NCR

NCR DECISION MATE V

Bedienungsanleitung

Bestellnummer 017-0032302

MS-PRODUCT	: GW (TM)-BASIC
MS VERSION NO.	: 1.4
PRODUCT NO.	: D006-0064-0101
NCR PART NO.	: 017-0031772
SERIAL NO.	: 08386

3. Auflage, Januar 1984

NCR ist ständig bemüht, die Produkte im Zuge der Entwicklung von Technologie, Bauteilen, Soft- und Firmware dem neuesten Stand anzupassen. NCR behält sich deshalb das Recht vor, Spezifikationen ohne vorherige Ankündigung zu ändern.
Nicht alle hier beschriebenen Leistungen werden von NCR in allen Teilen der Welt vertrieben. Nähere Informationen bezüglich eventueller Einschränkungen oder auch Erweiterungen sowie den aktuellen Stand erfahren Sie von Ihrem Händler oder der nächstegelegenen NCR-Geschäftsstelle.

VORWORT

Herzlichen Glückwunsch zu Ihrer Wahl des NCR DECISION MATE V. Decision Mate heißt soviel wie Entscheidungs-Helfer. Sie haben hiermit schon die erste wichtige Entscheidung für Ihren Einstieg in die Welt der Computer getroffen. Vielleicht sind Sie aber bereits erfahrener Anwender. In jedem Fall wird Sie Ihr NCR DECISION MATE V nun als professioneller Partner bei Ihrer täglichen Arbeit begleiten, Sie optimal unterstützen, und von zeitraubenden Nebentätigkeiten entlasten.

Die hohe Leistungsfähigkeit Ihres NCR DECISION MATE V wurde durch das modulare Systemkonzept und die konsequente Anwendung der neuesten Technologien erzielt. Eine große Anzahl von zusätzlichen Funktionen und ein breites Spektrum an einsetzbarer Software erlauben einen weiten Anwendungsbereich. Zudem ist Ihr NCR DECISION MATE V wegen seiner Einfachheit der Bedienung sehr benutzerfreundlich, wozu auch die ergonomisch richtige Gestaltung ganz wesentlich beiträgt.

Auch wenn Sie bereits Erfahrung im Umgang mit Computern haben, sollten Sie sich etwas Zeit nehmen, diese Anleitung zu lesen, da sie wichtige Informationen über Ihren Computer enthält.

Diese Bedienungsanleitung stellt Ihnen in fünf Kapiteln Ihren NCR DECISION MATE V vor. Die beiden ersten Kapitel machen Sie mit den Hauptelementen des Systems, dem Prozessor (Hardware) und dem Betriebssystem (Software) vertraut und beschreiben deren Installation. Kapitel 3 beschäftigt sich mit dem Betrieb, — also der Bedienung des Gerätes — sowie der Inbetriebnahme des Systems. Die letzten beiden Kapitel helfen Ihnen bei eventuell auftretenden Schwierigkeiten und geben Ihnen nützliche Empfehlungen, wie Sie den Computer pfleglich behandeln.

Nach dem Studium der Bedienungsanleitung und der Dokumentation des Betriebssystems sind Sie mit Ihrem Computer hinreichend vertraut und können bereits zahlreiche Operationen durchführen. Nun werden Sie sich für weitere Dokumentationen interessieren, die Sie in der Anwendung der Software weiterführen. Mit Hilfe dieser neu erworbenen Kenntnisse werden Sie in der Lage sein, Computersprachen und zusätzliche Betriebssyteme in Ihrem NCR DECISION MATE V einzusetzen. Hinweise dazu finden Sie im Kapitel *Systembeschreibung.*

Ihr NCR DECISION MATE V ist das Produkt eines erfahrenen Computerherstellers. Es wurde in Deutschland entwickelt und gefertigt.

Fortlaufende Forschung und Weiterentwicklungen erlauben es, Ihren Computer einfach und dem jeweiligen Einsatzzweck angepaßt auszubauen. Ihr Händler oder die nächstgelegene NCR Geschäftsstelle berät Sie hierzu gerne. Zum Abschluß möchten wir Ihnen nun viel Freude und Erfolg an Ihrem neuen Computer wünschen,

Ihre

NCR GmbH, Augsburg

NCR DECISION MATE V
BEDIENUNGSANLEITUNG

INHALT

INBETRIEBNAHME
Betriebsbereit in zwanzig Minuten ..... A
Drucker und Plotter 1-1

SYSTEMBESCHREIBUNG
Hardware 2-1
Software .....
Verbindungen innerhalb des Systems .....

BEDIENUNG DES COMPUTERS
Einführung 3-1
Disketten .....
Festplatten .....
Bedienungselemente .....
Was wird beim Einschalten geprüft? .....
Der Bildschirm .....
Die Tastatur .....
Wie kann ich Software laden? .....
Einige praktische Übungen .....

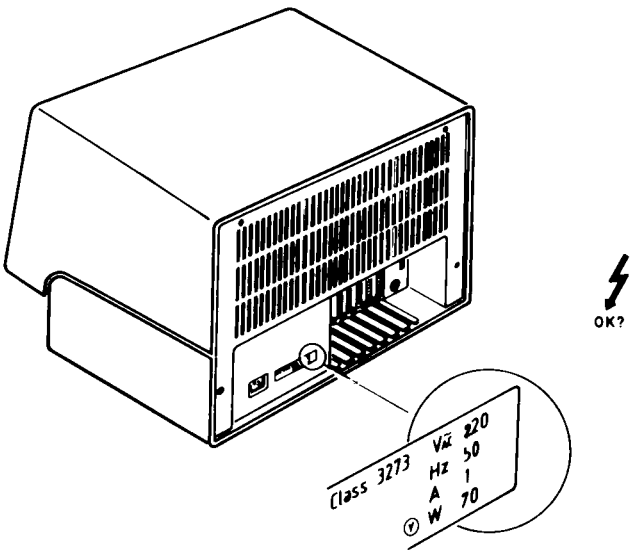
WAS NUN?
Erkennen von Störungen 4-1
Service-Vereinbarungen .....

PRAKTISCHE HINWEISE
Aufstellen des Computers 5-1
Arbeiten mit dem Computer .....
Behandlung der Disketten .....
Ortswechsel des Computers .....
Hinweise zur Pflege des Computers .....

LEISTUNGSERWEITERUNGEN 6-1

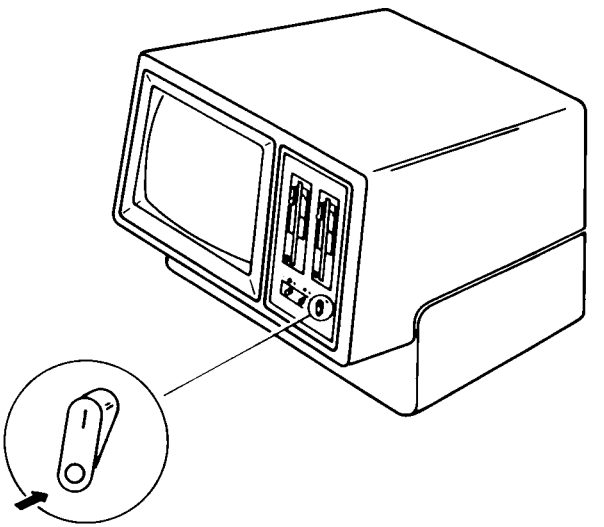
Die nachfolgenden Seiten zeigen Ihnen, wie Sie rasch und einfach Ihren NCR DECISION MATE V in Betrieb nehmen.

BETRIEBSBEREIT IN ZWANZIG MINUTEN!



Die auf dem Typenschild angegebene Spannung muß mit der an der Steckdose vorhandenen Spannung übereinstimmen.

Wenn dies nicht der Fall ist, sollten Sie sich mit Ihrem NCR-Partner in Verbindung setzen. Er wird Ihnen dann weiterhelfen.



Ihr NCR DECISION MATE V muß zunächst ausgeschaltet sein.

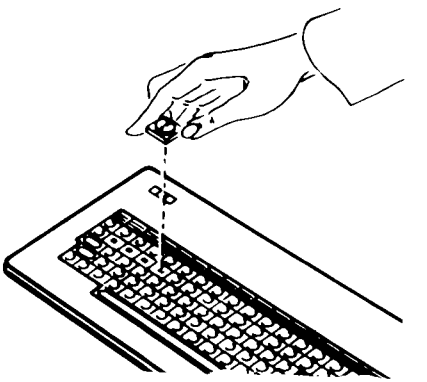
B

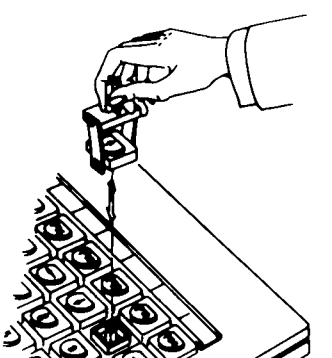
Sie haben wahrscheinlich bereits bemerkt, daß die Tastatur nicht vollständig belegt ist. Dies erklärt sich dadurch, daß der NCR DECISION MATE V in vielen Ländern der Welt eingesetzt wird. Jede einzelne Sprache erfordert einige besondere Buchstaben oder andere sprachspezifische Zeichen. Auch die deutsche Sprache enthält besondere Buchstaben (ä, ö, ü, ß).

Die fehlenden Tasten befinden sich in einem durchsichtigen Kunststoffbehälter, der mit Ihrem NCR DECISION MATE V geliefert wurde.

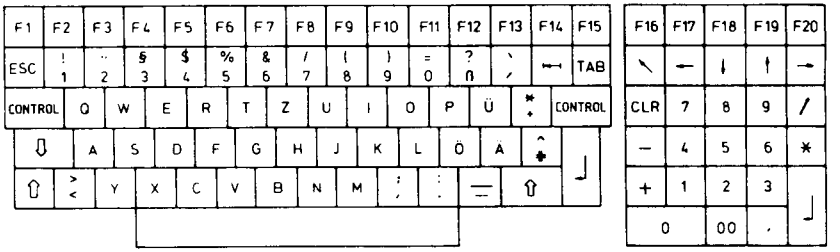
Sie müssen lediglich die einzelnen Tasten in die Tastatur mit sanftem Druck einsetzen. Die Tasten rasten hörbar ein. Die Abbildungen "Tastatur-Belegung" (siehe nächste Seite) zeigen Ihnen, an welcher Stelle auf der Tastatur sich die jeweilige Taste befinden muß.

Im Fall, daß Sie versehentlich eine Taste an der falschen Stelle einsetzen, oder daß Sie später die Tastaturbelegung für eine andere Sprache auslegen möchten, sollten Sie das kleine, mitgelieferte Werkzeug verwenden (siehe Abbildung rechts).

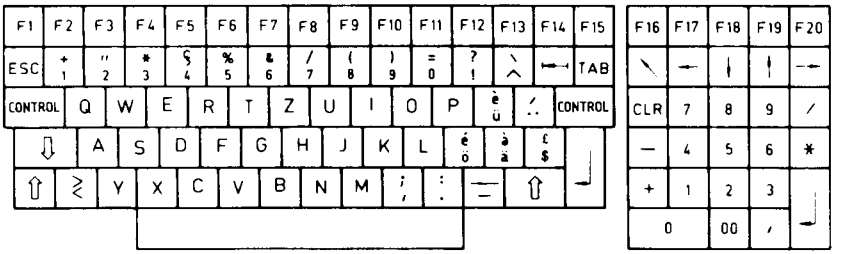




DIE TASTATUR-BELEGUNG



DEUTSCHSPRACHIGE TASTATUR



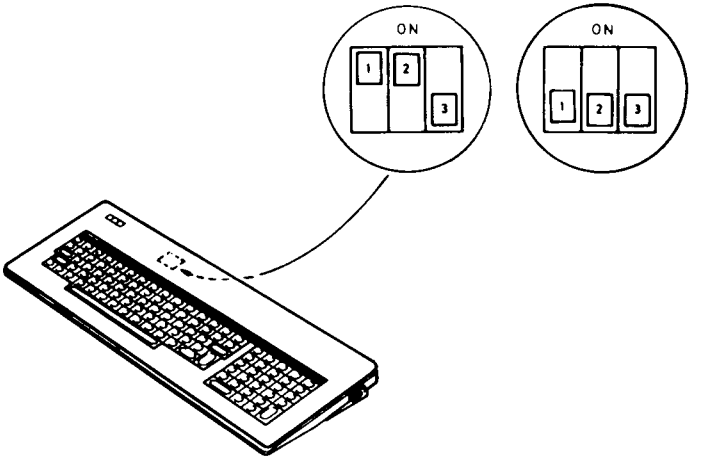
DEUTSCHSPRACHIGE TASTATUR (SCHWEIZ)

D

Sie müssen nun Ihrem NCR DECISION MATE V die Sprache mitteilen, die der von Ihnen gewählten Tastaturbelegung entspricht.

Die hierfür vorgesehenen Schalter befinden sich an der Bodenplatte der Tastatur. Diese drei roten Schalter sind mit den Nummern 1, 2 bzw. 3 gekennzeichnet. Neben den Schaltern befindet sich ein Aufkleber, der Ihnen Aufschluß über die Schalterstellungen für die verschiedenen Sprachen gibt.

Die zwei hier abgebildeten Schalterkombinationen entsprechen den zwei angebotenen deutschsprachigen Tastaturbelegungen.



Language Code

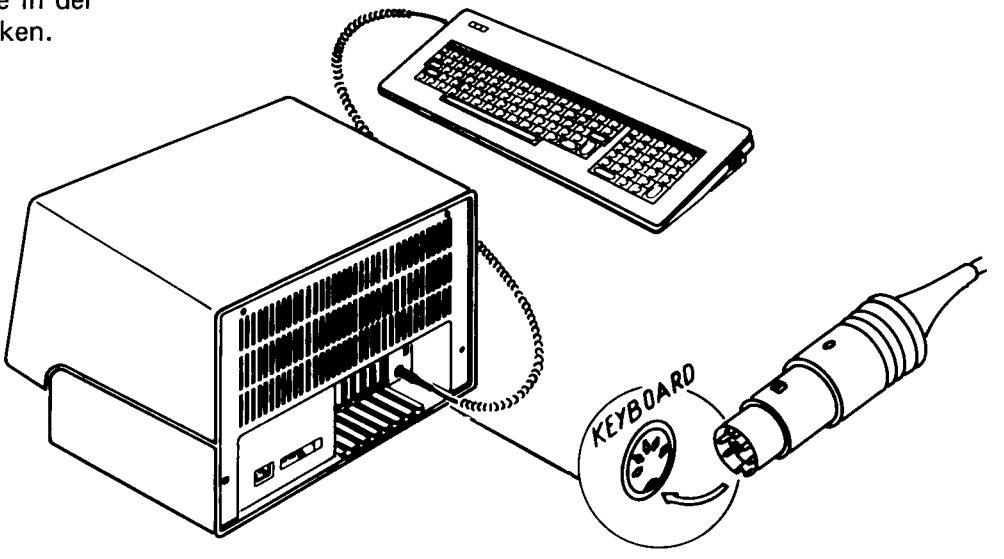
	1 2 3	
US English	○○○	Swiss-German
UK/Int. English	●○○	Swiss-French
Danish	○●○	French
German	●●○	Canadian/Australian
Swedish/Finnish	○○●	Canadian(Bilingual)
Norwegian	●○●	South African
Spanish	○●●	Portuguese
Italian	●●●	Yugoslavian
	1 2 3	

○ = off ● = on

Made in West Germany

Sie können nun die Tastatur an Ihren Computer anschließen.

Dabei müssen Sie darauf achten, daß die Stellung der Einkerbung am Stecker mit derjenigen der Buchse übereinstimmt. Die fünf Stifte lassen sich dann mühelos in die in der Buchse befindlichen Fassungen einstecken.



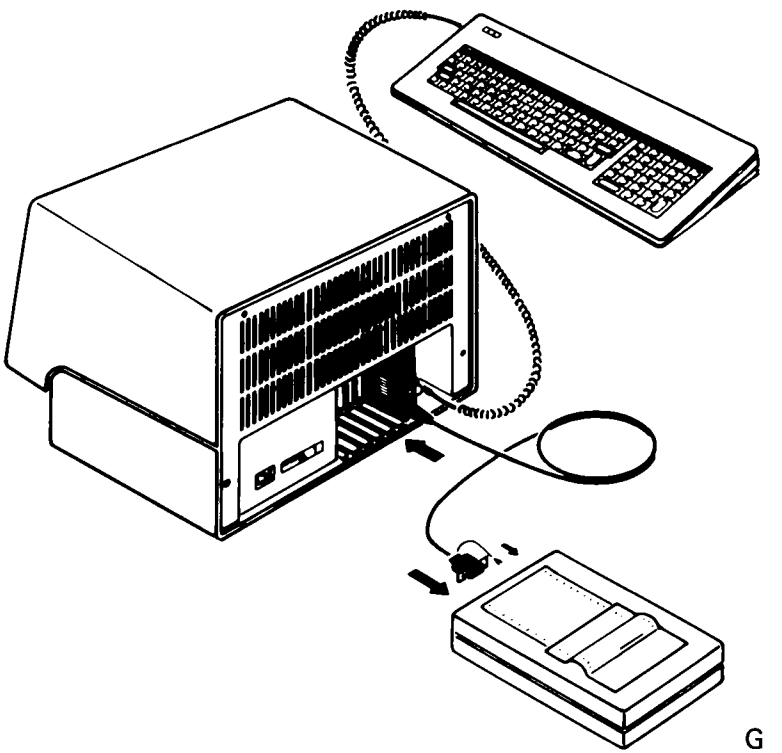
Vielleicht möchten Sie einen Drucker in Verbindung mit Ihrem NCR DECISION MATE V benutzen. Hierzu ist eine entsprechende Verbindung mittels eines Interface (Schnittstelle) herzustellen.

