Handbuch 128K

ENTERPRISE®
COMPUTERS

PRÄDIKAT: DER ZUKUNFTSSICHERE
mit benutzerfreundlicher Peripherie und Software

INHALT

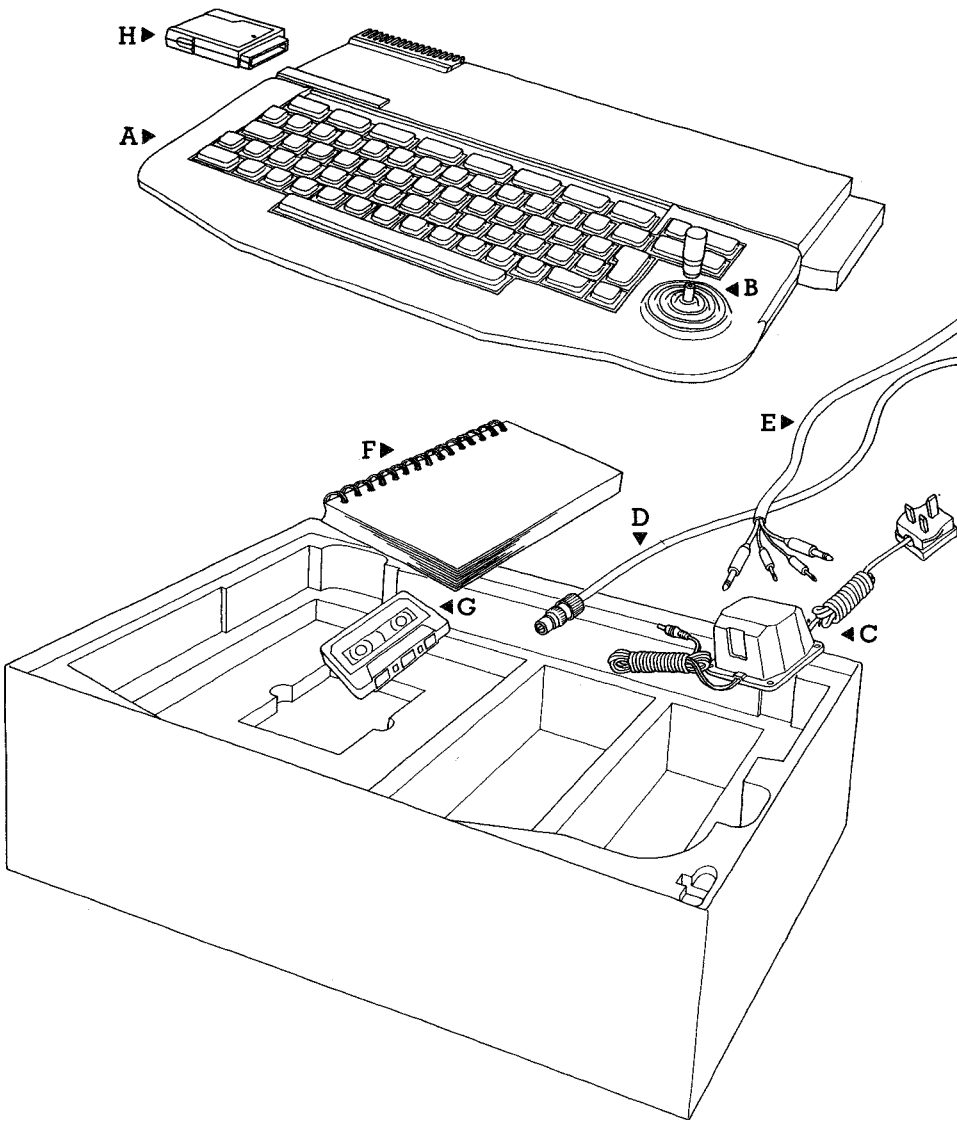
Die einzelnen Bestandteile	4
Der Enterprise	6
Das Einstellen des Fernsehers	9
Die Anwendung der Programme	11
Die Tastatur	12
Der Gebrauch des Cassettenrecorders	14
Der Gebrauch von Programmsteckmodulen	15
Hilfe!!!	16

EINFÜHRUNG	19
Einleitung	20
So starten Sie Ihren Enterprise-Computer	21
Erste Programme	31
Über das Programmieren	43
Variablen	49
Operatoren und Ausdrücke	55
Gestaltung von Texten	59
Editieren von Programmen	63
Textbearbeitung	69
Die Funktionstasten	74
Programme auf Cassetten	76

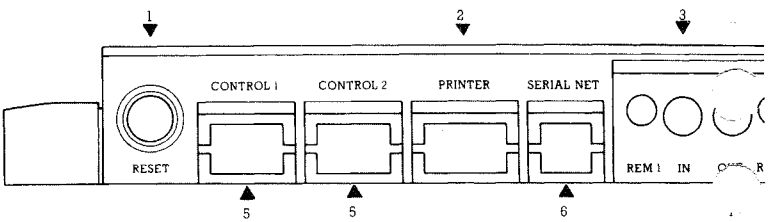
ÜBUNGSTEIL	81
Strings	82
Schleifen	89
Vergleiche mit IF/THEN	94
Speichern größerer Datenmengen	100
Definition von Funktionen	107
Graphik	117
Der Zeichensatz	131
Klang und Rhythmus	134
Vom Problem zum Programm	142
Minimal-Basic	150
Logische Kanäle	155
Ausnahmefälle	157
Das intelligente Netzwerk	159
Maschinensprache	163

NACHSCHLAGETEIL 167
Basic-Regeln 168
Programmierhilfen 172
Befehle 177
*Trigonometrische Funktionen 179
*Mathematische Funktionen 180
*String-Funktionen 182
*Andere Funktionen 184
Strukturen und Schleifen 185
Entscheidungen und Verzweigungen 187
Unterprogramme in Basic 190
Fehler und Interrupt-Behandlung 194
Variablen 197
Daten Ein/und Ausgabe 198
Logische Kanäle 204
Laden und Speichern 207
Graphik 210
Videooptionen 212
Text und Graphik-Fenster 217
Sound 218
Editor/Textbearbeitung 221
Betriebssystem 223
Maschinensprache 225
Serielle Schnittstelle 227
Exos 229
Fehlermeldungen 233
Glossar 238
Kurzreferenz 251

Diese Broschüre zeigt Ihnen, wie Sie die einzelnen Computerteile verbinden müssen, so daß Ihr Computer ordnungsgemäß funktioniert. Weiterhin wird Ihnen der Gebrauch von Programm-Kassetten und Programm-Steckmodulen erklärt. Im ENTERPRISE-Handbuch erhalten Sie alle Informationen über den Umgang mit dem Textverarbeitungsprogramm und über die BASIC-Programmiersprache.
Sie sollten diese Broschüre aufmerksam lesen, auch wenn Sie schon einmal mit einem Computer gearbeitet haben. Selbst bei leichter Handhabung des Computers können gelegentlich Fehler unterlaufen.
Nehmen Sie sich einige Minuten Zeit, prüfen Sie den Inhalt Ihres Computerpaketes und machen Sie sich damit vertraut.
Ihr Paket müßte beinhalten:
A Den ENTERPRISE-Computer
B Den Cursor-Steuerhebel
C Das Netzteil
D Das Fernsehkabel
E Das Recorderkabel
F Das ENTERPRISE-Handbuch
G Die Demonstrations-Kassette
H Das BASIC-Steckmodul
und die Garantie- und Wartungsbedingungen für Ihr Gerät. Die Garantiekarte enthält einen Abschnitt für den ENTERPRISE-Club. Tragen Sie dort Ihren Namen, Anschrift, die von Ihnen verwendeten Programme und Ihre derzeitige Geräte Konfiguration ein. Senden Sie diese Karte an die Clubadresse.
Wollen Sie nun mit Ihrem Programm arbeiten, so brauchen Sie einen Kassettenrecorder und alle Bestandteile wie auf Seite 5 abgebildet.


H
A
B
F
E
D
G
C

Alle Elemente des ENTERPRISE-Computers arbeiten so zusammen, daß eine neue Generation im Computerbereich entsteht. Die Kapazität dieses Systems läßt sich erweitern, so daß es in der Lage ist, größere EDV-Arbeiten zu übernehmen.
Durch eine der Schnittstellen besteht die Möglichkeit, mehrere ENTERPRISE-Computer über ein „Netz" miteinander zu verbinden.



1. Der RESET-Knopf bringt den Computer in die Anfangsstellung zurück.

2. Centronics Anschluß, über ein Kabel kann hier ein Drucker angeschlossen werden.

3. Hier wird der Kassettenrecorder angeschlossen.

4. Anschluß für einen Monitor oder hi-fi.

5. An diese beiden Steckbuchsen können externe Geräte angeschlossen werden, wie z.B. Joysticks.

6. RS 232 Anschluß für Drucker, Modem oder andere ENTERPRISE-Computer.

7. Anschluß für einen Kopfhörer oder Kassettenrecorder.

8. Anschluß für einen Fernseher.

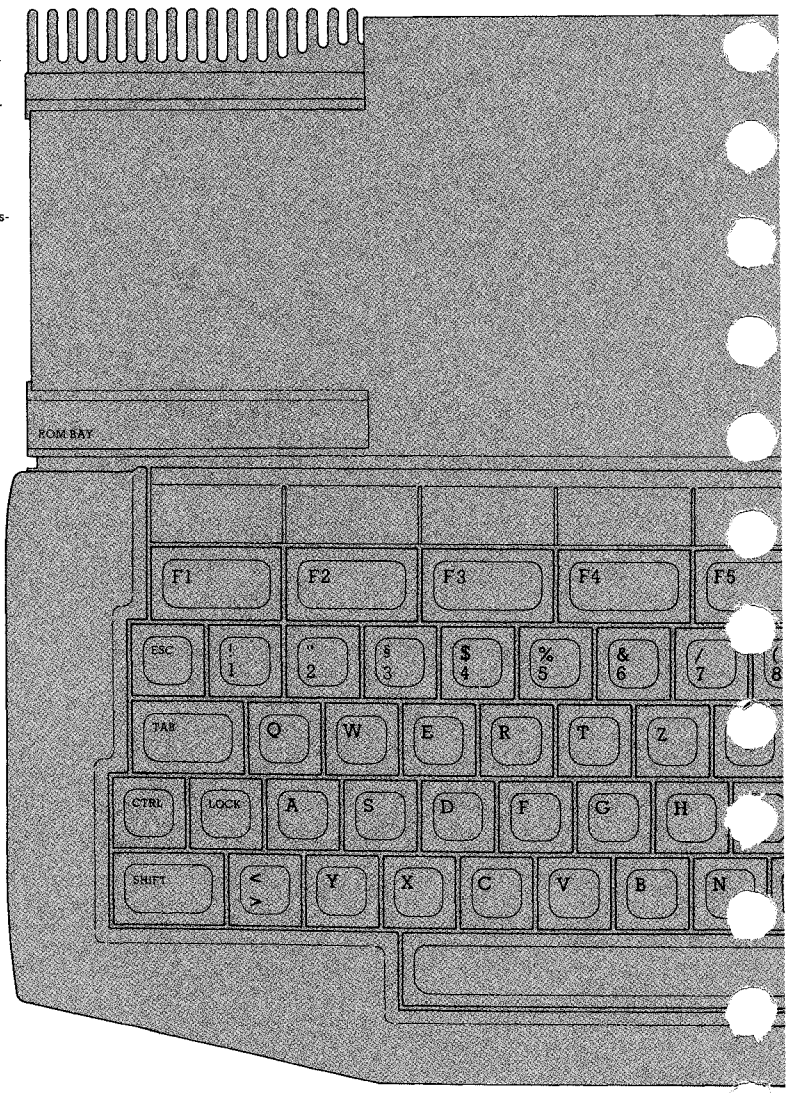
9. Reservierter Anschluß.

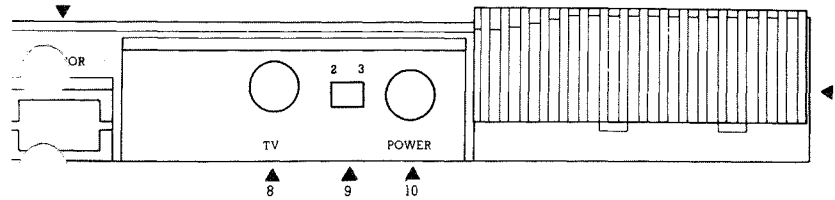
10. Stromversorgung.

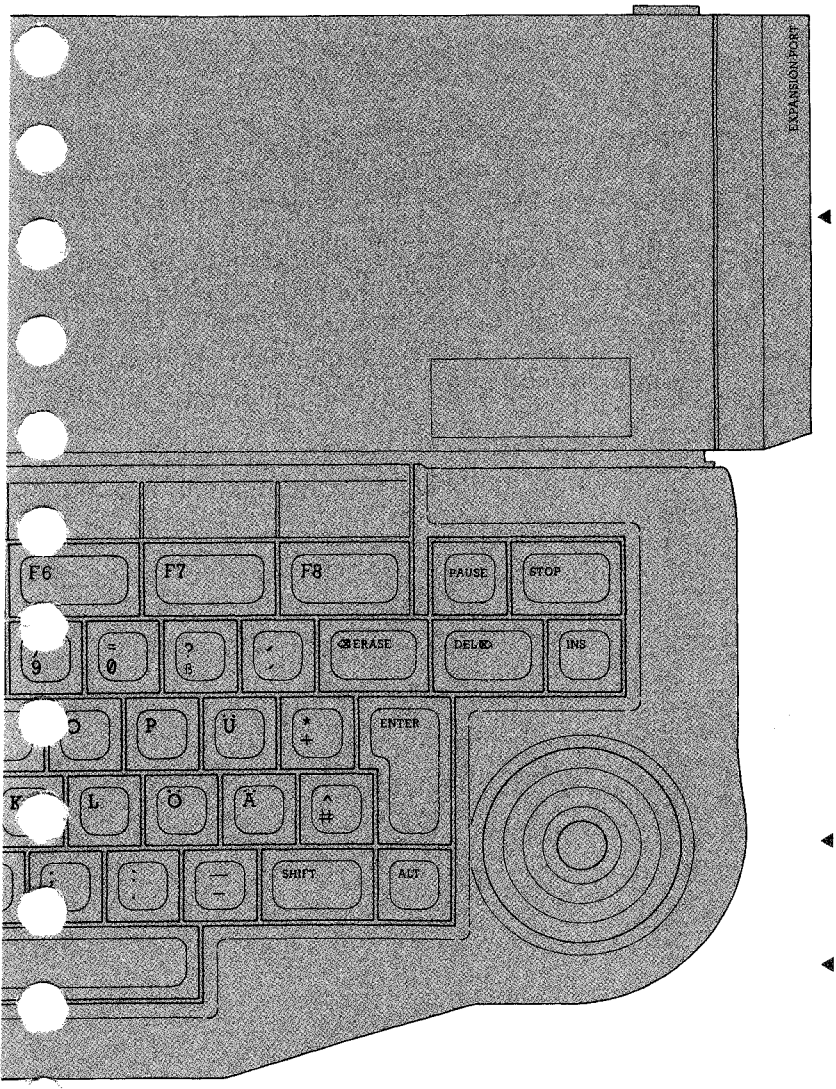
11. Schacht für Programm-Steckmodule.

Halterung für die Tastaturschablone ▶

Funktionstasten ▶



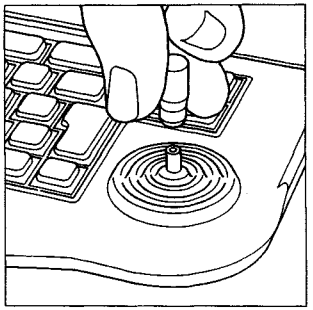


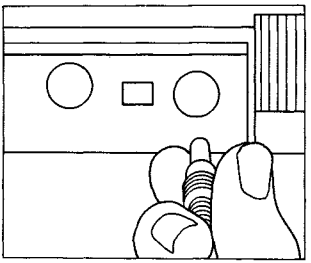


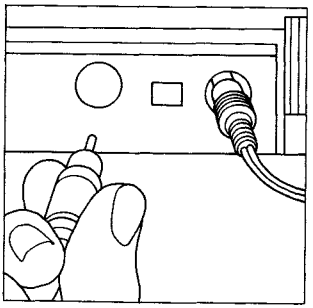
Schacht für Systemerweiterungen

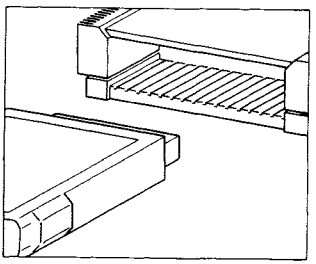
Mit dem Cursor-Steuerhebel wird der Cursor auf dem Bildschirm bewegt

Leertaste









Nun zum Aufbau Ihres Computers. Hier ist die richtige Reihenfolge zu beachten:

– Das Programm-Steckmodul IS-BASIC wird in den dafür vorgesehenen Schacht (ROM BAY) an der linken Seite Ihres Computers eingeschoben. Achten Sie darauf, daß das Steckmodul leicht einrastet.

– Setzen Sie den mitgelieferten Cursor-Steuerhebel auf die dafür vorgesehene Vorrichtung auf der rechten Seite der Tastatur.

– Auf der linken Rückseite Ihres Computers befindet sich eine Buchse, die mit „POWER" gekennzeichnet ist. Hier wird das Netzteil angeschlossen.

– Daneben befindet sich die TV-Buchse für den Anschluß an das Fernsehen. Verbinden Sie mit dem beiliegenden Fernsehkabel den ENTERPRISE-Computer mit dem Antenneneingang Ihres Fernsehers.

– Schalten Sie Ihren Fernseher ein. Stecken Sie das Netzteil des ENTERPRISE-Computers in die Steckdose. Eine kleine rote Kontrollampe leuchtet nun auf Ihrer Tastatur. Es erfolgt eine Selbstüberprüfung des Computers, die einige Sekunden dauert.

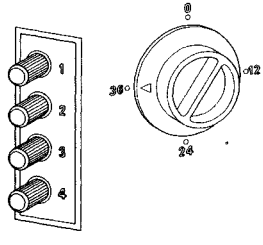
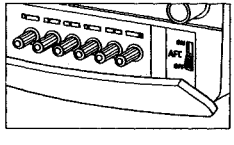
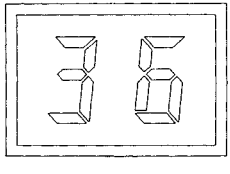
DAS EINSTELLEN DES FERNSEHERS

Ihr Computer ist nun angeschlossen. Als nächstes stellen Sie Ihren Fernseher so ein, daß er die Signale vom ENTERPRISE-Computer „versteht". Der ENTERPRISE-Computer sendet sein Bild für den Fernseher auf dem UHF Kanal 36. Wählen Sie eine freie Programmtaste. Drehen Sie dann solange den Knopf für die Feineinstellung, bis folgendes Bild erscheint:

ENTERPRISE
© 1985 Intelligent Software Ltd

Drücken Sie eine beliebige Taste auf Ihrer Tastatur. Sie sind jetzt im Basic. Nun können Sie Ihrem Computer Anweisungen geben. Sollte nicht Basic sondern die englische Textbearbeitung auf den Bildschirm erscheinen, so prüfen Sie bitte ob das Steckmodul richtig eingerastet ist.

VERSCHIEDENE MÖGLICHKEITEN DEN FERNSEHER AUF DEN COMPUTER EINZUSTELLEN

Einstellung	Ihr Fernseher hat:	Was nun zu tun ist:
	Einen Drehknopf zur Kanalwahl	Drehen Sie den Knopf langsam, bis Sie ein klares Bild haben. (Ungefähr bei Kanal 36).
	Mehrere Knöpfe; die Kanalwahl erfolgt durch Drücken eines der Knöpfe.	Wählen Sie einen Kanal durch Herunterdrücken des Knopfes. Drehen Sie den Knopf solange bis Sie ein Bild bekommen.
	Die Feineinstellung erfolgt durch Drehen des gedrückten Knopfes.	Sollte nach fünf oder sechs Umdrehungen kein Bild erscheinen, drehen Sie den Knopf in die andere Richtung.
	Fernbedienung (1) die Kanalwahl erfolgt mit der Fernbedienung, kann jedoch auch durch Drehknöpfe erfolgen.	Wählen Sie Kanal 7 oder 8 mit Ihrer Fernbedienung. Öffnen Sie die Abdeckung; hier sind die Drehknöpfe für die Auswahl der Kanäle. Drehen Sie den entsprechenden Knopf, bis ein Bild erscheint.
	Fernbedienung (2) die Kanalwahl erfolgt durch die Fernbedienung.	Die Digitalanzeige des Fernsehers zeigt Ihnen, welchen Kanal Sie gerade benutzen. Wählen Sie Kanal 36 oder Video mit Ihrer Fernbedienung. Durch Drücken von „+" oder „-" erscheint ein klares Bild.

Sollten Sie damit Probleme haben, sehen Sie bitte in der Gebrauchsanweisung Ihres Fernsehers nach oder suchen Sie Ihren Fachhändler auf.

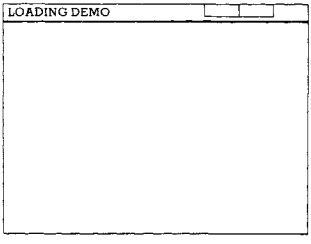
Stellen Sie Ihren Fernseher richtig ein, sonst kann es vorkommen, daß das Bild „flimmert". Die Buchstaben könnten „verschwommen" oder asymmetrisch sein; das Bild könnte von einer Seite zur anderen „wandern".
Es ist sehr einfach, diese Korrekturen am Fernseher vorzunehmen. In diesem Fall drehen Sie die Feineinstellung bis ein optimales Bild erscheint. Bitte vorsichtig; nicht daß Ihnen das Bild dabei „verloren" geht. Hat Ihr Fernseher einen AFC- oder einen AFT-Schalter, betätigen Sie diesen. Damit bleibt die richtige Frequenz erhalten.
Das ist alles was Sie wissen müssen, um Ihren ENTERPRISE-Computer zu installieren. Wenn Sie diesen Vorgang durchgeführt haben, wird Ihnen das sehr einfach vorkommen.

DER ANSCHLUSS EINES KASSETTEN-RECORDERS

Das Anschlußkabel für einen Kassettenrecorder hat zwei Stecker an jedem Ende. Führen Sie einen der größeren Stecker in die Steckbuchse, die mit „IN" bezeichnet ist, ein. Diese Buchse befindet sich auf der Rückseite des Computers. Den gleichfarbigen Stecker auf der anderen Seite des Kabels, führen Sie in die Buchse „EAR" (Kopfhörer) ein. Stecken Sie nun einen von den kleinen Steckern in die Buchse REM 1. Den Stecker auf der anderen Seite Ihres Kabels stecken Sie in die Buchse REM von Ihrem Kassettenrecorder. Die restlichen Stecker sind momentan noch ohne Gebrauch. Der EP Computer arbeitet nahezu mit allen gängigen Kassettenrecordern.

Nun haben Sie Ihren ENTERPRISE-Computer installiert und wollen damit arbeiten. Sie können sich selbst Ihre Programme schreiben, aber es gibt auch eine Vielzahl von fertigen Programmen, die Sie kaufen können. In der Grundausstattung Ihres Computers befindet sich eine Kassette. Der Gebrauch dieser Kassette ist sehr einfach. Bitte probieren Sie diese Kassette aus.

STARTEN DER DEMONSTRATIONS-KASSETTE



Legen Sie die Demonstrationskassette in Ihren Kassettenrecorder ein. Versichern Sie sich, daß die Kassette an den Anfang der ersten Seite gespult ist. Wenn die Rückfunktion Ihres Kassettenrecorders nicht funktioniert, ist der Motor des Recorders über die Fernbedienung blockiert. Betätigen Sie die Funktionstaste 4 (REMOTE 1) um diese Funktion zu stoppen.
Drücken Sie nun die START-Taste (Funktionstaste 1) auf Ihrer Tastatur und starten Sie Ihren Kassettenrecorder wie gewohnt.
Nach einigen Sekunden erscheint die Mitteilung „SEARCHING" auf dem Bildschirm. Kurz darauf wechselt diese Meldung in „LOADING xxxxxxxx" um. Wenn das Programm geladen ist, arbeitet es unter der Kontrolle des Computers automatisch weiter. Daß der Computer die Befehle von der Kassette „versteht", wird Ihnen durch ein Feld am oberen Rand des Bildschirms angezeigt. Dieses Feld kann rot oder grün sein oder es wechselt zwischen den beiden Farben. Wenn die Farbe der Zeile durchgehend rot ist, drehen Sie die Lautstärke Ihres Kassettenrecorders langsam herunter, bis die Farbe wechselt.

DEMO

xxxx ist der Name des Demonstrationsprogramms auf Ihrer Kassette. Jedes Programm auf der Kassette hat einen eigenen Namen. Diese Namen erscheinen immer auf dem oberen Bildschirmrand, so daß Sie immer wissen, mit welchem der Programme gerade gearbeitet wird.

FERNBEDIENUNG

Die Steckbuchsen REM 1 und REM 2 auf der Rückseite Ihres Computers ermöglichen es, den Kassettenrecorder zu bedienen. Wenn Ihr Kassettenrecorder einen REM-Eingang hat, kann er mit dem ENTERPRISE gestartet oder gestoppt werden.

DIE TASTATUR

Die Bedienung des Computers erfolgt fast immer über die Tastatur.
– Durch die Benutzung der Funktionstasten wird die Eingabe von Befehlen einfacher, z.B. die Starttaste (Funktionstaste 1) startet das erste Programm auf Ihrer Programm-Kassette.
– Die Eingabe von Großbuchstaben erfolgt durch das gleichzeitige Drücken der Tasten SHIFT und der gewünschten Buchstaben.
– Soll ein Zeichen, z.B. ein x, mehrfach hintereinander geschrieben werden, so halten Sie diese Taste solange gedrückt, bis eine neue Eingabe erfolgen soll.
– Mit der Leertaste stellen Sie Leerschritte dar.

DIE FUNKTIONSTASTEN: Diese Tasten ermöglichen Ihnen mit einem Tastendruck bestimmte Befehle an den Computer zu geben. Die Tasten haben festgelegte Funktionen, können jedoch individuell gewechselt werden.

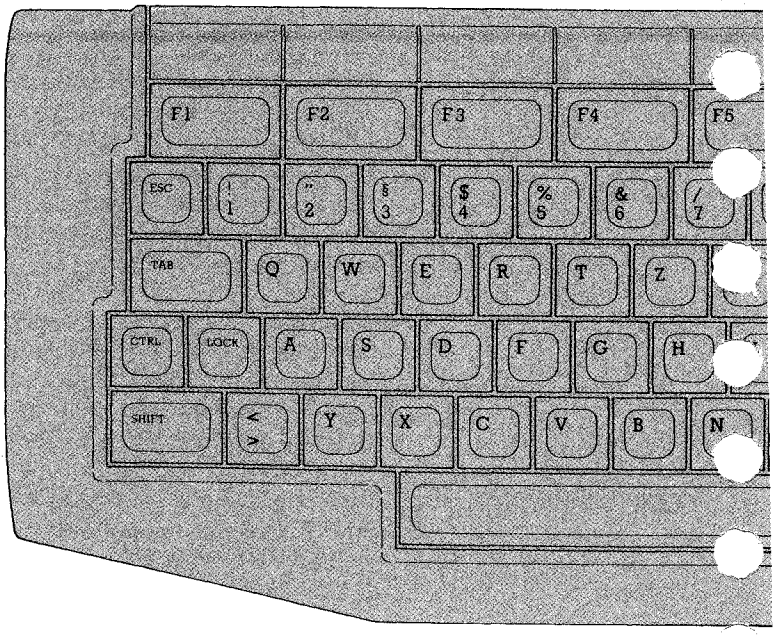
ESC: Die „ESCapetaste". Ein Programm kann Unterprogramme haben. Mit ESC gelangen Sie immer wieder zur nächsthöheren Programmstufe.

TAB: Durch die Benutzung von CTRL und der Funktionstaste 3 können Sie Tabulatorstops setzen. Mit der TABulatortaste können Sie nun von Stop zu Stop springen.

CTRL: Benutzen Sie diese Taste immer im Zusammenhang mit anderen Tasten um Zusatzfunktionen zu erhalten. Eine detaillierte Beschreibung finden Sie im ENTERPRISE-Handbuch.

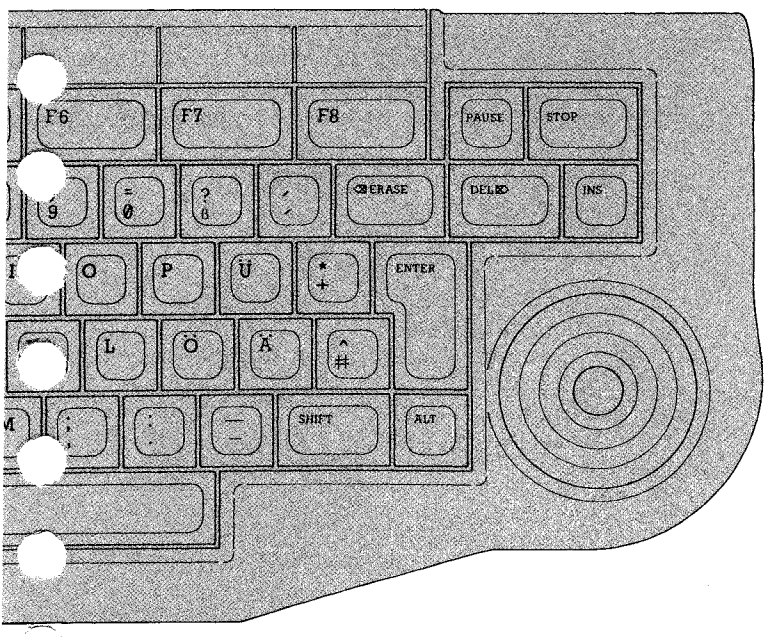
LOCK: Wenn Sie diese Taste gleichzeitig mit CTRL gedrückt halten, können Sie Ihre Tastatur auf Großbuchstaben umschalten. Drücken Sie LOCK um diese Funktion wieder abzuschalten. Siehe auch bei ALT und SHIFT.

SHIFT: BUCHSTABEN- und Zahlentasten werden auf dem Bildschirm so dargestellt wie sie auf der Schreibmaschinentastatur abgebildet sind. Mit der Taste SHIFT können Großbuchstaben und abgebildete Sonderzeichen dargestellt werden. Mit den Tasten CTRL oder ALT können zusätzliche Sonderfunktionen erreicht werden.



Für detaillierte Informationen sehen S

– Die ENTER-Taste ist eine Auslösetaste. Sie bestätigen mit ihr Eingaben und senden Befehle an den Computer.
– Mit dem Cursor-Steuerhebel wird der Cursor über den Bildschirm bewegt. Versuchen Sie es!



STOP: Stopt ein Programm. Durch Schreiben von CONTINUE wird die Arbeit fortgesetzt. Mit RUN wird das Programm neu gestartet.

ERASE: Löscht Zeichen von rechts nach links.

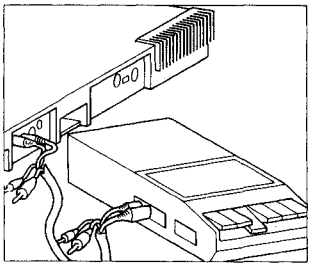
DEL: Löscht Zeichen von links nach rechts. ERASE oder DEL sind in der Lage in Verbindung mit CTRL ganze Blöcke von Text zu löschen.

INS: Im Normalfall werden Buchstaben und Zeichen einfach überschrieben. Wenn Sie die Taste INS im Zusammenhang mit CTRL betätigen, fügen Sie Zeichen ein. Nachfolgende, bereits geschriebene Buchstaben werden nach rechts vorgeschoben. Leerstellen fügen Sie ein, indem Sie nur die INS-Taste drücken.

ALT: In Verbindung mit anderen Tasten kann man eine Anzahl von Sonderzeichen darstellen.

LEERTASTE: Betätigen Sie diese Taste um Leerzeichen darzustellen.

bitte in das ENTERPRISE-Handbuch.



Damit Sie Ihre Programme benutzen können, schließen Sie einen gewöhnlichen Kassettenrecorder an den ENTERPRISE an. Der Computer holt sich die Programme in seinen Speicher (Laden der Programme). Sie können auch Programme aus dem Speicher auf Ihre Kassette spielen (speichern). Dafür sind keine speziellen Kassetten erforderlich. Gewöhnliche hi-fi Kassetten sind dafür gut geeignet. Im ENTERPRISE-Handbuch wird Ihnen der Vorgang des Speicherns genau erklärt.

DAS LADEN VON PROGRAMMEN

Es dauert nicht lange, bis ein Programm von einer Kassette in den Computer geladen ist. Bei großen Programmen kann das fünf Minuten dauern.
Hier noch ein paar Tips, die Sie beachten sollten:
Überprüfen Sie die Steckverbindungen. Wann immer Sie ein käuflich erworbenes Programmm – oder irgendein anderes z.B. selbstgeschriebenes Programm benutzen, machen Sie sich mit der fachgerechten Bedienung vertraut. Bei den gekauften Programmen befindet sich eine genaue Bedienungsanleitung.

PFLEGE DER KASSETTEN

Die Kassetten sollten vorsichtig behandelt werden, damit sie eine lange Lebensdauer haben; die folgende Liste zeigt Ihnen, was Sie mit den Kassetten nicht tun dürfen und was Sie tun sollten.
– Nicht in vollem Sonnenlicht liegen lassen
– Vor Feuchtigkeit schützen
– Vor Staub schützen
– Nicht mit den Fingern auf das Band fassen
– Nicht in die Nähe elektrischer Geräte legen, da sie sich magnetisch aufladen könnten
– Reinigen Sie den Tonkopf Ihres Kassettenrecorders regelmäßig. Sie erhalten die Reinigungskassetten in jedem Fachgeschäft.
– Nach Gebrauch sollten Sie die Kassetten in ihre Hüllen zurücklegen.

DER GEBRAUCH VON PROGRAMMSTECKMODULEN

Ein Programmsteckmodul ist einfacher zu handhaben als eine Programmkassette.
Das IS-BASIC Modul, welches zum Lieferumfang der Grundausstattung gehört, ermöglicht Ihnen das Schreiben von eigenen Programmen. Mehr darüber können Sie im ENTERPRISE-Handbuch nachlesen. Sie können fertige Programme als Steckmodul kaufen und direkt auf Ihrem ENTERPRISE-Computer nutzen. Die Handhabung dieser Module ist einfach. Stecken Sie Ihr Programmsteckmodul einfach in den Modulsteckplatz (ROM BAY). Drücken Sie dann den RESET-Knopf auf der Rückseite Ihres Computers. Das Programm wird nun automatisch geladen. Um die Arbeit mit diesem Steckmodul zu beenden, entfernen Sie es einfach aus dem Steckplatz. Drücken Sie wieder den RESET-Knopf. Jetzt können Sie andere Steckmodule oder Kassetten benutzen.
Das ist alles was Sie zur Benutzung von Programmsteckmodulen wissen sollten.

FEHLER

Sollten Sie einmal etwas Falsches eingeben, wird Sie der ENTERPRISE-Computer darauf hinweisen. Wollen Sie neu mit Ihrer Aufgabe beginnen, so drücken Sie einfach den RESET-Knopf auf der Rückseite des Computers.
Auch wenn Sie nicht immer die richtige Taste drücken, wird Ihr ENTERPRISE-Computer keinen Schaden nehmen.
Gehen Sie auf „Entdeckungsreise", spielen Sie mit der Tastatur.
Ein kurzer Blick in das ENTERPRISE-Handbuch wird alle Probleme lösen.

DIAGNOSE TABELLE – FÜHRUNG FÜR DEN ANWENDER

MERKMALE	GRÜNDE	ABHILFE
Der Computer zeigt eine Fehlermeldung an.	1) Das Netzanschlußkabel wackelt. 2) Interner Fehler im Computer.	Drücken Sie den RESET-Knopf. Sollte das nichts nützen, ziehen Sie den Netzanschluß heraus. Nach 10 Sekunden können Sie das Netzkabel wieder einstöpseln. Sollte auch das nicht nutzen, gehen Sie zu Ihrem Fachhändler.
Schlechtes Bild oder kein Ton:	Der Fernseher ist nicht auf den Computer eingestellt.	Kontrollieren Sie die Einstellung Ihres Fernsehers.
Keine Farbe.	1) Der Fernseher ist nicht richtig eingestellt. 2) Die Farbe ist nicht eingestellt.	1) Kontrollieren Sie den Fernseher. 2) Stellen Sie die Farbe im Fernseher ein.
Kein Bild, obwohl Kontrollleuchte auf dem Computer leuchtet.	1) Der Fernseher ist nicht richtig eingestellt. 2) Fernseher ist nicht am Computer angeschlossen. 3) Fernseher ist nicht am Stromnetz angeschlossen.	1) Kontrollieren Sie den Fernseher. 2) Prüfen Sie die Verbindung Fernseher – Computer. 3) Prüfen Sie den Stromanschluß.
Basic funktioniert	4) Basicsteckmodul nicht korrekt eingesteckt.	4) Prüfen Sie die Steckverbindung des Basicmoduls.
Kontrolleuchte am Computer leuchtet nicht.	1) Computer ist nicht am Stromnetz angeschlossen. 2) Computer ist nicht richtig angeschlossen.	1) Prüfen Sie die Stromverbindung mit dem Netz. 2) Suchen Sie Ihren Fachhändler auf.

Bemerkung: Links auf der Rückseite Ihres ENTERPRISE Computers befindet sich eine Vorrichtung, aus der die Wärme entweichen kann.

Der ENTERPRISE ist ein äußerst vielseitiger Mikrocomputer. Wir wollen gleich beginnen, mit ihm zu arbeiten, denn auf diese Weise werden wir schnell feststellen, was er alles kann. Später werden wir auf die Einzelheiten genauer eingehen.
Am besten ist es, wenn Sie den ersten Teil dieses Handbuchs der Reihe nach durchgehen. Er soll Ihnen helfen, mit dem neuen Computer vertraut zu werden. Überall dort, wo Sie bestimmte Themen oder Techniken gleich vertiefen wollen, finden Sie Querverweise, so daß Sie je nach Vorkenntnissen beliebig schnell voranschreiten können. Im zweiten Teil werden die einzelnen Bereiche der Programmierung – sie hängen natürlich alle zusammen, wie Sie feststellen werden – in weitaus größerem Detail als im ersten Teil behandelt. Im letzten Teil finden Sie dann, nachdem Sie die Arbeitsweise des Computers grundsätzlich kennengelernt haben, viele weitere Feinheiten und technische Details zum Nachschlagen.
Bevor Sie mit den ersten Programmversuchen beginnen, sollten Sie ein wenig mit der Tastatur experimentieren. Sie können alles mögliche eintippen, ohne den Computer zu beschädigen. Falls er einmal nicht auf die Tastatur reagieren sollte, können Sie ihn mit Hilfe der 'RESET'-Taste auf der Rückseite des Rechners wieder „zur Besinnung bringen".
Mit dem Cursorkontrollhebel (das ist der Hebel auf der rechten Geräteseite) läßt sich besonders gut spielen. Wie nützlich er ist, werden Sie bald erkennen. Zuvor sollten Sie aber den Computer ein wenig kennenlernen. Er hat Ihnen viel zu bieten.

Anmerkung: Wenn Sie den Computer in BASIC programmieren wollen, muß das IS-BASIC-Modul im ROM BAY (Modul-Steckplatz) auf der linken Geräteseite eingesteckt sein.

Um Ihren Computer in Betrieb zu nehmen, siehe Erklärung in der Installationsanleitung, verbinden Sie IN auf dem ENTERPRISE mit OUT/EAR auf dem Kassettenrecorder und REM1 auf dem ENTERPRISE mit REM, ebenfalls auf dem Kassettenrecorder. Falls Ihr Kassettenrecorder keine Steuerungsbuchse REM hat, arbeitet die Demonstrations-Kassette auch.

PROGRAMM LADEN

Drücken Sie die START-Taste auf dem ENTERPRISE (Funktions-Taste 1) und drücken dann PLAY auf Ihrem Kassettenrecorder. Sie sehen nun, daß das Wort SEARCHING auf dem Bildschirm erscheint und danach LOADING INTRO. Der Computer lädt jedes Programm und startet es automatisch, wenn Sie eine Steuerungsbuchse an Ihrem Kassettenrecorder haben, die mit dem Anschluß (REM1) am Computer durch das Kabel verbunden ist. Wenn dies nicht der Fall ist, beachten Sie die Worte LOADING INTRO oder LOADING MODES oder bis die anderen Abschnitte auf dieser Kassette beendet sind. Sie hören dann einen Pfeifton. Drücken Sie dann umgehend die STOP-Taste auf Ihrem Kassettenrecorder. Wenn das nächste Programm der Kassette geladen werden soll, werden Sie durch den Hinweis SEARCHING aufgefordert, am Kassettenrecorder die PLAY-Taste zu drücken. Sobald das Programm geladen ist und der Pfeifton zu hören ist, drücken Sie auf dem Kassettenrecorder die STOP-Taste.
Diese Demo-Kassette hat zwei Seiten. Damit Sie mit der zweiten Kassettenseite arbeiten können, wird die Kassette umgedreht und rückgespult.
Wenn Sie Kopfhörer haben, stecken Sie das Kabel in die OUT-Buchse am ENTERPRISE. Somit können Sie die ersten zwei Musikprogramme auf Ihrer Demonstrations-Kassette in Stereoton hören.
Diese Kassette veranschaulicht Ihnen die verschiedenartigen Möglichkeiten der Grafik und der Tonausgabe Ihres ENTERPRISE-Computers. Es werden alle benötigten Anweisungen auf dem Bildschirm angezeigt. Dieses Handbuch wird Ihnen jedoch erklären, wie die Spiele zu handhaben sind, und bei einigen Spielen, wie diese arbeiten und wie Sie diese selbst ändern können. Sie sind alle in der BASIC-Programmiersprache geschrieben; Sie werden sehen, daß mit Hilfe einfacher Anweisungen eindrucksvolle Effekte erzielt werden.
Betätigen Sie die STOP-Taste, um die Kassette anzuhalten, wenn Sie prüfen wollen, wie die einzelnen Pro-

gramme arbeiten. Nun drücken Sie die Funktionstaste 5. Dies löscht die gesamte Grafik auf dem Bildschirm. Dann drücken Sie die Funktionstaste 2 und Sie sehen das Programm. Drücken Sie die HOLD-Taste, wenn das Programm zu schnell abrollt und drücken diese nochmal, um weiter aufzulisten.
Haben Sie schon einmal eine Berichtigung vorgenommen und mit der ENTER-Taste bestätigt, können Sie ein Programm neu starten durch Drücken der START-Taste (Funktionstaste 1) somit beginnt das Programm erneut zu arbeiten.
Die verschiedenen Programme auf dieser Kassette sind alle miteinander verkettet, so daß der Computer automatisch alle Programme nacheinander laden und starten kann. Um ein Programm zu ändern, müssen Sie aus der Verkettung heraus.
Es ist das beste, zuerst die gesamte Kassette durchzugehen, bevor Sie irgendetwas ändern.
Jedes Programm hat einen eigenen Namen. So können Sie das gewünschte Programm einzeln laden, wenn Sie z.B. LOAD "DRAW" eingeben.
Sollte Ihnen bei der Berichtigung eines Programms ein Fehler unterlaufen, wird auf keinen Fall das Demonstrationsprogramm zerstört. Das schlimmste, was passieren kann, ist, daß Sie das Programm erneut von der Kassette laden.
Ist das Programm einmal geladen, läuft ein Demonstrationsprogramm ab, das Ihnen die Tastenfunktionen des ENTERPRISE-Computers erklärt.

DEMONSTRATION

Dieses Programm führt Sie durch das Innere des ENTERPRISE-Computers und erklärt Ihnen die wichtigsten Stellen.

1. Z80 Mikroprocessor

Hier wird über die fortschrittliche Speicherverwaltung des ENTERPRISE berichtet, über das Betriebssystem (EXOS) und IS-BASIC, der mitgelieferten Computersprache.

2. GRAPHICS CHIP

Der ENTERPRISE ist in der Lage, eine hochauflösende Grafik mit bis zu 256 Farben auf den Bildschirm zu bringen. Dieser Abschnitt zeigt Ihnen Beispiele von verschiedenen Darstellungen, die im Grafik-Modus möglich sind.
Sie sehen, daß bei jedem Beispiel auch die gleich-

zeitige Mischung der verschiedenen Modes möglich ist. Wollen Sie wissen wie diese Bilder entstehen, drücken Sie bitte die STOP-Taste und folgen den Anweisungen auf Seite 1.

3. Der Sound Chip

Der ENTERPRISE kann mit 4 Tongeneratoren, im Bereich von 8 Oktaven vollen Stereoton erzeugen. Der spezielle „sound chip" ermöglicht alle Arten der Tonerzeugung und es können alle Befehle mit BASIC direkt angesprochen werden. Diese klangvollen Töne kommen aus dem ENTERPRISE Lautsprecher oder durch die Kopfhörer (3,5 mm Stereo-Jack Stecker) oder den Stereo-Lautsprechern. Das Programm durchläuft einige Musikroutinen und Sie können so die Klangerzeugung testen. Dieses Programm zeigt Ihnen 4 Beispiele; für weitere Information sh. ENTERPRISE-Handbuch.

4. Der Word-Processor

Dieser Abschnitt beschreibt die ENTERPRISE-Textbearbeitung, die in der Grundausstattung enthalten ist.

5. Hilfsfunktion

Es wird ein sogenanntes „Hilfsmenü" mit einer Auswahl von verschiedenen Möglichkeiten angezeigt.

1. DEMONSTRATION	= wiederholen der Demonstration
2. BASIC	= zurück ins BASIC
3. WORD-PROCESSOR	= Übergang zur Textbearbeitung
4. BOMBER Game	= spielen des Bomberspieles
5. CHARACTER Definer	= eigene Zeichenerstellung
6. DRAW Program	= Programm zum Malen
7. INTERLACE Treiber	= Treiber für hochauflösende Grafik

MENÜ

Seite 2 der Demonstrations-Kassette enthält eine Anzahl nützlicher Programme. Wenn Sie an das Ende der Seite 1 gelangen, drehen Sie bitte die Kassette um und spulen zurück. Drücken Sie dann auf Ihrem ENTERPRISE-Computer die Funktionstaste 1 „Start" und auf Ihrem Kassettenrecorder die Play-Taste.

Ist das Programm geladen, wird auf dem Bildschirm folgendes Menü zur Auswahl angezeigt:

1. BOMBER Game	= spielen des Bomberspieles

2. CHARACTER Definer = eigene Zeichenerstellung
3. DRAW program = Programm zum Malen
4. INTERLACE driver = laden des INTERLACE Treibers
5. BASIC = zurück ins BASIC

PROGRAMM ZUR ZEICHENERSTELLUNG UND BOMBER SPIEL

Jedes ENTERPRISE-Zeichen – A,B,C...$%,,* usw. besteht aus einem Netz (Matrix) von 9 x 8 Punkten. Die Punkte, die die Zeichen bilden, haben die Vordergrundfarbe (INK COLOUR), der Rest der Matrix hat die Hintergrundfarbe (PAPER COLOUR). Alle Buchstaben und andere Zeichen, die der Computer kennt, haben eine Nummer, unter der sie gespeichert sind. Mit dem ENTERPRISE können Sie selbst Zeichen mit den Nummern von 128 bis 159 erstellen und diese wie alle anderen Zeichen benutzen. Durch das Programm für die Zeichenerstellung ist dies ganz leicht.

Der Computer stellt ein Menü dar:
1 Play Game (spielen)
2 Character Definer (Zeichen erstellen)
ESC to leave Program (Programm beenden)
Geben Sie die 2 ein, damit rufen Sie das Programm zum Zeichen erstellen auf.
Der Computer zeigt an:
* links, ein Netz bestehend aus 9 x 8 Feldern;
* in der Mitte, eine Auswahl vordefinierter Zeichen;
* rechts, ein Befehlsmenü;
jetzt werden Sie gefragt:
CHARACTER NUMBERS (128 – 159)

Geben Sie die Zeichennummer ein, die Sie erstellen oder ändern möchten. Der Computer zeichnet das Netz neu, mit Hilfe eines blinkenden roten Cursors in der linken oberen Ecke. Betätigen Sie den Cursor-Steuerhebel, um den Cursor zu bewegen, und beschreiben Sie das Feld, indem Sie ein Muster entwerfen. Dies wird mit Hilfe nachstehender Befehle gemacht. Wenn Sie ein Zeichen erstellt haben, wird dieses gleich in die Zeichentabelle in der Bildschirmmitte eingefügt.

SPACE BAR – Füllt oder leert das Quadrat an der Stelle, wo sich der Cursor gerade befindet.
L – leert das Koordinatennetz
D – Erstellt Zeichen, d.h. es programmiert die von Ihnen gemalten Zeichen in den Spei-

cher des Computers, zeigt dies am Bildschirm an, und listet die Befehle auf, die Sie brauchen, um das neue Zeichen in einem BASIC-Programm anwenden zu können.

U – Horizontal, dreht das Zeichen um die Horizontalachse.

T – Negatives Bild (intervertiert). Wenn Sie diese Taste drücken, entstehen aus den gefüllten Quadraten nicht gefüllte Quadrate und umgekehrt.

W – Nächste. Verläßt das momentan bearbeitete Zeichen und bereitet das Koordinatennetz für das nächste Zeichen vor.

E – Beendet die Zeichenerstellung und geht zurück ins Menü.

R – Vertikal. Dreht das Zeichen um die vertikale Achse.

K – Ruft ein bereits vorher erstelltes Zeichen oder ein anderes aus dem ENTERPRISE Zeichensatz auf. Sie können ein x-beliebiges Zeichen aus diesem Zeichensatz des Computers wählen; die Zeichennummern liegen zwischen 32 bis 159. Testen Sie dies und versuchen Sie herauszufinden, welches Zeichen welche Nummer hat.

Beachten Sie, daß die Zeichen 128 – 134 für das CITY BOMBER Spiel erforderlich sind.
128 und 129 Gebäude
130 und 131 Flugzeug
132 Bomben
133 und 134 Explosion
Wenn Sie diese Zeichen neu erstellen, beeinflußt dies das Spiel. Sie lernen bald, die Formen so zu entwerfen, so daß die Gebäude in Feuersäulen aufgehen. Eventuell können Sie das Flugzeug in ein Raumschiff umwandeln und statt der Bombe eine Rakete zeichnen. Die Explosion kann auch optisch gewaltiger dargestellt werden.

PROGRAMM ZUR EIGENEN ZEICHENERSTELLUNG

Um eine vielseitige Version des Zeichenerstellungsprogrammes abzuspeichern, löschen Sie das CITY BOMBER Programm.

Machen Sie folgendes:
DELETE FIRST TO 1035
DELETE 1470

DELETE 2580 TO LAST
Um das Spiel zu löschen:
Ändern Sie die Zeile 1440 in:
1440 IF LC$="q"THEN KEY=27
Dann sichern Sie (SAVE) die neue Version auf eine neue Kassette (nicht auf die Demo-Kassette).
Wenn Sie nun ein Programm schreiben, das selbsterstellte Zeichen beinhalten soll, verwenden Sie dafür das Programm zur eigenen Zeichenerstellung. Benutzen Sie dafür die am Bildschirmrand stehenden Werte, um die Zeichen in Ihrem Programm zu verwenden.
Löschen Sie das Programm zur eigenen Zeichenerstellung. Die von Ihnen erstellten Zeichen sind dann im Computer gespeichert, bis diese gelöscht werden oder der Computer ausgeschaltet wird. Dadurch können Sie Ihre Zeichen erstellen und dann ein Programm schreiben, so daß diese Zeichen verwendet werden können. Beachten Sie bitte, wenn Sie Zeichen in Ihrem Programm bestimmen, kennzeichnen Sie diese mit 0, 1, 2, 3 ... nicht mit 128, 129, 130 ...
Beispiel: Sie erstellen Zeichen 128 mit:
SET CHARACTER 0
oder Sie erstellen Zeichen 129 mit:
SET CHARACTER 1
usw.

CITY BOMBER SPIEL

Sie steuern ein Flugzeug über eine Stadt von Wolkenkratzern. Das Flugzeug fliegt tiefer und tiefer. Die einzige Hoffnung um sicher zu landen, ist, Sie bombardieren die Stadt und zerstören die Gebäude. Drücken Sie die Leertaste, um eine Bombe abzuwerfen. Ob Sie nun abstürzen oder landen – der Computer gibt Ihnen eine passende Nachricht und startet erneut. Wenn Sie das nächstfolgende Programm laden wollen, drücken Sie die ESCAPE-Taste.

VERÄNDERN DES SPIELES

Sie können das Spiel schwieriger machen, wenn Sie das Programm geringfügig ändern. Unterbrechen Sie das Programm mit Hilfe der STOP-Taste und geben Sie den TEXT-Modus ein, indem Sie die TEXT-Taste (Funktion 5) drücken.

FLUGZEUG TIEFER STARTEN LASSEN

Sie können tiefer auf dem Bildschirm beginnen, um das Flugzeug zu starten, indem Sie einen Teil der Zeile 170 ändern.

Ändern Sie:
I=1
in eine größere Zahl wie z.B. I=5.
Ändern Sie jedoch keine anderen Werte.

FLUGZEUGABSTIEG BESCHLEUNIGEN

Sie können den Flugzeugabstieg schneller machen durch Änderung der Zeile 350:
Ändern Sie bitte:
350 LET I= I+2
in:
350 LET I= I+3
oder Sie machen das Spiel etwas langsamer, indem Sie ändern auf:
350 LET I=I+1
Haben Sie bei diesem Programm Kopfhörer auf, wird Ihnen die Qualität der Stereo-Tonausgabe Ihres ENTERPRISE Computers auffallen; man hört das Flugzeuggeräusch von einem Ohr zum anderen „kreuzen".

MALEN MIT HILFE DES PROGRAMMS

Mit diesem einfachen, in BASIC geschriebenen Grafik-Paket, können Sie Ihren ENTERPRISE wie ein Malbuch benutzen. Mit dem Cursor-Steuerhebel beschreiben Sie die Formen und füllen diese aus.

Dies ist zu tun mit folgenden Befehlen:

DRAW – Bewegt den Stift nach unten und zeichnet Linien mit dem Cursor.

LINE – Damit wird eine gerade Linie zwischen 2 Punkten gezeichnet, die vorher vom Benutzer definiert wurden.

CIRCLE – Diese Funktion zeichnet einen Kreis, mit einem bestimmten Radius.

FILL – Damit wird der Bereich ausgefüllt, in welchem sich der Cursor gerade befindet, und mit der momentan gültigen Farbe ausgefüllt.

X-TYPE – Stellt eine Auswahl von 3 verschiedenen Cursor-Zeichen zur Verfügung.
Einen kleinen leuchtenden Punkt.
Ein kleines Kreuz.
Ein großes Kreuz.

INK – Damit wählen Sie die Farbe (Vordergrundfarbe) aus der Palette am Bildschirmrand.

PALETTE – Damit kann jede Vordergrundfarbe aus der Palette geändert werden; dies ist möglich durch den Zugriff auf alle 256 Farben des ENTERPRISE-Computers.

RUBOUT – Mit dieser Funktion können Sie bereits gezeichnete Formen „ausradieren". Aus diesem Grund ist es möglich, Formen zu übermalen.

HELP – Durch Drücken dieser Taste erhalten Sie Information über die einzelnen Befehle.

BLOCK – Mit diesem Befehl können Sie einen Bereich löschen, der vorher vom Benutzer erstellt wurde. Dieser Bereich kann wieder übermalt werden.

QUIT – Mit dieser Funktion gelangen Sie ins Befehlsmenü zurück.

DER INTERLACE-TREIBER

Der INTERLACE-Treiber kann von Kassette geladen werden, um die Vertikalauflösung auf dem Bildschirm zu verdoppeln. Am besten ist das möglich mit Monitoren, die eine lange Nachwirkdauer haben. Das beseitigt das „Flimmern" des Bildes auf dem Monitor. Legen Sie die Kassette, die den INTERLACE-Treiber enthält, in Ihren Kassetten-Recorder und geben Sie ein:
LOAD „INTERLACE". Damit laden Sie den INTERLACE-Treiber.
Sie können aber auch nur LOAD eingeben, wenn der INTERLACE-Treiber sowieso das nächstfolgende Programm auf Ihrer Kassette ist.
Ist der INTERLACE-Treiber von der Kassette geladen, startet er automatisch und arbeitet solange, bis das System-RESET erfolgt. In diesem Falle wird der Bildschirm gelöscht und oben links erscheint „OK". Der INTERLACE-Treiber hat einen Software Text-Modus und einen hochauflösenden Grafik-Modus. Die Standard BASIC-Textseite besteht aus 80 Spalten und 50 Zeilen. (TEXT 80 und TEXT 40 werden beide auf dem Bildschirm im selben Format angezeigt.) Standardmäßig besteht die BASIC-GRAPHIC Seite aus 20 Zeilen (ergibt eine Auflösung von 640 horizontalen und 360 vertikalen Punkten im 2-Farben Modus). Außerdem besteht diese auch noch aus 8 Zeilen unterhalb des Grafik-Fensters, in genau demselben Format, wie das vom normalen Modus her bekannt ist. Sie

werden auch bemerken, daß der Grafik-Bildschirm die doppelte Vertikalauflösung hat.
Es werden dieselben Koordinaten benutzt, wie auch bei normaler Grafik; 1280 x 720 Punkte für den BASIC-GRAPHIC-Bildschirm. So wie alle Grafikmerkmale der Standard-Hochauflösung auf den Grafik-Seiten durch den INTERLACE-Treiber unterstützt werden, so sollten auch Programme, die für normale Grafik geschrieben wurden, mit geringer oder gar keiner Abänderung arbeiten können.
Mit dem INTERLACE-Treiber benötigt der Bildschirm bis zu 40KB vom Speicher, so daß auf einem ENTERPRISE 64 der Benutzer ca. 7KB für Programme und Daten zur Verfügung hat. Bei einem Computer mit größerem RAM gibt es keinerlei Begrenzung, was die Programmgröße anbelangt; bis auf die üblichen Grenzen natürlich.
Die Funktionen der ENTERPRISE-Textbearbeitung sind nicht kompatibel zum INTERLACE-Treiber. Ein eigenes Programm der Textbearbeitung bedeutet einen Vorteil; erfordert jedoch eine gewisse Speichererweiterung. Der INTERLACE-Treiber wird unterbrochen, falls Sie versuchen die Textbearbeitung einzugeben.
Beachten Sie bitte, wenn der INTERLACE-Treiber aus dem Hauptmenü (Funktionstaste 4) aufgerufen wird, so starten die Programme aus dem Speicher. Soll nur der INTERLACE-Treiber geladen werden, spulen Sie die Kassette zurück und geben Sie LOAD "INTERLACE" ein.
Alles auf der Demonstrations-Kassette kann geladen werden, indem man LOAD benutzt und den dementsprechenden Namen in Anführungszeichen setzt. Z.B. LOAD "INTERLACE"

© 1984 Enterprise Computers Ltd. Hergestellt bei Words & Pictures Ltd. Übersetzt von: Textprojekt München
Beachten Sie, daß diese Broschüre nicht die Beschreibung der Produkte enthält. Änderungen und technische Weiterentwicklungen sind vorbehalten.

Versuchen Sie zunächst, den Inhalt des nachfolgenden Kastens einzutippen. Sie müssen alles ganz genau machen, denn Computer sind sehr empfindlich in bezug auf Tippfehler. Am Ende einer jeden Zeile müssen Sie die 'ENTER'-Taste drücken. Die Zahlen am Anfang der einzelnen Zeilen sind auch sehr wichtig. Dagegen brauchen Sie sich nicht um die Leerzeichen nach den Zahlen zu kümmern. Der ENTERPRISE wird manchmal selbständig solche Leerzeichen einfügen, um Programme übersichtlicher zu machen. Achten Sie darauf, daß die Null (0) und das große O vom Computer ganz verschieden interpretiert und daher auf dem Bildschirm unterschiedlich abgebildet werden.

```
100 graphics
110 plot 640,360,
120 !
130 do
140 for radius = 250 to 1 step -16
150 set ink rnd(3) + 1
160 plot ellipse radius,radius,
170 plot paint
180 next radius
190 ping
200 loop
210 !
220 end
```

FEHLER KORRIGIEREN

Wenn Sie beim Tippen einen Fehler machen, können Sie ihn leicht korrigieren. Falls Sie sich noch in der fehlerhaften Zeile befinden, drücken Sie die Löschtaste 'ERASE'. Der 'Cursor' (der rote blinkende Schreibzeiger auf dem Bildschirm) bewegt sich nach links und löscht das letzte Zeichen. Wenn Sie einen Fehler in einer vorherigen Zeile verbessern wollen, bewegen Sie den Cursor mit Hilfe des Steuerknüppels zum Ende der entsprechenden Zeile und löschen Sie die Zeile nach links, bis der Fehler erreicht ist. Sie können jetzt die Zeile berichtigen und am Ende wieder 'ENTER' drücken. Danach können Sie mit dem Steuerknüppel den Cursor zum Programmende oder zu einer beliebigen anderen Stelle bringen. Denken Sie daran, daß alle Zeichen, die Sie auf der Tastatur eingeben, immer dort auf dem Bildschirm erscheinen, wo sich der Cursor gerade befindet.

