
02CD	B7	OR	A		
02CE	C8	RET	Z		
02CF	E5	PUSH	HL		
02D0	4F	LD	C,A		
02D1	CD 02E0	CALL	CONOUT		
02D4	E1	POP	HL		
02D5	23	INC	HL		
02D6	18 F4	JR	MESAS		

; ALLES KÖNNEN
 ; UND WARTEN BIS ER ES GELESEN HAT
 ; HP ICH KOMME
 ; DIE MELDUNG EINES COMPUTERS ENTHÄLT
 ; IM GEGENSATZ ZU UNSEREN POLITIKERN
 ; NUR AM ENDE EIN LEERES WORT
 ; ANSONSTEN VERFÜGT SIE
 ; ÜBER BEWEGBARE SUBSTANZ
 ; DIE SICHTBAR,
 ; WIEDERAUFFINDBAR
 ; UND IN DER LÄNGE VORHERSEHBAR IST

;
 ; EIN PAAR PROMROUTINEN SIND GANZ NÜTZLICH
 ;
 ;

02D8	CD E40C	CONST:	CALL	CRTST	
02D8	C9		RET		
02DC	CD E40F	CONIN:	CALL	CRTIN	
02DF	C9		RET		
02E0	CD E412	CONOUT:	CALL	CRTOUT	
02E3	C9		RET		

; SEHE NACH OB EINER ANGEKLOPFT HAT
 ; DANN LASS IHN REIN
 ; UND ZEIG IHM WAS DU HAST

; BLEIBEN WIR MAL EINEN MOMENT ERNSTHAFT, FUER DIE INTERRUPTSERVICEROUTINE.
 ; SIE KÖNNTE AUCH FUER ANDERE ZWECKE NÜTZLICH SEIN.

02E4	C3 02E7	RECJMP:	JP	RECINT	
02E7		RECINT:			
02E7	F3		DI		

; SPRUNGBEFEHL FUER INTERRUPTADRESSE
 ; RECEIVE INTERRUPTSERVICEROUTINE
 ; NUR ZUR INFORMATION

```

02E8 ED 73 1B      (SPSAVS),SP
02EC 31 1BC8     SP,SPSAVS
02EF F5         PUSH
02F0 E5         PUSH
02F1 D5         PUSH
02F2 2A 1344    HL,(NETPI4)
02F5 DB 40      A,(USARTD)
02F7 77        (HL),A
02F8 23        HL
02F9 22 1344    (NETPI4),HL
02FC 11 1B46    DE,NETEND
02FF B7        OR A
0300 ED 52      HL,DE
0302 20 0F      JR NZ,RECNO
0304 21 1346    LD HL,NETANF
0307 22 1344    LD (NETPI4),HL
030A 3E 01      LD A,1
030C 32 0345    LD (RTSFLG),A
030F 3E 05      LD A,00000101B
0311 D3 41      OUT (USARTC),A
0313          RECNO:
0313 2A 1344    LD HL,(NETPI4)
0316 23        INC HL
0317 23        INC HL
0318 23        INC HL
0319 23        INC HL
031A 11 1B46    LD DE,NETEND
031D B7        OR A
031E ED 52      SBC HL,DE
0320 20 03      JR NZ,RECNOX
0322 21 134B    LD HL,NETANF+5
;
; LOKALER STACK
; IST EINE SICHERE SACHE
; ALLE VERWENDETEN REGISTER
; MÜSSEN !!!
; GESICHERT WERDEN
; EINGABEZEIGER LADEN (DORT KOMMT ZEICHEN HIN)
; HOLEN ZEICHEN VON USART
; NACH RINGPUFFER
; UND ERHÖHE ZEIGER
;
; VIELLEICHT SCHON PUFFERENDE ?
; OHNE CARRY
; DIFFERENZ BILDEN
; WIR SIND NOCH NICHT
; ABER HIER AM ENDE
; UND LADEN DEN ANFANG DES PUFFERS
; WIR SETZEN HIER MAL PUFFER VOLL
; UND HANDSHAKE
; OFF (RTS)
; BEI USART, UM IMMER MAL WIEDER AUFZUHOLEN
;
; ABER AUCH
; WENN WIR NICHT AM PUFFERENDE WAREN
; KANN DER PUFFER VOLL WERDEN
; UND ES SOLLTEN EIN PAAR ZEICHEN
; RESERVE BLEIBEN, UM EINE ESCAPE
; SEQUENZ NICHT MITTEN DRIN
; ABZUSCHNEIDEN
; JETZT SCHAUEN WIR NACH
; OB WIR DEN
; OUTPUT ZEIGER

```



```

02E8 ED 73 1B      (SPSAVS),SP
02EC 31 1BC8      SP,SPSAVS
02EF F5          AF
02F0 E5          PUSH
02F1 D5          PUSH
02F2 2A 1344      LD
02F5 DB 40        IN
02F7 77          LD
02F8 23          INC
02F9 22 1344      LD
02FC 11 1B46      LD
02FF B7          OR
0300 ED 52        SBC
0302 20 0F        JR
0304 21 1346      LD
0307 22 1344      LD
030A 3E 01        LD
030C 32 0345      LD
030F 3E 05        LD
0311 D3 41        OUT
0313              RECNO:
0313 2A 1344      LD
0316 23          INC
0317 23          INC
0318 23          INC
0319 23          INC
031A 11 1B46      LD
031D B7          OR
031E ED 52        SBC
0320 20 03        JR
0322 21 134B      LD
;
; (SPSAVS),SP
; SP,SPSAVS
; AF
; PUSH
; PUSH
; PUSH
; LD
; IN
; LD
; INC
; LD
; LD
; OR
; SBC
; JR
; LD
; LD
; LD
; LD
; LD
; LD
; OUT
;
; RECNO:
;
; LD
; INC
; INC
; INC
; INC
; LD
; OR
; SBC
; JR
; LD
;
; LOKALER STACK
; IST EINE SICHERE SACHE
; ALLE VERWENDETEN REGISTER
; MÜSSEN !!!
; GESICHERT WERDEN
; EINGABEZEIGER LADEN (DORT KOMMT ZEICHEN HIN)
; HOLEN ZEICHEN VON USART
; NACH RINGPUFFER
; UND ERHÖHE ZEIGER
;
; VIELLEICHT SCHON PUFFERENDE ?
; OHNE CARRY
; DIFFERENZ BILDEN
; WIR SIND NOCH NICHT
; ABER HIER AM ENDE
; UND LADEN DEN ANFANG DES PUFFERS
; WIR SETZEN HIER MAL PUFFER VOLL
; UND HANDSHAKE
; OFF (RTS)
; BEI USART, UM IMMER MAL WIEDER AUFZUHOLEN
;
; ABER AUCH
; WENN WIR NICHT AM PUFFERENDE WAREN
; KANN DER PUFFER VOLL WERDEN
; UND ES SOLLTEN EIN PAAR ZEICHEN
; RESERVE BLEIBEN, UM EINE ESCAPE
; SEQUENZ NICHT MITTEN DRIN
; ABZUSCHNEIDEN
; JETZT SCHAUEN WIR NACH
; OB WIR DEN
; OUTPUT ZEIGER

```

```

0325 ED 5B 13 RECNOX: LD DE, (NETP04) ; SCHON FAST EINGEHOLT HABEN
0325 B7 OR A ; OHNE CARRY
0329 ED 52 SBC ; WIRD DIFFERENZ VERSCHIEDEN NULL
032A 20 09 JR NZ, RECNOY ; WENN KEINE GEFahr
032C 3E 01 LD A, 1 ; UM EINEN HACKENTRITT DES ZEIGERS
0330 32 0345 LD (RTSELG), A ; ZU VERMEIDEN, WIRD HANDSHAKE
0333 3E 05 LD A, 00000101B ; (RTS) AM
0335 D3 41 OUT (USARTC), A ; USART ZURÜCKGESETZT.

0337 RECNOY: POP DE ; RÜCKLADEN REGISTER
0337 D1 POP HL ; KEINS VERGESSEN
0338 E1 LD A, 20H ; UND PIC ENDE INTERRUPT SAGEN
0339 3E 20 OUT (PIC70), A ; BENUTZT WIRD "NON SPECIFIC EOI"
033B D3 70 POP AF ; AUCH AKKU DARF NICHT VERÄNDERT WERDEN.
033D F1 LD SP, (SPSAVS) ; UND DER STACK ERST RECHT NICHT
033E ED 7B 1B LD EI ; JETZT DARFST DU WIEDER
0342 FB RETI ; DEN BRAUCHEN WIR HIER EIGENTLICH NICHT
0343 ED 40 ; AM ES IST NETT ZU SEHEN, DA FUER Z80
; PERIPHERIE ERFORDERLICH
; MERKER FUER PUFFER VOLL
0345 00 RTSFLG: DB 0 ; *****
; ; ; ; ; *****
; ; INTERNES VIDEOAM AUF BILDSCHIRM ; ; ; ; ; *****
; ; ; ; ; *****
VIDDIS:
0346 21 25CA LD HL, VIDPUF
0346 11 F000 LD DE, VRAMAD ; ADRESSE TEXT RAM
0349 01 0780 LD BC, 1920 ; 24 ZEILEN (MAN KÖNNTE AUCH 25)
034C