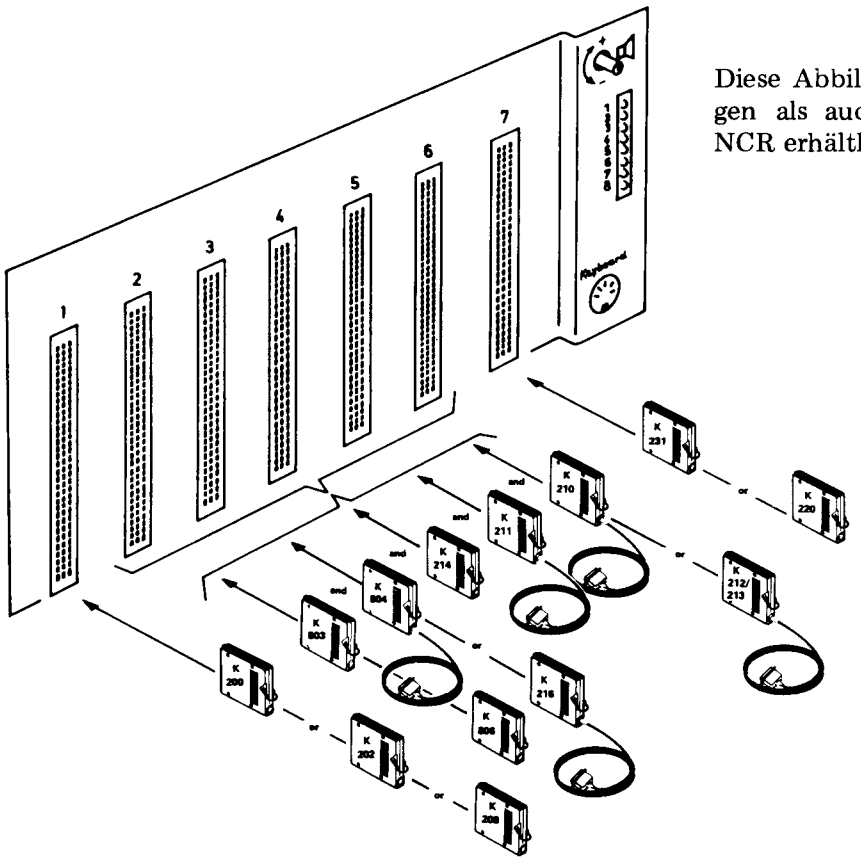
Die Schnittstelle besteht aus einem Verbindungskabel und einem Steckmodul, das Sie in eine der Fassungen 2. . . 6 an der Rückseite Ihres Computers mit mäßigem Druck einschieben müssen.

Die Art von Schnittstelle, die Sie benötigen, richtet sich nach dem Drucker, den Sie betreiben wollen. Sie müssen grundsätzlich zwischen zwei Arten unterscheiden:

- Parallel (Centronics)
- Seriell (RS-232C)

Die nachfolgende Seite zeigt Ihnen auch andere Anschlußmöglichkeiten für Ihren NCR DECISION MATE V.





Diese Abbildung zeigt Ihnen sowohl die Drucker-Verbindungen als auch die anderen Leistungserweiterungen, die von NCR erhältlich sind.

K200 64 KB-SPEICHERERWEITERUNG
K202 192 KB-SPEICHERERWEITERUNG
K208 448 KB-SPEICHERERWEITERUNG
K210 PARALLEL-SCHNITTSTELLE (CENTRONICS)
K211 RS-232C-SCHNITTSTELLE (FÜR MODEM)
K212 RS-232C-SCHNITTSTELLE (FÜR DRUCKER)
K213 RS-232C-SCHNITTSTELLE (FÜR PLOTTER)
K214 LEERADAPTER UND BUSANSCHLUSS
K215 ADAPTER FÜR SYNCHRONE/ASYNCHRONE DATEN-ÜBERTRAGUNG
K220 DIAGNOSTIK-MODUL
K231 8-/16-BIT-PROZESSORERWEITERUNG
K801 RS-232C-SCHNITTSTELLE (UMSCHALTBAR)
K803 ECHTZEIT-UHR
K804 IEEE-488-SCHNITTSTELLE
K806 MAUS-SCHNITTSTELLE

ANMERKUNG:

- K801 kann anstelle der Erweiterung K211, K212 oder K213 verwendet werden.
- Die Steckfassung 7 ist ausschließlich für das Diagnostik-Modul (K220) und die Prozessorerweiterung (K231) vorgesehen. Die Steckfassung 1 ist nur für eine Speichererweiterung zu verwenden.

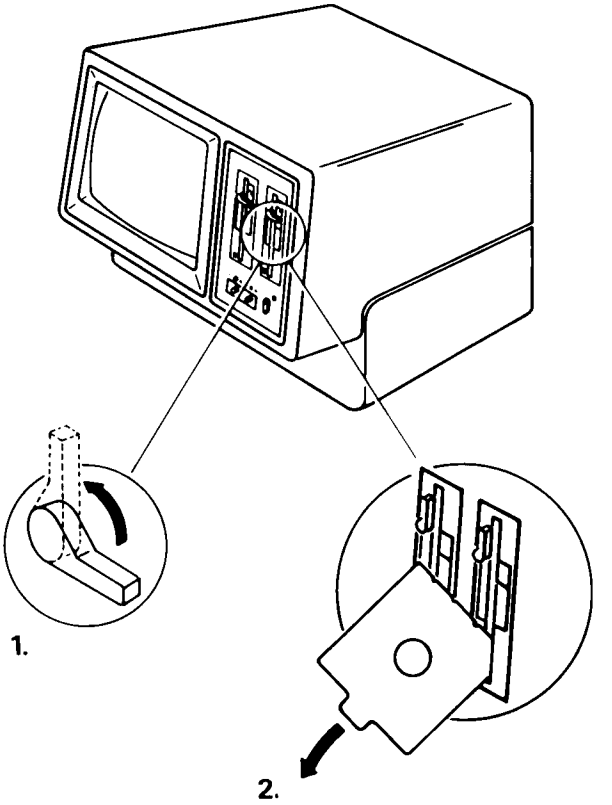
H

Jedes Disketten-Laufwerk Ihres NCR DECISION MATE V enthält bei der Lieferung eine Scheibe aus Karton. Diese dient lediglich der Transportsicherung und sollte nun entfernt werden.

Bringen Sie den Sicherungshebel in die Senkrechtstellung, indem Sie ihn nach links drehen (siehe Abbildung 1 rechts). Sie können nun die Scheibe entfernen (Abbildung 2).

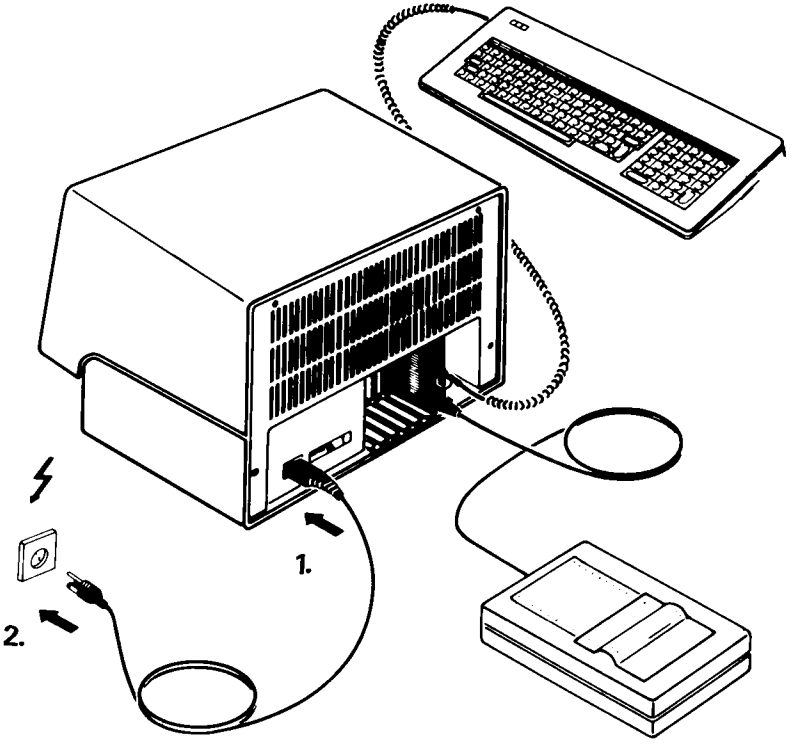
Bewahren Sie die Transportsicherungsscheiben gut auf. Bevor Sie Ihren Computer an einen anderen Ort verlegen, sollten Sie sie in die Disketten-Laufwerke wieder einlegen und die Sicherungshebel durch Rechtsdrehen in die Waagrechtstellung bringen.

Beim Entfernen und Einlegen der Transportsicherungsscheiben darf der Computer auf keinen Fall eingeschaltet sein.



Setzen Sie den Stecker des Netzkabels in die dafür vorgesehene Fassung an der Rückseite Ihres NCR DECISION MATE V ein.

Schließen Sie den Computer an die Wandsteckdose an.



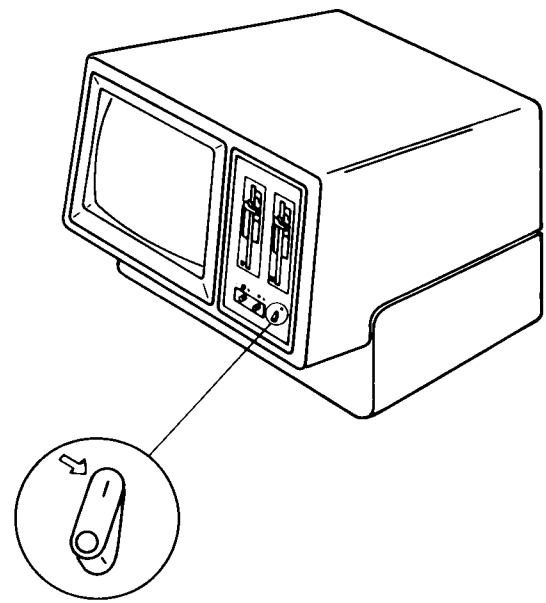
J

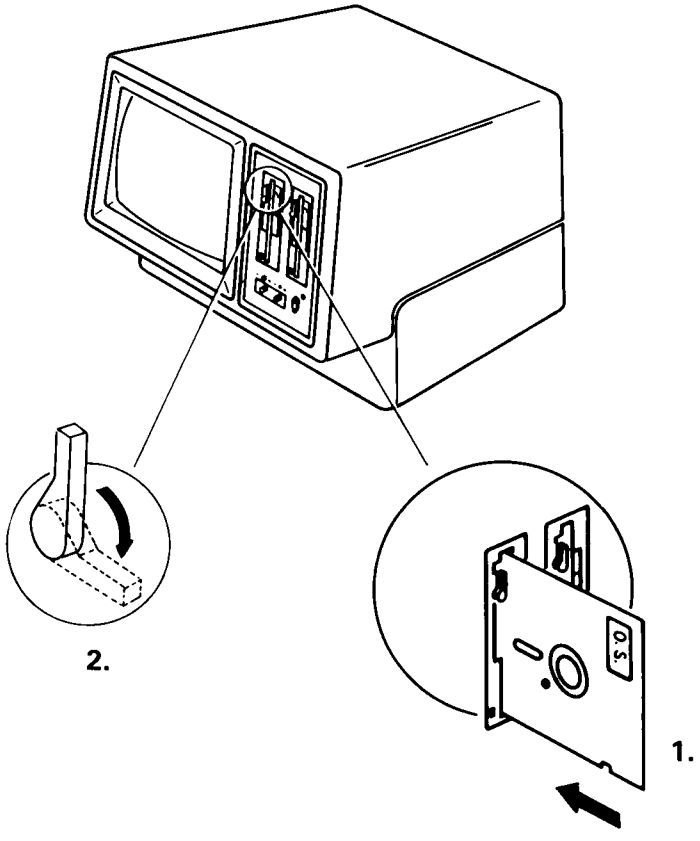
Ihr NCR DECISION MATE V ist nun betriebsbereit.

Schalten Sie ein, indem Sie die orangefarbene Taste betätigen. Ihr Computer teilt Ihnen dann mit, daß er das Einlegen einer Diskette erwartet:

DISK A: NOT READY <CR>

Diese Mitteilung erscheint in der unteren linken Ecke des Bildschirms.





2.

1.

Ihr NCR DECISION MATE V kann nun das sogenannte Betriebssystem aufnehmen. Dieses befindet sich auf einer Diskette. Ein Betriebssystem besorgt die Verständigung zwischen den Bestandteilen Ihres Systems (Bildschirm, Tastatur, Speicher, Drucker usw.) und ermöglicht den Einsatz von Anwendungs-Software (z.B. ein Programm für die Erstellung einer Geschäftsbilanz).

Nehmen Sie die Betriebssystem-Diskette aus ihrer Schutzhülle heraus. Halten Sie die Diskette zwischen Daumen und Zeigefinger am Aufklebe-Etikett. Die Diskette läßt sich mühelos in das am weitesten links befindliche Disketten-Laufwerk einlegen. Dabei ist die links abgebildete Stellung der Diskette zu beachten (Abb. 1).

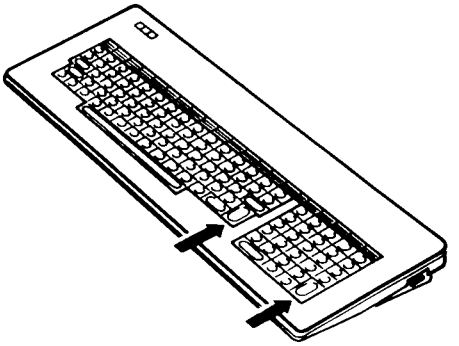
Nachdem Sie die Diskette eingelegt haben, müssen Sie den Sicherungshebel durch Rechtsdrehen in die Waagrechtstellung bringen (Abb. 2).

WICHTIG: Vor dem Einlegen einer Diskette muß der Computer bereits eingeschaltet sein.

Sie sollten eine Diskette nie knicken oder an der länglichrunden Öffnung anfassen. Sogar Fingerabdrücke, die vom menschlichen Auge nicht wahrgenommen werden, können eine Beschädigung der Speicherfläche verursachen. Es empfiehlt sich deshalb, jede Diskette in ihrer Schutzhülle aufzubewahren, wenn sie nicht gerade benötigt wird.

Die in der Abbildung mit Pfeilen gekennzeichneten Tasten erfüllen eine besondere Funktion. Je nach Ausführung der Tastatur Ihres NCR DECISION MATE V tragen diese Tasten die Bezeichnung ↵ oder NEW LINE.

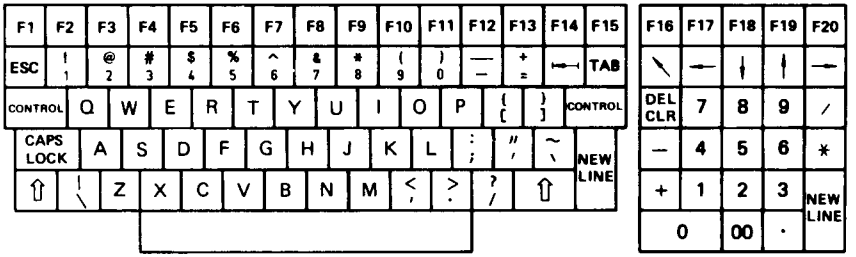
Mit dieser "Abschlußtaste" wird eine Aufforderung oder Mitteilung an den Computer Ihrerseits abgeschlossen und dem Computer übergeben.



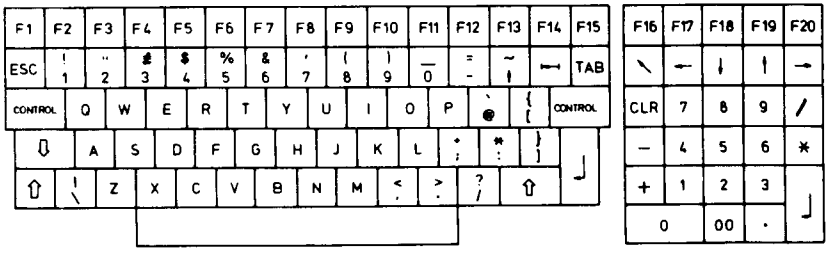
Nachdem Sie die Betriebssystem-Diskette gemäß der auf Seite L enthaltenen Beschreibung in das Disketten-Laufwerk eingelegt haben, müssen Sie jetzt nur noch eine der zwei Abschlußtasten betätigen. Das Betriebssystem wird dann von der Diskette in den Computer geladen.

Sie können nun die Handhabung des von Ihnen gewählten Betriebssystems mit Hilfe des dazugehörigen Handbuchs kennenlernen. Dieses Handbuch enthält ein Kapitel ("Die ersten Schritte" oder "Starthilfe"), in dem die Erstellung der ersten, sehr wichtigen Kopie Ihrer Betriebssystem-Diskette beschrieben ist.

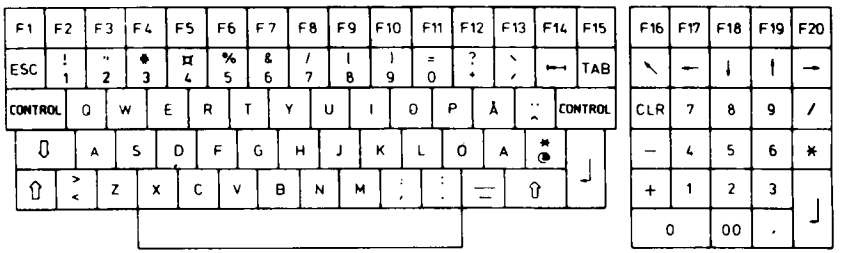
Die nachfolgenden Seiten (N. . .T) zeigen Ihnen die Tastaturbelegungen für die verschiedenen Landessprachen. Wenn Sie Ihre Tastatur für eine dieser Sprachen auslegen möchten, sollten Sie auch die Schalter an der Bodenplatte der Tastatur (siehe Seite E) entsprechend stellen.