Es gibt wesentlich elegantere Methoden, Fehler zu korrigieren, aber dazu kommen wir im Kapitel „Editieren von Programmen" (Seite 63).

DAS PROGRAMM STARTEN

Nachdem Sie das obige Programm eingegeben haben, tippen Sie das Wort RUN und drücken Sie 'ENTER', um das Programm zum Laufen zu bringen. Sie können statt dessen auch die **Funktionstaste 1** benutzen. Das ist die blaue Taste oberhalb der Zahlentasten mit der Bezeichnung 'FUNCTION 1'. Das hat dieselbe Wirkung wie RUN und 'ENTER'. (Sie erfahren mehr über die Funktionstasten auf Seite ??).
Nun beobachten Sie den Bildschirm. Wenn Sie beim Abschreiben des Programms einen Tippfehler gemacht haben, wird „nicht verstanden" oder eine andere Fehlermeldung auf dem Bildschirm erscheinen. Machen Sie sich keine Sorgen, drücken Sie einfach **Funktionstaste 5 und dann Funktionstaste 2**. Schauen Sie sich das Programm nochmals an. Verbessern Sie den Fehler und starten Sie das Programm noch einmal.

BEFEHLE, ANWEISUNGEN, KOMMANDOS

In diesem Handbuch sind alle Befehle oder Anweisungen (z.B. RUN), die Sie eintippen sollen, in Großbuchstaben angegeben. Das geschieht, um sie im Text hervorzuheben, aber auch, weil der Rechner selbst BASIC-Anweisungen in Großbuchstaben anzeigt. Sie können jedoch die Befehle auch in Kleinbuchstaben eingeben, d.h. ohne mit 'SHIFT' auf Großbuchstaben umzuschalten. Der Computer wird sie in jeder Schreibweise ohne Probleme verstehen.
Die Begriffe 'Befehl', 'Kommando' und 'Anweisung' werden in diesem Handbuch weitgehend identisch verwendet. Wenn Sie übrigens bestimmte Begriffe aus der Computersprache nicht kennen, sollten Sie im Glossar auf den Seiten 238–250 nachschlagen.
Um das laufende Programm zu stoppen, drücken Sie die rote 'STOP'- Taste. Auf dem Bildschirm sehen Sie

```
STOP in Zeile ---
ok
-
```

Der Computer gibt an, in welcher Zeile er unterbrochen wurde ("---" ist eine Zeilennummer). Wollen Sie das Programm weiterlaufen lassen, geben Sie das Wort CONTINUE ein (vergessen Sie nicht, am Ende noch 'ENTER' zu

drücken). Dasselbe erreichen Sie, indem Sie die SHIFT-Taste drücken und gleichzeitig die Funktionstaste 1. Das Programm läuft von der Stelle an weiter, wo Sie es unterbrochen haben.

WAS IST EIN PROGRAMM

Ein Programm ist eine Reihe von Anweisungen, mit denen man dem Computer mitteilt, was er tun soll. Es gibt nichts Geheimnisvolles daran, aber ein Computer kann ohne ein Programm nichts tun. Programme können sehr genau und detailliert sein, und wie bei jeder Tätigkeit, die Geschicklichkeit erfordert, kann man ernsthaft programmieren oder es nur aus Spaß tun.

BILDSCHIRM LÖSCHEN

Wenn Sie das obige Programm laufen lassen und dann wieder stoppen, ist der Bildschirm voller Graphik. Das macht es nicht gerade einfach, neue Anweisungen einzugeben. Um den Bildschirm in den normalen Textmodus zurückzubringen, tippen Sie TEXT und ENTER. Hier können Sie wiederum denselben Effekt durch Drücken der Funktionstaste 5 erreichen.
Sie können das Programm jetzt noch versuchsweise erweitern. Tippen Sie noch die folgende Zeile ein:

```
135 SET PALETTE RND(256), RND(256), RND(256),
RND(256)
```

LIST

Wenn Sie die Zeile eingetippt haben, geben Sie LIST ein. (Sie haben sicherlich inzwischen gemerkt, daß man nach solchen Anweisungen grundsätzlich 'ENTER' drücken muß. Mit 'ENTER' sagen Sie dem Computer, daß die Anweisung vollständig eingetippt ist und er sie nun ausführen soll. Normalerweise passiert nichts, bis Sie mit 'ENTER' eine Zeile „abgeschickt" haben).
Statt LIST einzugeben können Sie übrigens auch einfach die Funktionstaste 2 drücken. Sie haben sicherlich gemerkt, daß der Computer bei den Funktionstasten die entsprechenden Befehle ausführt, ohne daß Sie 'ENTER' drücken müssen.
Die LIST-Anweisung bewirkt, daß das ganze Programm übersichtlich strukturiert auf den Bildschirm gebracht wird – jedenfalls soviel davon, wie auf einmal auf den Bildschirm paßt. Sie werden erkennen, daß die neue Zeile ganz automatisch vom Computer in das Programm eingegliedert wurde. Alle Programmzeilen sind in der richtigen

Reihenfolge – und in dieser Reihenfolge wird der Computer sie abarbeiten.

PROGRAMME LÖSCHEN

Wenn Sie vom obigen Programm genug haben, tippen Sie NEW (+ 'ENTER'). Das Programm wird aus dem Computerspeicher gelöscht und kann nicht mehr ausgeführt werden.

ÜBUNGSPROGRAMME

Mit Ihrem Computer können Sie allerlei Farben, Zeichnungen und Töne erzeugen. Aber er kann auch umfangreiche Berechnungen sehr schnell durchführen, Listen sortieren, mathematische Aufgaben lösen, und vieles andere mehr. Auf den nächsten Seiten sind einige einfache Programmbeispiele, mit denen Sie die vielfältigen Talente des ENTERPRISE entdecken können. Probieren Sie alle! Es wird Ihnen bestimmt Spaß machen.

Denken Sie daran, daß Sie mit der Anweisung TEXT immer den Bildschirm für weitere Texteingaben freimachen können. Dasselbe können Sie mit CLEAR SCREEN (= Bildschirm löschen) erreichen. Keiner dieser Befehle löscht das Programm aus dem Speicher und Sie können es jederzeit mit LIST wieder auf den Bildschirm bringen.

```
 70 ! Dieses Programm zeichnet einen Tunnel
 80 ! mit explodierenden Feuerkugeln.
 90 !-------------------------------------------
100 GRAPHICS HIRES 256
110 LET X = 640: LET Y = 360
120 FOR R = 1 TO 255
130         SET INK R
140         LET A = X - R - 220
150         LET A1 = Y - R - 50
160         LET C = X + R + 220
170         LET C1 = Y + R + 50
180         PLOT A,A1; A,C1; C,C1; C,A1; A,A1
190         PRINT R
200 NEXT R
210 FOR N = 1 TO 100
220         CALL FIREBALL (256, X, Y)
230 NEXT N
240 END
250 !-------------------------------------------
260 DEF FIREBALL (COLOURS, A, B)
270         SET LINE MODE 3
280         SET INK RND(COLOURS)
290         FOR G = 1 TO 2
300                 FOR H = 1 TO 650 STEP 30
310                         PLOT A,B, ELLIPSE H,H
320                 NEXT H
330         NEXT G
340         SET LINE MODE 0
350 END DEF
```

```
 90 ! Dieses Programm zeichnet Rechtecke.
100 CLEAR SCREEN
110 PRINT AT 5,11: "DIESES PROGRAMM WIRD"
120 PRINT AT 6,9: "RECHTECKE FÜR SIE ZEICHNEN"
130 PRINT AT 8,1: "Das Programm wird Sie bitten,
    Koordinaten einzugeben."
140 PRINT "Die erste Zahl sollte nicht größer als
    1279 sein"
150 PRINT "und die zweite nicht größer als 719."
160 PRINT "Drücken Sie >ENTER< nach jeder Eingabe!"
170 WAIT 10 ! 10 Sekunden warten
180 !-------------------------------------------
190 DO
200         CLEAR SCREEN
210         INPUT AT 5,1, PROMPT "Koordinaten für die
            erste Ecke: ": X
220         INPUT AT 6,34, PROMPT " ": Y
230         INPUT AT 8,1, PROMPT "Für die gegenüber-
            liegende Ecke: ": V
240         INPUT AT 9,38, PROMPT " ": W
250         PRINT AT 11,1:"Wie lange soll das Rechteck
            stehenbleiben?"
260         INPUT AT 13,5, PROMPT "Sekunden: ": ZEIT
270         !-----------------------------------------
280         ! Die nächsten Zeilen zeichnen
290         ! Rechtecke und lassen Sie für die
300         ! gewünschte Zeit auf dem Bildschirm.
310         !-----------------------------------------
320         GRAPHICS
330         PLOT X,Y; X,W; V,W; V,Y; X,Y
340         WAIT ZEIT
350         TEXT
360         PRINT AT 15,18: "Weiter? "
370         DO
380                 INPUT AT 17,17, PROMPT "j oder n ":
                    ANTW$
390         LOOP WHILE ANTW$ <> "j" AND ANTW$
    <> "n"
400 LOOP WHILE ANTW$ = "j"
410 END ! Ende des Programms
```

```
100 !       Dieses Programm bringt 10 Zahlen
105 !       in die richtige Reihenfolge.
106 !       ------------------------------------------
110 NUMERIC F(1 TO 10)
120 NUMERIC VAR, ZAHL, GROSS
130 CLEAR SCREEN
150 PRINT AT 10,10: "Zahlen sortieren"
160 FOR N = 1 TO 10
170         PRINT AT 14,10: N;
180         INPUT PROMPT ". Zahl = ":F(N)
190         PRINT AT 14,24: " "
200 NEXT N
210 CLEAR SCREEN
220 PRINT AT 20,20: "Ein Augenblick..."
230 LET SCHLUSS = 10
240 FOR X = 1 TO 10
250         LET GROSS = 0
260         FOR Y = 1 TO SCHLUSS
270                 IF F(Y) > GROSS THEN LET
                    GROSS = F(Y)
280                 IF F(Y) = GROSS THEN LET
                    ZAHL = Y
290         NEXT Y
300         LET VAR = F(SCHLUSS)
310         LET F(SCHLUSS) = GROSS
320         LET F(ZAHL) = VAR
330         LET SCHLUSS = SCHLUSS - 1
340 NEXT X
350 CLEAR SCREEN
360 FOR X = 1 TO 10
370         PRINT F(X)
380 NEXT X
390 END
```

```
100 ! Mit diesem Programm berechnen Sie
105 ! Fläche und Umfang eines Kreises
106 !-----------------------------------------
110 LET A$ = "des Kreises ist "
115 LET B$ = "Durchmesser des Kreises = "
120 NUMERIC DURCHM, FLAECHE, UMFANG
125 DO
130         CLEAR SCREEN
135         PRINT AT 10,10: "1) Fläche"
140         PRINT AT 11,10: "2) Umfang"
145         PRINT AT 12,10: "3) Schluß"
150         DO
155                 INPUT AT 14,1, PROMPT "Wählen Sie
                    bitte eine der Zahlen ":ZAHL
160         LOOP WHILE ZAHL < 1 OR
            ZAHL > 3 OR ZAHL <> INT(ZAHL)
165         CLEAR SCREEN
170         IF ZAHL = 1 THEN
180                 INPUT AT 10,10, PROMPT B$: DURCHM
190                 LET FLAECHE = PI * (DURCHM/2) ^ 2
200                 PRINT AT 15,5:"Die Fläche ist "; A$;
                    FLAECHE; "."
210                 WAIT 10
220         ELSE IF ZAHL = 2 THEN
230                 INPUT AT 10,10 PROMPT B$: DURCHM
240                 LET UMFANG = PI * DURCHM
250                 PRINT AT 15,5: "Der Umfang ist "; A$;
                      UMFANG; "."
260                 WAIT 10
270         END IF
280 LOOP WHILE ZAHL <> 3
290 END
```

```
100 ! Dieses Programm erteilt
105 ! kulinarische Ratschläge.
106 !-----------------------------------------
110 LET A$ = "Darf ich "
115 LET B$ = " vorschlagen?"
120 STRING NACHTISCH$(1 TO 7) * 50
130 LET NACHTISCH$(1) = "einen Apfelstrudel"
140 LET NACHTISCH$(2) = "eine Birne Helene"
150 LET NACHTISCH$(3) = "einen Pfirsich Melba"
160 LET NACHTISCH$(4) = "eine Kirschtorte"
170 LET NACHTISCH$(5) = "ein Zitroneneis"
180 LET NACHTISCH$(6) = "einen Bananenflip"
190 LET NACHTISCH$(7) = "einen Pflaumenkuchen"
195 CLEAR SCREEN
200 PRINT AT 10,5: "Was mögen Sie am liebsten?"
210 PRINT AT 14,20: "1) Apfel"
220 PRINT AT 15,20: "2) Birne"
230 PRINT AT 16,20: "3) Pfirsich"
240 PRINT AT 17,20: "4) Kirschen"
250 PRINT AT 18,20: "5) Zitrone"
260 PRINT AT 19,20: "6) Banane"
270 PRINT AT 20,20: "7) Pflaumen"
280 DO
290         INPUT AT 22,1, PROMPT "Geben Sie die
            gewünschte Zahl ein: ": ZAHL
300 LOOP WHILE ZAHL < 1 OR ZAHL > 7 OR ZAHL <>
    INT(ZAHL)
310 SELECT CASE ZAHL
320 CASE 1
330         PRINT A$ & NACHTISCH$(1) & B$
340 CASE 2
350         PRINT A$ & NACHTISCH$(2) & B$
360 CASE 3
370         PRINT A$ & NACHTISCH$(3) & B$
380 CASE 4
390         PRINT A$ & NACHTISCH$(4) & B$
400 CASE 5
410         PRINT A$ & NACHTISCH$(5) & B$
420 CASE 6
430         PRINT A$ & NACHTISCH$(6) & B$
440 CASE 7
450         PRINT A$ & NACHTISCH$(7) & B$
460 END SELECT
470 END
```

```
100 ! Dieses Programm schreibt Mitteilungen
110 DO
120         CLEAR SCREEN
130         PRINT AT 10,5: "MITTEILUNGEN"
140         INPUT AT 12,1, PROMPT "Bitte schreiben Sie
            eine Mitteilung: ": MIT$
150         INPUT PROMPT "Wie oft soll ich diese
            Mitteilung schreiben?": ANZAHL
160         PRINT "Bitte wählen Sie:"
170         PRINT "1) Untereinander"
180         PRINT "2) Nebeneinander"
190         PRINT "3) In Spalten"
200         DO
205             INPUT WAHL
210         LOOP WHILE WAHL < 1 OR WAHL >
            3 OR WAHL <> INT(WAHL)
215         CLEAR SCREEN
220         SELECT CASE WAHL
230         CASE 1
240             FOR L = 1 TO ANZAHL
250                 PRINT MIT$
260             NEXT L
270         CASE 2
280             FOR L = 1 TO ANZAHL
290                 PRINT MIT$;
300             NEXT L
310         CASE 3
320             FOR L = 1 TO ANZAHL
330                 PRINT MIT$,
340             NEXT L
350         END SELECT
355         PRINT:PRINT
360           DO
370             INPUT PROMPT "Nochmal? (j/n)": E$
375             LET E$ = UCASE$(E$(1:1))
380         LOOP WHILE E$ <> "J" AND E$ <> "N"
390 LOOP WHILE E$ = "J"
400 END
```

PROGRAMME VERÄNDERN

Wenn Sie wollen, können Sie mit den vorangegangenen Programmen experimentieren. Sie sollten nichts an der Schreibweise der Anweisungen im Programm verändern, da der Computer sie nicht mehr verstehen würde. Aber alle Mitteilungen und Texte, die in Anführungszeichen stehen, können Sie getrost verändern, ohne daß es den Ablauf des Programms wesentlich beeinflußt. Sie können auch einige der Zahlen variieren, um zu sehen, was sie im Programm bewirken. Aber dabei kann es eher vorkommen, daß der Computer mit einer Fehlermeldung reagiert. Über eines können Sie sicher sein: Egal was Sie eintippen, es wird den Computer **niemals beschädigen können!** Schlimmstenfalls wird er eine Anweisung nicht verstehen. Wenn Sie beispielsweise die folgende Zeile eingegeben haben:

```
250 FOR X = 1 TO 10
```

Dann wird er diese Zeile beanstanden:

```
*** nicht verstanden
250 FOR X = 1 TO 10
```

Eine ausführliche Liste aller Fehlermeldungen finden Sie auf S. 234–237).

STOPPEN UND STARTEN

Wenn der Computer gar nicht mehr richtig reagieren sollte, können Sie immer die 'RESET'-Taste hinten am Gerät drücken (diese Taste ist eine Art Notbremse). Das versetzt den Computer in den Zustand, in dem er beim Einschalten war – außer daß er das zuletzt eingegebene Programm noch im Speicher behält. Mit RUN können Sie dann das Programm wieder starten.

Im Normalfall stoppen Sie ein laufendes Programm mit der 'STOP'- Taste. Mit der 'HOLD'-Taste können Sie das Programm „einfrieren". Das ist dann nützlich, wenn Sie z.B. ein bestimmtes Bild in einem Grafik-Programm genauer betrachten wollen – oder auch wenn Sie während eines Weltraumspiels eine Kaffeepause einlegen möchten. Wenn Sie dann im Programm fortfahren wollen, drükken Sie einfach die 'HOLD'-Taste ein zweites Mal. Diese Taste ist auch besonders nützlich, wenn man ein sehr langes Programm „auslistet", das nicht mehr auf dem Bildschirm paßt. Mit 'HOLD' kann man den Vorgang anhalten und einzelne Programmzeilen genau untersuchen.

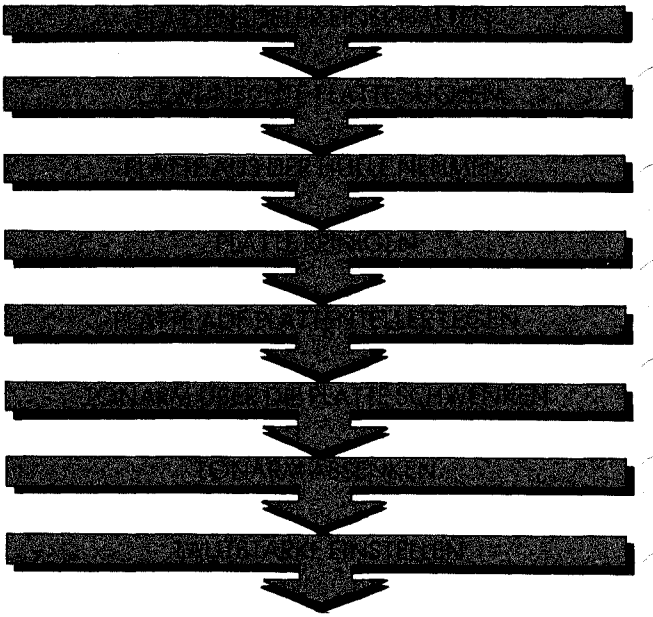
Mit der 'STOP'-Taste können Sie ein laufendes Programm unterbrechen und es z.B. verändern oder löschen – oder auch nur auslisten, bevor Sie CONTINUE eingeben und es weiterlaufen lassen.

Sie haben inzwischen Ihren Computer kennengelernt und mit ihm hoffentlich ein wenig Spaß gehabt. Sicherlich haben Sie nicht alle Programmteile im einzelnen verstanden, so daß wir nun beginnen wollen, zum Spaß auch noch Wissen hinzuzufügen.

Ein Programm ist – wie wir gesehen haben – eine Reihe von Anweisungen an den Computer. Man benutzt ein Programm dazu, ein Problem mit Hilfe des Computers zu lösen. Der Computer ist ein Werkzeug, mit dem man bestimmte Fähigkeiten des menschlichen Gehirns verstärken und erweitern kann. Oder man kann einfach mit ihm spielen, wobei Sie ein faszinierenderes Spielzeug kaum finden werden.

Ein Computer versteht Anweisungen, die in der Form von Wörtern oder Zahlen abgefaßt sind. Wenn wir mit dem ENTERPRISE kommunizieren, werden wir dies mit Hilfe von Wörtern tun. Jedes Wort ist eine kleine Anweisung und man kann viele Wörter wie ein Puzzle zusammensetzen, um kompliziertere Anweisungen und schließlich eine vollständige Problemlösung zu erhalten. Darum geht es beim Programmieren. Wir geben dem Computer Anweisungen – in der Reihenfolge, in der sie befolgt werden sollen.

Die Abbildung unten zeigt mit einem alltäglichen Beispiel, wie ein Problem in viele kleine Einzelaufgaben aufgeteilt und zu einem „Programm" zusammengesetzt wird.



COMPUTER-SPRACHEN

Die Menschen können auf vielfältige Weise miteinander sprechen, und ebenso gibt es viele Möglichkeiten, einen Computer zu programmieren. Genauso wie die Menschen verschiedene Sprachen benutzen, gibt es auch eine Reihe von „Computersprachen". Diese sind entwickelt worden, um ganz bestimmten Aufgaben gerecht zu werden. Mit einigen kann man besonders gut große Datenmengen verwalten, andere sind für die Erzeugung von Bildern hervorragend geeignet. Mit einigen kann man das Programmieren besonders gut erlernen.

BASIC

Die Sprache, die wir im Umgang mit dem ENTERPRISE kennengelernt haben, heißt BASIC. Sie verwendet Ausdrücke, die dem Englischen entlehnt und ganz einfach zu verstehen sind – auch wenn man wenig Erfahrung mit Computern hat. Alle Programme in diesem Handbuch sind in BASIC geschrieben. Solange das IS-BASIC-Modul eingesteckt ist, versteht der ENTERPRISE diese Sprache. Er kann auch andere Computersprachen verarbeiten, aber diese müssen vorher von Cassette oder Diskette „geladen" werden.
Schauen Sie sich die folgende Anweisung an:

```
10 PRINT "Hallo!"
```

Wie Sie wissen, ist das eine Programmzeile. Es ist eine kleine Anweisung, die Teil eines wesentlich längeren Programms sein könnte. Genauso wie in einer Geschichte enthält jede Zeile eines Programms Informationen. Sie sagen dem Computer, was er als nächstes tun soll. Soll er eine längere Arbeit verrichten, sind meistens viele Zeilen nötig.

ALLES IN DER RICHTIGEN REIHENFOLGE

Die Zahl am Anfang der Programmzeile ist von besonderer Wichtigkeit. Sie wissen bereits, daß BASIC-Programme in „Zeilen" aufgeteilt werden und daß jede dieser Zeilen eine Nummer braucht. Die Reihenfolge, in der die einzelnen Programmzeilen eingetippt werden, spielt dabei keine Rolle. Alle Zeilen werden je nach Zeilennummer im Speicher automatisch richtig sortiert.
Es sind die Zeilennummern selbst, auf die es also ankommt. Wenn Sie das Programm starten, wird der

Computer die Zeile mit der niedrigsten Nummer zuerst ausführen, dann die nächsthöhere, usw. (es sei denn, das Programm enthält eine anderslautende Anweisung).
Diese letzte Einschränkung ist, wie Sie später sehen werden, ziemlich wichtig. Es gibt viele Arten, wie Sie ein Programm zusammenstellen können und es wird nicht selten vorkommen, daß wir die Reihenfolge innerhalb des Programms bewußt verändern werden. Es gibt viele BASIC-Anweisungen, mit denen man das bewerkstelligen kann. Das bedeutet aber, daß man sich immer im klaren sein muß, was genau der Computer tun soll. Man muß Programme sehr sorgfältig planen, bevor man sie eingibt.
Falls Sie übrigens eine Zeilennummer wählen, die im Programm bereits verwendet wurde, wird die ursprüngliche Zeile mit dieser Nummer im Speicher gelöscht (die Zeile wird „überschrieben"). Wenn Sie eine bestimmte Zeile aus dem Programm nur löschen wollen, tippen Sie die Zeilennummer und drücken Sie 'ENTER'. Die Programmzeile verschwindet, da Sie damit eine leere Zeile eingegeben haben. Man kann genauso gut die Anweisung DELETE (= Löschen) verwenden, um eine Zeile -- aber auch eine Reihe von Zeilen gleichzeitig -- zu löschen. Die Anweisung DELETE 10 TO 100 löscht alle Programmzeilen zwischen diesen beiden Zahlen, einschließlich 10 und 100.

DIREKTMODUS UND SCHLÜSSELWÖRTER

Wenn Sie eine Anweisung eingeben, ohne daß diese eine Zeilennummer hat, wird sie sofort ausgeführt. Tippen Sie z.B. folgendes ein:

```
PRINT "Hallo!"
```

und drücken Sie 'ENTER'. Der Computer schreibt sofort das Wort „Hallo!" auf den Bildschirm, direkt unter der Anweisung. Wenn Sie jedoch eine Zahl vor die Anweisung setzen, wie in dem Beispiel oben, dann wird der Computer auch nach 'ENTER' vorerst nichts tun. Die Programmzeile wird nur im Speicher abgelegt und erst dann ausgeführt, wenn Sie RUN eingeben oder mit der Funktionstaste 1 das Programm starten.

DER DIREKTMODUS

Anweisungen ohne Zeilennummern werden im sogenannten **„Direktmodus"** ausgeführt. Man kann in diesem Modus sehr bequem kleine Rechenaufgaben lösen. Auf der Tastatur finden Sie alle Zeichen für die Grundrechenarten: „+" und „-" sind wie gewohnt, aber die Zeichen für 'mal' ('*') und 'geteilt durch' ('/') sind anders – wenngleich in Computerkreisen durchaus geläufig.
Tippen Sie folgendes ein:

```
PRINT 2 * (7 – 1)
```

Wenn Sie 'ENTER' drücken, erscheint das Ergebnis der Aufgabe, 12, unter der Anweisung. Sie können im Direktmodus auch kompliziertere Rechenaufgaben lösen. Geben Sie beispielsweise die folgende Anweisung ein:

```
PRINT SQR(81)
```

Der Computer wird die Wurzel von 81 ausrechnen und auf den Bildschirm schreiben. SQR ist ein besonderes BASIC-Wort und hat für den Computer die Bedeutung „Wurzel von der nachfolgenden in Klammern geschriebenen Zahl". Es gibt viele solche Ausdrücke in BASIC und Sie finden die komplette Liste im Nachschlageteil dieses Handbuchs (Kapitel: Eingebaute Funktionen und Variablen). Die wichtigsten werden auch im Textteil erklärt, sobald sie in den Beispielprogrammen auftauchen.
Die meisten BASIC-Anweisungen können genausogut im Direktmodus wie auch in einem BASIC-Programm verwendet werden. Sie können damit experimentieren, indem Sie Programmzeilen aus diesem Handbuch ohne Zeilennummer eingeben. Im Nachschlageteil bekommen Sie genaue Auskunft darüber, welche Anweisungen im Direktmodus verwendet werden können und welche nicht.
Übrigens können Sie den Direktmodus auch während des Programmierens verwenden, ohne daß Sie damit die bereits eingegebenen Programmteile beeinflussen. Das ist besonders nützlich, wenn man bestimmte Werte, die im Programm eine Rolle spielen, zwischendurch schnell ausrechnen muß.

SCHLÜSSELWÖRTER

Jetzt wissen Sie einiges über das Programmieren in BASIC und wie der Direktmodus funktioniert. Es gibt viele „Schlüsselwörter", die in BASIC eine ganz bestimmte Bedeutung haben. Der Computer versteht sie als Anwei-

sung, etwas Bestimmtes zu tun. Das Wort PRINT, z.B., heißt soviel wie „Schreibe" oder „Drucke", und veranlaßt den Computer, etwas auf den Bildschirm zu bringen.

STRINGS

Wir haben am Anfang dieses Kapitels mit PRINT eine Zeichenkette auf den Bildschirm gebracht (Hallo!). Das, was auf dem Bildschirm erscheinen sollte, stand in Anführungszeichen ("---"). Solche Zeichenketten nennt der Fachmann „Strings" und man kann sie – für den Augenblick jedenfalls – mit Zitaten vergleichen (für weitere Details darüber siehe S. 82). Wenn man nach einem PRINT- Befehl eine solche Zeichenkette in Anführungszeichen angibt, kümmert sich der Computer nicht um den Inhalt des Strings, sondern schreibt ihn einfach auf den Bildschirm. Er zitiert ihn gewissermaßen.

ZEICHEN

Die einzelnen Zeichen eines Strings können Buchstaben, Zahlen oder auch andere Formen sein, die der Computer auf dem Bildschirm abbilden kann. Die Gesamtheit aller Zeichen, die der Computer auf dem Bildschirm erzeugen kann, nennt man den „Zeichensatz".

Sie wissen nun, wie ein Programm aussieht, Sie kennen Zeilennummern, BASIC-Anweisungen und den Direktmodus. Es ist Zeit, ein wenig weiterzugehen.
Stellen Sie sich zwei Kästchen vor, die mit X und Y beschriftet sind. In jedes Kästchen kann man eine Zahl hineingeben. Später kann man sie wieder herausnehmen oder durch eine neue ersetzen.
In dem nachfolgenden Programm werden wir dem Computer sagen, daß er zwei Kästchen bereithalten soll. Wenn wir diese mit Zahlen gefüllt haben, soll er sie zusammenzählen und das Ergebnis auf den Bildschirm bringen.

```
 10 ! ----------------------------------------------------
 20 !       In den Zeilen 60 und 70 wird der Computer
 30 !       nach zwei Zahlen fragen. Tippen Sie
 40 !       diese ein und drücken Sie ENTER!
 50 ! ----------------------------------------------------
 60 INPUT PROMPT "Erste Zahl: ": X
 70 INPUT PROMPT "Zweite Zahl: ": Y
 80 ! ----------------------------------------------------
 90 !       Zeile 130 addiert die Zahlen X und Y
100 !       und speichert das Ergebnis unter der
110 !       Variablen Z ab.
120 ! ----------------------------------------------------
130 LET Z = X + Y
140 ! ----------------------------------------------------
150 !       Zeile 170 zeigt Ihnen das Ergebnis.
160 ! ----------------------------------------------------
170 PRINT X; "+"; Y; "="; Z
180 END ! Ende des Programms.
```

KOMMENTARZEILEN

Zunächst einmal werden Sie bemerken, daß viele Programmzeilen mit einem Ausrufezeichen beginnen. Das sind Kommentarzeilen, die nur der besseren Übersicht dienen. Der Computer speichert sie zusammen mit den anderen Programmzeilen ab, ignoriert sie aber beim Abarbeiten des Programms. Es ist sehr wichtig, solche Kommentarzeilen in Ihre Programme einzubauen, damit Sie später erkennen, wozu die einzelnen Teile dienen. Ein Programm mit ausführlichen Kommentaren ist nicht nur für Computer sondern auch für Menschen leicht verständlich!
Grundsätzlich werden alle Zeilen, die mit einem Ausrufezeichen oder mit dem Wort REM (für „Remark" = Anmer-

kung) beginnen, beim Programmablauf ignoriert. Das Ausrufezeichen kann auch mitten in einer Programmzeile stehen, wie in der Zeile 180. In einem solchen Fall wird der Rest der Zeile ignoriert. REM darf dagegen nur am Anfang der Zeile, unmittelbar nach der Zeilennummer stehen.

INPUT PROMPT

Das Zweite, was Ihnen aufgefallen sein dürfte (wahrscheinlich bereits bei den Programmen im Einführungskapitel), ist die Anweisung INPUT PROMPT. Normalerweise bewirkt die Anweisung INPUT (= Eingabe) alleine, daß der Computer in dieser Programmzeile anhält, ein Fragezeichen auf den Bildschirm bringt und auf die Eingabe einer Information wartet. Das ist nicht besonders übersichtlich, denn in längeren Programmen ist es nicht immer klar, nach welchen Werten der Computer gerade fragt. Als Erinnerung könnte man natürlich mit Hilfe des PRINT-Befehls einen Text auf den Bildschirm bringen, z.B.

```
60 PRINT "Erste Zahl: ";
65 INPUT X
70 PRINT "Zweite Zahl: ";
75 INPUT Y
```

Aber mit INPUT PROMPT geht das eleganter. Die Zeilen 60 und 70 im ursprünglichen Programm bewirken genau das gleiche wie die vier Zeilen oben. Der PROMPT ist ein Erinnerungstext, den Sie beliebig wählen können. Er muß jedoch in Anführungszeichen unmittelbar hinter der Anweisung INPUT PROMPT stehen. Wenn Sie bei der Datenabfrage einen solchen Erinnerungstext angeben, wird der Computer nicht mehr – wie bei der reinen INPUT-Anfrage – ein Fragezeichen auf den Bildschirm bringen.
Wie wir gesehen haben, läßt INPUT (mit oder ohne PROMPT) den Computer im Programm anhalten und nach einer Information fragen. Die Zeile „65 INPUT X" im Beispiel oben bewirkt, daß sich der Computer einen Wert für die Variable X holt, d.h. der Benutzer gibt einen Wert ein und der Computer legt diesen Wert unter dem Namen X ab.

NOCHMALS VARIABLEN

Eine **Variable** ist ein Name für Zahlen, deren Werte sich verändern können. In unserem Beispielprogramm holt sich der Computer (in den Zeilen 60 und 70) zwei Werte vom Benutzer und legt sie unter den Namen X und Y ab.

Sie können beliebige Zahlen eintippen – 0.00004 oder 400000. Wenn Sie für X die Zahl 400000 eingeben, dann hat die Variable X den Wert 400000.
In Zeile 130 addiert der Computer die Werte, die unter X und Y abgelegt sind und legt das Ergebnis unter dem Namen Z ab. In Zeile 170 bringt er schließlich das Ergebnis auf den Bildschirm: Er schreibt zunächst den Wert von X, dann das Zeichen „+", dann den Wert von Y, das Zeichen „=" und schließlich den Wert von Z. Läßt man das Programm nochmals laufen, kann man ganz andere Werte eingeben, die unter den Variablennamen X und Y gespeichert und weiterverarbeitet werden.
Variablen benutzt man meistens dann, wenn man nicht von vornherein weiß, welchen Wert bestimmte Zahlen annehmen werden. Teilweise werden diese Zahlen während des Programms eingegeben, teilweise werden sie vom Computer aus anderen Zahlen zusammengestellt. Manchmal benutzt man Variablen, um im Programm mit langen, häufig vorkommenden Zahlen bequem umgehen zu können. Man braucht nicht jedesmal die Zahl einzutippen, sondern verwendet statt dessen die kurze – oder auch einprägsame – Variable.
Wir wollen mit einem weiteren Beispiel die Verwendung von Variablen in einem Programm demonstrieren:

```
10 LET A = 0 !                  Variable A bekommt den Wert 0
20 DO !                         Anfang der DO/LOOP-
                                Schleife
30        A = A + 1 !           A wird um eins erhöht
40        PRINT A, !            Bringt A auf den Bildschirm
50 LOOP UNTIL A = 20 !          Schleife wiederholen
60 !                            bis A den Wert 20 erreicht
70 END
```

Das ist ein Zählprogramm, das mit Hilfe einer sogenannten DO/LOOP-Schleife alle Zahlen von 1 bis 20 auf den Bildschirm schreibt. Jedesmal, wenn der Computer die Schleife durchläuft, erhöht er den Wert der Variablen A um 1 und schreibt diesen Wert auf den Bildschirm. Am Ende der Schleife prüft er jeweils, ob A den Wert 20 erreicht hat. Ist das der Fall, beendet er das Programm. Das Komma in Zeile 40 ist übrigens nur dazu da, um die Zahlen in Spalten zu setzen. Probieren Sie aus, was passiert, wenn Sie es weglassen!

VARIABLEN EINFÜHREN

In Zeile 10 des obigen Programms wird eine Variable eingeführt. Man sagt gewissermaßen: „Computer, wir werden eine Variable namens A verwenden, die im Augenblick gleich 0 ist". Versuchen Sie, die Zeile 10 wegzulassen. Das Programm läuft nicht. Eine Variable darf nicht plötzlich im Programm auftauchen. Das gilt nicht für Variablen, die gleich einen bestimmten Wert zugewiesen bekommen: In den Anweisungen INPUT X oder FOR X = 1 TO 100 bekommt die Variable X jeweils gleich einen Wert (im ersten Fall vom Benutzer) und braucht vorher nicht eingeführt zu werden.
Mit LET kann man also eine Programmvariable einführen. Es ist nicht immer notwendig, LET tatsächlich auszuschreiben (im obigen Zählprogramm funktioniert 10 A = 0 ebensogut), aber aus Gründen der Übersicht ist es besser, LET zu verwenden. So kann man sofort die Zeile erkennen, in der die Variablen eingeführt wurden.
Variablen können auch den Wert von anderen Variablen annehmen. Wenn G bereits als Variable mit dem Wert 10 eingeführt wurde, dann kann man mit der Anweisung LET S = G die neue Variable S einführen, die ebenfalls den Wert 10 annimmt.

VARIABLENNAMEN

X, Y und Z sind nur einige Namen für Variablen. Sie können auch längere Namen verwenden, sogar bis 31 Buchstaben lang, wenn Sie wollen. Häufig ist es nützlich, beschreibende Variablennamen zu wählen, z.B. SUMME für die Summe von zwei Zahlen, DOLLARKURS für den Umrechnungskurs des US-Dollars. Es macht übrigens keinen Unterschied, ob Sie eine Variable SUMME, Summe, summe oder sUmmE nennen. Der Computer unterscheidet bei Variablennamen nicht zwischen Groß- und Kleinbuchstaben.
Der Computer versteht keine Variablennamen, die Leerzeichen, deutsche Umlaute, mathematische Operatoren (+, -, *, /, >, <, usw.) oder Interpunktionszeichen enthalten. Nur die Unterstreichung „_" ist zugelassen und kann das Leerzeichen optisch gut ersetzen. **Variablennamen müssen immer mit einem Buchstaben beginnen,** nie mit einer Zahl oder einem Interpunktionszeichen. Hier sind einige Beispiele für zulässige Variablennamen:

```
Zahl      A2$       NAME$  Hallo1
LINKER_RAND_2  NaCl  A1234
```

Wie Sie sehen, ist das Dollarzeichen auch zulässig, allerdings hat es – am Ende des Variablennamen – eine besondere Bedeutung. Solche Variablen sind nämlich für Zeichenketten (nicht für Zahlenwerte) reserviert. LET NAME$ = „Manfred Petersen, Versicherungskaufmann" speichert den in Anführungszeichen gesetzten Text unter dem Variablennamen NAME$ ab, PRINT NAME$ bringt ihn dann wieder auf den Bildschirm. Solche Variablennamen für Zeichenketten nennt man „String-Variablen". Sie werden auf den Seiten 82–88 in größerem Detail erläutert.
Hier sind zum Vergleich einige Variablennamen, die der Computer nicht versteht – der Grund dürfte Ihnen jeweils bekannt sein.

```
Erste Zahl 2A !!!A$
<>RADIUS  3*UMFANG  NAME,BERUF$
```

Zwei Variable können auch sehr ähnliche Namen haben – z.B.:

```
MEHRWERTSTEUER_ABRECHNUNG_JUNI_1985
und
MEHRWERTSTEUER_ABRECHNUNG_JULI_1985
```

Der Computer hat keinerlei Probleme, die beiden Namen auseinanderzuhalten. Aber wie ist es mit Ihnen? Bei solchen langen Variablennamen kann man sich leicht vertippen (MEHRWERTSTEUER_ABBRECHNUNG_JUNI_1985 wird der Computer nicht kennen). Meist reichen kürzere Namen vollkommen aus, um als Merkhilfe zu dienen (MWST_JUNI).
Auch BASIC-Ausdrücke können übrigens als Variablennamen dienen, aber dann **müssen** sie mit LET eingeführt werden (LET INPUT = 10). Das sollten Sie allerdings vorerst vermeiden, da man leicht durcheinanderkommt. Einige BASIC-Wörter lassen sich zudem nicht als Variablennamen verwenden, wie Sie mit einigen Versuchen leicht feststellen können.
Mit einem letzten Beispiel wollen wir das soeben Besprochene demonstrieren. Sie sollten keine Schwierigkeiten

haben, die Verwendung der verschiedenen Variablen zu verstehen.

```
 10 LET A$ = "Die Summe der beiden Zahlen ist "
 20 !-----------------------------------------------------
 30 ! Die String-Variable A$ wird eingeführt.
 40 !-----------------------------------------------------
 50 INPUT PROMPT "Die erste Zahl lautet: ":
    ERSTE_ZAHL
 60 INPUT PROMPT "Die zweite Zahl lautet: ":
    ZWEITE_ZAHL
 70 !-----------------------------------------------------
 80 !   Diese beiden Variablen brauchen
 90 !   nicht mit LET eingeführt zu werden,
100 !   da sie durch die INPUT-Anweisung
110 !   sogleich einen Wert erhalten.
120 !-----------------------------------------------------
130 LET SUMME = ERSTE_ZAHL + ZWEITE_ZAHL
140 !-----------------------------------------------------
150 !   Die Variable SUMME wird eingeführt
160 !   und bekommt den Wert, den man durch
170 !   die Addition der Werte von ERSTE_ZAHL
180 !   und ZWEITE_ZAHL erhält.
190 !-----------------------------------------------------
200 PRINT A$; SUMME
210 !-----------------------------------------------------
220 !   Der Text aus Zeile 10 wird angezeigt,
230 !   dahinter der Wert von SUMME.
240 !-----------------------------------------------------
250 END
```

Das ist ein mathematischer Ausdruck: 4 + 2 * 3 – 5
Und das sind **Operatoren:** +, -, *, /, ∧, =, <, >, <>, <= und >=.
Alle die oben angeführten Operatoren kennt der ENTERPRISE. Einige sind Ihnen sicherlich geläufig, wie +, -, *, / und =. Wie wir bereits gesehen haben, verwendet der Computer * anstatt des Malzeichens x und / anstelle von ÷ („geteilt durch").
Hier sind die restlichen Operatoren, falls Sie sie nicht kennen:

∧ bedeutet „hoch", z.B. 2∧3 = zwei hoch drei.
< bedeutet „kleiner als", z.B. 2<3.
> bedeutet „größer als", z.B. 3>2.
<> bedeutet „ungleich" (größer oder kleiner als).
<= bedeutet „kleiner oder gleich".
>= bedeutet „größer oder gleich".

Die letzten fünf sind Vergleichsoperatoren, die in erster Linie dazu dienen, Variablen miteinander zu vergleichen. Hier ein Beispiel:

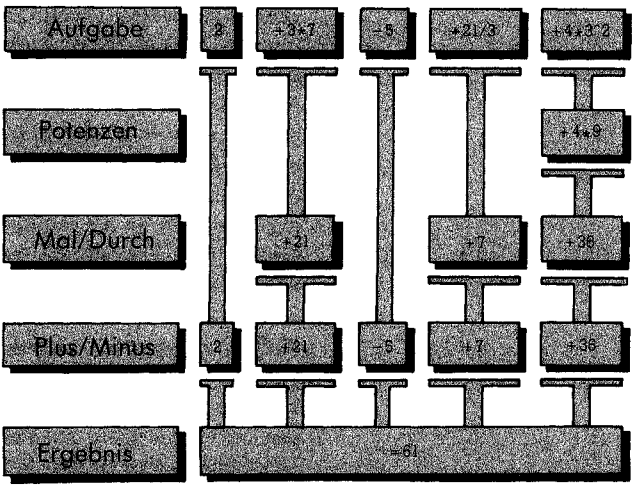
```
10 INPUT PROMPT "Bitte eine Zahl eingeben: ": A
20 IF SGN(A) = -1 THEN PRINT "Das ist eine negative
   Zahl."
30 IF A = 0 THEN PRINT "Das ist die Null."
40 IF A > 0 AND A < 10 THEN PRINT "Das ist eine
   einstellige positive Zahl."
50 IF A >= 10 AND A < 100 THEN PRINT "Das ist eine
   zweistellige positive Zahl."
60 IF A >= 100 AND A < 1000 THEN PRINT "Das ist eine
   dreistellige positive Zahl."
70 IF A >= 1000 THEN PRINT "Das ist eine große Zahl!"
80 END
```

In Zeile 10 fragt der Computer nach einer Zahl, die er dann unter dem Variablennamen A abspeichert. In Zeile 20 prüft er nach, ob die eingegebene Zahl kleiner ist als Null. Das geschieht mit Hilfe der Anweisung IF... THEN... (Wenn... dann...). Wenn A kleiner ist als Null, dann schreibt er „Das ist eine negative Zahl" auf den Bildschirm. IF ist das kritische Wort: Wenn die Bedingung A < 0 nicht zutrifft, ignoriert der Computer den Rest der Zeile und macht im Programm weiter.
Genauso funktioniert die Sache in den Zeilen 30 bis 60. Die Anweisung IF... AND... THEN ist eine Erweiterung der IF/THEN- Anweisung und bedeutet soviel wie „Wenn...

und... dann...", z.B. wenn A > 0 und A < 100 dann schreibe .... (Zeile 40). Nur wenn **beide** Bedingungen zutreffen, führt der Computer die Anweisung aus, die nach THEN steht (PRINT „Das ist eine zweistellige positive Zahl.")
In diesem Programm muß der Computer also Entscheidungen treffen und unterschiedlich reagieren, je nachdem welche Zahl eingegeben wurde. Dazu verwendet er IF/THEN-Anweisungen und die Vergleichsoperatoren <, >, >=, =. (Über andere Entscheidungsverfahren finden Sie mehr auf den Seiten ??-??).
Vergleichsoperatoren benutzt man, um Zahlen miteinander zu vergleichen. Dabei werden die Zahlen selbst nicht verändert. Die Operatoren, die wir als nächstes behandeln wollen, sind dagegen gerade dazu da, um Zahlen und Werte zu modifizieren.

HIERARCHIE DER OPERATOREN

Schauen Sie sich die folgende Aufgabe an: 4 + 2 * 3 – 5. Wie ist das Ergebnis? Dreizehn, weil man von links nach rechts arbeitet? Nein, fünf, denn die Multiplikation muß zuerst ausgeführt werden!
Es gibt eine Hierarchie der Operatoren, d.h. eine ganz bestimmte Reihenfolge, in der die verschiedenen Operationen vom Computer ausgeführt werden. Potenzen (z.B. 4∧3) werden zuerst ausgerechnet, danach Multiplikation und Division (2 * 3 oder 6 / 2) und zuletzt Addition und Subtraktion. In der oben angeführten Aufgabe wird also zunächst die Multiplikation durchgeführt (2 * 3 = 6), so daß in der zweiten Stufe 4 + 6 – 5 bleibt. Hier arbeitet der Computer tatsächlich von links nach rechts und ermittelt das Ergebnis 5.
Wir wollen eine längere Aufgabe anschauen, die alle Operatoren enthält:



2 + 3 * 7 – 5 + 21 / 3 + 4 * 3∧2
Der Computer geht folgendermaßen vor:
Zunächst die Potenzrechnung: 3∧2 = 9.
Nun lautet die Aufgabe: 2 + 3 * 7 – 5 + 21 / 3 + 4 * 9.
Jetzt Mal und Durch: 3 * 7 = 21, 21 / 3 = 7 und 4 * 9 = 36.
Nun lautet die Aufgabe: 2 + 21 – 5 + 7 + 36.
Und das wird von links nach rechts ausgerechnet:
2 + 21 = 23, 23 - 5 = 18, 18 + 7 = 25, 25 + 36 = 61.
Das Ergebnis ist also 61.
Die Reihenfolge der Operationen spielt also bei Aufgaben eine wichtige Rolle und es kann vorkommen, daß der Computer die Aufgabe anders versteht, als Sie sie gemeint haben. Sie müssen also immer darauf achten, daß die Priorität der Operatoren eingehalten wird. Sie können allerdings die Reihenfolge ohne weiteres ändern und bestimmte Operationen vorziehen. Dazu brauchen Sie nur die betreffenden Teile in Klammern zu setzen.
Betrachten Sie die folgende Aufgabe – als Gegensatz zu der vorigen:
(2 + 3) * 7 – 5 + 21 / 3 + 4 * 3∧2
Hier rechnet der Computer zunächst den Ausdruck innerhalb der Klammern aus: 2 + 3 = 5. Nun lautet die Aufgabe 5 * 7 – 5 + 21 / 3 + 4 * 3∧2. Danach geht es wie gehabt weiter: 3∧2 = 9, also 5 * 7 -5 + 21 / 4 * 9. Nun kommen Multiplikation und Division: 5 * 7 = 35, 21 / 3 = 7 und 4 * 9 = 36. Die Aufgabe lautet: 35 – 5 + 7 + 36 und das Ergebnis ist 73.
Man kann solche Aufgaben mit Hilfe von Klammern auf

vielfältige Weise ordnen und vom Computer ausrechnen lassen. Denken Sie aber daran, daß innerhalb der Klammern die ursprüngliche Reihenfolge wieder greift. In der Aufgabe (4 + 2 * 3)∧3 wird also zunächst die Multiplikation 2 * 3 = 6 und dann die Addition 4 + 6 = 10 ausgeführt, bevor die Dreierpotenz 10∧3 = 1000 errechnet wird.

Experimentieren Sie mit Beispielen wie den oben angegebenen. Das können Sie am einfachsten im Direktmodus mit Hilfe des PRINT- Befehls: z.B. PRINT (4 + 7 * (12 – 8)) / 2∧3. Sie können auch Dezimalzahlen (z.B. 7.23345 oder .0098) verwenden, allerdings nicht mit dem deutschen Komma sondern dem international üblichen Punkt als Dezimalzeichen. Auch negative Zahlen (z.B. -2) versteht der Computer ohne Schwierigkeiten.

Möglicherweise finden Sie diese Rechenvorschriften ein wenig langweilig, aber es ist wichtig, Zahlen so zu verwenden, wie sie der Computer versteht. Man braucht glücklicherweise kein großer Mathematiker zu sein, um schöne und nützliche Programme zu schreiben, aber einige einfache Berechnungen sind immer notwendig. Sogar bei reinen Grafik-Programmen muß man die Koordinaten ausrechnen, damit die Zeichnungen dort erscheinen, wo man sie haben will.

Betrachten sie das folgende Programm:

```
10 PRINT "Hallo, ";
20 PRINT "Ich ";
30 PRINT "bin ";
40 PRINT "ein ";
50 PRINT "Computer."
```

Wenn Sie das Programm starten, erscheint der Satz „Hallo, ich bin ein Computer." auf dem Bildschirm. Hätten Sie die Semikolons (';') in den ersten vier Zeilen weggelassen, wäre der Satz folgendermaßen ausgegeben worden:

```
Hallo,
ich
bin
ein
Computer.
```

Das Semikolon bewirkt also, daß der nächste Text nicht auf einer neuen Zeile erscheint, sondern unmittelbar hinter dem letzten Text. Daher müssen wir auch im Programm oben ein Leerzeichen hinter die einzelnen Wörter setzen, sonst erschiene der Satz „Hallo,ichbineinComputer."
Die PRINT-Anweisung bringt alles, was dahinter in Anführungszeichen gesetzt ist, auf den Bildschirm. Sie können alle Buchstaben, Sonderzeichen und Zahlen mit PRINT auf den Bildschirm schreiben lassen -- mit einer Ausnahme. Man kann die Anführungszeichen selbst nicht ohne weiteres ausdrucken lassen, da das den Computer durcheinanderbringen würde. Also ist z.B.
PRINT "Ich bin ein „Elektronengehirn""
nicht möglich, wohl aber
PRINT "Ich bin ein 'Elektronengehirn'"
oder sehr trickreich
PRINT "Ich bin ein „„Elektronengehirn""""
Mit dem Semikolon kann man also verschiedene Wörter oder Texte unmittelbar hintereinander drucken lassen. Auch das Komma hat, hinter den zweiten Anführungszeichen einer PRINT-Anweisung gesetzt, eine besondere Bedeutung: Man kann damit Texte oder Zahlen in Spalten auf den Bildschirm bringen. Versuchen Sie das folgende Programm:

```
10 LET A = 0
20 DO
30        LET A = A + 1
40        PRINT A,
50 LOOP UNTIL A = 20
60 END
```

Wie Sie sehen, druckt der Computer die Zahlen von 1 bis 20 feinsäuberlich in fünf Spalten. Normalerweise passen 40 Zeichen auf eine Zeile, so daß jede Spalte acht Zeichen enthalten kann. Experimentieren Sie mit dem Programm oben: Lassen Sie das Komma weg oder ersetzen Sie es durch ein Semikolon. Versuchen Sie auch folgendes:

```
40        PRINT "ENTERPRISE ",
```

Sie werden bald erkennen, wie man die PRINT-Ausgabe auf dem Bildschirm mit Semikolon und Komma hübsch gestalten kann.
Versuchen Sie noch das folgende Programm:

```
10 PRINT "        Limerick"
20 PRINT " ========="
30 PRINT
40 PRINT "Odysseus sprach: Diese Circe!"
50 PRINT "All meine Freunde verwirce."
60 PRINT "  Denkt etwa die Maid"
70 PRINT "  Ich sei auch nicht gefaid?"
80 PRINT "Wenn ja, muß ich sagen, da irce!"
90 END
```

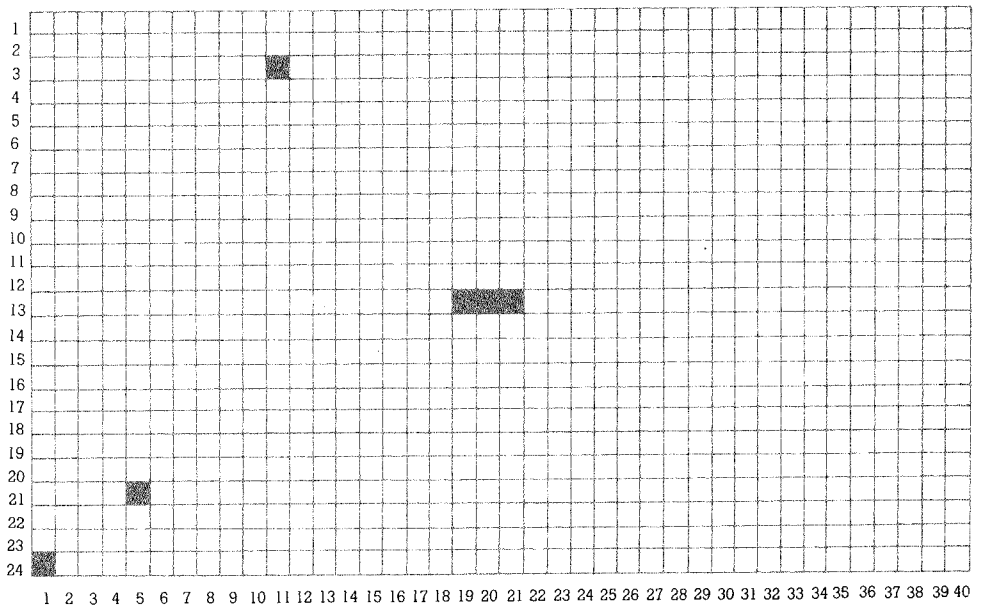
Die Zeile 30 bewirkt, daß eine Leerzeile auf den Bildschirm gebracht wird. Solche leeren PRINT-Anweisungen benutzt man häufig, um die Ausgabe von Zahlen und Texten ordentlicher zu gestalten.

PRINT AT

Eine weitere wichtige Gestaltungsmöglichkeit bietet PRINT AT. Mit dem Zusatz AT sagen Sie dem Computer, wo genau er auf dem Bildschirm einen Text ausdrucken soll. Versuchen Sie es mit der folgenden Anweisung (im Direktmodus):

```
PRINT AT 10,20: "Hier bin ich!"
```

Der Bildschirm hat normalerweise 24 Zeilen mit 40 „Spalten" (d.h. Plätze für Zeichen) in jeder Zeile. PRINT AT 10,20 bedeutet, der Computer soll in Zeile 10, Spalte 20 zu schreiben beginnen. Mit PRINT AT 15,20 kann man den Text etwas tiefer drucken, mit PRINT AT 10,10 weiter nach links.
Das Diagramm auf der nächsten Seite zeigt Ihnen, wie der Bildschirm normalerweise 24x40 (d.h. insgesamt 960) Plätze hat, in die jeweils genau ein Zeichen paßt. Sie können mit PRINT AT sehr leicht einen Text oder auch Zahlen an jeder beliebigen Stelle des Bildschirms ausgeben lassen. Denken Sie daran, daß Sie zuerst die Zeile (also wie weit nach unten) und dann die Spalte (wie weit nach rechts) angeben müssen.



Ein ähnliches System werden wir auch für die Grafik verwenden (siehe S. 117). Und über weitere, anspruchsvollere Möglichkeiten der Textgestaltung erfahren Sie im nächsten Kapitel mehr.

80-ZEICHEN-MODUS

Die Anweisung TEXT 80 gestattet Ihnen, 80 Zeichen pro Zeile auf den Bildschirm zu bringen. Die Buchstaben sind natürlich schmaler und auf einem normalen Fernseher nicht sehr deutlich. Aber auf einem Monitor ist die Schärfe

wesentlich größer und auch 80 Zeichen sind einwandfrei zu erkennen.
TEXT 80 löscht übrigens gleichzeitig den Bildschirm, wie auch die Anweisung TEXT 40, womit Sie zum 40-Zeichen-Modus zurückkehren.

Sie wissen bereits, wie man beim Programmieren Tippfehler mit Hilfe des Steuerknüppels und der Löschtaste (<X ERASE) korrigieren kann. Wir wollen nun anspruchsvollere Methoden betrachten, mit denen man Programme verändern, sie „editieren" kann. Es ist nämlich möglich, auf bequemste Weise Programmzeilen einzufügen, zu korrigieren, überschreiben, Zeilennummern zu verändern, usw. Das ist normalerweise recht umständlich, aber der ENTERPRISE bietet eine volle „Textverarbeitung", die man auch für das Editieren von Programmen einsetzen kann. Wir werden einen Teil dieser Möglichkeiten in diesem Kapitel vorstellen; der Rest folgt im nächsten Kapitel über die Textverarbeitung.

RENUMBER

Bei den bisherigen Programmen haben Sie sicher gemerkt, daß wir meist Zeilennummern wie 10, 20, 30, 40... verwendet haben. Natürlich kann man auch Programme mit den Zahlen 1, 2, 3... numerieren, aber das tut man aus einem sehr guten Grunde nicht. Es kann nämlich vorkommen, daß man etwas vergißt und später eine neue Zeile einfügen will. Eine Zeile mit der Nummer 25 kann man einfügen, nicht aber eine Zeile mit der Nummer 2,5.
Wenn Sie beim Programmieren entdecken, daß keine Nummer zwischen zwei Zeilen frei ist, um eine neue Zeile einzufügen, können Sie das rasch beheben. Tippen Sie einfach RENUMBER (oder drücken Sie SHIFT und FUNKTION 3, was dasselbe bewirkt). Das ganze Programm wird vom Computer automatisch neu numeriert, und zwar so, daß die erste Zeilennummer 100 ist und alle weiteren Zeilen in Zehnerschritten numeriert werden (100, 110, 120...). Dabei achtet der Computer darauf, daß alle eventuell vorhandenen Sprunganweisungen (z.B. GOTO 20) entsprechend verändert werden.

STEP

Sie können auch den Zeilenabstand selbst bestimmen. Geben Sie RENUMBER STEP 20 ein und listen Sie das Programm aus. Wie Sie sehen, sind die Zeilennummern nun in 100, 120, 140... umgeändert worden. Es gibt auch weitere Möglichkeiten, Programme mit RENUMBER umzunumerieren. Diese finden Sie im Nachschlageteil dieses Handbuches.

AUTO

Wenn Sie es lästig finden, beim Programmieren immer Zeilennummer einzutippen, können Sie diese Aufgabe dem Computer überlassen. Tippen Sie einfach AUTO.