```



```

;*****
; CURSOR AUF AKTUELLE POSITION STELLEN
;*****

```

037E

CURUP:

```

; WENN NOCH ZEICHEN IM PUFFER, DANN VERZICHTE AUF CURSOR UPDATE
;

```

```

037E 3E 01
0380 32 037C
0383 C9

```

```

LD A,1
LD (ANZEI),A
RET

```

```

; CURSORUPDATE VERLANGEN
; FUER HAUPTPROGRAMM
; DAS SOLL ENTSCHEIDEN OB CURSOR

```

```

; AKTUALISIERT WIRD

```

```

; HIER PASSIERTS NUN WIRKLICH

```

```

; NACHSEHEN

```

```

; OB AUCH WIRKLICH VERLANGT

```

```

; WAR NUR EIN GERÜCHT

```

```

; NICHT NOCHMAL

```

```

; SOLANGE NICHTS ANDERES VORLIEGT

```

```

; HIER WIRD DIE LOGISCHE

```

```

; CURSORROUTINE

```

```

; DES SYSTEMPROMS

```

```

; BENUTZT

```

```

; WEIL INTERN DIE

```

```

; ZEILEN UND

```

```

; SPALTEN

```

```

; SO GEZÄHLT WERDEN,

```

```

; DASS NICHT SOVIEL GERECHNET WERDEN MUSS

```

```

; UND EINE AUSGABE UEBER PORT AN 6845

```

```

; ETWAS UMSTÄNDLICH WÄRE

```

```

; IST IMMER NOCH SCHNELL GENUG

```

```

; HP ICH KOMME

```

CURAK:

```

0384 3A 037C
0387 B7
0388 C8
0389 AF

```

```

LD A,(ANZEI)
OR A
RET Z
XOR A

```

```

LD
LD
LD

```

```

LD (ANZEI),A
C,1BH

```

```

CALL CRTOUT
C,"Y"

```

```

LD
CALL

```

```

LD A,(ZEIL)
ADD A,20H

```

```

LD C,A
CALL CRTOUT

```

```

LD A,(SPAL)
ADD A,20H

```

```

LD C,A
CALL CRTOUT

```

```

0397 3A 03AA
039A C6 20
039C 4F

```

```

LD
CALL

```

```

039D CD E412
03A0 3A 03AB
03A3 C6 20
03A5 4F

```

```

LD A,(SPAL)
ADD A,20H
LD C,A

```

```

CALL CRTOUT

```

```

CALL

```

```

03A6 CD E412
03A9 C9

```

```

CALL
RET

```

```

03AA 00      ; ZEIL:      DEFB 0      ; AKTUELLE CURSORZEILE
03AB 00      ; SPAL:      DEFB 0      ; AKTUELLE CURSORSPALTE
03AC 00      ; SPSAV:     DEFB 0      ; HILFSFELD FUER SPALTE
03AD 00      ; ZESAV:     DEFB 0      ; HILFSFELD FUER ZEILE
;-----;
;
;   SPEICHERBEREICHE FUER FLAGS MIT VORBELEGUNG.
;-----;
03AE 07      ; ATTRB:     DEFB 07     ; ATTRIBUT (DEFAULT IST NORMAL VIDEO)
03AF 00      ; WRAP:      DEFB 0      ; WRAP IST NACH START EINGESCHALTET
03B0 00      ; ESCFLG:    DEFB 0      ; MERKE DIR EIN ESCAPE
03B1 00      ; CURFLG:    DEFB 0      ; MERKE DIR, WENN CURSOR POSITIONIERT WIRD
03B2 00      ; CURENB:    DEFB 0      ; CURSOR IST NORMALERWEISE EINGESCHALTET
03B3 00      ; PAGE:      DEFB 0      ; PAGEMODE IST NORMALERWEISE AUSGESCHALTET
03B4 00      ; LOCFLG:    DEFB 0      ; ONLINE MODE NACH START
03B5 00      ; TRAFGL:    DEFB 0      ; TRANSPARENTMODE NUR AUF ANFORDERUNG
03B6 00      ; WORKFL:    DEFB 0      ; HILFSFELD
;-----;
;
;   HIER BEGINNEN DIE ROUTINEN, DIE DIE KNOCHENARBEIT AUSFÜHREN.
;-----;
;
;   IM AKKU STEHT EIN ZEICHEN, DAS IN DAS VIDEOGRAM AUSGEGEBEN WERDEN SOLL
;-----;
03B7 4F      ; CONCHR:    LD      C,A      ; AUFHEBEN ZEICHEN
03B8 3A 03B5 ;           LD      A,(TRAFGL) ; TRANSPARENTMODE VERLANGT ?
03B8 B7      ;           OR      A      ; BEI <> NULL SOLL ZEICHEN ALS HEX - ASCII
03BC C2 0414 ;           JP      NZ,CONASC ; AUSGEGEBEN WERDEN

```

03BF	3A	03B1	LD	A, (CURFLG)	;	GEHÖRT DAS ZEICHEN ZU EINER
03C2	B7		OR	A	;	CURSORPOSITIONIERUNG
03C3	C2	047F	JP	NZ, CONCUR	;	FLAG <> 0 DANN IST POSITIONIERUNG AKTIV
03C6	3A	03B0	LD	A, (ESCFLG)	;	GEHÖRT ZEICHEN ZU EINER ESCAPE SEQUENZ
03C9	B7		OR	A	;	FLAG <> 0 ZEIGT SEQUENZ AKTIV
03CA	C2	04A6	JP	NZ, CONESC	;	UND WIRD WEITER BEARBEITET
03CD	79		LD	A, C	;	RÜCKLADEN ZEICHEN
03CE	FE	08	CP	0BH	;	BACKSPACE GEDRÜCKT ?
03D0	CA	055E	JP	Z, COCULE	;	JA DANN FÜHRE CURSOR LINKS AUS
03D3	FE	07	CP	07	;	BELL VERLANGT
03D5	CA	03ED	JP	Z, COBELL	;	JA DANN BELLE
03D8	FE	18	CP	1BH	;	BEGINNT ESCAPE SEQUENZ
03DA	28	16	JR	Z, CONESS	;	JA, DANN SETZE FLAG
03DC	FE	0D	CP	0DH	;	CR ?
03DE	28	18	JR	Z, CONCR	;	JA DANN FÜHRE AUS
03E0	FE	0A	CP	0AH	;	LF ?
03E2	28	1C	JR	Z, CONLF	;	JA
03E4	FE	20	CP	20H	;	ANDERE SONDERZEICHEN ?
03E6	38	2B	JR	C, CONSON	;	NICHT ZULASSEN
03E8	FE	7F	CP	7FH	;	NORMALES ASCII - ZEICHEN
03EA	38	28	JR	C, CONASC	;	JA
					;	
					;	HIER KÖNNTE DIE BEHANDLUNG VON GRAPHIKZEICHEN EINGEBAUT WERDEN.
					;	ZUR ZEIT WERDEN NOCH KEINE GRAPHIKZEICHEN UNTERSTÜTZT.
					;	NICHT ZUGELASSENE ZEICHEN WERDEN EINFACH IGNORIERT.
					;	
03EC	C9			RET	;	
					;	***** 07 IN AKKU = BELL
03ED	4F			COBELL: LD	;	BELLE
03EE	CD	E412		CALL CRTOUT	;	MIT SYSTEMPROM
03F1	C9			RET	;	

```

03F2 3E 01 ;+++++ 1B IN AKKU = ESCAPE
03F4 32 03B0 CONESS: LD A,1 ; ZEIGE AN, DASS JETZT ZEICHEN
03F7 C9 (ESCFLG),A ; KOMMEN, DIE ZU EINER ESC SEQUENZ GEHÖREN.
      RET

03F8 AF ;+++++ OD IN AKKU = WAGENRÜCKLAUF
03F9 32 03AB CONCR: XOR A ; WAGENRÜCKLAUF SETZT
03FC CD 037E LD (SPAL),A ; SPALTE AUF NULL
03FF C9 CALL CURUP ; UPDATE CURSOR
      RET

0400 3A 03AA ;+++++ OA IN AKKU = LINE FEED
0403 3C INC A,(ZEIL) ; AKTUELLE ZEILE LADEN
0404 FE 18 CP A ; NAECHSTE ZEILE
0406 38 04 JR ZMAX ; VIELLEICHT BILDSCHIRMEDE ?
0408 CD 086C C,CONLFB ; NEIN, DANN IST ALLES OK
040B C9 CALL CONSCR ; SCROLL SCREEN DA BILDSCHIRMEDE
      RET

040C 32 03AA CONLFB: LD (ZEIL),A ; NEUE AKTUELLE ZEILE MERKEN
040F CD 037E CALL CURUP ; UND CURSOR SETZEN WENN ZEIT IST
0412 C9 RET

0413 ;+++++ WEITERE SONDERZEICHEN
0413 CONSON: RET

;+++++ NOCH KEINE WEITERE SONDERZEICHENBEHANDLUNG
;+++++ NORMALES ASCII ZEICHEN
0414 CONASC:
0414 LD A,C ; RUECKLADEN ZEICHEN
0415 F5 PUSH AF ; UND AUFHEBEN
0416 3A 03B5 LD A,(TRAFGL) ; TRANSPARENTMODE VERLANGT
0419 B7 OR A ; 0 HEISST NORMALE DARSTELLUNG
041A 28 08 JR Z,CONDI1 ; DANN TU AUCH SO
041C F1 POP AF ; ZEICHEN WIEDER HOLEN