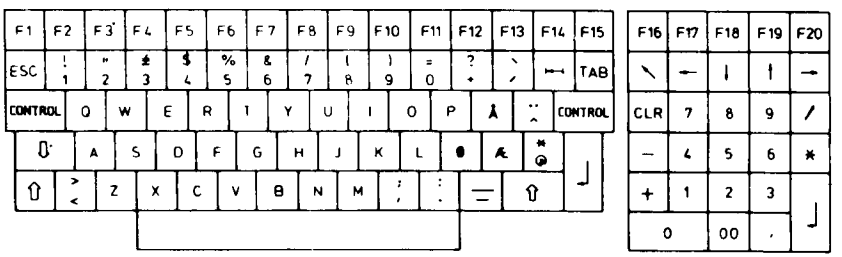
ENGLISCH (USA)



ENGLISCH (GB)

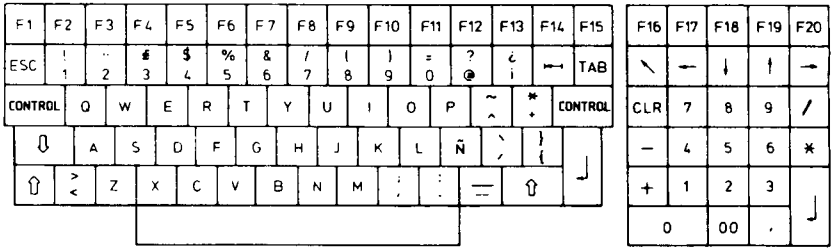


SCHWEDISH/FINNISCH

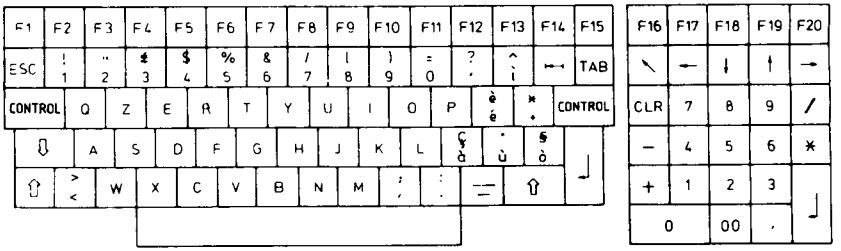


NORWEGISCH

O

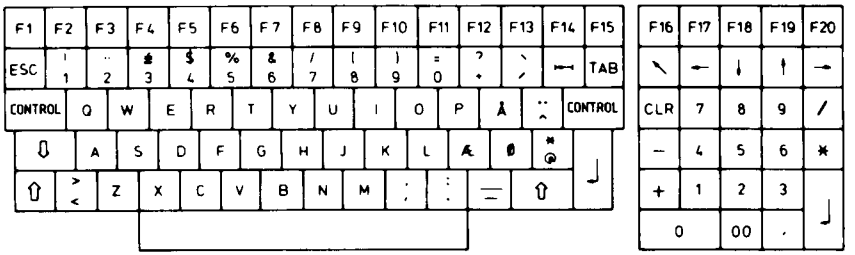


SPANISCH

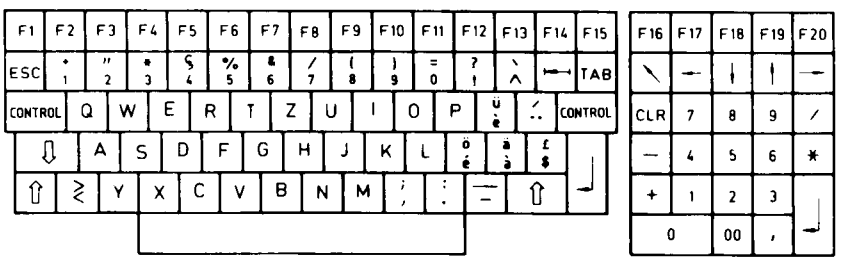


ITALIENISCH

P

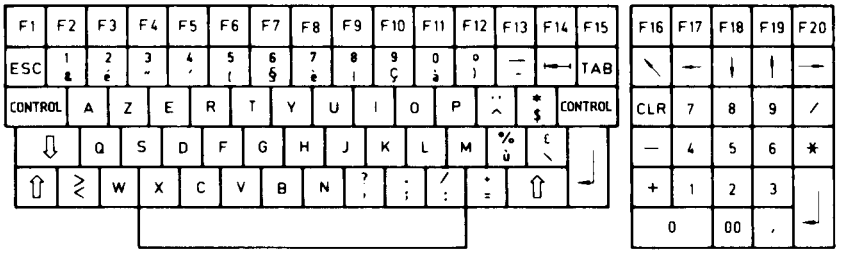


DÄNISCH

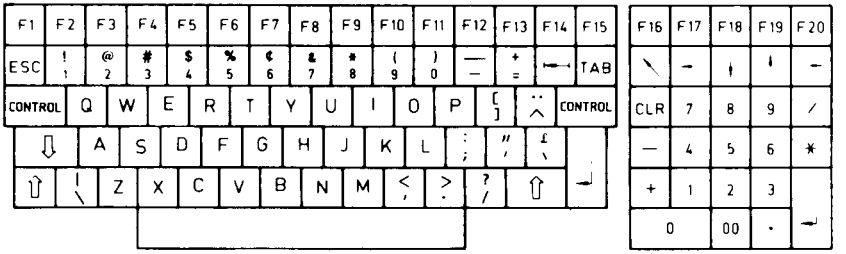


FRANZÖSISCH (SCHWEIZ)

Q



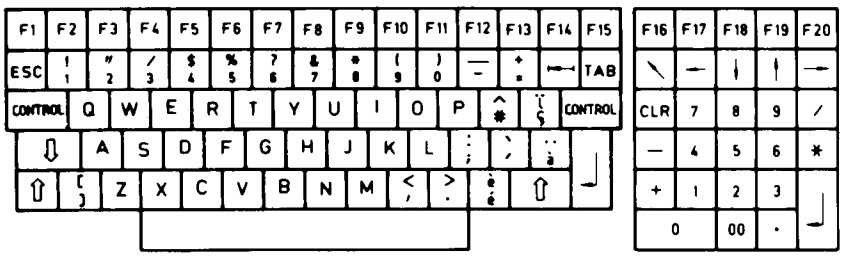
FRANZÖSISCH



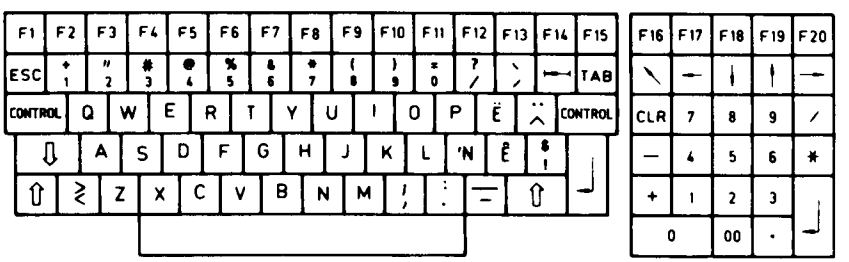
AUSTRALIEN

R

(((

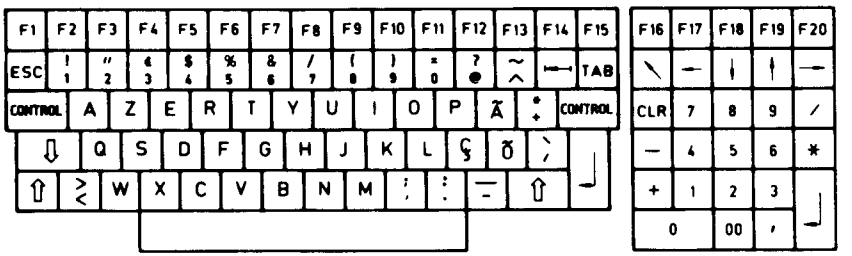


KANADA (ZWEISPRACHIG)

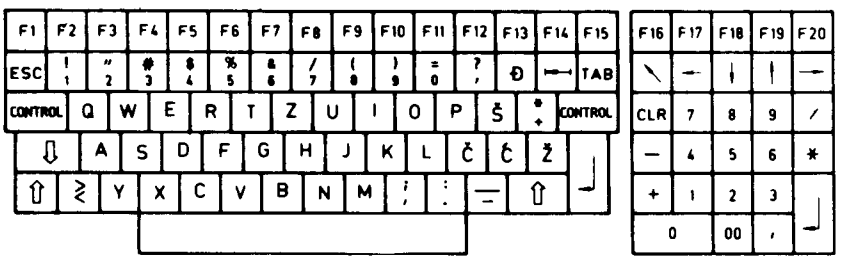


SÜDAFRIKA

S



PORTUGIESISCH



JUGOSLAWISCH

T

DRUCKER UND PLOTTER

Empfohlene Drucker und Plotter für den NCR DECISION MATE V.

- NCR 5403 (seriell) — Plotter
- NCR 6411 (seriell oder parallel) — Drucker
- NCR 6442 (parallel) — Drucker
- NCR 6455 (seriell) — Drucker

Die auf den folgenden Seiten beschriebenen Schalterstellungen der Drucker und Plotter gelten für den NCR DECISION MATE V. Weitere Informationen entnehmen Sie dem jeweiligen Drucker- bzw. Plotter-Handbuch:

- NCR 5403 Operator Instructions
- NCR 6411 Printer User Manual
- NCR 6442 Matrix Printer Operator Information
- NCR 6442 Matrix Printer Hardware Installation
- NCR 6455 Printer User Manual

Nur ein Drucker oder ein Plotter darf jeweils am System angeschlossen sein.

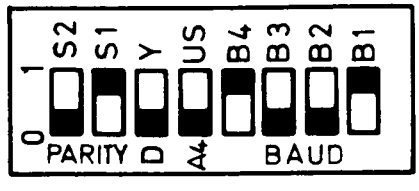
Ihre Betriebssystem-Software muß dem von Ihnen benutzten Drucker oder Plotter angepaßt werden. Der Abschnitt "Die ersten Schritte" oder "Starthilfe" Ihres Betriebssystem-Handbuchs zeigt Ihnen, wie Sie Ihre Betriebssystem-Software entsprechend anpassen können.

NCR 5403

Die Datenübertragung vom NCR DECISION MATE V erfolgt seriell über den RS-232C-Plotter-Adapter (K213). Die Schalter befinden sich am Plotter (siehe "NCR 5403 Operator Instructions").

Die für den NCR DECISION MATE V erforderlichen Stellungen der Schalter:

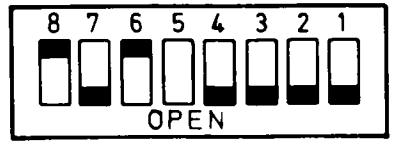
SCHALTER BEZEICHUNG	STELLUNG	FUNKTION
S2	0	Keine Prüfung auf Parity
S1	1	
Y	0	Anschluß an Computer
US	0	Papierformat: A4
	1	Papierformat: US
B4	1	Übertragungs-geschwindigkeit 4800 b/s
B3	0	
B2	0	
B1	1	



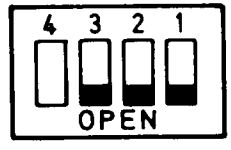
NCR 6411

Die Datenübertragung vom NCR DECISION MATE V erfolgt seriell über den RS-232C-Peripherie-Adapter (K212). Hierzu müssen Sie Schalter an vier Schalteinheiten am Drucker stellen (siehe NCR 6411 Printer User Manual).

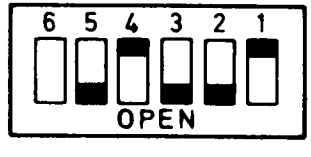
SCHALTER SW21		
SCHALTER-NUMMER	**STELLUNG**	**FUNKTION**
1	OPEN	Anzahl der Stop-Bits = 1
2	OPEN	SD
3	OPEN	Parity-Prüfung: gerade
4	OPEN	
5		Nicht benutzt
6	nicht OPEN	Zeichenlänge: 7 bit
7	OPEN	X-ON/X-OFF-Protokoll
8	nicht OPEN	



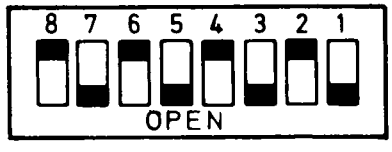
SCHALTER SW22		
SCHALTER-NUMMER	**STELLUNG**	**FUNKTION**
1	OPEN	Datenübertragungs-geschwindigkeit: 9600 b/s
2	OPEN	
3	OPEN	
4		Nicht benutzt



SCHALTER SW23		
SCHALTER-NUMMER	**STELLUNG**	**FUNKTION**
1	nicht OPEN	RS-232C
2	OPEN	
3	OPEN	DSR, RS-232C
4	nicht OPEN	
5	OPEN	
6		Nicht benutzt



SCHALTER SW24		
SCHALTER-NUMMER	**STELLUNG**	**FUNKTION**
1 2	OPEN nicht OPEN	} DTR, X-ON/X-OFF
3 4	OPEN nicht OPEN	} RTS
5 6	OPEN nicht OPEN	} CTS wird benutzt
7 8	OPEN nicht OPEN	} CD wird nicht anerkannt



Der NCR 6411 kann mit dem Centronics-Peripherieadapter (K210) als Paralleldrucker betrieben werden. In diesem Fall ist das Stellen der Schalter nicht erforderlich.

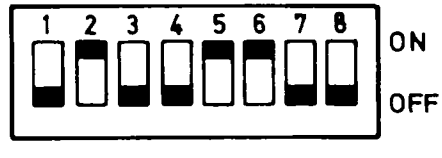
NCR 6442

Die Datenübertragung vom NCR DECISION MATE V erfolgt parallel über den Centronics-Peripherieadapter (K210). Hierzu müssen die Schalter nicht gestellt werden.

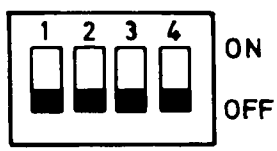
NCR 6455

Die Datenübertragung vom NCR DECISION MATE V erfolgt seriell über den RS-232C-Peripherieadapter (K212). Hierzu müssen Sie Schalter an fünf Schalteinheiten am Drucker stellen (siehe NCR 6455 Printer User Manual).

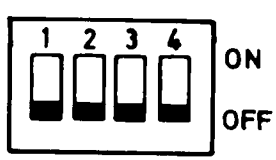
SCHALTER SW1		
SCHALTER-NUMMER	**STELLUNG**	**FUNKTION**
1	OFF	Einstellung auf externe Steuerung bereits beim Einschalten
2	ON	LF wird als Seitenvorschub anerkannt
3	OFF	einzelner horiz. Tab. zurückgesetzt
4	OFF	Auto-Return außer Kraft
5	ON	X-ON/X-OFF-Protokoll
6	ON	Interrupt-Signal wird beim Warnton gesetzt
7	OFF	Schalter für Seitenlänge außer Kraft
8	OFF	Normaler Betrieb (Fehlermonitor)



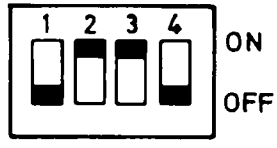
SCHALTER SW2		
SCHALTER-NUMMER	**STELLUNG**	**FUNKTION**
1 2	OFF OFF	} Konstante Druckstellenbreite
3	OFF	Normaler Betrieb (Papierende)
4	OFF	Normaler Betrieb (Test)



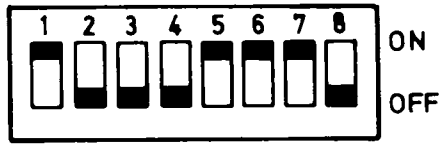
SCHALTER SW3		
SCHALTER-NUMMER	**STELLUNG**	**FUNKTION**
1	OFF	6 Zeilen/Zoll
2 3 4	OFF OFF OFF	} 10 Zeilen/Zoll



SCHALTER SW4		
SCHALTER-NUMMER	**STELLUNG**	**FUNKTION**
1	OFF	Schalter am Drucker für Zeilenvorschub außer Kraft
2 3	ON ON	} Prüfung auf Parity bei Senden/Empfangen: gerade
4	OFF	Halbduplex



SCHALTER SW1 AUF DER INTERFACE-PLATINE		
SCHALTER-NUMMER	**STELLUNG**	**FUNKTION**
1	ON	DSR eingeschaltet
2	OFF	CTS ausgeschaltet
3	OFF	Normale Prüfung v. CD (Modem)
4	OFF	Gegenreichtungskanal — active high
5	ON	Test
6	ON	2K-Puffer
7	ON	Nicht benutzt
8	OFF	Hammer aktiviert



Ferner müssen Sie anhand der am Drucker befindlichen Bedienungselemente folgende Werte einstellen:

Geschwindigkeit: 8 (= 9600 baud)
Zeilen/Seite : 72
RS-232C X-ON/X-OFF

TECHNISCHE DATEN	
Größe Computer Tastatur	Höhe 378 mm Breite 461 mm Tiefe 370 mm Höhe 37 mm Breite 430 mm Tiefe 216 mm
Gewicht Computer Tastatur	22 kg 2 kg
Netzspannung	Nennwert 100 Volt ~ Bereich 90 bis 107 Volt ~ 120 104 bis 127 220 198 bis 235 230 207 bis 246 240 216 bis 257
Netzfrequenz	50/60 Hz (49 – 61 Hz)
Leistungsaufnahme	70 Watt (nur Computer, ohne Peripherie)
Temperatur °C	Betrieb 10 bis 35, Lagerung -10 bis 50, Transport -40 bis 60
Temperaturänderung	10° C per Stunde
Relative Feuchtigkeit	20% bis 80%
Kabellängen	Netzkabel 2,5 bis 3,5 Meter Tastatur 0,5 Meter, dehnbar Centronics Peripherie-Adapter 2,0 Meter RS-232C Peripherie-Adapter 2,0 Meter
Schalldruckpegel	Bereitschaftsbetrieb 33dB(A), mit Diskette 41dB(A)
Produktsicherheit	USA UL 478 UL Canada CSA 22,2–154 Europa IEC 380, werkseitig geprüft Deutschland VDE 0806, GS-Zeichen
Funkentstörung	USA FCC Docket No. 20780, Class B Deutschland VDE 0871, Klasse A; FTZ-geprüft
Strahlung	USA Public Law 90–602 DHEW Publication No. (FDA) 75-8003 Deutschland Röntgenverordnung

SYSTEMBESCHREIBUNG

Ihr NCR DECISION MATE V besteht aus zwei zusammengehörenden Hauptelementen — der Hardware und der Software. Während die Hardware alle physikalischen Komponenten des Computers wie Prozessor und Tastatur umfaßt, besteht die Software aus verschiedenartigen Programmpaketen zum Betrieb des Computers. Dieser Abschnitt macht Sie mit der Grundausstattung der Hardware und Software bekannt und gibt Ihnen einen Überblick bezüglich zusätzlicher von NCR erhältlicher Leistungen und Erweiterungen.