Sobald Sie anschließend die 'ENTER'-Taste drücken, erscheint die Zahl 100 auf einer neuen Zeile. Sie können ganz normal eine Programmzeile eintippen und 'ENTER' drücken. Nun erscheint die Zahl 110 und Sie können weiterprogrammieren. Der Computer bietet Ihnen solange neue Zeilennummern an, bis Sie mit der 'STOP'-Taste die automatische Zeilennumerierung beenden.
Sie können übrigens auch bestimmen, mit welcher Zeilennummer die automatische Numerierung beginnen soll und welche Schrittweite sie haben soll. AUTO AT 200 bedeutet, die nächste Zeile soll die Nummer 200 bekommen (also 200, 210, 220...). Das ist sehr nützlich, wenn man ein Programm erweiteren will. Die Schrittweite stellen Sie mit STEP ein: Nach AUTO STEP 20 bietet Ihnen der Computer die Zeilennummern 100, 120, 140... an, nach AUTO AT 550 STEP 2 bekommen Sie 550, 552, 554...
Denken Sie daran, daß bestehende Programmzeilen durch die neuen wie immer ersetzt werden, d.h. wenn ein Programm die Zeilen 100, 110, 120... bereits enthält und Sie mit AUTO zu programmieren beginnen, dann werden die ursprünglichen Zeilen gelöscht. Denken Sie ebenfalls daran, daß die höchste zulässige Zeilennummer 9999 ist und auch im AUTO-Modus nicht überschritten werden darf.

DELETE

Wir haben bereits gesehen, daß der Befehl NEW das ganze Programm, das im Augenblick geladen ist, aus dem Computerspeicher löscht. Auch mit der Anweisung DELETE kann man Programmzeilen löschen.
Wenn Sie eine einzelne Zeile löschen wollen, können Sie das am einfachsten, indem Sie die Zeilennummer tippen und dann 'ENTER' drücken. Aber man kann es auch mit DELETE. Die Anweisung DELETE 100 löscht die Zeile 100 aus dem Programm.
DELETE kann man auch zusammen mit den Begriffen FIRST, LAST und TO (Anfang, Ende und bis) verwenden: DELETE 100 TO 140 löscht alle Zeilen von 100 bis 140. DELETE FIRST TO 100 löscht alle Zeilen vom Anfang bis 100, DELETE 200 TO LAST alle Zeilen von 200 bis zum Programmende. DELETE FIRST TO LAST hat natürlich dieselbe Wirkung wie NEW.

LIST

Wie Sie bereits wissen, benutzt man, um ein ganzes Programm auf den Bildschirm zu bringen, den Befehl LIST. Wenn Sie zwecks Korrektur eine einzige Zeile sehen wol-

len, können Sie das mit LIST gefolgt von der Zeilennummer tun. LIST 100 TO 200 bringt alle Zeilen von 100 bis 200 auf den Bildschirm. Man kann auch FIRST und LAST in Verbindung mit LIST verwenden, genauso wie bei DELETE. Wenn Sie die gewünschte Programmzeile auf dem Bildschirm sehen, können Sie den Cursor mit Hilfe des Steuerhebels zu der Stelle bringen, an der Sie Änderungen vornehmen wollen. Nun können Sie mit den Editiertasten ERASE und DEL einzelne Zeichen oder Wörter löschen. In Tabelle 1 am Ende dieses Kapitels finden Sie eine Beschreibung aller Editiertasten. Sobald Sie mit den Änderungen fertig sind, drücken Sie 'ENTER', um die Programmzeile abzuspeichern (wobei der Cursor aber noch in derselben Zeile stehen muß). Die ursprüngliche Programmzeile wird im Computerspeicher durch die neue Zeile ersetzt. Wenn Sie allerdings vergessen, nach der Korrektur 'ENTER' zu drücken, bleibt die alte Zeile unverändert.

EINFÜGEN ODER ÜBERSCHREIBEN

Normalerweise kann man beim Editieren eines Programms alle Fehler einfach überschreiben. Das Zeichen unter dem Cursor wird nämlich durch das neu eingetippte ersetzt. Wenn Sie Beispielsweise die Zahlen 20,20 in der folgenden Zeile

250 PRINT AT 20,20: "Das ist eine Primzahl!"

in 10,10 verändern wollen, gehen Sie einfach mit dem Cursor zur ersten 2 und tippen Sie die neuen Zahlen ein. Drücken Sie 'ENTER', um die veränderte Zeile abzuspeichern.

Wenn Sie aber in der Programmzeile oben etwas einfügen wollen, etwa ein „k" vor „eine", gehen Sie mit dem Cursor zum „e" und drücken Sie 'INS' (für Insert = Einfügen). Sie werden sehen, daß alle Zeichen bis zum Ende der Zeile um eine Stelle nach rechts rücken, so daß unter dem Cursor Platz für das „k" entsteht.

Sie können den Computer auch vom üblichen „Überschreib-Modus" in den „Einfüge-Modus" schalten, indem Sie die 'CTRL'-Taste festhalten und 'INS' drücken. Der Cursor verändert sein Aussehen (er sieht aus wie ein Pfeil, der nach links zeigt) und es wird fortan alles, was Sie mitten in eine Zeile eintippen, eingefügt, d.h. die Zeichen rechts vom Cursor werden automatisch jedesmal um eine Stelle verschoben, um Platz für die neuen Zeichen zu machen. Es kann dabei vorkommen, daß einige Zeichen über den rechten Bildschirmrand verschwinden, aber der Computer hat sie nicht vergessen! Sie erscheinen sofort

wieder, wenn Sie 'CTRL' und 'FUNCTION 1' drücken (darüber mehr im nächsten Kapitel). Solange Zeichen rechts vom Bildschirmrand stehen, erscheint ein „>" am Zeilenende.
Wenn Sie in den Überschreib-Modus zurückkehren wollen, drücken Sie nochmals 'CTRL' + 'INS'. In der „Statuszeile" am oberen Bildschirmrand erkennen Sie, welcher Modus gerade eingestellt ist.

ZEILEN EINFÜGEN

Wenn Sie eine ganze Programmzeile einfügen wollen, brauchen Sie nicht zur entsprechenden Stelle im ausgelisteten Programm zu gehen und dort die Einfügung zu machen. Ganze Programmzeilen werden, wie wir bereits gesehen haben, automatisch vom Computer an der richtigen Stelle eingefügt. Also schreiben Sie die neue Zeile an beliebiger Stelle und listen Sie zur Kontrolle das Programm nochmals aus.

CURSOR-BEWEGUNG

Sie werden sicher bereits gemerkt haben, daß der Cursor sich rasch in eine Richtung bewegt, wenn man den Steuerhebel in dieser Richtung festhält. Wenn der Cursor ganz oben oder unten ankommt, erscheint der Text darüber oder darunter, der zuvor unsichtbar war, auf dem Bildschirm (der Text wird „gescrollt"). Wenn Sie 'SHIFT' drükken und den Steuerhebel nach oben oder unten bewegen, wird der Text bildschirmweise „geblättert". Das ist sehr nützlich, wenn man bei einem sehr langen Programm schnell vom einem Teil zum anderen gelangen will.
Wenn Sie 'CTRL' drücken und den Steuerhebel nach links oder rechts bewegen, springt der Cursor wortweise nach links oder rechts, wenn Sie 'SHIFT' drücken, springt er gleich zum Zeilenanfang bzw. zum Zeilenende.
Sie sollten einmal mit den Cursorbewegungen und den anderen Editiermöglichkeiten ausgiebig experimentieren. Auch wenn Sie am Anfang die vielen Möglichkeiten vielleicht noch etwas verwirrend finden, Sie werden sie später bestimmt sehr zu schätzen wissen und das Programmieren wird Ihnen richtig Spaß machen.
In den beiden nachfolgenden Tabellen finden Sie alle wichtigen Funktionen der Editier-Tasten.

TABELLE 1: LÖSCHEN

FUNKTION TASTEN-KOMBINATION ANMERKUNGEN	
ERASE	**Zeichen links löschen:** Löscht das Zeichen links vom Cursor und rückt den Rest der Zeile nach. Bewegt den Cursor eine Stelle nach links. Zeichen rechts löschen: Löscht das Zeichen unter dem Cursor und rückt den Rest der Zeile nach.
DEL	Der Cursor bleibt an seiner Position, nur die Zeichen rechts vom Cursor bewegen sich. **Wort links löschen:** Löscht nach links bis zum ersten Zeichen eines Wortes.
CTRL + ERASE	Der Cursor bewegt sich ein Wort nach links. Nur Buchstaben oder Zahlen werden zu einem „Wort" gezählt, alle Sonderzeichen dazwischen werden mitgelöscht. **Wort rechts löschen:** Löscht nach rechts bis zum letzten Zeichen eines Wortes.
CTRL + DEL	Der Cursor bleibt stehen. Ansonsten wie oben. **Zeile links löschen:** Löscht die Zeile von der Cursorstellung bis zum Zeilenanfang.
SHIFT + ERASE	Bringt den Cursor zum linken Bildschirmrand. Zeile rechts löschen: Löscht die Zeile von der Cursorstellung bis zum Zeilenende.
SHIFT + DEL	Der Cursor bewegt sich nicht.

TABELLE 2: EINFÜGEN

FUNKTION TASTEN-KOMBINATION ANMERKUNGEN	
	Zeichen einfügen: Fügt ein Leerzeichen unter dem Cursor ein und bewegt alle Zeichen von der Cursorstellung bis zum Zeilenende eine Stelle nach rechts.
INS	Im Einfüge-Modus hat das dieselbe Wirkung wir die Leertaste, jedoch bewegt sich der Cursor nicht. **Zeile einfügen:** Bringt alle Zeichen von der Cursorstellung bis zum Zeilenende auf eine neue Zeile.
SHIFT + INS	Cursor bewegt sich nicht. War er am Zeilenanfang, bekommt man eine Leerzeile. **Einfüge-Modus ein/aus:** Schaltet zwischen Einfüge- und Überschreib-Modus.
CTRL + INS	Je nach Modus erscheint ein unterschiedlicher Cursor auf dem Bildschirm.
EINFÜGE-MODUS:	Erlaubt Ihnen, an beliebigen Stellen Zeichen einzufügen. Die nachfolgenden Zeichen der Zeile werden nach rechts bewegt, um Platz für die neuen Zeichen zu schaffen.
ÜBERSCHREIB-MODUS:	Neue Zeichen ersetzen die ursprünglichen, die unter dem Cursor stehen, man „überschreibt" den Text auf dem Bildschirm.

Eine besonders wertvolle Eigenschaft Ihres Computers ist, daß Sie ihn auch als anspruchsvolles Schreibgerät einsetzen können. Sie können Briefe, Aufsätze, ja ganze Romane auf bequemste Art schreiben und, wenn Sie einen Drucker besitzen, das Geschriebene zu Papier bringen. Die Vorteile gegenüber einer herkömmlichen Schreibmaschine sind gewaltig: Sie können Tippfehler sehr leicht korrigieren, den Text sauber und übersichtlich gestalten, Absätze bewegen, und vieles andere mehr.

DAS TEXTPROGRAMM

Der ENTERPRISE besitzt ein eingebautes Textbearbeitungs-Programm. Nachdem Sie das BASIC-Modul eingesteckt haben, geben Sie das Wort TYPE ein. Die Anzeige auf dem Bildschirm verändert sich und Sie sehen oben und unten einige Wörter, die Sie auf die Funktionstastenbelegung bei der Textbearbeitung aufmerksam machen. Das ist übrigens auch die Anzeige, die immer dann erscheint, wenn Sie den Computer ohne BASIC-Modul einschalten.
Der Befehl TYPE löscht übrigens alle BASIC-Programme aus dem Computerspeicher.
Machen Sie einen ersten Versuch. Tippen Sie das Wort TYPE und drücken Sie 'ENTER' (oder drücken Sie einfach 'SHIFT' und die Funktionstaste 8). Nun schreiben Sie einige Buchstaben und Zeichen und drücken Sie nochmals 'ENTER'. Normalerweise müßte der Computer mit einer Fehlermeldung protestieren, aber diesmal geht der Cursor einfach in die nächste Zeile. Das liegt daran, daß der Computer im Textbearbeitungs-Modus nicht versucht, das, was Sie an Text eingeben, als BASIC-Anweisung zu interpretieren. Er speichert den Text einfach ab und erlaubt Ihnen, damit wie auf einer Superschreibmaschine umzugehen.
Im Textbearbeitungs-Modus hat die 'ENTER'-Taste eine besondere Bedeutung. Sie funktioniert ähnlich wie der Zeilenvorschub einer elektrischen Schreibmaschine. Allerdings verwendet man 'ENTER' nur am Ende eines Absatzes, nicht am Ende einer jeden Zeile. Wenn Sie längere Sätze eingeben (ohne 'ENTER' zu drücken), werden Sie merken, daß der Computer am Zeilenende das ganze Wort, das nicht mehr auf eine Zeile paßt, automatisch in die nächste Zeile bringt (das nennt man „automatischer Zeilenumbruch"). 'ENTER' bedeutet Absatzgrenze und der Text zwischen einem 'ENTER' und dem nächsten gilt als Absatz.

Veränderungen am Text können Sie übrigens genauso vornehmen wie Veränderungen an einem Programm (siehe letztes Kapitel). Die zusätzlichen Funktionen, die wir im folgenden behandeln werden, dienen in erster Linie der Textbearbeitung. Aber sie können ebenfalls bei der Programmierung in BASIC eingesetzt werden.

SPEZIELLE FUNKTIONEN

Für die Textbearbeitung bietet der ENTERPRISE eine ganze Reihe von zusätzlichen Sonderfunktionen. Sie werden mit den Funktionstasten aufgerufen, indem Sie 'CTRL' oder 'ALT' festhalten und die entsprechende Funktionstaste drücken.

'CTRL' + FUNCTION 1 – Absatz formatieren
Mit dieser Funktion können Sie die Zeilenlängen in einem Absatz, der durch Korrekturen und Einfügungen ungleichmäßig geworden ist, leicht normalisieren. Sie wirkt immer nur auf den Absatz, in dem der Cursor steht.

'ALT' + FUNCTION 1 – Blocksatz (Randausgleich)
Mit dieser Funktion können Sie auch den rechten Rand eines Absatzes als gerade Linie erscheinen lassen (wie im Buchdruck). Der Computer fügt dabei so viele Leerzeichen ein, bis alle Zeilen die gleiche Länge haben.

'CTRL' + FUNCTION 2 – Zeile zentrieren
Die Zeile, in der der Cursor steht, wird automatisch zentriert. Das ist nützlich für Überschriften und Titel.

'ALT' + FUNCTION 2 – sämtliche Tabs aufheben
Hiermit können alle Tabulator-Stops (siehe unten) auf einmal gelöscht werden.

'CTRL' + FUNCTION 3 – Tabs setzen/aufheben
Genauso wie bei einer Schreibmaschine können Sie Tabulator-Stellen angeben, die bei der Texteingabe mit Hilfe der 'TAB'-Taste unmittelbar angesprungen werden können. Das ist besonders hilfreich, wenn man Tabellen erstellen will. Um ein Tab-Stop zu setzen, gehen Sie mit dem Cursor in die gewünschte Spalte und drücken Sie 'CTRL' + FUNCTION 3. War in dieser Spalte bereits ein Tab-Stop gesetzt, wird er durch diese Funktion gelöscht.

'ALT' + FUNCTION 3 – Lineal ein/aus
Die Formatzeile erscheint am oberen Bildschirmrand und zeigt Ihnen, wie der linke und rechte Rand gesetzt ist sowie die Positionen aller Tab-Stops (senkrechte Striche). Die Tastenkombination 'ALT' + FUNCTION 3 schaltet die Formatzeile ein und aus.

'CTRL' + FUNCTION 4 – Linker Rand setzen
Vielleicht wollen Sie die Breite Ihres Textes verändern.

Dazu können Sie -- genau wie bei einer Schreibmaschine -- den Rand verstellen. Gehen Sie mit dem Cursor in die gewünschte Spalte und drücken Sie 'CTRL' + FUNCTION 4 (die Veränderung können Sie in der Formatzeile beobachten). Nunmehr werden alle Zeilen in dieser Spalte beginnen.

'ALT' + FUNCTION 4 – Rechter Rand setzen

Genauso wie oben beschrieben können Sie hiermit die rechte Randbegrenzung verstellen.

'CTRL' + FUNCTION 5 – Schrift außerhalb d. Ränder

Mit dieser Funktion können Sie auch außerhalb der eingestellten Randbegrenzungen schreiben. Das Symbol * erscheint in der Statuszeile oben, um anzuzeigen, daß die Randbegrenzung außer Kraft gesetzt wurde. Drücken Sie dieselbe Tastenkombination, um die Randbegrenzung wieder einzuschalten.

'ALT' + FUNCTION 5 – normale Ränder und Tabs

Damit setzen Sie die Ränder auf die Standardwerte, also auf die äußeren Spalten links und rechts. Auch alle Tabs werden auf die Standardwerte zurückgesetzt.

'CTRL' + FUNCTION 6 – Absatz nach oben versetzen

Wenn Sie diese Tastenkombination drücken, bewegt sich der ganze Absatz, in dem sich der Cursor befindet, um eine Zeile nach oben. Die Zeile, die unmittelbar vor dem Absatz stand, erscheint hinter dem Absatz. Man kann damit Absätze beliebig bewegen: Halten Sie dazu die 'CTRL'-Taste fest und drücken Sie wiederholt FUNCTION 6, bis der Absatz an der richtigen Stelle ist.

Wenn Sie übrigens nur eine Zeile oder einen Satz bewegen wollen, können Sie das auch problemlos tun, indem Sie aus der Zeile einen Absatz machen! Drücken Sie im Einfüge-Modus am Anfang und Ende der Zeile die 'ENTER'-Taste. Dasselbe können Sie auch bei einem einzigen Wort. Wenn Sie zwei Absätze bewegen wollen, müssen Sie sie zuerst verbinden -- gehen Sie dazu zum Anfang des ersten Absatzes und drücken Sie 'ERASE'.

'ALT' + FUNCTION 6 – Absatz nach unten verschieben

Damit bewegen Sie den Absatz um eine Zeile nach unten.

'CTRL' + FUNCTION 7 – Farbe der Zeile ändern

Damit können Sie die Farbe der Zeile, in der der Cursor steht, sowie die des Hintergrundes verändern. Im 80-Zeichen-Modus gibt es vier verschiedene Farbkombinationen, die Sie durch wiederholtes Drücken dieser Taste wählen können. Im 40-Zeichen- Modus gibt es zwei Farbpaare.

'ALT' + FUNCTION 7 – Farbe des Absatzes ändern
Damit verändern Sie die Farbe des ganzen Absatzes.
Wenn man die Funktionstasten ohne 'CTRL' oder 'ALT' verwendet, reagieren sie unterschiedlich, je nachdem ob man im Programmier- oder Textbearbeitungs-Modus ist. Es gibt einige „Sicherheitsvorkehrungen" – z.B. wenn man im Textbearbeitungs-Modus mit FUNCTION 5 auf 80 Zeichen umschalten will, dann fragt der Computer nochmals nach, ob das wirklich gewünscht ist ('ENTER' für ja und 'ESC' für nein). Das verhindert, daß man durch einen unbeabsichtigten Tastendruck den ganzen Text löscht.
Eine vollständige Liste aller Funktionstasten finden Sie auf S. 75.

TEXT AUSDRUCKEN

Nachdem Sie einen Text fertiggeschrieben haben, möchten Sie ihn vielleicht noch auf Papier ausdrucken oder auf einer Cassette speichern. Wenn Sie den Text ausdrucken wollen, brauchen Sie natürlich einen Drucker, der mit dem Computer verbunden sein muß (am „PRINTER"-Anschluß hinten).
Schalten Sie den Drucker ein und auf Empfangsbereitschaft ('on line'). Nun drücken Sie die Funktionstaste 3. Eine Mitteilung auf dem Bildschirm wird Sie daran erinnern, daß der Drucker vorbereitet sein muß. Drücken Sie 'ENTER' und der Computer wird Ihren Text ausdrucken.

TEXTE AUF CASSETTE

Texte können genauso wie Programme (siehe S. ??) auf Cassette gespeichert werden, nur mit dem Unterschied, daß man den Namen, unter dem der Text abgespeichert werden soll, nicht in Anführungszeichen setzt.
Wenn Sie einen Text abspeichern wollen, drücken Sie die Funktionstaste 2. Der Computer wird Sie nach einem Namen für Ihren Text fragen. Nachdem Sie diesen eingetippt haben (z.B. BRIEF), schalten Sie den Cassetten-Rekorder auf 'Aufnahme' und drücken Sie 'ENTER'. Ihr Text wird auf die Cassette geschrieben.
Später kann man natürlich jeden Text, den man auf Cassette gespeichert hat, wieder auf den Bildschirm bringen. Der Text wird genauso wie ein Programm geladen: Drükken Sie dazu die Funktionstaste 1, tippen Sie den Namen des Textes – z.B. BRIEF –, und drücken Sie 'ENTER'. Der Text erscheint auf dem Bildschirm und kann weiter bearbeitet oder umgeschrieben werden.

RÜCKKEHR ZU BASIC

Wenn Sie mit der Textbearbeitung fertig sind und zu BASIC zurückkehren wollen, drücken Sie die Funktionstaste 8. Der Computer wird Sie fragen, ob Sie zu BASIC zurückschalten wollen. Drücken Sie 'ENTER' (wenn Sie ein anderes Modul – z.B. LISP – anstelle des IS-BASIC-Moduls eingesteckt haben, müssen Sie den entsprechenden Namen eingeben). Falls Sie die Funktionstaste 8 unbeabsichtigt gedrückt haben, können Sie das Kommando mit 'ESC' wieder rückgängig machen.
Achtung: Die deutsche Textbearbeitung funktioniert nur, wenn das BASIC-Modul ordnungsgemäß gesteckt ist. Ist kein Modul gesteckt, können Sie nach Drücken einer Taste die englische Textbearbeitung benutzen. Beachten Sie aber, daß die Tastatur Ihres Computers mit englischem Zeichensatz belegt ist.

Sie haben bereits einige der sogenannten Funktionstasten benutzt. Das sind die blauen Tasten oberhalb des Schreibfelds, die von 1 bis 8 numeriert sind.
Diese Tasten können vom Benutzer mit fast beliebigen Funktionen belegt werden. Sie werden bereits beim Einschalten des Computers mit nützlichen Funktionen versehen, aber man kann sie leicht „umdefinieren". Zum Beispiel kann man folgendes eingeben:

```
SET FKEY 1 "CLEAR SCREEN"
```

Die Zahl nach FKEY bestimmt, welche Funktionstaste umdefiniert wird, während die Funktion, die diese Taste ausführen soll, in Anführungszeichen stehen muß.
Nachdem Sie die obige Anweisung eingegeben haben, können Sie die Funktionstaste 1 ausprobieren. Auf dem Bildschirm erscheint CLEAR SCREEN. Wenn Sie jetzt 'ENTER' drücken, wird der Bildschirm tatsächlich gelöscht. Man kann also häufig verwendete Befehlswörter auf die Funktionstasten legen und dann einfach diese Tasten drücken, anstatt die ganzen Wörter auszuschreiben. Es ist sogar möglich, auch noch das 'ENTER' in die Belegung aufzunehmen, indem man

```
&CHR$(13)
```

hinzufügt, z.B.

```
SET FKEY 1 "CLEAR SCREEN" &CHR$(13)
```

Wenn Sie nun die Funktionstaste 1 drücken, wird der Befehl direkt ausgeführt und der Bildschirm gelöscht.
Nach einer Umbelegung der Funktionstasten können Sie zur Standardbelegung zurückkehren, indem Sie die 'RESET'-Taste hinten am Gerät betätigen.

Die folgende Tabelle zeigt Ihnen, mit welchen Kommandos die Funktionstasten beim Einschalten des Computers automatisch belegt werden. Beachten Sie, daß Sie die Funktionstasten auch in Verbindung mit 'SHIFT', 'CTRL' und 'ALT' verwenden können, so daß jede Taste vierfach belegt werden kann. Wie Sie im letzten Kapitel erfahren haben, sind die Funktionstasten im Textverarbeitungs-Modus ein wenig anders belegt. Darüber gibt die kleine Tabelle Auskunft.

nur BASIC **BASIC oder Textverarbeitung**

Taste	NORMAL	+ SHIFT	+ CTRL	+ ALT
1	START Startet das Programm. Falls keins geladen ist, wird Programm von Diskette geladen und gestartet, falls keine Diskettenstation, dann von Cassette.	CONTINUE Setzt Programmablauf nach STOP fort.	FORMATIEREN Formatiert Absatz, d.h. bringt alle Zeilen auf gleiche Länge (innerhalb der Randeinstellungen).	BLOCKSATZ Fügt Leerzeichen ein, um den rechten Rand zu begradigen.
2	LIST Bringt das Programm auf den Bildschirm.	LLIST Druckt das Programm auf dem Drucker aus.	ZENTRIEREN Zentriert die Zeile.	TABS LÖSCHEN Löscht alle Tabulator-Stops.
3	AUTO Automatische Zeilennumierung. Ausschalten mit 'STOP'.	RENUMBER Numeriert das ganze Programm in Zehnerschritte um.	TAB SET./LÖSCH. Setzt oder löscht Tabulator in der augenblicklichen Cursorspalte.	LINEAL Formatzeile mit Rand- und Tabulator-Angaben ein- bzw. ausblenden.
4	REMOTE 1 Rem 1 Schalter ein- bzw. ausschalten.	REMOTE 2 Rem 2 Schalter ein- bzw. ausschalten.	LINKER RAND Linken Rand auf Cursorspalte einstellen.	RECHTER RAND Rechten Rand auf Cursorspalte einstellen.
5	TEXT Schaltet auf Text-Modus. Löscht augenblickliche Text- und Graphikseiten.	DISPLAY TEXT Schaltet von der Graphik- auf die Textseite um, ohne dabei etwas zu löschen.	RÄNDER FREI Gibt die eingestellten Ränder für Text wieder frei.	RÄNDER NORMAL Setzt Ränder auf Standardwerte.
6	GRAPHICS Schaltet auf Graphik-Modus. Löscht augenblickliche Text- und Graphikseiten.	DISPLAY GRAPHICS Schaltet von der Text- auf die Graphikseite um, ohne dabei etwas zu löschen.	ABSATZ RAUF Bewegt Absatz eine Zeile nach oben.	ABSATZ RUNTER Bewegt Absatz eine Zeile nach unten.
7	CLICK Tastatur-Klick ein- bzw. ausschalten.	SPEAKER Schaltet Computer-Ton aus. Erneut drücken, um den Ton wieder einzuschalten.	ZEILENFARBE Farbwahl für die Cursorzeile.	ABSATZFARBE Farbwahl für den Absatz.
8	INFO Informiert über die Programme, die im Speicher sind.	TYPE Textverarbeitungs-Modus einschalten.		

'NORMALE' FUNKTIONEN BEI DER TEXTVERARBEITUNG

1. LADEN Lädt Text von der Cassette.	2. ABSPeichern Speichert Text auf Cassette oder Diskette.	3. DRucKen Sendet Text an Drucker.	4. INFO Information über Textverbeitungs-Funktionen.
5. TEXT 80 Löscht Text und richtet 80-Zeichen-Modus ein.	6. TEXT 40 Löscht Text und richtet 40-Zeichen-Modus ein.	7. KLICK Tastatur-Klick ein- bzw. ausschalten.	8. ENDE Ausstieg aus der Textverarbeitung.

Im Aufstell-Handbuch haben Sie gelernt, wie man ein Programm von Cassette in den Computerspeicher lädt. Sie können auch einen Cassetten-Rekorder verwenden, um eigene Programme auf Band zu speichern.
Vielleicht wollen Sie ein Programm, das Sie im ersten Teil dieses Handbuches eingegeben – und womöglich verändert – haben, abspeichern. Wir wollen jedenfalls davon ausgehen, daß Sie zur Zeit ein Programm im Computerspeicher haben und es auf Band schreiben wollen.
Zunächst sollten Sie sich vergewissern, daß der Cassetten-Rekorder ordnungsgemäß mit dem Computer verbunden ist. Über die Buchse 'OUT' auf der Rückseite des Computers werden Daten zum Rekorder geschickt. Stekken Sie einen der vier Verbindungsstecker in diese Buchse und das andere Ende (d.h. den Stecker mit derselben Farbe) in die Aufnahmebuchse des Rekorders. Diese trägt meist die Aufschrift „MIC".
Nun stecken Sie einen der kleineren Stecker des Verbindungskabels in die Computerbuchse 'REM2' und das andere Ende (wiederum der Stecker mit derselben Farbe) in den Fernsteuerungsanschluß des Rekorders. Diese hat meist die Aufschrift 'REM'.
Es spielt keine Rolle, ob der Computer dabei an- oder ausgeschaltet ist!
Nun müssen Sie einen Namen für das Programm, das Sie abspeichern wollen, erfinden. Der Name kann bis 28 Zeichen lang sein und Buchstaben, Zahlen sowie die folgenden Interpunktionszeichen enthalten: '.', '-', '–', ' ' und '/'. Sie könnten das Programm 'Mein-erstes-Programm' nennen, aber wir wollen annehmen, daß es „Programm1" heißen soll.
Stellen Sie nun den Cassetten-Rekorder auf Aufnahme und tippen Sie

```
SAVE "Programm1"
```

und drücken Sie 'ENTER'. Sie sehen die Mitteilung

```
SAVING "Programm1"
```

und, sobald das Programm abgespeichert wurde, die Mitteilung

```
OK_
```

Sofern Ihr Cassetten-Rekorder über eine Fernsteuerung verfügt und Sie diese wie oben beschrieben angeschlossen haben, wird der ENTERPRISE den Rekorder automatisch ein- und ausschalten.
Während das Programm auf Band geschrieben wird, hören Sie die Signale des Computers leise im eingebauten Lautsprecher. So haben Sie immer die Kontrolle, daß alles richtig abläuft.

SICHER IST SICHER

Natürlich ist es immer besser, nochmals zu kontrollieren, ob das Programm einwandfrei abgespeichert wurde. Das können Sie mit der Anweisung VERIFY schnell erledigen.
Zunächst müssen Sie die Cassette bis zu der Stelle zurückfahren, wo sie vor dem Abspeichern des Programms war.
Wenn Sie die Fernsteuerung des Rekorders mit dem Computer verbunden haben, werden Sie feststellen, daß der Rekorder blockiert ist und Sie nicht zurückspulen können. Um ihn wieder frei zu bekommen, tippen Sie TOGGLE REM 1 oder drücken Sie einfach SHIFT + FUNCTION 4.
Nachdem Sie die Cassette zurückgespult haben, tippen Sie

```
VERIFY "Programm1"
```

Starten Sie nun den Cassetten-Rekorder und drücken Sie 'ENTER'. Auch hier werden Sie merken, daß der Computer den Rekorder steuert.
Wenn das Programm fehlerfrei abgespeichert wurde, erscheint nun:

```
ok
```

Wenn es Probleme beim Abspeichern gegeben hat, sehen Sie stattdessen eine Fehlermeldung. In einem solchen Fall sollten Sie die Verbindungen zwischen Cassetten-Rekorder und Computer überprüfen und das Programm erneut abspeichern.

VERKETTUNG VON PROGRAMMEN

Zu einem Programm, das Sie im Computerspeicher haben, können Sie leicht ein weiteres Programm vom Band hinzufügen. Das tun Sie mit MERGE. Diese Anweisung ist besonders nützlich, wenn Sie viele kleine Programme zu einem großen zusammenfügen. Allerdings müssen Sie darauf achten, daß die Zeilennummern nir-

gendwo überlappen. Das hat den folgenden Grund.
Nehmen wir an, Sie haben ein kleines Programm mit den Zeilennummern 100, 110, 140 und 160 im Computerspeicher. Sie wollen ein zweites Programm mit den Zeilennummern 140, 160, 180 und 200 hinzufügen. Wenn Sie das mit MERGE tun, werden die Zeilen 140 und 160 im ursprünglichen Programm durch die neuen Zeilen mi. diesen Nummern überschrieben und gehen somit verloren. Sie müssen also vorher das erste Programm mit RENUMBER umnumerieren, so daß es keine Überschneidungen gibt.
MERGE funktioniert genau wie das normale Laden eines Programms vom Band (wie das geht, wird in der Aufstell Anleitung genau beschrieben). Allerdings wird der Inhalt des Computerspeichers nicht zuvor gelöscht. Wenn Sie ihrem augenblicklichen Programm ein weiteres hinzufügen wollen, spulen Sie den Cassetten-Rekorder bis zu der Stelle, wo das neue Programm beginnt, und tippen Sie

```
MERGE "------"
```

wobei Sie den Namen des neuen Programms angeben. Starten Sie den Cassetten-Rekorder und drücken Sie ENTER. Beide Programme werden automatisch zusammengefügt. Versuchen Sie es!

DATEI

Programme, die auf Cassette (oder auf anderen Datenträgern, wie z.B. Floppy Disks) gespeichert sind, nennt man auch „Dateien". Auch das englische Wort „Files" wird häufig verwendet. Eine Datei kann auch eine Sammlung von Informationen (etwa Adressen oder Telephonnummern) enthalten, die vom Computer verarbeitet werden können.

DISKETTEN

Wenn am Computer eine Diskettenstation angeschlossen ist, gelten alle oben behandelten Befehle unverändert. Allerdings wird der Computer die Programme automatisch auf Diskette schreiben oder von Diskette laden, was im übrigen sehr viel schneller geht.

Bisher haben wir in diesem Handbuch „Strings" (das sind Zeichenketten) nicht sehr ausführlich behandelt. Strings sind eine von zwei möglichen Arten von Informationen – Zahlen sind die andere –, die der Computer verarbeiten kann. Und beim Computer geht es ja um die Verarbeitung von Informationen.
Es ist wahrscheinlich bei Ihren ersten Versuchen mit dem ENTERPRISE deutlich geworden, daß der Computer genau das tut, was man von ihm verlangt, mit Informationen, die man ihm eingibt. Im Grunde ist das Programmieren lediglich eine besondere Art, Informationen zu sammeln, zu sortieren und zu verarbeiten.
In einem früheren Kapitel haben wir Strings mit wörtlicher Rede verglichen. Wenn man zitiert, wiederholt man genau, was gesagt wurde, ohne notwendigerweise das Gesagte auch noch zu verstehen. Das ist das Prinzip hinter dem Begriff des Strings. Der Computer versteht nicht, was sie bedeuten, er behandelt sie wie zufällige und bedeutungslose Zeichen.
Alles, was in einem Programm in Anführungszeichen gesetzt ist, betrachtet der Computer als String. Hier ist ein Beispiel:

„Wie alt sind Sie?"

So etwas könnte man ohne weiteres in einem Buch, als wörtliche Rede, finden:

„Wie alt sind Sie?", fragte Hans.

Beachten Sie, daß in wörtlicher Rede mitunter auch Zahlen vorkommen können:

„Ich bin 22", sagte Ines.

Geben Sie folgendes in den Computer ein:

```
PRINT 2*2
```

Diese Anweisung veranlaßt den Computer, auch im Direktmodus eine kleine Rechenoperation durchzuführen und das Ergebnis auf den Bildschirm zu bringen. Die Anweisung besagt: Schreibe den Wert von 2 mal dem Wert von 2 auf den Bildschirm. Also schreibt er die 4.

```
PRINT "2*2"
```

Das zweite Beispiel sieht fast genauso aus wie das erste, aber die Anführungszeichen machen aus 2*2 einen String. Die Anweisung an den Computer lautet in diesem Fall: Schreibe 2, dann * und dann 2 auf den Bildschirm. Also schreibt er 2*2.
Mit der Funktion LEN kann der Computer die Länge einer Zeichenkette bestimmen, was häufig sehr nützlich ist. Versuchen Sie es mit dem folgenden Programm:

```
100 INPUT PROMPT "Geben Sie ein Wort ein : ": W$
110 LET A = LEN(W$)
120 PRINT "Das Wort "; W$; "ist "; A; "Zeichen lang."
130 END
```

In Zeile 100 bekommt der Computer ein Wort vom Benutzer und legt dieses unter dem Variablennamen W$ ab. Solche „String-Variablen" funktionieren ähnlich wie normale Zahlen-Variablen, mit dem Unterschied, daß sie für Zeichenketten reserviert sind. Um String-Variablen zu kennzeichnen, steht am Ende des Variablennamen ein Dollarzeichen ($). Da am Ende der INPUT-Anweisung W$ steht, erwartet der Computer eine Zeichenkette, nicht einen Wert.
Wenn Sie das Programm starten, werden Sie feststellen, daß der Computer die Länge des eingegebenen Wortes einwandfrei bestimmt.

LEERZEICHEN

Geben Sie im obigen Programm statt eines Wortes einen ganzen Satz ein. Sie werden sehen, daß der Computer auch die Länge des Satzes bestimmt. Dabei zählt er jedoch auch die Leerzeichen mit, die zwischen den einzelnen Wörtern stehen.
Ein Leerzeichen bedeutet vielleicht nichts für Sie, und es wäre verständlich, wenn Sie meinten, daß Leerzeichen nicht zählen. Aber für den Computer sind sie Zeichen wie alle anderen auch. Ersetzen Sie im obigen Programm die Zeile 100 durch

```
100 LET W$ = "          "
```

Für den Computer ist W$ dann "   " eine Zeichenkette mit zehn Leerzeichen.

Es gibt Strings, die wirklich nichts enthalten (man nennt sie „Null-Strings" oder auch „leere Strings"). Das sieht folgendermaßen aus:

""

Wie Sie sehen, enthält dieser String nichts, nicht einmal ein Leerzeichen.

STRINGS INNERHALB VON STRINGS

Der ENTERPRISE kann recht vielseitig mit Strings umgehen. Zum Beispiel kann er einen Teil aus einem String herausholen. Versuchen Sie es mit dem folgenden Programm:

```
100 LET A$ = "Eine ganze Menge Wörter"
110 LET B$ = A$(12:23)
120 PRINT A$
130 PRINT B$
140 END
```

In Zeile 100 wird A$ mit einer Zeichenkette belegt. In Zeile 110 wird B$ mit einem Teil von A$ gleichgesetzt, und zwar mit dem Teil, der sich vom 12. bis zum 23. Zeichen von A$ erstreckt. Der Computer hat diese Zeichen aus A$ herauskopiert, unter dem Namen B$ gespeichert und dann auf den Bildschirm gebracht.
Es ist übrigens nicht nötig, in dem obigen Programm den Teil-String zunächst unter B$ abzuspeichern und dann erst auszudrucken. Sie können Zeile 110 löschen und Zeile 130 folgendermaßen verändern:

```
130 PRINT A$(12:23)
```

Das Programm läuft genauso ab. Welche Methode Sie verwenden, hängt in erster Linie von der Verwendung des Teil-Strings ab. Wenn Sie ihn ständig brauchen, dann ist es besser, ihm einen eigenen Namen zu geben. Wenn nicht, dann genügt die zweite Methode.
Teil-Strings können oft recht nützlich sein. Man kann beispielsweise den Anfangsbuchstaben eines Namens sehr leicht mit NAME$(1:1) ermitteln.

INKEY$

Alle BASIC-Kommandos, die auf Zeichenketten angewendet werden, nennt man String-Funktionen. Die meisten enden auch mit dem Dollarzeichen, um anzuzeigen, daß

ihre Anwendung wiederum einen String produzieren wird. Eine sehr wichtige String-Funktion ist INKEY$.
INKEY$ erlaubt Ihnen, eine Taste zu drücken, während der Computer arbeitet und so den Ablauf des Programms zu beeinflussen. Es hat Ähnlichkeit mit INPUT, allerdings wird der Unterschied bald klar sein. Versuchen Sie folgendes Programm:

```
100 PRINT "Haben Sie dieses Kapitel verstanden?"
110 PRINT "Drücken Sie j oder n"
120 DO
130         LET A$ = INKEY$
140 LOOP UNTIL A$ <> ""
150 IF A$ = "j" THEN
160         PRINT "Das ist sehr erfreulich!"
170 ELSE IF A$ = "n" THEN
180         PRINT "Das ist aber betrüblich!"
190 ELSE
200         PRINT "Falsche Taste!"
210 END IF
220 END
```

In Zeile 130 speichert der Computer die Taste, die gerade gedrückt wird, unter den Variablennamen A$. Die DO-LOOP-Schleife wird so lange durchlaufen, bis A$ ungleich "" (Null-String) ist, d.h. bis tatsächlich eine Taste gedrückt wurde. Sobald das geschieht, läuft das Programm weiter und der Computer prüft, ob j, n oder etwas anderes unter A$ abgespeichert ist und reagiert entsprechend. Sie kennen bereits die IF/THEN (wenn/dann) Anweisung. ELSE bedeutet „sonst". Sie werden wahrscheinlich keine Probleme haben, die Struktur des obigen Programms zu verstehen. Im Kapitel über Entscheidungen werden wir solche Wenn-dann-sonst-Anweisungen ausführlich behandeln.
Und nun zurück zur INKEY$-Funktion. Im Gegensatz zu INPUT verlangt INKEY$ nur einen Tastendruck, und man braucht nicht danach 'ENTER' zu drücken. Das ist sehr nützlich für Spiele, für schnelle Ja-Nein-Fragen, usw. Denken Sie daran, daß 'SHIFT' nicht als Tastendruck gilt und nur in Verbindung mit einer anderen Taste erkannt wird.
Ein weiterer Unterschied zwischen INKEY$ und INPUT ist folgender: Die INKEY$-Funktion läßt den Computer nicht anhalten und auf eine Eingabe warten. Deswegen war die

DO-LOOP-Schleife (Zeilen 120 und 140) im obigen Programm nötig. Wenn man sie wegläßt (versuchen Sie es!), läuft das ganze Programm in Bruchteilen einer Sekunde ab und Sie haben keine Zeit, auf die Frage des Computers zu antworten.
INKEY$ kann man als Variable betrachten, deren Inhalt sich sehr schnell – fünfzigmal in der Sekunde – verändert. Jedesmal, wenn eine Taste gedrückt wird, speichert der Computer das Zeichen für 0,02 Sekunden unter INKEY$ ab. Eine fünfzigstel Sekunde später wird INKEY$ wieder zum Null-String. Deswegen war es nötig, in Zeile 130 des obigen Programms den Inhalt von INKEY$ unter einem separaten Variablennamen (A$) abzuspeichern, so daß der Computer die anschließende Überprüfung des Inhalts (Zeilen 150 – 210) vornehmen konnte.
Machen Sie einen Versuch: Löschen Sie die Zeile 130 und verändern Sie die Zeilen 140, 150 und 170 folgendermaßen:

```
140 LOOP UNTIL INKEY$ <> ""
150 IF INKEY$ = "¡" THEN
170 ELSE IF INKEY$ = "n" THEN
```

Das Programm funktioniert nicht mehr richtig. Auch wenn eine Taste gedrückt und in Zeile 140 unter INKEY$ gespeichert wurde, springt INKEY$ zurück auf "", bevor der Computer mit der Abprüfung in Zeile 150 beginnen kann.
Natürlich ist es nur dann notwendig, den Inhalt von INKEY$ dauerhaft zu speichern, wenn man hinterher damit etwas anfangen will. Oft will man nur das Programm so lange anhalten, bis der Benutzer weitermachen will. In solchen Fällen kann man z.B. die folgende Zeilen in das Programm einbauen:

```
100 PRINT "Drücken Sie eine beliebige Taste zum
    Weitermachen"
110 DO
120 LOOP UNTIL INKEY$ <> ""
```

UCASE$ UND LCASE$

Es gibt viele interessante Dinge, die man mit Strings machen kann. Es ist durchaus möglich, Groß- und Kleinbuchstaben automatisch auszuwechseln. Die beiden Anweisungen, mit denen das geschieht, lauten UCASE$ (für Großbuchstaben) und LCASE$ (für Kleinbuchstaben).

Versuchen Sie folgendes Programm:

```
100 INPUT PROMPT "Geben Sie einige Buchstaben ein:
    ":A$
110 PRINT A$
120 PRINT UCASE(A$)
130 PRINT LCASE(A$)
140 END
```

Die Wörter, die Sie eintippen, werden zunächst unverändert auf den Bildschirm gebracht (Zeile 110), dann in Großbuchstaben (120) und schließlich in Kleinbuchstaben (130).

VAL

Manchmal ist es nützlich, wenn man die Zahlen-Symbole in einer Zeichenkette in eine echte Zahl verwandelt, z.B. "24" in 24. Das geschieht mit dem BASIC-Kommando VAL (für Value = Wert). Versuchen Sie folgendes:

```
100 INPUT PROMPT "Geben Sie einige Zeichen ein: ":A$
110 PRINT A$
120 PRINT VAL(A$)
130 END
```

Bei der Ermittlung des Wertes von A$ berücksichtigt der Computer nur die Zahlen, die vor dem ersten Buchstaben in der Zeichenkette stehen. Wenn Sie beispielsweise 123AB45 eingeben, wird 123 als VAL(A$) ermittelt, und VAL(„AB12345") ist gleich 0.
Sie können übrigens auch umgekehrt eine Zahl in eine Zeichenkette verwandeln. Das BASIC-Kommando dafür ist STR$.
Beispiel: A$ = STR$(123) läßt A$ = "123" werden.

VERKETTUNG VON STRINGS

Wie wir bereits gesehen haben, kann man aus einem String einen Teil-String herausholen. Man kann aber auch Strings aneinanderketten. Dazu setzt man ein kaufmännisches „Und" ('&', auch Et-Zeichen oder Ampersand genannt) zwischen die einzelnen Strings – z.B. A$ & B$ & C$.
Im folgenden Beispiel werden verschiedene String-Funktionen verwendet:

```
100 INPUT PROMPT "Bitte tippen Sie ein Wort ein: ":
    WORT$
```

```
110 LET A = LEN(WORT$)
120 LET ABKUERZUNG$ = WORT$(1:1) & WORT$(A:A)
    & "."
130 PRINT "Die Abkürzung von "; WORT$; " ist ";
    ABKUERZUNG$
140 END
```

In Zeile 100 wird das Wort, das Sie eintippen, unter dem Variablennamen WORT$ abgelegt. In Zeile 110 stellt der Computer fest, wie lang das Wort ist und speichert diesen Wert unter A ab. Zeile 120 verdient einige Aufmerksamkeit: Hier wird ein String gebildet aus dem Anfangsbuchstaben des Wortes (WORT$(1:1)), dem letzten Buchstaben (WORT$(A:A)) und einem Punkt. Dieser neue String wird unter ABKUERZUNG$ gespeichert. 130 bringt das Ergebnis auf den Bildschirm. Dabei muß man beachten, daß es nicht erlaubt ist, in Variablennamen deutsche Umlaute zu verwenden. Der Grund dafür liegt in den internationalen grammatischen Regeln der Sprache BASIC.
Für Leute, die die Übersicht behalten können, lassen sich natürlich die Zeilen 110 – 130 zusammenfassen:

```
110 PRINT "Die Abkürzung von "; WORT$; " ist ";
    WORT$(1:1) & WORT$(LEN(WORT$):LEN(WORT$))
    & "."
```

wobei hier die Et-Zeichen auch durch Semikolons ersetzt werden können.

Sie haben bereits einige kurze Programme kennengelernt, in denen wir „Schleifen" eingesetzt haben, damit der Computer einen Vorgang wiederholt. Immer wenn eine Operation mehrfach hintereinander ausgeführt werden muß, ist es zweckmäßig, solche Schleifen zu verwenden – statt die betreffenden Anweisungen immer wieder in BASIC abzuschreiben.
Mit Schleifen kann man, wie Sie bald erkennen werden, den ganzen Programmablauf auf einfache Art steuern.
Mit einer Ausnahme (GOTO, welches auch als Schleife betrachtet werden kann und im Kapitel über Minimal-BASIC behandelt wird) beginnt jede Schleife mit einer besonderen Anweisung, die den Anfang markiert. Sie endet mit einer Zeile, die den Computer veranlaßt, so oft wie nötig die Schleife zu wiederholen.
Wir wollen mit den einfachsten Schleifen beginnen: DO/LOOPs. Sie sind Ihnen im ersten Teil dieses Handbuchs bereits begegnet und sehen etwa folgendermaßen aus:

```
100 INPUT PROMPT „Bitte geben Sie eine Zahl ein und ich
    werde das 'Einmaleins' dafür ausdrucken. ": A
110 LET B = A
120 LET Z = 1
130 CLEAR SCREEN
140 DO
150         PRINT Z;" mal "; A; " ist gleich "; B
160         LET B = B + A
170         LET Z = Z + 1
180 LOOP UNTIL Z = 21
190 END
```

In Zeile 110 wird die Zahl, die Sie eingegeben haben (Zeile 100), zusätzlich unter der Variablen B abgespeichert. Ein Zähler Z wird in 120 auf 1 gesetzt. In Zeile 140 fängt die Schleife an. Zunächst wird einmal die Zahl ausgedruckt (Z ist gleich 1 und B ist gleich A) und anschließend der Wert von B um die Zahl A erhöht (Zeile 160). Das bereitet den nächsten Durchgang vor. In Zeile 170 wird der Zähler Z um eins erhöht. Am Schleifenende (Zeile 180) prüft der Computer, ob der Zähler den Wert 21 erreicht hat und springt immer wieder zum Schleifenanfang (Zeile 140), sofern dies nicht der Fall ist.
Mit DO/LOOP-Schleifen kann man also den Computer gewissermaßen im Kreise laufen lassen, bis eine bestimmte Bedingung erfüllt ist. Diese Bedingung kann mit

UNTIL (= bis, so lange nicht) oder WHILE (= während, so lange wie) angegeben werden. Der Unterschied dürfte klar sein: WHILE Z = 21 in Zeile 180 würde bewirken, daß die Schleife nur einmal durchlaufen wird, da der Computer sofort feststellt, daß Z nicht gleich 21 ist.
Das Bedingungswort (UNTIL oder WHILE) können Sie übrigens auch am Anfang der Schleife verwenden: DO WHILE oder DO UNTIL funktionieren ebensogut, besonders in einem Programm wie das oben angegebene, das keine besonders subtile Abprüfung von Bedingungen verlangt.
Die Bedingung sollte immer dann am Anfang einer Schleife stehen, wenn der Computer unter gewissen Umständen die Anweisungen in der Schleife überhaupt nicht ausführen soll. Die Bedingung wird gleich zu Anfang abgeprüft und, wenn sie nicht zutrifft, kommt der Computer überhaupt nicht in die Schleife. Steht die Bedingung am Ende, wird die Schleife auf jeden Fall mindestens einmal durchlaufen.

SCHLEIFE VORZEITIG VERLASSEN

Mit EXIT können Sie den Computer jederzeit aus einer Schleife aussteigen lassen. Im obigen Programm kann man z.B. die Zeile

```
145 IF A > 10000 THEN EXIT DO
```

einfügen, um zu verhindern, daß eine zu große Zahl eingegeben wird. Ist das der Fall, springt der Computer gleich zur Zeile 190.
Da EXIT auch für die im nachfolgenden behandelten FOR/NEXT-Schleifen verwendet werden kann, muß man immer angeben, was für eine Schleife man verlassen will – EXIT DO oder EXIT FOR.
EXIT ist die einzig richtige Art, eine Schleife vorzeitig zu verlassen. Sie sollten das nie mit GOTO (siehe Kapitel über Minimal-BASIC) tun, da Sie damit leicht sich selbst und den Computer verwirren könnten. Er glaubt nämlich, immer noch in einer Schleife zu sein, nachdem er längst mit etwas anderem beschäftigt ist. Nach EXIT weiß der Computer, daß er die Schleife verlassen hat und geht zur Tagesordnung – zur ersten Zeile nach der Schleife – über.

FOR/NEXT-SCHLEIFEN

FOR/NEXT-Schleifen sind anders als DO/LOOPs und haben ihre bestimmten Anwendungen. Betrachten Sie das folgende Programm:

```
100 INPUT PROMPT "Wie oft soll ich die Schleife
    durchlaufen? ": X
120 FOR P = 1 TO X
130         PRINT "herum und ..."
140         WAIT 1
150 NEXT P
160 PRINT "Fertig!"
170 END
```

Erkennen Sie, daß die Zahl, die Sie eingeben, gleich der Anzahl der Schleifendurchläufe ist? Mit FOR/NEXT-Schleifen kann man schnell und unkompliziert angeben, wie oft die Schleife durchlaufen werden soll.
Beachten Sie, daß die FOR-Bedingung in einer FOR/NEXT-Schleife gleich zu Anfang geprüft wird. Wenn Sie im obigen Beispiel die Zahl 0 (oder –10) eingeben, wird die Schleife überhaupt nicht durchlaufen.
Bei FOR/NEXT-Schleifen gibt es einen Zähler (im obigen Beispiel war das P), der automatisch vom Computer bei jedem Durchlauf um eins erhöht wird. Man kann den Zähler um jeden gewünschten Betrag erhöhen, beispielsweise um 1000 oder 0.2 oder 466.666. Man kann den Computer auch rückwärts zählen lassen. Das Schlüsselwort für diese Anweisungen ist STEP (= Schritt). Hier ist ein kleines Programm, das rückwärts in Zweierschritten zählen kann:

```
100 FOR I = 40 TO 0 STEP -2
110         PRINT I
120 NEXT I
130 END
```

Der Computer kann nur dann von 40 bis 0 zählen, wenn man eine zusätzliche STEP-Anweisung hinzufügt. Wenn STEP fehlt, wird grundsätzlich vorwärts in Einerschritten gezählt. Beachten Sie auch, daß der Zähler selbst ausgedruckt werden kann (Zeile 110). Sogar PRINT I*10 oder PRINT 2*I-1 ist möglich.
Mit einer FOR/NEXT-Schleife können wir das „Einmaleins„-Programm verkürzen:

```
100 INPUT PROMPT "Bitte geben Sie eine Zahl ein und ich
    werde das >Einmaleins< dafür ausdrucken. ": A
110 FOR P = 1 TO 20
```

```
120         PRINT P; " mal "; A; " ist gleich "; P * A
130 NEXT P
140 END
```

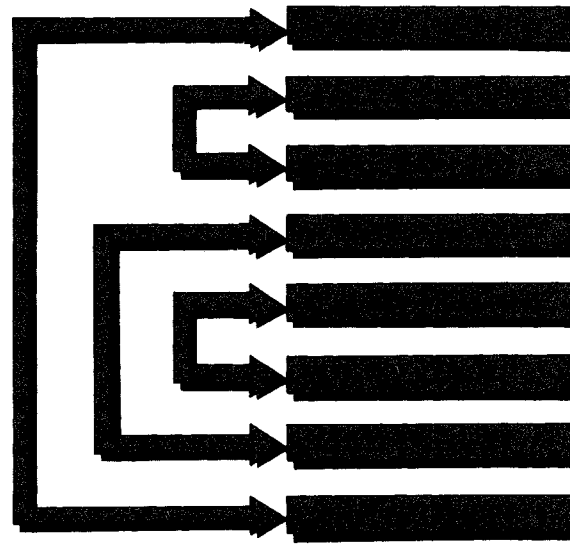
Das Programm tut dasselbe wie die DO/LOOP-Schleife, aber auf eine andere Art. DO/LOOPs sind etwas leichter verständlich, besonders wenn man keine große Erfahrung mit Computern besitzt.
Vielleicht möchten Sie versuchen, solche Programme wie das obige etwas auszuweiten. Wenn Sie Kinder oder kleinere Geschwister haben, kann man für sie lustige Lernprogramme basteln. Aber diesmal wollen wir Ihnen nichts mehr sagen! Programmieren ist ein Abenteuer, bei dem Sie vieles selbst herausfinden sollen. Die besten Ideen stammen aus der Praxis, wenn Sie mit Ihrem Computer selbst experimentieren.

VERSCHACHTELTE SCHLEIFEN

Schleifen kann man auch innerhalb von Schleifen anbringen. Das klingt vielleicht merkwürdig, ist aber relativ einfach zu verstehen. Hier ist ein Beispiel:

```
100 FOR P = 1 TO 20 !          Erste Schleife beginnt
110   PRINT P,
120   FOR Q = 1 TO 2 !         Zweite Schleife beginnt
130     PRINT P + 10; P - 10
140   NEXT Q !                 Zweite Schleife endet
150 NEXT P !                   Erste Schleife endet
160 END
```

Wichtig ist nur, daß man die zweite Schleife nicht außerhalb der ersten beendet. Wir hätten auch innerhalb der zweiten Schleife eine dritte einbauen können, und eine weitere in die dritte einbetten können. Das könnte folgendermaßen aussehen:



Wie Sie sehen, gibt es keine Überschneidungen. Eine so komplexe Verschachtelung von Schleifen benötigt jedoch große Aufmerksamkeit. Um Ihnen dabei zu helfen, rückt der ENTERPRISE die Schleifen unterschiedlich ein, um Anfang und Ende leichter erkennbar zu machen. Kommentarzeilen helfen auch bei der „Buchhaltung", und Sie sollten wie im letzten Programm ausgiebig davon Gebrauch machen. Wir werden aber bald sehen (S. ??), daß man anstelle von vielen kleinen Schleifen auch selbstdefinierte Funktionen verwenden kann, womit der Zweck und Fluß des Programms gewahrt bleibt.

ENTSCHEIDUNGEN, ENTSCHEIDUNGEN!

In Romanen kann man häufig von einem „Elektronenhirn" lesen, das ebenso intelligent ist wie ein Mensch. Vielleicht haben Sie früher auch, ebenso wie die meisten Menschen, die Intelligenz von Computern überschätzt.
Computer können nicht „denken", aber sie können einfache Vergleiche durchführen und Entscheidungen treffen.

Wir wollen mit den Vergleichen beginnen. Erinnern Sie sich an die Vergleichsoperatoren aus dem ersten Teil dieses Handbuchs? Diese sind der Schlüssel für die Art, in der Computer Vergleiche anstellen und Entscheidungen aufgrund solcher Vergleiche treffen.
Versuchen Sie das folgende Programm. Sie werden erkennen, wie der Computer mit Hilfe eines Vergleichsoperators feststellt, ob zwei Zeichenketten gleich sind oder nicht.

```
100 INPUT PROMPT "Wie heißen Sie? ": A$
110 PRINT 120 IF A$ = "Martin" THEN PRINT
    "Hallo Martin!"
130 IF A$ <> "Martin" THEN PRINT "Tut mir leid, ich
    rede nur mit Martin."
140 END
```

IF/THEN (wenn/dann) ist eine von zwei wichtigen Arten, wie der ENTERPRISE Entscheidungen trifft. Meist werden logische Operatoren verwendet, um festzustellen, ob Variablen oder Zahlen bestimmte Bedingungen erfüllen. IF/THEN benutzt man nicht, um Berechnungen durchzuführen, sondern um das Programm zu steuern: Wenn eine bestimmte Eigenschaft in einer Zahl oder einer Zeichenkette vorhanden ist, dann tue folgendes...!
Sie könnten ein Ratespiel programmieren, in dem jeder Spieler eine bestimmte Anzahl von Versuchen bekommt. Sie bestimmen eine Variable, die nach jedem Versuch um eins erhöht wird. Eine IF/THEN-Anweisung prüft jedesmal, ob die vorgegebene Anzahl von Versuchen erreicht ist. Sobald dies der Fall ist, teilt der Computer dem augenblicklichen Spieler seine Punktzahl mit, setzt die Variable wieder auf Null und läßt den zweiten Spieler beginnen.