```

0410	CD	0902	CALL	HEXOUT	:	UND ALS ZWEI HEX NIBBLE IN ASCII UMWANDELN
0420	3E	2D	LD	A, '-'	:	MIT BINDESTRICH VON NACHFOLGERN TRENNEN
0422	18	01	JR	CONDIS	:	UND DEN '-' AUCH AUSGEBEN
0424	F1		COND11: POP	AF	:	RUECKLADEN ZEICHEN
0425	F5		CONDIS: PUSH	AF	:	UND WIEDER AUFHEBEN
0426	CD	0849	CALL	VIDADD	:	BERECHNET ADRESSE IN INTERNEM VIDEOGRAM
					:	UNTER VERWENDUNG VON ZEIL UND SPAL
0429	F1		POP	AF	:	ZEICHEN WIEDER HOLEN
042A	EB		EX	DE, HL	:	RAM ADRESSE WAR IN HL VON VIDADD
042B	21	25CA	LD	HL, VIDPUF	:	INTERNER PUFFERANFANG
042E	19		ADD	HL, DE	:	PLUS RELATIVE POSITION
042F	77		LD	(HL), A	:	ERGIBT ADRESSE ZUM SPEICHERN DES ZEICHENS
0430	21	F000	LD	HL, VRAMAD	:	JETZT DIREKT BILDSCHIRMDRESSE AUSRECHNEN
0433	19		ADD	HL, DE	:	DURCH F000 + DISTANZ AUS VIDADD
0434	D5		PUSH	DE	:	DISTANZ AUFHEBEN
0435	11	0800	LD	DE, 0800H	:	UND ATTRIBUTADRESSE AUSRECHNEN
0438	D5		PUSH	DE	:	DIESER AUFSTAND WIRD GEMACHT
0439	E5		PUSH	HL	:	UM NACHFOLGEND BEI
043A	EB		EX	DE, HL	:	AUSGESCHALTETEM INTERRUPT
043B	21	03B6	LD	HL, WORKFL	:	MÖGLICHT SCHNELL
043E	77		LD	(HL), A	:	DAS ZEICHEN UND DAS ATTRIBUT
043F	F3		DI		:	IN DAS VIDEOGRAM ZU ÜBERTRAGEN
0440	DB	10	SCRHX: IN	A, (SPORT)	:	MIT EINEM MINIMUM AN FLACKERN
0442	17		RLA		:	DAS KENNEN WIR SCHON VON OBEN
0443	30	FB	JR	NC, SCRHX	:	
0445	ED	A0	LDI		:	DAS IST DER SCHNELLSTE WEG
0447	FB		EI		:	ZWISCHENDURCH MAL ZEICHEN REIN LASSEN
0448	3A	03AE	LD	A, (ATTRB)	:	LADEN AKTUELLES ATTRIBUT
044B	E1		POP	HL	:	UND DEN GLEICHEN
044C	D1		POP	DE	:	AUFSTAND
044D	19		ADD	HL, DE	:	WIEDER WIE BEIM

```

044E EB      EX      DE,HL      ; TEXTZEICHEN
044F LD      LD      HL,WORKFL ; UM NACH
0452 LD      LD      (HL),A     ; AUSSCHALTEN DES
0453 DI      DI      ; INTERRUPTS
0454 DB 10    IN      A,(SPORT) ; BEI ERKENNEN
0456 LD      LD      RLA        ; EINES STRAHLRÜCKLAUFIMPULSES
0457 JR      JR      NC,SCRHY   ; MÖGLICHST
0459 ED A0    LD      LD      ; SCHNELL DAS ATTRIBUT ZU SCHREIBEN, UM
045B FB      FB      EI         ; DEN INTERRUPT WIEDER ZUZULASSEN
045C D1      D1      POP        ; DAS ATTRIBUT
045D LD      LD      HL,VIDPUA  ; MUSS NUN AUCH NOCH
0460 LD      LD      HL,DE      ; IM INTERNEN ATTRIBUT RAM
0461 LD      LD      A,(ATTRB)  ; AKTUALISIERT
0464 LD      LD      (HL),A     ; WERDEN.
0465 LD      LD      A,(SPAL)   ; NACH EINEM EINZELNEN ZEICHEN
0468 LD      LD      INC       ; MUSS DIE SPALTENNUMMER ERHÖHT WERDEN.
0469 FE 50    CP      CP      SPMAX ; IST SPALTENENDE NOCH NICHT ERREICHT
046B JB 0B    JR      JR      C,CONASN ; IST ALLES OK
046D LD      LD      A,(WRAP)   ; BEI SPALTENENDE WIRD GEPRÜFT, OB WRAP
0470 LD      LD      OR      A   ; ( ÜBERGANG AUF NEUE ZEILE ) AUSGEFÜHRT
0471 LD      LD      RET      NZ  ; WERDEN SOLL. BEI WRAP <> 0 NEIN.
0472 LD      LD      (SPAL),A   ; SPALTE NULL IN NEUER ZEILE
0475 LD      LD      CONLF      ; UND LINE FEED AUSFÜHREN
0478 LD      LD      (SPAL),A   ; NEUE SPALTE MERKEN
047B LD      LD      CALL      ; UND GEBEBENFALLS CURSOR AKTUALISIEREN
047E C9      RET      RET      ;
; *****
; ; CURSOPositionierung Aktiv
; ; IN AKKU STEHT 1 WENN ZEILE UND 2 WENN SPALTE EMPFANGEN WURDE
; *****