HARDWARE

Die Grundausstattung des NCR DECISION MATE V besteht aus dem Prozessor und der Tastatur. Die gesamte Hardware ist nach den neuesten Konstruktionsrichtlinien für Ergonomie sowie entsprechend den Sicherheitsvorschriften konzipiert.

BASIS-KONFIGURATION

Das Kompakt-Gehäuse des Computers ist aufgrund seines Designs ideal für den Einsatz am Schreibtisch eingerichtet. Es enthält folgende wichtige Module:

- Prozessorplatine mit 64 KB Arbeitsspeicher
- 1 oder 2 Disketten-Laufwerke oder ein Disketten- und ein Festplattenlaufwerk
- Ein- oder mehrfarbiger Bildschirm, 305 mm (12 Zoll)
- Netzteil
- Steckerleisten für den Anschluß peripherer Einheiten und Leistungserweiterungen

Anmerkung: Ein KB (Kilobyte) = 1024 Zeichen (Bytes). Ein 64 KB-Speicher kann daher über 65000 Zeichen aufnehmen.

Der Prozessor ist in zwei Ausführungen lieferbar; als 8-Bit-Version für die Verwendung von 8-Bit-Software oder als 8/16-Bit-Version, um sowohl 8- als auch 16-Bit-Software verwenden zu können.

Die Eingabetastatur ist als freistehende Einheit über ein wendelförmiges, hochflexibles Kabel mit dem Prozessor verbunden. Die Tastatur verfügt über ein alphanumerisches Tastenfeld mit einigen Kontroll- und Funktionstasten und ein zusätzliches Ziffern-Tastenfeld für die schnelle Eingabe von ausschließlich numerischen Daten.

Ihr NCR DECISION MATE V verwendet Standard-5 1/4-Zoll-Disketten (130 mm) für zweiseitige Aufzeichnung mit doppelter Aufzeichnungsdichte. Dies ergibt eine Disketten-Speicherkapazität von bis zu 360 KB. Diese Disketten werden im Fachhandel oft als "double-sided, double-density"-Disketten bezeichnet.

Nur Disketten, die dieser Norm entsprechen, können eine einwandfrei Funktion gewährleisten. Sie können Ihre Disketten direkt von NCR oder einem anderen namhaften Hersteller beziehen.

Der interne Winchester-(Fest-)plattenspeicher bietet Ihnen eine noch erheblich größere Speicherkapazität von 10 MB.

Abbildung 2.1 zeigt die wichtigen Hardware-Bestandteile am Beispiel eines NCR DECISION MATE V mit einem Disketten- und einen Festplattenlaufwerk. Die Funktionsweise dieser Bestandteile wird im Kapitel 3 ("Bedienung des Computers") eingehend beschrieben.

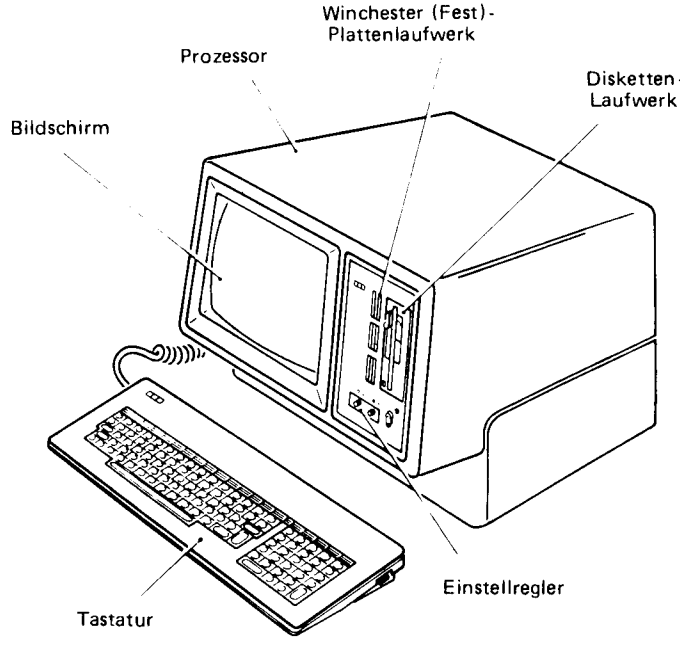


Abb. 2.1 NCR DECISION MATE V

Für technisch-wissenschaftliche Anwendungen ist eine besondere Hardware und Software-Ausstattung zu empfehlen. Diese besteht aus einem Hilfsprozessor und dem UCSD-p-Betriebssystem. Mittels dieser Ausstattung können technisch-wissenschaftliche Berechnungen besonders schnell ausgeführt werden. Dieser Einsatz Ihres NCR DECISION MATE V setzt folgende Hardware-Bestandteile voraus:

- 16-Bit-Prozessor
- Zwei Diskettenlaufwerke, oder ein Disketten- mit einem Festplattenlaufwerk

Mit Rücksicht auf dieses Einsatzgebiet des NCR DECISION MATE V werden folgende Hardware-Leistungserweiterungen angeboten:

- Echtzeit-Uhr (K803)
- IEEE-488-Interface (K804)

Ferner können bei Verwendung des UCSD-p-Betriebssystems ein RS-232C Drucker und ein RS-232C-Plotter gleichzeitig betrieben werden.

LEISTUNGSERWEITERUNGEN

Der NCR DECISION MATE V wurde für einen weltweiten Einsatz konzipiert. Mit seinen verschiedenen Standardleistungen und Zusatzpaketen, die jetzt oder später verfügbar sind, läßt er sich jeder von Ihnen gewünschten Anforderung leicht anpassen. Die gegenwärtig erhältlichen Leistungserweiterungen werden nachstehend beschrieben. Sofern keine ausdrückliche Angabe einer zu benutzenden Steckfassung vorhanden ist, können Sie für ein Steck-Modul jede beliebige Fassung 2 . . . 6 an der Rückseite des Computers verwenden.

8/16-Bit Prozessorerweiterung

Diese Prozessorerweiterung ist in zwei Ausführungen erhältlich:

- als werkseitig eingebautes Modul (F230). Sämtliche Anschlüsse befinden sich innerhalb des Computer-Gehäuses. Infolgedessen stehen Ihnen alle sieben Steckfassung an der Rückwand des Computers für anderweitige Verwendungen zur Verfügung.
- als Steckmodul (K231) für die Fassung am weitesten rechts (7) an der Rückwand des Computers.

Die Funktionsweisen dieser zwei Ausführungen sind identisch: Ihr Computer kann nun sowohl mit 8-Bit- als auch mit 16-Bit-Betriebssystemen und den mit ihnen kompatiblen Anwenderprogrammen arbeiten.

Speichererweiterungsmodule

Die von Ihnen gewünschte Speicherkapazität können Sie durch Auswahl des geeigneten Moduls bestimmen. Schieben Sie es in die Steckfassung am weitesten links (1) an der Rückwand des Computers ein. Derzeit sind drei Speichererweiterungen erhältlich:

- K200 — erweitert den Speicher von 64 KB auf 128 KB
- K202 — erweitert den Speicher von 64 KB auf 256 KB
- K208 — erweitert den Speicher von 64 KB auf 512 KB

Centronics-Peripherie-Adapter (K210)

Mit diesem Adapter können Sie einen Centronics-kompatiblen Drucker (Parallel-Schnittstelle) anschließen. Hierzu müssen Sie den Adapter in eine der Steckfasssungen (2 . . . 6) an der Rückseite Ihres NCR DECISION MATE V einschieben und das Kabel am Drucker anschließen. Der Adapter wird mit Kabel (2m) und einem geeigneten Stecker geliefert.

RS-232C-Communications-Peripherie-Adapter (K211)

Mit diesem Adapter können Sie ein Gerät anschließen, das mit serieller Datenübertragung arbeitet. Anschluß und Ausstattung wie beim Adapter K210.

RS-232C-Drucker-Peripherie-Adapter (K212)

Dieser Adpapter ist dem Adapter K211 ähnlich. Der K212 ist für den Anschluß eines RS-232C-kompatiblen Druckers zu benutzen.

RS-232C-Plotter-Peripherie-Adapter (K213)

Mit diesem Adapter können Sie einen kompatiblen Plotter (z.B. NCR 5403) an Ihren NCR DECISION MATE V anschließen.

RS-232C-Peripherie-Adapter (K801)

Sie können diesen Adapter anstelle eines K211, K212 oder K213 benutzen. Die Umschaltung muß an zwei Stellen vorgenommen werden: Sie müssen einen mitgelieferten Stecker entsprechend der gewünschten Schnittstelle in die am Adapter dafür vorgesehene Fassung einsetzten. Der Stecker bietet Ihnen drei Möglichkeiten: K801-CC1 (für eine Communications-Schnittstelle), K801-CC2

(Drucker) und K801-CC3 (Plotter). Ferner müssen Sie die vier am Adapter befindlichen Schalter gemäß Abbildung 2.2 stellen.

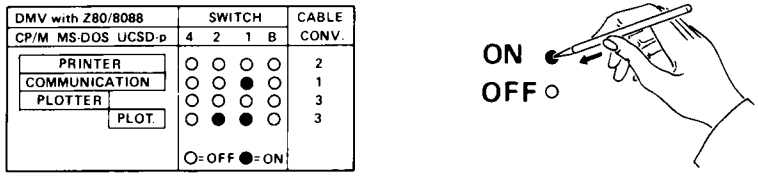


Abb. 2.2 Schalter (K801)

Anmerkungen:

1. Die Schalterstellungen für Communications und Drucker finden im Zusammenhang mit CP/M-, MS-DOS- und p-System-Software Anwendung.
2. Wenn Sie CP/M- oder MS-DOS-Software benutzen, müssen Sie beim Anschluß eines Plotters die Schalterstellungen für einen Drucker anwenden.
3. "Cable Conv." bezieht sich auf die Nummer des Steckers (CC1, CC2 oder CC3), den Sie mit der jeweiligen Stellung dieser Schalter benutzen sollten.

Falls Sie aus bestimmten Gründen einen von dieser Abbildung abweichenden Übertragungskanal (IFSEL = "Interface Select") einstellen wollen, sollten Sie Abbildung 2.3 heranziehen.

IFSEL	Schalter 4 2 1 B	Port-Adressen (hexadecimal)
0A	o o o o	60-67
0B	o o o ●	68-6F
1A	o o ● o	70-77
1B	o o ● ●	78-7F
2A	o ● o o	30-37
2B	o ● o ●	38-3F
3A	o ● ● o	B0-B7
3B	o ● ● ●	B8-BF
4A	● o o o	C0-C7
4B	● o o ●	C8-CF
o = OFF ● = ON		

Abb. 2.3 IFSEL (K801, K803)

Echtzeit-Uhr (K803)

Die Echtzeit-Uhr wird in eine der an der Rückwand Ihres NCR DECISION MATE V befindlichen Steckfassungen eingeschoben. Die Echtzeit-Uhr enthält auch einen Vierjahreskalender. Die Aufzeichnungseinheiten sind Monate, Tage, Stunden, Sekunden sowie Bruchteile einer Sekunde. Eine intern vorhandene Batterie sorgt dafür, daß die Uhr auch nach Einschalten des Computers den Zeitverlauf registriert.

Die Schalter, die sich am Gehäuse der Uhr befinden, sind gemäß nachstehender Tabelle zu stellen. Hierbei handelt es sich um die üblichen Stellungen für den Betrieb mit p-System-Software.

SCHALTER	STELLUNG
B	ON
1	OFF
2	OFF
4	ON

Abb. 2.4 Echtzeit-Uhr (Standard-Stellungen)

Sie können eine vom Standardwert (4B) abweichende IFSEL-Nummer bestimmen (siehe Abbildung 2.3).

IEEE-488-Interface-Adapter (K804)

Dieser Adapter beinhaltet eine Schnittstelle zwischen Ihrem NCR DECISION MATE V und Meßgeräten, die der IEEE-488-Norm entsprechen. Sie können bis zu 15 Geräte in einer Ketten- (daisy chain) oder Stern-Konfiguration anschließen. Die Verkabelung darf nicht länger als 20 m sein. Ferner darf die Kabellänge zwischen zwei Geräten nicht mehr als 2 m betragen.

Sie können zusätzlich einen im Fachhandel erhältlichen Adapter und ein Kabel anschließen, damit Sie Ihr Interface-Modul mit Geräten benutzen können, die den IEC-625-Normen entsprechen.

Das Modul K804 unterstützt die Controller-, Listener- und Talker-Funktionen.

Die Schalter, die sich am Gehäuse des Interface-Adapters befinden, sind gemäß Abbildung 2.5 zu stellen. Hierbei handelt es sich um die üblichen Stellungen für den Betrieb mit p-System-Software.

SCHALTER	STELLUNG
A	OFF
1	OFF
2	ON
4	OFF

Abb. 2.5 IEEE-488-Interface (Standard-Stellungen)

Sie können eine vom Standardwert (2B) abweichende IFSEL-Nummer bestimmen (siehe Abbildung 2.6).

IFSEL	Schalter 4 2 1 A	Port-Adressen (hexadecimal)
0A	o o o ●	60-67
0B	o o o o	68-6F
1A	o o ● ●	70-77
1B	o o ● o	78-7F
2A	o ● o ●	30-37
2B	o ● o o	38-3F
3A	o ● ● ●	B0-B7
3B	o ● ● o	B8-BF
4A	● o o ●	C0-C7
4B	● o o o	C8-CF
o = OFF ● = ON		

Abb. 2.6 IFSEL (K804)

Maus-Adapter (K806)

Dieser Adapter stellt die Verbindung zwischen Ihrem NCR DECISION MATE V und einer Maus her. Sie können mit diesem Adapter die folgenden Mäuse benutzen:

- Alps, Encoder-Mouse
- Hawley, Mark II
- Depraz, Souris P4
- Microsoft Mouse
- Logitech P4 oder LM-P-5
- Mouse Systems, Quad Mouse

Der Übertragungskanal (IFSEL-Nummer) läßt sich mit sieben Schaltbrücken innerhalb des Adapter-Gehäuses einstellen (siehe Abbildung 2.7). Mit Rücksicht auf die anderen Leistungserweiterungen Ihres NCR DECISION MATE V ist die IFSEL-Nummer 2A als Standardwert zu betrachten.

IFSEL	Schaltbrücken 1	2	3	4	5	6	7	Port-Adressen (hexadezimal)
0A	●	o	o	o	o	●	o	60-67
0B	●	o	o	o	o	o	●	68-6F
1A	o	●	o	o	o	●	o	70-77
1B	o	●	o	o	o	o	●	78-7F
2A	o	o	●	o	o	●	o	30-37
2B	o	o	●	o	o	o	●	38-3F
3A	o	o	o	●	o	●	o	B0-B7
3B	o	o	o	●	o	o	●	B8-BF
4A	o	o	o	o	●	●	o	C0-C7
4B	o	o	o	o	●	o	●	C8-CF

o = offen (open) ● = geschlossen (closed)

Abb. 2.7 IFSEL (Maus)

Drei weitere Schalter (J8, J9, J10) sind gemäß Abbildung 2.8 zu stellen:

Schalter	Maus
J8, J9, J10 – OFF	Hawley, Alps, Microsoft, Mouse Systems
J8, J9, J10 – ON	Depraz, Logitech

Abb. 2.8 Auswahlschalter (Maus)

Adapter für synchrone/asynchrone Datenübertragung mit Zwischenspeicher (K215)

Dieser Adapter stellt die Verbindung zwischen Ihrem NCR DECISION MATE V und einem RS-232C-Modem her. Ein 2KB-Speicher (Buffer) ermöglicht eine hohe Übertragungsgeschwindigkeit.

Numerischer Hilfsprozessor (K232)

Sie können dieses Zusatzmodul in Verbindung mit dem 16-Bit-Prozessor (F230 oder K231) benutzen. Der Hilfsprozessor ermöglicht eine besonders hohe Rechengeschwindigkeit bei arithmetischen, trigonometrischen, logarithmischen und Exponentialbefehlen. Der Einbau des Hilfsprozessors setzt Erfahrung im Umgang mit modernen Halbleiter-Bauteilen voraus.

Diagnostik-Modul (K220)

Dieses Modul wurde für Techniker zwecks Diagnose etwaiger Fehlerzustände in der Hardware entwickelt. Nachdem Sie es in die Fassung 7 eingeschoben haben, können Sie zahlreiche Funktionsprüfungen durchführen. Der Spannungspegel der Logik-Signale (+5 und +12Vdc) wird ebenfalls geprüft; es wird angezeigt, ob diese Spannungen innerhalb des zulässigen Bereichs liegen. Die Serviceanleitung für den NCR DECISION MATE V enthält eine vollständige Beschreibung.