IF-BLÖCKE

Manchmal passen die ganzen Bedingungen, Vergleiche und Entscheidungen nicht in eine Zeile. In solchen Fällen können Sie einen sogenannten IF-Block verwenden, der sich über mehrere Zeilen erstreckt. Damit kann man eine ganze Reihe von Bedingungen abprüfen und entsprechende Entscheidungen treffen lassen.
Wie das funktioniert, erkennen Sie am besten im nächsten Beispiel (ELSE bedeutet „sonst"):

```
100 INPUT PROMPT "Wie heißen Sie? ": A$
110 IF A$ = "Martin" THEN
```

```
120         PRINT "Hallo Martin!"
130 ELSE IF A$ = "Agnes" THEN
140         PRINT "Hallo Agnes!"
150 ELSE
160         PRINT "Tut mir leid, ich rede nur mit Martin
            oder Agnes."
170 END IF
180 END
```

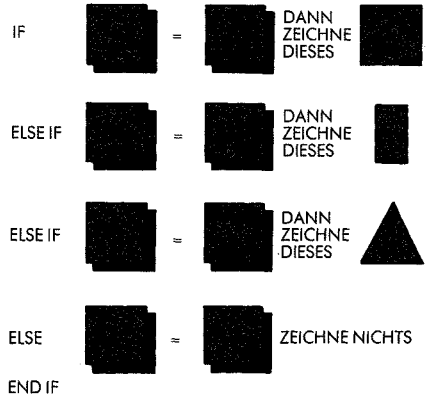
Der Computer hat drei Alternativen. Wenn Martin oder Agnes ihren Namen eingeben (Zeile 100), dann ist die Bedingung in Zeile 110 bzw. 130 erfüllt und die jeweilige Person wird begrüßt. Andernfalls (Zeile 150 – ELSE ohne weitere Bedingung) winkt der Computer ab. Mit END IF (Zeile 170) teilt man dem Computer mit, daß er keine Bedingungen mehr abprüfen muß.
ELSE muß übrigens immer am Anfang einer Zeile stehen, nie zusammen mit der IF/THEN-Anweisung. Mit IF/THEN sagt man dem Computer, was er tun soll, wenn das und das zutrifft; mit ELSE was er anderenfalls tun soll.
Unser nächstes Beispiel enthält auch einen IF-Block. Sie finden in diesem Programm auch eine Reihe von Funktionen, die wir noch nicht behandelt haben. Dennoch sollte es Ihnen Spaß machen.

```
100 RANDOMIZE
110 LET A = RND(4)
120 GRAPHICS
130 PLOT 600,300;
140 OPTION ANGLE DEGREES
150 PLOT ANGLE 0;
160 IF A = 0 THEN                    ! Quadrat
170         PLOT FORWARD 100;
180         PLOT LEFT 90;
190         PLOT FORWARD 100;
200         PLOT LEFT 90;
210         PLOT FORWARD 100;
220         PLOT LEFT 90;
230         PLOT FORWARD 100
240 ELSE IF A = 1 THEN               ! Rechteck
250         PLOT FORWARD 50;
260         PLOT LEFT 90;
270         PLOT FORWARD 100;
280         PLOT LEFT 90;
290         PLOT FORWARD 50;
```

```
300         PLOT LEFT 90;
310         PLOT FORWARD 100
320 ELSE IF A = 2 THEN !              Dreieck
330         PLOT FORWARD 100;
340         PLOT LEFT 120;
350         PLOT FORWARD 100;
360         PLOT LEFT 120;
370         PLOT FORWARD 100
380 ELSE
390         TEXT
400       PRINT "Diesmal will ich gar nichts zeichnen!"
410 END IF
420 END
```

In Zeile 110 wird eine Zufallszahl zwischen 0 und 3 erzeugt. Das tut der Computer mit Hilfe einer Formel, die Sie nicht zu kennen brauchen – es sei denn, Sie interessieren sich für die innere Arbeitsweise des Computers. Um zu gewährleisten, daß nicht immer dieselbe Zahl beim Starten des Programms erscheint, setzt man die Anweisung RANDOMIZE an den Anfang (Zeile 100).

Zufallszahlen sind sehr nützlich bei Spielprogrammen, aber sie können auch, wie hier, für die Erzeugung von verschiedenen Mustern auf dem Bildschirm verwendet werden. Sie könnten mit Hilfe einer DO/LOOP beliebig viele Zufallszahlen ausdrucken lassen und würden keine Muster erkennen.
Die Zeilen 120 bis 150 bereiten den Computer auf die Ausgabe von Graphik vor. Wenn die Zufallszahl 0 ist, dann wird ein Quadrat gezeichnet (Zeilen 170 bis 230); wenn sie 1 ist, so erscheint ein Rechteck (Zeilen 250 bis 310); und wenn sie 2 ist, zeichnet der Computer ein Dreieck (Zeilen 330 bis 370). Wenn sie aber keines von diesen ist – also die 3 –, dann zeichnet der Computer gar nichts (Zeile 400).
Aus dem letzten Programm wird sicherlich klar, wie man mit IF/THEN/ELSE recht komplexe Vorgänge steuert und daß man dazu IF-Blöcke verwenden kann (für einfache Entscheidungen reicht meist eine einzige IF/THEN-Zeile). Sie sehen auch, wie IF und ELSE IF die einzelnen Entscheidungen des Computers voneinander trennen. Schematisch sieht das folgendermaßen aus:



Vergessen Sie nie, einen IF-Block mit END IF zu beenden, da der Computer sonst den Rest des Programms nicht mehr ausführen würde (sobald nämlich die letzte IF- oder ELSE IF-Bedingung nicht zutrifft, sucht der Computer nach dem END IF).

SELECT CASE

Mit SELECT CASE kann man ähnliche Vergleiche und Entscheidungen treffen wie bei IF/THEN. Man kann mit CASE eine von mehreren möglichen Entscheidungen besonders kurz und einfach formulieren. Das Wort CASE bedeutet soviel wie „im Falle daß...". Beachten Sie allerdings im nachfolgenden Beispiel, daß der Variablennamen nicht hinter CASE auftaucht: CASE 1, 2, 3 ist richtig, CASE X = 1, 2, 3 falsch. Die Variable steht in der SELECT-Zeile.
Genauso wie bei END IF müssen Sie das Ende eines SELECT-Blocks kennzeichnen – in diesem Fall mit END SELECT.

```
100 CLEAR SCREEN
110 PRINT AT 9,8:"Was soll ich schreiben?"
120 PRINT AT 11,10:"1. Meinen Namen"
130 PRINT AT 12,10:"2. Ihren Namen"
140 PRINT AT 13,10:"3. Unsere beiden Namen?"
145 !-------------------------------------------------
150 DO
160         INPUT AT 16,10, PROMPT "Bitte wählen Sie. ":
A
170 LOOP WHILE A < 1 OR A > 3 OR A <> INT(A)
180 CLEAR SCREEN
```

```
185 !-------------------------------------------------
190 SELECT CASE A
200 CASE 1
210         PRINT AT 9,18: "ENTERPRISE !!!"
220 CASE 2
230         INPUT PROMPT "Dann sagen Sie mir bitte,
            wie Sie heißen ": NAME$
240         PRINT AT 9,18: NAME$
250 CASE 3
260         INPUT PROMPT "Dann sagen Sie mir bitte,
            wie Sie heißen ": NAME$
270         PRINT AT 9,18: NAME$
280         PRINT AT 11,18: "ENTERPRISE "
290 END SELECT
300 END
```

Nachdem in den Zeilen 110 bis 140 dem Benutzer drei verschiedene Möglichkeiten angeboten werden, wartet der Computer in Zeile 160 auf seine Auswahl. Die DO/LOOP-Schleife (150 – 170) sorgt dafür, daß die Antwort zwischen 1 und 3 liegt: Der Computer kehrt zur Zeile 150 zurück, solange („WHILE") A < 1 oder A > 3, oder aber A keine ganze Zahl ist (A <> INT(A)).

In dem SELECT-Block (Zeilen 190 bis 290) trifft der Computer eine Entscheidung, je nachdem, ob A = 1, 2 oder 3 ist. Die Zeilen 200 und 210 besagen beispielsweise: Falls 1 (d.h. A = 1) dann schreibe „ENTERPRISE !!!". Zeile 290 ist notwendig, um dem Computer mitzuteilen, daß keine weiteren Entscheidungen folgen.

Mit IF/THEN/ELSE oder SELECT CASE können Sie ein besonders hübsches Würfel-Programm schreiben. Wie im vorletzten Programm können Sie Zufallszahlen mit RANDOMIZE und RND erzeugen. Diesmal müßten es natürlich Zufallszahlen zwischen 1 und 6 sein, also

```
A = RND(6) + 1
```

Je nachdem, welchen Wert A annimmt, können Sie dann mit Hilfe der Graphik-Funktionen einen Würfel mit der entsprechenden Augenzahl zeichnen lassen.

Ein wichtiger Vorteil von SELECT CASE besteht darin, daß man verschiedene Bedingungen zusammenlegen und eine gemeinsame Entscheidung für sie treffen kann:

```
CASE 2,3 INPUT PROMPT "Dann sagen Sie mir bitte, wie
Sie heißen ": NAME$
```

Ebenso wie bei IF-Blöcken haben Sie die Möglichkeit, CASE ELSE zu verwenden, um alle übrigen Fälle gemeinsam abzudecken. Und Sie können auch Zeichenketten verwenden (SELECT CASE A$, CASE "MARTIN"). Durch Experimentieren werden Sie bald herausfinden, wie vielfältig man mit SELECT CASE programmieren kann. Eine ausführliche Beschreibung aller möglichen Verwendungen von CASE finden Sie im Nachschlage-Teil dieses Handbuches.

Sie haben sicherlich entdeckt, daß Programmieren in erster Linie die Verarbeitung von Information ist. Bislang haben wir nur sehr kleine Datenmengen benutzt – Wörter oder Zahlen, die man per INPUT-Anweisung eingegeben hat, oder Sätze, die der Computer auf den Bildschirm brachte.
Vielleicht haben Sie überlegt, wie man mit größeren Datenmengen umgeht – z.B. einer langen Liste von Namen oder Zahlen, oder ausführlichen Anweisungen für ein Spiel oder einem Buchhaltungsprogramm. Der ENTERPRISE bietet eine ganze Reihe von leistungsfähigen Funktionen, die die Verarbeitung von sehr großen Datenmengen unterstützen.
Wenn Sie beispielsweise eine Liste von Namen oder Telefonnummern speichern wollen, benutzt man dazu ein „Array" (zu Deutsch „Feld" oder „Bereich"). Ein Feld ist eine große Variable, die viele kleine Variablen enthält. Man kann sich das als eine Art Tabelle oder als Seite aus einem Notizheft vorstellen.

NUMERISCHE FELDER

Genauso wie bei Variablen unterscheidet der Computer zwischen Feldern für Zeichenketten und Feldern für numerischen Daten. Beide muß man „deklarieren", z.B.

```
100 NUMERIC ZAHL(1 TO 10)
```

Damit sagt man dem Computer, er soll einen „Behälter" namens ZAHL bereithalten, der 10 Variablen enthält. Diese Variablen sind die „Elemente" des Feldes und man kann beliebige Zahlen darin abspeichern. Die siebte Variable dieses Feldes heißt ZAHL(7) und kann z.B. die Zahl 25 enthalten.
Jetzt fügen Sie die folgenden Zeilen hinzu:

```
110 FOR S = 1 TO 10
120         PRINT "Die "; S; ". Zahl lautet: ";
130         INPUT ZAHL(S)
140 NEXT S
```

Damit können Sie zehn Zahlen abspeichern und jederzeit wieder abrufen. Wenn Sie beispielsweise PRINT ZAHL(9) eingeben, bringt der Computer die Zahl, die Sie an neunter Stelle eingegeben hatten, auf den Bildschirm. Sie können das Programm auch erweitern und den Computer die Summe und den Durchschnitt der eingegebenen Zahlen ausrechnen lassen:

```
150 SUMME = 0
160 FOR T = 1 TO 10
170         SUMME = SUMME + ZAHL(T)
180 NEXT T
190 PRINT "Die Summe der zehn Zahlen ist "; SUMME
200 PRINT "Der Durchschnitt der zehn Zahlen ist ";
    SUMME / 10
```

Wie Sie sehen, kann man die Zahlen abrufen und mit ihnen mathematische Operationen durchführen – oder sie einfach ausdrucken lassen. Wenn Sie die Tageshöchsttemperaturen für den Monat Dezember abspeichern wollen, deklarieren Sie ein Feld NUMERIC DEZEMBER(1 TO 31) und geben Sie die entsprechenden Werte ein. Die jeweilige Temperaturangabe wird in einem Feldelement abgespeichert, der Elementname – z.B. DEZEMBER(23) – bezeichnet das Datum. Sie können anschließend eine Liste der Temperaturen (FOR I = 1 TO 31 – PRINT I; ". Dezember – "; DEZEMBER(I) – NEXT I) sowie nach dem obigen Muster die Durchschnittstemperatur für den gesamten Monat leicht ausdrucken lassen.
Ein Feld kann ohne weiteres aus Elementen bestehen, die von 56 bis 76 numeriert sind, oder von 1900 bis 1985. Auch negative Zahlen sind möglich: NUMERIC WERTE (–10 TO 10) oder NUMERIC WERTE(–20 TO –10). Die Zahlen sind nur eine Kennzeichnung für den einfachen Umgang mit den einzelnen Elementen.

ZWEIDIMENSIONALE FELDER

Die Felder in unseren Programmen oben waren „eindimensional" und glichen einer Liste:

1
2
3
4
5
6
7
8
9
10

Man kann aber auch zweidimensionale Felder einsetzen, die eher einer Tabelle gleichen:

1.1	1.2	1.3	1.4
2.1	2.2	2.3	2.4
3.1	3.2	3.3	3.4
4.1	4.2	4.3	4.4

Ein solches zweidimensionales Feld kann zum Beispiel in Brettspielen wie Dame oder Schach eingesetzt werden, wobei der Wert, der in jedes Element eingetragen wird, die Figur bezeichnet, die auf dem entsprechenden Feld des Spielbretts steht.
Zweidimensionale Felder werden ganz analog den eindimensionalen deklariert: NUMERIC A(5,5) oder NUMERIC WERTE(–10 TO 10,100 TO 200).

STRING ARRAYS ZEICHENKETTEN-FELDER

Felder von Zeichenketten („String arrays") werden genauso deklariert wie numerische Felder, allerdings verwenden wir das Wort STRING. Hier ist es – im Gegensatz zu numerischen Feldern – zusätzlich möglich, die maximale Länge der Zeichenketten im voraus zu bestimmen. Das hat einen wichtigen Grund.
Wie Sie wissen, hat der ENTERPRISE eine bestimmte Menge an Speicherplatz, der Programme und Daten enthalten kann. Wenn man ein Zeichenketten-Feld deklariert, hält der Computer genügend Speicherraum frei, um den Inhalt aller Elemente unterbringen zu können. Dabei geht er davon aus, daß maximal 132 Zeichen in einem Element

des Feldes abgespeichert werden. Natürlich brauchen Sie nicht immer diese Anzahl von Zeichen tatsächlich zu verwenden. Wenn Sie allerdings sicher sind, daß Sie immer mit weitaus weniger Zeichen pro Element auskommen werden, dann sollten Sie unbedingt die maximale Länge der Eintragungen im voraus begrenzen, so daß der Computer nicht unnötig viel Speicherplatz bereithält. Das geschieht so:

```
STRING FELD$(31 TO 50) * 10
```

Was soviel bedeutet wie: Das Zeichenketten-Feld FELD$ wird 20 Eintragungen haben können – FELD$(31) bis FELD$(50) –, wobei jedes Element im Höchstfall 10 Zeichen enthalten darf.

VARIABLEN DEKLARIEREN

Auch eine einfache Variable kann mit NUMERIC oder STRING deklariert werden. NUMERIC A reserviert Speicherplatz für eine numerische Variable namens A, und NUMERIC A,M,N,H,D tut das gleiche für die fünf Variablen auf einmal. STRING A$,B$,MONAT$*10 deklariert drei Zeichenketten-Variablen, wobei MONAT$ höchstens 10 Buchstaben enthalten darf. Es ist gute Programmier-Praxis, alle vorkommenden Variablen und Felder am Anfang des Programms zu deklarieren.

READ/DATA

Es gibt viele BASIC-Wörter und Anweisungen, die zusammengehören, so auch READ und DATA (und auch das Wort RESTORE). Wir behandeln sie an dieser Stelle des Handbuches, weil sie eine andere Möglichkeit bieten, größere Datenmengen in ein Programm einzubauen. Man kann sie auch dazu verwenden, Zahlen oder Zeichenketten in ein Feld einzutragen. Die eigentliche Arbeit leistet das Wort READ. Hier ein kurzes Beispiel:

```
100 DO
110         READ P
120         PRINT P
130 LOOP UNTIL P = 0
140 DATA 1,2,3,4,5,6,7,8,9,10
150 DATA 11,12,13,14,15,16,17,18
160 DATA 19,20,21,22,23,24,25,0
170 END
```

Die DATA-Zeilen 140 bis 160 enthalten einfach Daten. Jede Information (hier sind es Zahlen) wird von der nächsten durch ein Komma getrennt, damit der Computer weiß, wann eine Dateneinheit zu Ende ist. Die Anweisung READ veranlaßt den Computer, eine DATA-Eintragung zu „lesen" und unter der nachfolgenden Variablen P abzuspeichern. Danach kann man mit P beliebiges tun, z.B. auf den Bildschirm bringen, wie in unserem Programm.
Sie haben sicher bemerkt, daß wir einen einzigen Variablennamen benutzen, um alle 25 Zahlen auf den Bildschirm zu bringen. P ändert also seinen Wert 25mal – genauer gesagt 26mal, denn zum Schluß bekommt er entsprechend der letzten Eintragung in Zeile 160 den Wert 0. Diese Eintragung benutzen wir, um die DO/LOOP-Schleife zu beenden (UNTIL P = 0). Das ist aus folgendem Grund sehr wichtig.
Der Computer liest jede DATA-Eintragung nur einmal. Jedesmal, wenn eine Eintragung gelesen wurde, merkt sich der Computer, wie weit er gekommen ist. Bei der nächsten READ-Anweisung liest er dann die nächste DATA-Einheit. Das geschieht natürlich von links nach rechts und von oben nach unten. Wenn alle Daten gelesen wurden, darf keine READ-Anweisung mehr erfolgen, da das den Computer verwirren würde (er findet keine ungelesene Daten mehr). Also baut man sinnvollerweise eine Bremse ein, wie die Anweisung LOOP UNTIL P = 0 im obigen Programm. Nachdem die letzte Eintragung gelesen wurde, erfolgen keine READ-Anweisungen mehr.
READ/DATA kann man auch in Verbindung mit Zeichenketten verwenden, wie das nächste Beispiel demonstriert.

```
100 CLEAR SCREEN
110 PRINT "Ich will Ihnen eine Geschichte erzählen."
120 WAIT 5                    ! 5 Sekunden warten
130 PRINT "Fangen wir an."
140 PRINT:PRINT               ! Das sind Leerzeilen
150 !-----------------------------------------
160 WAIT 10                   ! 10 Sekunden warten
170 !-----------------------------------------
180 DO
190       READ A$
200       IF A$ = "ENDE" THEN EXIT DO
210       PRINT A$; " ";
220 WAIT 1                    ! 1 Sekunde warten
230 LOOP
```

```
240 PRINT:PRINT
250 PRINT"                    Das Ende"
260 END
270 !-------------------------------------------
280 DATA Es, war, einmal, ein, kleiner, Computer,
290 DATA namens, ENTERPRISE ., Er, war, ein,
    glücklicher,
300 DATA "Computer,", denn, viele, gute, Techniker,
310 DATA hatten, mit, viel, Liebe, daran, "gearbeitet,",
320 DATA ihn, zu, einem, sehr, guten, Computer,
    zu, machen.,
330 DATA Heute, lernen, Sie, mit, ihm, zusammen,
340 DATA BASIC., Macht, das, nicht, Spaß?, ENDE
```

Sicherlich haben Sie keine große Mühe, dieses Programm zu verstehen. Vielleicht können Sie erraten, warum einige Wörter in den DATA-Zeilen in Anführungszeichen stehen? Normalerweise ist das nicht nötig, denn die Anweisung READ A$ sagt dem Computer bereits, daß er Zeichenketten lesen soll. Aber wenn ein Komma als Bestandteil einer Zeichenkette vorkommt, muß diese in Anführungszeichen gesetzt werden, da der Computer sonst davon ausgehen müßte, daß das Komma das Ende der Zeichenkette bedeutet.
Der Computer liest also die Geschichte aus den DATA-Zeilen, Wort für Wort, und bringt sie mit kurzer Zeitverzögerung auf den Bildschirm. In Zeile 200 wird die DO/LOOP-Schleife unterbrochen, sobald der Computer das Wort „ENDE" liest.
Versuchen Sie folgendes Experiment: Setzen Sie in das Programm die folgenden Zeilen ein:

```
 90 DO
255 LOOP
```

Das sollte eigentlich bewirken, daß der Computer die Geschichte immer wieder erzählt. Aber wenn Sie das Programm laufen lassen, werden Sie merken, daß er sie nur einmal auf den Bildschirm bringt und anschließend mit einer Fehlermeldung aus dem Programm aussteigt. Der Grund hierfür ist, daß er beim zweiten Durchlauf keine ungelesenen Daten mehr findet. Aber das können wir leicht ändern. Fügen Sie eine weitere Zeile ein:

252 RESTORE

Schon läuft das Programm einwandfrei. RESTORE gibt alle Daten wieder frei, so daß der Computer die Geschichte von vorne aus den DATA-Zeilen ablesen kann. Sie könnten auch RESTORE 330 schreiben, was bewirkt, daß alle Daten ab Zeile 330 freigegeben werden. Beim zweiten Mal würde der Computer dann nur noch den letzten Satz auf den Bildschirm bringen. Auf diese Weise kann man ganz gezielt bestimmte Daten mehrmals verwenden.
Eine letzte Anmerkung: Die DATA-Zeilen dürfen an beliebiger Stelle im Programm stehen. Wenn sie ganz vorne sind, wird der Computer sie einfach ignorieren, bis eine READ-Anweisung kommt. Dann wird er sie aber finden, egal wo sie im Programm stehen!

Eine Funktion ist eine Art Programm innerhalb eines Programms, eine Folge von Anweisungen, die eine bestimmte Aufgabe erfüllen und immer wieder benötigt werden (man nennt sie auch „Unterprogramme" oder „Routinen").
Ein einfaches Beispiel wäre eine Warnung, die in einem Spiel den Benutzer zur Eile gemahnen soll.

```
1000 DEF WARNUNG
1010        CLEAR SCREEN
1020        PRINT AT 10,7: "Vorsicht, Sie werden
            gleich die Zeit überschreiten!"
1030        SOUND  !  Tonsignal
1040 END DEF
```

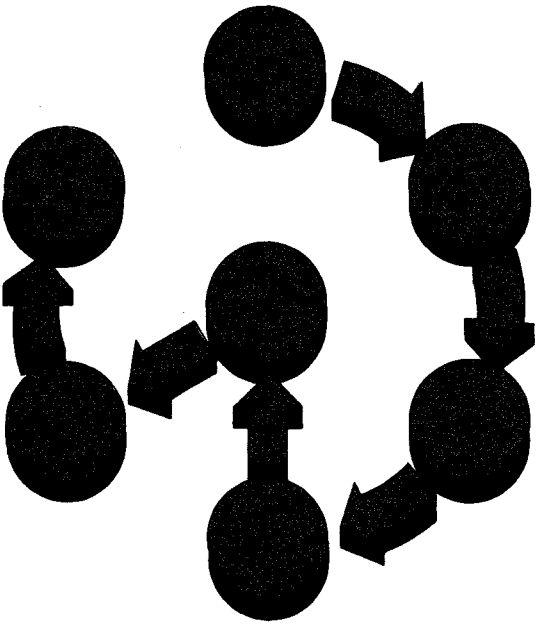
Eine solche Funktion muß immer einen Namen haben (hier WARNUNG), der hinter dem Wort DEF steht. Die eigentlichen Anweisungen (das, was die Funktion tun soll) können eine Zeile umfassen – oder auch 1000. Am Ende muß jedoch immer END DEF stehen.

AUFRUF VON FUNKTIONEN

Die Funktion arbeitet nicht von alleine. Wenn Sie das obige Programm eingeben und laufen lassen, passiert vorerst gar nichts. Wenn Sie aber anschließend – sogar im Direktmodus – CALL WARNUNG eingeben, sehen Sie, daß die Funktion aufgerufen und gleich ausgeführt wird. Das können Sie auch im Programm tun. Sie können beispielsweise Zeile 1050 CALL WARNUNG eingeben und bekommen sofort die Warnung auf dem Bildschirm, wenn Sie das Programm starten.
Die Definition der Funktion kann an beliebiger Stelle im Programm stehen, auch hinter der Zeile mit dem Aufruf – ja sogar hinter END. Machen Sie einen Versuch:

```
100 CALL WARNUNG
110 END
```

Die Funktion wird einwandfrei ausgeführt. Immer wenn der Computer eine CALL-Anweisung antrifft, sucht er das ganze Programm nach der entsprechenden DEFinition ab, führt die darin enthaltenen Anweisungen durch und kehrt zur ursprünglichen Stelle im Programm zurück. Das nachfolgende Diagramm zeigt, wie das prinzipiell funktioniert.



Denken Sie daran, daß eine Funktion so lange inaktiv bleibt, wie sie nicht explizit aufgerufen wird. Wenn der Computer ein Programm abarbeitet und auf eine Funktion trifft (DEF ....), springt er einfach zum Ende der Definition (END DEF) und arbeitet von dort an weiter. Die Funktion wird nicht ausgeführt, ohne daß sie irgendwo im Programm namentlich aufgerufen wird.

LOKALE UND GLOBALE VARIABLEN

Normalerweise kommen auch in einer Funktion Variable vor. Es gibt einige wichtige Regeln, die im Umgang mit diesen Variablen beachtet werden müssen. Geben Sie das folgende Programm ein:

```
100 DEF HOCHDREI
110         INPUT PROMPT "Zahl: ": Z
120         PRINT Z; " hoch drei ist "; Z*Z*Z
130 END DEF
140 CALL HOCHDREI
```

Wenn Sie das Programm laufen lassen, funktioniert alles wie erwartet: Der Computer springt sofort zu Zeile 140, ruft die Funktion HOCHDREI auf und arbeitet sie ab. Sie müssen eine Zahl eingeben und es erscheint die dritte

Potenz dieser Zahl auf dem Bildschirm. So weit, so gut. Nun geben Sie eine zusätzliche Zeile ein

```
150 PRINT Z
```

und lassen Sie das Programm nochmals laufen. Man würde erwarten, daß der Computer genauso wie zuvor reagiert, nur daß er zusätzlich noch die Zahl ausdruckt. Aber er bringt eine Fehlermeldung für die Zeile 150. Warum?

Die Variable Z, die in der Funktion vorkommt, ist lediglich eine sogenannte **lokale Variable**. Sie wird innerhalb der Funktion, die ja vom Computer als eigenständiges kleines Programm betrachtet wird, verwendet und dann einfach vergessen. Diese Variable bedeutet nichts für den Rest des Programms, so daß der Computer in Zeile 150 nicht weiß, was er auf den Bildschirm bringen soll.

Nun versuchen Sie folgendes: Numerieren Sie die Zeile 110 in 90 um, so daß sie außerhalb der Schleife steht. Jetzt fragt der Computer nach einer Zahl (Zeile 90), schreibt die dritte Potenz (120) und anschließend nochmal die Zahl selbst auf den Bildschirm. Die Zeile 150 bereitet ihm keine Schwierigkeiten mehr.

Der Grund hierfür ist, daß Sie durch die Verwendung von Z außerhalb der Funktion daraus eine globale Variable gemacht haben, die im ganzen Programm zur Verfügung steht.

Jetzt fügen Sie die folgende Zeile in das Programm:

```
125 LET Z = 20
```

Das Programm schreibt zum Schluß die Zahl 20. Die Funktion hat erkannt, daß Z eine globale Variable ist und hat sie daher an das Hauptprogramm zurückgereicht – mit dem neuen Wert 20.

Sie sollten immmer daran denken: wenn eine Funktion auf eine Variable trifft, die im Hauptprogramm vor dem Aufruf der Funktion nicht verwendet wurde, wird diese Variable automatisch als private, nur für die Funktion gültige (also lokale) Variable betrachtet und nicht an das Hauptprogramm weitergereicht. Wenn andererseits die Funktion auf eine bereits im Hauptprogramm verwendete Variable trifft, reicht sie diese (globale) Variable an das Hauptprogramm zurück, sobald ihre Arbeit verrichtet ist.

Wir haben bereits gesagt (S. 52), daß man Variable, die in

einem Programm vorkommen, bereits am Anfang mit NUMERIC oder STRING „deklarieren" sollte. In unserem modifizierten Beispiel oben (mit der INPUT-Zeile außerhalb der Funktion) erhielt Z noch vor dem Abruf der Funktion einen Wert und war damit ebenfalls eingeführt. Obwohl es für den Programmablauf nicht immer notwendig ist, alle Variablen am Programmanfang mit NUMERIC oder STRING (oder auch LET) formell zu deklarieren, verhindert das doch solche Probleme, wie sie in der ersten Programmversion auftraten (die Variable Z wurde erstmals innerhalb der Funktion eingeführt).
Innerhalb einer Funktion hat NUMERIC (oder STRING) immer die Auswirkung, eine lokale Variable einzuführen, auch wenn es bereits eine globale Variable mit demselben Namen gibt. Experimentieren Sie mit der folgenden abermals modifizierten Version des obigen Programms:

```
 90 INPUT PROMPT "Zahl: ": Z   !   Globale Variable
100 DEF HOCHDREI
110         NUMERIC Z   !   lokale Variable
115         LET Z = 100
120       PRINT Z;" hoch drei ist "; Z*Z*Z
130 END DEF
140 CALL HOCHDREI
150 PRINT Z   !   ursprüngliche globale Variable
```

Sie werden erkennen, daß die Funktion HOCHDREI hier mit zwei verschiedenen Z's arbeitet, einem lokalen, nur für die Funktion existierenden, und einem globalen aus dem Hauptprogramm. Wenn Sie die Zeile 110 löschen, gibt es dagegen nur eine (globale) Variable, denn die Anweisung LET führt keine neue (lokale) Variable ein, sondern verändert lediglich den Wert der ursprünglichen globalen Variablen, die in Zeile 90 bereits eingeführt wurde.
Das nächste Programm demonstriert, wie mehrere Funktionen in einem Programm verwendet werden. Es ist die überarbeitete Version eines Programms, das wir im Abschnitt über Entscheidungen behandelt haben. Das Beispiel soll nicht nur die Verwendung von Funktionen nochmals verdeutlichen, sondern Ihnen gleichzeitig vor Augen führen, daß es immer mehrere Wege gibt, ein Programm zusammenzustellen. Manche sind klar und einsichtig, andere erschreckend kompliziert und unverständlich. Sie werden diese Version des Programms wahrscheinlich „ordentlicher" finden als die ursprüngliche. Wir haben

übrigens auch noch eine Abfrage hinzugefügt, die den Computer veranlaßt, immer wieder von vorne anzufangen, wenn man das wünscht.

```
100 NUMERIC A
110 STRING A$, NAME$
120 ! ------ Deklaration der globalen Variablen
130 DO
140         CLEAR SCREEN
150         CALL FRAGE   !   Funktionsaufruf
160         CLEAR SCREEN
170         SELECT CASE A
180         CASE 1
190                 PRINT AT 9,18: "ENTERPRISE !!!"
200         CASE 2
210                 CALL NAME   !   Funktionsaufruf
220                 PRINT AT 9,18: NAME$
230         CASE 3
240                 CALL NAME   !   Funktionsaufruf
250                 PRINT AT 9,18: NAME$
260                 PRINT AT 11,18: "ENTERPRISE"
270         END SELECT
280         !-----------------------------------------------------
300         CALL WIEDERHOLUNG   !   Funktionsaufruf
310 LOOP WHILE A$ = "J"
320 END
330 !-----------------------------------------------------------------
340 DEF FRAGE   !   Funktion FRAGE
350         PRINT AT 9,8: "Was soll ich schreiben?"
360         PRINT AT 11,10: "1. Meinen Namen"
370         PRINT AT 12,10: "2. Ihren Namen"
380         PRINT AT 13,10: "3. Unserer beiden Namen?"
390         PRINT AT 16,10: "Bitte wählen Sie."
400         DO
410                 INPUT A
420          LOOP WHILE A < 1 OR A > 3 OR A <>
INT(A)
430 END DEF
440 !-----------------------------------------------------------------
450 DEF NAME   !   Funktion NAME
            INPUT AT 10,1, PROMPT "Dann sagen Sie mir
460         bitte, wie Sie heißen: ": NAME$
470         CLEAR SCREEN
480 END DEF
490 !-----------------------------------------------------------------
```

```
500 DEF WIEDERHOLUNG  !  Funktion WIEDERHO-
LUNG
510       DO
520             INPUT AT 15,10, PROMPT
                "Nochmals? (J/N) ":A$
530             LET A$ = UCASE$(A$(1:1))
540       LOOP UNTIL A$ = "J" OR A$ = "N"
550 END DEF
```

Wie Sie sehen, haben wir alle Variablen gleich zu Anfang des Programms deklariert, damit wir keine Probleme mit den Funktionen bekommen (wenn Sie die Zeilen 100 und 110 entfernen, läuft das Programm nicht mehr). Die Funktion FRAGE wird als erste aufgerufen und fragt den Benutzer, was er auf dem Bildschirm sehen will. Der Computer prüft die Antwort in der SELECT/CASE Schleife und handelt entsprechend. Ist es notwendig, den Namen des Benutzers zu wissen, wird die Funktion NAME aufgerufen. Zum Schluß wird noch die Funktion WIEDERHOLUNG aufgerufen.

In dieser Funktion fragt der Computer (Zeile 510), ob das ganze noch einmal wiederholt werden soll, und die Antwort des Benutzers (genauer gesagt: der erste Buchstabe von A$) wird in Zeile 520 in Großbuchstaben umgewandelt. Nur wenn das Ergebnis „J" oder „N" ist, wird die DO/LOOP-Schleife verlassen. Der Computer springt zum Hauptprogramm zurück (d.h. zur Zeile 300). Wenn die Antwort „J" war, wird das Programm wiederholt, bei „N" wird das Programm beendet. Eine solche Abfrage-Funktion ist sehr nützlich, denn sie kann überall im Programm eingesetzt werden.

Bislang haben wir Funktionen verwendet, die mit CALL aufgerufen wurden und eine Reihe von Aufgaben erfüllten. Sie haben Mitteilungen oder Warnungen auf den Bildschirm gebracht, Fragen an den Benutzer gestellt, und anderes mehr. Wir wollen jetzt eine andere Art von Funktionen betrachten, die eine Zahl verändern und sie an das Hauptprogramm zurückreichen.

Viele solche Funktionen sind bereits im Computer „eingebaut" und müssen deswegen gar nicht vom Benutzer definiert werden. Nehmen wir als Beispiel die Wurzel einer Zahl. Die Funktion SQR zieht völlig automatisch die Wurzel und braucht nicht einmal mit CALL aufgerufen zu werden: In den Anweisungen PRINT SQR(121) oder PRINT SQR(P*Q) oder LET M = 2 * SQR(N) + 1 wird jeweils die

Wurzel des Ausdrucks in Klammern errechnet. Ähnlich funktioniert die Funktion INT, die Sie bereits kennen. Solche BASIC-Ausdrücke geben Ihnen die Möglichkeit, häufig vorkommende Berechnungen schnell und unkompliziert durchzuführen.
Wenn Sie jedoch eine mathematische Funktion benötigen, die nicht im BASIC-Wortschatz vorhanden ist, müssen Sie diese selber definieren. Nehmen wir als Beispiel die Fakultät einer Zahl (Fakultät 4 bedeutet 1*2*3*4, Fakultät 6 ist 1*2*3*4*5*6, usw.). Diese Funktion existiert nicht in BASIC, aber Sie können sie leicht definieren:

```
100 DEF FAKULTAET
110         FOR Y = 1 TO F-1
120                 LET F = F * Y
130         NEXT Y
140 END DEF
```

Nun können Sie im Programm die Fakultät einer Zahl immer schnell ermitteln. Zum Beispiel:

```
200 LET F = 13
210 CALL FAKULTAET
220 PRINT F
```

Das ist längst nicht so bequem wie bei den eingebauten BASIC Funktionen. Lästig ist vor allem, daß man die Funktion jedesmal aufrufen muß und daß man die Zahl, von der man die Fakultät ausrechnen will, zuvor in F umbenannt haben muß. Aber der ENTERPRISE bietet Ihnen hier die Möglichkeit, das ganze wesentlich eleganter zu lösen. Löschen Sie das letzte Programm und geben Sie folgendes ein:

```
100 DEF FAK(X)
110         FOR Y = 1 TO X-1
120                 LET X = X*Y
130         NEXT Y
140         LET FAK = X
150 END DEF
```

Mit diesem kleinen Programmteil haben Sie gewissermaßen ein neues BASIC-Wort geschaffen, das Sie überall im Programm verwenden können, genauso wie INT oder SQR. Die Anweisung PRINT FAK(13) schreibt die Fakultät

von der Zahl 7 auf den Bildschirm, und LET N = FAK(7) * 1.5 + 3 ist ebenfalls zulässig.
Zwei Anmerkungen zu der obigen Funktion sind angebracht. In Zeile 100 bewirkt das in Klammern gesetzte X, daß die Funktion im Hauptprogramm hinter dem Wort FAK eine Zahl in Klammern erwartet. Diese Zahl wird automatisch unter der lokalen Variablen X abgelegt, die dann in der Funktion verarbeitet wird. In Zeile 140 wird das Ergebnis unter einer Variablen abgelegt, die denselben Namen hat wie die Funktion selbst. Damit kann die Funktion den Wert an die Stelle des Hauptprogramms zurückreichen, wo die ursprüngliche Funktion benutzt wurde. Insofern ist es auch nicht nötig, eine CALL-Anweisung zu verwenden.

DUMMY-VARIABLEN

Das in Klammern gesetzte X im obigen Programm nennt man in der Fachsprache einen formalen Parameter (oder auch „Dummy Variable"). Man sagt der Funktion damit, daß sie genau eine Zahl zur Weiterverarbeitung erwarten soll (im letzten Beispiel dieses Kapitels finden Sie eine Funktion mit zwei Dummy-Variablen). Diese Zahl kann auch in Form einer globalen Variablen an die Funktion gereicht werden. Geben Sie folgenden Zusatz zu dem obigen Programm ein:

```
160 LET A = 13
170 PRINT FAK(A)
```

Das funktioniert einwandfrei. Die Funktion schaut unter A nach, kopiert den Wert in die Dummy-Variable X und benutzt diese für die nachfolgenden Berechnungen. Der Wert von X ändert sich, aber A bleibt unverändert.
Was passiert aber, wenn wir die beiden Zeilen oben folgendermaßen verändern:

```
160 LET X = 13
170 PRINT FAK(X)
```

Versuchen Sie es! Sie werden feststellen, daß die Funktion sehr genau zwischen globalen Variablen aus dem Hauptprogramm und ihren eigenen Dummy-Variablen unterscheidet, auch wenn sie denselben Namen haben. Das Programm läuft also einwandfrei. Sie sollten mit solchen Funktionen weiterexperimentieren, bis Sie sicher sind, wie

man mit ihnen umgeht. Geben Sie beispielsweise eine zusätzliche Zeile in die Funktion ein:

```
145 LET X = 100
```

und eine zusätzliche Zeile in das Hauptprogramm:

```
180 PRINT X
```

Der Computer druckt 13 in Zeile 180, nicht 100, da die Zeile 145 nur das lokale X verändert.

VARIABLEN-PARAMETER

Sie haben gerade gesehen, wie eine Funktion mit Hilfe einer Dummy-Variablen eine Zahl verarbeitete, die sie aus einer globalen Variablen „kopiert" hatte. Aber obwohl die Funktion eine Zahl an das Hauptprogramm zurückreichte, blieb die ursprüngliche Variable unverändert.
Das ist nicht der Fall, wenn Sie REF vor die Dummy-Variable setzen – wie sich im folgenden Programm zeigen wird.
In diesem Programm gibt man zwei Zahlen – A und B – ein. Die Funktion errechnet A hoch B und B hoch A und reicht diese Werte als A und B zurück:

```
100 INPUT A
110 INPUT B
120 CALL POTENZEN(A,B)
130 PRINT A,B
140 END
150 !-----------------------------------------
160 DEF POTENZEN(REF X, REF Y)
170         LET Z = X
180         LET X = X^Y
190         LET Y = Y^Z
200 END DEF
```

In Zeile 160 werden zwei Dummy-Variable eingeführt. Wenn die Funktion aufgerufen wird, kopiert sie den Wert der ersten Variablen A (in Zeile 120) in die Variable X und den Wert von B in die Variable Y. Das funktioniert genauso wie in den früheren Beispielen, nur daß wir in diesem Fall gleich zwei Werte in die Funktion hineingeben. Da die Funktion außerdem zwei Werte an das

Hauptprogramm zurückreichen wird, muß sie mit CALL aufgerufen werden.
Der wesentlichste Unterschied zu den anderen Beispielen ist jedoch die Verwendung von REF. Das bewirkt, daß die Funktion am Ende den neuen Wert von X auf die globale Variable A überträgt (und den neuen Wert von Y auf B). Diese Eigenschaft kann in bestimmten Situationen sehr nützlich sein, wie Sie sich sicherlich vorstellen können. A und B nennt man übrigens **Variablen-Parameter.**

Der ENTERPRISE besitzt eine ungewöhnliche Fülle von graphischen Fähigkeiten, die man einsetzen kann, um eindrucksvolle Bilder und optische Effekte zu erzielen. Sie haben das wahrscheinlich schon beim Betrachten der Programme auf der Demonstrationscassette erkannt, und auch einige der bisher in diesem Handbuch vorgestellten Beispielprogramme haben von den Möglichkeiten der hochauflösenden Graphik Gebrauch gemacht.
Im ersten Teil des Handbuchs wurde die Anweisung PRINT AT erklärt. Dabei wurde der Bildschirm in 24 Zeilen mit je 40 Spalten aufgeteilt, so daß man genau angeben konnte, wo ein Text erscheinen sollte. PRINT AT 1,1 setzte den Text in die obere linke Ecke des Bildschirms.
Die Graphik-Kommandos benutzen ein ähnliches System, um Punkte und Linien auf den Bildschirm zu bringen. Hier ist ein kurzes Programm, das eine Linie zeichnet:

```
100 GRAPHICS
110 PLOT 640,360;1000,700
120 END
```

Die Anweisung in Zeile 100 ist Ihnen in einigen der früheren Beispiele bereits begegnet. Das Wort GRAPHICS ist eine einfache und schnelle Art, eine leere „Graphikseite" einzurichten, auf der man Bilder zeichnen kann.
Die Anweisung PLOT benutzt man, um Punkte oder Linien zu zeichnen. Die vier Zahlen, die hinter dieser Anweisung in Zeile 110 stehen, entsprechen zwei Stellen auf der Graphikseite. Das Programm zeichnet eine Linie von der Mitte des Bildschirms (Position 640,360) bis zu einer Stelle (1000,700) in Richtung der oberen rechten Ecke.
Es fällt auf, daß bei der Graphik die Koordinaten wesentlich größer sind als bei PRINT AT. Das bedeutet nicht, daß eine größere Fläche für das Zeichnen von Bildern zur Verfügung steht, sondern vielmehr, daß das Raster bei Graphik wesentlich feiner ist. Die Graphikseite ist in eine viel größe Anzahl von kleineren Positionen aufgeteilt, so daß Sie noch genauer positionieren können als bei einer Textseite. Mit anderen Worten: die Auflösung ist viel höher.
Ein weiterer Unterschied zum Textmodus besteht darin, daß auf der Graphikseite die Nullposition (0,0) in der unteren linken Ecke ist und die erste Zahl sich auf die horizontale Position bezieht. Das entspricht der x/y-Konvention allgemeiner graphischer Darstellungen.

Wenn Sie das vorige Programm laufen lassen, merken Sie, daß vier Zeilen der normalen Textseite ganz unten im Bildschirm stehenbleiben. Das gestattet Ihnen, weiter Texte und Anweisungen einzugeben, ohne daß die gezeichneten Linien verschwinden.
Diese Einteilung in eine Standard-Graphikseite mit vier Zeilen für Text wird immer eingerichtet, wenn Sie einfach das Kommando GRAPHICS eingeben. Sie bekommen dabei eine Graphikseite, die 1280 horizontale und 720 vertikale Positionen (also insgesamt 921600 Bildpunkte) umfaßt. Die Koordinaten der oberen rechten Ecke sind dementsprechend (1279,719).
Beide Bildschirmbereiche können mit den Anweisungen CLEAR GRAPHICS und CLEAR TEXT getrennt gelöscht werden. CLEAR SCREEN löscht beide Bereiche gleichzeitig. Die Anweisung DISPLAY TEXT schaltet auf die normale Textseite um, DISPLAY GRAPHICS auf die Graphikseite – ohne jeweils die Seiten zu löschen. Im Gegensatz dazu bewirken die einfachen Befehle TEXT oder GRAPHICS, daß die bestehende Text- bzw. Graphikseite zunächst gelöscht wird, bevor sie auf dem Bildschirm erscheint.
Im Nachschlageteil dieses Handbuchs können Sie übrigens erfahren, wie man anspruchsvollere Befehle benutzt, um die Größe von Graphikseiten festzulegen und sie beliebig auf dem Bildschirm zu plazieren.

PUNKTE ODER LINIEN

Geben Sie das folgende Programm ein:

```
100 GRAPHICS
110 PLOT 640,360
120 END
```

Wenn Sie das Programm starten, erscheint ein Punkt auf dem Bildschirm. Nun fügen Sie die folgende Zeile ein:

```
115 PLOT 1000,700
```

Es erscheinen zwei Punkte. Nun setzen Sie ein Semikolon hinter die Koordinaten in Zeile 110 (also PLOT 640,360;). Das Programm zeichnet diesmal eine Linie. Warum?
Die Antwort ist, daß ohne Semicolon der „Graphikstrahl" (engl.: „beam") zwischen den beiden PLOT-Anweisungen ausgeschaltet wird. Nur wenn er anbleibt, wird eine Linie zwischen den beiden Punkten gezeichnet. Um den Strahl

anzubehalten, muß man also ein Semikolon am Ende der ersten PLOT-Anweisung setzen.
Den Graphik-Strahl kann man als eine Art Schreibfeder betrachten. Eine PLOT-Anweisung setzt die Feder auf die Graphikseite und zeichnet zumindest einen Punkt. Wenn zwei Positionen durch ein Semikolon getrennt werden, so wird die Feder bei der Bewegung von der einen Position zur zweiten nicht abgehoben. Wenn man das Semikolon wegläßt, wird die Feder abgehoben und zur zweiten Position gebracht.
Wenn Sie übrigens hinter ein Koordinatenpaar ein Komma setzen, wird kein Punkt an der Stelle gezeichnet. Der Computer bewegt den Graphik-Strahl dorthin und schaltet ihn aus. Wie Sie sehen werden, ist dies für die Anweisung PLOT PAINT wichtig.
Sie können auch die Anweisungen SET BEAM ON und SET BEAM OFF (Graphikstrahl ein- bzw. ausschalten) verwenden, um die Feder auf der Seite zu lassen oder sie abzuheben.
Hier ist ein Windpocken-Programm:

```
100 RANDOMIZE
110 INPUT PROMPT "Wie viele Windpocken?": ANZAHL
120 GRAPHICS
130 LET Z = 0
140 DO
150         LET X = RND(1280)
160         LET Y = RND(720)
170         PLOT X,Y
180         LET Z = Z + 1
190 LOOP UNTIL Z = ANZAHL
200 END
```

Das Programm zeichnet Punkte auf den Bildschirm. Setzen Sie versuchsweise ein Semikolon hinter die Koordinaten in Zeile 170. Statt Punkte erscheinen nun Linien.
Ein Graphikprogramm kann natürlich auch Teil einer Funktion sein, wie Sie im folgenden Beispiel sehen können. Sie sehen daran außerdem, daß man mehrere Koordinaten hinter eine PLOT-Anweisung setzen darf.

```
100 DEF DIAGRAMM
110         GRAPHICS
120         PLOT 504,544;564,464;516,448;504,544;
            460,464;516,448
```

```
130 END DEF
140 CALL DIAGRAMM
```

DIE SCHILDKRÖTE

Wir wollen nun eine weitere Reihe von Anweisungen betrachten, mit denen Sie Linien zeichnen können. Man nennt sie „Schildkröten Kommandos", weil sie ursprünglich bei einigen Computersystemen der Steuerung eines langsamen Zeichenapparats dienten, der wie eine Schildkröte aussah. Der Vorteil von diesen Kommandos ist, daß man nicht jedesmal die Koordinaten ausrechnen muß.

```
100 OPTION ANGLE DEGREES
110 GRAPHICS
120 PLOT 300,150;
130 PLOT ANGLE 80;
140 PLOT FORWARD 500;
150 PLOT BACK 320;
160 PLOT RIGHT 35;
170 PLOT FORWARD 420;
180 PLOT BACK 285;
190 PLOT RIGHT 100;
200 PLOT FORWARD 340;
210 END
```

Sie erkennen, daß man in diesem Programm gewissermaßen eine Schildkröte über dem Bildschirm führt. FORWARD heißt „vor", BACK heißt „zurück", RIGHT heißt „rechts" und LEFT heißt „links".
Einige Zeilen bedürfen einer Erläuterung. OPTION ANGLE DEGREES sagt dem Computer, daß die nachfolgenden Angaben in Grad (ANGLE) und nicht in Rad angegeben werden. Der Rad (für Radiant) ist eine Einheit des in der höheren Mathematik und Physik verwendeten Winkelmaßes und entspricht ca. 57 Grad. Bei einigen mathematischen Operationen ist es günstiger, Rad zu verwenden.
Die GRAPHICS-Anweisung in Zeile 110 bringt den Graphik-Strahl in die Position 0,0 und schaltet ihn aus (CLEAR GRAPHICS würde das auch tun). Zeile 120 gibt die erste Position an, von der aus der Graphik-Strahl (die „Schildkröte") zeichnen soll.
Als nächstes müssen wir die gewünschte Richtung angeben. Nach PLOT ANGLE 0 würde die Schildkröte in waagerechter Richtung nach rechts schauen, nach PLOT

ANGLE 90 direkt nach oben. Die PLOT ANGLE-Anweisung verändert also die Richtung um den nachfolgenden Betrag, und zwar im Gegenuhrzeigersinn.
Der Anweisung PLOT FORWARD oder PLOT BACK folgt jeweils eine Angabe über die Anzahl von Graphik-Positionen, die sich die Schildkröte in die eingeschlagene Richtung vor oder zurück bewegen soll. PLOT RIGHT oder PLOT LEFT verändert die Richtung nach rechts oder nach links, und zwar um den Winkel, der diesen Anweisungen folgt.
Achten Sie darauf, daß Sie auch bei den Schildkröten-Kommandos Semikolons verwenden müssen, um den Grafik-Strahl anzulassen. Probieren Sie, was passiert, wenn Sie einige der Semikolons im letzten Programm entfernen oder sie durch Kommas ersetzen.

ELLIPSEN UND KREISE

Das folgende Programm zeichnet eine Ellipse:

```
100 GRAPHICS
110 PLOT 640,250,
120 PLOT ELLIPSE 100,200,
130 END
```

Zeile 110 bestimmt den Mittelpunkt der Ellipse. Die erste Zahl hinter der Anweisung PLOT ELLIPSE gibt die horizontale Entfernung (in Graphik-Positionen) zwischen dem Mittelpunkt und dem Umfang der Ellipse an, die zweite Zahl die vertikale Entfernung. Wenn beide Zahlen gleich sind, wird ein Kreis gezeichnet. Der Grund, weshalb wir Kommas am Ende der Zeilen 110 und 120 gesetzt haben, ist, daß anderenfalls im Mittelpunkt der Ellipse tatsächlich ein Punkt stehen würde. So bleibt der Graphik-Strahl in Position 640,250, aber in ausgeschaltetem Zustand.

FARBEN

Sie wissen sicherlich schon, daß der ENTERPRISE nicht weniger als 256 Farben auf den Bildschirm bringen kann. Bislang hatten Sie wenig Gelegenheit, von dieser Fähigkeit Gebrauch zu machen. Das wollen wir hier nachholen. Das folgende Programm zeigt Ihnen alle 256 Farben auf einmal.

```
100 GRAPHICS 256
110 LET Z = 0
120 FOR Y = 0 TO 560 STEP 80
130         FOR X = 32 TO 1024 STEP 32
```

```
140              SET INK Z
150              PLOT X, Y; X, Y + 70
160              LET Z = Z + 1
170         NEXT X
180 NEXT Y
190 END
```

Wie Sie erkennen können, hat der ENTERPRISE für jede Farbe eine Codezahl zwischen 0 und 255. Es gibt auch andere Möglichkeiten, Farben auszuwählen, wie wir in diesem Kapitel noch erläutern werden.

DIE FARBMODI

Im obigen Programm mußten Sie hinter die GRAPHICS-Anweisung die Zahl 256 setzen, um dem Computer mitzuteilen, daß er 256 Farben bereitstellen soll. Mit anderen Worten: Sie haben den entsprechenden **Farbmodus** mit 256 Farben eingestellt.
Sicherlich haben Sie gemerkt, daß der Computer die Linien ungewöhnlich dick zeichnete, jedenfalls dicker als in den vorherigen Graphikprogrammen. Je mehr Farben er bereithält, desto gröber wird die Graphik. Das ist notwendig, um mit dem Speicherplatz auszukommen.
Wir haben bereits gesehen, daß die „Graphikseite" eine höhere Auflösung hat als eine „Textseite". Aber wir müssen weiter differenzieren. Ganz allgemein kann man sagen, daß man umso weniger Punkte auf einer horizontalen Zeile unterbringen kann, je mehr Farben der Farbmodus bietet.
Es gibt in der hochauflösenden Graphik („high-resolution" = HIRES) vier verschiedene Farbmodi.
GRAPHICS HIRES 2 – Nach diesem Kommando können nur zwei Farben auf einmal verwendet werden. Dafür passen 640 Punkte in eine Zeile.
GRAPHICS HIRES 4 – Bei vier zur Verfügung stehenden Farben passen 320 Punkte in eine Zeile.
GRAPHICS HIRES 16 – Hier sind 160 Punkte in einer Zeile möglich.
GRAPHICS HIRES 256 – Bei 256 möglichen Farben passen nur 80 Punkte in eine Zeile.
Wenn Sie einfach nur das Kommando GRAPHICS verwenden, wird der Computer denselben Modus wählen, der zuletzt eingestellt war. Wenn Sie den Computer ausschalten oder Reset drücken und dann GRAPHICS eingeben, wird GRAPHICS HIRES 4 eingestellt.
Die Anzahl der Punkte in senkrechter Richtung wird vom

Farbmodus nicht beeinflußt. In der Standard-Graphikseite (mit vier Textzeilen unten) gibt es – unabhängig vom eingestellten Farbmodus – immer 190 Zeilen.
Trotz der unterschiedlichen Auflösungen verwenden alle Farbmodi dasselbe Koordinatensystem. PLOT 0,0;640, 360 wird immer eine Linie von der linken unteren Ecke bis zur Mitte der Graphikseite zeichnen, obwohl die Feinheit der Zeichnung je nach Modus unterschiedlich sein wird.

LORES-GRAPHIK

Neben der hochauflösenden Graphik, die relativ speicherintensiv ist, gibt es auch Graphikmodi, die nur die halbe horizontale Auflösung besitzen und mit LORES aufgerufen werden. Diese Anweisung funktioniert genauso wie HIRES: z.B. richtet GRAPHICS LORES 16 eine 16-Farben-Graphikseite mit halber Auflösung ein.

FARBEN MISCHEN

Kehren wir zum 256-Farben-Modus zurück. Man kann in diesem Modus Figuren in beliebigen Farben zeichnen, indem man den jeweiligen PLOT-Anweisungen das Kommando SET INK zusammen mit der Codezahl für die Farbe voranstellt. Sie können ebenfalls mit SET PAPER (+ Farbcode) die Hintergrundfarbe wählen, die dann nach CLEAR GRAPHICS verwendet wird.
Man muß ein wenig mit den Farben experimentieren, bis man ein Gefühl dafür gewinnt, welche Farben zu welchen Zahlen gehören. 18 ist ein helles Grün, 91 ist ein leuchtendes Gelb, usw. Wenn Sie aber mit diesem System nicht zurechtkommen, gibt es eine andere Methode: Man kann die Farben selber mischen!
Jede Farbe kann durch eine Mischung von Rot, Grün und Blau hergestellt werden. Weiß erzeugt man mit allen drei Farben, Gelb durch die Mischung von Rot und Grün. Das könnte Sie ein wenig verwundern, aber vergessen Sie nicht, daß Ihr Fernseher oder Farbmonitor genauso funktioniert! Wir mischen hier nämlich Licht und nicht Malfarben. Komplexe Farben werden dadurch erzeugt, daß man unterschiedliche Mengen von jeder der drei Farben zusammenmischt. Schwarz bedeutet überhaupt keine Farbe.
Beim ENTERPRISE können Sie genau auf diese Weise arbeiten. Sie können jede Farbe als Mischung der drei Grundfarben definieren. Die Anweisung hierzu lautet RGB (für Rot, Grün, Blau) gefolgt von drei Zahlen in Klammern. Diese Zahlen, die mit Kommas voneinander getrennt werden und zwischen 0 und 10 liegen müssen,

geben den Anteil von Rot, Grün und Blau (in dieser Reihenfolge) an. SET INK RGB (1,.5,.5) erzeugt Rosa, RGB (.4,.4,0) ein mattes Gelb und RGB (.6,.6,.4) eine Schattierung von Grau.
Die acht Standard-Farben des Teletext-Systems können auf noch einfachere Weise eingestellt werden. Man braucht hinter SET INK oder SET PAPER nur ihren (englischen) Namen zu setzen (z.B. SET INK GREEN). In der nachfolgenden Tabelle finden Sie die Farben, ihre englischen Namen, sowie die RGB-Mischung, die ihnen entsprechen.

Schwarz	BLACK	RGB(0,0,0)
Rot	RED	RGB(1,0,0)
Grün	GREEN	RGB(0,1,0)
Gelb	YELLOW	RGB(1,1,0)
Blau	BLUE	RGB(0,0,1)
Violett	MAGENTA	RGB(1,0,1)
Türkis	CYAN	RGB(0,1,1)
Weiß	WHITE	RGB(1,1,1)

DIE FARB-PALETTE

Mit der Anweisung GRAPHICS HIRES 16 wählt man, wie wir gesehen haben, den 16-Farben-Modus, d.h. der Computer kann bis zu sechzehn verschiedene Farben gleichzeitig auf den Bildschirm bringen. Welche Farben er verwenden soll, kann man ziemlich frei bestimmen. Man legt dazu am Anfang eine „Farb-Palette" an, d.h. ein Liste von Farben oder Farbmischungen, die genauso wie bei einer Maler-Palette vorbereitet werden und anschließend zur Verfügung stehen.
Zunächst schreiben Sie SET PALETTE und dahinter eine Liste von acht Farben, die Sie frei aus den 256 möglichen wählen können. Sie können sie entweder mit ihren Standard-Codezahlen oder als „Mischung" oder, falls sie in der obigen Liste sind, mit ihren englischen Namen bezeichnen. Zum Beispiel könnten Sie eingeben:

```
100 SET PALETTE 67, 31, WHITE, 4, RGB(0,.3,.8),
RGB(.7,.7,.1), 187, 190
```

Diese acht Farben ihrer „Palette" können Sie später einfach mit den Nummern 0 bis 7 aufrufen (z.B. SET INK 5 stellt die RGB Farbmischung (.7,.7,.1) ein).
Bei den verbleibenden acht Farben sind Sie in Ihrer Auswahl eher eingeschränkt. Die Farben 8 bis 15 in Ihrer

Palette müssen alle aus einer einzigen Gruppe (siehe Nachschlageteil) stammen. Dafür kann man die Gruppe ganz einfach mit der Anweisung SET BIAS, gefolgt von der Codenummer einer beliebigen Farbe aus dieser Gruppe, einstellen. Wenn Sie beispielsweise SET BIAS 67 eingeben, dann haben Sie die Gruppe 64-71 gewählt. Die Farbe mit der Standard-Codenummer 64 wird fortan mit 8 aufgerufen, 65 mit 9, 66 mit 10, usw.
Sie können nun jede der 16 Farben aus Ihrer Palette für die nächste Linie oder Figur, die der Computer zeichnen soll, sehr einfach bestimmen. Vergessen Sie nicht, daß abgesehen vom 256 Farben-Modus Anweisungen wie SET INK 6 oder SET PAPER 6 sich auf die Farbe 6 in der Palette beziehen, nicht auf die Farbe 6 in der Standard-Codetabelle. Denken Sie auch daran, daß die Farbnummern immer bei 0 beginnen, nicht bei 1.
Wenn man eine Palette für den 4-Farben-Modus bestimmt, gibt man nur vier Farben an, die dann mit 0 – 3 ausgewählt werden können. Im 2-Farben-Modus bestimmt man mit SET PALETTE nur die Farben für 0 und 1.
Man kann in einer Palette die Farben jederzeit durch neue ersetzen. Auch einzelne Farben lassen sich ändern. Wenn Sie beispielsweise Farbe 3 der obigen Palette in Gelb umändern wollen, können Sie das mit der folgenden Anweisung tun:

```
100 SET COLOUR 3,YELLOW
```

Sie können selbstverständlich alle Farbmodi benutzen, ohne sich die Mühe zu machen, eine Farb-Palette zu bestimmen. In diesem Fall wird der Computer bestimmte vorprogrammierte Farben für Sie bereitstellen.