```


0483	CA 053E	JP	Z, COCUDO	TUE ALS OB LINEFEED
0486	FE 43	CP	'C'	CURSOR RIGHT
0488	CA 0544	JP	Z, COCURG	TUE ALS OB ZEICHEN EINGEGEBEN
048B	FE 44	CP	'D'	CURSOR LEFT
048D	CA 055E	JP	Z, COCULE	
04C0	FE 45	CP	'E'	ERASE SCREEN
04C2	CA 0581	JP	Z, COERAS	
04C5	FE 48	CP	'H'	CURSOR HOME
04C7	CA 058A	JP	Z, COCUHO	
04CA	FE 49	CP	'I'	REVERSE INDEX (SCROLL UP)
04CC	CA 0598	JP	Z, COREVI	
04CF	FE 4A	CP	'J'	ERASE TO END OF SCREEN
04D1	CA 05AE	JP	Z, COERES	
04D4	FE 4B	CP	'K'	ERASE TO END OF LINE
04D6	CA 05E4	JP	Z, COEREL	
04D9	FE 64	CP	'd'	ERASE FROM BEGINNING OF SCREEN
04DB	CA 0624	JP	Z, COERBS	
04DE	FE 6C	CP	'l'	ERASE ENTIRE LINE
04E0	CA 0644	JP	Z, COERWL	
04E3	FE 6F	CP	'o'	ERASE FROM BEGINNING OF LINE
04E5	CA 068B	JP	Z, COERBL	
04E8	FE 4C	CP	'L'	INSERT BLANK LINE
04EA	CA 06D1	JP	Z, COINSL	
04ED	FE 4D	CP	'M'	DELETE LINE
04EF	CA 0725	JP	Z, CODELL	
04F2	FE 4E	CP	'N'	DELETE ONE CHARACTER
04F4	CA 077B	JP	Z, CODELC	
04F7	FE 65	CP	'e'	ENABLE CURSOR
04F9	CA 07AD	JP	Z, COCUEB	
04FC	FE 66	CP	'f'	DISABLE CURSOR
04FE	CA 07B5	JP	Z, COCUDB	


```

053E CD 0400          ; BEARBEITE WIE LINEFEED
0541 C3 0538          ; ESCAPE SEQUENZ ENDE
          ;+++++
0544 CD 0465          ; WIE ZEICHEN EINGEGEBEN
0547 C3 0538          ; ESCAPE SEQUENZ ENDE
          ;+++++
054A 3E 01           ; CURSOR POSITIONIEREN
054C 32 03B1          ; MARKIERE POSITIONIERUNG AKTIV
054F C9              ; FÜR VERARBEITUNG ZEILE
          ;+++++
          ; ESCAPE FLAG BLEIBT AN
          ; CURSOR UP
          ;+++++
0550 3A 03AA          ; LADE AKTUELLE ZEILE
0553 B7              ; PRÜFE OB BEREITS OBERER BILDRAND
0554 C8              ; JA, DANN TUE NICHTS
0555 3D              ; SONST ZEILE - 1
0556 32 03AA          ; MERKEN
0559 CD 037E          ; UND CURSOR AKTUALISIEREN
055C 18 DA           ; ENDE ESCAPE SEQUENZ
          ;+++++
          ; CURSOR LINKS
          ;+++++
055E 3A 03AB          ; PRÜFEN, OB SPALTE
0561 B7              ; BEREITS NULL
0562 28 09           ; JA, DANN SONDERBEHANDLUNG
0564 3D              ; SONST SPALTE - 1
0565 32 03AB          ; MERKEN
0568 CD 037E          ; UND CURSOR AKTUALISIEREN
056B 18 CB           ; ENDE ESCAPE SEQUENZ
056D 3A 03AA          ; WRAP AUFWÄRTS
0570 B7              ; NUR WENN ZEILE NOCH NICHT NULL
          ;+++++
0571 28 C5           ; SONST IGNORIERE UND ESCAPE SEQUENZ ENDE
0573 3D              ; WENN OK ZEILE - 1
0574 32 03AA          ; MERKEN
0577 3E 4F           ; UND SPALTENPOSITION AM ZEILENENDE
          ;+++++
          ; A, SPMAX-1
          ;+++++

```

```

0579 32 03AB LD (SPAL),A ; MERKEN
057C CD 037E CALL CURUP ; CURSOR AKTUALISIEREN
057F 18 B7 JR ESCEND ; UND ESCAPE SEQUENZ ENDE
;+++++++ ERASE SCREEN
0581 CD 08DA COERAS: CALL VIDINI ; LOESCHT TEXT UND ATTRIBUT
0584 CD 0346 CALL VIDDIS ; UND ZEIGT LEEREN BILDSCHIRM AN
0587 C3 0538 JP ESCEND ; ENDE ESCAPE SEQUENZ
;+++++++ CURSOR HOME
058A 3E 00 COCUHO: LD A,0 ; NULL NACH
058C 32 03AA LD (ZEIL),A ; AKTUELLE ZEILE
058F 32 03AB LD (SPAL),A ; UND SPALTE
0592 CD 037E CALL CURUP ; AKTUALISIEREN CURSOR
0595 C3 0538 JP ESCEND ; ENDE ESCAPE SEQUENZ
;+++++++ REVERSE INDEX (SCROLL UP)
0598 3A 03AA COREVI: LD A,(ZEIL) ; WENN OBERSTE BILDSCHIRMZEILE
059B B7 OR A ; ERREICHT IST
059C 28 0A JR Z,CORVS ; WIRD VIDEOGRAM GEROLLT
059E 30 DEC A ; SONST WIRD
059F 32 03AA LD (ZEIL),A ; ZEILE NACH OBEN VERSETZT
05A2 CD 037E CALL CURUP ; DER CURSOR AKTUALISIERT
05A5 C3 0538 JP ESCEND ; UND DIE ESCAPE SEQUENZ BEENDET
05A8 CD 089E CORVS: CALL REVSCR ; REVERSE SCROLL
05AB C3 0538 JP ESCEND ; UND ESCAPE SEQUENZ ENDE
;+++++++ ERASE TO END OF SCREEN
COERES: ; OFFSET AKTUELLE POSITION VIDEOGRAM IN HL
; NACH DE ZUM ADDIEREN
05AE CD 0849 CALL VIDADD ; SCREENLAENGE IN ZEICHEN
05B1 EB DE,HL ; LOESCHEN CARRY
05B2 21 0780 LD HL,SCLNG ; - AKTUELLE DISTANZ IST RESTCOUNTER ERASE
05B5 B7 OR A ; NACH COUNTERREGISTER
05B6 ED 52 SBC HL,DE
05B8 E5 PUSH HL

```

C1	POP	BC	HL,VIDPUF	IST DER EINFACHSTE WEG
21 25CA	LD	HL,VIDPUF		
19	ADD	HL,DE		
3E 20	LD	A,20H		
05	PUSH	DE		
05C0	PUSH	BC		
05C1	CALL	DOERA		
00 05D6	POP	BC		
05C5	POP	DE		
05C6	LD	HL,VIDPUA		
05C7	ADD	HL,DE		
05CA	LD	A,07H		
05C8	CALL	DOERA		
05C0	CALL	VIDDIS		
0346	JP	ESCOND		
05D3	DOERA:	LD	(HL),A	
05D6	LD	D,H		
05D7	LD	E,L		
05D8	LD	DE		
05D9	INC	DE		
05DA	LD	A,B		
05DB	OR	C		
05DC	RET	Z		
05DD	DEC	BC		
05DE	LD	A,B		
05DF	OR	C		
05E0	RET	Z		
ED 80	LDIR			
05E1	RET			
05E3	*****			
0849	COEREL:	CALL	VTDADD	
EB	EX	DE,HL		