Kippvorrichtung (K240)

Sie können diese Vorrichtung an die Bodenplatte des Gehäuses befestigen. Sie haben dann die Möglichkeit, Ihren NCR DECISION MATE V in eine Schrägstellung (7°)zu bringen.

Anschluß an Decision Net (K600)

Mit diesem Adapter können Sie Ihren NCR DECISION MATE V an das Netzwerk Decision Net anschließen.

Schutzsperre für Adapter (K880)

Diese Sperre verhindert das Entfernen der Zusatzmodule, die Sie in die Steckfassungen an der Rückseite Ihres NDR DECISION MATE V eingesetzt haben.

Ein zusätzliches integriertes Diskettenlaufwerk (K018, K019)

Diese Zusatzmoduln erweitern die Plattenspeicherkapazität eines Systems mit einem Diskettenlaufwerk. Der einzige Unterschied zwischen K018 und K019 besteht darin, daß die Farbgestaltung des Gehäuses gemäß DIN (K018) und die Farbgestaltung gemäß dem Corporate Appearance Plan (CAP) nicht identisch sind.

Festplatten (Winchester Disk)-Laufwerke (NCR 3282)

Sie können bis zu drei externe Festplattenlaufwerke an Ihren NCR DECISION MATE V als Ketten-Konfiguration anschließen. Dies setzt voraus, daß Ihr NCR DECISION MATE V kein eingebautes

Festplattenlaufwerk beinhaltet. Die Speicherkapazität eines Festplattenlaufwerks nach Formatierung beträgt 10 MB.

Das Festplattenlaufwerk NCR 3282 ist in drei Ausführungen erhältlich:

- Als alleinstehendes Festplattenlaufwerk beinhaltet es eine eigene Plattensteuerung und einen Schnittstellen-Adapter für den Anschluß an den NCR DECISION MATE V. Bei der Benutzung dieser Ausführung können keine weiteren Festplattenlaufwerke angeschlossen werden.
- Als Hauptgerät beinhaltet es zusätzliche Hardware, die den Anschluß von bis zu zwei zusätzlichen Geräten in einer Ketten-Konfiguration ermöglicht.
- Als Zusatzgerät besitzt es weder eine eigene Plattensteuerung noch einen Schnittstellen-Adapter. Es kann in Verbindung mit einem weiteren Zusatzgerät mit einem Hauptgerät oder allein mit einem Hauptgerät betrieben werden.

Die hier beschriebenen Ausführungen werden mit Gehäuse, Netzteil, Kühlventilator, Netzkabel sowie den für die problemlose Aufstellung als Ketten-Konfiguration erforderlichen Datenkabeln geliefert.

Die Aufstellung erfordert keine umständliche Vorarbeit: Sie müssen lediglich die Laufwerke anschließen und (sofern Sie mehr als ein externes Laufwerk benutzen wollen) den mit einem Widerstand ausgerüsteten Endstecker in das Endgerät der Kette einsetzen. Sie können dann die Geräte einschalten.

Printer/Plotter

Sie können beinahe jeden RS-232C oder Centronics-kompatiblen Drucker oder Plotter an Ihren NCR DECISION MATE V anschließen. Die folgenden Geräte sind besonders zu empfehlen:

- NCR 6411, Drucker (RS-232C oder Centronics)
- NCR 6442, Drucker (Centronics)
- NCR 6445, Drucker (RS-232C)
- NCR 5403, Plotter (RS-232C)

SOFTWARE

Im allgemeinen arbeitet ein Computer mit drei Arten von Software: Betriebssystem, Programmsprachen und Anwenderprogrammen. Die folgenden Definitionen sollen diese näher erläutern:

- Das Betriebssystem steuert den internen Ablauf im Computer, z.B. den Datenfluß zwischen den Funktionseinheiten untereinander, sowie die Organisation und Zuordnung des Speichers und der Plattenlaufwerke.
- Programmsprachen enthalten eine Reihe von Programmbefehlen und deren Interpretation. Sie sind die Grundlage für Computerprogramme.
- Anwenderprogramme gibt es für eine Fülle von Applikationen. Der Benutzer kann sie fertig kaufen oder auch auf seinen speziellen Einsatzzweck hin selbst entwickeln.

DAS BETRIEBSSYSTEM

Der wichtigste Teil der Software ist das Betriebssystem, denn ohne dieses kann der Computer nicht betrieben werden. Neben Kontrollprogrammen beinhaltet das Betriebssystem alle Informationen zur Verbindung der Hardware mit der Software. Das Betriebssystem befindet sich normalerweise auf einer Diskette. Derzeit sind viele Standardbetriebssyteme verfügbar. Diese Programmpakete sind jeweils für bestimmte Mikroprozessoren geschrieben und deshalb manchmal als 8-Bit oder als 16-Bit-Software bezeichnet. Für jedes dieser Betriebssysteme sind meist einige Anpassungen an die jeweilige Hardware erforderlich. Vier sehr weit verbreitete Betriebssysteme stehen Ihrem NCR DECISION MATE V zur Verfügung: CP/M-80, CP/M-86, MS-DOS, und das UCSD-p-System.

Wenn Ihr NCR DECISION MATE V einen 8-Bit-Prozessor verwendet, können Sie das CP/M-80-Betriebssystem einsetzen. Falls Sie sich für einen NCR DECISION MATE V mit einem 16-Bit-Prozessor entschieden haben oder eine Prozessorerweiterung benutzen (F230 oder K231), können Sie jedes beliebige der vier Betriebssysteme einsetzen.

CP/M ist eine Abkürzung für "Control Program for Microprocessors." CP/M-80 wurde von Digital Research, Inc. als Betriebs-

CP/M-86 , ebenfalls von Digital Research entwickelt, ist ein Betriebssystem für 16-Bit-Prozessoren. Diese Software ist vollkommen kompatibel mit CP/M-80 und kann alle Dateien ohne Umwandlung weiterbenutzen. Die Verwendung von CP/M-80-kompatiblen 16-Bit-Anwenderprogrammen ist ebenfalls möglich.

MS-DOS ist ein Betriebssystem, das von der Microsoft Corporation für 16-Bit-Prozessoren entwickelt wurde. Dieses Programm arbeitet mit einer anderen Dateienstruktur als CP/M, ist aber mit dem IBM Dateienformat kompatibel und erlaubt, wie CP/M-86, die Verwendung einer Reihe von 16-Bit-Anwenderprogrammen.

Bei Verwendung des UCSD-p-Systems können Sie UCSD-Pascal, Fortran-77 sowie BASIC in Ihrem NCR DECISION MATE

V einsetzten. Die vollständige Software-Kompatibilität mit dem IBM-PC UCSD-p-System ist gewährleistet. Das UCSD-p-System eignet sich besonders für CAD-Andwendungen, Grafik, Programmentwicklung, technisch-wissenschaftliche Aufgaben sowie für den Unterricht.

Diese Betriebssysteme wurden für den NCR DECISION MATE V nicht nur wegen ihrer hervorragenden Leistungen ausgewählt, sondern auch wegen des sehr großen Angebots an geeigneten 8- und 16-Bit-Anwenderprogrammen.

ANWENDUNGS-PROGRAMME/PROGRAMMSPRACHEN

Im Fachhandel ist bereits eine große Anzahl von Anwenderprogrammen erhältlich. Diese Pakete, die von verschiedenen Softwarehäusern angeboten werden, bieten Ihnen bestimmte Leistungen und Funktionen an, um Ihnen die Mühe und den Zeitaufwand für die Erstellung eigener Programme zu ersparen. Grundsätzlich müssen Sie nur noch die Programmdiskette in das Laufwerk legen und ein paar Hardwareparameter definieren. Sie können dann das Programm ausführen.

Aus der Tatsache, daß ein Programm in Ihrem Computer "läuft", kann man nicht folgern, daß dieses Programm eine optimale Ausnutzung der Leistungen Ihres NCR DECISION MATE V darstellt. Aus diesem Grunde hat NCR mehrere Anwenderprogramme gestestet. Eine Auswahl der getesteten Programme wird von NCR empfohlen und auch angeboten.

Neben dieser Anwendersoftware stehen Ihnen auch Programm-Sprachen zur Verfügung, damit Sie eigene Programme entwickeln können.

ANPASSUNG DER SOFTWARE

Die Leistungserweiterungen K215, K801, K803, K804 und K806 können mit jedem beliebigen Datenübertragungskanal (IFSEL 0A, 0B, 1A, 1B, 2A, 2B, 3A, 3B, 4A oder 4B) eingesetzt werden. Der in der jeweiligen Beschreibung angegebene Standardwert bezieht sich auf die Voreinstellung der dazugehörigen, von NCR angebotenen Software. Sie können die Schalter am Adapter und die Software auf andere IFSEL-Nummern einstellen. Damit sind auch zukünftige Leistungserweiterungen Ihres NCR DECISION MATE V berücksichtigt. Wenn Sie eine neue IFSEL-Nummer an einem Adapter einstellen, müssen Sie dafür sorgen, daß die von der Software benutzten Port-Adressen der neuen IFSEL-Nummer entsprechen.

Eine Neueinstellung der IFSEL-Nummer ist bei normaler Verwendung Ihres Computers und seiner Zusatzgeräte kaum erforderlich. Wenn Sie aber eigene Programme zur Steuerung dieser Geräte schreiben wollen, sollten Sie die Zuteilung der Port-Adressen flexibel gestalten, damit auch diese Software die Leistungsfähigkeit Ihres NCR DECISION MATE V voll ausnutzen kann.

Abbildung 2.9 ist besonders für Programmierer von Interesse. Sie beinhaltet eine Aufstellung der IFSEL-Nummern und der Port-Adressen, über die sie verfügen. Die für die IFSEL-Nummern vorgesehenen Leistungserweiterungen sind ebenfalls angegeben. Bei den obengenannten Leistungserweiterungen handelt es sich, wie bereits erwähnt, nicht um verbindliche Werte.

Leistungserweiterung	IFSEL									Steck-Fassung		
	0A	0B	1A	1B	2A	2B	3A	3B	4A	4B		DMA-KANAL
DRUCKER Seriell/parallel K212/K213	■	■									2-6	
PLOTTER K213 (oder K801)	■	■									2-6	
COMMUNICATIONS K211			■	■							2-6	
RS-232C (umschaltbar) K801	■	■	■	■	■	■	■	■	■	■	2-6	
PLOTTER K801 (oder K213)	■			(mit CP/M oder MS-DOS)							2-6	
PLOTTER K801					(mit p-System)		■				2-6	
SYNCHR./ASYNCHR. ÜBERTRAGUNG K215			■								2-6	
ECHTZEIT-UHR K803										■	2-6	
IEEE-488 K804						■					2-6	
MAUS K806					■						2-6	
DECISION NET K600								■			2-6	1
FESTPLATTE (extern) NCR 3282									■		2-6	
FESTPLATTE (intern)									■			
PORT-ADRESSE von	60H	68H	70H	78H	30H	38H	80H	88H	C0H	C8H		
bis	67H	6FH	77H	7FH	37H	3FH	87H	8FH	C7H	CFH		

■ Software und Hardware
■ nur von der Hardware benötigt

Abb. 2.9 IFSEL-Zusammenfassung

BEDIENUNG DES COMPUTERS

EINFÜHRUNG

Sie haben Ihren NCR DECISION MATE V bereits aufgestellt und ggf. die Leistungserweiterungen angeschlossen.

Dieses Kapitel beschreibt die Bedienung der Hauptbestandteile des Systems:

- Disketten und Diskettenlaufwerke
- Winchester (Fest-)plattenlaufwerk
- Bedienungselemente am Computer
- Bildschirm
- Tastatur

Nachdem Sie diese Abschnitte gelesen haben, können Sie das Laden Ihrer Betriebssystem-Diskette vornehmen. Dieses Verfahren wird im Abschnitt "Wie kann ich Software laden?" beschrieben. Am Ende dieses Kapitels finden Sie einige praktische Übungen.

DISKETTEN

Der Computer arbeitet mit zweiseitig beschreibbaren 5 1/4-Zoll-Disketten zur Speicherung des Betriebssystems und anderer Dateien. Zur näheren Betrachtung ziehen Sie die Diskette vorsichtig aus ihrer Schutzhülle heraus; die Abbildung 3.1 zeigt Ihnen die einzelnen Merkmale der Diskette.

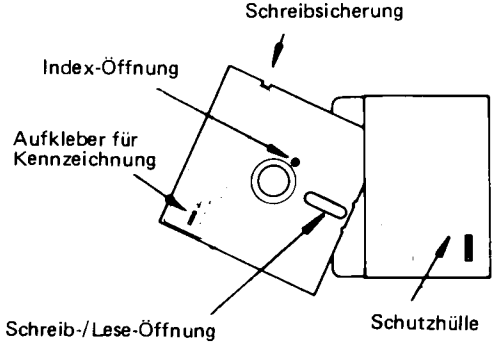


Abb. 3.1 Diskette

HANDHABUNG DER DISKETTEN

Die Diskette sollte sich immer in ihrer Schutzhülle befinden, wenn sie nicht gerade im Laufwerk benutzt wird. Zur Aufbewahrung mehrerer Disketten empfehlen sich die hierfür erhältlichen speziellen Sammelboxen.

Halten Sie die Disketten von Magnetfeldern wie Motoren, Transformatoren oder sonstigen Magneten fern. Die Disketten sind ferner vor Staub- und Feuchtigkeit zu schützen.

Disketten sollen möglichst im gleichen Umgebungsklima wie der Computer selbst aufbewahrt werden. Bei extremen Unterschieden ist es zweckmäßig, die Disketten vor Betrieb etwa eine Stunde lang am Standort des Computers zu lagern. Disketten, die wichtige Daten wie Betriebssystem oder Anwenderprogramme beinhalten, sollten mit einer Schreibsicherung versehen werden. Diese Schreibsicherung wird dadurch erreicht, daß Sie einen kleinen (mit Disketten meistens mitgelieferten) Aufkleber über der Einkerbung für die Schreibsicherung anbringen. Auf diese Weise sorgen Sie dafür, daß die Diskette im Computer nicht versehentlich gelöscht wird. Die Betriebssystemdiskette, die Sie von NCR erhielten, ist mit einem permanenten Schreibschutz versehen.

DISKETTENLAUFWERKE

Ihr NCR DECISION MATE V ist mit einem oder zwei Laufwerken für 5 1/4-Zoll-Disketten ausgestattet. Das linke Laufwerk wird mit "A", das rechte mit "B" bezeichnet. Bei den Ausführungen mit einem Diskettenlaufwerk erhält dieses die Bezeichnung "A". Hier handelt es sich um die Bezeichnungen, die die Betriebssysteme den Laufwerken zuteilen, wobei das UCSD-p-System eine Ausnahme bildet: Das linke Laufwerk wird als "4" und das rechte Laufwerk als "5" bezeichnet.

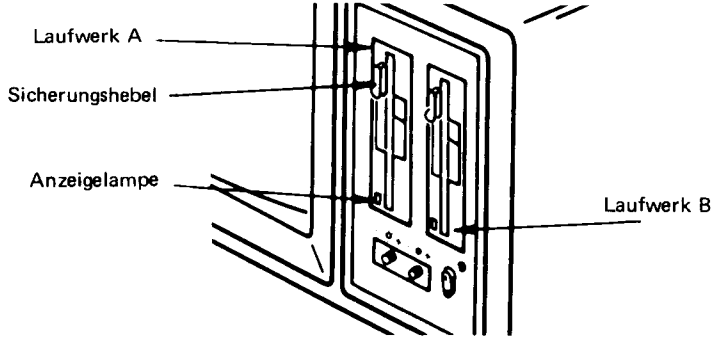


Abb. 3.2 Diskettenlaufwerke

Jedes Laufwerk hat eine Öffnung zur Aufnahme der Diskette, einen Sicherungshebel und eine rote Anzeigelampe (LED), siehe Abbildung 3.2. Die rote Anzeigelampe leuchtet immer dann auf, wenn auf die Diskette geschrieben oder von ihr gelesen wird. Es kann jeweils nur ein Laufwerk im Computer aktiv sein.

DAS EINLEGEN VON DISKETTEN

Um eine Diskette in ein Diskettenlaufwerk einzulegen, verfahren Sie wie folgt:

- Öffnen Sie das Laufwerk, indem Sie den Verschlußhebel nach links drehen. Er befindet sich dann in einer Senkrechtstellung (Abbildung 3.2).
- Entfernen Sie die Diskette aus der Schutzhülle.
- Schieben Sie die Diskette in das Laufwerk ein. Dabei zeigt die Schreib-/Lesekopföffnung voraus und der Aufkleber in Richtung Bildschirm (siehe Abbildung 3.3).
- Drehen Sie den Verschlußhebel nach rechts

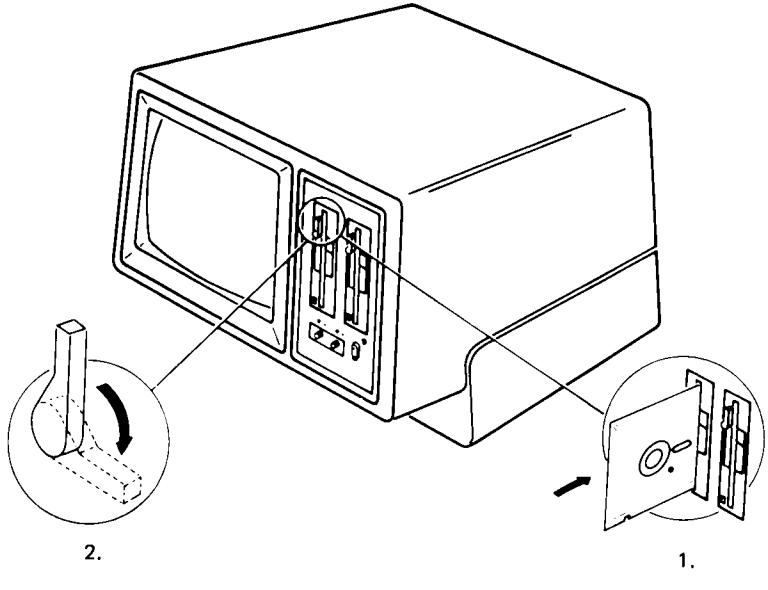


Abb. 3.3 Laden einer Diskette

Die Diskette ist nun im Laufwerk betriebsbereit. Die rote Anzeigelampe leuchtet allerdings erst dann auf, wenn das Betriebs-

system einen Schreib- oder Lesezugriff auf die Diskette vornimmt.