VERWENDUNG DER FARBPALETTE

Eine sehr wichtige Eigenschaft der Farb-Palette besteht darin, daß sie nicht nur die danach gezeichneten Linien und Figuren beeinflußt, sondern auch diejenigen, die bereits auf dem Bildschirm sind. Die Fähigkeit, alle Farben eines Bildes auf einen Schlag auszuwechseln, kann man für dramatische und schnelle Graphik-Effekte ausnutzen.
In dem folgenden Programm, das Ellipsen von zufälliger Größe und Farbe zeichnet, beginnen wir damit, alle Farben der Palette gleich zu machen: Eine Linie, die in Farbe Nummer 2 gezeichnet wird, sieht genauso aus, wie eine in Farbe Nummer 3. Auch die Hintergrundfarbe wird gleich

sein, so daß die Zeichnungen effektiv unsichtbar sind. Dann, wenn eine Taste gedrückt wird, hört der Computer auf zu zeichnen und zeigt Ihnen das fertige Bild. Das geschieht durch Wechseln der Farb-Palette, so daß das Bild schlagartig sichtbar wird. Danach wechselt der Computer die Farben der Ellipsen ständig, bis Sie das Programm mit der 'STOP'-Taste anhalten (den letzten Teil des Programms ist eine endlose Schleife).

```
100 RANDOMIZE
110 GRAPHICS HIRES 4
120 SET PALETTE BLUE,BLUE,BLUE,BLUE
130 !-----------------------------------------------------
140 DO
150         SET INK RND(3)-1
160         PLOT 625,330,
170         PLOT ELLIPSE RND(500),RND(300),
180 LOOP WHILE INKEY$ = ""
190 !-----------------------------------------------------
200 DO
210         SET PALETTE BLUE,BLUE,RED,GREEN
220         WAIT 1
230         SET PALETTE BLUE,RED,GREEN,BLUE
240         WAIT 1
250         SET PALETTE BLUE,GREEN,BLUE,RED
260         WAIT 1
270 LOOP
```

Übrigens können die Anweisungen PALETTE, INK und PAPER auch für Textseiten verwendet werden. Das Kapitel „Video-Optionen" im Nachschlageteil gibt hierzu ausführliche Informationen. Fürs erste versuchen Sie folgende Anweisungen:

```
SET #102: COLOUR 1,MAGENTA
SET #102: INK 3
```

PLOT PAINT

Diese Anweisung füllt eine abgeschlossene Figur mit einer Farbe aus, die Sie mit INK bestimmen können. Hier als Beispiel ein Programm, das einen Kreis innerhalb eines anderen zeichnet und dann beide mit verschiedenen Farben füllt.

```
100 GRAPHICS HIRES 4
110 SET PALETTE WHITE,YELLOW,BLUE
120 PLOT 400,400,
130 PLOT ELLIPSE 200,200,
140 PLOT ELLIPSE 80,80,
150 SET INK 3
160 PLOT PAINT
170 PLOT 400,250,
180 SET INK 2
190 PLOT PAINT
200 END
```

PLOT PAINT füllt jeweils die Fläche aus, die den Graphik-Strahl enthält und von einer Linie umgeben ist, die andere Farben hat als die augenblickliche Farbe des Strahls selbst. Ist die Fläche nicht geschlossen, wird die Farbe „auslaufen" und wenn möglich den ganzen Bildschirm ausfüllen. Die Kommas am Ende der Zeilen 120, 130, 140 und 170 sind notwendig, da sonst ein Punkt an der Position des Strahls gezeichnet wird und PLOT PAINT nur diesen Punkt ausfüllen würde.

LINE STYLE UND LINE MODE

LINE STYLE gestattet Ihnen, verschiedenartige Linien zu zeichnen. Zum Beispiel bewirkt SET LINE STYLE 10, daß alle nachfolgenden Linien (bis LINE STYLE wieder verändert wird) aus langen, eng beieinander liegenden Strichen gezeichnet werden. SET LINE STYLE 9 zeichnet Linien, die abwechselnd aus Strichen und Punkten bestehen. Es gibt 14 Arten von Linien (1–14), mit denen Sie experimentieren können.

Mit SET LINE MODE können Sie bestimmen, wie die auf dem Bildschirm gezeichneten Linien und Flächen miteinander „reagieren". LINE MODE 0, der voreingestellte Modus, bewirkt, daß neue Linien schon vorhandene Figuren „überschreiben". In den anderen Modi (1–7) werden die Farben auf der Graphikseite auf vielfältige Weise miteinander gemischt, wie Sie mit einigen Versuchen leicht feststellen können.

SEITEN UND KANÄLE

Bislang haben wir die Anweisungen GRAPHICS oder TEXT verwendet, um eine leere „Seite" einzurichten, auf der wir dann zeichnen oder schreiben können. Sie werden diese Methode wahrscheinlich in den meisten Fällen als völlig ausreichend empfinden, aber wenn Sie anfangen,

sehr anspruchsvolle Programme zu schreiben, werden Sie vielleicht andere Text- und Graphikseiten verwenden wollen. Das können Sie, indem Sie „Channel"-Nummern zwischen 1 und 100 spezifizieren.
„Channels" sind logische Kanäle, die in einem späteren Kapitel ausführlicher erörtert werden. Das nachfolgende sollten Sie in Verbindung mit diesem Kapitel sowie den Abschnitten über Video-Optionen und OPEN im Nachschlageteil lesen.
Es gibt zwei wichtige Vorteile, wenn Sie für Ihre Seiten neue Channels eröffnen:
(1) Verschiedene Text- oder Graphikseiten, die aufwendige Texte oder Zeichnungen enthalten, können gleichzeitig im Speicher enthalten sein und dann mit einem einfachen Kommando auf den Bildschirm gebracht werden.
(2) Sie können die Größe einer Seite selber bestimmen. Eine Graphikseite kann dann beispielsweise den ganzen Bildschirm umfassen – mit Ausnahme der Statuszeile an der oberen Kante –, oder Sie können eine sehr kleine Seite einrichten, um besondere Effekte zu erzielen oder Speicherplatz zu sparen.
Das nachfolgende Beispiel schafft ein kleines Textfenster mitten auf dem Bildschirm.

```
 90 SET BORDER CYAN
100 SET VIDEO MODE 0
110 SET VIDEO COLOUR 0
120 SET VIDEO X 20
130 SET VIDEO Y 10
140 OPEN #1: "VIDEO:"
150 DISPLAY #1: AT 7 FROM 1 TO 10
160 PRINT #1: "Textfenster"
170 END
```

In Zeile 140 wird Channel-Nummer 1 für die neue Video-Seite bestimmt. Bevor das aber geschieht, müssen die Video- und Farbmodi sowie die Größe der Seite spezifiziert werden: VIDEO MODE 0 ist eine Textseite mit 40 Spalten (MODE 1 ist eine Graphikseite, MODE 2 eine Textseite mit 80 Spalten). Mit VIDEO COLOUR wird der Farbmodus eingestellt, und zwar nach der folgenden Konvention:

VIDEO COLOUR 0 – 2-Farben-Modus
VIDEO COLOUR 1 – 4-Farben-Modus

VIDEO COLOUR 2 – 16-Farben-Modus
VIDEO COLOUR 3 – 256-Farben-Modus
Die Textseite sollte immer im 2-Farben-Modus 0 eingestellt sein, wobei Sie immer noch die Farben selber bestimmen können. Beachten Sie bitte, daß die PRINT-Anweisung in Zeile 160 die Channel-Nummer enthalten muß.
Die Zeilen 120 und 130 geben die Breite bzw. Höhe des Textfensters in Spalten bzw. Zeilen an. Zeile 150 besagt, daß der obere Teil der Textseite (genauer: die ersten 10 Zeilen) auf den Bildschirm gebracht werden sollen, und zwar ab Zeile 7 des Bildschirms. Da die Seite nur 10 Zeilen hoch ist, wird sie also komplett abgebildet.
Die Höhe einer Text- oder Graphikseite kann auch größer als die Höhe des Bildschirms sein (allerdings höchstens 255 Zeilen lang). Das nächste Programm definiert eine Graphikseite, die 30 Zeilen hoch und 8 Spalten breit ist. Dann zeichnet der Computer eine Ellipse auf diese Graphikseite und bringt zunächst den unteren und dann den oberen Teil auf den Bildschirm.

```
100 SET VIDEO MODE 1          ! Graphikseite
110 SET VIDEO COLOUR 2        ! 16-Farben-Modus
120 SET VIDEO X 8             ! 8 Spalten breit
130 SET VIDEO Y 30            ! 30 Zeilen hoch
140 OPEN #1: "VIDEO:"         ! Channel Nr.1 f. Videoseite
150 PLOT #1: 128,540,         ! Mittelpunkt der Ellipse
160 PLOT #1:                  ! Ellipse zeichnen
    ELLIPSE 115,500,
170 DISPLAY #1: AT 10         ! Oberen Teil anzeigen
    FROM 21 TO 30
180 WAIT 10
190 DISPLAY #1: AT 10         ! Unteren Teil anzeigen
    FROM 1 TO 10
200 END
```

Das nächste Beispiel richtet 3 Seiten ein und bringt sie an verschiedenen Stellen auf den Bildschirm.

```
100 SET VIDEO MODE 0          ! Textseite
110 SET VIDEO COLOUR 0        ! 2 Farben
120 SET VIDEO X 42
130 SET VIDEO Y 8
140 OPEN #1: "VIDEO:"
150 !----------------------------------------------------
```

```
150 SET VIDEO MODE 1       ! Graphikseite
160 SET VIDEO COLOUR 3     ! 256 Farben
170 OPEN #2: "VIDEO:"
180 SET VIDEO COLOUR 1     ! 4-Farben-Graphikseite
190 OPEN #3: "VIDEO:"
200 ! Fenster einrichten
210 DISPLAY #1: AT 9 FROM 1 TO 8
220 DISPLAY #2: AT 1 FROM 1 TO 8
230 DISPLAY #3: AT 17 FROM 1 TO 8
240 !-----------------------------------------------------
250 PRINT #1: "Textfenster"
260 SET #2: BEAM ON
270 PLOT #2: 100,100;
280 SET #3: BEAM ON
290 PLOT #3: 100,100;
300 END
```

Nachdem das Programm gelaufen ist, werden Sie nicht mehr sehen können, was Sie eintippen. Geben Sie TEXT ein oder drücken Sie die Funktionstaste 5, um zum Normalzustand zurückzukehren.
Achten Sie darauf, daß jede Seite ihre eigene „Farb-Palette" hat. Allerdings würde eine SET BIAS-Anweisung auf alle Seiten gleichermaßen wirken.

SET BORDER

Man kann beim ENTERPRISE auch noch eine Umrandung um die gesamte Anzeige auf dem Bildschirm setzen und dadurch eine Art „Tafel" schaffen, auf der verschiedene Text- und Graphikseiten eingeblendet werden können. Die SET BORDER-Anweisung ist unabhängig von allen eingestellten Farb-Paletten – die sich nur auf die jeweilige Seite beziehen –, so daß die Farbe der Umrandung als zusätzlicher Farbcode angegeben werden muß. Zum Beispiel erzeugen SET BORDER 255, SET BORDER WHITE und SET BORDER RGB(1,1,1) alle eine weiße Umrandung. Wenn die Umrandungsfarbe mit SET BORDER bestimmt wird und Channel #101 (für die Standard-Graphikseite) nicht offen ist, muß eine geeignete Channel-Nummer (normalerweise #102 für die Standard-Textseite) vorher eingegeben werden, z.B.: SET #102: BORDER 116.

Der ENTERPRISE hat einen vordefinierten Satz von Zeichen, die er auf den Bildschirm bringen kann. Diese sind nach dem System des American Standard Code for Information Interchange (ASCII) definiert.
Zu jedem Zeichen im Zeichensatz gehört eine Codenummer (ASCII Code genannt) zwischen 0 und 127. Man kann ein Zeichen direkt ansprechen, indem man die entsprechende Taste drückt oder es mit PRINT auf den Bildschirm bringt, oder man kann die zugehörige Codenummer verwenden. Probieren Sie PRINT "N" und dann PRINT CHR$(78).
Die Codes werden intern vom Computer verwendet, um die Zeichen selbst anzusprechen. Wie das genau geschieht, brauchen Sie nicht zu wissen, bis Sie ein sehr fortgeschrittener Programmierer geworden sind. Aber eine einfache Beschreibung ist angebracht. Wenn Sie eine Taste -- sagen wir „a" -- drücken, erhält der Computer ein Signal von der Tastatur, daß diese Taste gedrückt wurde. Er übersetzt dieses Signal in den entsprechenden Zeichencode und sendet diesen zu dem Teil der Maschine, der für die Anzeige auf dem Bildschirm verantwortlich ist. Dieser übersetzt den Code in ein Videosignal, das alle Informationen zur Darstellung des Zeichens enthält. Ein kleines „a" erscheint auf dem Bildschirm.
Für Sie sieht das so aus, als würde das Zeichen gleichzeitig mit dem Tastendruck auf dem Bildschirm erscheinen. Man bekommt eine Vorstellung davon, wie schnell ein Computer arbeitet, besonders wenn man bedenkt, daß die oben beschriebenen Arbeitsschritte in Wirklichkeit wesentlich komplizierter und in eine Reihe von kleineren Einzelschritten aufgeteilt sind.
Hier ist ein Programm, das den ganzen Zeichensatz des ENTERPRISE auf den Bildschirm bringt – mit Ausnahme der ersten 32, die sogenannte „Control"-Zeichen sind.

```
100 FOR N = 32 TO 159
110 PRINT CHR$(N),
120 NEXT N
130 END
```

Das erste Zeichen – CHR$(32) – ist ein Leerzeichen und bleibt für Sie unsichtbar auf dem Bildschirm.
Das nächste Programm übersetzt Zeichen, die Sie eingeben, in die entsprechenden Zeichencodes.

```
100 DO
110         INPUT PROMPT "Geben Sie ein Zeichen ein:
            ": Z$
120         IF LEN(Z$) > 1 THEN PRINT "Nur ein
            Zeichen auf einmal!"
130         LOOP WHILE LEN(Z$) > 1
140 PRINT
150 PRINT "Der Zeichencode für "; Z$
160 PRINT " ist "; ORD(Z$)
```

Wie Sie sehen, sind ORD und CHR$ gegensätzliche Funktionen. ORD ermittelt den ASCII-Code eines Zeichens, während CHR$ für eine ASCII-Nummer das entsprechende Zeichen erzeugt.

DIE DEFINITION VON EIGENEN ZEICHEN

Für jede ASCII-Nummer hat der Computer eine Beschreibung der Form gespeichert. Diese kann man umdefinieren, so daß andere als die Standard-Zeichen erzeugt werden können.
Jedes Zeichen wird aus einer Matrix von 8 x 9 Punkten zusammengesetzt. Um ein Zeichen auf den Bildschirm zu bringen, werden einige dieser 72 Punkte „angeschaltet". Die Punkte sind so klein und so eng beieinander, daß sie zu der gewünschten Form zusammenschmelzen. Wenn Sie ein neues Zeichen entwerfen wollen, ist es nützlich, eine 8x9 Matrix auf kariertem Papier zu umranden und einzelne Karos auszufüllen.

```
   X  X  X  X  X
X                 X
X     X     X     X
X                 X
   X        X
         X  X
         X
```

Danach können Sie das Zeichen leicht programmieren:

```
100 NUMERIC N(1 TO 9)  ! Feld deklarieren
110 !-----------------------------------------------------------
120 DATA 00111110
130 DATA 01000001
140 DATA 01010101
150 DATA 01000001
160 DATA 00100010
```

```
170 DATA 00010100
180 DATA 00001000
190 DATA 00000000
200 DATA 00000000
210 !-----------------------------------------------------------
220 FOR REIHE = 1 TO 9
230         READ N(REIHE)
240 NEXT REIHE
250 !-----------------------------------------------------------
260 SET CHARACTER 63,BIN(N(1)),BIN(N(2)),
BIN(N(3)),BIN(N(4)),BIN(N(5)),BIN(N(6)), BIN(N(7)),
BIN(N(8)),BIN(N(9))
270 !-----------------------------------------------------------
280 PRINT "?"
290 PRINT CHR$(63)
300 END
```

Zeile 260 ist die entscheidende, die alle notwendigen Informationen für das neue Zeichen in den Speicher bringt. Diese wurden in der FOR/NEXT-Schleife gelesen und unter N(1) bis N(9) abgespeichert. Das Wort BIN sagt dem Computer, daß es sich bei diesem Zifferblock um Binärzahlen handelt. Die Nullen stellen ausgeschaltete Bildpunkte dar, die Einsen angeschaltete. Die gesamte Information wird unter dem Zeichencode 63 abgelegt. Damit haben Sie das Fragezeichen umdefiniert! Sie sehen, was der Computer in Zeile 280 und Zeile 290 produziert hat. Versuchen Sie es auch im Direktmodus!

Nachdem ein solches Programm zum Umdefinieren von Zeichen einmal gelaufen ist, bleiben die neuen Zeichen solange erhalten, bis man den Computer ausschaltet oder die „Reset"-Taste drückt.

Sie werden bald erkennen, daß das Umdefinieren von Zeichen viele interessante Möglichkeiten bietet. Man kann das gesamte Alphabet umgestalten, ausländische Schriften programmieren, gotische oder kursive Zeichen entwerfen, und vieles andere mehr. Für Weltraumspiele kann man Raumschiffe und Figuren aus Gruppen von selbstdefinierten Zeichen zusammensetzen. Auf der Demonstrations-Cassette gibt es Programme, die Ihnen einen Eindruck von den Anwendungsmöglichkeiten des selbstdefinierten Zeichensatzes vermitteln.

Töne aus dem Computer sind eine wertvolle Bereicherung für viele Programme. Bei „ernsthaften" Anwendungen können sie als Signale verwendet werden, um die Aufmerksamkeit des Benutzers zu erregen. Aber erst bei Spielprogrammen kommen gute Klangeffekte voll zur Geltung.
Der ENTERPRISE ist ein wahrer „Tonkünstler". Er besitzt einen eingebauten Lautsprecher zur Tonerzeugung, aber darüber hinaus bietet er Ihnen die Möglichkeit, Klangeffekte in Stereo zu programmieren und diese dann über Kopfhörer oder über eine Hifi-Anlage zu genießen. (Wenn dabei der eingebaute Mono-Lautsprecher stören sollte, können Sie ihn mit dem Befehl SET SPEAKER OFF ausschalten – oder aber mit der Taste SHIFT + FUNCTION 7, was dasselbe bewirkt.)
Die Töne, die der Computer erzeugt, werden im wesentlichen durch zwei BASIC-Kommandos gesteuert: SOUND und ENVELOPE. In der SOUND-Anweisung bekommt der Computer alle notwendigen Informationen über Tondauer, Tonhöhe und Lautstärke. Eine ENVELOPE-Anweisung enthält Informationen über Variationen in Tonhöhe und Lautstärke, die ein einzelner Ton während des Abspielens erfahren soll.

DER SOUND-BEFEHL

Hier ist ein Beispiel für einen SOUND-Befehl:

```
100 SOUND PITCH 40, LEFT 127, RIGHT 191,
    DURATION 200, ENVELOPE 10
```

Wir wollen alle Einzelangaben, die hinter SOUND stehen, nacheinander betrachten.
Die Zahl hinter PITCH (= Tonhöhe) gibt an, wie hoch der Ton erklingen soll, jedenfalls am Anfang (die Tonhöhe kann während des Abspielens durch ENVELOPE verändert werden). Hinter PITCH kann eine beliebige Zahl zwischen 0 und 127 stehen. Den tiefsten Ton (0) kann man kaum noch hören, und die musikalisch ansprechenden Töne liegen etwa zwischen 1 und 83. Jede Erhöhung um eine Zahl entspricht etwa einem Halbtonschritt. Der Wert 37 entspricht dem „mittleren C" und wird abgespielt, wenn bei einem SOUND-Befehl keine anderslautende PITCH-Angabe gemacht wurde.
LEFT und RIGHT bestimmen die Lautstärke auf dem linken und dem rechten Stereo-Kanal. 255 ist die höchste Lautstärke und 0 die niedrigste (unhörbar). Wenn keine Anga-

ben über die Lautstärke erfolgen, wählt der Computer die Voreinstellung 255. Im obigen Beispiel erklingt der Ton mit halber Lautstärke links (127) und mit dreiviertel Lautstärke rechts (191). Das sind Maximalangaben, denn auch die Lautstärke von einzelnen Tönen kann durch ENVELOPE beeinflußt werden.
Die Zahl, die hinter DURATION steht, bestimmt die Tondauer, und zwar in „Ticks". Ein Tick ist eine fünfzigstel Sekunde, so daß DURATION 200 eine Tondauer von vier Sekunden zur Folge hat. Wenn keine Angaben über die Tondauer gemacht werden, stellt der Computer 50 Ticks (= 1 Sekunde) ein.

ENVELOPE bezieht sich auf eine bestimmte „Hüllkurve", die an anderer Stelle im Programm definiert und mit einer Kennzahl versehen ist (siehe unten). Gültige Kennzahlen sind 0 bis 254 (die Zahl 255 ist für die Voreinstellung des Computers reserviert).
Die Angaben hinter der SOUND-Anweisung können in beliebiger Reihenfolge aufgeführt sein. Da der Computer für jede Angabe bestimmte vorprogrammierte Werte besitzt (PITCH 37, LEFT 255, RIGHT 255, DURATION 50, ENVELOPE 255), kann man auch mit der Anweisung SOUND allein einen Ton erzeugen. Es gibt auch weitere Angaben, die hinter SOUND stehen können, aber dazu werden wir später kommen.

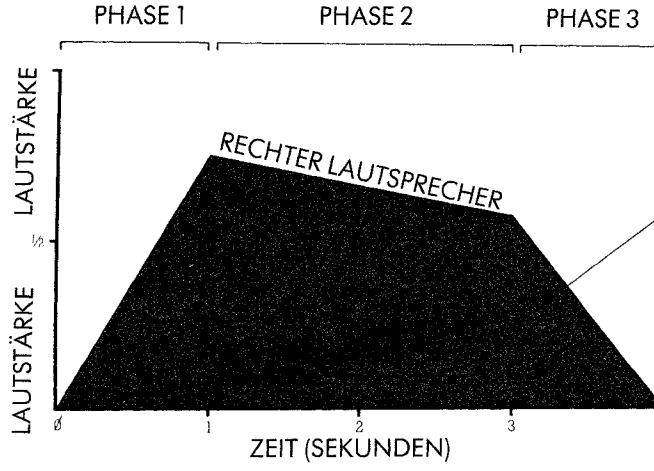
ENVELOPE

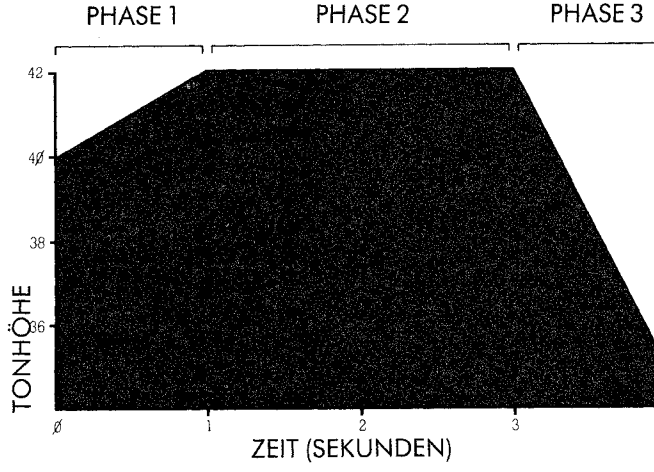
Für das obige Programm wollen wir nun eine Hüllkurve mit der Kennzahl 10 bestimmen.

```
90 ENVELOPE NUMBER 10; 2, 8, 63, 50; 0, 24, -16,
   100; -5, 47, -39, 50
```

Auch hier folgen dem Schlüsselbegriff eine Reihe von Informationen.
NUMBER 10 bestimmt die Kennzahl der Hüllkurve, mit der sie in SOUND-Anweisungen aufgerufen wird. Hinter der NUMBER-Angabe muß ein Semikolon stehen. Danach folgen drei Gruppen von je vier Zahlen, wobei jede Gruppe von der nächsten ebenfalls durch ein Semikolon getrennt wird.
Fangen wir mit der ersten Gruppe an. Die vier Zahlen dieser Gruppe beschreiben die Veränderungen in Tonhöhe und Lautstärke, die der Ton in der ersten Phase seines Erklingens durchlaufen soll.

Die erste Zahl der Gruppe gibt an, daß der Ton um 2 Halbtöne ansteigen soll, und zwar ausgehend von dem Wert, der in der SOUND-Anweisung angegeben wird. Wenn wir statt der 2 eine 0 eingetragen hätten, wäre die Tonhöhe in der ersten Phase gleichgeblieben, –1.5 hätte sie um einen Dreiviertelton sinken lassen.
Die nächste Zahl (8) beschreibt eine Anderung in der Lautstärke des linken Tonkanals. Man kann eine Zahl zwischen –63 und +63 angeben (größere oder kleinere Zahlen werden auf diese Maximalwerte gebracht). 64 ist die Maximallautstärke, die im SOUND-Befehl für den linken Lautsprecher angegeben wird, –63 macht den Ton unhörbar.
Ganz zu Anfang, wenn ein Ton abgespielt wird, hat er keine Lautstärke. In unserem Beispiel wird der Ton in der ersten Phase von Null bis zu einem Achtel (d.h. acht Vierundsechzigstel) der Maximallautstärke anschwellen. Da der SOUND-Befehl in Zeile 100 für links die halbe Lautstärke (127) als Maximum festlegt, wird der Ton ein Sechzehntel der überhaupt möglichen Lautstärke erreichen.
Die dritte Zahl (63) hat dieselbe Funktion für den rechten Kanal. Hier wird der Ton etwa dreiviertel der im SOUND-Befehl festgelegten Maximallautstärke erreichen. SOUND und ENVELOPE arbeiten zusammen, um für jeden Augenblick zwei Werte für die Lautstärke zu errechnen – einen für den linken Kanal und einen für den rechten. Wenn Sie weder Stereo-Kopfhörer noch Hifi-Anlage verwenden, dann gilt die Summe beider Werte für den Mono-Lautsprecher.
Die letzte Zahl in der Gruppe gibt an, wie lange die erste Phase dauern soll. Auch hier wird die Tondauer in „Ticks" angegeben, so daß die Zahl 50 die erste Phase auf eine Sekunde begrenzt.
Die zweite und dritte Gruppe von je vier Zahlen bestimmen die Hüllkurven für die zweite und dritte Phase — genauso wie wir das für die erste Phase beschrieben haben.
Wenn Sie nun die beiden Programmzeilen eingeben und das Programm laufen lassen, hören Sie einen Ton, der in den beiden nachfolgenden Diagrammen grafisch abgebildet ist.





Man kann natürlich auch wesentlich komplexere Hüllkurven bestimmen – man kann bis zu 255 Angaben machen, wie wir später sehen werden. Denken Sie daran, daß die Angaben in einer Hüllkurve relativ sind, d.h. das Ergebnis hängt von der im SOUND-Befehl bestimmten Lautstärke bzw. Tonhöhe ab. Das hat den Vorteil, daß man mit einem ENVELOPE eine ganze Serie von Tönen beeinflussen kann.

Die Hüllkurve muß übrigens immer in einer Zeile definiert werden, die vor dem SOUND-Befehl liegt, in dem sie aufgerufen wird.

DER TON-STAPEL

Die Tonerzeugung wird von einem separaten Baustein besorgt und verzögert nicht die anderen Aktivitäten des Computers. Im nachfolgenden Programm läuft der Ton im Hintergrund ab, während der Computer mit Schreiben und Zeichnen beschäftigt ist.

```
100 ENVELOPE NUMBER 20; -2, 63, 63, 100; -3, 0, 0, 100;
    3, -36, -36, 100; 2, -12, -12, 100; 0, 0, 0, 100
110 SOUND PITCH 61, LEFT 255, RIGHT 0, DURATION
    500, ENVELOPE 20
120 !----------
130 CLEAR SCREEN
140 PRINT AT 10,5: "Ich werde gleich eine Kurve
    zeichnen,"
150 PRINT AT 11,5: "die die Hüllkurve dieser Töne dar-
    stellt."
160 PRINT AT 12,5: "Die Lautstärke wird durch die blaue"
170 PRINT AT 13,5: "Fläche dargestellt, die Tonhöhe"
180 PRINT AT 14,5: "durch die gelbe Linie."
190 !----------
200 SOUND PITCH 43, LEFT 127, RIGHT 127, DURATION
    200, ENVELOPE 20
210 ! -- Dieser Ton hört nach den ersten beiden
220 ! -- Phasen der Hüllkurve auf.
230 !----------
240 WAIT 4
250 SOUND PITCH 25, LEFT 0, RIGHT 255, DURATION
    500, ENVELOPE 20
260 GRAPHICS
270 PLOT 0, 0; 250, 540; 500, 540; 750, 270; 1000, 180;
    1250, 180
280 PLOT 100, 20,
290 PLOT PAINT
300 SET INK 3
310 PLOT 0, 450; 250, 350; 500, 100; 750, 350; 1000, 450;
    1250, 450
320 END
```

In Zeile 110 empfängt der Computer den SOUND-Befehl und beginnt, einen Ton zu erzeugen. Aber er wartet nicht, bis der Ton zu Ende ist, sondern macht gleich mit den PRINT-Anweisungen weiter. Der Ton läuft im Hintergrund ab. In Zeile 200 kommt die zweite SOUND-Anweisung, aber der erste Ton ist noch nicht beendet. Der zweite Ton kommt auf einen Stapel und der Computer arbeitet weiter.

In Zeile 250 kommt der dritte Ton, der ebenfalls auf dem Stapel abgelegt wird. Der Computer spielt den zweiten Ton erst dann ab, wenn der erste zu Ende ist, und den dritten, wenn der zweite abgespielt wurde. Währenddessen zeichnet er die Kurven.

RELEASE

Der ENTERPRISE gestattet Ihnen, eine oder mehrere Phasen am Ende einer Hüllkurve auf besondere Weise zu beeinflussen. Diese Phasen werden mit dem Wort RELEASE gekennzeichnet. Die RELEASE-Phase ist eine Periode, in der der Ton abklingt – in der er eigentlich zu Ende ist, aber eine gewisse Nachwirkung haben soll.
Hier ist ein Beispiel:

```
100 ENVELOPE NUMBER 4; .1, 63, 40, 5; -.3, -32, -20, 20;
    2, 0, 0, 25; RELEASE 0, -16, -10, 10; 0, -15, -10, 15;
```

Die drei Phasen vor RELEASE beanspruchen insgesamt eine Sekunde, die beiden RELEASE-Phasen zusammen eine halbe Sekunde. Der Unterschied zwischen den normalen und den RELEASE-Phasen ist folgender: Wenn die im SOUND-Befehl spezifizierte Tondauer kürzer ist als die Summe der normalen Phasen, dann werden diese abgeschnitten; aber die RELEASE-Phasen werden auch dann ausgeführt, wenn die Tondauer abgelaufen ist – sofern keine weiteren Töne auf dem Stapel sind.
Wenn also in einem SOUND-Befehl, der die obige Hüllkurve verwendet, DURATION 25 angegeben ist, wird die dritte Phase nicht ausgeführt. Der Computer springt direkt von der zweiten Phase in die RELEASE-Phasen – oder zum nächsten Ton, falls einer auf dem Stapel ist.
Bei DURATION 60 hingegen, werden alle normalen Phasen ausgeführt und anschließend noch die erste RELEASE-Phase, bevor der nächste Ton gespielt wird. Ist DURATION 100 angegeben, dann werden sämtliche Phasen ausgeführt und es gibt eine Pause von einer halben Sekunde, bevor der nächste Ton erklingt.

INTERRUPT

Sie haben bereits die Grundlagen der Klangerzeugung beim ENTERPRISE kennengelernt, aber es gibt eine Reihe von weiteren interessanten Möglichkeiten, mit denen Sie experimentieren können. Man kann zum Beispiel einen Ton einen anderen „unterbrechen" lassen. Versuchen Sie folgendes Programm:

```
100 SOUND PITCH 61, LEFT 122, RIGHT 122, DURATION
    5000
110 ! Ein Ton erklingt für 100 Sekunden
120 PRINT "Unterbrechen mit s"
130 DO
140 LOOP UNTIL INKEY$ = "s"
150 SOUND PITCH 37, LEFT 255, RIGHT 255, DURATION
    25, INTERRUPT
160 END
```

Das Wort INTERRUPT in Zeile 150 bedeutet, daß sobald diese Zeile erreicht wird, der Computer den augenblicklichen Ton abbricht und den neuen erklingen läßt. Auch werden alle Töne, die eventuell noch auf dem Stapel sind, ignoriert.

MEHRSTIMMIGKEIT

Es ist nicht nur möglich, sondern auch sehr reizvoll, mehrere Töne auf einmal abzuspielen. Der ENTERPRISE hat vier Tongeneratoren, von denen jeder einen eigenen Tonstapel verwalten kann. Damit kann man mehrstimmige Melodien programmieren.
Die Tongeneratoren sind von 0 bis 3 numeriert (Nummer 3 ist ein Geräusch-Generator, der alle Angaben über Tonhöhe ignoriert). Bislang haben wir nur den Tongenerator 0 verwendet, der voreingestellt ist. Wenn Sie einen anderen Tongenerator ansprechen wollen, benützen Sie die Anweisung SOURCE in Ihrem SOUND-Befehl. SOURCE 2 bedeutet, daß der Ton von Tongenerator 2 erzeugt bzw. auf dem Stapel dieses Tongenerators abgelegt werden soll. Wenn man mehrere Töne gleichzeitig spielen will, müssen diese natürlich sorgfältig synchronisiert werden. Mit dem Wort SYNC in einem SOUND-Befehl können Sie erreichen, daß Töne absolut gleichzeitig abgespielt werden. SYNC 2 besagt z.B., daß, wenn der Ton abspielbereit ist (d.h. im Stapel ganz oben ist), der Computer ihn zurückhalten soll, bis zwei weitere Töne auf den Stapeln anderer Tongeneratoren erscheinen. Sobald drei Töne vorhanden sind, werden sie gleichzeitig abgespielt.

Versuchen Sie das folgende Programm:

```
100 SOUND PITCH 37, LEFT 255, RIGHT 0,
    DURATION 100, SOURCE 0, SYNC 2
110 WAIT 1
120 SOUND PITCH 41, LEFT 0, RIGHT 255,
    DURATION 100, SOURCE 1, SYNC 2
130 WAIT 1
140 SOUND PITCH 44, LEFT 127, RIGHT 127,
    DURATION 100, SOURCE 2, SYNC 2
```

Wie Sie sehen, werden alle drei Töne gleichzeitig abgespielt, obwohl der Computer wegen der Warteschleifen mit einiger Verzögerung auf die verschiedenen SOUND-Befehle trifft. Probieren Sie, was passiert, wenn man die SYNC-Anweisungen aus den Zeilen 100, 120 und 140 entfernt!

Die Anweisung CLEAR QUEUE, gefolgt von der Nummer eines Tongenerators, unterbricht den augenblicklichen Ton dieses Generators und löscht alle Töne vom Stapel. Ein INTERRUPT in einem SOUND-Befehl unterbricht ebenfalls den Ton, ohne die Tätigkeit der anderen Tongeneratoren zu beeinflussen.

KOMPLEXERE KLÄNGE

Normalerweise nimmt der Computer an, daß Sie höchstens 20 „Phasen" in einer Hüllkurve angeben werden. Wenn Sie noch komplexere Töne erzeugen wollen, müssen Sie zusätzlichen Speicherplatz zu diesem Zweck freihalten. Das tun Sie mit der Anweisung SET SOUND BUFFER, gefolgt von einer Zahl zwischen 1 und 255 (wobei allerdings die Anzahl der möglichen Daten durch die maximale BASIC-Zeilenlänge von 250 Zeichen begrenzt wird). Zuvor müssen Sie allerdings den „Kanal" für die Tonerzeugung (103) schließen und zum Schluß wieder öffnen. Darüber finden Sie mehr im Nachschlageteil unter dem Stichwort OPEN.

Der Zusatz STYLE (gefolgt von einer Zahl zwischen 0 und 255) beeinflußt ebenfalls den Ton in einem SOUND-Befehl. Sie sollten damit experimentieren, besonders in Verbindung mit dem Geräusch-Generator (SOURCE 3), mit dem sich die interessantesten Effekte erzielen lassen. Auch hierüber können Sie im Nachschlageteil dieses Handbuchs (im Kapitel über Ton-Optionen) mehr erfahren.

Wir haben eine Menge Zeit damit verbracht, das Programmieren in einzelne Abschnitte zu unterteilen und diese nacheinander zu betrachten. Noch haben wir uns aber nicht mit grundsätzlichen Techniken der Programmierung beschäftigt –– wie eine Aufgabe computergerecht zu bewältigen ist, wie man Programme schreibt, die nicht nur eine Aufgabe optimal lösen, sondern auch noch klar und verständlich sind. Man will ja auch später einmal ein solches Programm sofort verstehen und es womöglich für andere Zwecke modifizieren.
Wir wollen in diesem Kapitel ein einfaches Problem betrachten und demonstrieren, wie man das Programm dazu entwirft. Wenn Sie einmal die Programmplanung im Prinzip verstanden haben, werden auch komplexere Probleme keine Schwierigkeiten mehr bereiten.

DAS PROBLEM

Wenn ein Händler einen Rabatt von soundsoviel Prozent gewährt, muß man ausrechnen, wie hoch der zu zahlende Preis tatsächlich ist. Das kann der Computer schnell und elegant erledigen. Er braucht nur zwei Zahlen zu kennen: den Originalpreis und den Nachlaß in Prozent. Daraus kann er errechnen, wieviel weniger Sie (der Kunde) zu zahlen haben und wie hoch der neue Preis ist.
Als erstes müssen Sie beim Programmieren festlegen, was das Programm genau tun soll. In unserem Fall sind es die folgenden Schritte:

1. den Originalpreis und den Nachlaß erfragen;
2. den Nachlaß in DM ausrechnen;
3. diesen Betrag vom Originalpreis abziehen;
4. den abgezogenen Betrag und den neuen Preis anzeigen;
5. fragen, ob Sie eine weitere Berechnung durchführen wollen.

Das ist recht einfach. Nun müssen Sie aber festlegen, wie das Programm diese Arbeit verrichten soll. Es gibt immer mehrere Wege, um ein Programm zusammenzustellen, aber es gibt einige allgemeine Prinzipien, die man dabei beachten sollte.

MODULARE PROGRAMME

Ein ordentliches, gut geschriebenes Programm sollte **modular** sein – d.h. es soll aus klar strukturierten, ineinandergreifenden Teilen bestehen. Es ist meist verhängnisvoll, wenn man drauflosprogrammiert und eine Menge BASIC-Zeilen produziert, die man später nicht mehr entziffern kann.

Ihr Hauptziel beim Programmieren sollte es sein, ein lauffähiges Programm zu erstellen, das man später, auch nach Jahren, auf Anhieb wieder versteht. Sie müssen immer in der Lage sein, eventuell auftretende Fehler schnell zu finden und zu korrigieren, sowie das Programm ohne Probleme zu erweitern oder zu modifizieren, wenn es eine andere Aufgabe erfüllen soll.
Ein modulares Programm sollte aus den folgenden Teilen zusammengesetzt sein:
1. der Deklaration aller globalen Variablen;
2. dem Hauptprogramm;
3. einer END-Anweisung, die das Programm beendet;
4. Funktionen, die das Hauptprogramm unterstützen;
5. DATA-Zeilen, wenn diese notwendig sind.

Eine solche Programmstruktur macht es leicht, auch komplizierte Aufgaben schnell und effektiv zu programmieren. Vor allem ist es sehr angenehm, wenn man später sofort weiß, wo man bestimmte Programmteile (z.B. die Funktionen) in einem beliebigen Programm finden kann.

DIE PROGRAMMTEILE

Globale Variable (d.h. Variable, die im Hauptteil des Programms verwendet werden) sollten immer am Anfang stehen. Das verhindert, daß der Computer auf eine unbekannte Variable trifft und das Programm mit einer Fehlermeldung abbricht. Wenn Sie beim Programmieren entdecken, daß Sie eine weitere Variable benötigen, sollten Sie diese immer mit den anderen am Anfang des Programms deklarieren. Damit haben Sie immer eine Liste der verwendeten Variablen und können leicht feststellen, welche Rolle sie im Programm spielen.
Bei lokalen Variablen, die nur in einer Funktion auftreten, sollten diese am Anfang der Funktion deklariert werden.
Als nächstes sollte das Hauptprogramm folgen. Das hat mehrere Gründe. Zum einen läßt sich ein Programm leichter schreiben, wenn der Teil, der alle anderen Teile steuert, am Anfang steht. Andererseits kann man später immer schon durch Lesen des ersten Programmteils erkennen, was das Programm insgesamt macht. Und schließlich werden die meisten globalen Variablen im Hauptteil verwendet, der also sinnvollerweise gleich hinter den Zeilen stehen sollte, in denen diese Variablen eingeführt wurden.
Das Hauptprogramm ruft Unterprogramme (meist sind es Funktionen) auf, die als nächstes kommen sollten. Jede Funktion wird als getrenntes kleines Programm behandelt, das wiederum ein anderes Unterprogramm aufrufen

kann. Man sollte dabei eine Reihenfolge einhalten, die die Hierarchie der Unterprogramme widerspiegelt.
DATA-Zeilen brauchen nicht unbedingt am Ende des Programms zu stehen, aber sie sollten alle an einer klar erkennbaren Stelle im Programm untergebracht sein. Es kann sehr lästig sein, wenn man sie suchen muß. Wenn sie alle zusammenstehen, kann man sie rasch finden, überprüfen, die einzelnen Daten zählen, usw. DATA-Zeilen können auch in einer Funktion stehen, wenn sie nur innerhalb der Funktion gelten.
Die END-Anweisung sollte immer am Ende des Hauptprogramms stehen und nicht unbedingt in der letzten Zeile des gesamten Progamms. Funktionen und Unterprogramme können ruhig dahinter untergebracht werden, denn der Computer „springt" aus dem Hauptprogramm, um sie zu finden.
Schließlich sollten Sie immer viele Kommentarzeilen im Programm verwenden und diese auch sehr auffällig gestalten. Es ist durchaus angemessen, wenn ein Programm mehr Kommentarzeilen enthält als BASIC-Anweisungen. Die Kommentare sollten die Funktion der einzelnen Programmteile in klaren Worten beschreiben, so daß man auch nach Jahren sofort weiß, wie das Programm aufgebaut ist.
Wir wollen nicht den Eindruck erwecken, daß Sie hier zur zwanghaften Ordnung gedrängt werden. Aber genauso wie beim Schreiben eine gewisse Systematik von großem Nutzen sein kann, ist es längerfristig sehr hilfreich, wenn man die oben geschilderten Prinzipien beherzigt und immer eine klare Programmstruktur anstrebt.

DAS BEISPIEL

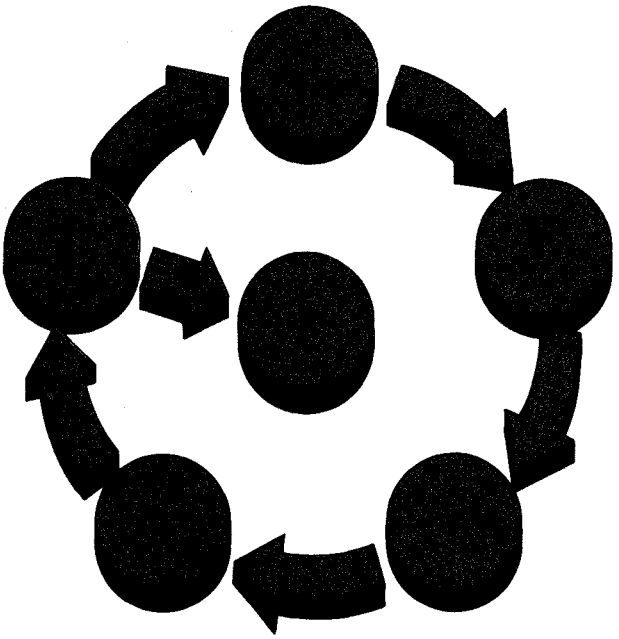
Wir wollen nun zu unserem Beispiel mit der Rabatt-Berechnung zurückkehren.
Als erstes wollen wir festlegen, was das Hauptprogramm tun soll. Es wird zwei Zahlen – den Originalpreis und den Rabattsatz – vom Benutzer verlangen, diese von einem Unterprogramm (einer Funktion) verarbeiten lassen und das Ergebnis bekanntgeben. Zum Schluß soll der Computer dann fragen, ob der Benutzer weitere Preisnachlässe berechnen will.
Als nächsten wollen wir feststellen, wie viele Funktionen wir brauchen. In diesem Fall reicht eine, um den Preisnachlaß und den Nettopreis auszurechnen. Wir wollen diese Funktion RABATTBERECHNUNG nennen.

Jetzt wollen wir überlegen, wie der Computer die Berechnungen durchführen kann.
Der Computer wird zuerst den Originalpreis bekommen und dann den Rabattsatz in Prozent. Der Preisnachlaß in DM ist der Originalpreis durch 100mal den Rabattsatz, also:
Preisnachlaß = (Preis/100) * Rabatt
Um den Nettopreis zu bekommen, müssen wir dann diesen Preisnachlaß vom Originalpreis abziehen, also:
Nettopreis = Originalpreis – Preisnachlaß
Es dürfte Ihnen jetzt klar sein, welche Variablen wir brauchen:
1. Originalpreis in DM – PREIS
2. Rabatt in Prozent – RABATT
3. Preisnachlaß in DM – NACHLASS
4. Nettopreis in DM – NETTO

Die ersten beiden Variablen werden in einer INPUT-Anweisung stehen, brauchen daher nicht deklariert zu werden. Wir wissen also, daß wir am Anfang nur zwei globale Variablen gesondert einführen müssen. Aus Gründen der besseren Übersicht wollen wir aber alle vier Variablen am Anfang deklarieren.
Die Funktion können wir gleich aufschreiben. Sie wird die beiden INPUT-Angaben (PREIS und RABATT) verwenden, um die beiden anderen Zahlen (NACHLASS und NETTO) auszurechnen:

```
DEF RABATTBERECHNUNG
      LET NACHLASS = (PREIS/100) * RABATT
      LET NETTO = PREIS – NACHLASS
END DEF
```

Das nachfolgende Diagramm gibt den Programmablauf schematisch wieder. Wenn Sie bei der INPUT-Anweisung beginnen, können Sie genau verfolgen, welche Schritte der Computer durchläuft.



Wir wissen nun, welche Programmteile in welcher Reihenfolge stehen werden und was wir mit dem Programm genau erreichen wollen. Bei einem komplizierteren Programm sind die Vorbereitungen naturgemäß aufwendiger und Sie werden in einem solchen Fall wahrscheinlich verschiedene Programmteile zunächst auf Papier entwerfen wollen.
Aber jetzt wollen wir das Hauptprogramm schreiben.

```
100 NUMERIC PREIS, RABATT, NACHLASS, NETTO
110 !-----------------------------------------------------------
120 DO
130         CLEAR SCREEN
140         INPUT PROMPT "Originalpreis in DM:
            ":PREIS
150         INPUT PROMPT "Rabatt in Prozent: ":RABATT
160         !-------------------------------------------------
170         CALL RABATTBERECHNUNG
180         !-------------------------------------------------
190         CLEAR SCREEN
200         PRINT "Originalpreis = "; PREIS
210         PRINT "Rabatt = "; RABATT
220         PRINT "Preisnachlaß = "; NACHLASS
```

```
230         PRINT "Nettopreis = "; NETTO
240         !-----------------------------------------------------
250         PRINT
260         INPUT PROMPT "Möchten Sie einen weiteren
            Preisnachlaß berechnen? ": ANTW$
270 LOOP WHILE UCASE$(ANTW$) = "J"
280 END
```

Das ist das Hauptprogramm. Es fehlt noch die Funktion RABATTBERECHNUNG.

```
300 DEF RABATTBERECHNUNG
310         LET NACHLASS = (PREIS/100) * RABATT
320         LET NETTO = PREIS – NACHLASS
330 END DEF
```

Wenn Sie das Programm starten, werden Sie feststellen, daß es einwandfrei funktioniert. Aber man sollte ein Programm nie in diesem Zustand belassen. Es kann ein wenig hübscher gestaltet werden und mit Kommentarzeilen versehen werden. Wir fügen die neuen Zeilen einfach dazwischen, dann kann man zum Schluß mit RENUMBER die Zeilennumerierung schnell in Ordnung bringen.

```
40 ! ----------------------------------------------------------
50 ! Programm zur Berechnung von Preisnachlässen
60 ! ----------------------------------------------------------
70 !
80 ! Hauptprogramm beginnt
90 ! ----------------------------------------------------------
100 NUMERIC PREIS, RABATT, NACHLASS, NETTO
120 DO
130         CLEAR SCREEN
132         PRINT AT 4,10: "Rabattberechnungen"
134         PRINT AT 5,10: "******************"
136         PRINT: PRINT
140         INPUT PROMPT "Originalpreis in DM  : ": PREIS
150         INPUT PROMPT "Rabatt in Prozent    : ": RABATT
160         !-----------------------------------------------------
170         CALL RABATTBERECHNUNG
180         !-----------------------------------------------------
190         CLEAR SCREEN
192         PRINT AT 4,7: "Preisnachlaß, Nettopreis"
194         PRINT AT 5,7: "**********************"
196         PRINT: PRINT
```

```
200         PRINT "Originalpreis =DM "; PREIS
210         PRINT "Rabatt =              "; RABATT;" %"
215         PRINT
220         PRINT "Preisnachlaß = DM "; NACHLASS
230         PRINT "Nettopreis =    DM "; NETTO
240         !-----------------------------------------------------
250         PRINT: PRINT
260         INPUT PROMPT "Möchten Sie einen weiteren
            Preisnachlaß berechnen? ": ANTW$
270 LOOP WHILE UCASE$(ANTW$) = "J"
280 END
285 ! Hauptprogramm endet
290 ! ---------------------------------------------------------------
295 ! Definition der Funktion
300 DEF RABATTBERECHNUNG
310         LET NACHLASS = (PREIS/100) * RABATT
320         LET NETTO = PREIS - NACHLASS
330 END DEF
```

Das Programm hat noch einen kleinen Fehler: Wenn der Benutzer auf die Frage in Zeile 260 mit einer Antwort reagiert, die nicht vorgesehen war (z.B. „ja"), dann wird das Programm beendet. Der Computer versteht nur „j" oder „J" als Aufforderung, eine weitere Berechnung durchzuführen. Um das zu beheben, ersetzen Sie die Zeile 270 durch die folgende:

```
270 LOOP WHILE UCASE$(ANTW$(1:1)) = "J"
```

Hier prüft das Programm nach, ob der erste Buchstabe von ANTW$, wenn er als Großbuchstabe übersetzt wird (UCASE$), mit dem Zeichen "J" identisch ist. Der Computer wird also "j", "ja", "J", "Ja", usw. richtig verstehen.

Mit ein wenig Geschick kann man weitere Fehleingaben abfangen. Falls der Benutzer nicht "j" oder "ja" oder "n" oder "nein" (mit oder ohne Großschreibung) eingibt, könnte der Computer „Wie bitte?" schreiben und auf eine erneute Antwort warten. Wirklich gute Programme sollten immer Anweisungen enthalten, die Fehleingaben des Benutzers korrigieren und ein ungewolltes Aussteigen aus dem Programm verhindern.

Sie sollten sich auch die Mühe machen, die Ausgaben des Programms in netter Form zu präsentieren. So ist es z.B. angenehm, wenn der Bildschirm nicht mit zu vielen Infor-

mationen vollgeschrieben ist. Besonders wenn man von einem Programmteil zum nächsten übergeht, sollte man daher den Bildschirm löschen und mit einer übersichtlichen Anzeige neu gestalten.

Die Computer-Sprache BASIC hat, wie viele natürliche Sprachen auch, eine große Anzahl von „Dialekten". Viele Computer-Hersteller haben ihre eigenen privaten Versionen von BASIC erfunden und mit unterschiedlichen Anweisungen versehen. Aber der Kern der Sprache und ihre grundsätzliche Philosophie ist gleich geblieben.
Der ENTERPRISE verwendet den Standard-BASIC-Dialekt, der von der **European Computer Manufacturers' Association** und dem **American National Standards Institute** empfohlen wird.
Standard-BASIC wurde entwickelt, um die Schwierigkeiten, die bei der Verwendung der vielen verschiedenen BASIC-Dialekte entstanden, zu überwinden. Es ist daher möglich, Programme zwischen dem ENTERPRISE und anderen Maschinen, die Standard-BASIC verwenden, auszutauschen. Diese Version von BASIC besitzt zudem einige sehr leistungsfähige Merkmale, die in anderen, nicht standardisierten Dialekten fehlen.
Es kann allerdings vorkommen, daß Sie auch Programme, die in einem herkömmlichen BASIC geschrieben wurden, auf dem ENTERPRISE implementieren wollen. Glücklicherweise benutzen solche Programme meist ein „Minimal-BASIC", das vor einigen Jahren von vielen Computer-Herstellern eingeführt wurde. Vielleicht haben Sie von dem sehr weitverbreiteten Microsoft-BASIC gehört, das auf einer Mehrzahl der Maschinen läuft und den Befehlssatz von Minimal-BASIC unterstützt.
Um die Austauschbarkeit von BASIC-Programmen zu gewährleisten, kennt der ENTERPRISE neben Standard-BASIC auch noch die wichtigsten Anweisungen von Minimal-BASIC. In diesem Kapitel wollen wir uns diesen Anweisungen zuwenden.

SPRÜNGE UND VERZWEIGUNGEN

Man kann Programme so strukturieren, daß sie ihre Arbeit streng sequentiell verrichten, aber die meisten Programmierer ziehen es vor, ein Hauptprogramm zu schreiben, das an geeigneten Stellen mit Verzweigungen versehen ist. Man kann solche Programme mit einem Baum vergleichen, der einen Stamm und Zweige besitzt. Man sollte es jedoch vermeiden, allzu sorglos von Ast zu Ast zu springen! Solche Dinge bergen gewisse Gefahren.
Die zwei wichtigsten Verzweigungen von Minimal-BASIC werden mit GOTO und GOSUB eingeleitet. Sie veranlassen den Computer, von einer Stelle im Programm zu einer ganz anderen zu springen.

Hinter GOTO muß immer eine Zeilennummer stehen, die dem Computer mitteilt, wohin er springen soll. Hier ein Beispiel:

```
100 CLEAR SCREEN
110 INPUT PROMPT "Mögen Sie Computer? ": ANTW$
120 IF UCASE$(ANTW$(1:1)) = "J" THEN
130         GOTO 190
140 ELSE IF UCASE$(ANTW$(1:1)) = "N" THEN
150         GOTO 210
160 ELSE
170         GOTO 110
180 END IF
190 PRINT "Das freut mich aber!"
200 END
210 PRINT "Das ist aber schade!"
220 END
```

Natürlich können Sie das Programm auch mit DO/LOOP, CASE oder mit Funktionen schreiben, das Ergebnis wäre das gleiche. Sie werden feststellen, daß GOTO weniger handlich in größeren Programmen ist. Man kann sich leicht in den vielen Zweigen des Baumes verlieren!

GOSUB

Ebenso wie GOTO läßt GOSUB den Computer zu einer anderen Stelle im Programm springen, allerdings kehrt er anschließend zurück zur Anweisung, die hinter dem GOSUB-Aufruf steht.
Wenn man GOSUB verwendet, muß man den angesprungenen Programmteil — man nennt ihn eine „Subroutine" — mit RETURN abschließen. Das sagt dem Computer, daß er zu der GOSUB-Zeile zurückkehren soll. Man kann solche Subroutinen natürlich auch in Form von Funktionen verwenden. Das hat den Vorteil, daß man sie statt mit einer Zeilennummer mit einem Namen aufrufen kann.

WIE MAN FEHLER VERMEIDET

Wenn Sie GOSUB/RETURN verwenden wollen, gibt es einige Dinge, die Sie beachten sollten.
Die Subroutine, die von GOSUB angesprungen wird, muß **immer** mit RETURN abgeschlossen werden. Wenn Sie das vergessen, wird der Computer nicht mehr zum Hauptprogramm zurückkehren.
Es gibt eine weitere Gefahr: Im Gegensatz zu Funktionen ist eine solche Subroutine kein abgetrennter Programmteil, den der Computer ignoriert, solange sie nicht aufge-

rufen wird. Wenn der Computer in die Subroutine gerät, ohne daß er durch GOSUB dorthin geschickt wurde, wird ihn die RETURN-Anweisung verwirren (wohin soll er zurückkehren?).
Um dieses Problem zu vermeiden, ist es sinnvoll, alle Subroutinen für die GOSUB-Anweisungen am Ende des Programms zu plazieren und eine END- oder STOP-Anweisung davorzusetzen. Der Computer beendet das Programm, bevor er zu den Subroutinen kommt.
Das nachfolgende Programm zeigt, wie das gemeint ist.

```
100 CLEAR SCREEN
110 PRINT "Lebensraum pro Kopf der Bevölkerung"
120 PRINT "**********"
130 PRINT
140 PRINT "Dieses Programm berechnet die durch-"
150 PRINT "schnittliche Quadratmeter-Zahl, die"
160 PRINT "jedem Bürger einer Stadt zur Ver-"
170 PRINT "fügung steht. Sie können vergleichen,"
180 PRINT "welche Städte übervölkert sind."
190 PRINT
200 WAIT 10
210 CLEAR SCREEN
220 PRINT "Geben Sie bitte folgende Daten ein:"
230 PRINT "1. Name der Stadt,"
240 PRINT "2. Fläche in Quadratkilometer,"
250 PRINT "3. Bevölkerung,"
260 PRINT
270 INPUT PROMPT "Stadt: ": STADT$
280 INPUT PROMPT "Fläche: ": FLAECHE
290 INPUT PROMPT "Bevölkerung: ": BEVOELK
300 GOSUB 2000
310 CLEAR SCREEN
320 PRINT
330 PRINT "Die Stadt "; STADT$
340 PRINT "mit einer Fläche von "; FLAECHE; "
    "Quadratkm."
350 PRINT "und einer Bevölkerung von "; BEVOELK
360 PRINT "bietet jedem seiner Bürger"
370 PRINT RAUM; "Quadratmeter Lebensraum --"
380 PRINT "im Durchschnitt!"
390 PRINT: PRINT
400 WAIT 5
410 INPUT PROMPT "Noch eine Stadt? (j/n) "; ANTW$
420 IF UCASE(ANTW$(1:1)) = "J" THEN GOTO 220
```

```
430 END
440 !-----------------------------------------------------------
2000 ! Berechnung des Lebensraums
2010 LET QMETER = FLAECHE * 1000 * 1000
2020 LET RAUM = QMETER/BEVOELK
2030 RETURN
```

Wir haben in diesem Programm GOSUB verwendet, um die eigentliche Berechnung durchzuführen (Subroutine Zeile 2000 bis 2030). In Zeile 420 finden Sie auch noch eine GOTO-Anweisung, die eine neue Berechnung einleitet, wenn der Benutzer die Frage mit „j" beantwortet.
Sehr wichtig ist die Zeile 430, die das Hauptprogramm ordnungsgemäß mit END abschließt. Wenn Sie diese Zeile weglassen, würde der Computer bei einem „n" in das Unterprogramm geraten und auf ein RETURN treffen, ohne eine GOSUB-Anweisung bekommen zu haben. Er würde mit einer Fehlermeldung reagieren.
Vielleicht haben Sie das Gefühl, daß GOTO und GOSUB weniger elegant und übersichtlich sind als DO/LOOP und die Funktionen. Das ist sicherlich der Fall. Sie sind jedenfalls wesentlich fehleranfälliger und gute Programmierer kommen völlig ohne sie aus.