05E8	3E 50	LD	A, SPMAX	;	ZEILENLÄNGE IN ZEICHEN
05E9	21 03AB	LD	HL, SPAL	;	- AKTUELLE SPALTE
05ED	96	SUB	(HL)	;	ERGIBT COUNTER IN A
05EE	06 00	LD	B, 0	;	LDIR ZÄHLT
05F0	4F	LD	C, A	;	BC
05F1	21 25CA	LD	HL, VIDPUF	;	DIE STARTADRESSE IST PUFFERANFANG
05F4	19	ADD	HL, DE	;	PLUS AKTUELLE DISTANZ
05F5	3E 20	LD	A, 20H	;	ZEICHEN ZUM LOESCHEN
05F7	D5	PUSH	DE	;	AUFHEBEN OFFSET FUER ATTRIBUT RAM
05F8	C5	PUSH	BC	;	AUFHEBEN COUNTER
05F9	CD 05D6	CALL	DOERA	;	LOESCHE IN TEXTMEMORY
05FC	C1	POP	BC	;	RUECKLADEN RESTCOUNTER
05FD	D1	POP	DE	;	RUECKLADEN OFFSET
05FE	21 37CA	LD	HL, VIDPUA	;	ATTRIBUTSPEICHER
0601	19	ADD	HL, DE	;	AKTUELLE POSITION
0602	3E 07	LD	A, 07H	;	ATTRIBUT ZUM LOESCHEN
0604	D5	PUSH	DE	;	AUFHEBEN OFFSET
0605	C5	PUSH	BC	;	AUFHEBEN COUNTER
0606	CD 05D6	CALL	DOERA	;	LOESCHE ATTRIBUTSPEICHER
0609	C1	POP	BC	;	COUNTER
060A	D1	POP	DE	;	OFFSET
060B	21 F000	LD	HL, VRAMAD	;	TUE DAS GLEICHE MIT DEM
060E	19	ADD	HL, DE	;	VIDEORAM WIE
060F	3E 20	LD	A, 20H	;	MIT DEM INTERNEN
0611	D5	PUSH	DE	;	SPEICHER,
0612	C5	PUSH	BC	;	WEIL DAS SCHNELLER
0613	CD 05D6	CALL	DOERA	;	GEHT, ALS WENN
0616	C1	POP	BC	;	WIR VIDDIS BENUTZEN, UM ALLES
0617	D1	POP	DE	;	NEU AUSZUGEBEN
0618	21 F800	LD	HL, ARAMAD	;	MIT DEM ATTRIBUTSPEICHER
0618	19	ADD	HL, DE	;	MUSS NATÜRLICH GENAUSSO

```

061C 3E 07 LD A,07H ; VERFAHREN WERDEN, WIE
061E CD 05D6 CALL DOERA ; MIT DEM TEXTSPEICHER
0621 C3 0538 JP ESCEND ; ENDE ESCAPE SEQUENZ

;+++++++ ERASE FROM BEGINNING OF SCREEN
0624 CD 0849 COERBS: CALL VIDADD ; OFFSET VIDEOAM IN HL
0627 EB DE,HL ; NACH DE
0628 D5 DE ; NACH COUNTERREGISTER
0629 C1 BC ; DA OFFSET GLEICHZEITIG COUNT
062A 21 25CA HL,VIDPUF ; PUFFERANFANG
062D 3E 20 LD A,20H ; ZEICHEN ZUM LOESCHEN
062F D5 DE ; AUFHEBEN OFFSET
0630 C5 BC ; AUFHEBEN COUNTER
0631 CD 05D6 CALL DOERA ; LOESCHE IN TEXTMEMORY
0634 C1 BC ; RUECKLADEN RESTCOUNTER
0635 D1 DE ; RUECKLADEN OFFSET
0636 21 37CA HL,VIDPUA ; ATTRIBUTSPEICHER
0639 3E 07 LD A,07H ; ATTRIBUT ZUM LOESCHEN
063B CD 05D6 CALL DOERA ; LOESCHE ATTRIBUTSPEICHER
063E CD 0346 CALL VIDDIS ; ZEIGE VIDEOAM AN
0641 C3 0538 JP ESCEND ; ENDE ESCAPE SEQUENZ

;+++++++ ERASE ENTIRE LINE
0644 3A 03AB COERWL: LD A,(SPAL) ; AKTUELLE SPALTENPOSITION
0647 F5 AF ; AUFHEBEN
0648 AF A ; LOESCHEN SPALTE
0649 32 03AB (SPAL),A ; FUER VIDADD, DA WIR ZEILENANFANG BRAUCHEN
064C CD 0849 CALL VIDADD ; OFFSET VIDEOAM IN HL
064F EB DE,HL ; NACH DE
0650 06 00 LD B,0 ; ZÄHLER IST GESAMTE ZEILE
0652 0E 50 LD C,SPMAX ; NACH COUNTERREGISTER
0654 21 25CA HL,VIDPUF ; PUFFERANFANG

```

0657	19	ADD	HL,DE	:	AKTUELLE ADRESSE ZEILENANFANG
0658	3E 20	LD	A,20H	:	ZEICHEN ZUM LOESCHEN
065A	D5	PUSH	DE	:	AUFHEBEN OFFSET
065B	C5	PUSH	BC	:	AUFHEBEN COUNTER
065C	CD 05D6	CALL	DOERA	:	LOESCHE IN TEXTMEMORY
065F	C1	POP	BC	:	RUECKLADEN RESTCOUNTER
0660	D1	POP	DE	:	RUECKLADEN OFFSET
0661	21 37CA	LD	HL,VIDPUA	:	ATTRIBUTSPEICHER
0664	19	ADD	HL,DE	:	AKTUELLE POSITION
0665	3E 07	LD	A,07H	:	ATTRIBUT ZUM LOESCHEN
0667	D5	PUSH	DE	:	DISTANZ
0668	C5	PUSH	BC	:	COUNTER
0669	CD 05D6	CALL	DOERA	:	LOESCHE ATTRIBUTSPEICHER
066C	C1	POP	BC	:	COUNTER
066D	D1	POP	DE	:	DISTANZ
066E	21 F000	LD	HL,VRAMAD	:	PUFFERANFANG
0671	19	ADD	HL,DE	:	AKTUELLE ADRESSE ZEILENANFANG
0672	3E 20	LD	A,20H	:	ZEICHEN ZUM LOESCHEN
0674	D5	PUSH	DE	:	AUFHEBEN OFFSET
0675	C5	PUSH	BC	:	AUFHEBEN COUNTER
0676	CD 05D6	CALL	DOERA	:	LOESCHE IN TEXTMEMORY
0679	C1	POP	BC	:	RUECKLADEN RESTCOUNTER
067A	D1	POP	DE	:	RUECKLADEN OFFSET
067B	21 F800	LD	HL,ARAMAD	:	ATTRIBUTSPEICHER
067E	19	ADD	HL,DE	:	AKTUELLE POSITION
067F	3E 07	LD	A,07H	:	ATTRIBUT ZUM LOESCHEN
0681	CD 05D6	CALL	DOERA	:	LOESCHE ATTRIBUTSPEICHER
0684	F1	POP	AF	:	RÜCKLADEN SPALTE
0685	32 03AB	LD	(SPAL),A	:	FUER AKTUELLE POSITION
0688	C3 0538	JP	ESCND	:	UND SEQUENZ ENDE
				;	ERASE FROM BEGINNING OF LINE

0688	3A 03AB	COERBL: LD	A, (SPAL)	: AUFHEBEN SPALTE
068E	F5	PUSH	AF	: WEGEN VIDADD
068F	AF	XOR	A	: LOESCHEN SPALTE
0690	32 03AB	LD	(SPAL), A	: FUER ZEILENANFANG
0693	CD 0849	CALL	VIDADD	: OFFSET VIDEOAM IN HL
0696	EB	EX	DE, HL	: NACH DE FUER ZEILENANFANG
0697	F1	POP	AF	: RUECKLADEN SPALTE
0698	32 03AB	LD	(SPAL), A	: WIEDERHERSTELLEN
0698	06 00	LD	B, 0	: SPALTENPOSITION IST COUNTER
069D	4F	LD	C, A	: NACH COUNTERREGISTER FUER LDIR
069E	21 25CA	LD	HL, VIDPUF	: PUFFERANFANG
06A1	19	ADD	HL, DE	: AKTUELLE ADRESSE
06A2	3E 20	LD	A, 20H	: ZEICHEN ZUM LOESCHEN
06A4	D5	PUSH	DE	: AUFHEBEN OFFSET
06A5	C5	PUSH	BC	: AUFHEBEN COUNTER
06A6	CD 05D6	CALL	DOERA	: LOESCHE IN TEXTMEMORY
06A9	C1	POP	BC	: RUECKLADEN RESTCOUNTER
06AA	D1	POP	DE	: RUECKLADEN OFFSET
06AB	21 37CA	LD	HL, VIDPUA	: ATTRIBUTSPEICHER
06AE	19	ADD	HL, DE	: AKTUELLE POSITION
06AF	3E 07	LD	A, 07H	: ATTRIBUT ZUM LOESCHEN
06B1	D5	PUSH	DE	: DISTANZ
06B2	C5	PUSH	BC	: COUNTER
06B3	CD 05D6	CALL	DOERA	: LOESCHE ATTRIBUTSPEICHER
06B6	C1	POP	BC	: COUNTER
06B7	D1	POP	DE	: DISTANZ
06B8	21 F000	LD	HL, VRAMAD	: VIDEOAMADRESSE
06B8	19	ADD	HL, DE	: PLUS DISTANZ IST AKTUELLE ADRESSE
06BC	3E 20	LD	A, 20H	: ZEICHEN ZUM LOESCHEN
06BE	D5	PUSH	DE	: AUFHEBEN OFFSET
06BF	C5	PUSH	BC	: AUFHEBEN COUNTER