Wenn Sie die Diskette herausnehmen wollen, müssen Sie folgendes beachten:

- Warten Sie, bis die rote Lampe erlischt.
- Drehen Sie den Verschlußhebel nach rechts.
- Entfernen Sie die Diskette aus dem Laufwerk und legen Sie sie in ihre Schutzhülle.

WICHTIG

Nehmen Sie die Diskette nie vor Abschluß des Arbeitsvorgangs und dem Erscheinen des Bereitschaftszeichens des Betriebssystems (z.B. A>) aus dem Laufwerk.

FESTPLATTENSPEICHER

Im Prinzip arbeiten Systeme mit Festplatten (Winchester)-Laufwerken (siehe Abbildung 3.4) wie diejenigen mit zwei Disketten. Die Winchesterplatte ist ein Massenspeicher mit einer Kapazität von 10 MB und erfordert keine besondere Berücksichtigung in der Handhabung. Der Bereitschaftszustand des Laufwerks wird durch eine rote Leuchtdiode angezeigt. Bie Geräten mit Festplatte ist das Diskettenlaufwerk rechtsseitig angeordnet und mit "A" bezeich-

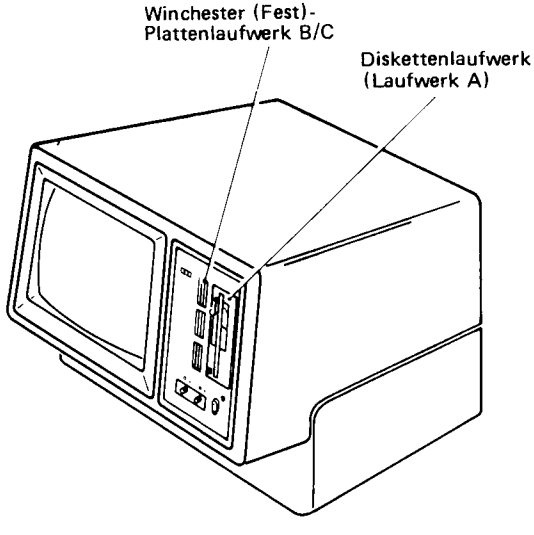


Abb. 3.4 System mit Festplatte

net. Das Winchesterlaufwerk enthält zwei Platten, die "B" und "C" genannt werden.

BEDIENUNGSELEMENTE

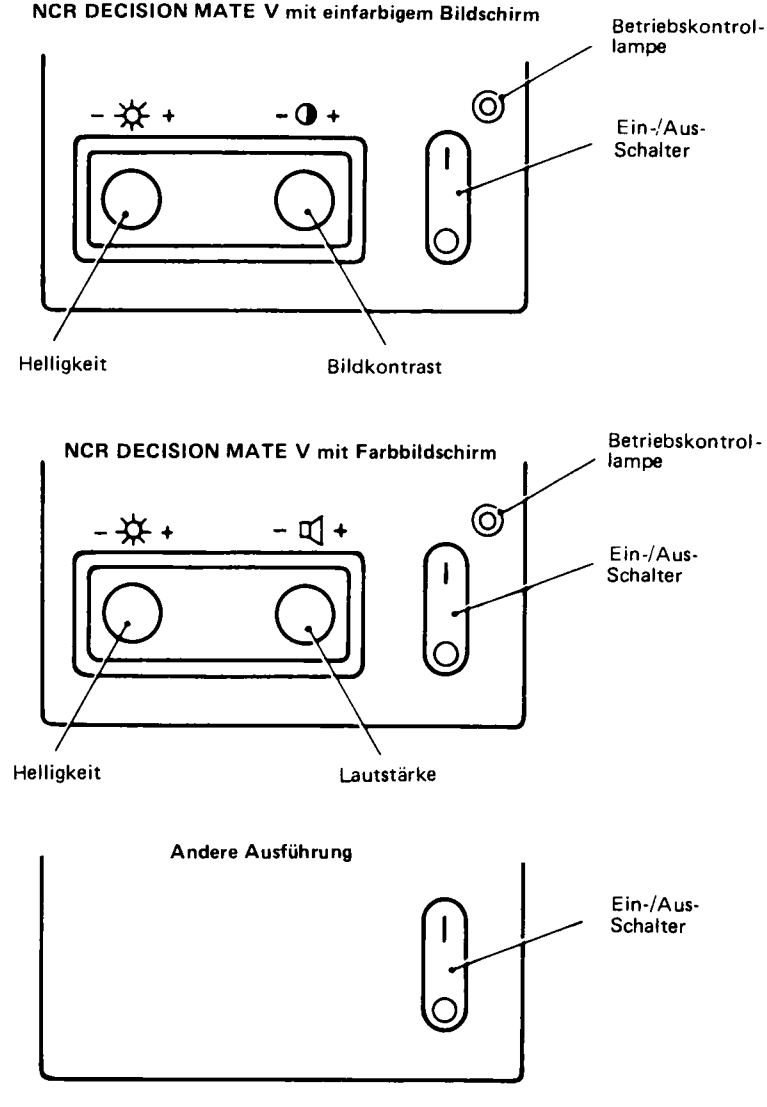


Abb. 3.5 Die Bedienungselemente

Die wesentlichen Bedienungselemente des Computers befinden sich unterhalb der Diskettenlaufwerke (siehe Abbildung 3.5). Sie umfassen je nach Auführung den Ein-/Aus-Schalter, die Betriebskontrollampe und die Einstell-Drehknöpfe für den Bildschirm.

NCR DECISION MATE MIT EINFARBIGEM BILDSCHIRM

Die Bedienungselemente umfassen einen Ein-/Aus-Schalter, eine Lampe, die bei eingeschaltetem Betriebszustand aufleuchtet, sowie zwei Drehknöpfe zur Steuerung des Bildschirms.

Ein-/Aus-Schalter

Drücken Sie den oberen Teil (1) des Schalters. Der Computer ist nun eingeschaltet, und die grüne Lampe leuchtet auf. Nachdem Sie Ihre Arbeiten mit dem Computer beendet haben, drücken Sie den unteren Teil (0) des Schalters, um den Computer auszuschalten.

Helligkeit und Kontrast

Stellen Sie mit den beiden Drehknöpfen den für Sie am angenehmsten erscheinenden Bildschirmzustand ein.

Lautstärke

Bei Systemen mit einfarbigem Bildschirm ist die Lautstärke des eingebauten Lautsprechers mit dem an der Rückseite angebrachten Regler einstellbar (siehe Abbildung 3.6).

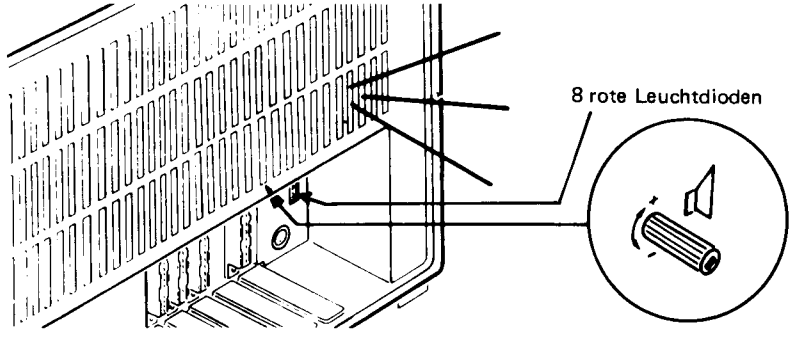


Abb 3.6 Regler für Lautstärke

NCR DECISION MATE V MIT FARBBILDSCHIRM

Bei einem System mit einem Farbbildschirm sind der Ein-/Aus-Schalter, der Helligkeits-Drehknopf und die grüne Kontrollampe wie bei einem System mit einfarbigem Bildschirm angebracht. Der

Drehknopf für Bildkontrast entfällt. An dieser Stelle befindet sich der Drehknopf für die Lautstärke des im NCR DECISION MATE V vorhandenen Lautsprechers. Dadurch entfällt der Regler für Lautstärke, die sich bei der vorher beschriebenen Ausführung an der Rückseite des Gehäuses befindet.

ANDERE AUSFÜHRUNG

Das einzige Bedienungselement ist der Ein-/Aus-Schalter. Die "fehlenden" Funktionen — Bildkontrast und Lautstärke — werden von den Funktionstasten besorgt, die sich in der oberen Reihe der Tastatur befinden. Die Lautstärke kann mit acht Stufen, der Bildkontrast mit fünf Stufen eingestellt werden. Beim Einschalten des Computers werden für diese Funktionen normalen Einsatztbedingungen entsprechende Werte automatisch eingestellt, die Sie aber nach Wunsch gemäß folgender Beschreibung ändern können.

- Wenn Sie die Funktionstaste F1 bei gedrückter CONTROL-taste betätigen, wird die Lautstärke vermindert.
- Wenn Sie die Funktionstaste F2 bei gedrückter CONTROL-Taste betätigen, wird die Lautstärke erhöht.
- Die Betätigung der Funktionstaste F3 bei gedrückter CONtrol-Taste setzt die Helligkeit des Bildschirms herab.
- Die Betätigung der Funktionstaste F4 bei gedrückter CONTROL-Taste setzt die Helligkeit des Bildschirms herauf.

WAS WIRD BEIM EINSCHALTEN GEPRÜFT?

Sobald Sie Ihren NCR DECISION MATE V einschalten, führt er eine Prüfung seiner internen Hardware durch. Das Ergebnis dieser Überprüfung wird Ihnen mittels der acht an der Rückseite des Gehäuses befindlichen roten Leuchtdioden mitgeteilt (siehe Abbildung 3.6): Im Falle, daß nicht alle Lampen nach einigen Sekunden erlöschen, liegt ein Defekt vor.

DER BILDSCHIRM

Der Bildschirm dient der visuellen Anzeige von Tastatur-Eingaben, Programmbefehlen, Systemmeldungen und anderen Informationen. Der Bildschirm ist für die alphanumerische Darstellung in 25 Zeilen zu je 80 Zeichen eingeteilt. Für graphische Darstellungen steht eine Auflösung von 640 horizontalen und 400 vertikalen Bildpunkten

zur Verfügung. Beide Darstellungsarten sind getrennt oder kombiniert anwendbar.

Die grünen Zeichen auf dunklem Hintergrund ergeben in Verbindung mit dem reflexionsfreien Bildschirm eine scharfe und flimmerfreie Anzeige. Mit dem Helligkeits- bzw. Kontrastregler läßt sich die Darstellung optimal an das jeweilige Bedienerblickfeld sowie die Lichtverhältnisse anpassen.

Systeme mit einem Farbbildschirm können folgende Farben darstellen: schwarz, weiß, rot, grün, blau, gelb, Magentarot und Zyan. Die Farbauswahl erfolgt durch das Programm. Bei diesen Systemen entfällt die Einstellung des Bildkontrastes.

DIE TASTATUR

Die Tastatur wird zur Eingabe von Befehlen und Daten benutzt. Das sogenannte "Roll-over" ermöglicht eine sehr schnelle Arbeitsweise durch überlappende Eingabe. Das heißt, Sie können eine Taste drücken, noch bevor die vorher gedrückte Taste in seine Ausgangsstellung zurückgegangen ist. Ein 8-Zeichen-Eingabespeicher übernimmt die Zwischenspeicherung, bis das Programm die jeweils anliegenden Daten abruft und verarbeitet. Wenn Sie eine Taste im gedrückten Zustand halten, wird das entsprechende Zeichen wiederholt erzeugt.

Akzeptiert der Computer Ihre Eingaben, so hören Sie einen kurzen Ton; fehlerhafte Eingaben quittiert er mit einem längeren Ton. In letzterem Fall ist eine erneute Eingabe notwendig.

Die Tastatur ist in zwei Felder aufgeteilt: einen alphanumerischen Teil, ähnlich dem einer Schreibmaschninentastatur, und ein rein numerisches Feld. Jedes Feld enthält eine Anzahl von Spezialtasten, deren programm- und anwenderabhängige Funktionen getrennt in der jeweiligen Softwaredokumentation aufgeführt sind. Die überwiegende Mehrzahl der Tasten erfüllen aber einheitliche, von der jeweils benutzten Software unabhängige Funktionen. Diese Tasten sind in den folgenden Abschnitten beschrieben.

DIE ALPHANUMERISCHE TASTATUR

Das alphanumerische Feld besteht aus einer standardisierten Schreibmaschinenanordnung und einigen Spezialtasten zur Steuerung des Computers. Die Funktion der Spezialtasten ist unabhängig von der Landessprache. Für die übrigen Tasten bestehen nur geringe Unterschiede zwischen den Ländervarianten. Die nachfolgenden Abbildungen beziehen sich auf die deutsche Ausführung.

Alphabetische Tasten

Bei Betätigung einer der in Abbildung 3.7 durch Umrandung hervorgehobenen Tasten erscheint das entsprechende Zeichen als Kleinbuchstabe auf dem Bildschirm und der Computer übernimmt den dazugehörigen Code. Sollen die Zeichen als Großbuchstaben eingegeben und angezeigt werden, so ist vorher die Umschalt-Feststelltaste zu betätigen.

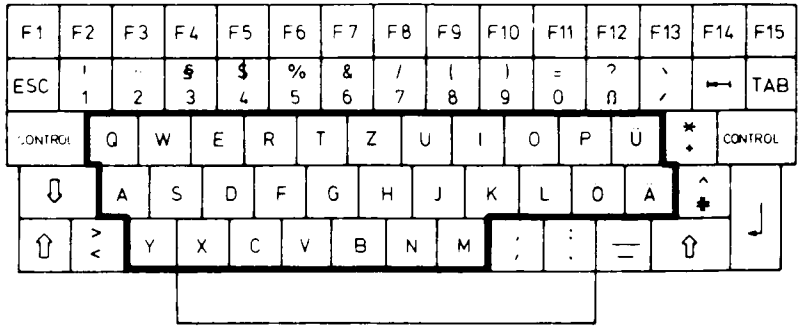


Abb 3.7 Alphabetische Tasten

Die Umschalt-Feststelltaste

Nach dem Drücken bleibt diese Taste (siehe Abbildung 3.8) fest in aktivierter Position (Umschalt-Feststeller); alle alphabestischen Zeichen der Abbildung 3.7 erscheinen als Großbuchstaben. Die Funktion wird durch nochmaliges Drücken aufgehoben.

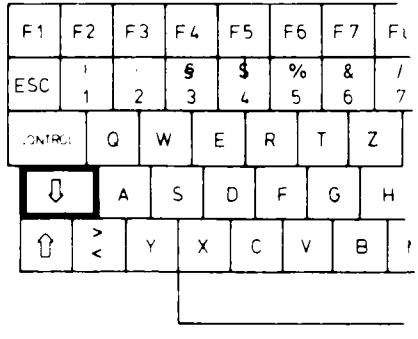


Abb. 3.8 Die Umschalt-Feststelltaste

Numerische Tasten

Bei Betätigung einer der in Abbildung 3.9 durch Umrandung hervorgehobenen Tasten erfolgt die Eingabe und Anzeige des jeweils in der unteren Hälfte der Taste gezeigten Zeichens. Wenn Sie die Taste zusammen mit einer der Umschalttasten (Abbildung 3.10) drücken, wird das auf der oberen Hälfte der Taste abgebildete Zeichen eingegeben und angezeigt.

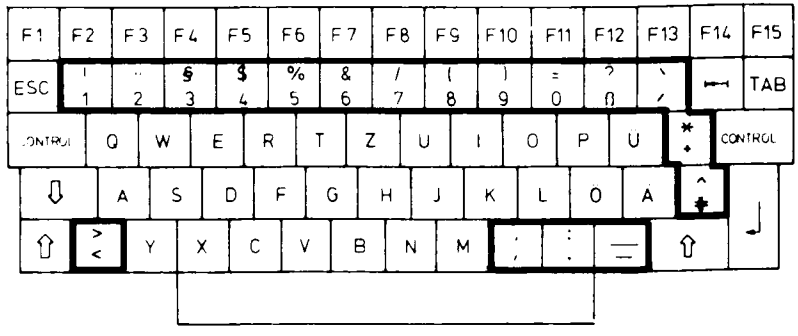


Abb. 3.9 Numerische und Symbol-Tasten

Die Umschalt-Tasten

Zur leichteren Bedienung enthält die Tastatur rechts und links eine Umschalt-Taste (Abbildung 3.10). Diese Tasten sind nicht feststellbar und gehen daher nach Wegnahme des Fingers wieder in ihre Ausgangsstellung zurück. Solange diese Taste in gedrückter Stellung ist, wird das in der oberen Hälfte einer Taste abgebildete Zeichen bzw. ein Großbuchstabe erzeugt. Wenn gleichzeitig die Umschalt-Feststelltaste sich in gedrückter Stellung befindet, wird ein Kleinbuchstabe statt eines Großbuchstabens erzeugt.