DIE DIM-ANWEISUNG

Schauen Sie nach, wie wir im Kapitel über den Umgang mit größeren Datenmengen Speicherplatz für Felder reserviert haben. Es gibt in Minimal-BASIC eine andere, weniger vielseitige Methode dafür.
DIM A(10) macht Speicherplatz für ein eindimensionales Feld mit den Elementen A(0) bis A(10) frei, DIM A(4,4) für ein zweidimensionales Feld mit 16 Elementen. Man kann, ebenso wie bei STRING oder NUMERIC, auch bei DIM einen Zahlenbereich spezifizieren (X TO Y). Aber anders als bei STRING kann man nicht angeben, wie lang ein Zeichenketten-Element im Höchstfall sein darf. Wenn man sehr lange Programme schreibt, die mehrere Zeichenketten-Felder enthalten, kann das ein gravierender Nachteil sein. Ohne die Längenbegrenzung kann man nämlich sehr leicht die Speicherkapazität des Computers überschreiten.

DIM wurde nur deswegen zur Verfügung gestellt, um die Kompatibilität mit anderen BASIC-Dialekten zu gewährleisten. Hier ist ein kurzes Programm, das die Verwendung vorführt. Denken Sie daran, daß ebenso wie bei STRING A$(8) die Anweisung DIM A$(8) 9 Elemente schafft: A$(0) bis A$(8).

```
100 DIM A$(8)
110 FOR N = 0 TO 8
120         READ A$(N)
130 NEXT N
140 !-----------------------------------------------------------
150 FOR N = 0 TO 8
160         PRINT A$(N); " ";
170 NEXT N
180 !-----------------------------------------------------------
190 DATA Das, ist, ein, "Feld,", das,
200 DATA mit, DIM, deklariert, wurde.
```

Damit Sie die Fähigkeiten des Computers und der angeschlossenen Peripherie-Geräte auf vielfältigste Weise nutzen können, gestattet der ENTERPRISE Ihnen, die logischen „Kanäle" im Innern des Computers zu bestimmen.
Ein Kanal (engl.: channel) ist ein Weg oder eine Verbindung zwischen zwei Teilen der Maschine. Man kann die Kommunikation zwischen den beiden Teilen einrichten, indem man das „Zaunzeichen" # (oder in manchen Zeichensätzen das Pfundzeichen) zusammen mit der Kanalnummer der eigentlichen Anweisung vorausschickt, z.B. PRINT #104:"Hallo".
In dieser Weise können Sie alle Eingabe/Ausgabe-Kanäle immer selber bestimmen. Da Sie jedoch nicht genötigt werden sollen, vor jedem Umgang mit dem Computer alle logischen Verbindungen zwischen den einzelnen Teilen selber einzurichten, sind eine ganze Reihe von Kanälen „voreingestellt". Man nennt sie „Default Kanäle". Der ENTERPRISE geht z.B. davon aus, daß Sie mit der PRINT-Anweisung normalerweise eine Mitteilung auf den Bildschirm bringen wollen. Also bewirkt die Anweisung PRINT „Hallo" (ohne Kanal-Nummer), daß die Mitteilung auf dem Bildschirm erscheint.
Wenn Sie andererseits die PRINT-Anweisung an den Drucker schicken möchten (sofern ein solcher angeschlossen ist), kann man das mit PRINT #104:„Hallo" erreichen. Sobald der Computer nämlich eingeschaltet wird, verbindet er den Kanal 104 mit dem Drucker-Anschluß, so daß alle Anweisungen, die an Kanal 104 gerichtet sind, automatisch an den Drucker weitergeleitet werden.
Man kann die Kanalnummer in einem Programm als Variable angeben, so daß der Computer selbst entscheiden kann, wohin eine Mitteilung geht.

```
100 INPUT PROMPT "Bitte geben Sie eine Mitteilung
    ein: ": A$
110 PRINT
120 PRINT "Wo soll die Mitteilung erscheinen?"
130 PRINT
140 PRINT "0    = Bildschirm"
150 PRINT "104 = Drucker"
160 DO
170         INPUT K
180 LOOP UNTIL K = 0 OR K = 104
```

```
190 PRINT #K: A$
200 END
```

Das Programm setzt natürlich voraus, daß Sie einen Drukker angeschlossen (und eingeschaltet) haben. Sie können das Experiment auch mit Kanal 101 (statt 104) durchführen, die eine Verbindung zur Grafik herstellt. Setzen Sie aber dann die Anweisung GRAPHICS in das Programm ein und sehen Sie, was passiert.
Sie sollten immer daran denken, daß sämtliche Ein- und Ausgabe-Operationen über Kanäle laufen, einschließlich Tongenerator und Cassetten-Rekorder. Wenn Sie allzu sorglos mit den Kanälen umgehen, können merkwürdige Dinge mit dem Computer passieren. Es ist sehr unwahrscheinlich, daß Sie damit Schäden anrichten können, aber es kann vorkommen, daß der Computer „abstürzt" und Sie ihn nur noch mit der ‚RESET'-Taste wieder zur Besinnung bringen können.
Weitere Details über die logischen Kanäle des ENTERPRISE erfahren Sie im Nachschlageteil dieses Handbuchs unter dem Stichwort OPEN (siehe S. 199)

Um Fehler zu lokalisieren und besondere, vom Programm unabhängige Ausnahmezustände zu bewältigen, aber auch um mit dem Netzwerk und bestimmten Zusatzgeräten problemlos umgehen zu können, besitzt der ENTERPRISE sogenannte „Ausnahme-Handler" (Engl.: exception handlers). Sie haben eine gewisse Ähnlichkeit mit Funktionen, brauchen aber nicht mit CALL aufgerufen zu werden.
Ein Ausnahmefall in einem Programmablauf ist etwas, das mehr oder minder unabhängig vom augenblicklich laufenden Programm passiert, das aber den Programmablauf beabsichtigt oder unbeabsichtigt beeinflußt. Die ‚STOP'-Taste verursacht einen Ausnahmefall, denn auch wenn das Programm diese Taste nicht explizit abfragt, wird es von ihr beeinflußt. Programmfehler sind auch Ausnahmefälle, ja sogar die häufigsten.
Wenn ein Programm einen Fehler enthält, der zum Programmabbruch führt, wird der Computer eine Fehlermeldung zusammen mit einer Kennzahl auf den Bildschirm bringen. Man kann den Computer aber auch dazu veranlassen, in gezielter Weise auf einen solchen Fehler zu reagieren. Das ENTERPRISE-BASIC gestattet Ihnen sogar, eigene Ausnahmefälle zu definieren, die normalerweise den Programmablauf nicht stören würden. Diese bekommen eine Kennzahl zwischen 1 und 999. Ebenso wie bei den anderen Ausnahmen kann man für den Fall, daß solche selbstdefinierten Ausnahmen eintreffen, Vorkehrungen treffen.
Wir wollen das gleich mit Hilfe eines praktischen Beispiels demonstrieren, in dem ein Ausnahmefall definiert wird, um Eingabefehler zu identifizieren. Dieser wird dann von einem Ausnahme-Handler bewältigt.

```
100 WHEN EXCEPTION USE EINGABEFEHLER
110 INPUT PROMPT "Geben Sie bitte ein langes Wort
    ein :": WORT$
120 IF LEN$(WORT$) < 12 THEN
130       CAUSE EXCEPTION 10
140 ELSE
150       PRINT WORT$; " ist schön lang!"
160 END IF
170 END WHEN
180 END
190 !-------------------------------------------------------------
----
200 HANDLER EINGABEFEHLER
```

```
220         IF EXTYPE = 10 THEN
230           PRINT WORT$; " ist kein langes Wort!"
240         ELSE IF EXTYPE <> 10 THEN
250           EXIT HANDLER
260         END IF
270 END HANDLER
```

Das Programm akzeptiert nur Wörter, die 12 Buchstaben oder länger sind. Bei kürzeren Wörtern tritt ein Ausnahmefall ein, der von dem in der zweiten Programmhälfte (Zeilen 200 bis 280) definierten Handler bewältigt wird. Natürlich könnte man dasselbe Resultat mit anderen Methoden erzielen, aber Sie sehen hier, wie ein Ausnahme-Handler funktioniert.

WHEN EXCEPTION USE EINGABEFEHLER bedeutet soviel wie „Wenn ein Fehler auftaucht, benutze den Ausnahme-Handler namens EINGABEFEHLER". Diese Bedingung ist solange gültig, bis eine Zeile mit END WHEN auftaucht.

In Zeilen 120 – 130 wird der Ausnahmefall definiert. Ist WORT$ weniger als 12 Buchstaben lang, verursacht das Programm die Ausnahme 10 (CAUSE EXCEPTION 10). Da wir gesagt haben, daß der Computer in einem solchen Fall den Ausnahme-Handler EINGABEFEHLER verwenden soll, unterbricht er nicht das Programm mit einer Fehlermeldung, sondern geht zu Zeile 200.

Der Handler EINGABEFEHLER prüft zunächst, ob der Fehler die Kennzahl 10 hat (EXTYPE = 10). Wenn das der Fall ist, bringt er die nachfolgende Mitteilung auf den Bildschirm. EXIT HANDLER bewirkt, daß der Computer, ähnlich wie bei EXIT DO oder EXIT FOR den Handler-Block verläßt.

Der ENTERPRISE hat die Fähigkeit, über eine einfache Kabelverbindung mit anderen Computern zu kommunizieren. Die anderen Computer müssen ebenfalls die gleichen Kommunikationsroutinen (Intelligent Net) unterstützen – was natürlich kein Problem ist, wenn es sich ebenfalls um ENTERPRISEs handelt.
Der besondere Vorteil vom Intelligent Net ist, daß man viele Computer miteinander verbinden kann. Wenn zwei Computer miteinander sprechen, halten sich die anderen da heraus, genau wie bei einem Telefongespräch.
Ihr Computer hat allerdings keine feste „Telephonnummer". Wenn Sie sich am Netzgespräch beteiligen wollen, müssen Sie eine Nummer selber wählen, die zwischen 1 und 16 liegen kann. Geben Sie beispielsweise folgendes ein:

```
SET NET NUMBER 5
```

Später können Sie übrigens immer mit ASK NET NUMBER feststellen, welche Nummer Sie gewählt haben.

ÜBERSPIELEN VON PROGRAMMEN

Sobald Ihr Computer einen Kanal eingerichtet und eine Netznummer hat, ist es sehr einfach, Programme mit anderen Maschinen auszutauschen. Tippen Sie einfach auf dem Computer, der das Programm empfangen soll, LOAD "NET-0:" ein. Damit kann er ein Programm empfangen, das von einem beliebigen anderen Computer des Netzwerks „abgeschickt" wird.
Wenn Sie ein Programm von einem bestimmten Computer – z.B. dem mit der Nummer 15 – empfangen wollen, schreiben Sie

```
LOAD "NET-15:"
```

Ihr Computer wird das Programm laden, das von dem Computer mit der Netznummer 15 stammt.
Natürlich muß der andere Computer das Programm abschicken. Sein Besitzer muß ihm eine Anweisung wie die folgende geben:

```
SAVE "NET-5:"
```

Das schickt das gegenwärtig geladene Programm an Computer Nummer 5.

ALLGEMEINE NACHRICHTEN

NET-0 hat eine besondere Verwendung. Man kann diese Kennzahl nicht an einen Rechner des Netzwerks vergeben, denn sie ist für die Kommunikation mit allen Rechnern bestimmt.
Diese Einrichtung ist sehr nützlich, wenn Sie kurze Mitteilungen haben, die an alle Rechner gerichtet sind, oder wenn Sie die Nummer eines bestimmten Rechners nicht kennen. Aber Sie sollten daran denken, daß Ihr Computer dabei keine „Empfangsbestätigung" erwartet. Bei normalen, gerichteten Kommunikationen sendet der Computer dagegen so lange, bis der angesprochene Computer ihm bestätigt, daß er die Nachricht empfangen hat. Auch kann man auf NET-0 die Übertragungsgeschwindigkeit nicht einstellen, was bei zielgerichteter Kommunikation ohne weiteres geht.
Die Anweisung

```
SAVE "NET-0:"
```

sendet also das Programm an alle Rechner, aber es ist nicht sehr wahrscheinlich, daß einer es empfängt. Auch hat der Absender keine Gewißheit darüber, ob die Übertragung geklappt hat oder nicht.

EMPFANGSBEREIT FÜR ALLE NACHRICHTEN

NET-0 kann auch für die allgemeine Empfangsbereitschaft verwendet werden. Wenn ein Kanal für NET-0 geöffnet wurde, können Sie alle Nachrichten von anderen Computern empfangen, einschließlich solche, die gezielt an Sie gerichtet sind. Aber Sie können nicht bei gerichteten Übertragungen zwischen zwei anderen Rechnern „mithören".
Wenn Sie dagegen eine direkte Verbindung zu einem anderen Computer des Netzwerks eingerichtet haben, dann können Sie in der Zeit Nachrichten, die auf NET-0 ausgestrahlt werden, natürlich nicht empfangen.

KOMMUNIKATIONS-KANÄLE

Wenn Sie nun mit einem anderen Computer eine direkte Kommunikation einrichten wollen, ist es zweckmäßig, für diesen Zweck einen besonderen Kanal zu öffnen, z. B.

```
OPEN #110: "NET-15:"
```

Damit ist Kanal 110 für Kommunikation mit einem Computer mit der Kennzahl 15 eingerichtet. Dieser Kanal

kann fortan für normale Ein/Ausgabeanweisungen verwendet werden, wie z.B.:

```
PRINT #110: "Hallo Computer 15, hier ist eine
      Nachricht für Dich!"
```

Die zwischen Computern über das Netzwerk ausgetauschten Daten werden zunächst in einem „Buffer" gesammelt, d.h. vor dem Senden oder nach dem Empfang werden sie in einem Zwischenspeicher festgehalten. Es gibt zwei Befehle, die auf diese Buffer direkt wirken.
FLUSH #Kanal zwingt den Computer, die im Buffer gesammelten Daten sofort ins Netzwerk zu schicken. Normalerweise werden Nachrichten erst dann gesendet, wenn sie 256 Zeichen lang sind, oder wenn der Kanal geschlossen wird. Daher benutzt man FLUSH, wenn kurze Nachrichten sofort losgeschickt werden sollen.
CLEAR #Kanal:NET löscht die Ein- und Ausgabe-Buffer. Der Computer empfängt keine neuen Nachrichten, bevor der Empfangs-Buffer leer ist. Das ist notwendig, damit bereits empfangene, aber noch nicht verarbeitete Daten nicht überschrieben werden.
Wenn bereits empfangene Daten nocht nicht aus dem Buffer entfernt wurden (beispielsweise mit INPUT #Kanal), und diese Daten nicht mehr benötigt werden, dann kann man sie mit einem CLEAR-Befehl löschen. Wenn im Buffer Daten sind, die noch gesendet werden sollen, verwenden Sie zunächst FLUSH.
Die folgenden Anweisungen sind typisch für eine Kommunikation.

```
PRINT #110: "Nachricht für Computer 15!"
FLUSH #110
CLEAR #110:NET
LINE INPUT #110: A$
```

Man sollte für die Kommunikation über das Netzwerk keine Kanalnummern wählen, die kleiner sind als 100 (aber man sollte auch die Default-Kanäle vermeiden – siehe OPEN im Nachschlageteil). Die Kanäle von 1 bis 99 werden nämlich immer geschlossen, wenn Variablen insgesamt gelöscht werden, was im Direktmodus relativ häufig geschieht.

KOMMUNIKATION IM HINTERGRUND

Die interessantesten Anwendungen des Netzwerks verlangen, daß die Kommunikation im Hintergrund stattfindet, während im Computer ein anderes Programm läuft. Das Betriebssystem des ENTERPRISE unterstützt diese Einrichtung durch die Verwendung von ‚Interrupts' während des Netzbetriebs. Läuft ein BASIC-Programm, das die Anweisung SET INTERRUPT NET ON enthält, dann erfolgt beim Empfang einer Nachricht automatisch ein Interrupt. Dieser kann als definierter Ausnahmefall einem Ausnahme-Handler (siehe letztes Kapitel) übergeben werden, der eine entsprechende Reaktion des Computers bewirkt.
Dabei sollte der Ausnahme-Handler immer die Anweisung SET INTERRUPT NET OFF als erste Zeile enthalten. Sonst würde der Computer innerhalb des Handlers ständig Interrupts erhalten und gar nicht weiterkommen.
Der Ausnahme-Handler wird mit ASK NET CHANNEL die Netzwerk-Eingabe-Buffer abfragen. Verschiedene Kanäle können nämlich gleichzeitig eine Nachricht enthalten, und neue Nachrichten können während der Zeit eintreffen, in der der Ausnahme-Handler arbeitet. Nachdem Daten von einem Kanal übertragen wurden, sollte daher ASK NET CHANNEL so lange verwendet werden, bis der Wert 255 zurückgereicht wurde. Das bedeutet nämlich, daß keine weiteren Nachrichten vorhanden sind. SET INTERRUPT NET ON sollte dann wieder die letzte Zeile im Handler sein.
Da die Ausnahme-Handler von ‚Software-Interrupts' angesprochen werden, funktionieren sie nicht im Direktmodus. Es muß also ein BASIC-Programm im Vordergrund laufen, wenn in Verbindung mit der Netzwerk-Kommunikation Ausnahme-Handler eingesetzt werden sollen.

Das ‚Gehirn' des ENTERPRISE ist ein Z80-Mikroprozessor, der rund 500 verschiedene Operationen ausführen kann. Jede Operation wird durch eine Codezahl (‚Maschinen-Code') gekennzeichnet. Wenn Sie den Prozessor in Maschinen-Code programmieren, sprechen Sie mit ihm in seiner eigenen Sprache, nicht durch einen BASIC-Interpreter.
Es gibt zwei Gründe, beim ENTERPRISE Programmteile in Maschinen-Code zu schreiben. Entweder wollen Sie eine größere Geschwindigkeit (z.B. bei der Grafik oder Tonerzeugung) erreichen, oder Sie möchten bestimmte Fähigkeiten des Computers benutzen, die nicht von BASIC unterstützt werden.
Die Programmierung in Maschinensprache ist ein viel zu umfangreiches Thema, als daß wir es in einem einzigen Kapitel behandeln könnten. Es gibt aber eine Reihe von guten Büchern über die Z80-Programmierung, die Sie konsultieren können.
ENTERPRISE-BASIC kennt einige Befehle, die es Ihnen gestatten, Programmteile in Maschinen-Code (also in Hexadezimal-Zahlen) zu schreiben und dann innerhalb des BASIC-Programms zu verwenden (allerdings sind diese Befehle nicht Teil des ANSI-Standards).

ALLOCATE

Zunächst muß man für das Maschinen-Programm Speicherplatz reservieren. Entscheiden Sie, wie lang das Maschinen-Programm sein soll und schreiben Sie
ALLOCATE Zahl der Bytes
Der Befehl ALLOCATE löscht alle Variablen-Werte und sollte daher am Anfang des Programms stehen.

CODE UND HEX$

Das Maschinen-Programm wird mit CODE und einem Programmnamen abgespeichert:
CODE Name = Routine in Hex-Codes
Der Name hat dasselbe Format wie ein Variablenname.
CODE kann nur eine Zeichenkette abspeichern, so daß Ihre Hex-Codes vorher in das richtige Format gebracht werden müssen. Dazu verwendet man HEX$ wie folgt:
HEX$("Hex,Hex,...")
wobei Sie die Anführungszeichen sowie die Kommas zwischen den Zahlen nicht vergessen dürfen.
Möglich ist ebenfalls:
HEX$ (String)
Zum Beispiel gibt es eine Z80-Routine, die eine beliebige Zahl verdoppelt:

```
TEST: 29 ADD HL, HL ; verdoppelt die Zahl
      C9 RET
```

Diese Routine kann so in ein BASIC-Programm eingebaut werden:

```
100 ALLOCATE 2
110 CODE TEST = HEX$("29,C9")
```

Wenn Sie das Programm laufen lassen, wird diese Routine in dem bereitgestellten Speicherplatz abgelegt.

AUSFÜHRUNG DER ROUTINE

Ganz allgemein gesprochen schafft Ihr Maschinen-Code eine neue Funktion. Wir haben im Kapitel über Funktionen gesehen, daß es zwei Arten von Funktionen gibt. Die eine Art berechnet ein Ergebnis – die eingebaute BASIC-Funktionen fallen unter diese Kategorie. Ein gutes Beispiel ist SIN, das den Sinus-Wert eines Winkels berechnet:

```
PRINT SIN(53)
```

Die Zahl, die von der Funktion verarbeitet wird (sie steht in Klammern), nennt man das Argument der Funktion. Einige Funktionen haben kein Argument – z.B. RND –, aber alle liefern einen Wert, ein Ergebnis.
Die andere Art von Funktion ist eher wie eine Anweisung, eine Reihe von Operationen, die eine Aufgabe erfüllen – vielleicht den Bildschirm löschen oder den Grafikmodus einstellen. Eine solche Funktion liefert kein ‚Ergebnis'.
Wie Sie Ihr Maschinen-Programm aufrufen, hängt davon ab, zu welcher der beiden Arten es gehört. Wenn es einen Wert liefert, wie in unserem Beispiel mit der Verdopplung von Zahlen, benutzen Sie den folgenden Befehl
USR (Name, Argument)
zum Beispiel:

```
PRINT USR(TEST,2)
```

was die Zahl 4 auf den Bildschirm bringt; oder

```
LET A = 3 * USR(TEST,2)
```

wodurch A den Wert 12 bekommt.
Das Argument von USR wird am Anfang der Routine in HL gegeben, und der Wert, der von USR geliefert wird, ist der Inhalt von HL am Ende der Routine.

Wenn Ihre Routine eine befehlsähnliche Operation ist, benutzen Sie
CALL USR
Diese Anweisung liefert keinen Wert. Hier ein Beispiel:

```
CALL USR(NAME,0)
```

USR muß auch hier ein Argument erhalten, aber dessen Wert ist unwesentlich.
Maschinen-Befehle können aneinandergereiht werden, wie im folgenden Beispiel:

```
CALL USR(CLEAR,0) + USR(GRAPH,0) + USR(BILD,0)
```

WORD$

Die Anweisung WORD$ ermittelt die Anfangsadresse einer Maschinen-Routine. Zum Beispiel liefert

```
WORD$(TEST)
```

die Anfangsadresse der TEST-Routine als 2-Byte-String – LSB MSB. Das ist für Sprünge und Aufrufe besonders nützlich.

Der Nachschlage-Teil dieses Handbuchs enthält ein Verzeichnis aller BASIC-Wörter, die der ENTERPRISE kennt, zusammen mit ihrer Funktion und der Art ihrer Verwendung. Einige wurden im Lehrteil nur kurz erwähnt, einige überhaupt nicht. Sie sollten mit den Wörtern experimentieren und dabei alle Möglichkeiten des ENTERPRISE kennenlernen.
Wenn Sie BASIC schon kennen, werden Sie feststellen, daß dieser Abschnitt die beste Anleitung für das Intelligent Standard BASIC (IS-BASIC, Copyright Intelligent Software Ltd, 1984) ist, das auf dem ENTERPRISE läuft.

ALLGEMEINE REGELN

Groß- und Kleinbuchstaben sind für BASIC gleichwertig, so daß FOR, For, for und fOr die gleiche Bedeutung haben.
Eine Programmzeile darf bis zu 250 Zeichen lang sein und eine Zeilennummer zwischen 1 und 9999 haben. Sie kann mehrere Anweisungen enthalten, die durch Doppelpunkte voneinander getrennt sind. Alles was nach dem THEN einer IF/THEN-Anweisung erlaubt ist, kann in einer Zeile mit mehreren Anweisungen stehen. Ein Bezeichner kann bis zu 31 Zeichen lang sein, und alle Zeichen sind signifikant. Ein Bezeichner kann Buchstaben, Zahlen, Punkte und das Unterstreichungszeichen enthalten, wobei das erste Zeichen immer ein Buchstabe sein muß.
Das Ausrufezeichen (!) wird verwendet, um den Rest der Zeile als Kommentar zu kennzeichnen.
Im Direktmodus bewirkt ein Doppelpunkt am Anfang einer Zeile, daß der Rest der Zeile an das Betriebssystem weitergegeben wird, ohne interpretiert zu werden.
Der Interpreter löscht Leerzeichen vor und nach der Zeilennummer und dem ersten Schlüsselwort und am Ende der Zeile. Dann rückt er das Programm für jeden neuen Block ein. Bei DEF, DO, SELECT, FOR, HANDLER und WHEN wird die nächste Zeile um 2 Zeichen eingerückt. ELSE und CASE innerhalb eines eingerückten Blockes werden um 2 Zeichen nach rechts verschoben. LOOP, END und NEXT beenden den eingerückten Abschnitt. Vor 1- bis 3stelligen Zeilennummern werden Leerzeichen eingefügt, um einen glatten rechten Rand zu erhalten.

```
  1 LET A = 0
 10 DO WHILE A < 10
100     LET A = A + 1
110     SELECT CASE A
```

```
 120   CASE 1
 130     PRINT "ERSTES MAL"
 140   CASE ELSE
 150     PRINT "NICHT DAS ERSTE MAL"
 160   END SELECT
 170   PRINT A
 180 LOOP
1000 END
```

Für weitere Informationen über die Syntax und die Vereinbarungen über Standard-BASIC siehe den „Draft Proposal for Standard BASIC" vom ANSI-KOMITEE X3J2/82-17.
Im nachfolgenden stehen Schlüsselwörter in **halbfetten GROSSBUCHSTABEN** auf der linken Seite. Es folgt das Format für die Verwendung der Schlüsselwörter in Normalschrift. Alle Beispiele sind kursiv gedruckt.

MEHRFACH-PROGRAMME

IS-BASIC auf dem ENTERPRISE bietet die Möglichkeit, mehrere Programme zur gleichen Zeit im Computer bereitzuhalten. Jedes Programm hat seine eigenen Zeilennummern und seine eigenen Variablen.
Ein Programm kann entweder durch eine Zahl oder durch einen in einer PROGRAM-Zeile angegebenen Namen angesprochen werden. Siehe **CHAIN**, **EDIT** und **PROGRAM**.
Zu jeder Zeit ist eines der Programme (Default ist Programm 0) das „aktuelle" Programm, auf das Befehle wie LIST und RENUMBER wirken. Die Nummer dieses Programms wird in der ‚Statuszeile' an der Oberkante des Bildschirms angezeigt.
Programm 0 kann etwa 42K des Speichers verwenden. Andere Programme sind auf jeweils 32K beschränkt.

ERWEITERUNGEN

Es besteht die Möglichkeit, das BASIC zu erweitern. Diese Erweiterungen können entweder von Cassette oder Diskette geladen oder in einem Zusatzteil für den Computer untergebracht werden. Erklärungen der zusätzlichen Kommandos oder Funktionen werden in den mit diesen Produkten mitgelieferten Anleitungen enthalten sein.

DATENTYPEN

Es gibt zwei verschiedene Datentypen: numerische Daten und Zeichenketten (Strings). Numerische Variablen haben Namen, die den Regeln für Bezeichner (siehe Allgemeine

Regeln) entsprechen. Bezeichner für String-Variablen müssen mit dem $-Zeichen enden.
Numerische Werte werden in binär codierter Dezimal-Arithmetik (BCD) berechnet und mit 10 Stellen ausgedruckt. Zahlen zwischen 1e-64 und 9.999999999e62 können verarbeitet werden.
Strings haben eine Maximallänge von 254 Zeichen, wenn sie für diese Länge deklariert wurden (siehe STRING). Auf Teile von Strings kann in der Form String-Bezeichner(x:y) zugegriffen werden, wobei der Teil-String mit dem Zeichen an der Stelle x des ursprünglichen Strings beginnt und mit dem Zeichen an der Stelle y endet. Wenn x bzw. y nicht angegeben wird, so wird der Anfang bzw. das Ende des Strings als Default angenommen.

OPERATOREN

Arithmetische Operatoren:
* -- Multiplikation
/ -- Division
∧ -- hoch
\+ -- Addition
\- -- Subtraktion
String-Operator:
& -- Verkettung
Relationale Operatoren:
> -- größer als
< -- kleiner als
= -- gleich
≧ -- größer oder gleich
≦ -- kleiner oder gleich
≦ -- ungleich
AND -- logisches UND (wahr/falsch)
OR -- logisches ODER (wahr/falsch)
BAND -- binäres logisches UND
BOR -- binäres logisches ODER

Zeilennummer

Zeilennummer Text
Zeilennummer Leerzeichen
Zeilennummer
Ersetzt eine BASIC-Zeile oder fügt eine neue hinzu. Wenn der Zeilennummer nur ein Leerzeichen folgt, wird eine Zeile, die nur ein ! enthält, eingefügt. Wird nur die Zeilennummer eingegeben, so wird diese Zeile gelöscht.
Nur im Direktmodus möglich. Alle Variablen werden dabei gelöscht. Beachten Sie, daß Anweisungen, die Variablen löschen, außerdem alle offenen Kanäle von 1 bis 99 schließen (siehe OPEN).

AUTO

Schaltet die automatische Zeilennumerierung ein. Nur im Direktmodus möglich.
AUTO
AUTO AT 100 STEP 10
AUTO STEP 100
Default für die Anfangsnummer ist 100, die Default-Schrittlänge ist 10. Neue Zeilen ersetzen alte Zeilen mit denselben Zeilennummern.
AUTO wird mit der ‚STOP'-Taste beendet.

CONTINUE

Im Direktmodus setzt dieser Befehl das Programm in der Zeile nach einer STOP-Anweisung oder nach der Stelle, an der das Programm durch die ‚STOP'-Taste abgebrochen wurde, fort.
Am Ende eines Ausnahme-Handlers setzt es den normalen Programmablauf fort, und zwar bei der Anweisung nach derjenigen, die den Ausnahme-Aufruf veranlaßt hat.

DELETE

DELETE Zeilenangabe TO Zeilenangabe,...
DELETE Zeilenangabe – Zeilenangabe,...
DELETE Blockname
Löscht Zeilen eines Programmes. Nur im Direktmodus möglich. Löscht alle Variablen.
DELETE LAST
DELETE FIRST TO 100
DELETE 1 TO 199, 300, 500 TO 9999
Erlaubt ist auch, „-" anstatt TO zu verwenden. Wenn die erste (oder die letzte) Zahl einer Bereichsangabe weggelassen wird, wird als Default die erste (bzw. die letzte) Zeile des Programms angenommen.
Also z.B. DELETE FIRST-100, 500-LAST
oder DELETE TO 100, 500-
statt DELETE FIRST TO 100, 500 TO LAST
Zeilen, die eine Funktion P definieren, können durch DELETE P gelöscht werden. DELETE allein löscht alle Programmzeilen, was durch die ‚STOP'-Taste abgebrochen werden kann.

EDIT

EDIT Programm-Nummer
EDIT „Name"
Macht das angegebene Programm zum aktuellen Programm, so daß sich LIST, RENUMBER, RUN usw. darauf beziehen.
Nur im Direktmodus möglich. Siehe **CHAIN**, **INFO** und **PROGRAM**.

INFO

Druckt den Betrag des im System vorhandenen Speichers und die Anzahl freier Bytes aus. Informationen über die im Speicher befindlichen Programme werden außerdem in Tabellenform angezeigt:

Programmnummer	Anzahl der Bytes in dem Programm	Erste Zeile des Programms

INFO wird nur im Direktmodus ausgeführt und löscht alle Variablen.

NEW

Löscht das momentane Programm. Nur im Direktmodus möglich. Löscht alle Variablen.

NEW ALL

Löscht alle Programme im Speicher des Computers und kehrt zu Programm 0 zurück.

LIST

LIST #Kanal:Zeilenangabe TO Zeilenangabe
LIST #Kanal:Zeilenangabe – Zeilenangabe
LIST Blockname
Listet das ganze Programm oder einen Teil davon. **Kann durch die ‚STOP'-Taste abgebrochen und durch die ‚HOLD'-Taste unterbrochen werden.** Wird nur im Direktmodus ausgeführt. Der Default-Kanal ist #0.
LIST 300
LIST 300 TO 400
LIST FIRST TO 900, 1000, 2000 TO LAST
LIST TO 500, 700 TO
LIST MEINE–FUNKTION
LIST LAST
Statt TO ist auch „-" erlaubt (vgl. **DELETE**).
z.B. LIST FIRST – 100, 500 – LAST
statt LIST FIRST TO 100, 500 TO LAST

LLIST

LLIST List-Ausdruck
Genau wie LIST, Default-Kanal ist hier jedoch #104, der Drucker-Listing-Kanal.

REM

REM Kommentar
Kommentar-Zeile. REM muß am Anfang der Zeile stehen. Um größere Flexibilität zu erreichen, sollte man ‚!' verwenden.

RENUMBER

RENUMBER Zeilenangabe TO Zeilenangabe AT Ausdruck STEP Ausdruck. RENUMBER Blockname AT Ausdruck STEP Ausdruck

Numeriert das ganze Programm oder einen Teil davon neu. Nur im Direktmodus möglich.
RENUMBER FIRST TO 100
RENUMBER 10 TO 100 AT 300 STEP 10
RENUMBER STEP 100
RENUMBER MEINE_FUNKTION AT 5000
STEP und AT können in beliebiger Reihenfolge stehen oder weggelassen werden. Wenn STEP weggelassen wird, ist STEP 10. Wenn AT weggelassen wird, wird die erste Zeilennummer des neu zu numerierenden Programmstücks verwendet. Wenn kein Zeilennummern-Bereich angegeben ist, wird das gesamte Programm neu numeriert, wobei der Default-Wert für AT 100 ist.
Der Name eines DEF- oder HANDLER-Blockes kann statt eines Zeilennummern-Bereichs angegeben werden.
Bezüglich der Syntax der Zeilenangaben siehe DELETE.
Alle Zugriffe im Programm auf neu numerierte Zeilen werden automatisch geändert.
RENUMBER kann die Reihenfolge der Zeilen innerhalb des Programms nicht ändern. Daher wird der RENUMBER-Befehl nicht ausgeführt, wenn er die Reihenfolge der Zeilen ändern oder zu hohe Zeilennummern erforderlich machen würde.

RUN

RUN (Parameter-Liste)
RUN Zeilennummer
RUN #Kanal:Dateiname (Parameter-Liste)
Run Gerätename (Parameter-Liste)
RUN alleine startet das Programm in der ersten Zeile. Wenn eine Zeilennummer angegeben ist, wird in dieser Zeile gestartet. Ist ein Programmname angegeben (die Kanalangabe ist optional), so wird das Programm geladen und dann gestartet.
Parameter können Programmen mit RUN übergeben werden, doch müssen diese zu den für das Programm deklarierten Parametern passen. Siehe PROGRAM. Löscht alle Variablen.

START

Wenn kein Programm im Speicher steht, lädt und startet dieser Befehl das erste Programm von Kanal 106. Steht jedoch ein Programm im Speicher, so funktioniert er wie RUN.

STOP

Beendet die Programmausführung und druckt die STOP-Mitteilung.

Beachten Sie, daß CONTINUE nach dem STOP-Befehl erlaubt ist.

TRACE

TRACE ON TO #Kanal
TRACE OFF
Wenn TRACE eingeschaltet ist, wird jeweils die Zeilennummer der Zeile, die momentan bearbeitet wird, ausgegeben. Die Ausgabe erfolgt auf den Kanal 0, wenn sie nicht durch eine angegebene Kanalnummer umgeleitet wurde.

OPTIONEN

FKEY

SET #Kanal:FKEY Tastennummer-String
Ordnet der Funktionstaste den angegebenen String zu, der beim Drücken dieser Taste produziert wird (ein leerer String verursacht einen Ausnahmezustand). Der Default-Kanal ist 105.
Die Funktionstasten sind von 1 – 16 durchnumeriert. Die Nummern 1 – 8 sind die ohne „shift" gedrückten Funktionstasten. Die Nummern 9 – 16 sind die Tasten 1 – 8 mit zusätzlich gedrückter „shift"-Taste.
Die Funktionstasten werden durch das System mit Default-Strings belegt und Neudefinition der Tasten löscht die jeweiligen Default-Belegungen. CLEAR FKEYS setzt alle Tasten auf die ursprüngliche Belegung zurück.
Eine automatische Eingabe des Strings erreicht man, indem man CHR$(13) (= ‚ENTER') an den String anhängt.

KEY CLICK

SET KEY CLICK ON/OFF
Gibt ab, ob bei jedem Tastendruck ein Klicken zu hören sein soll oder nicht. Kann auch in Verbindung mit TOGGLE verwendet werden.

KEY DELAY

SET KEY DELAY Ausdruck
Stellt die Zeit (in Einheiten zu je 1/50 Sekunde) ein, nach der die automatische Zeichenwiederholung (Auto-Repeat) beginnt. Kann in Verbindung mit ASK verwendet werden.

KEY RATE

SET KEY RATE Ausdruck
Stellt die Zeit in Einheiten zu je 1/50 Sekunde ein, die zwischen den einzelnen Zeichen bei der automatischen Zeichenwiederholung liegt. Kann in Verbindung mit ASK verwendet werden.

:BRD

Schaltet das Betriebssystem, eingebaute Textbearbeitung und BASIC in deutschen Modus. Stellt alle Fehlermeldungen und den Zeichensatz um. Gleich **EXT "BRD"**

:UK

Schaltet das Betriebssystem, eingebaute Textbearbeitung und BASIC in englischen Modus. Stellt alle Fehlermeldungen und den Zeichensatz um. Gleich EXT "UK"

DATE

DATE Datums-String
Bestimmt das laufende Datum, das vom Computer geführt wird. Das Datum wird automatisch erhöht, wenn die ebenfalls vom Computer geführte laufende Zeit Mitternacht erreicht.
Das Datum wird im internationalen Standardformat JJJJMMTT angegeben.
DATE "19850727"
entspricht dem 27.Juli 1985.
Kann auch im Direktmodus verwendet werden. Siehe **DATE$**-Funktion.

DIM

DIM Array-Liste
Deklariert numerische oder String-Arrays. Der Default-Wert für die Untergrenze ist 0. Eine oder zwei Dimensionen sind erlaubt. Die maximale Stringlänge kann nicht durch DIM bestimmt werden, daher wird hier die Default-Länge von 132 Zeichen verwendet (vgl. **STRING**).
DIM A(1 TO 10), FRED$(9), B(-7899 TO -7890)
Alle Arrays in diesem Beispiel haben 10 Elemente.

LET

LET Variablenliste = Ausdruck
Einfache Zuweisung. LET kann weggelassen werden, wenn der Variablenname kein Schlüsselwort ist. Beim Listen oder Abspeichern des Programms wird LET eingefügt, damit das Programm dem Standard entspricht. Kann auch im Direktmodus ausgeführt werden.
Ein Wert kann mehreren Variablen gleichzeitig zugeordnet werden:
LET A, B(4), C = 0
A_VAR = A_VAR + 1
A$, FRED$ = "Er sagte "&"Nein"&"."&FRED$(I:J)
LET INPUT = 3

NUMERIC

NUMERIC Variablen/Array-Liste
Deklariert numerische Variablen oder Felder (lokal, wenn sie in einer DEF-Funktion deklariert wurden). Die Default-Untergrenze ist 0. Vgl. DIM.
NUMERIC I, A(10), B(-10 TO 20, 2 TO 4)

OPTION

OPTION ANGLE DEGREES/RADIANS
Wählt die Basiseinheit für alle folgenden Operationen, die Winkel verwenden. Default ist RADIANS (Radiant).

RANDOMIZE

Normalerweise gibt es in einem Programm immer die gleiche Folge von Zufallszahlen. RANDOMIZE verändert die Zufallszahlen zu einer neuen Folge.

STRING

STRING Variablen/Array-Liste #n
Deklariert Stringvariablen oder Arrays (Felder) mit Maximallänge. Defaultlänge ist 132, #n nach dem Wort STRING oder der Variablendeklaration setzt die Länge auf n. Die Default-Untergrenze für ein Array ist 0.
STRING #8 NACHNAME$ #20, VORNAME$, WOHNORT$
In diesem Beispiel wird für NACHNAME$ eine Maximallänge von 20 Zeichen festgelegt, für VORNAME$ und WOHNORT$ jeweils 8 Zeichen.
STRING NAME$
Hier hat NAME$ eine maximale Länge von 132 Zeichen.
STRING NAMEN$(4 TO 99)#10
Dieses Array hat 96 Elemente, jedes davon ist zehn Zeichen lang.
Beachten Sie: Ein DIM-Befehl kann nicht verwendet werden, um die Länge einer Stringvariablen zu definieren.

TIME

TIME Zeit-String
Stellt die „innere Uhr" des Computers ein. Die angegebene Zeit wird jede Sekunde automatisch erhöht.
Die Zeit wird im Format HH:MM:SS angegeben.
TIME "15:35:00"
entspricht 15 Uhr 35.
Kann auch im Direktmodus verwendet werden. Siehe **TIME$**-Funktion, **WAIT DELAY**-Befehl und die Maschinenoption **TIMER**.

WAIT DELAY

WAIT DELAY Ausdruck
Bewirkt, daß das Programm für eine in Sekunden angegebene Zeitspanne angehalten wird.
WAIT DELAY 60
läßt das Programm eine Minute warten.
Die maximal erlaubte Wartezeit ist 32767 Sekunden.
Eine automatische Zeitbegrenzung, bei der das Programm weiterläuft, ermöglicht die **TIMER** Maschinenoption.
Das Wort DELAY ist optional.

*TRIGONOMETRISCHE FUNKTIONEN

Trigonometrische Funktionen arbeiten in Grad oder Radiant. Siehe OPTION-Anweisung. Minimal BASIC-Funktionen sind ABS, ATN, COS, EXP, INT, LOG, RND, SGN, SIN, SQR, TAB und TAN.

ACOS(X) Liefert den Arcuscosinus von X (den Winkel, dessen Cosinus X ist). ACOS ist die Umkehrfunktion zu COS. Also ist **ACOS(COS(X))** gleich X.

ANGLE(X,Y) Liefert den Winkel zwischen der positiven X-Achse und der Linie, die den Punkt (0,0) mit dem Punkt (X,Y) verbindet.

ASIN(X) Liefert den Arcussinus von X (den Winkel, dessen Sinus X ist). ASIN ist die Umkehrfunktion zu SIN. Also ist **ASIN(SIN(X))** gleich X.

ATN(X) Liefert den Arcustangens von X (den Winkel, dessen Tangens X ist). ATN ist die Umkehrfunktion zu TAN. Also ist **ATN(TAN(X))** gleich X.

COS(X) Cosinus von X.

COSH(X) Hyperbolischer Cosinus (cosinus hyperbolicus) von X.

COT(X) Cotangens von X.

CSC(X) Cosekans von X.

DEG(X) Wandelt X von Radiant in Grad um. DEG(X) = X#180/PI.

RAD(X) Wandelt X von Grad in Radiant um. RAD(X) = X#PI/180.

SEC(X) Der Sekans von X.

SIN(X) Der Sinus von X.

SINH(X) Der hyperbolische Sinus von X.

TAN(X) Tangens von X.

TANH(X) Hyperbolischer Tangens von X.

ABS(X) Liefert den Absolutwert (Betrag) einer Zahl. Dies bedeutet einfach, daß das Vorzeichen (falls vorhanden) entfernt wird. Aus ABS(-9) wird also 9 und aus **ABS(89)** wird 89.

CEIL(X) Liefert die kleinste ganze Zahl, die nicht kleiner als X ist (das bedeutet, daß X zur nächsten ganzen Zahl aufgerundet wird). **CEIL(3.45)** ergibt 4 und **CEIL(–3.45)** ergibt –3.

EPS(X) Der kleinste Wert, der zu X addiert oder von X abgezogen werden kann, damit der Computer eine Wertveränderung im Speicher registriert.

EXP(X) Die Funktion e∧X (Exponentialfunktion). Die Zahl „e" (= 2.71828...) ist die Basis des natürlichen Logarithmus.

FP(X) Die Funktion FP liefert die Nachkommastellen einer Zahl. **FP(1.23)** ergibt 0.23 und **FP(–1.23)** ergibt –0.23. FP ist das direkte Gegenstück zu IP.

INF Die größte positive Zahl, die der ENTERPRISE kennt – seine Vorstellung vom Unendlichen. Diese Zahl ist 9.999999999*10∧ 62.

INT(X) Liefert die größte ganze Zahl, die nicht größer als X ist. **INT(3.4)** ergibt 3 und **INT(–3.4)** ergibt –4 (–3.4 ist größer als –4!).

IP(X) Liefert den ganzzahligen Anteil von X, d.h. alle Nachkommastellen werden entfernt. **IP(9.9)** ergibt also 9 und **IP (–9.9)** ergibt –9.

LOG(X) Der natürliche Logarithmus (Logarithmus zur Basis e) von X.

LOG10(X) Logarithmus von X zur Basis 10.

LOG2(X) Logarithmus von X zur Basis 2.

MAX(X,Y) Liefert die größere Zahl von X und Y. MAX(6,99) ergibt 99.

MIN(X,Y) Wie MAX(X,Y), liefert aber die kleinere Zahl.

MOD(X,Y) X modulo Y. Oder, mit anderen Worten, der ganzzahlige Rest von X geteilt durch Y. MOD(7,3) = 1, aber MOD (–1,3) = 2. Siehe auch REM(X,Y).

PI Liefert den Wert der Zahl Pi. Beim ENTERPRISE wurde diese Zahl gerundet zu 3.141592654.

REM(X,Y) Der Rest von X geteilt durch Y. Beachten Sie, daß REM (–1,3) = –1. Siehe MOD(X,Y).

RND Generiert eine Zufallszahl zwischen 0 und 1. In vielen Fällen ist es sinnvoll, Zufallszahlen zu multiplizieren und dadurch in größere Zahlen zu verwandeln. RND * 100 ergibt eine Zahl, die größer oder gleich 0 und kleiner als 100 ist. (RND wird niemals 1). Siehe auch RANDOMIZE.

RND(X) Generiert eine ganzzahlige Zufallszahl, die kleiner als X ist. Der größte erlaubte Wert für X ist 32767.

ROUND(X,N) Rundet X auf N Dezimalstellen. ROUND(1.7668,2) würde 1.77 ergeben, ROUND(–1.7668,2) würde –1.77 ergeben.

SGN(X) Liefert das Vorzeichen von X. Liefert –1 wenn X negativ ist, 0 wenn X = 0 ist und 1 wenn X größer als 0 ist.

SQR(X) Die Quadratwurzel von X. X muß positiv sein.

TRUNCATE(X,N) Trennt nach N Dezimalstellen alle folgenden Stellen von X ab.

CHR$(X) Liefert das Zeichen, dessen ASCII-Code-Nummer X ist.

INKEY$ Liefert das Zeichen, das der gedrückten Taste entspricht, oder einen leeren String (auch als Nullstring bezeichnet und als "" dargestellt), falls keine Taste gedrückt wurde.

LBOUND(A) Untergrenze der Dimension eines eindimensionalen Array A.

LBOUND(A,N) Untergrenze der Dimension N eines Array A.

LCASE$(A$) Wandelt alle Großbuchstaben in Kleinbuchstaben um.

LEN(A$) Die Anzahl der Zeichen (Länge) von A$.

LTRIM$(A$) Entfernt alle Leerzeichen, die am Anfang des Strings A$ stehen. **LTRIM$(" Hallo")** ist also „Hallo".

MAXLEN(A$) Liefert die maximale Länge, die für eine Stringvariable oder ein Array festgelegt wurde.

ORD(A$) Liefert den ASCII-Code für das Zeichen in Anführungszeichen bzw. den ASCII-Code des ersten Zeichens eines Strings.

POS(X$,Y$,M) Liefert die Position (von links nach rechts innerhalb des Strings) des ersten Zeichens beim ersten Auftreten von Y$ in X$, oder 0, falls Y$ nicht in X$ gefunden werden konnte. Wenn man eine Zahl nach dem zweiten String hinzufügt (also POS(X$,Y$,M)), kann man das Gerät veranlassen, mit der Suche nach Y$ in X$ an einer beliebigen Stelle zu beginnen. Ist X$ zum Beispiel „LONDON" und Y$ „ON", so würde **POS(X$,Y$)** 2 ergeben. **POS(X$,Y$,4)** teilt dem Computer mit, die Suche nach „ON" bei dem „D" in „LONDON" zu beginnen, und würde als Ergebnis 5 liefern.

RTRIM$(A$) Entfernt alle Leerzeichen, die am Ende des Strings A$ stehen. RTRIM$("Hallo ") ist also „Hallo".

SIZE(A) Die Anzahl der Elemente des Array A.

SIZE(A,N) Die Anzahl der Elemente, die in der Dimension N des Array A erlaubt sind.

STR$(X) Macht aus dem Wert von X einen String aus Zahlen ohne führende oder folgende Leerzeichen, aber mit einem Minuszeichen, falls X negativ ist.

UBOUND(A) Die Obergrenze der Dimension eines eindimensionalen Arrays.

UBOUND(A,N) Obergrenze der Dimension N des Arrays A.

UCASE$(A$) Wandelt alle Buchstaben im String A$ in Großbuchstaben um.

VAL(A$) Wandelt einen String in eine Zahl um (Gegenstück zu STR$). VAL beginnt mit der Umwandlung beim ersten Zeichen des Strings und beendet die Umwandlung, wenn das erste Zeichen erreicht wird, das keine Zahl ist.

DATE$

Liefert das momentane Datum im Standard-Format (JJJJMMTT). Siehe **DATE**-Anweisung.

FREE

Liefert die Menge des Speichers, der für das momentane Programm frei und verfügbar ist, in Bytes (siehe Glossary). Das ist nicht dasselbe wie die Menge des Speichers, die für BASIC frei und verwendbar ist (siehe INFO-Anweisung).

JOY(N)

Liefert einen Wert, der vom jeweiligen Zustand der Schalter des angegebenen Cursor-Steuerhebels oder Joysticks abhängt:

1 – rechts
2 – links
4 – unten
8 – oben
16 – Drucktaste

Beachten Sie, daß die Werte zusammengezählt werden, falls der Cursor-Steuerhebel in eine diagonale Richtung bewegt oder der Druckknopf zusätzlich zur Bewegung des Steuerhebels betätigt wurde.
Die Joysticks (durch N angegeben) sind von 0 bis 2 durchnumeriert, wobei 0 der eingebaute Cursor Steuerhebel ist. Für diesen wird die Leertaste als Druckknopf verwendet.

TAB(X)

Nur in PRINT-Anweisungen erlaubt. Bewegt die Cursorposition zur Spalte X der momentanen Zeile.

TIME$

Liefert die momentane Zeit im Standard-Format (HH:MM:SS). Siehe **TIME**-Anweisung.

DO

Do-Zeile
    beliebige Anzahl Befehle oder Blöcke
Loop-Zeile
Do-Zeile:
DO
DO WHILE vergleichender Ausdruck
DO UNTIL vergleichender Ausdruck
Loop-Zeile:
LOOP
LOOP WHILE vergleichender Ausdruck
LOOP UNTIL vergleichender Ausdruck
Die Struktur einer LOOP-Schleife ist als Block definiert, der eine DO-Zeile, eine beliebige Anzahl Anweisungen und eine LOOP-Zeile enthält. DO und LOOP können nicht in einer bedingten Zeile stehen.

```
DO WHILE A > 3 AND A < 10
   LET A = A + 1
   PRINT A
LOOP
```

Von außerhalb kann nicht in eine LOOP-Schleife hineingesprungen werden. Siehe auch **EXIT DO**.

END

Beendet die Programmausführung, markiert das Programmende.
Die Befehle **END DEF**, **END HANDLER**, **END IF**, **END SELECT** und **END WHEN** beenden den zugehörigen Block.

EXIT

EXIT DO
EXIT FOR
EXIT DEF
Bricht einen DO-, FOR- oder DEF-Block ab. Nur gültig innerhalb eines Blockes der jeweiligen Art.

FOR

For-Zeile
    beliebige Anzahl Anweisungen oder Blöcke
Next-Zeile
For-Zeile:
FOR einfache Variable = Ausdruck TO Ausdruck STEP Ausdruck
Die STEP-Anweisung darf weggelassen werden, der Default-Wert ist 1.
Next-Zeile:
NEXT

NEXT Variable
Die Struktur einer FOR-Schleife ist als Block definiert, der eine FOR-Zeile, eine beliebige Anzahl Anweisungen und eine NEXT-Zeile enthält. FOR und NEXT können nicht in einer bedingten Zeile stehen. Erlaubt in Minimal Basic.

```
FOR Y = 0 TO 10 STEP 2
    PRINT Y
NEXT Y
```

Der Wert der Schleifenvariablen nach dem Ende der Schleife ist der Schlußwert plus STEP-Ausdruck – im obigen Beispiel hat Y also den Wert 12.
Ineinander verschachtelte FOR-Schleifen dürfen nicht dieselbe Schleifenvariable verwenden. Die Grenze und die STEP-Zahl werden in interne Speicher übertragen, wenn die FOR-Zeile ausgeführt wird. Diese Werte können also nicht durch die Anweisungen innerhalb der Schleife verändert werden. Von außerhalb kann nicht in eine FOR Schleife hineingesprungen werden. Siehe auch **EXIT FOR**.

LOOP Siehe **DO**.

NEXT Siehe **FOR**.

CASE Siehe **SELECT**.

ELSE Siehe **IF**.

GOSUB

GOSUB Zeilennummer
Aufruf des Unterprogramms, das mit der angegebenen Zeile beginnt.

GOTO

GOTO Zeilennummer.
Die Programmausführung wird mit der Zeile fortgesetzt, die die angegebene Zeilennummer trägt. Kann verwendet werden, um DEF, DO, FOR oder HANDLER-Blöcke zu verlassen, was jedoch nach Möglichkeit vermieden werden sollte.

IF

IF vergleichender Ausdruck THEN Zeilennummer
IF vergleichender Ausdruck THEN einfache Anweisung
Nicht erlaubt sind in einer IF-Zeile die Befehle DATA, DEF, END, DIM, NUMERIC, STRING, ein weiteres IF und alle Anweisungen, die Blöcke einleiten.
IF A >= 3 AND A <= 9 THEN 100
IF A => 3 THEN GOTO 100
If-Zeile
    beliebige Anzahl Befehle oder Blöcke
Else-if-Zeilen (möglich)
    beliebige Anzahl Befehle oder Blöcke
Else-Zeile (möglich)
    beliebige Anzahl Befehle oder Blöcke
End-if-Zeile
If-Zeile:
IF vergleichender Ausdruck THEN
Else-if-Zeile:
ELSE IF vergleichender Ausdruck THEN
Es können beliebig viele ELSE IF-Zeilen verwendet werden.
Else-Zeile:
ELSE
End-if-Zeile:
END IF
IF-Blöcke dürfen jede Anweisung enthalten, die nicht auf den Direktmodus beschränkt ist.

```
IF A < 10 THEN
   PRINT A
ELSE IF A > 30 AND A <= 40 OR A > 50 THEN
   PRINT A + 100
```

ELSE
    PRINT B
END IF
Die ELSE- und ELSE IF-Zeilen können benutzt werden, um einen Block in Unterblöcke zu unterteilen. ELSE kann nur einmal benutzt werden, ELSE IF dagegen so oft wie nötig. Es darf nicht von außerhalb in einen IF-Block hineingesprungen werden.

ON

ON Ausdruck GOTO Zeilennummern-Liste
ON Ausdruck GOSUB Zeilennummern-Liste
Berechnet den Ausdruck, wandelt das Ergebnis in eine ganze Zahl um und verwendet das Ergebnis N um die n-te Zeile der Zeilennummern-Liste auszuwählen (Die Numerierung beginnt mit 1). Die Programmausführung wird dann mit dieser Zeile fortgesetzt. Kein Sprung erfolgt, wenn es keine N-te Zeile in der Liste gibt. Statt dieser Anweisung sollten Sie besser SELECT- oder IF-Blöcke verwenden, um lesbarere Programme zu erhalten.
ON i+2 GOTO 100, 200, 300, 400, 99, 700

RETURN

Rückkehr von einem Unterprogramm, das durch GOSUB aufgerufen wurde.

SELECT

select-Zeile case-Zeile
    beliebige Anzahl Anweisungen oder Blöcke
case-Zeile möglich
    beliebige Anzahl Anweisungen oder Blöcke
end-select-Zeile
select-Zeile:
SELECT CASE Ausdruck
case-Zeile:
CASE Ausdruck
CASE Ausdruck TO Ausdruck
CASE IS vergleichender Ausdruck
CASE ELSE
end-select-Zeile:
END SELECT
Der Select-Block besteht aus einer Anzahl von Anweisungen, um eine Variable oder einen Ausdruck auf verschiedene Bedingungen zu untersuchen.
Das Wort CASE in der SELECT-Zeile kann weggelassen werden, wenn der Ausdruck nicht mit dem Bezeichner CASE beginnt.

Z. B. SELECT CASE CASE + 23
Die Anzahl der CASE-Zeilen ist beliebig. Die Bedingungen werden in der Reihenfolge der Zeilennummern geprüft. Es hat keinen Sinn, nach CASE ELSE weitere CASE-Zeilen einzufügen, da diese normalerweise nicht erreicht werden können. Verschiedene CASE-Angaben können in einer Zeile miteinander kombiniert werden, indem sie durch Komma getrennt werden.

```
Z. B. CASE 1, 2, 3 to 6, 99
SELECT CASE N
CASE 1
   PRINT "Erster Fall"
CASE 2 TO 9, 11, 21
   PRINT "Weitere Fälle"
CASE IS < = A + 20
   PRINT "Noch mehr Fälle"
CASE ELSE
   PRINT "Restliche Fälle"
END SELECT
```

Die CASE ELSE-Zeile kann nur einmal verwendet werden und muß hinter allen anderen CASE-Zeilen stehen. Die übrigen CASE-Zeilen können in beliebiger Reihenfolge verwendet werden. Die Zeilen zwischen zwei CASE-Zeilen stellen einen Block dar. In einen SELECT-Block kann nicht von außerhalb hineingesprungen werden.

String-SELECT ist auch möglich.

THEN Siehe **IF**.

CALL

CALL Funktion CALL Funktion (Parameterliste)
Ruft eine Funktion auf (entweder eine eingebaute Funktion oder eine durch DEF definierte) wenn von der Funktion kein Ergebnis benötigt wird. Jeder Ausdruck, der CALL folgt, wird berechnet und das Ergebnis wird ignoriert. CALL USR(A,B)+USR(C,D) wird daher einfach zwei Maschinencode-Programme aufrufen.
Auch im Direktmodus möglich.

CHAIN

CHAIN Programm-Nummer
CHAIN "Name" (Parameterliste)
Wird verwendet, um BASIC-Programme vom laufenden Programm aus auszuführen.
Parameter können als Werte von einem Programm zum anderen weitergegeben werden, z.B.:
CHAIN „Mein Programm" (1,"FRED")
Siehe auch **PROGRAM**.

DEF

DEF Numerischer Bezeichner = Ausdruck
DEF Numerischer Bezeichner (Parameterliste)
= Ausdruck
DEF String-Bezeichner = String-Ausdruck
DEF String-Bezeichner (Parameterliste) = String Ausdruck
Funktionsdefinition in einer Zeile:
DEF DURCHSCHNITT (x,y) = (x+y)/2
Def-Block: Dieser besteht aus mehreren Anweisungen, die entweder als eine Funktion aufgerufen werden können und ein Ergebnis liefern, oder als Prozedur eine Aufgabe erfüllen. Es gibt einige kleine Unterschiede zur ANSI-Definition. Siehe auch **CALL**, **EXIT DEF**, **NUMERIC** und **STRING**.
Def-Zeile
beliebige Anzahl von Anweisungen oder Blöcken
End-def-Zeile
Def-Zeile:
DEF numerischer Bezeichner
DEF numerischer Bezeichner (Parameterliste)
DEF String-Bezeichner
DEF String-Bezeichner (Parameterliste)
End-def-Zeile:
END DEF
Wenn die Funktion einen Wert liefern soll, muß dieser Wert innerhalb des DEF-Blockes dem Funktionsnamen zugeordnet werden.

```
DEF ANTWORT (A$)
   IF UCASE$(A$(1:1)) = "J" THEN
      ANTWORT = 1
   ELSE IF UCASE$(A$(1:1)) = "N" THEN
      ANTWORT = 0
   ELSE
      ANTWORT = -1
   END IF
END DEF
```

Der Geltungsbereich von Variablen ist dynamisch, d.h. die Variablen hängen an jeder Stelle in einem Programm vom Programmverlauf ab und nicht von der statischen Anordnung des Programms.

```
100 NUMERIC FRED
110 LET FRED = 1
120 CALL Q
130 PRINT FRED
140 END
200 DEF P
210 LET FRED = 123    ! Dieser FRED ist global
220 END DEF
300 DEF Q
310 NUMERIC FRED
320 LET FRED = 0      ! Ein lokaler FRED
330 CALL P
340 PRINT FRED
350 END DEF
```

In diesem Beispiel wird der Bezeichner FRED sowohl für eine globale als auch für eine lokale Variable verwendet. Wenn Zeile 210 ausgeführt wird, wird der FRED von Zeile 310 verändert und nicht der von Zeile 100. Das Programm wird 123 und 1 ausdrucken. In einer Sprache mit statischem Geltungsbereich würde das Programm 0 und 123 ausdrucken. Das kann passieren, wenn dasselbe Programm unter einem Compiler-BASIC läuft.