06C0	CD	05D6	CALL	DOERA	; LOESCHE IN TEXTMEMORY
06C3	C1		POP	BC	; RUECKLADEN RESTCOUNTER
06C4	D1		POP	DE	; RUECKLADEN OFFSET
06C5	21	F800	LD	HL, ARAMAD	; ATTRIBUTSPEICHER
06C8	19		ADD	HL, DE	; AKTUELLE POSITION
06C9	3E	07	LD	A, 07H	; ATTRIBUT ZUM LOESCHEN
06CB	CD	05D6	CALL	DOERA	; LOESCHE ATTRIBUTSPEICHER
06CE	C3	0538	JP	ESSEND	; SEQUENZ ENDE
					; INSERT BLANK LINE
					; NACHHER IST SPALTE NULL
06D1	3E	00	COINSL: LD	A, 0	; UND WIR FÜGEN GANZE ZEILE EIN
06D3	32	03AB	LD	(SPAL), A	; OFFSET ZEILENANFANG NACH HL
06D6	CD	0849	CALL	VIDADO	; UND AUFHEBEN IN DE
06D9	EB		EX	DE, HL	; LAENGE DES BILDSCHIRMS IN ZEICHEN
06DA	21	0780	LD	HL, SCLNG	; (LOESCHEN CARRY)
06DD	B7		OR	A	; - AKTUELLER ZEILENANFANG IST RESTCOUNT HL
06DE	ED	52	SBC	HL, DE	; EINE ZEILE WENIGER SCROLLEN
06E0	01	0050	LD	BC, SPMAX	; DESWEGEN SPALTENBREITE ABZIEHEN
06E3	ED	42	SBC	HL, BC	; AUFHEBEN OFFSET
06E5	D5		PUSH	DE	; SCROLLCOUNTER
06E6	E5		PUSH	HL	; NACH COUNTERREGISTER
06E7	C1		POP	BC	; UND WIEDER AUFHEBEN
06E8	C5		PUSH	BC	; NOCHMAL DISTANZ AUFHEBEN
06E9	D5		PUSH	DE	; (SPMAX+1) ; EINE ZEILE
06EA	21	2CF9	LD	HL, VIDPUF+SCLNG-	; VERSATZ IST EIN
06ED	11	2D49	LD	DE, VIDPUF+SCLNG-1	; SCROLL TEXT
06F0	ED	B8	LDDR		; OFFSET FUER ZEILE LOESCHEN
06F2	D1		POP	DE	; ADRESSE ZEILENANFANG
06F3	21	25CA	LD	HL, VIDPUF	; DURCH PUFFERADRESSE + DISTANZ
06F6	19		ADD	HL, DE	; UND WIEDER AUFHEBEN
06F7	E5		PUSH	HL	; IN DE
06F8	D1		POP	DE	

```

06F9 13      INC      ; WEGEN LDIR IST DESTINATION EINS WEITER
06FA 3E 20    LD      ; LEERZEICHEN ZUM LOESCHEN
06FC 01 0050 LD      ; GANZE ZEILE
06FF 08      DEC     ; -1 WEGEN LDIR
0700 77      LD      ; 1. ZEICHEN FUER SOURCE
0701 ED 80    LDIR   ; LOESCHEN ZEILENREST
0703 C1      POP     ; RUECKLADEN COUNTER
0704 D1      POP     ; RUECKLADEN OFFSET
;
; ATTRIBUTRAM AUCH SCROLLEN
;
0705 D5      PUSH    ; WIEDER AUFHEBEN FUER LOESCHEN
0706 21 3EF9  HL,VIDPUA+SCLNG-(SPMAX+1)
0709 11 3F49  LD      ;
070C ED 88    LDDR   ; SCROLL TEXT
070E D1      POP     ; OFFSET
070F 21 37CA  LD      ; PUFFERANFANG
0712 19      ADD     ; PLUS OFFSET
0713 E5      PUSH    ; ERGIBT ZEICHEN ADRESSE
0714 D1      POP     ; IN DESTINATIONREGISTER
0715 13      INC     ; WIEDER PLUS 1 WEGEN LDIR
0716 3E 07    LD      ; LEERZEICHEN ZUM LOESCHEN
0718 01 0050 LD      ; GANZE ZEILE
;
071B 08      DEC     ; - 1 FUER LDIR
071C 77      LD      ; DA HIER EXPLIZIT EIN ZEICHEN GELÖSCHT WIRD
071D ED 80    LDIR   ; LOESCHEN ZEILENREST
071F CD 0346  CALL    ; ANZEIGEN VIDEOGRAM
0722 C3 0538  JP      ; ENDE SEQUENZ
;+++++++
CODELL: LD      ; DELETE LINE
;
; SPALTE NULL

```

0727	32	03AB	LD	(SPAL),A	; NACH AUSGEFÜHRTER OPERATION
072A	CD	0849	CALL	VIDADD	; OFFSET ZEILENANFANG NACH HL
072D	EB		EX	DE,HL	; WIE GEWOHNT IN DE
072E	21	0780	LD	HL,SCLNG	; LAENGE DES BILDSCHIRMS IN ZEICHEN
0731	B7		OR	A	; (LOESCHEN CARRY)
0732	ED	52	SBC	HL,DE	; - AKTUELLE POSITION IST RESTCOUNT IN HL
0734	01	0050	LD	BC,SPMAX	; EINE ZEILE WENIGER SCROLLEN
0737	ED	42	SBC	HL,BC	; DESHALB SPALTENZAHL ABZIEHEN
0739	D5		PUSH	DE	; AUFHEBEN OFFSET
073A	E5		PUSH	HL	; COUNTER UMLADEN
073B	C1		POP	BC	; IN COUNTERREGISTER
073C	21	25CA	LD	HL,VIDPUF	; PUFFERADRESSE
073F	19		ADD	HL,DE	; PLUS OFFSET IST DESTINATION
0740	C5		PUSH	BC	; AUFHEBEN COUNTER
0741	E5		PUSH	HL	; AUFHEBEN DESTINATION
0742	11	0050	LD	DE,SPMAX	; QUELLE IST EINE ZEILE WEITER
0745	19		ADD	HL,DE	; QUELLE JETZT IN HL
0746	D1		POP	DE	; ZEILE HOEHER DESTINATIONREGISTER
0747	ED	B0	LDIR		; SCROLL TEXT
0749	21	2CFA	LD	HL,VIDPUF+SCLNG-SPMAX	
074C	01	0050	LD	BC,SPMAX	; LETZTE ZEILE LOESCHEN
074F	E5		PUSH	HL	; DAS SPIEL WIE OBEN WEGEN LDIR
0750	D1		POP	DE	; SO BEKI'WIR DE UND HL AUF GLEICHEN STAND
0751	13		INC	DE	
0752	3E	20	LD	A,20H	; LEERZEICHEN ZUM LOESCHEN
0754	08		DEC	BC	; ZAHLER - 1
0755	77		LD	(HL),A	; 1. ZEICHEN EXPLIZIT
0756	ED	B0	LDIR		; LOESCHEN ZEILENREST
0758	C1		POP	BC	; RUECKKLADEN COUNTER
0759	D1		POP	DE	; RUECKKLADEN OFFSET
					; ATTRIBUTRAM AUCH SCROLLEN

```

075A      21 37CA      LD      HL,VIDPUA      ;
075D      19          ADD      HL,DE          ; ANFANG SCROLL DOWN IN HL
075E      E5          PUSH     HL            ;
075F      11 0050      LD      DE,SPMAX      ; QUELLE IST EINE ZEILE WEITER
0762      19          ADD      HL,DE          ;
0763      01          POP      DE            ; ZEILE HOEHER DESTINATIONREGISTER
0764      ED 80        LDIR     ; SCROLL TEXT
0766      21 3EFA      LD      HL,VIDPUA+SCLNG-SPMAX ;
0769      01 0050      LD      BC,SPMAX      ; EINE GANZE ZEILE
076C      E5          PUSH     HL            ;
076D      01          POP      DE            ;
076E      13          INC      DE            ;
076F      3E 07        LD      A,07          ; LEERZEICHEN ZUM LOESCHEN
0771      0B          DEC      BC            ; IST INZWISCHEN HINREICHEND BEKANNT
0772      77          LD      (HL),A         ;
0773      ED 80        LDIR     ; LOESCHEN ZEILENREST
0775      C0 0346      CALL     VIDDIS       ; VIDEOGRAM ANZEIGEN
0778      C3 0538      JP      ESCEND        ; ENDE SEQUENZ

;+++++++;
077B      C0 0849      CODELC: CALL VIDADDD  ; DELETE ONE CHARACTER
077E      EB          EX      DE,HL         ; OFFSET VIDEOGRAM IN HL
077F      3E 50        LD      A,SPMAX      ; NACH DE
0781      21 03AB      LD      HL,SPAL      ; ZEILENLAENGE IN ZEICHEN
0784      96          SUB      (HL)         ; ZEILE BEWEGEN DAFUER
0785      3D          DEC      A            ; RESTCOUNTER NACH A
0786      06 00        LD      B,0          ; RESTCOUNT -1
0788      4F          LD      C,A           ; NACH COUNTERREGISTER
0789      21 25CA      LD      HL,VIDPUF    ; PUFFERANFANG
078C      19          ADD      HL,DE        ; AKTUELLE ADRESSE
078D      D5          PUSH     DE           ; AUFHEBEN OFFSET