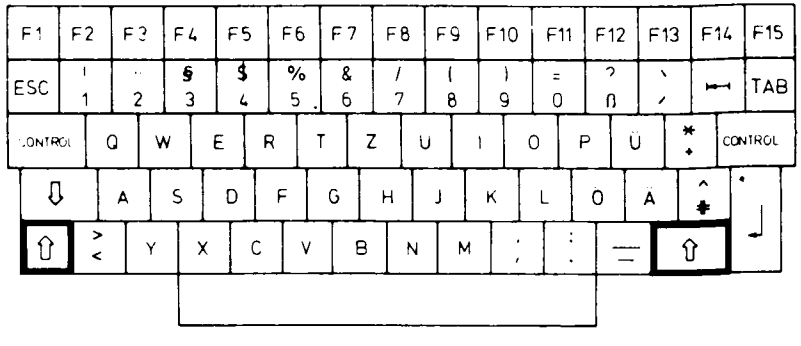


Abb. 3.10 Die Umschalt-Tasten

Die Rücktaste

Die Betätigung der Rücktaste (Abbildung 3.11) löscht das zuletzt eingegebene Zeichen und setzt die Schreibmarke (Cursor) am Bildschirm um eine Stelle zurück.

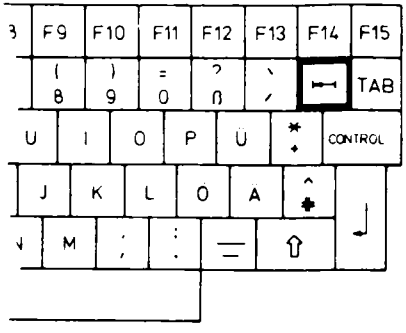


Abb. 3.11 Die Rücktaste

Die CONTROL-Tasten

Diese Tasten (Abbildung 3.12) verändern die Funktion einiger alphabetischer Tasten zur Steuerung des Computers. Die speziellen Funktionen sind von der Software abhängig und werden an anderer Stelle beschrieben. Die CONTROL-Funktion ist durch Betätigung einer der beiden CONTROL-Tasten zu aktivieren. Der Finger muß auf der Taste bleiben, während die gewünschte alphabetische Taste gedrückt wird. Nach Loslassen der CONTROL-Taste nimmt die alphabetische Taste ihre normale Funktion wieder an.

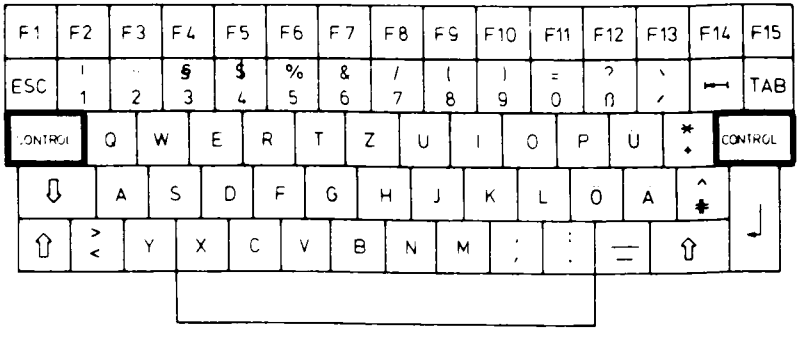


Abb. 3.12 Die CONTROL-Tasten

Die Abschlußtaste

Diese ist die meistgebrauchte Taste (Abbildung 3.13); sie ist deshalb sowohl in der alphanumerischen Tastatur wie auch in der rechts befindlichen numerischen Tastatur vorhanden. Sie meldet dem Computer, daß eine Eingabe abgeschlossen ist.

Anmerkung: Diese Taste wird oft als Carriage Return (CR) oder NEW LINE bezeichnet.

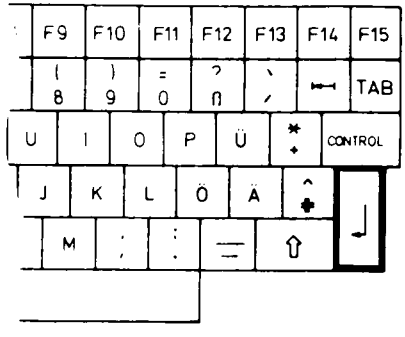


Abb. 3.13 Die Abschlußtaste

Programmierbare Tasten

Die Funktionsweise der in der alphanumerischen Tastatur verbleibenden Tasten (siehe Abbildung 3.14) — ESC, TAB, F1 bis F15 — ist von der benutzten Software (Betriebssystem, Anwendungsprogramm) abhängig. Oberhalb der Tasten ist eine Schiene vorhanden, die einen Papierstreifen aufnehmen kann. Ein entsprechender Papierstreifen wird mit der Betriebssystem-Software von NCR mitgeliefert. Er enthält den Wortlaut der etwa vom Be-

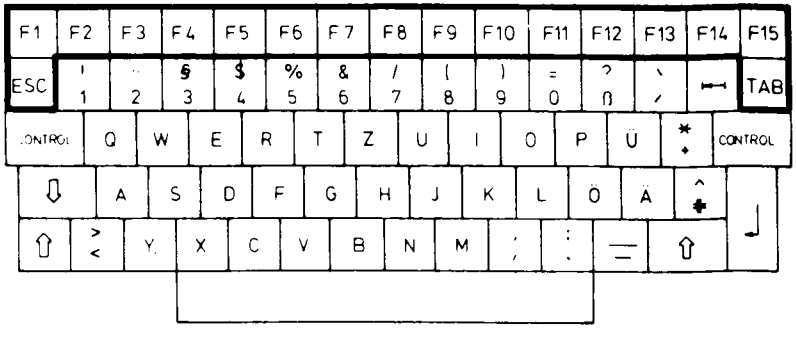


Abb. 3.14 Programmierbare Tasten

triebssystem bewirkten Vorprogrammierung der jeweiligen Taste. Sie können bei Neuprogrammierung der Funktionstasten diesen Streifen ergänzen oder ersetzen.

DIE NUMERISCHE TASTATUR

Das numerische 10-Tasten-Feld ermöglicht eine zügige Eingabe numerischer Daten. Zusätzlich enthält die numerische Tastatur folgende Tasten:

- Fünf programmierbare Tasten (F16 bis F20)
- Fünf Tasten für die Schreibmarke (Cursor)
- Tasten für die arithmetischen Funktionen Addition, Subtraktion, Multiplikation, Division und Löschen
- Null-Taste
- Doppelnull-Taste
- Dezimal-Komma
- Abschlußtaste

Programmierbare Tasten

Die fünf Tasten F16 bis F20 (Abbildung 3.15) sind genauso einsetzbar wie die Tasten F1 bis F15 der alphanumerischen Tastatur. Auch hier können Sie über der Tastenreihe eine Beschriftung anbringen.

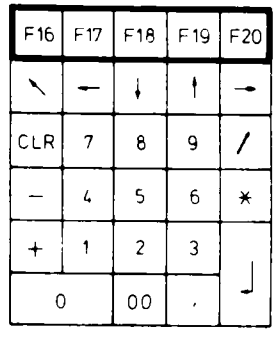


Abb. 3.15 Programmierbare Tasten

Positionszeiger-Tasten

Bei Betätigung einer dieser Tasten (Abbildung 3.16) wird der Positionszeiger auf dem Bildschirm entsprechend der Pfeilrichtung der Taste bewegt. Bitte beachten Sie hierzu auch die detaillierten Hinweise in der jeweiligen Software-Dokumentation.

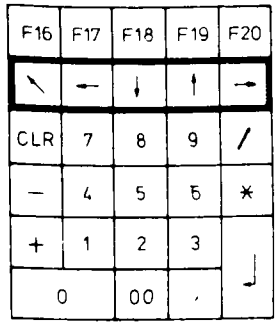


Abb. 3.16 Bildschirm-Positionszeiger

Arithmetische Funktionen

Diese Tasten (Abbildung 3.17) dienen den üblichen Grundrechnungsarten Addieren, Subtrahieren, Multiplizieren und Dividieren sowie der Löschfunktion. Die Anordnung der Zifferntasten entspricht dem Tastenfeld vieler Taschenrechner.

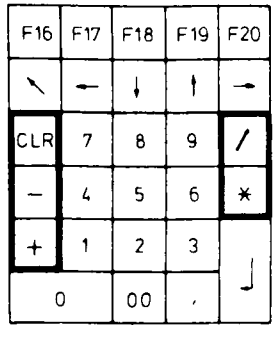


Abb. 3.17 Arithmetik-Tasten

Die Doppelnull-Taste

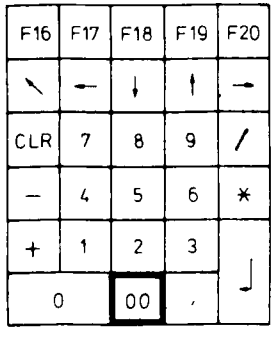


Abb. 3.18 Die Doppelnull-Taste

Zur schnelleren Eingabe ganzer Reihen von Nullen benutzen Sie die Doppelnull Taste (Abbildung 3.18). Gleichzeitiges Drücken von Null und Doppelnull ergibt 3 Nullen.

Das Dezimal-Komma

Diese Taste (Abbildung 3.19) verwenden Sie zum Setzen des Dezimal-Kommas in einer Ziffernfolge. Wenn Sie wollen, daß von Computer und Drucker statt des Kommas (,) ein Dezimalpunkt (.) angezeigt wird, müssen Sie Komma (,) bei gedrückter CONTROL-Taste betätigen. Nun wird bei Benutzung der Kommataste ein Dezimalpunkt angezeigt oder gedruckt. Die Rückstellung auf Normalbetrieb erfolgt durch erneutes Betätigen von CONTROL mit Komma oder durch Ausschalten des Systems.

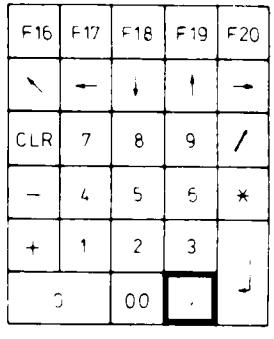


Abb. 3.19 Das Dezimal-Komma

Die Abschluß-Taste

Die Funktion dieser Taste (Abbildung 3.20) und die Funktion der Abschlußtaste in der alphanumerischen Tastatur sind identisch.

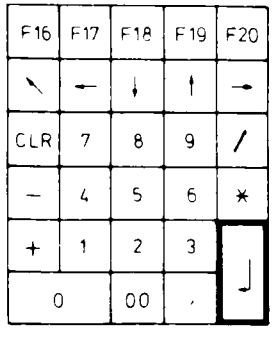


Abb. 3.20 Die Abschlußtaste

"KALTSTART"

Sie werden besonders bei der Entwicklung Ihrer eigenen Programme manchmal in einer Lage sein, in der das Aus- und ein erneutes Einschalten Ihres Computers zweckmäßig ist. In einem solchen Fall erübrigt sich das Ein- und Ausschalten mit dem dazugehörigen Neueinlegen Ihrer Diskette, wenn Sie die Funktionstaste F20 bei gedrückter CONTROL-Taste betätigen. Ihr NCR DECISION MATE V verhält sich dann, als ob Sie den Ein-/Aus-Schalter betätigt hätten.

WIE KANN ICH SOFTWARE LADEN?

Sie haben nun gelernt, wie Sie das System einschalten und eine Diskette einlegen können. Die wichtigsten Schritte sind nachstehend noch einmal zusammengefaßt:

Schalten Sie Ihren Computer ein (Ein-/Aus-Schalter auf "1") und warten Sie ein paar Sekunden. Der Computer meldet sich mit

DISK A: NOT READY <CR>

Schieben Sie eine Diskette, die ein Ihrem Computer entsprechendes Betriebssystem (8-Bit bzw. 16-Bit) enthält, mit der Schreib-/Lese-Öffnung voraus und dem Aufkleber in Richtung Bildschirm zeigend in das Laufwerk A ein. (Dasselbe Laufwerk wird im UCSD-p-System als "4" bezeichnet.) Danach drehen Sie den Verschlußhebel nach rechts.
Betätigen Sie die Abschlußtaste ↵.
Der Computer meldet sich mit einer Software-Mitteilung und dem Bereitschaftszeichen >. (Beim UCSD-p-System erscheint eine Befehlsauswahl in der ersten Zeile des Bildschirms).
Nach Beendigung Ihrer Arbeit mit dem Computer, sollten Sie den Ein-/Aus-Schalter erst nach Erscheinen des Bereitschaftszeichens betätigen.

EINIGE PRAKTISCHE ÜBUNGEN

Wenn Sie das MS-DOS-Betriebssystem oder eines der CP/M-Betriebssysteme verwenden, können Sie den folgenden Übungen nachgehen, damit Sie mit den Funktionen der Tastatur Ihres NCR DECISION MATE V vertraut werden. (Diese Übungen eignen sich

nicht bei Verwendung des UCSD-p-Systems, da dieses bei der Eingabe bestimmter Einzelbuchstaben mit der Verarbeitung sofort beginnt.) Beim MS-DOS-Betriebssystem müssen Sie zunächst die Betriebssystem-Diskette kopieren. Dieser Kopiervorgang wird automatisch eingeleitet (siehe "MS-DOS Benutzer-Handbuch").

Nachdem das System-Bereitschaftszeichen erschienen ist, geben Sie einfach irgendeinen belanglosen Text ein, z.B.

```
A> irgendwelche Informationen, die nicht von Bedeutung sind
```

Drücken Sie anschließend die Abschlußtaste. Durch diese Eingabe können Sie weder Ihrem Computer noch Ihrer Software einen Schaden zufügen — der Computer weiß nämlich zunächst gar nicht, worüber Sie reden. Er meldet sich daher durch Anzeige des ersten eingegebenen Wortes und eines Fragezeichens.

```
IRGENDWELCHE?
A>
```

Nun geben Sie unter Benutzung der in diesem Abschnitt beschriebenen Tasten weitere Daten ein. Schalten Sie auf Großbuchstaben um:

```
A> DATEN IN GROSSBUCHSTABEN EINGEBEN ↵
DATEN?
A> und dann auf kleinbuchstaben zurückschalten ↵
und?
A>
```

Falls Sie dabei Fehler machen, korrigieren Sie diese mit der Rücktaste; diese löscht ein einzelnes Zeichen oder auch alle Zeichen einer Zeile. So werden Sie schon bald mit der Tastatur vertraut sein. Bevor Sie mit der Arbeit am Computer beginnen, lesen Sie bitte sorgfältig die Einführung in dem zum Betriebssystem gehörenden Handbuch. Dort lernen Sie, wie Sie Disketten formatieren, Sicherungskopien von wichtigen Disketten anfertigen und das System für Ihre Anwendungen vorbereiten können. Lernen Sie, den Computer für Sie arbeiten zu lassen.

WAS NUN?

ERKENNEN VON STÖRUNGEN

Dieser Abschnitt erläutert Ihnen, wie Sie eventuell auftretende Störungen erkennen und beseitigen können. Schon mit ein paar Minuten Zeit, die Sie für eine selbständige Fehlersuche aufwenden, läßt sich oft das Herbeirufen des technischen Kundendienstes vermeiden.

Falls Sie doch einmal Hilfe in Anspruch nehmen müssen, so ist es von Vorteil, wenn Sie den Fehler genau beschreiben können. Der Kundendiensttechniker weiß dann, welche Prüfgeräte und Ersatzteile er mitbringen muß. Am besten, Sie notieren den aufgetretenen Fehler und die entsprechenden Fehleranzeigen in Stichworten nach folgendem Schema:

- Wann ist der Fehler aufgetreten?
- Bei welchem Betriebssystem und Anwenderprogramm?
- Genauer Wortlaut der Meldung am Bildschirm.
- Ob und ggf. welche Leuchtdioden an der Rückseite des Computers nicht ordnungsgemäß erlöschen.
- Leuchtanzeige des Diagnostik-Moduls (K220), falls vorhanden.

SERVICE-VEREINBARUNG

Benötigen Sie die Hilfe eines gut geschulten Kundendienstes, so kann Ihnen NCR ein erfahrenes Team von Technikern anbieten. Etwa 1200 über die ganze Erde verteilte Geschäftsstellen stehen zur Verfügung, von denen sich sicher auch eine in Ihrer Nähe befindet. Sie können dabei unter folgenden verschiedenen Wartungsaufträgen auswählen. Ihre nächstgelegene NCR Geschäftsstelle berät Sie gerne.

- NCR Voll-Service — bietet den Vorteil einer alles umfassenden Systembetreuung für eine günstige Jahresgebühr. Dieser Service durch speziell geschultes Personal steht Ihnen während der üblichen Geschäftszeiten zur Verfügung und schließt gewöhnlich die Anfahrts-, Reparatur- und Ersatzteilkosten mit ein.

- Zeit- und Material Service — hier zahlen Sie bei einem Kundendienstbesuch die anfallende Zeit sowie die Ersatzteile.
- Reparatur im Service-Center — Sie bringen Ihren Computer zur nächstgelegenen NCR-Geschäftsstelle. Nach durchgeführter Reparatur wird Ihnen das Gerät von NCR angeliefert, oder Sie erhalten Nachricht, daß es zur Abholung bereit ist.

Die Wartung des Computers kann auch auf andere Weise abgewickelt werden

- Haben Sie Ihren Computer nicht direkt von NCR, sondern über einen Vertriebsbeauftragten erworben, so können Sie auch direkt mit diesem Händler eine entsprechende Service-Vereinbarung treffen.
- Mit Hilfe des von NCR angebotenen Service-Handbuchs können technisch geschulte Personen in Ihrem Unternehmen den Computer selbst reparieren. Jedoch sollten solche Arbeiten nur von Technikern durchgeführt werden, die mit der Wartung von elektronischen Geräten sowie dem Umgang mit gedruckten Schaltungen und integrierten Schaltkreisen ausreichend ver; traut sind.

  Zur Unterstützung bietet Ihnen NCR Schulungskurse an. Näheres hierzu erfahren Sie bei Ihrer nächstgelegenen NCR-Geschäftsstelle.

BESEITIGUNG VON FEHLERN

In den folgenden Tabellen finden Sie nun einige typische Beispiele von Fehlermöglichkeiten, ihre Ursachen sowie Hinweise zu ihrer Behebung.