Alles, was innerhalb eines DEF-Blockes deklariert wird, ist in bezug auf diesen Block lokal und wird nach jeder ersten Ausführung der Deklaration nach dem Aufruf zugewiesen. Alles, was nicht deklariert wird, kann lokal oder global sein, was vom vorherigen Programmverlauf abhängt.

Es ist am besten, wenn man alle Variablen am Anfang jedes Programms und jeder Funktion deklariert, um unerwartete Ergebnisse zu vermeiden.

```
100 CALL P                  ! Dieser Aufruf von P hat I als
                              lokale Variable
110 LET I = 9               ! zu P
120 CALL P                  ! dieser Aufruf von P verändert
                              die globale
130 END                     ! Variable I
200 DEF P
210 LET I = 6
220 END DEF
```

Um konsistente Ergebnisse zu erhalten, sollte die Zeile

```
90 NUMERIC I
```

hinzugefügt werden, so daß I bei beiden Aufrufen von P global ist.

Der Speicher, in dem die lokalen Variablen abgelegt wurden, wird beim Verlassen der Funktion gelöscht. Diese Eigenschaft kann man für eine effiziente Speicherverwendung nutzen – zum Beispiel kann man ein temporäres Datenfeld innerhalb einer Funktion unterbringen.

Fast alles kann als Referenz-Parameter übergeben werden. Normalerweise werden Parameter als Werte übergeben, d.h. es werden „Kopien" dieser Werte an die Funktion übergeben, so daß keine Operationen innerhalb der Funktion einen Einfluß auf die externen Variablen haben. Referenz-Parameter werden dagegen durch Veränderungen innerhalb der Funktion beeinflußt, so daß auch die externen Variablen verändert werden.

```
100 DEF VERTAUSCHEN (REF A, REF B)
110     NUMERIC T
120     LET T = A
130     LET A = B
140     LET B = T
150 END DEF
200 LET X = 99
210 LET Y = 23
220 CALL VERTAUSCHEN (X,Y)
230 PRINT X,Y
```

druckt 23 99

Felder und Funktionen müssen immer durch Referenz übergeben werden.

```
100 NUMERIC A(10)
110 OPTION ANGLE DEGREES
120 DEF P (REF FN,X)
130     PRINT FN(X),
140 END DEF
150 LET A(2) = 66
```

160 CALL P(A,2)
170 CALL P(SIN,30)
druckt 66 .5
Die Übergabe von eingebauten oder benutzerdefinierten Funktionen kann sehr nützlich sein. Eine graphenzeichnende Funktion kann die Funktion, die gezeichnet werden soll, als Parameter erhalten, einer Sortierfunktion können die Austausch- und Vergleichsroutinen als Funktionen übergeben werden.
Funktionen können sich selbst rekursiv aufrufen.

PROGRAM

PROGRAM Name (Variablenliste)
Definiert den Namen des momentanen Programms für die Verwendung in CHAIN-Anweisungen und als Default-Name für SAVE. Der Programmname muß in der Standardform für Bezeichner sein (Siehe ALLGEMEINE REGELN).
PROGRAM "Mein_PROGRAMM"(A,B$)
Die Variablenliste (falls angegeben) ermöglicht es, daß die angegebenen Parameter von einem anderen Programm durch ihren Wert übergeben werden können. Siehe **CHAIN** und **EDIT**.

CAUSE EXCEPTION

CAUSE EXCEPTION Ausdruck
Verursacht einen Fehler und ordnet ihn der Kategorie zu, die durch den Ausdruck bestimmt wird. Eigene Werte sollten zwischen 1 und 999 liegen, da diese nicht vom BASIC verwendet werden.
Das Wort EXCEPTION ist optional.

EXIT HANDLER

Bricht einen Ausnahme-Handler ab und gibt den Ausnahmezustand an die Programmumgebung weiter. Das führt dazu, daß ein weiterer Handler aktiviert wird, entweder ein vom Benutzer definierter Handler oder der Default-Handler.

HANDLER

HANDLER Handler-Name
    Ausnahme-Handler-Anweisungen
END HANDLER
Der Handler-Block wird verwendet, um Ausnahmezustände im Programm, die durch Fehler, den CAUSE EXCEPTION-Befehl oder durch Maschinenunterbrechungen verursacht wurden, zu behandeln.
Der Handler, der verwendet werden soll, wird durch den Handlernamen festgelegt, der im laufenden WHEN-Block angegeben ist.
Siehe **CONTINUE**, **RETRY**, **EXIT HANDLER**, und die Funktionen **EXLINE** und **EXTYPE**.
Die Anweisungen eines Ausnahme-Handlers können nur während einer Ausnahme aufgerufen werden (nicht durch GOTO oder GOSUB).

Falls innerhalb eines Ausnahme-Handlers eine Ausnahme auftritt, hat das einen ähnlichen Effekt wie EXIT HANDLER, da das Programm in der nächstäußeren Stufe des Handlers, die durch die nächstäußere Stufe des WHEN-Blockes festgelegt wird, fortgesetzt wird. Natürlich werden die vorherigen Werte von EXTYPE und EXLINE durch neue Werte ersetzt.

RETRY

Wird als Ausgang aus einem Ausnahme-Handler verwendet. Dieser Befehl bewirkt, daß das Programm mit der Anweisung fortgesetzt wird, die den Ausnahmezustand verursacht hat (vgl. **CONTINUE**).
Wenn ein Ausnahme-Handler verwendet wird, um das Drücken der 'STOP'-Taste abzufangen, sollte RETRY verwendet werden, um das Programm fortzusetzen.

WHEN

WHEN EXCEPTION USE Handler-Name
    Anweisungen
END WHEN
Gibt den Ausnahme-Handler an, der verwendet werden soll, wenn innerhalb des WHEN-Blockes ein Ausnahmezustand (engl. Exception) entsteht.
Die Programm-Anweisungen können weitere verschachtelte WHEN-Blöcke enthalten.
Siehe **HANDLER**.

OPTIONEN

INTERRUPT

ASK INTERRUPT CODE
Fragt den Software-Interrupt-Code des letzten Interrupts ab.
SET INTERRUPT KEY ON/OFF
Wenn eingeschaltet, wird durch jeden Tastendruck ein Software-Interrupt verursacht. Kann auch in Verbindung mit TOGGLE verwendet werden.
SET INTERRUPT NET ON/OFF
Schaltet den Software-Interrupt ein/aus, der verursacht wird, wenn Daten vom Netzwerk empfangen werden.
SET INTERRUPT STOP ON/OFF
Schaltet den Software-Interrupt ein/aus, der durch die 'STOP'-Taste verursacht wird. Kann auch in Verbindung mit TOGGLE verwendet werden.

TIMER

SET TIMER Ausdruck
Startet einen Timer, der einen Software-Interrupt verursacht, sobald er zu 0 heruntergezählt ist. Der Ausdruck gibt den Betrag der Zeit, die der Timer laufen soll, in Sekunden an. Der Maximalwert ist 255. Wird der Wert auf 0 gesetzt, so hält das den Timer an, ohne daß ein Interrupt verursacht wird.
Der Timer hält grundsätzlich an, sobald 0 erreicht ist.
Der Software-Interrupt-Ausnahmezustand (EXTYPE 9229) muß durch einen Ausnahme-Handler behandelt werden, wenn das Programm nicht abgebrochen werden soll. Nach dem Ausnahmezustand weist ASK INTERRUPT CODE A der Variablen A den Wert 64 zu, falls der Interrupt durch den Timer verursacht wurde.

VARIABLEN

EXLINE

Liefert die Zeilennummer der letzten Anweisung, die einen Ausnahmezustand (Exception) verursacht hat.

EXSTRING$(N)

Liefert den Mitteilungs-String, der zum Ausnahmefall N gehört. Beachten Sie, daß dieser String mit einem Leerzeichen beginnt.

EXTYPE Liefert die Kennummer des letzten Ausnahmefalls.

DATA

DATA Datenliste
Ermöglicht es, eine Liste von Konstanten, Zahlen und/oder Strings in ein Programm einzubauen und diese dann nacheinander zu lesen. Siehe auch READ.

GET

GET #Kanal:String-Bezeichner
Holt ein einzelnes Zeichen von einem Kanal und liefert einen Null-String ("") wenn kein Zeichen verfügbar ist.
Default ist Kanal 105 (KEYBOARD:), und bei einfachen Anwendungen ist GET der Funktion INKEY$ ähnlich.

IMAGE

IMAGE: Format-Angabe
Wird verwendet in Verbindung mit PRINT Anweisungen, um das Format für die Ausgabe festzulegen. Das Format wird durch einen String bestimmt, der Zeichen enthält, die in diesem Zusammenhang die folgenden Bedeutungen haben.
Zahlen-Formatier-Zeichen:

, – druckt ein Komma in der Zahl.
$ – druckt ein Dollarzeichen vor das Vorzeichen der Zahl.
– – druckt ein Leerzeichen oder '-'-Zeichen.
+ – druckt ein '+' oder '-'-Zeichen.
% – druckt eine Zahl mit führenden Nullen.
# – druckt eine Zahl oder ein Leerzeichen mit angehängten Nullen nach einem Dezimalpunkt.
* – druckt eine Zahl oder führende '*'.
. – druckt einen Dezimalpunkt.
^ – druckt den Exponenten (mindestens 4 Zeichen).

Wenn die Zahl nicht in den Format-Platz paßt, verursacht dies einen Fehler.

String-Formatier-Zeichen:
< Links-Justierung eines Strings in dem Feld, das durch #-Zeichen definiert wird.
druckt ein Zeichen.
> Rechts-Justierung eines Strings.
Das Formatzeichen für Justierung muß am Anfang des Feldes stehen; wenn kein Justierzeichen verwendet wird, wird der String zentriert.
Das Format in der IMAGE-Zeile beginnt direkt nach dem ":" und endet mit dem letzten Zeichen der Zeile.

INPUT

INPUT #Kanal, IF MISSING Aktion, AT Zeilen-Nummer, Spalten-Nummer, PROMPT String: Variablenliste

Liest Daten vom angegebenen Kanal in eine Liste von Variablen. Default-Kanal ist der Editor (Kanal 0). Sind mehrere Eingaben erforderlich, so müssen sie durch Komma getrennt werden.
INPUT PROMPT K$&"Geben Sie bitte die nächste Zahl ein ":N
INPUT A(I),B$
Die IF MISSING- und PROMPT-Teile können in beliebiger Reihenfolge stehen oder ganz fehlen. Der Default-Prompt (Eingabeaufforderung) ist "?". PROMPT ersetzt den Default-Prompt durch den String. Die AT-Option (mit Zeilen-Nummer und Spalten-Nummer) ist unabhängig von der PROMPT-Option.
Die IF MISSING-Aktion wird ausgeführt, wenn auf dem Kanal eine Dateiende-Bedingung auftritt oder eine leere Eingabe an einer Stelle erfolgt, an der eine numerische Eingabe erwartet wird. Diese Aktion folgt denselben Regeln wie bei READ.
Siehe auch **LINE INPUT**.

LINE INPUT

Ähnlich dem INPUT-Befehl, aber liest eine ganze Zeile (auch Komma etc.) für jeden Wert in der Variablenliste (die nur Stringvariablen enthalten darf).

LPRINT

LPRINT Print-Ausdruck
Genau wie PRINT, Default-Kanal ist hier jedoch #104, der Drucker-Listing-Kanal.

OPEN

OPEN #Kanal:NAME Gerät/Dateiname ACCESS Modus
OPEN #Kanal:Gerät/Dateiname
Der Zugriffs-Modus ist entweder INPUT oder OUTPUT. ACCESS OUTPUT versucht, eine neue Datei anzulegen (im Falle von Cassette oder Disk); ACCESS INPUT versucht eine bestehende Datei zu benutzen. Für Geräte wie VIDEO: kann beides benutzt werden. Default ist INPUT.
OPEN verbindet ein Gerät, oder eine Datei im Falle von Cassette oder Disk, mit einem Kanal. Befehle können dann Daten von und zu dem Gerät (oder der Datei) lesen, schreiben oder in anderer Weise bearbeiten, indem sie auf den Kanal mit Hilfe der Kanalnummer zugreifen.
OPEN #8: "DISK-1: TEST_PROGRAMM" ACCESS OUTPUT
Einem bestimmten Kanal darf nur ein Gerät (oder eine Datei) auf einmal zugeordnet werden. Es können aller-

dings verschiedene Geräte (Dateien) nacheinander einem Kanal zugeordnet werden.
Um einen Kanal von einem Gerät (einer Datei) zu trennen wird der CLOSE-Befehl verwendet.
Kanalnummern dürfen im Bereich von 0 bis 254 sein (255 ist eine ungültige Kanalnummer, die für spezielle Zwecke verwendet wird.
Das BASIC-System verwendet verschiedene Kanäle als Defaults wenn in Anweisungen kein Kanal angegeben ist. Diese Kanäle sind:
0 – Wird für Kommando-Eingabe und normale Text-Ausgabe (z.B. LIST und PRINT) verwendet. Dieser Kanal ist nach Reset (oder nach dem Einschalten) mit dem Gerät "EDITOR:" verbunden.
Das Gerät "EDITOR:" selbst verwendet die Geräte "KEYBOARD:" und "VIDEO:", eingestellt auf Bildschirmmodus 0 mit der Seitengröße 24,40.
Dieser Kanal wird als Default angenommen für COPY FROM und REDIRECT FROM. Kanal 0 wird bei Reset automatisch geöffnet und bleibt offen, bis er explizit geschlossen wird.
Beachten Sie, daß Kanal 0 als Default-Kommandokanal für ANSI-Kompatibilität festgelegt ist. Alle anderen Default-Kanäle haben Nummern über 100, um dem Benutzer einfache Kanalnummern für eigene Definitionen zu lassen.
101 – Graphik Ein- und Ausgabeanweisungen. Dieser Kanal ist bei der ersten Benutzung des GRAPHICS Befehls mit dem Gerät „VIDEO:" verbunden, das auf Bildschirmmodus 1 und Farbmodus 1 mit der Seitengröße 20,40 eingestellt ist.
Kanal 101 bleibt offen, bis er explizit geschlossen wird (z.B. durch einen TEXT-Befehl).
102 – Die Standard-Textseite (Seitengröße 24,40). Dieser Kanal wird bei Reset automatisch geöffnet.
103 – Standard-Tonausgabe. Dieser Kanal wird bei Reset mit dem Gerät „SOUND:" verbunden.
Kanal 103 wird bei Reset automatisch geöffnet und bleibt offen, bis er explizit geschlossen wird.
104 – Wird für hard-copy Anweisungen verwendet. Dieser Kanal wird bei Reset mit dem Gerät „PRINTER:" verbunden. Er wird als Default für COPY TO und REDIRECT TO angenommen.
Kanal 104 wird bei Reset automatisch geöffnet und bleibt offen, bis er explizit geschlossen wird.

105 – Tastatur-Operationen (wird bei Reset mit dem Gerät "KEYBOARD:" verbunden). Bleibt offen, bis er explizit geschlossen wird.
106 – Wird für auf Dateien basierende Ein- und Ausgabe-Anweisungen verwendet. Dieser Kanal wird mit „DISK-1:" verbunden, falls ein Diskettenlaufwerk angeschlossen ist. Ist keine Diskettenstation angeschlossen, so wird der Kanal mit "TAPE:" verbunden.
Standardbefehle sind LOAD, MERGE und VERIFY.
Kanal 106 wird nur geöffnet wenn dies notwendig ist, und wird geschlossen, wenn die Anweisungen beendet sind – wenn er nicht mit OPEN explizit geöffnet wurde.
107 – Netzwerk-Operationen. Dieser Kanal ist der Default-Kanal für SET CAPTURE FROM.
Kanal 107 wird nur automatisch geöffnet, wenn ein Befehl ausgeführt wird, der diesen Kanal als Default annimmt. Er wird am Ende der Operation geschlossen.
Die Kanäle 100 bis 254 bleiben offen, bis sie ausdrücklich geschlossen werden, die Kanäle 0 bis 99 sind dagegen immer geschlossen wenn RUN eingegeben oder irgendeine andere Operation ausgeführt wird, die alle Variablen löscht. Wenn BASIC einen Default-Kanal geschlossen vorfindet, so schließt es alle Kanäle (0 bis 254) und versucht, die Default-Kanäle wiederzueröffnen. Wenn dies nicht möglich ist, nimmt der Computer an, daß ein nicht behebbarer Fehler aufgetreten ist und läßt den Bildschirmrand blinken, bis ein Reset erfolgt.
Gerätenamen, die an das Betriebssystem weitergegeben werden, erhalten einen Doppelpunkt, damit sie erkannt werden können. Wo mehr als ein Gerät unter demselben Namen bekannt ist, wird eine Zahl an den Namen angehängt (z.B. "DISK-2:").
Die gültigen Namen sind:

"DISK-n:"	Diskettenlaufwerke
"EDITOR:"	Bildschirmeditor. Dieser wiederum verwendet die Geräte "VIDEO:" und "KEYBOARD:".
"KEYBOARD:"	Tastatur inklusive externe Joysticks.
"NET-n:"	Eingebautes lokales Netz.

Die Zahl „n" ist die Netzwerk-Adresse (im Bereich 1-32) des Computers, mit dem Verbindung aufgenommen werden soll. Wenn „n" 0 ist, wird dadurch ein „allgemeiner" Kanal definiert, der für das Senden an alle Computer oder das Empfangen ohne Festlegung des sendenden Computers verwendet wird.

"PRINTER:" 'Centronics'-artige Druckerschnittstelle.
"SERIAL:" Serielle RS423 Ein /Ausgabeschnittstelle.
"SOUND:" Tongenerator.
"TAPE-n:" Cassettenrekorder.
"VIDEO:" Bildschirmseiten.

Wenn andere Geräte an den Computer angeschlossen werden, erhalten sie weitere Namen im Betriebssystem.
In den meisten Fällen wird nur ein Gerätename für eine OPEN-Operation benötigt. Für Ein- und Ausgabeoperationen auf Dateibasis ist ein Dateiname erforderlich. Die vollständige Beschreibung eines Dateinamens lautet:
"Gerät-N:Name"
„Gerät" ist optional, und wenn diese Angabe fehlt, wird der Default-Systemmassenspeicher verwendet. Bei einem nicht ausgebauten System ist dies "TAPE:".
„n" ist die Gerätenummer, und die Defaultnummer ist hier 1 (z.B. "DISK:" entspricht "DISK-1:".
"Name" ist der Bezeichner der Datei innerhalb eines Gerätes. Er folgt denselben Regeln wie ein BASIC-Bezeichner mit dem Unterschied, daß nur die ersten 28 Zeichen signifikant sind.
Falls kein Doppelpunkt im Dateinamen enthalten ist, wird angenommen, daß der Gerätename weggelassen worden ist. Also ist "SOUND" eine Datei auf TAPE:, aber "SOUND:" ist das Gerät Tongenerator.
Zum Beispiel beziehen sich die Dateinamen "DISK1:DATEN", "DISK-1:DATEN", "1:DATEN" bzw. "DATEN" alle auf die selbe Datei.
Der "Name"-Teil eines Dateinamens wird bei allen im Augenblick definierten Geräten außer "TAPE:" und "DISK:" ignoriert. So ist zum Beispiel "PRINTER:LISTING" äquivalent zu "PRINTER:".
Es gibt einige Befehle, die es erlauben, sowohl den Kanal als auch den Dateinamen in einer Anweisung anzugeben (z.B. LOAD und SAVE). Die vollständige Spezifikation hat in diesen Fällen die Form:
#Kanal:Dateiname
Wenn #Kanal fehlt, wird ein Default-Kanal verwendet.

PRINT

PRINT #Kanal, AT Zeilen-Ausdruck, Spalten-Ausdruck: Ausgabe-Liste
PRINT #Kanal, USING Zeilennummer: Ausgabe-Liste
PRINT #Kanal, USING String: Ausgabe-Liste
Elemente der Ausgabe-Liste können entweder Ausdrücke sein oder das Wort TAB, gefolgt von einer Spaltennummer

in Klammern. Die einzelnen Elemente können durch Komma oder Semikolon getrennt werden. Ein Semikolon erzeugt einen leeren String. Ein Komma erzeugt einen Zwischenraum bis zum nächsten Druckbereich. TAB erzeugt einen Zwischenraum bis zur angegebenen Spalte. Eine Ausgabezeile, die mit einem Komma oder einem Semikolon endet, erzeugt keinen Sprung in die nächste Zeile. PRINT kann auch im Direktmodus ausgeführt werden.
Die AT-Angabe positioniert den Cursor auf die angegebene Zeile und Spalte bevor die Liste gedruckt wird. Die (optionale) Kanalnummer ändert den Ausgabe-Kanal (Default-Kanal ist die Standard-Textseite).
Die Zeilen- und Spaltenangaben für AT entsprechen den Regeln für Text-Positionierung. Die linke obere Ecke der Bildschirmseite hat die Text Koordinaten (1,1). Die 15. Spalte in der 2. Zeile hat die Text-Koordinaten (2,15).
PRINT "Wert = ";A
PRINT AT x,y:"o"
Die USING-Angabe kann verwendet werden, um das Format für die Ausgabe festzulegen. Die Zeilennummer muß die Nummer einer IMAGE-Zeile sein. Siehe IMAGE bezüglich Einzelheiten der Format-Angabe.

READ

READ Variablenliste
READ IF MISSING Zeilennummer: Variablenliste
READ IF MISSING EXIT DO: Variablenliste
Liest Daten aus den DATA-Zeilen. Die IF MISSING Anweisung wird ausgeführt, wenn versucht wird, Daten über das Ende der Datenliste hinaus zu lesen.
READ A,B$(i)

RESTORE

RESTORE
RESTORE Zeilennummer
Setzt den Anfang von DATA für READ-Anweisungen auf den Programmanfang oder die angegebene Zeilennummer.

CAPTURE

CAPTURE FROM #Kanal TO #Kanal
Nimmt Eingaben vom erstgenannten Kanal entgegen, die dann Eingaben, die vom zweitgenannten Kanal erwartet werden, ersetzen. Eingaben vom zweiten Kanal werden gesperrt bis 'STOP' gedrückt wird, eine Datei-Ende-Bedingung auf dem ersten Kanal oder ein Fehler auftritt. CAPTURE FROM von einem einzelnen Kanal kann auch beendet werden, indem in einer folgenden Anweisung #255 (normalerweise nicht gültig) als TO-Kanal angegeben wird.

CLEAR

CLEAR #Kanal
CLEAR ENVELOPE
CLEAR FKEYS
CLEAR FONT
CLEAR GRAPHICS
CLEAR #Kanal:NET
CLEAR QUEUE Tonquellennummer
CLEAR SCREEN
CLEAR SOUND
CLEAR TEXT
Löscht verschiedene Optionen. Kann auch im Direktmodus ausgeführt werden.

CLOSE

CLOSE #Kanal
Sendet alle Daten in den Ausgabe-Buffern ab, schließt den Kanal und gibt die Buffer frei.

COPY

COPY FROM #Kanal TO #Kanal
Kopiert den Inhalt eines Kanals in einen anderen Kanal (beide Kanäle müssen offen sein). Das Kopieren wird durch ein Dateiende, einen Fehler, oder die ‚STOP'-Taste beendet. Der Default-Eingabekanal ist Kanal 0, der Default-Ausgabekanal ist 104.

COPY

kopiert von #0 nach #104
COPY FROM #5
kopiert von #5 nach #104. Erfordert, daß Kanal 5 geöffnet ist.

FLUSH

FLUSH #Kanal
Erzwingt das Senden der in einem Kanal-Buffer noch verbliebenen Daten, ohne den Kanal zu schließen oder

ein Dateiende zu signalisieren. Diese Operation ist nur bei bestimmten Geräten (z.B. NET:) sinnvoll. Kann auch im Direktmodus verwendet werden.

REDIRECT

REDIRECT FROM #Kanal TO #Kanal
Liest Eingaben vom erstgenannten Kanal und leitet diese in den zweiten, bis das Dateiende erreicht ist, die ‚STOP'-Taste gedrückt wird, oder bei einem der Kanäle ein Fehler auftritt. Die Umleitung kann auch dadurch beendet werden, daß in einer späteren REDIRECT-Anweisung die ungültige Kanalnummer #255 als TO-Kanal angegeben wird.

OPTIONEN

DEFAULT CHANNEL

SET DEFAULT CHANNEL Ausdruck
Bestimmt den Default-Systemkanal. Dieser Kanal-Wert kann dann von Service-Programmen verwendet werden, die mit dem Benutzer in Verbindung treten wollen, aber die Funktion der momentan verwendeten Kanäle nicht kennen.
Insbesondere wird dieser Kanal von Programmen verwendet werden, die auf die HELP-Anweisung antworten, die über das Betriebssystem gesendet wurde.
Solange er nicht durch SET neu definiert wird, hat dieser Kanal die Nummer 0.

LOAD

LOAD #Kanal:Dateiname
LOAD Gerätename
Lädt eine Datei vom angegebenen Kanal oder, wenn kein Kanal angegeben wurde, von Kanal 106 (Cassette oder, wenn Diskettenlaufwerke angeschlossen sind, Diskette). Wenn bei LOAD keine Parameter angegeben sind, wird als Default die „Boot-Datei" verwendet (bei Cassette ist dies die erste gefundene Datei).
LOAD
LOAD "MEIN_PROGRAMM"
LOAD "NET-0:"
Wenn die Datei ein BASIC-Programm enthält, ersetzt es das momentan im Speicher befindliche Programm. Falls mehrere BASIC-Programme als eine Datei abgespeichert wurden, ersetzen diese alle im Speicher befindlichen Programme und kehren zu Programm 0 zurück.
Die Datei kann andere Daten und Programmtypen (wie z.B. Erweiterungen oder andere Anwender-Programme) enthalten, die vom Betriebssystem automatisch verwaltet werden.
Definition von Gerätenamen und Dateinamen siehe **OPEN**.
Löscht alle Variablen. Nur im Direktmodus möglich (**RUN** kann auch in Programm-Anweisungen verwendet werden um Programme zu laden und zu starten).

MERGE

MERGE #Kanal:Dateiname
Fügt ein Programm von Diskette, Cassette oder einem anderen Kanal mit dem im Speicher befindlichen Programm zusammen. Bei Überschneidungen der Zeilennummern ersetzen Zeilen des neuen Programms die des alten. Nur im Direktmodus möglich. Löscht alle Variablen.

SAVE

SAVE #Kanal:Dateiname
SAVE Gerätename
SAVE ALL #Kanal:Dateiname
Speichert das momentane Programm ab. Der Default-Kanal hierfür ist 106.
Wenn kein Dateiname angegeben ist, wird der **PROGRAM**-Name verwendet, falls ein solcher existiert.
SAVE ALL speichert alle momentan im Speicher befindlichen Programme ab.
Programme werden in codiertem Format abgespeichert. Um ein Programm als Text (ASCII) abzuspeichern, kann man LIST #Kanal:Dateiname verwenden – solche Pro-

gramme können später geladen werden, wenn dies erforderlich ist.
Nur im Direktmodus möglich. Siehe LOAD.

VERIFY

VERIFY #Kanal:Dateiname
Überprüft, ob ein Programm korrekt abgespeichert wurde. Vergleicht das momentane Programm mit der angegebenen Datei und bewirkt eine Fehlermeldung, wenn die beiden Dateien nicht übereinstimmen. Kanal 106 ist der Default-Kanal. Nur im Direktmodus möglich.

OPTIONEN

FAST SAVE

SET FAST SAVE ON/OFF
Schaltet die hohe Geschwindigkeit für Cassetten-Operationen ein/aus. Diese Geschwindigkeit ist etwa 2400 Baud und wird als Default angenommen. Wenn FAST SAVE ausgeschaltet ist, wird die Geschwindigkeit halbiert.
Beim Laden von Cassette wird die richtige Geschwindigkeit automatisch eingestellt.
Kann in Verbindung mit TOGGLE verwendet werden.

REM1

SET REM1 ON/OFF
Schaltet Fernsteuerungs-Schalter 1 ein/aus. (Dieser Schalter wird auch durch Cassetten-Operationen beeinflußt).

REM2

SET REM2 ON/OFF
Schaltet Fernsteuerungs-Schalter 2 ein/aus. (Dieser Schalter wird auch durch Cassetten Operationen beeinflußt).

TAPE LEVEL

SET TAPE LEVEL Ausdruck
Bestimmt die Lautstärke-Einstellung für das Speichern auf Cassette. Annehmbare Werte liegen zwischen 1 und 6, wobei die Lautstärke von Wert zu Wert verdoppelt wird. Der Wert 1 entspricht etwa 40 mV Amplitude. Der Default-Wert ist 2.

TAPE SOUND

SET TAPE SOUND ON/OFF
Kontrolliert die Übertragung von Tönen vom Cassetteneingang zur Tonausgabe. Erlaubt direkten Durchgang von Musik oder Sprache von Cassette zum eingebauten Lautsprecher oder zur Hi-Fi-Ausgabe. Kann auch in Verbindung mit TOGGLE verwendet werden.

GRAPHICS

GRAPHICS
GRAPHICS HIRES/LORES Anzahl Farben
GRAPHICS ATTRIBUTE
Der Befehl GRAPHICS bewirkt, daß die Default Graphikseite und die Default-Textseite (#101 und #102) geschlossen und neu geöffnet werden und daß die Default-Graphikseite auf dem Bildschirm angezeigt wird, wobei die unteren vier Zeilen für Textdarstellung reserviert sind.
GRAPHICS setzt außerdem den Default-Kanal (101) für Video-Maschinenoptionen wie z.B. PALETTE fest.
Gültige Werte für die Anzahl Farben sind 2, 4, 16 und 256. Wenn bei der HIRES/LORES-Option keine Farbenanzahl angegeben ist, wird der Wert des vorherigen GRAPHICS-Befehls verwendet, bei der ersten Verwendung wird eine hochauflösende Graphikseite mit 4 Farben eingerichtet.
Erläuterungen zu GRAPHICS ATTRIBUTE siehe **ATTRIBUTES**-Videooption.

LOOK

LOOK #Kanal AT x,y: Variable
Weist der Variablen die Farbe am Punkt (x,y) auf der momentanen Graphik-Seite bzw. auf der durch die Kanalnummer angegebenen Seite zu. Sowohl die Kanalnummer als auch der AT-Teil sind optional. Wenn der AT-Teil weggelassen wird, wird die momentane Position des Graphikstrahls verwendet.
Beachten Sie, daß die Verwendung von AT den Graphikstrahl ausschaltet und nach (x,y) bewegt.

PLOT

PLOT #Kanal:Punkte-Liste
PLOT #Kanal:ANGLE Ausdruck
PLOT #Kanal:FORWARD/BACK Ausdruck
PLOT #Kanal:LEFT/RIGHT Ausdruck
PLOT #Kanal:ELLIPSE Ausdruck, Ausdruck
PLOT #Kanal:PAINT
Wenn der PLOT-Befehl von einer Punkte-Liste gefolgt wird, zeichnet er Punkte und Linien. Wenn ein PLOT-Befehl mit einem Semikolon endet, wird der Beam (Graphikstrahl) angelassen, nachdem der Befehl ausgeführt worden ist, andernfalls wird er ausgeschaltet. Also:
PLOT x,y
bewegt den Beam an die Position (x,y) (und zeichnet dabei eine Linie, falls der Beam eingeschaltet war) und schaltet den Beam dann aus.
PLOT x,y;

läßt den Beam an. Beide Anweisungen zeichnen einen Punkt bei (x,y). Wenn dem Koordinatanpaar ein Komma folgt, wird der Beam an die angegebene Position bewegt ohne dort einen Punkt zu zeichnen (und wird ausgelassen).

PLOT x1,y1;x2,y2;...

zeichnet mit angeschaltetem Beam Linien zwischen den angegebenen Punkten und läßt den Beam an, wenn die Anweisung mit einem Semikolon endet. Außerdem wird eine Linie zwischen der Position, an der der Beam vor der Anweisung stand, und dem Punkt (x1,y1) gezogen, wenn der Beam eingeschaltet war.

Es wird in der momentan eingestellten Farbe, der momentanen Linienart und dem momentanen Linienmodus gezeichnet (siehe dazu den Abschnitt über Videoptionen).

Die Koordinaten, die in PLOT-Anweisungen verwendet werden, entsprechen den Vereinbarungen für Graphik-Positionen. Die linke untere Ecke der Bildschirmseite ist (0,0). Bei der Koordinatenangabe (x,y) ist x die horizontale Position von links gezählt und y die vertikale Position von unten gezählt.

ELLIPSE zeichnet eine Ellipse mit dem Mittelpunkt in der momentanen Beam-Position. Die beiden hinter ELLIPSE stehenden Parameter geben den horizontalen und den vertikalen Abstand vom Mittelpunkt zur Peripherie der Ellipse in Graphik-Bildschirmpositionen an. Die Ellipse muß mit ausgeschaltetem Beam gezeichnet werden, wenn kein Punkt im Mittelpunkt erscheinen soll.

PLOT 300,350, ELLIPSE 200,300,

zeichnet eine Ellipse ohne Punkt in der Mitte.

PAINT füllt eine geschlossene Fläche (in der der Beam steht) mit der momentanen INK-Farbe. Die ausgefüllte Fläche ist von einer ununterbrochenen Linie umrundet, deren Farben sich von der ursprünglichen Farbe der Beam-Position unterscheiden.

Falls der Beam auf einer Position ist, auf der ein Punkt der momentanen INK-Farbe gezeichnet wurde, hat PAINT keinen Effekt, da sofort die Bedingung einer ausgefüllten umrandeten Fläche erfüllt ist. Wie bei ELLIPSE sollten Vorsichtsmaßnahmen getroffen werden, um das Zeichnen eines Punktes zu vermeiden.

Z.B.: PLOT 400, 300, PAINT

Der Default-Kanal für PLOT ist 101.

Die Videooptionen arbeiten mit dem eingebauten Video-Prozessor, der viele Videoseiten mit jeweils verschiedenen Parametern handhaben kann. Die Befehle, die auf einzelne Seiten wirken, können eine Kanal-Angabe enthalten. Ansonsten ist der Default-Kanal in einigen Fällen die Standard-Textseite (#102) und in anderen Fällen die Standard-Graphikseite (#101), wie im folgenden angegeben.
Beachten Sie, daß anstelle von COLOUR immer COLOR verwendet werden darf.

ATTRIBUTES

SET ATTRIBUTES Ausdruck
Stellt bestimmte Schalter ein, um im Attribut Videomodus (Nummer 15) bestimmte Operationen zu steuern. Die Werte dieser Schalter haben die folgenden Bedeutungen:

1 – Zeichnen in der Hintergrundfarbe
2 – Zeichnen ohne Veränderung der Einzelpunkt-Daten
4 – Zeichnen ohne Veränderung der Vordergrund-Attribute
8 – Zeichnen ohne Veränderung der Hintergrund-Attribute
16 – Schreiben in der Hintergrundfarbe
32 – Schreiben ohne Veränderung der Einzelpunkt-Daten
64 – Schreiben ohne Veränderung der Vordergrund-Attribute
128 – Schreiben ohne Veränderung der Hintergrund-Attribute

Um eine Kombination mehrerer Effekte zu erreichen, müssen die Zahlen addiert werden. Der Default-Wert ist 0.

BEAM

SET #Kanal:BEAM ON/OFF
Die momentane Graphik-Zeichenposition wird die 'Beam'-Position genannt. Jedesmal wenn der Beam (Graphikstrahl) bewegt wird, hinterläßt er eine Spur oder auch nicht (je nachdem, ob er ein- oder ausgeschaltet ist). Die Default-Kanalnummer ist #101.

BIAS

SET #Kanal:BIAS Farbcode
Bestimmt, welche Gruppe von Farben als Nummern 8 bis 15 auf der Palette verwendet werden. Die in der Anweisung angegebene Nummer ist die Standard-Codenummer irgendeiner Farbe in der Gruppe. Es gibt 32 effektive Werte. Der Farbcode kann auch mit Hilfe der RGB-Funktion bestimmt werden.

SET BIAS RGB (0,.6,.4)
Die Default-Kanalnummer ist #101. Die BIAS Einstellung gilt für alle dargestellten Paletten.

BORDER

SET #Kanal:BORDER Farbcode.
Bewirkt, daß der Bildschirmrand in der Farbe dargestellt wird, die dem angegebenen Standard-Farbcode entspricht. Die Default-Kanalnummer ist #101.

CHARACTER

SET #Kanal:CHARACTER n,a,b,c,d,e,f,g,h,i
Definiert das Aussehen des Zeichens mit dem ASCII Code n. Jeder der Parameter a-i bestimmt eine Zeile des Zeichens n (von oben angefangen).
Um die Definition von Zeichen zu erleichtern, kann die BIN-Funktion verwendet werden, um jedes Pixel einer Reihe als 0 oder 1 darzustellen.
Auch wenn die Kanalnummer angegeben wird, betrifft die Anweisung alle Videoseiten. Der Default-Kanal ist #102.
Um alle Zeichen auf ihre Default-Einstellungen zurückzuschalten, kann CLEAR FONT verwendet werden.

COLOUR

SET #Kanal:COLOUR Paletten-Nummer, Farbcode
Stellt den Wert einer Farbe der Palette ein (siehe **PALETTE**). Paletten-Nummern sind aus dem Bereich 0 bis 7. Der Farbcode ist aus dem Standard-Bereich 0 bis 255 (oder wird mit Hilfe der RGB-Funktion angegeben).

*** FARBEN**

BLACK
Die Farbe Schwarz, entspricht RGB(0,0,0).
BLUE
Die Farbe Blau, entspricht RGB(0,0,1).
CYAN
Die Farbe Zyan, entspricht RGB(0,1,1).
GREEN
Die Farbe Grün, entspricht RGB(0,1,0).
MAGENTA
Die Farbe Magenta, entspricht RGB(1,0,1).
RED
Die Farbe Rot, entspricht RGB(1,0,0).
RGB(R,G,B)
Liefert die Maschinen-abhängige Farbnummer, die der angegebenen Mischung aus Rot, Grün und Blau entspricht. R gibt den Rot-Anteil (0 bis 1) an, G den Grün-Anteil (0 bis 1) und B den Blau-Anteil (0 bis 1) an.

Beispiel: SET INK RGB(1/2, 1/3, 1/4).

WHITE
Die Farbe Weiß, entspricht RGB(1,1,1).

YELLOW
Die Farbe Gelb, entspricht RGB(1,1,0).

CURSOR

SET #Kanal:CURSOR CHARACTER-Code
SET #Kanal:CURSOR COLOUR Paletten-Nummer
Gibt den ASCII-Code des Cursors und/oder die Palettennummer der Farbe, die für den Cursor verwendet werden soll, an. Die Default-Kanalnummer ist #102.

INK

SET #Kanal:INK Farben-Nummer
Stellt die Farbe ein, die für folgendes Zeichnen verwendet wird. Die Farben-Nummer ist eine Paletten-Nummer – außer in Farbmodus 3 (256 Farben), wo es eine Standard-Code-Nummer ist. Die Default-Kanalnummer ist #101.

LINE MODE

SET #Kanal:LINE MODE Parameter
Bestimmt die Wechselwirkung zwischen den Farben des bestehenden Bildschirminhaltes mit den neuen Linien, die gezeichnet werden. Im Modus 0 (dem Default-Modus) überschreibt eine neue Linie alles, was vorher dort stand. In den Modi 1 bis 3 wird die Farbe, die für jeden Teil der neuen Linie verwendet wird, dadurch bestimmt, daß die Palettennummern der alten und der neuen INK-Farbe in der folgenden Weise kombiniert werden:
Modus 1 – ‚Oder'
Modus 2 – ‚Und'
Modus 3 – ‚Exklusiv-Oder'
Die Default-Kanalnummer ist #101.

LINE STYLE

SET #Kanal:LINE STYLE Parameter
Die momentane Linienart kann auf jeden beliebigen Wert im Bereich von 1 bis 14 eingestellt werden, so daß verschiedene Arten von unterbrochenen Linien gezeichnet werden können. Die Default-Kanalnummer ist #101.

PALETTE

SET #Kanal:PALETTE a,b,c,d,e,f,g,h
Definiert die Werte der ersten acht Farben der Palette, die dann durch Videooptionen wie SET PAPER und SET INK verwendet werden. Die Default-Kanalnummer ist #101.
Nur die ersten 4 Farben können in Farb-Modus 1 verwendet werden, in Farb-Modus 0 nur die ersten 2. Wenn nur

die ersten 2 oder 4 Farben angegeben werden, ist 0 die Default-Farbe für den Rest.
Die Farben für die benutzerdefinierbare Palette werden durch Standard-Code-Nummern im Bereich von 0 bis 255 oder durch die RGB-Funktion angegeben (siehe ‚Eingebaute Funktionen und Variablen'). Die ‚Teletext primary'-Farben können durch ihren Namen angegeben werden (z.B. MAGENTA).
Die Palette enthält insgesamt 16 Farben, allerdings können nur die ersten 8 frei gewählt werden. Siehe BIAS-Option für Einzelheiten über die übrigen 8 Farben.

PAPER

SET PAPER #Kanal:Farben-Nummer
Wählt die Hintergrundfarbe aus, die für Drucken und Zeichnen verwendet wird. In Farb-Modus 3 wird die PAPER-Farbe durch eine Standard-Codenummer angegeben; in anderen Modi durch eine Paletten-Nummer. Die Default-Kanalnummer ist #101.
Für Graphik-Videoseiten (Modi 1 und 5 – siehe **VIDEO MODE**-Option) hat der PAPER-Befehl nur dann eine Bedeutung, wenn die Seite gelöscht wird und eine neue Hintergrundfarbe gewählt wurde.
Bei einer 80-Zeichen-Textseite (Videomodus 2) sind die Paletten-Farben 0, 2, 4 und 6 gültige Hintergrundfarben. Zu diesen gehören jeweils die INK-Farben 1, 3, 5 bzw. 7; ein Zeichen, das in einer bestimmten INK-Farbe abgebildet wird, erhält automatisch die zugehörige PAPER-Farbe als eigenen Hintergrund. Ein Farben-Paar für Vordergrund- und Hintergrund-Farbe wird durch SET PAPER oder SET INK, gefolgt von einer der beiden entsprechenden Paletten-Farben, ausgewählt. Bei einer 40-Zeichen Textseite (Videomodus 0) ist es ähnlich, mit dem Unterschied, daß es dort nur 2 verfügbare Farbenpaare gibt.

SCROLL

SET #Kanal:SCROLL ON/OFF
Schaltet das automatische Bildschirmrollen ein/aus. Die Default-Kanalnummer ist #102.
SET #Kanal:SCROLL UP/DOWN n,m
Rollt den Bildschirminhalt von Zeile (n-32) bis (m-32) aufwärts/abwärts. Die Default-Kanalnummer ist #102.

STATUS

SET STATUS ON/OFF
Schaltet die Statuszeile (an der Oberkante des Bildschirms) ein/aus. Kann auch in Verbindung mit TOGGLE verwendet werden.

VIDEO COLOUR

SET VIDEO COLOUR Ausdruck
Stellt den Farb-Modus für Videoseiten ein, die anschließend geöffnet werden. Die Kanalnummer wird ignoriert.
Wenn eine Text-Videoseite definiert wird, muß immer Farb-Modus 0 gewählt werden. Für hochauflösende Graphikseiten haben die Farb-Modi die folgenden Bedeutungen:
Modus 0 – 2 Farben; horizontale Auflösung 640
Modus 1 – 4 Farben; horizontale Auflösung 320
Modus 2 – 16 Farben; horizontale Auflösung 160
Modus 3 – 256 Farben; horizontale Auflösung 80
Bei einer LORES-(niedrigauflösenden) Graphikseite, die halbsoviel Speicher wie eine HIRES (hochauflösende) Graphikseite verwendet, ist die Anzahl der Farben wie oben; die horizontale Auflösung wird halbiert.

VIDEO MODE

SET VIDEO MODE Ausdruck
Stellt den Video-Modus für anschließend geöffnete Videoseiten ein. Kanalnummern werden ignoriert.
Die Parameter-Werte sind wie folgt:
Modus 0 – 40-Zeichen-Textseite (2 Farbenpaare)
Modus 1 – hochauflösende Graphikseite
Modus 2 – 80-Zeichen-Textseite (4 Farbenpaare)
Modus 5 – niedrigauflösende Graphikseite
Modus 15 – ‚Attribut'-Graphikseite

DISPLAY

DISPLAY #Kanal:AT a FROM b TO c
Definiert ein Fenster für eine Text- oder Graphikseite. Die Bildschirmzeile ‚a' ist die Position, an der die obere Zeile des Auschnitts dargestellt wird. Die Parameter ‚b' und ‚c' sind Zeilen auf der Seite, die angezeigt werden soll, und geben die obere und die untere Zeile des Ausschnitts an. Die Numerierung der Zeilen folgt den Vereinbarungen für Text, unabhängig davon, ob die angezeigte Seite eine Textseite oder eine Graphikseite ist. Siehe **PRINT**.
DISPLAY GRAPHICS
Bewirkt, daß die ersten 20 Zeilen für Graphik verwendet werden und bringt die vorherige Graphikseite auf den Bildschirm, falls eine solche offen war (#101). Löscht die Textseite nicht.
DISPLAY TEXT
Schaltet den gesamten Bildschirm auf Textmodus um und stellt eine ganze Textseite dar, falls sie vorher offen war (#102). Wenn nur eine kleine Textseite vorher offen war, so wird diese gelöscht und als eine neue ganzseitige Textseite geöffnet. Löscht den Graphikbildschirm nicht.

TEXT

TEXT
TEXT 40
TEXT 80
Der Befehl TEXT bewirkt, daß eine Textseite geöffnet und auf dem gesamten Bildschirm (mit Ausnahme des Bereichs der Statuszeile) dargestellt wird. Die Standard-Graphikseite wird geschlossen (falls sie offen war).
40 oder 80 gibt die Anzahl der Spalten auf dem Bildschirm an. Wenn diese Angabe fehlt, wird der vorherige Wert verwendet.
Siehe **DISPLAY TEXT**.

VIDEO X

SET VIDEO X Ausdruck
Definiert die horizontale Größe anschließend geöffneter Videoseiten. Kanalnummern werden ignoriert. Die Größe wird durch die Anzahl der Zeichenpositionen im Bereich von 2 bis 42 unter Verwendung der Koordinaten-Vereinbarungen für Textseiten angegeben.

VIDEO Y

SET VIDEO Y Ausdruck
Wie VIDEO X, mit dem Unterschied, daß VIDEO Y die vertikale Größe als eine Anzahl von Zeilen zwischen 1 und 255 angibt.

ENVELOPE

ENVELOPE #Kanal:NUMBER
a;b,c,d,e;f,g,h,i;...;RELEASE j,k,l,m;...
Definiert eine Ton-Hüllkurve, die dann in Verbindung mit einer SOUND-Anweisung verwendet werden kann. Die Nummer ‚a', die die Hüllkurve kennzeichnet, muß im Bereich von 0 bis 254 sein.
Die Parameter ‚b', ‚c', ‚d' und ‚e' definieren die erste Phase der Hüllkurve; ‚b' gibt die Veränderung der Tonhöhe in Halbtönen an (Dezimalstellen sind dabei erlaubt), ‚c' und ‚d' geben die Veränderung in der Lautstärke für den linken bzw. den rechten Lautsprecher, und ‚e' gibt die Dauer der Phase in ‚Ticks' an (ein Tick ist 1/50 Sekunde).
Die Werte für ‚c' und ‚d' sind im Bereich von 0 bis 63. Sie bestimmen den Wechsel der Lautstärke im Verhältnis zur Gesamtlautstärke, die durch die Sound-Anweisung festgelegt wird. Ein Wert von -63 schaltet den Ton aus (zu große Zahlen werden ignoriert). Am Anfang der Hüllkurve wird angenommen, daß der Ton „aus" ist. Wird keine Stereoanlage benutzt, so ergibt sich die Gesamtlautstärke zu jeder Zeit aus der Summe der Werte für den linken und den rechten Lautsprecher (in Abhängigkeit von den SOUND- und ENVELOPE Anweisungen).
Die nächste Phase wird definiert durch ‚f', ‚g', ‚h' und ‚i'... Für die Anzahl der möglichen Phasen siehe SOUND BUFFER im Abschnitt über ‚Tonoptionen'.
RELEASE (= Nachklang) ist optional; es kann von einer beliebigen Anzahl Phasen mit den jeweiligen Parametern gefolgt sein. Die RELEASE-Phase wird nach Abschluß der vorhergehenden Phasen oder nach Ablauf der in der SOUND-Anweisung angegebenen Zeit ausgeführt, falls es keinen nachfolgenden Ton auf demselben Kanal gibt.

PING

Erzeugt ein ‚ping'-Geräusch.

SOUND

SOUND #Kanal:PITCH Ausdruck, DURATION Ausdruck, LEFT Ausdruck, RIGHT Ausdruck, SOURCE Ausdruck, STYLE Ausdruck, ENVELOPE Ausdruck, SYNC Ausdruck, INTERRUPT
Ermöglicht die vollständige Gestaltung eines Tones. Die Parameter können in beliebiger Reihenfolge angegeben werden.
Die Tonhöhe wird mit PITCH angegeben, wobei der Bereich von 0 bis 127 zur Verfügung steht (gute Ergebnisse können allerdings nur im Bereich von 0 bis 83 erwartet werden). In diesem Bereich wird durch das Erhö-

hen der Zahl der Ton um einen Halbton erhöht. Der Tonhöhen-Wert 37 (der Default) entspricht dem mittleren C.
Mit DURATION wird die Dauer des Tones (ohne die RELEASE-Phase) in ‚Ticks' angegeben (ein Tick ist 1/50 Sekunde). Default ist 50 Ticks.
Die LEFT- und RIGHT-Parameter geben die Lautstärke des Tones für die zwei Stereo-Ausgabekanäle im Bereich von 0 (kein Ton) bis 255 (volle Lautstärke, der Default-Wert) an. Wenn keine Stereoanlage verwendet wird, so wird die Lautstärke durch die Summe der Werte bestimmt, die für die einzelnen Kanäle angegeben wurden.
SOURCE gibt den Tongenerator an, der verwendet werden soll (Default 0). Die Werte sind hierfür 0 bis 3. Tongenerator 3 ist der Geräuschgenerator, der Tonhöhenangaben ignoriert.
Der STYLE-Parameter ist im Bereich von 0 bis 255 (Default 0). Siehe ‚Tonoptionen' bezüglich der Effekte von STYLE.
ENVELOPE gibt die Nummer der Hüllkurve an, die verwendet wird. Siehe **ENVELOPE**-Anweisung. 255 (der Default) ist eine eingebaute Hüllkurve.
SYNC ermöglicht es, den Anfang eines Tones mit 1, 2 oder 3 anderen Tönen von verschiedenen Quellen genau zu synchronisieren. Wenn zum Beispiel drei Töne gleichzeitig beginnen sollen, kann jedem davon die Anweisung SYNC 2 gegeben werden, so daß er mit den beiden anderen synchronisiert wird. Default-Wert ist 0.
INTERRUPT bewirkt, daß der neue Ton jeden momentan klingenden Ton der gleichen Quelle ersetzt.

TONOPTIONEN

SOUND BUFFER

SET SOUND BUFFER Ausdruck
Stellt die Größe des Tonhöhen-Hüllkurven Speicherbereichs für folgende OPEN-Anweisungen zum „SOUND"-Gerät ein. Der Ausdruck ist die Anzahl der Phasen. Erlaubte Werte sind 1 bis 255; Default ist 20. Kann in Verbindung mit ASK verwendet werden.

SOUND STYLE

Die Werte des STYLE-Parameters einer SOUND Anweisung (siehe Abschnitt über „ANWEISUNGEN und Befehle") haben die folgenden Bedeutungen:
Für Tonkanal 0:
16 – Niedrige Verzerrung.

32 – Mittlere Verzerrung.
48 – Hohe Verzerrung.
64 – High-Pass-Filter. Tonkanal 1 gibt den Takt an.
128 – Ring-Modulation mit Kanal 2.
Für Tonkanal 1:
Wie Kanal 0, jedoch verwendet der High-Pass-Filter den Tonkanal 2 und der Ring-Modulator verwendet den Geräuschkanal (Kanal 3).
Für Tonkanal 2:
Wie Kanal 1, jedoch verwendet der High-Pass-Filter den Geräuschkanal (Kanal 3) und der Ring-Modulator verwendet den Tonkanal 0.
Für Tonkanal 3 (Geräuschkanal):
1,2,3 – Verwendet Tonkanal 0,1 oder 2 als Taktfrequenz anstatt der Standardfrequenz 31.25 KHz.
4,8,12 – Wählt die Geräuschfrequenz aus 15, 11 oder 9-Bit polinomschen Zählern anstatt des 17-Bit Standard-Zählers.
16 – Wählt einen polinomischen 7-Bit Zähler anstatt des 17-Bit Zählers.
32 – Verwendet einen Low-Pass-Filter für den Geräuschkanal mit Tonkanal 2 als Takt.
64 – Verwendet einen High-Pass-Filter für den Geräuschkanal mit Tonkanal 0 als Takt.
128 – Verwendet einen Ring-Modulator mit Tonkanal 1.
Eine Kombination von SOUND STYLE-Optionen wird dadurch gewählt, daß die Werte für die einzelnen Optionen zusammengezählt werden und das Ergebnis dann als STYLE-Parameter angegeben wird.

SPEAKER

SET SPEAKER ON/OFF
Kontrolliert die Tonausgabe des internen Lautsprechers; SET SPEAKER OFF wird verwendet, um das Gerät schnell stummzuschalten.

TAPE SOUND

SET TAPE SOUND ON/OFF
Kontrolliert die Übertragung von Tönen vom Cassetteneingang zur Tonausgabe. Erlaubt direkten Durchgang von Musik oder Sprache von Cassette zum eingebauten Lautsprecher oder zur Hi-Fi-Ausgabe. Kann auch in Verbindung mit TOGGLE verwendet werden.

TYPE

TYPE
Besondere Anweisung, um BASIC zu verlassen und das eingebaute Textverarbeitungsprogramm zu starten. Alle BASIC-Programme und Variablen werden durch diese Anweisung gelöscht. Daher wird der Benutzer aufgefordert, die 'ENTER'-Taste zu drücken, um die Anweisung zu bestätigen.

OPTIONEN

EDITOR BUFFER

SET EDITOR BUFFER Ausdruck
Definiert die Größe des Editor-Buffers in 256 Byte-Stükken für Editor-Kanäle, die anschließend geöffnet werden. Kann in Verbindung mit ASK verwendet werden.

EDITOR KEY

SET EDITOR KEY Kanalnummer
Ermöglicht es, daß der angegebene Kanal als Tastatureingabe-Kanal für Editor-Kanäle, die anschließend geöffnet werden, verwendet werden kann. Kann in Verbindung mit ASK verwendet werden.

EDITOR VIDEO

SET EDITOR VIDEO Kanalnummer
Ermöglicht es, daß der angegebene Kanal als Textseite für Editor-Kanäle, die anschließend geöffnet werden, verwendet werden kann. Kann in Verbindung mit ASK verwendet werden.

ASK

ASK Maschinenoption Variable
Dieser Befehl wird verwendet, um Maschinenoptionen wie z.B. KEY RATE abzufragen. Siehe „Maschinenoptionen",„Videooptionen" und „Tonoptionen". Vgl. auch SET und TOGGLE. Die Variable nimmt den momentanen Wert der Maschinenoption an.
z.B. ASK KEY RATE A
weist die momentane Tastatur-Wiederholungsrate der Variablen A zu.

EXT

EXT Parameter-String
Gibt einen String an das Betriebssystem weiter, der dann gültigen äußeren Programmen im Speicher (entweder RAM oder ROM) übergeben wird. Der String wird dann von diesen Programmen interpretiert.
Im allgemeinen gibt das erste Wort des Parameter-Strings die auszuführende Anweisung oder ein neues Programm, das gestartet werden soll, an.
Zum Beispiel:
EXT "WP"
startet das eingebaute Textverarbeitungs-Programm.
Das Wort „HELP" hat eine besondere Bedeutung, da es die Namen aller äußeren Programme anfordert. Die Antworten der äußeren Programme werden zum Default-Systemkanal weitergegeben. Siehe **DEFAULT CHANNEL**-Maschinenoption.
Weitere Anwender- und Service-Programme werden ihre eigenen Anweisungen und Parameter-Anforderungen definieren. Oft werden diese Programme auf die Anweisung
EXT "HELP NAME" (wobei „NAME" der Name des Hauptprogramms ist) mit einer Liste der verfügbaren String-Anweisungen antworten.
Derselbe Effekt wie bei EXT kann dadurch erzielt werden, daß im Direktmodus eine Zeile mit einem Doppelpunkt begonnen wird. In diesem Fall sind keine Anführungszeichen um den Parameter-String erforderlich.
:HELP NAME

SET

Setzt Betriebssystem-Variablen. Vgl. **ASK** und **TOGGLE**.

TOGGLE

Bewirkt bei Maschinenoptionen, die nur zwei Werte (z.B. ON und OFF, d.h. ein und aus) annehmen können, daß von dem einen auf den anderen Wert umgeschaltet wird. Siehe 'Maschinenoptionen', 'Videooptionen' und 'Tonoptionen'. Vgl. SET und ASK.

VARIABLE

SET Variablen-Nummer, Ausdruck
ASK Variablen-Nummer Variable
TOGGLE Variablen-Nummer
Stellt die angegebene Betriebssystem-Variable ein (bzw. fragt diese ab oder schaltet sie um). Eine Beschreibung dieser Variablen ist im ENTERPRISE Technical Manual enthalten.

ALLOCATE

ALLOCATE Ausdruck
Wird in Verbindung mit Maschinencode-Unterprogrammen verwendet. Verschiebt den Programmtext nach oben, um so eine Lücke mit der angegebenen Zahl von Bytes zu schaffen, in der der Benutzer Maschinencode ablegen kann. Setzt den Speicherplatzzähler auf das erste freie Byte in der Lücke. Beachten Sie, daß dieser Befehl alle Variablen zerstört, so daß er nur am Anfang eines Programms verwendet werden sollte.

CODE

CODE = String
CODE Variablenname = String
Wird in Verbindung mit Maschinencode-Unterprogrammen verwendet. Kopiert einen String an die Position, die durch den momentanen Speicherplatzzeiger bestimmt wird. Falls eine Variable angegeben wird, so erhält diese den Wert des Speicherplatzzeigers zugewiesen. Der Speicherplatzzeiger zeigt links neben das Byte, das dem String folgt, der den Maschinencode enthält. Der Variablenname kann später verwendet werden, um die Routine aufzurufen, die Bestimmungsadresse von Sprüngen zu berechnen, usw.

OUT

OUT n, a
Schreibt das Byte 'a' in die Ein-/Ausgabe Speicherstelle 'n'.

POKE

POKE Adresse, Wert
Bringt den Wert in die angegebene Z80-Speicherstelle.

SPOKE

SPOKE Segment, Adresse, Wert
Wie POKE, schreibt jedoch den Wert zu der Systemadresse innerhalb des angegebenen Segments.

FUNKTIONEN und VARIABLEN

BIN(X)

Liefert die Zahl, die der angegebenen binären Darstellung entspricht. Zum Beispiel liefert BIN(11001) das Ergebnis 25.

HEX$(X$) Liefert einen String, der durch die Hexadezimal-Werte der Bytes in X$ angegeben wird. Die Hexadezimal-Bytes können in Groß- oder Kleinbuchstaben angegeben werden und werden durch Kommas getrennt (z.B. HEX$(„21, E3, ff").

IN(N) Liest ein Byte vom Ein/Ausgang N.

PEEK(N) Liefert das Byte von Z80-Speicherplatz N.

SPEEK(S,N) Wie PEEK, liefert jedoch das Byte mit der Systemadresse N innerhalb des Segments S.