```

```

078E      C5      PUSH      BC      ; AUFHEBEN COUNTER
078F      E5      INC       HL      ; SOURCE IST EIN BYTE WEITER
0790      23      INC       HL      ; SOURCE IN HL
0791      D1      POP       DE      ; DESTINATION REGISTER
0792      ED 80   LDIR      ; MOVE LINE
0794      3E 20   LD        A,20H  ; BLANK ALS LETZTES ZEICHEN DER ZEILE
0796      2B      DEC       HL      ; WAR EINS ZU WEIT NACH LDIR
0797      77      LD        (HL),A  ; HL ZEIGT AUF LETZTE POSITION
;
;      ; ATTRIBUTERAM AUCH BEHANDELN
;
0798      C1      POP       BC      ; RUECKLADEN RESCOUNTER
0799      D1      POP       DE      ; RUECKLADEN OFFSET
079A      21 37CA LD        HL,VIDPUA ; ATTRIBUTSPEICHER
079D      19      ADD       HL,DE   ; AKTUELLE POSITION
079E      E5      PUSH      HL      ; SOURCE IST EIN BYTE WEITER
079F      23      INC       HL      ; SOURCE IN HL
07A0      D1      POP       DE      ; DESTINATION REGISTER
07A1      ED 80   LDIR      ; MOVE LINE
07A3      3E 07   LD        A,7H   ; BLANK ALS LETZTES ZEICHEN DER ZEILE
07A5      2B      DEC       HL      ; WAR EINS ZU WEIT WEGEN LDIR
07A6      77      LD        (HL),A ; HL ZEIGT AUF LETZTE POSITION
07A7      CD 0346 CALL      VIDDIS ; ANZEIGE VIDEOGRAM
07AA      C3 0538 JP        ESCEND ; ENDE ESCAPE SEQUENZ
;+++++++
;COCUEB: LD      A,0      ; MERKE IN FLAG WAS WAR
;          LD      (CURENB),A ; UND SONST
;          JP      ESCEND ; NICHTS WEITER
;+++++++
;COCUEB: LD      A,1      ; SETZE FLAG CURSOR OFF
;          LD      (CURENB),A ;

```

```

078A C3 0538 JP ESCEND ; ENDE SEQUENZ
;+++++++;
COCUSA: LD A,(SPAL) ; SAVE CURSOR POSITION
078D 3A 03AB LD (SPSAV),A ; AUFHEBEN
07C0 32 03AC LD A,(ZEIL) ; SPALTE
07C3 3A 03AA LD (ZESAV),A ; AUFHEBEN
07C6 32 03AD LD ESCEND ; ZEILE
07C9 C3 0538 JP ESCEND ; SEQUENZ ENDE
;+++++++;
COCURE: LD A,(SPSAV) ; RESTORE CURSOR
07CC 3A 03AC LD (SPAL),A ; RUECKLADEN
07CF 32 03AB LD A,(ZESAV) ; SPALTE
07D2 3A 03AD LD (ZEIL),A ; RUECKLADEN
07D5 32 03AA LD CURUP ; ZEILE
07D8 CD 037E CALL CURUP ; AKTUALISIEREN CURSOR
07DB C3 0538 JP ESCEND ; ENDE SEQUENZ
;+++++++;
CORVON: LD A,(ATTRB) ; REVERSE ON
07DE 3A 03AE LD (ATTRB) ; ATTRIBUT FELD
07E1 F6 80 OR REVMSK ; ZUM SCHREIBEN VON ZEICHEN
07E3 32 03AE LD (ATTRB),A ; MIT REVERSE ON VERSORGEN
07E6 C3 0538 JP ESCEND ; ENDE SEQUENZ
;+++++++;
CORVOF: LD A,(ATTRB) ; REVERSE OFF
07E9 3A 03AE LD (ATTRB) ; REVERSE WIEDER
07EC F6 80 OR REVMSK ; AUS ATTRIBUTFELD
07EE EE 80 XOR REVMSK ; ENTFERNEN
07F0 32 03AE LD (ATTRB),A ; UND AUFHEBEN ZUM SCHREIBEN
07F3 C3 0538 JP ESCEND ; ESCAPE ENDE
;+++++++;
COINON: LD A,(ATTRB) ; INTENSITY ON
07F6 3A 03AE LD (ATTRB) ; INTENSITY WIE REVERSE BEHANDELT
07F9 F6 80 OR REVMSK
07FB 32 03AE LD (ATTRB),A
07FE C3 0538 JP ESCEND

```

```

0801      3A 03AE      ;+++++ INTENSITY OFF
0804      F6 80        COINOF: LD A,(ATTRB) ; WIE REVERSE
0806      EE 80        OR REVMSK
0808      32 03AE      XOR REVMSK
080B      C3 0538      LD (ATTRB),A
                        JP ESCEND
;+++++ BLINK ON
080E      3A 03AE      COBLON: LD A,(ATTRB) ; BLINKEN SETZEN
0811      F6 40        OR BLKMSK ; IN ATTRIBUT FELD
0813      32 03AE      LD (ATTRB),A ; UND AUFHEBEN ZUM SCHREIEBN
0816      C3 0538      JP ESCEND ; ENDE ESCAPE
;+++++ BLINK OFF
0819      3A 03AE      COBLOF: LD A,(ATTRB) ; BLINK MODE
081C      F6 40        OR BLKMSK ; ZURÜCKSETZEN
081E      EE 40        XOR BLKMSK ; IN ATTRIBUTFELD
0820      32 03AE      LD (ATTRB),A ; ZUM SCHREIBEN
0823      C3 0538      JP ESCEND ; ENDE ESCAPE
;+++++ LINE WRAP ON
0826      3E 00        COMRON: LD A,0 ; SETZEN FLAG
0828      32 03AF      LG (WRAP),A ; FUER WRAP ON
082B      C3 0538      JP ESCEND ; ENDE ESCAPE
;+++++ LINE WRAP OFF
082E      3E 01        COMROF: LD A,1 ; SETZE FLAG
0830      32 03AF      LD (WRAP),A ; FUER WRAP OFF
0833      C3 0538      JP ESCEND ; ENDE ESCAPE
;+++++ RESTORE ATTRIBUT
0836      3E 07        COATTR: LD A,7 ; SETZE NORMAL VIDEO
0838      32 03AE      LD (ATTRB),A ; IN ATTRIBUTFELD
083B      3E 00        LD A,0 ; UND CURSOR
083D      32 03AB      LD (SPAL),A ; HOME
0840      32 03AA      LD (ZEIL),A ;

```

```

0843 CD 037E                                ; CALL CURUP      ; AKTUALISIERE CURSOR
0846 C3 0538                                ; JP ESCEND      ; ENDE
;*****
; BERECHNEN AKTUELLE VIDEOCONTROLLER ADRESSE
;*****
; AUSGANGSPARAMETER
;*****
; DISTANZ IM VIDEOGRAM
;*****
VIDADD: LD A, (SPAL)      ; HOLEN SPALTE
        LD E, A           ; AUFHEBEN SPALTE
        LD A, (ZEIL)      ; ZEILE LADEN
        LD B, A           ; MULTIPLIZIERE
        SLA B             ; MIT 2
        ADD A, B           ; ZEILE * 3
        ADD A, B           ; ZEILE * 5
        OR A              ; CLEAR CARRY
        LD B, 0           ; HIGH BYTE NOCH NULL
        RLA              ; SCHIFT AKKU
        RL               ; GET OVERFLOW
        RLA              ; INSGESAMT MAL 16
        RL               ; FORMEL :
        RL               ; ZEILE * 80 + SPALTE
        AND OFOH         ; ERGEBNIS IN A UND B
        LD H, B           ; LETZTE TETRADE INTERESSIERT NICHT
        LD L, A           ; 16 BIT DISTANZ IN VIDEOGRAM
        LD D, 0           ; SPALTE ADDIEREN
        ADD HL, DE        ; FERTIG
        RET
0849 3A 03AB                                ;
084C 5F                                     ;
084D 3A 03AA                                ;
0850 47                                     ;
0851 C8 20                                ;
0853 80                                     ;
0854 80                                     ;
0855 B7                                     ;
0856 06 00                                ;
0858 17                                     ;
0859 C8 10                                ;
085B 17                                     ;
085C C8 10                                ;
085E 17                                     ;
085F C8 10                                ;
0861 17                                     ;
0862 C8 10                                ;
0864 E6 F0                                ;
0866 60                                     ;
0867 6F                                     ;
0868 16 00                                ;
086A 19                                     ;
086B C9                                     ;