USR(N,X) Ruft eine Adresse N auf (die im allgemeinen durch CODE bestimmt wurde) und gibt die ganze Zahl X in HL an das Maschinencode-Unterprogramm weiter. Der Wert, der in HL zurückgelassen wird, ist der Wert, der durch USR geliefert wird.

WORD$(N) Liefert einen 2 Byte-String, der das höherwertige und das niedrigerwertige Byte von N enthält. N ist normalerweise eine Adresse, die durch eine CODE-Anweisung definiert wurde und Rückwärtssprünge u.ä. unter Verwendung von Labels (Markierungen) erlaubt. Das erste Byte des Strings wird das niedrigerwertige sein.

NET CHANNEL

ASK NET CHANNEL Variable
Liefert die Nummer des Kanals, von dem Daten in einem Netzwerk-Buffer sind, die darauf warten, gelesen zu werden.
Sobald das erste Byte von diesem Kanal gelesen wurde, zeigt NET CHANNEL auf den nächsten Kanal, der darauf wartet, bedient zu werden. Der Wert 255 wird geliefert, wenn keine weiteren Kanäle Daten haben.

NET MACHINE

ASK NET MACHINE Variable
NET MACHINE wird zur selben Zeit wie NET CHANNEL verändert und liefert die Netzwerk-Nummer des Gerätes, das die Daten gesendet hat. Das ist besonders wichtig, wenn die Daten vom „allgemeinen" Kanal (NET-0:) erhalten wurden.

NET NUMBER

SET NET NUMBER Ausdruck
Stellt die Netzwerk-Nummer des Computers ein. Diese muß zwischen 1 und 32 liegen. Der Computer hat beim Einschalten die Nummer 0, die keine gültige Netzwerk-Adresse ist.
Eine Netzwerk-Nummer muß festgelegt werden, bevor das Netzwerk verwendet wird, und es sollte keine Nummer sein, die schon ein anderer Computer im Netzwerk hat.

SERIAL BAUD

SET SERIAL BAUD Ausdruck
Der Parameter (aus dem Bereich von 0 bis 15) bestimmt die Baudrate der RS232-Schnittstelle und des Netzwerks entsprechend der untenstehenden Tabelle. Kann in Verbindung mit ASK verwendet werden.

0 = >	50 Baud	6 = >	300 Baud	11 = >	3600 Baud
1 = >	75 Baud	7 = >	600 Baud	12 = >	4800 Baud
2 = >	110 Baud	8 = >	1200 Baud	13 = >	7200 Baud
3 = >	134.5 Baud	9 = >	1800 Baud	14 = >	9600 Baud
4 = >	150 Baud	10 = >	2400 Baud	15 = >	9600 Baud
5 = >	200 Baud				

Die Default-Baudrate ist 9600 (Wert 15).

SERIAL FORMAT

SET SERIAL FORMAT Ausdruck
Definiert das Wort-Format für die serielle Übertragung. Das Format wird durch die binären Bits der angegebenen Zahl wie folgt bestimmt:

BIT	WERT	EFFEKT
0	0	8 Bits
	1	7 Bits
1	0	keine Parität
2	0	gerade Parität wird ignoriert, wenn Bit 1 = 0 ist
	1	ungerade Parität
3	0	2 Stop-Bits
	1	1 Stop-Bit

Alle Bits ab Bit 4 müssen 0 sein. Die Default-Einstellung ist 8 Bits, keine Parität, 2 Stop-Bits. Bei Verwendung des Netzwerks wird dieses Format automatisch wieder eingestellt.

EXOS ist das Betriebssystem des ENTERPRISE. Ein Betriebssystem ist ein Programm, das den Umgang mit dem Computer erleichtert, seine Möglichkeiten so einfach wie möglich zugänglich macht, und dafür sorgt, daß man den Computer möglichst gut ausnutzen kann. Es stellt die Schnittstelle zwischen höheren Programmen (wie der BASIC-Sprache) und dem Computer dar.
Die wichtigsten Teile eines Computersystems sind seine Geräte und seine Peripherie. Dieses sind zum Beispiel der Bildschirm, die Cassettenschnittstelle, ein Drucker, usw. Daher steuert der größte Teil eines Betriebssystems die Geräte und die Peripherie: das Eingabe/Ausgabe-System. Auch die Einteilung des Speichers wird durch das Betriebssystem gehandhabt.

Das Eingabe/Ausgabe-System

Der ENTERPRISE ist äußerst komplex; selbst das Ausführen einfachster Funktionen wie das Ausdrucken eines Strings auf dem Bildschirm erfordert auf der Maschinenebene Tausende von Instruktionen, und denselben String auf einem Drucker auszudrucken erfordert einige hundert Instruktionen mehr. Das ENTERPRISE-Betriebssystem vereinfacht die Schnittstelle zwischen einem Programm und dem Mikrocomputer und ermöglicht es, einen String genauso einfach auf den Drucker wie auf den Bildschirm zu bringen. Das wird dadurch erreicht, daß es Programmen ermöglicht wird, alle Ein- und Ausgaben in derselben Weise zu behandeln. Die gesamte Ein- und Ausgabe erfolgt durch 'Kanäle' (ein Kanal verbindet das Programm einfach mit einem 'Gerät', einem bestimmten Teil des Computersystems). Die Kanäle sind von 0 bis 254 durchnumeriert. Das Betriebssystem erlaubt die folgenden Funktionen mit Kanälen:

Codenummer Funktion

0 System-Reset
1 Einen Kanal öffnen (ein Gerät verbinden)
2 Einen Kanal einrichten und öffnen
3 Einen Kanal schließen (Gerät abtrennen)
4 Einen Kanal schließen und löschen
5 Ein Zeichen von einem Kanal lesen
6 Einen Block lesen
7 Ein Zeichen auf einen Kanal schreiben
8 Einen Block schreiben
9 Den Status des Kanals liefern
10 Den Kanal-Status einstellen und lesen
11 Eine spezielle Funktion ausführen

16 Eine Systemvariable lesen, schreiben oder umschalten
17 Die Eingabe von Kanal zu Kanal einfangen
18 Einen Kanal umleiten
19 Default-Gerätenamen einstellen
20 System-Status liefern
21 Gerät verbinden
22 System-Grenze lesen
23 Benutzer-Grenze einstellen
24 Ein Segment zuteilen
25 Ein Segment freimachen
26 Peripherie abfragen
27 Kanal-Buffer zuteilen
28 Fehlermeldung liefern
29 Modul laden
30 Verschiebbare Module laden
31 Zeit einstellen
32 Zeit lesen
33 Datum einstellen
34 Datum lesen

Diese Funktionen werden vom BASIC verwendet, um Ein- und Ausgabemöglichkeiten bereitzustellen. Sie sind für alle Sprachen verfügbar, so daß sie eine einheitliche Methode der Kommunikation mit den Computerteilen bieten. Sie stehen auch dem Maschinensprache-Programmierer zur Verfügung, was das Schreiben von Programmen in Maschinensprache sehr vereinfacht.

Um das Betriebssystem von einem Maschinenspracheprogramm aus aufzurufen, ist eine einzige Anweisung erforderlich, gefolgt von dem Code für die Funktion. Zum Beispiel wird, um einen Kanal zu öffnen, der folgende Code benötigt:

Maschinencode	Assemblercode
F7	RST 30H
01	DB 1

Das ENTERPRISE-Betriebssystem enthält wesentlich mehr Funktionen als die oben genannten. Eine vollständige Liste der Funktionen und der Aufrufvereinbarungen ist im ENTERPRISE Technical Manual enthalten.

Speicherverwendung

Das Betriebssystem basiert auf ROM (Read Only Memory = Nur-Lesespeicher). Das bedeutet, daß das Programm in ROM gespeichert ist, doch auch RAM-Platz benötigt, um seine Daten zu speichern. Der ENTERPRISE ist in der Lage, eine sehr große Menge RAM- und ROM-Speicher zu verwalten.

Dieser Speicher wird durch Aufteilung in Seiten (nicht zu verwechseln mit den Videoseiten) verwaltet. Jede Seite ist 16 KBytes groß und es gibt 256 Seiten, die zusammen eine maximale Speicherkapazität von 4 MBytes ergeben. Der Z80 im ENTERPRISE kann nur vier dieser Seiten gleichzeitig verwenden (ähnlich wie beim Lesen eines Buches, wo man nur zwei Seiten gleichzeitig sehen kann).

BETRIEBSSYSTEM-VARIABLEN

Nr.	Systembezeichnung	
0 –	IRQ_ENABLE_ STATE	b0 – sound b2 – 1 Hz b4 – video b6 – extern
	b1,b3,b5,b7 müssen den Wert 0 haben	
1 –	FLAG_SOFT_IRQ	Wenn gesetzt, wird die Auslösung eines Software-Interrupts ermöglicht
2 –	CODE_SOFT_IRQ	Erkennungskode des FLAG_SOFT_IRQ
3 –	DEF_TYPE	0 => TAPE 1 => DISK
4 –	DEF_CHAN	Aktuelle Kanal-Nummer
5 –	TIMER	1 Hz Zähler abwärts. Bei 0 wird ein Interrupt ausgelöst.
6 –	LOCK_KEY	Groß/Kleinschreibe-modus
7 –	CLICK_KEY	0 => kein Tasten-Klick
8 –	STOP_IRQ	0 => Interrupt bei STOP-Taste 1 => kein Interrupt
9 –	KEY_IRQ	0 => jeder Tastendruck erzeugt ein Soft-IRQ
10 –	RATE_KEY	Tasten-Wiederholrate in 1/50 sec
11 –	DELAY_KEY	Zeit zwischen Tastendrk. und Wiederh. 0 => keine Wiederholung
12 –	TAPE_SND	0 => Tape-sound aus
13 –	WAIT_SND	1 => Fehler bei vollem Ton-Stapel
14 –	MUTE_SND	0 => Interne Lautsprecher ein
15 –	BUF_SND	Hüllkurvengröße

16 –	BAUD_SER	Serielle Baudrate
17 –	FORM_SER	Format seriell
18 –	ADR_NET	Netzwerkadresse
19 –	NET_IRQ	0 => bei Empfang von Netzwerk wird ein Soft-IRQ erzeugt
20 –	CHAN_NET	Kanal-Nummer des empfangenen Blockes
21 –	MACH_NET	Absenderadresse im Netzwerk
22 –	MODE_VID	Video-mode
23 –	COLR_VID	Colour-mode
24 –	X_SIZ_VID	X-Dimension der Videoseite
25 –	Y_SIZ_VID	Y-Dimension der Videoseite
26 –	ST_FLAG	0 => Statuszeile wird angezeigt
27 –	BORD_VID	Hintergrundfarbe
28 –	BIAS_VID	Erweiterung für Farbpalette 8-16
29 –	VID_EDIT	Kanalnummer für Anzeige vom EDITOR
30 –	KEY_EDIT	Kanalnummer der Tastatur für EDITOR
31 –	BUF_EDIT	Buffergröße in 256 Byte Seiten
32 –	FLG_EDIT	Flags zur Zugriffssteuerung
33 –	SP_TAPE	0 => Schnelles Speichern auf Kassette
34 –	PROTECT	0 => Kein Kassettenschutz
35 –	LV_TAPE	Aufnahmelautstärke für Kassette
36 –	REM1	0 => Remote-Ausgang aus
37 –	REM2	dto.
38 –	SPRITE	Externe Farbpriorität
39 –	RANDOM_IRQ	Wird bei jedem Interrupt erhöht

Alle Betriebssystemvariablen können vom BASIC aus verändert werden. Dafür gibt es die Befehle SET, ASK und TOGGLE. Der einfachste Weg bei Veränderungen ist das Verwenden von Variablennummern. Einige Beispiele dafür sind:

```
SET 36,1                    TOGGLE 37
ASK 36 A                    ASK 37 X
PRINT A                     PRINT X
1                           255
```

Fast immer enthält ein gerade geschriebenes Programm Fehler. Es kann schwierig sein, einen Fehler zu finden – oder herauszufinden, was für ein Fehler es ist.
Hier hilft der Computer, indem er Ihnen so genau wie möglich mitteilt, was falsch ist. Wenn Sie ein Programm ablaufen lassen, das einen BASIC-Fehler enthält, wird der Computer an der Stelle anhalten, an der er das Programm nicht mehr versteht. Er teilt Ihnen dann mit, was die Ursache des Problems ist.
Beachten Sie: Der Computer kann Ihnen bei anderen Fehlern nicht in der gleichen Weise weiterhelfen. Wenn Sie zum Beispiel die Operatorenprioritäten vergessen oder ein anderes Ergebnis erwarten, so wird dies normalerweise das Programm nicht daran hindern, bis zum Ende weiterzulaufen. Mit anderen Worten: Das Programm tut einfach etwas anderes, als Sie von ihm erwarten. Der Computer kann nur Fehler in der Syntax oder der Organisation Ihres BASIC feststellen oder Probleme, die dadurch verursacht wurden, daß eine unmögliche Handlung von ihm verlangt wurde.

```
*** Nicht verstanden
```

Wenn während eines laufenden Programms ein Problem auftritt, das eine Fortsetzung des Programms unmöglich macht, enthält die Fehlermeldung die betreffende Zeile; z.B.:

```
*** ungültiges Argument für SQR 300 PRINT SQR(Y)
```

Sie können jetzt den Cursor aufwärtsbewegen und die Zeile 300 editieren.
Die Funktionen EXLINE, EXTYPE und EXSTRING$ helfen Ihnen, Fehler und andere Ausnahmezustände zu bewältigen. Jeder Fehler hat seine eigene Kennzahl, die durch EXTYPE abgefragt werden kann. Für die meisten Fehler werden besondere Mitteilungen auf den Bildschirm gebracht, es sei denn, diese Einrichtung wird explizit unterdrückt.
Wenn ein Ausnahmezustand auftritt, der von keiner der eingebauten Fehlermeldungen gekennzeichnet wird, dann wird die allgemeine Fehlerkategorie zusammen mit der Ausnahme-Kennzahl ausgegeben; z.B.:

* Überlauf 1234

Die einzelnen Fehlermeldungen sind:

1000- unerwarteter Wert eingegeben
1001- Überlauf numerischer Konstante
1002- Überlauf numerischen Ausdrucks
1051- Überlauf des String-Ausdrucks
1106- Überlauf der String-Zuweisung
2000- falscher Index
2001- Feld außerhalb des erlaubten Bereichs
2002- falscher Index
3000- mathematisch
3001- Division durch Null
3004- ungültiges Argument für LOG
3005- ungültiges Argument für SQR
3007- ungültiges Argument für ASIN oder ACOS
4000- ungültige DEF-Parameter
4001- Parameter
4002- CHR$-Argument außerhalb des erlaubten Bereichs
4003- ungültiges Argument für ORD
4004- SIZE-Index außerhalb des erlaubten Bereichs
4005- TAB-Argument außerhalb des erlaubten Bereichs
4008- LBOUND-Index außerhalb des erlaubten Bereichs
4009- UBOUND-Index außerhalb des erlaubten Bereichs
4301- CHAIN-Parameter-Fehler
5000- Speicherplatz zu klein
5100- Stapel-Speicher zu klein
5110- Speichererweiterung zu klein
5120- ALLOCATE-Speicher zu klein
6000- Matrix
7000- Datei-Benutzung
7001- ungültige Kanal-Nummer
7003- Kanal schon offen
7004- Kanal nicht offen
7401- TRACE-Kanal nicht offen
8000- Ein-/Ausgabe
8001- Keine Daten mehr für READ/INPUT
8101- numerische Daten erwartet
8201- ungültiger USING-String
8202- keine Format-Angabe im USING-String
8203- USING-Format-Angabe zu kurz
9000- EXOS (Betriebssystem)

9110- Datei nicht gefunden
9111- Diskette voll
9207- Copyright INTELLIGENT SOFTWARE LTD
9208- Kassetten-Lese-Fehler
9209- Editor: Überlänge der zu ladenden Datei
9210- Fehler in der zu ladenden Datei
9211- Editor: Tastatur-Kanal-Fehler
9212- Editor: Video-Kanal-Fehler
9213- Netzwerk-Verbindung schon vorhanden
9214- Netzwerk-Adresse nicht gesetzt
9215- Schnittstelle schon belegt
9216- Graphik-Cursor-Koordinaten außerhalb des erlaubten Bereichs
9217- Cursor-Koordinaten außerhalb des erlaubten Bereichs
9218- zu scrollender Zeilenbereich außerhalb des erlaubten Bereichs
9219- ungültige Bildschirmseiten-Datei
9220- Display-Parameter außerhalb des erlaubten Bereichs
9221- Bildschirm-Modus außerhalb des erlaubten Bereichs
9222- Bildschirm-Format außerhalb des erlaubten Bereichs
9223- Ton-Stapel voll
9224- Hüllkurvenstapel voll
9225- Hüllkurve außerhalb des erlaubten Bereichs
9226- Funktionstasten-String außerhalb des erlaubten Bereichs
9227- Schutz-Verletzung
9228- Datei-Ende
9229- STOP-Taste gedrückt
9230- ungültiges Steuerzeichen
9231- Aufruf dieses Gerätes nicht möglich
9232- Geräte-Nummer außerhalb des erlaubten Bereichs
9233- Gerät wird schon benutzt
9234- ungültiger Aufruf einer Spezialfunktion
9235- Datum oder Uhrzeit außerhalb des erlaubten Bereichs
9236- Ende der Modul-Datei
9237- Modul nicht verlegbar
9238- unbekannte Modulart
9239- ungültiger Datei-Anfang
9240- unbekannter Befehls-String

9241- ungültige Gerät-Bezeichnung
9242- unbekannte EXOS-Variablen-Number
9243- EXOS (Betriebssystem)
9246- Video-Speicher-Platz zu klein
9247- Speicherplatz zu klein
9248- EXOS (Betriebssystem)
9249- Kanal schon vorhanden
9250- Gerät nicht vorhanden
9251- Kanal nicht vorhanden
9252- EXOS-Stapel-Speicher Überlauf
9253- ungültiger EXOS-String
9254- verbotener EXOS-Funktionsaufruf
9255- ungültiger EXOS Funktions-Code
10000- Programmablauffehler
10002- RETURN ohne GOSUB
10004- kein CASE angegeben
10005- Programm nicht vorhanden
11000- Graphisch
12000- Echt-Zeit
13000- System
20000- nicht verstanden
20001- Zeilennummer außerhalb des erlaubten Bereichs
20002- ungültiger Zeilennummern-Bereich
20004- Zeilennummer ist nicht vorhanden
20010- RENUMBER kann nicht ausgeführt werden
20020- CONTINUE nicht möglich
20030- Identifikator wird erwartet
20031- String-Identifikator wird erwartet
20032- Feld-Identifikator wird erwartet
20034- Typen-Verwechslung
20040- Variable nicht initialisiert
20041- Identifikator doppelt initialisiert
20042- Identifikator zu lang
20043- Anführungszeichen nicht geschlossen
20050- Blockende fehlt
20051- ungültiges Blockende
20052- zu viele Blocks ineinander verschachtelt
20060- ungültige Anweisung an das Betriebssystem
20071- Programm-Befehl im Direktmodus unzulässig
20072- Direktmodus-Befehl im Programm unzulässig
20073- Befehl nach THEN ungültig
20074- ungültige Mehr-Befehls-Zeile
20075- Zeile zu lang
20080- ungültiges Datei-Format
20081- Programme stimmen nicht überein

21000- System
30000- BASIC-Interpreter schadhaft
30001- System

Fehlermeldungen können, wenn erforderlich, durch die Anweisung WHEN EXCEPTION und einen Handler Block abgefangen werden (siehe Seite $$$). Ein Ausnahme-Handler kann verwendet werden, um jede Art von Fehlern abzufangen, auch solche wie Syntax-Fehler (z.B. ein falsch geschriebenes Schlüsselwort) oder einen Speicher-Overflow abzufangen. Hierbei muß man vorsichtig sein, da ein RETRY bei einem permanenten Fehler bewirkt, daß das Programm in eine unendliche Schleife gerät. Fehler wie das Teilen durch 0 oder ein negatives SQR-Argument können abgefangen werden, ohne daß das Programm unterbrochen wird.

Dieses Glossar wurde geschrieben, damit Sie den Computer-Jargon kennenlernen, dem Sie begegnen werden, wenn Ihr Interesse für Computer wächst. Die meisten der hier erklärten Wörter erscheinen irgendwo in diesem Handbuch, andere allerdings nicht, denn wir haben versucht, alle Erklärungen nach Möglichkeit ohne Computer-Jargon zu präsentieren. Sie werden dieses Glossar beim Lesen von Computerbüchern oder Computerzeitschriften nützlich finden, die voller Wörter sind, die Sie nicht verstehen.

ADRESSE Ein 'Platz' im Speicher des Computers. Angegeben durch eine Zahl, entweder dezimal (im normalen Zahlensystem), hexadezimal (zur Basis von 16) oder binär (zur Basis von 2). Eine Adresse innerhalb eines Computers kann eine oder mehrere Zahlen enthalten, je nachdem, was der Computer oder das Programm, das er verwendet, gerade macht. Zum Beispiel enthalten Adressen, die für die Bildschirmdarstellung zuständig sind, verschiedene Zahlen, je nachdem was dort erscheinen soll.

ALGORITHMUS Eine Strategie zur Lösung einer Aufgabe. Die Ideen, die ein Programm funktionieren lassen. Zunächst denkt man sich einen Algorithmus aus, dann schreibt man das Programm.

ALPHANUMERISCHE ZEICHEN Buchstaben oder Zahlen. Der Zeichensatz ohne Sonderzeichen, Satzzeichen und graphische Zeichen wird alphanumerisch genannt.

ANSI American National Standards Institute. Ein gemeinsames Komitee der ANSI und der European Computer Manufacturers' Association entwickelte die Spezifikationen für Standard BASIC.

ANWEISUNG Siehe **KOMMANDO**. IF/THEN-Zeilen stellen Anweisungen dar. IF A = 2 hat allein nicht viel Sinn, IF A = 2 THEN PRINT „A = 2" ist dagegen eine sinnvolle Anweisung.

ARGUMENT Siehe **OPERAND** und **PARAMETER**.

ARRAY (FELD) Eine Variable, die selbst mehrere Variablen enthält. Ein Array kann man sich als Liste (eindimensional) oder als Gitter (zweidimensional) vorstellen.

ASCII

American Standard Code for Information Interchange. Ein System zur Codierung von Zeichen für die Verwendung in Computern und zum Austausch von Daten mit anderen Geräten. Jedem Zeichen wird dabei eine Nummer zugeordnet.

ASSEMBLER-SPRACHE

Eine Programmiersprache, in der man mit dem Computer in dessen eigener Sprache kommuniziert. Assembler hat einfachere Instruktionen als BASIC, ist aber schwieriger zu verwenden. Man kann jedoch in der Assembler-Sprache Programme sehr viel detaillierter als in BASIC schreiben.

AUFLÖSUNG

Die Anzahl der auf dem Bildschirm darstellbaren Punkte. Diese hängt von den auf dem Computer verfügbaren Graphik-Möglichkeiten ab. Auflösung kann hoch oder niedrig sein. Je höher sie ist, desto größer ist die Genauigkeit von Linien und Figuren, die auf dem Bildschirm gezeichnet werden. Die höchste Auflösung auf dem ENTERPRISE ist 672 Punkte horizontal und 512 Punkte vertikal.

AUFRUF

Eine Verzweigung in einem Programm zu einem Unterprogramm. Beim ENTERPRISE können benutzerdefinierte Funktionen innerhalb von BASIC-Programmen mit Hilfe des Schlüsselwortes CALL aufgerufen werden.

AUSDRUCK

Eine Gruppe von Zahlen oder Wörtern, die einen Wert ergibt, wenn sie ausgerechnet wird. Ein numerischer Ausdruck ist z.B. X * 6. In BASIC gibt es außerdem String-Ausdrücke.

AUSGABE (engl. Output)

Alles, was den Computer verläßt; Darstellungen auf dem Bildschirm, Töne, Ausgedrucktes, Ergebnisse.

BASIC

Die Programmiersprache, die der ENTERPRISE und die meisten anderen Microcomputer verstehen. Es gibt viele verschiedene Arten oder Dialekte von BASIC. Der Name ist eine Abkürzung für Beginners' All-purpose Symbolic Instruction Code. Ursprünglich auf Großrechenanlagen entwickelt als eine Hilfe, das Programmieren zu erlernen.

BAUD

Die Geschwindigkeit, mit der Daten aus dem Computer oder in den Computer gesendet werden können. Entspricht in etwa 'Bits pro Sekunde'. Der ENTERPRISE lädt

ein Programm von Cassette normalerweise mit 2400 Baud.

BEDINGUNG Der Computer verwendet Bedingungen, um Entscheidungen zu treffen. Eine IF/THEN-Anweisung kann man etwa übersetzen mit: „Unter der Bedingung, daß ... tue ...".

BEFEHL Siehe **Kommando**.

BENUTZER (engl. User) Derjenige, der einen Computer oder ein Programm benutzt. Der Begriff Benutzerfreundlichkeit bezieht sich auf die Einfachheit, mit der Programme bedient werden können.

BETRIEBSSYSTEM Ein Programm, das die gesamte Ein- und Ausgabe kontrolliert. Wird normalerweise bei Diskettensystemen verwendet, um die Laufwerke und die Organisation der Daten auf der Diskette zu steuern. CP/M ist ein Betriebssystem. Der ENTERPRISE enthält ein sehr umfassendes Betriebssystem, EXOS, um alle Hardware-Möglichkeiten des Computers und der Geräte, die angeschlossen werden können, zu steuern.

BEZEICHNER Ein Name, der eine Variable, eine Funktion, ein Gerät oder eine andere Komponente des Computers oder des Programms eindeutig kennzeichnet. Das Verwenden von Bezeichnern, aus deren Wortlaut man auf ihre Bedeutung schließen kann, bewirkt, daß Programme leichter zu verstehen sind.

BINÄR Zahlensystem mit Basis 2 (anstatt von 10). Binäre Zahlen können daher nur aus 0 und 1 bestehen. Der Dezimalzahl 9 entspricht also die Binärzahl 1001. Binäre Zahlen können wie normale Zahlen als Spalten gesehen werden. Anstelle der Einheiten Zehner und Hunderter hat man Einer, 2er, 4er, 8er, 16er, 32er und so weiter. Der Computer selber 'denkt' in Binärzahlen.

BIT Eine binäre Zahl (0 oder 1). Die kleinste Informationseinheit, die der Computer erkennen kann. Bits können auch als hohe/niedrige Spannung (wie innerhalb des Computers) oder positive/negative magnetische Impulse (wie auf Cassette) dargestellt werden.

BUG Ein Fehler im Programm. Manchmal leicht herauszufinden

– wie bei einem falsch geschriebenen BASIC-Wort. In anderen Fällen sind Fehler nicht so einfach zu lokalisieren. Die Wirkung eines Fehlers kann zwar erkannt werden, doch die eigentliche Ursache im Programm ist nur schwer zu finden.

BYTE 8 Bits. Im Computer werden sie zu einer Einheit zusammengefaßt. Der Speicher wird in Bytes oder Kilobytes (1024 Bytes, auch einfach K genannt) oder Megabytes (1024 K, auch M genannt) gemessen. Normalerweise erfordert das Abspeichern eines Zeichens ein Byte.

CHIP Ein kleiner Kasten voller mikroskopisch kleiner Schaltkreise im Computer. Ein Chip ist etwa 2*2 mm groß (es gibt allerdings verschieden große Chips) und wird aus Silizium hergestellt, das als Halbleiter bekannt ist. Er ist in eine Zelluloidhülle gepackt, damit die empfindlichen Schaltkreise geschützt werden, und hat Metallstifte, um Verbindungen zu anderen Chips innerhalb des Computers herzustellen. Chips werden auch IC (für 'Integrated Circuits') genannt.

CODE Siehe CONTROL-CODE. Code ist außerdem eine Bezeichnung für einen Programmtext (source code) und das Programm, wie es auf dem Computer läuft (object code). Siehe auch MASCHINENCODE.

CODIEREN Eine Bezeichnung für den technischen Prozeß des Schreibens von Programmzeilen im Gegensatz zum Entwurf eines Programms.

COMPILER Ein spezielles Programm, das von einer Computersprache in die andere übersetzt, normalerweise von einer höheren Programmiersprache in die Maschinensprache. Man kann ein compiliertes Programm im Gegensatz zu einem **INTERPRETER**-Programm nicht ohne weiteres ändern, aber im allgemeinen läuft ein Programm schneller, wenn es compiliert wurde.

CONTROL-CODE Ein Zeichen, das nicht auf dem Bildschirm sichtbar ist, das jedoch trotzdem eine Wirkung hat. Ein Beispiel dafür ist CHR$(13), welches einen Zeilenvorschub bewirkt.

CPU Central Processing Unit. Ein großer Chip im Computer, der alle anderen Chips ansteuert. Assembler-Sprache

wird geschrieben, um direkt mit der CPU zu 'sprechen' und variiert entsprechend des jeweiligen CPU-Typs. CPUs werden auch Prozessoren genannt. Der ENTERPRISE benutzt eine Z80 CPU.

CRASH Dramatische Fehlfunktion (entweder des Programms oder des Computers) aus verschiedenen Gründen. Man spricht auch von einem System-Absturz.

CTRL Abkürzung für Control. Als eine Taste vorhanden, um Control-Codes direkt über die Tastatur einzugeben.

DATEI (engl. File) Eine organisierte Sammlung von Daten. Kann ein Programm sein, oder Daten, die von einem Programm verwendet werden. Dateien werden auf Diskette oder Cassette abgespeichert (oder auf einem Drucker ausgedruckt).

DATEN Buchstaben oder Zahlen, die von einem Programm verwendet werden, um eine nützliche Information darzustellen oder eine Aufgabe zu erfüllen. Die Zahlen, die verwendet werden, um eigene Zeichensätze zu definieren sind Daten. DATA ist ein BASIC-Kommando, mit dem man im Programm dem Computer Daten zur Verfügung stellt. Diese werden mit READ abgerufen.

DATENBANK Eine Sammlung von Daten. Oft auch verwendet als Kurzbezeichnung für Programme, die ausschließlich entwikkelt wurden, um Daten zu speichern und zu verändern – zum Beispiel Adressen und Telefonnummern. Datenbanken können sehr einfach sein und lediglich ermöglichen, Daten einzugeben und genauso wieder zurückzuerhalten. Aber es gibt auch äußerst komplexe Programme, die es Ihnen erlauben, eine aufwendige Datenverwaltung auf dem Computer zu entwickeln, Daten nach bestimmten Kriterien zu sortieren und auszuwählen.

DEBUGGEN Fehler in einem Programm suchen und dann korrigieren. Dies ist die kritische Phase bei der Erstellung großer oder komplexer Programme. Fehler sind nicht immer offensichtlich und jedes Programm sollte sorgfältig auf alle denkbaren Möglichkeiten getestet werden.

DEFINIEREN Etwas detailliert angeben (insbesondere für spätere Verwendung). Definiert werden zum Beispiel die von Ihnen entworfenen Zeichen oder Funktionen.

DEKLARATION Die Mitteilung an den Computer, daß man eine bestimmte Variable oder ein bestimmtes Array verwenden will.

DEZIMAL Das Zahlensystem, an das wir alle gewöhnt sind und das die Basis 10 verwendet.

DIMENSIONEN Wird bei der Beschreibung von Arrays verwendet. ARRAY(X) ist ein eindimensionales Array, ARRAY(X,Y) ist zweidimensional.

DISKETTE Eine runde Scheibe aus magnetischem Material, die in Verbindung mit einem Diskettenlaufwerk verwendet wird, um Programme und Daten abzuspeichern, die dann später wieder zurückgelesen werden können. Das ist eine sehr viel schnellere Methode, Daten zu speichern und einzulesen, als die Verwendung eines Cassettenrecorders. Die Magnetscheiben, die aus ähnlichem Material wie Tonbänder bestehen, stecken in einer Plastikhülle (die man niemals aufmachen soll !) und sehen aus wie flexible kleine Schallplatten. Die normalen Diskettenlaufwerke für den Anschluß an den ENTERPRISE sind 3½ Zoll „Microfloppies".

EINGABE (engl. Input) Alle Daten oder Programme, die in einen Computer hineingehen, entweder von einem Speichergerät (Diskettenlaufwerk etc.) oder durch die Tastatur.

ELEMENT Ein Teil eines Arrays. Die Anzahl der Elemente in einem Array wird bei der Deklaration des Arrays festgelegt.

ESC Abkürzung für „ESCAPE". Ein Steuerzeichen, das angibt, daß darauf folgende Zeichen eine besondere Bedeutung haben.

EXPONENT 10 ∧ 3 entspricht 10 * 10 * 10. 3 ist hier der Exponent.

FIRMWARE Ein Programm, das ständig im Computer ist. Das EXOS-Betriebssystem zum Beispiel ist Firmware.

FUNKTION Beim ENTERPRISE ein Programmteil, der eine bestimmte Aufgabe oder Berechnung ausführt. Eine Funktion wird

betrachtet als „Programm innerhalb eines Programms". Der Computer verwendet sie nur, wenn er die Anweisung dazu erhält. Eine Funktion kann entweder vordefiniert sein (also als Teil von BASIC zur Verfügung stehen, wie z.B. INT, CHR$, LOG) oder sie kann durch den Benutzer definiert werden.

GERÄT (engl. Device) Der Fernseher oder ein Cassettenrecorder, die mit dem Computer verbunden sind und mit diesem in irgendeiner Weise zusammenarbeiten. Der Tongenerator und die Graphik werden beim ENTERPRISE ebenfalls als Geräte betrachet. An den ENTERPRISE angeschlossene Peripherie-Geräte werden durch BASIC über das Betriebssystem angesteuert.

GRAPHIK Die Fähigkeit des Computers, Bilder zu zeichnen oder Linien und spezielle Zeichen zu verwenden, um Informationen darzustellen.

HARDWARE Der physikalische Teil eines Computersystems, und zwar sowohl die Teile des Computers selbst als auch angeschlossene Geräte.

HEXADEZIMAL Ein Zahlensystem mit der Basis 16 (anstatt 10). Das hexadezimale Zahlensystem verwendet die Zahlen 0–9 und A–F (= 10–15 dezimal). Hexadezimalzahlen (kurz Hex-Zahlen) geht oft ein & voran oder sie enden mit H, um sie als Hex-Zahlen zu kennzeichnen (26H ist 38 dezimal). Hex-Zahlen sind eine sinnvolle Darstellungsform für Binärzahlen (mit einer Hex-Zahl kann man 4 Bit darstellen, für 1 Byte benötigt man also 2 Hex-Zahlen).

IC Integrated Circuit. Andere Bezeichnung für Silizium-Chip.

INFORMATIONEN Als solche werden manchmal die Ergebnisse der Verarbeitung von Daten bezeichnet. Das bedeutet die Ergebnisse von Sortieren oder Suchen in Dateien und/oder Feldern. Informationen sind Daten, die zu einer sinnvollen Form zusammengefügt wurden. Als Informationen bezeichnet man auch alles, was von einem Programm verarbeitet wird, also Zahlen, Strings, Variablen.

INTELLIGENT SOFTWARE Intelligente Programme. Die Firma **Intelligent Software Ltd** ist der Entwickler des ENTERPRISE .

INTERFACE

Ein Stück Hardware, das eine Verbindung zwischen dem Computer und einem äußeren Gerät herstellt. Die Cassettenrecorder- und Fernseher-Anschlüsse sind zum Beispiel mit Interfaces im ENTERPRISE verbunden.

INTERPRETER

Ein spezielles Computerprogramm, das Programme von einer Computersprache in eine andere übersetzt, während es abgearbeitet wird. Das BASIC im ENTERPRISE ist ein Interpreter. Interpreter unterscheiden sich von COMPILERN, die ein Programm in einem Stück übersetzen und dadurch erst in einen ausführbaren Code umwandeln.

I/O

Input/Output (= Eingabe/Ausgabe). Wird in bezug auf die Teile des Computers verwendet, die mit dem Datenfluß zu oder von dem Computer zu tun haben.

KANAL (engl. CHANNEL)

Ein ‚Weg', über den Informationen in den Computer gelangen oder diesen verlassen können. Eine Verbindung zwischen verschiedenen Teilen des Computers. Normalerweise kann die Verwendung eines Kanals durch ein Programm umdefiniert werden.

KILOBYTE

1024 Bytes (die der dezimalen Zahl 1000 am nächsten liegende glatte Binärzahl (2∧10)).

KOAXIALKABEL

Das Kabel, das den Computer mit dem Fernseher verbindet. Technisch gesehen ein Kabel, dessen innerer Leiter vollständig durch eine andere leitende Schicht umhüllt ist, um Störungen zu vermeiden.

KOMMANDO (ANWEISUNG, BEFEHL)

Ein Programmschlüsselwort wie z.B. GOTO, CALL etc, das dem Computer direkte Anweisungen gibt. Das erste Schlüsselwort einer Programmzeile ist normalerweise ein Kommando.

KONSTANTE

Eine Zahl oder ein String, der sich nicht ändert. Das Wort PI kennzeichnet eine Konstante (3.14...), die in Berechnungen von Kreisen verwendet wird. 2 ist eine Konstante; „A" ist eine Konstante. Es gibt Fälle, in denen etwas, das als 'Variable' bezeichnet ist, in einem Programm als Konstante verwendet wird.

KOORDINATE

Eine Zahl, die die Position von etwas spezifiziert, insbesondere auf dem Bildschirm. PRINT AT verwendet zwei Koordinaten, vertikal und horizontal, um festzulegen, wo

ein Zeichen gedruckt werden soll. PLOT verwendet ebenfalls Koordinaten, um die Positionen von Punkten oder den Anfang oder das Ende einer Graphik-Linie anzugeben.

LADEN Etwas von Cassette oder Diskette in den Speicher des Computers bringen.

LESEN Daten oder Informationen aus etwas herausholen (z.B. aus einer Datei).

LÖSCHEN Den Inhalt entfernen, entweder eines Speichermediums (wie Diskette oder Cassette) oder des Speichers des Computers.

LOGIK Wissenschaft der Beweisführung. Computer treffen alle Entscheidungen nach strengen logischen Regeln. Die Schlüsselwörter AND und OR sind logische Operatoren.

MAINFRAME Ein wirklich großer Computer. Einige davon sind groß genug, um mehrere Räume zu füllen.

MASCHINENCODE Tatsächlicher Maschinencode ist immer binär. Er ist das, was der Computer wirklich direkt versteht und der BASIC-Interpreter, den Sie verwenden, ist ein Programm aus Maschinencode-Instruktionen, das im Computer läuft. Assembler-Sprache ermöglicht es, Maschinencode direkt zu beschreiben, wobei Buchstaben und Symbole verwendet werden, um die Binärzahlen anschaulich darzustellen.

MEGABYTE Etwa eine Million Bytes. Entspricht 1024 Kilobytes (2∧20 Bytes).

MICROSOFT BASIC Eine allgemeine Bezeichnung für BASIC-Interpreter, die von Microsoft Inc. in den USA entwickelt wurden. Die meisten Befehle von Microsoft BASIC können als eine Untermenge des BASIC des ENTERPRISE verwendet werden.

MODULAR Gemacht aus kleinen, untereinander verbundenen Teilen.

NETZWERK Eine Anzahl von Computern, die alle miteinander verbunden sind und untereinander kommunizieren können. Der ENTERPRISE hat diese Möglichkeiten und verwendet dabei eine Kommunikation, die als Intelligent Net bekannt ist.

OPERAND Eine Zahl oder Variable, die der Gegenstand einer Operation ist. In dem Ausdruck 6 * 2 sind 6 und 2 die Operanden. Sie werden gelegentlich auch als Argumente bezeichnet.

OPERATOR Mathematisches Wort für Zeichen, die Operationen wie z.B. *, /, + und – (Multiplikation, Division, Addition und Subtraktion) bezeichnen.

PARAMETER Eine Variable, die einer Funktion übergeben wird oder der ein Wert zugewiesen wird. Bei TAN(X) ist X der Parameter (bzw. das ARGUMENT).

PERIPHERIE Eine Bezeichnung für jedes Gerät, das zwar kein Teil des Computers ist, aber unter der Kontrolle des Computers arbeitet – der Fernseher, der Cassettenrecorder oder der Drucker zum Beispiel.

PIXEL Ein einzelner Punkt auf dem Bildschirm.

PORT Eine Anschlußbuchse, die den Computer mit einem Peripheriegerät verbindet.

PROGRAMM Eine Folge von geordneten Befehlen, die den Computer veranlassen, eine Aufgabe zu erfüllen.

PROGRAMM-MODUL (CARTRIDGE) Ein Kunststoffkästchen mit einem oder zwei Chips, die ein Programm enthalten. Module werden in den Steckplatz an der Seite des ENTERPRISE eingesteckt und stellen ein Programm bereit, das sofort läuft. Telespielgeräte benutzen Module, die man auswechselt, wenn man ein anderes Spiel spielen möchte.

PROZEDUR Andere Bezeichnung für ein Unterprogramm. Beim BASIC des ENTERPRISE haben Funktionen alle normalen Eigenschaften von Prozeduren.

PSG Eine Abkürzung für Programmable Sound Generator (= Programmierbarer Tongenerator). Ein Chip im ENTERPRISE, der Signale an einen Lautsprecher sendet und dadurch Töne produziert.

RAM Random Access Memory (= Speicher mit wahlfreiem Zugriff). Speicher, der für das vorübergehende Abspeichern von Programmen oder Daten verwendet wird. Er

verliert seinen Inhalt, wenn der Computer ausgeschaltet wird.

REFERENZPARAMETER Daten, die an eine Funktion übergeben werden und tatsächlich eine Variable (nicht einfach den Wert der Variablen) enthalten.

RESERVIERTES WORT Ein BASIC-Wort, das nur als Schlüsselwort und nicht für andere Zwecke verwendet werden kann. Die meisten BASIC-Wörter können als Variablennamen verwendet werden. Die übrigen sind reservierte Wörter.

ROBUST Bezeichnung für ein Programm oder Software-Paket, das sehr wenige Möglichkeiten der Fehlbedienung hat und gut funktioniert, egal was der Benutzer damit macht.

ROM Read Only Memory (= Nur-Lese-Speicher). Speicher, der Firmware enthält. Der Inhalt eines ROMs kann nicht verändert, wohl aber gelesen werden. Der Inhalt bleibt auch nach dem Abschalten des Computers enthalten.

SCHLEIFE Wiederholung innerhalb eines Programms durch Rückkehr zu einer Zeile, die den Anfang der Schleife darstellt. Ein Programmteil, dessen Ende zu seinem Anfang zurückführt, bis eine Bedingung für das Beenden der Schleife erfüllt ist. Eine unendliche Schleife ist eine Schleife, bei der diese Bedingung nie erfüllt ist. In solchen Fällen kann der Computer scheinbar aufhören, etwas zu tun.

SCHLÜSSELWORT Ein Wort, das einen BASIC-Befehl darstellt und insofern eine festgelegte Bedeutung hat.

SCHREIBEN Daten oder Informationen in etwas hineintun (z.B. in eine Datei).

SOFTWARE Instruktionen für den Computer, die verändert werden können. Ein Programm ist Software.

SORTIEREN Teile einer Datei oder einer Liste in eine bestimmte Reihenfolge bringen (z.B. alphabetisch).

SPANNUNGSSPITZE Ein schnelles übermäßiges Ansteigen oder Abfallen der Netzspannung. Kann gelegentliche Probleme mit Diskettensystemen verursachen oder sogar einen Computer kurzzeitig abschalten, so daß im Speicher befindliche Programme und Daten verlorengehen (kommt selten vor).

SPEICHERN Informationen oder Daten für spätere Verwendung ablegen, entweder kurzzeitig in RAM oder dauerhaft auf Cassette oder Diskette.

SPRACHE Ein Hilfsmittel für die Programmierung eines Computers (ein System um mit ihm zu kommunizieren). BASIC ist eine Sprache. Andere Programmiersprachen sind zum Beispiel Pascal, Fortran, Forth, Lisp und Logo. Jede davon ist für eine bestimmte Art der Programmierung enwickelt worden.

STRING (ZEICHENKETTE) Eine Reihe von Zeichen, die vom Computer nicht verstanden werden, aber dennoch bearbeitet werden können. In BASIC erscheint ein String in einem Programmlisting entweder als Variable mit $ am Ende oder als eine Reihe von Zeichen, die in Anführungszeichen stehen.

TESTEN Ein Programm ausprobieren, um zu sehen, ob es funktioniert.

UNTERPROGRAMM (engl. Subroutine) Ähnlich einer Funktion. Ein Programmteil, der unter der Kontrolle des übrigen Programms eine bestimmte Aufgabe erfüllt. Ein Unterprogramm wird durch die Wörter GOSUB und RETURN gesteuert. Eine Funktion dagegen hat einen Namen, unter dem sie aufgerufen wird.

VARIABLE Eine Zahl oder ein String, der veränderbar ist, und einen Namen erhalten hat, unter dem der Computer ihn erkennen kann.

VERZWEIGUNG Eine Stelle im Programm, an der die Ausführung von Zeilen mit fortlaufenden Zeilennummern unterbrochen wird und der Computer in einen anderen Teil des Programms (z.B. eine Funktion oder ein Unterprogramm) springt.

ZEICHEN Ein Symbol, das verwendet wird, um Informationen darzustellen – Buchstaben, Zahlen, Rechen- und Satzzeichen.

ZUGRIFF Zurückholen von Informationen von einem äußeren Speichergerät (z.B. Drucker oder Cassette). Auch das Zurückholen von Informationen von einem Programm wie z.B einem Dateiverwaltungsprogramm.

AUTO (172)

AUTO
AUTO AT 100 STEP 10
AUTO STEP 100

CONTINUE (172)

DELETE (172)

DELETE LAST
DELETE FIRST TO 100
DELETE 1 TO 199, 300, 500 TO 9999
DELETE "Name"

EDIT (172)

EDIT Programm-Nummer
EDIT "Name"

INFO (173)

NEW (173)

NEW ALL (173)

LIST (173)

LIST #Kanal:Zeilenangabe TO Zeilenangabe
LIST #Kanal:Zeilenangabe – Zeilenangabe
LIST Blockname
LIST 300
LIST 300 TO 400
LIST FIRST TO 900, 1000, 2000 TO LAST
LIST – 500, 700 TO
LIST MEINE_FUNKTION
LIST LAST

LLIST (173)

LLIST List-Ausdruck

REM (173)

REM Kommentar

RENUMBER (173)

RENUMBER Zeilenangabe TO Zeilenangabe AT Ausdruck
STEP Ausdruck
RENUMBER Blockname AT Ausdruck STEP Ausdruck
RENUMBER FIRST TO 100
RENUMBER 10 TO 100 AT 300 STEP 10
RENUMBER STEP 100
RENUMBER "Name" AT 5000

LADEN (207)

Etwas von Cassette oder Diskette in den Speicher des Computers bringen.

RUN (174)	RUN (Parameter-Liste) RUN Zeilennummer RUN #Kanal:Dateiname (Parameter-Liste) RUN Gerätename (Parameter-Liste)
START (174)	
STOP (174)	
TRACE (175)	TRACE ON TO #Kanal TRACE OFF

OPTIONEN

FKEY (175)	SET #Kanal:FKEY Tastennummer String
KEY CLICK (175)	SET KEY CLICK ON/OFF
KEY DELAY (176)	SET KEY DELAY Ausdruck
KEY RATE (176)	SET KEY RATE Ausdruck
:BRD (176)	
:UK (176)	

DATE (177)	DATE "19850727"
DIM (177)	DIM A(1 TO 10), FRED$(9), B(–7899 TO –7890)
LET (177)	LET Variablenliste = Ausdruck
NUMERIC (177)	NUMERIC Variablen/Array-Liste
OPTION (177)	OPTION ANGLE DEGREES/RADIANS
RANDOMIZE (178)	
STRING (178)	STRING Variablen/Array-Liste *n STRING NAME STRING NAMEN$(4 TO 99)*10
TIME (178)	TIME "15:35:00"
WAIT DELAY (178)	WAIT <DELAY> Ausdruck

ACOS(X) (179)

ANGLE(X,Y) (179)

ASIN(X) (179)

ATN(X) (179)

COS(X) (179)

COSH(X) (179)

COT(X) (179)

CSC(X) (179)

DEG(X) (179)

RAD(X) (179)

SEC(X) (179)

SIN(X) (179)

SINH(X) (179)

TAN(X) (179)

TANH(X) (179)

ABS(X) (180)

CEIL(X) (180)

EPS(X) (180)

EXP(X) (180)

FP(X) (180)

INF (180)

INT(X) (180)

IP(X) (180)

LOG(X) (180)

LOG10(X) (180)

LOG2(X) (180)

MAX(X,Y) (180)

MIN(X,Y) (180)

MOD(X,Y) (181)

PI (181)

REM(X,Y) (181)

RND (181)

RND(X) (181)

ROUND(X,N) (181)

SGN(X) (181)

SQR(X) (181)

TRUNCATE(X,N) (181)

CHR$(X) (182)

INKEY$ (182)

LBOUND(A) (182)

LBOUND(A,N) (182)

LCASE$(A$) (182)

LEN(A$) (182)

LTRIM$(A$) (182)

MAXLEN(A$) (182)

ORD(A$) (182)

POS(X$,Y$,M) (182)

RTRIM$(A$) (182)

SIZE(A) (182)

SIZE(A,N) (182)

STR$(X) (183)

UBOUND(A) (183)

UBOUND(A,N) (183)

UCASE$(A$) (183)

VAL(A$) (183)

DATE$ (184)

FREE (184)

JOY(N) (184) 1 – rechts 2 – links 4 – unten 8 – oben 16 – Drucktaste

TAB(X) (184)

TIME$ (184)

DO (185) — DO WHILE vergleichender Ausdruck DO UNTIL vergleichender Ausdruck

END (185)

END DEF (185)

END HANDLER (185)

END IF (185)

END SELECT (185)

END WHEN (185)

EXIT DO (185)

EXIT FOR (185)

EXIT DEF (185)

FOR (185) — FOR einfache Variable = Ausdruck TO Ausdruck STEP Ausdruck

LOOP (186) — LOOP
LOOP WHILE vergleichender Ausdruck
LOOP UNTIL vergleichender Ausdruck

NEXT (186) — NEXT
NEXT Variable

CASE (187)

ELSE (187)

GOSUB (187) GOSUB Zeilennummer

GOTO (187) GOTO Zeilennummer.

IF (187) IF vergleichender Ausdruck THEN Zeilennummer
IF vergleichender Ausdruck THEN einfache Anweisung
beliebige Anzahl Befehle oder Blöcke
ELSE IF-Zeilen (möglich) beliebige Anzahl Befehle oder Blöcke
ELSE-Zeile (möglich) beliebige Anzahl Befehle oder Blöcke
END IF-Zeile

ON (188) ON Ausdruck GOTO Zeilennummern-Liste
ON Ausdruck GOSUB Zeilennummern-Liste

RETURN (188)

SELECT (188) SELECT CASE Ausdruck
CASE Ausdruck
CASE Ausdruck TO Ausdruck
CASE IS vergleichender Ausdruck
CASE ELSE

END SELECT

THEN (189)

CALL (190)

CALL Funktion
CALL Funktion (Parameterliste)
CALL USR(A,B)+USR(C,D)

CHAIN (190)

CHAIN Programm-Nummer
CHAIN "Name" (Parameterliste)

DEF (190)

DEF Numerischer Bezeichner = Ausdruck
DEF Numerischer Bezeichner (Parameterliste) = Ausdruck
DEF String-Bezeichner = String-Ausdruck
DEF String-Bezeichner (Parameterliste) = String-Ausdruck

Def-Zeile beliebige Anzahl von Anweisungen oder Blöcken
END DEF

Def-Zeile: DEF numerischer Bezeichner
DEF numerischer Bezeichner (Parameterliste)
DEF String-Bezeichner
DEF String-Bezeichner (Parameterliste)

Funktionen können sich selbst rekursiv aufrufen.

PROGRAM (193)

PROGRAM Name (Variablenliste)

CAUSE EXCEPTION (194)
CAUSE EXCEPTION Ausdruck

EXIT HANDLER (194)

HANDLER (194)
HANDLER Handler-Name
Ausnahme-Handler-Anweisungen

END HANDLER (194)

RETRY (194)

WHEN (195)
WHEN EXCEPTION USE Handler-Name Anweisungen
END WHEN

OPTIONEN

INTERRUPT (195)
ASK INTERRUPT CODE
SET INTERRUPT KEY ON/OFF
SET INTERRUPT NET ON/OFF
SET INTERRUPT STOP ON/OFF

TIMER (196)
SET TIMER Ausdruck

VARIABLEN

EXLINE (196)

EXSTRING$(N) (196)

EXTYPE (197)

DATA (198)

DATA Datenliste

GET (198)

GET #Kanal:String-Bezeichner

IMAGE (198)

IMAGE: Format-Angabe

Zahlen-Formatier-Zeichen:
, – druckt ein Komma in der Zahl.
$ – druckt ein Dollarzeichen vor das Vorzeichen der Zahl.
– – druckt ein Leerzeichen oder '-'-Zeichen.
+ – druckt ein '+' oder '-'-Zeichen.
% – druckt eine Zahl mit führenden Nullen.
# – druckt eine Zahl oder ein Leerzeichen mit angehängten Nullen nach einem Dezimalpunkt.
* – druckt eine Zahl oder führende '*'.
. – druckt einen Dezimalpunkt.
^ – druckt den Exponenten (mindestens 4 Zeichen).

String-Formatier-Zeichen:
< Links-Justierung eines Strings in dem Feld, das durch #-Zeichen definiert wird.
druckt ein Zeichen.
> Rechts-Justierung eines Strings.

Das Formatzeichen für Justierung muß am Anfang des Feldes stehen; wenn kein Justierzeichen verwendet wird, wird der String zentriert.

INPUT (198)

INPUT #Kanal, IF MISSING Aktion, AT Zeilen-Nummer, Spalten-Nummer, PROMPT String: Variablenliste

LINE INPUT (199)

LPRINT (199)

LPRINT Print-Ausdruck

OPEN (199)

OPEN #Kanal:NAME Gerät/Dateiname ACCESS Modus
OPEN #Kanal:Gerät/Dateiname

PRINT (202)

PRINT #Kanal, AT Zeilen-Ausdruck, Spalten-Ausdruck: Ausgabe-Liste
PRINT #Kanal, USING Zeilennummer: Ausgabe-Liste
PRINT #Kanal, USING String: Ausgabe-Liste

READ (203)

READ Variablenliste
READ IF MISSING Zeilennummer: Variablenliste
READ IF MISSING EXIT DO: Variablenliste

RESTORE (203)

RESTORE RESTORE Zeilennummer

LOGISCHE KANÄLE

CAPTURE (204)

CAPTURE FROM #Kanal TO #Kanal

CLEAR (204)

CLEAR #Kanal
CLEAR ENVELOPE
CLEAR FKEYS
CLEAR FONT
CLEAR GRAPHICS
CLEAR #Kanal:NET
CLEAR QUEUE Tonquellennummer
CLEAR SCREEN
CLEAR SOUND
CLEAR TEXT

CLOSE (204)

CLOSE #Kanal

COPY (204)

COPY FROM #Kanal TO #Kanal

FLUSH (204)

FLUSH #Kanal

REDIRECT (205)

REDIRECT FROM #Kanal TO #Kanal

OPTIONEN

DEFAULT CHANNEL (205)

SET DEFAULT CHANNEL Ausdruck

LOAD (207)
LOAD #Kanal:Dateiname
LOAD Gerätename

MERGE (207)
MERGE #Kanal:Dateiname

SAVE (207)
SAVE #Kanal:Dateiname
SAVE Gerätename
SAVE ALL #Kanal:Dateiname

VERIFY (208)
VERIFY #Kanal:Dateiname

OPTIONEN

FAST SAVE (208)
SET FAST SAVE ON/OFF

REM1 (208)
SET REM1 ON/OFF

REM2 (208)
SET REM2 ON/OFF

TAPE LEVEL (209)
SET TAPE LEVEL Ausdruck

TAPE SOUND (209)
SET TAPE SOUND ON/OFF

GRAPHICS (210)
GRAPHICS
GRAPHICS HIRES/LORES Anzahl Farben
GRAPHICS ATTRIBUTE

LOOK (210)
LOOK #Kanal AT x,y: Variable

PLOT (210)
PLOT #Kanal:Punkte-Liste
PLOT #Kanal:ANGLE Ausdruck
PLOT #Kanal:FORWARD/BACK Ausdruck
PLOT #Kanal:LEFT/RIGHT Ausdruck
PLOT #Kanal:ELLIPSE Ausdruck, Ausdruck
PLOT #Kanal:PAINT

VIDEOOPTIONEN

ATTRIBUTES (212)
SET ATTRIBUTES Ausdruck

BEAM (212)
SET #Kanal:BEAM ON/OFF

BIAS (212)
SET #Kanal:BIAS Farbcode

BORDER (213)
SET #Kanal:BORDER Farbcode.

CHARACTER (213)
SET #Kanal:CHARACTER n,a,b,c,d,e,f,g,h,i

COLOUR (213)
SET #Kanal:COLOUR Paletten-Nummer, Farbcode

CURSOR (214)
SET #Kanal:CURSOR CHARACTER Code
SET #Kanal:CURSOR COLOUR Paletten-Nummer

INK (214)
SET #Kanal:INK Farben-Nummer

LINE MODE (214)
SET #Kanal:LINE MODE Parameter

LINE STYLE (214)
SET #Kanal:LINE STYLE Parameter

PALETTE (214)
SET #Kanal:PALETTE a,b,c,d,e,f,g,h

PAPER (215)
SET PAPER #Kanal:Farben-Nummer

SCROLL (215)
SET #Kanal:SCROLL ON/OFF
SET #Kanal:SCROLL UP/DOWN n,m

STATUS (215) SET STATUS ON/OFF

VIDEO COLOUR (216) SET VIDEO COLOUR Ausdruck

VIDEO MODE (216) SET VIDEO MODE Ausdruck

TEXT und GRAPHIK-FENSTER

DISPLAY (217)
DISPLAY #Kanal:AT a FROM b TO c
DISPLAY GRAPHICS
DISPLAY TEXT

TEXT (217)
TEXT
TEXT 40
TEXT 80

VIDEO X (217) SET VIDEO X Ausdruck

VIDEO Y (217) SET VIDEO Y Ausdruck

ENVELOPE (218) ENVELOPE #Kanal:NUMBER a;b,c,d,e;f,g,h,j;...;
RELEASE j,k,l,m;...

PING (218)

SOUND (218) SOUND #Kanal:PITCH Ausdruck, DURATION Ausdruck, LEFT Ausdruck, RIGHT Ausdruck, SOURCE Ausdruck, STYLE Ausdruck, ENVELOPE Ausdruck, SYNC Ausdruck, INTERRUPT

TONOPTIONEN

SOUND BUFFER (219) SET SOUND BUFFER Ausdruck

SPEAKER (220) SET SPEAKER ON/OFF

TAPE SOUND (220) SET TAPE SOUND ON/OFF

STYLE (219) SOUND STYLE

TYPE (221)

EDITOR BUFFER (221)

EDITOR KEY (221)

EDITOR VIDEO (221)

OPTIONEN

SET EDITOR BUFFER Ausdruck

SET EDITOR KEY Kanalnummer

SET EDITOR VIDEO Kanalnummer

BETRIEBSSYSTEM

ASK (223)

ASK Betriebssystem-Variable Ziel-Variable

EXT (223)

EXT Parameter-String

SET (223)

SET Betriebssystem-Variable,Wert

TOGGLE (224)

TOGGLE Betriebssystem-Variable

MASCHINENSPRACHE

ALLOCATE (225)

ALLOCATE Ausdruck

CODE (225)

CODE = String
CODE Variablenname = String

OUT (225)

OUT n,a

POKE (225)

POKE Adresse, Wert

SPOKE (225)

SPOKE Segment, Adresse, Wert

FUNKTIONEN und VARIABLEN

BIN(X) (225)

HEX$(X$) (226)

IN(N) (226)

PEEK(N) (226)

SPEEK(S,N) (226)

USR(N,X) (226)

WORD$(N) (226)

NET CHANNEL (227) ASK NET CHANNEL Variable

NET MACHINE (227) ASK NET MACHINE Variable

NET NUMBER (227) SET NET NUMBER Ausdruck

SERIAL BAUD (227) SET SERIAL BAUD Ausdruck

SERIAL FORMAT (227) SET SERIAL FORMAT Ausdruck

ENTERPRISE
COMPUTERS GMBH
SONNENSTRASSE 3
D-8000 MÜNCHEN 2
TELEFON (0 89) 5 51 70 50
WEST GERMANY