```



```

08CE 3E 07          LD      A,07H          ; NORMAL VIDEO
08D0 06 50          LD      B,SPMAX        ; JEWEILS EINE
08D2 77             LD      (HL),A         ; GANZE ZEILE
08D3 23             INC     HL
08D4 10 FC          DJNZ    CONSCA
08D6 CD 0346        CALL    VIDDIS
08D9 C9             RET

;*****
;
; INITIALISIEREN VIDEOAM
;
;*****
08DA 21 18CA        VIDINI: LD      HL,VIDANF
08DD 01 1C00        LD      BC,0800H+0A00H+0A00H
08E0 1E 20          LD      E,20H
08E2 73             LD      (HL),E
08E3 23             INC     HL
08E4 08             DEC     BC
08E5 78             LD      A,B
08E6 B1             OR      C
08E7 20 F9          JR      NZ,VIDI1
08E9 21 37CA        LD      HL,VIDPUA
08EC 01 0940        LD      BC,940H
08EF 3E 07          LD      A,07
08F1 5F             LD      E,A
08F2 73             LD      (HL),E
08F3 23             INC     HL
08F4 08             DEC     BC
08F5 78             LD      A,B
08F6 B1             OR      C
08F7 20 F9          JR      NZ,VIDI2

;*****
;
; PUFFERADRESSE
; LAENGE VIDEOAM
; LEERZEICHEN ZUM LOESCHEN
; LOESCHEN VIDEOAM
; NAECHSTES BYTE
; WEITER BIS GANZES RAM LEER
;
;
; ATTRIBUTBEREICH AUCH LOESCHEN
; LAENGE
; NORMAL VIDEO
;
; LOESCHEN VIDEOAM
; NAECHSTES BYTE
; WEITER BIS GANZES RAM LEER
;

```

```

08F9 3E 00      LD      A,0
08FB 32 03AA    LD      (ZEIL),A
08FE 32 03AB    LD      (SPAL),A
0901 C9        RET

```

HILFSROUTINE ZUR AUSGABE VON AKKU ALS HEX-ASCII

0902	F5	HEXOUT: PUSH	AF	;	AUFHEBEN ZEICHEN
0903	1F	RRA		;	LINKE TETRADE
0904	1F	RRA		;	IN RECHTE TETRADE
0905	1F	RRA		;	HOLEN
0906	1F	RRA		;	4 BIT VERSCHIEBEN
0907	CD 0908	CALL	HEX	;	UND UNTERE TETRADE AUSGEBEN
090A	F1	POP	AF	;	ZEICHEN ZURÜCKLADEN
0908	E6 0F	AND	0FH	;	UND JETZT RECHTE TETRADE UMWANDELN
090D	C6 90	ADD	A, 90H	;	DIESER KLEINE
090F	27	DAA		;	TRICK
0910	CE 40	ADC	A, S'	;	HAT SCHON
0912	27	DAA		;	MANCHEN STAUNEN LASSEN
0913	4F	LD	C, A	;	ZEICHEN IN C
0914	CD 0425	CALL	CONDIS	;	ZUR AUSGABE AUF DEN BILDSCHIRM
0917	C9	RET		;	ALLES OK

Macros:

Symbols:

ACK 0006	ADOK	0042*	ANZZEI	037C	ARAMAD	F800
ATTRB 03AE	BLKMSK	0040	CHKLOC	0279	COATTR	0836
COBELL 03ED	COBLOF	0819	COBLON	080E	COCUDB	07B5
COCUDO 053E	COCUEB	07AD	COCUHO	058A	COCULO	056D
COCULE 055E	COCUPO	054A	COCURE	07CC	COCURG	0544
COCURI 0465	COCUSA	07BD	COCUUP	0550	CODELC	0778
CODELL 0725	COERES	0581	COERBL	068B	COERBS	0624
COEREL 05E4	COERES	05AE	COERWL	0644	COINOF	0801
COINON 07F6	COINSL	06D1	COMNOT	0254	COMNOW	0271
COMOUT 01B2	COMTW	0245	COMWS2	0263	COMWSC	0267
CONASC 0414	CONASN	0478	CONCHR	03B7	CONCR	03F8
CONCUR 047F	CONCZE	0498	CONDI1	0424	CONDIS	0425
CONESC 04A6	CONESS	03F2	CONIN	02DC	CONLF	0400
CONLFB 040C	CONOUT	02E0	CONSC1	08C7	CONSCA	08D2
CONSCL 088B	CONSCR	086C	CONSCW	0879	CONSCX	0896
CONSON 0413	CONST	02D8	COREVI	0598	CORVOF	07E9
CORVON 07DE	CORVS	05A8	COWROF	082E	COWRON	0826
CPADDR 0100	CPNOSB	00F0	CR	000D	CRTIN	E40F
CRTOUT E412	CRTST	E40C	CURAK	0384	CURENB	03B2
CURFLG 03B1	CURUP	037E	DOERA	05D6	ENQ	0005
ESC 001B	ESCEND	0538	ESCFLG	0380	F1	0080
F6 00F2	FALSE	0000	HEX	090B	HEXOUT	0902
INCHR 016F	INSTAT	018A	LF	000A	LOCFLG	03B4
LOWLIM 0100	MELANF	0918	MELSEQ	0EEF	MESANF	02B6

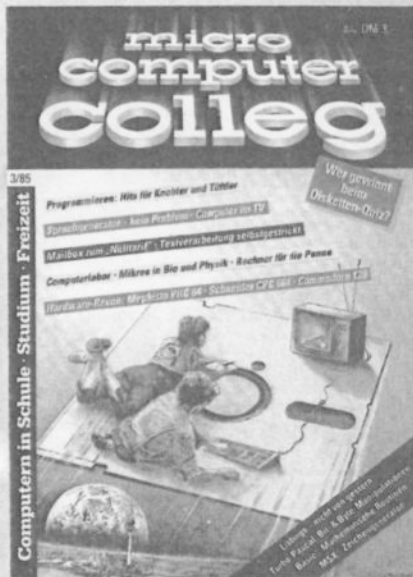
MESLUS	02CC	MESLES	02BC	NACK	0015	NETANF	1346
NETEND	1B46	NETPI4	1344	NETP04	1342	NLOCT	029D
NOSPK	022D	PAGE	03B3	PIC70	0070	PIC71	0071
PROGST	47CA	PROM	FFFF	RECINT	02E7	RECJMP	02E4
RECNO	0313	RECNOX	0325	RECNOY	0337	REVL0D	08C0
REVMSK	0080	REVNPA	08B3	REVSCR	089E	RTSFLG	0345
RXMSK	00FD	SCCHR	01AC	SCLNG	0780	SCREEN	0196
SCRHA	0368	SCRHN	034F	SCRHX	0440	SCRHY	0454
SFF0	E468	SFF1	E469	SPAL	03AB	SPMAX	0050
SPORT	0010	SPORT2	0020	SPSAV	03AC	SPSAVS	1BC8
STACK	1B88	STACK2	1B48	START	0100	TRAFGL	03B5
TRUE	FFFF	TXTRD	02AA	TXTST	0278	USARTC	0041
USARTD	0040	VIDADD	0849	VIDANF	1BCA	VIDDIS	0346
VIDI1	08E2	VIDI2	08F2	VIDINI	08DA	VIDPUA	37CA
VIDPUF	25CA	VIDRES	2DCA	VRAMAD	F000	WORKFL	03B6
WRAP	03AF	ZEIL	03AA	ZESAV	03AD	ZMAX	0018

No Fatal error(s)

DIE ZWEI



Die Zeitschrift für den
Microcomputeranwender
im Beruf
für Angestellte,
Selbständige und Freiberufler



Die Zeitschrift für den
Microcomputeranwender
im Bildungsbereich
für
Lehrer, Schüler und Studenten

SUUM CUIQUE...

• Kompetent und zielgruppensicher •

Bertelsmann Fachzeitschriften GmbH
Dingolfingerstr. 4 · 8000 München 80

Gutschein zum Prüfen
Senden Sie mir bitte ein Ansichtsexemplar
☐ micro ☐ micro computer colleg

Name, Vorname
Straße, Postfach
PLZ Ort

PC-A