Fehler	Mögliche Ursachen	Abhilfe
Betriebs-Kontrollampe leuchtet nicht auf	Netzkabel nicht am Computer angeschlossen	Netzkabel am Computer anschließen
	Netzkabel nicht an Wandsteckdose angeschlossen	Netzstecker in Wandsteckdose stecken
	Keine Spannung an der Wandsteckdose	Benachrichtigen Sie den Hauselektriker
	Computer nicht eingeschaltet	Ein-/Aus-Schalter betätigen
	Fehlerhaftes Netzkabel	Netzkabel austauschen
	Interner Fehler im Computer	Benachrichtigen Sie den Service-Techniker
Betriebssystemmeldung erscheint beim Laden des Betriebssystems nicht am Bildschirm	Kontrast-/Helligkeitsregler nicht richtig eingestellt	Reglereinstellung korrigieren
	Diskette mit dem Betriebssystem falsch eingelegt	Diskette im Laufwerk A richtig einlegen Siehe Kapitel "Bedienung des Computers"
	Falsche Diskette	Richtige Diskette auswählen
	Systemdiskette in Laufwerk B (UCSD-p-System:5) bei Systemen mit zwei Disketten-Laufwerken. Richtig:Laufwerk A (UCSD-p-System:4)	Legen Sie die Systemdiskette in das richtige Laufwerk ein
	Systemdiskette beschädigt	Verwenden Sie Ihre Sicherungskopie, nachdem Sie eine weitere Sicherungskopie angefertigt haben
	Interner Fehler im Computer	Benachrichtigen Sie den Service-Techniker
Dateneingabe über Tastatur nicht möglich	Umschalt-Feststelltaste ist falsch eingestellt	Taste drücken
	Tastatur-Eingabespeicher voll	Warten Sie, bis die laufende Verarbeitung abgeschlossen ist
	Tastaturkabel nicht am Computer angeschlossen	Kabel anschließen
	Interner Fehler in Tastatur oder Computer	Benachrichtigen Sie den Service-Techniker

Fehler	Mögliche Ursache	Abhilfe
Datenübertragung auf Diskette wird nicht ausgeführt	Diskette nicht im richtigen Laufwerk	Diskette in das richtige Laufwerk einlegen
	Diskette falsch eingelegt	Diskette richtig einlegen (siehe Kapitel "Bedienung des Computers")
	Schreibschutzaufkleber nicht entfernt	Aufkleber entfernen, wenn der Schreibschutz aufgehoben werden soll
	Diskette beschädigt	Verwenden Sie Ihre Sicherungskopie
	Diskette nicht formatiert	Diskette formatieren
	Interner Fehler im Computer bzw. Diskettenlaufwerk	Benachrichtigen Sie den Service-Techniker
Diskette kann nicht gelesen werden	Diskette nicht im richtigen Laufwerk	Diskette in das richtige Laufwerk einlegen
	Diskette falsch eingelegt	Diskette richtig einlegen (siehe Kapitel "Bedienung des Computers")
	Diskette beschädigt	Verwenden Sie Ihre Sicherungskopie
	Interner Fehler im Computer oder Diskettenlaufwerk	Benachrichtigen Sie den Service-Techniker
Undefinierter Fehler — keine weitere Verarbeitung möglich	Probleme mit der Betriebssoftware und/oder dem Anwendungsprogramm	Betriebssystem und Anwendungsprogramm neu laden
		Computer abschalten. Verwenden Sie die Sicherungskopie des Betriebssystems und laden Sie diese sowie das Anwendungsprogramm erneut
	Vorübergehende Schwierigkeiten mit der Netzspannungsversorgung	Benachrichtigen Sie den Hauselektriker
	Interner Fehler im Computer	Benachrichtigen Sie den Service-Techniker

PRAKTISCHE HINWEISE

Mit "Praktische Hinweise" wollen wir Ihnen einige nützliche Empfehlungen für Ihre tägliche Arbeit mit dem Computer geben. Die Hinweise beziehen sich auf den Arbeitsplatz und dessen Umgebung sowie die Arbeitsweise mit dem Computer. So liegen z. B. die Ursachen für Schwierigkeiten mit Disketten meist in deren falscher Handhabung. Deshalb ist diesem Punkt besondere Aufmerksamkeit zu widmen. Auch wenn Sie einmal Ihren Computer an einen anderen Ort verlegen wollen, beachten Sie bitte die Hinweise am Ende dieses Kapitels.

AUFSTELLEN DES COMPUTERS

Bei der Auswahl des Aufstellungsortes für Ihren Computer sollten Sie folgende Hinweise berücksichtigen:

- Wählen Sie einen Platz abseits von zu starkem Bürobetrieb, an dem Sie ungestört arbeiten können.
- Vermeiden Sie Räume mit extremer Temperatur und Feuchtigkeit sowie starker Staubentwicklung. Sie sollten Ihren Computer von unmittelbarer Sonneneinstrahlung und starken Heizungen fernhalten.
- Installieren Sie genügend Steckdosen für den Computer und die etwa vorhandenen Peripheriegeräte.
- Sorgen Sie für ausreichenden Ablageplatz für Disketten und ihre Hüllen, Druckpapier usw.
- Legen Sie Netz- und Verbindungskabel so, daß sie nicht behindern.
- Wenn Sie Ihren Computer in der Nähe von Geräten betreiben, die starke Magnetfelder erzeugen, kann sich die Bildqualität verschlechtern. Um dies zu vermeiden, sollte der Computer von der Störquelle entfernt werden.

ARBEITEN MIT DEM COMPUTER

Die folgenden Hinweise helfen Ihnen,die tägliche Arbeit mit Ihrem NCR DECISION MATE V so effektiv wie möglich zu gestalten:

- Organisieren Sie die Arbeit in folgerichtigen Abläufen. Schaffen Sie genügend Platz zur Ablage der Arbeitsmittel, und trennen Sie erledigte von nichterledigter Arbeit.
- Schalten Sie den Computer nie aus, bevor das Bereitschaftszeichen des Systems am Bidlschirm erscheint, sonst kann es vorkommen, daß Daten gelöscht werden.
- Legen Sie Ihre Disketten sofort nach Gebrauch wieder in das Ablagefach zurück.
- Erstellen Sie in regelmäßigen Zeitabständen Sicherungskopien und bewahren diese getrennt auf.
- Am Ende der Tagesarbeit ziehen Sie den Netzstecker aus der Wandsteckdose.

BEHANDLUNG DER DISKETTEN

Sie können eine sehr hohe Lebensdauer von Ihren Disketten erwarten, vorausgestzt, daß Sie sorgfältig mit ihnen umgehen:

- Für die Beschriftung der Diskettenaufkleber verwenden Sie Filzstifte. Abriebteilchen von Bleistiften, Radiergummis, Fettstiften oder Rückstände von Kugelschreibern könnten auf die Magnetschicht der Diskette gelangen und diese beschädigen. Am zweckmäßigsten beschriften Sie einen Aufkleber, bevor Sie ihn auf die Diskette kleben. Falls dies nicht mehr möglich ist, sollten Sie die Diskette wieder in ihre Schutzhülle stecken, bevor Sie den Aufkleber beschrifften.
- Legen Sie die Diskette weder in grelles Sonnenlicht noch in die Nähe von magnetischen Gegenständen.
- Bringen Sie an der Hülle keine Büroklammern oder Gummiringe an.
- Berühren Sie nicht die Diskettenoberfläche. Versuchen Sie auch nicht, die Oberflächen zu reinigen. Dies ist nicht notwendig, weil sich die Diskette während des Laufes automatisch säubert.
- Nach Herausnahme aus dem Laufwerk stecken Sie die Diskette immer sofort in ihre Schutzhülle.

ORTSWECHSEL DES COMPUTERS

Ihr NCR DECISION MATE V ist so konzipiert, daß bei einem Standortwechsel keine außergewöhnlichen Vorkehrungen notwendig sind. Zur Vermeidung möglicher Beschädigungen sollten Sie jedoch folgende Empfehlungen beachten:

- Auch bei Transport über kurze Entfernungen sollten Sie Ihren Computer und die etwa angeschlossenen Zusatzgeräte vorher ausschalten und die Verbindungskabel lösen.
- Falls Sie den Computer öfters über größere Entfernungen transportieren oder gar mit der Bahn bzw. Post verschicken wollen, ist es zweckmäßig, die Originalverpackung aufzubewahren, um sie dann für den Transport zu benutzen. Diese Maßnahme gilt auch für die Einlegscheibe aus Karton, die sich bei der Lieferung in jedem Diskettenlaufwerk befand.

REINIGUNG DES COMPUTERS

Ihr NCR DECISION MATE V erfordert keine besonderen Reinigungsmaßnahmen. Gelegentliches Abstauben mit einem weichen Tuch ist aber zu empfehlen. Achten Sie besonders auf den Bildschirm und die Tastatur. Hartnäckige Flecken sollen nur mit einem weichen, mit milder Seifenlösung angefeuchteten Lappen entfernt werden.
Während der Reinigung muß der Computer unbedingt ausgeschaltet sein.

LEISTUNGSERWEITERUNGEN

Wenn Sie jetzt oder zu einem späteren Zeitpunkt den Leistungsumfang Ihres NCR DECISION MATE V durch die Anschaffung zusätzlicher Module erweitern, können Sie die mitgelieferten Beschreibungen in dieses Kapitel einfügen.

Allgemeine Lizenzbedingungen der NCR GmbH

1. Der Lizenznehmer verpflichtet sich, die Lizenzprogramme vor unbefugtem Gebrauch zu schützen. Das/Die Lizenzprogramm(e) dürfen, soweit dies technisch möglich ist, nur in maschinenlesbarer oder gedruckter Form kopiert werden, wenn die Kopie zur Modifizierung oder als Sicherheitskopie benötigt wird. Jede Kopie muß den Vermerk tragen, daß die NCR GmbH (nachfolgend kurz NCR genannt) das Urheberrecht am Lizenzprogramm besitzt.

2. Der Lizenznehmer darf das/die Lizenzprogramm(e) modifizieren oder zur Benutzung mit anderen Programmen verbinden, wenn deren Benutzung auf dem Computersystem NCR Decision Mate V erfolgt. Die modifizierten oder verbundenen Programme unterliegen diesen Allgemeinen Lizenzbedingungen der NCR. Die Modifikationen und Programmverbindungen müssen die Programm-Nummer des Ursprungsprogrammes und den Vermerk aufweisen, daß die NCR das Urheberrecht daran besitzt.

3. Die Benutzer-Lizenz sowie die sonstigen Rechte und Pflichten aus diesem Vertragsverhältnis dürfen vom Lizenznehmer nur mit vorheriger schriftlicher Zustimmung der NCR an Dritte übertragen werden.

4. Weist der Lizenznehmer innerhalb einer Frist von 6 Monaten, gerechnet nach dem Datum seiner Programmempfangsbestätigung nach, daß das (die) Lizenzprogramm(e) fehlerhaft ist (sind), hat die NCR wahlweise das Recht, ein anderes, gleiches Programm zu liefern, den Fehler nachzubessern oder dem Lizenznehmer zuzugestehen, daß dieser vom Vertrag über das fehlerhafte Programm zurücktritt. Im letzteren Fall werden dem Lizenznehmer die bereits bezahlten Lizenzgebühren gegen Rückgabe der (des) Lizenzprogramme(s) zurückerstattet. Soweit gesetzlich zulässig, sind damit alle weitergehenden Gewährleistungs- und Ersatzansprüche ausgeschlossen.

5. Die NCR übernimmt keine Garantie dafür, daß das (die) Lizenzprogramm(e) den Vorstellungen und Anforderungen des Lizenznehmers entsprechen.

6. Die Benutzer-Lizenz ist zeitlich beschränkt auf die Dauer der Benutzung des in Ziffer 2 genannten NCR-Computersystems. Das (Die) Lizenzprogramm(e) nebst Kopien, Modifikationen oder Programmverbindungen ist (sind) sofort der NCR zurückzugeben oder zu vernichten, sobald der Lizenznehmer dieses Computersystem ständig nicht mehr benutzt.
   Der Lizenznehmer ist verpflichtet, der NCR innerhalb von 30 Tagen nach endgültiger Einstellung der Benutzung schriftlich mitzuteilen, seit wann das NCR-Computersystem nicht mehr verwendet wird und daß er entweder das (die) Lizenzprogramme(e), Kopien, Modifikationen und/oder Programmverbindungen an die NCR zurückgeben wird oder vernichtet hat.

7. Verstößt der Lizenznehmer gegen die vorstehenden Bedingungen, hat die NCR das Recht, den Vertrag über die Benutzer-Lizenz ohne Einhaltung einer Frist zu kündigen und vom Lizenznehmer die sofortige Rückgabe der (des) Lizenzprogramme(s) nebst Kopien, Modifikationen oder Programmverbindungen zu verlangen, ohne zur Erstattung bereits bezahlter Lizenzgebühren verpflichtet zu sein.

8. Für evtl. Streitigkeiten aus diesem Vertragsverhältnis ist Augsburg Gerichtsstand, es sei denn, daß ein ausschließlicher Gerichtsstand besteht.

BENUTZUNG DIESES HANDBUCHS

GW-BASIC stellt eine Erweiterung der Leistungen dar, die von MS-BASIC im MS-DOS-Betriebssystem angeboten sind. Mit den leicht zu lernenden GW-BASIC-Anweisungen können Sie Grafik-, Musik- und Datenübertragungsprogramme erstellen.

Dieses Handbuch, das Sie mit Ihrer GW-BASIC-Diskette erhalten, besteht aus drei Hauptteilen: MS-BASIC, der MS-DOS-Erweiterung sowie der Beschreibung der GW-BASIC-Erweiterung. Sie haben damit eine vollständige Beschreibung der Programmiersprache BASIC in einem Band.

Wenn Sie mit MS-BASIC im MS-DOS-Betriebssystem vertraut sind, werden Sie sich hauptsächlich für die Beschreibung der GW-BASIC-Erweiterung interessieren. Auch wenn Sie MS-BASIC oder das MS-DOS-Betriebssystem noch nicht kennengelernt haben, stellt dieser Band eine unentbehrliche Hilfe dar. Sie sollten dann die zwei ersten Teile lesen und die in ihnen beschriebenen MS-BASIC-Anweisungen und -Funktionen mit Ihrem NCR DECISION MATE V üben. Dadurch gewinnen Sie einen Einblick in die Aspekte der BASIC-Programmierung, die auch der GW-BASIC-Erweiterung zugrundeliegen.

NCR

NCR DECISION MATE V

GW™-BASIC

MS, GW, Music Language und Graphics Macro Language sind Warenzeichen der Microsoft Corporation.

1. Auflage, Dezember 1983

Diese Dokumentation repräsentiert den neuesten Stand zum Zeitpunkt der Veröffentlichung. NCR behält sich das Recht vor, im Zuge der technischen Weiterentwicklung den Inhalt jederzeit abzuändern und dem jeweiligen Stand anzupassen.

Nicht alle hier beschriebenen Leistungen werden von NCR in allen Teilen der Welt vertrieben. Nähere Informationen bezüglich eventueller Einschränkungen oder auch Erweiterungen sowie den aktuellen Stand erfahren Sie von Ihrem Händler oder der nächstgelegenen NCR-Geschäftsstelle.

WICHTIGER HINWEIS
FÜR DIE BENUTZUNG VON MS-DOS MIT 256KB-SPEICHER

Wenn Ihr System einen 256 KB-Speicher beinhaltet, sollten Sie bei Verwendung des MS-LINK-Dienstprogramms die /HIGH-Option nicht benutzen. (Diese Option kann ohnehin mit PASCAL- und FORTRAN-Programmen unabhängig von der verfügbaren Speichergröße nicht benutzt werden.) Der MS-LINK-Vorgang wird wie üblich durchgeführt. Sobald MS-DOS aber versucht, das maschinenausführbare .EXE-Programm in den Speicher zu laden, ist ein Neubetätigen des Netzschalters am Computer erforderlich.

Es ist möglich, ein Unterprogramm, das vom BASIC-Interpreter übersetzt werden soll, in den oberen Bereich des Speichers zu laden. Dies ist aber nur dann zu empfehlen, wenn Sie bereits mit der Funktionsweise der Programm-Testhilfe (debugger) vertraut sind:

1. Vergewissern Sie sich, daß Ihr Unterprogramm keine Fehler enthält.
2. Übertragen Sie das Unterprogramm in den oberen Speicherbereich.
3. Notieren Sie die neue Segmentadresse entsprechend der neuen Anfangsadresse des Unterprogramms.
4. Laden Sie den BASIC-Interpreter.
5. Laden Sie das BASIC-Programm.
6. Ändern Sie die DEF SEG-Instruktion, damit sie auf die neue Anfangsadresse zeigt.
7. Übertragen Sie das Unterprogramm auf Platte (BSAVE).

Die nachstehende Erläuterung sollten Sie lesen, wenn Sie Programme einsetzen wollen, die die Größe des Benutzerbereichs berechnen. Die folgenden Anweisungen sorgen dafür, daß das Programm einwandfrei geladen wird.

1. Laden Sie das Anwender-Programm.
2. Kopieren Sie den Inhalt von CS (Segmentadresse des Programms) in eine Speicherstelle (2 Byte), wo er nicht versehentlich gelöscht werden kann.
3. Rufen Sie die System-Funktion "Modify allocated memory block" auf, indem Sie folgende Anweisungen ausführen:
   - Übertragen Sie den Inhalt von CS in ES
   - Laden Sie BX mit der Länge des Programms

- Laden Sie das AH-Register mit 4AH
- INT 21H

4. Rufen Sie die System-Funktion "Allocate memory" auf, indem Sie folgende Anweisungen ausführen:
   - Laden Sie BX mit FFFFH
   - Laden Sie das AH-Register mit 48H
   - INT 21H

5. Multiplizieren Sie den Inhalt von BX mit 16, um die Größe des Benutzerbereichs zu berechnen.

(Das Kapitel "System Calls" im MS-DOS "Programmer's Manual" enthält eine ausführliche Beschreibung der hier benutzten System-Funktionen.

IMPORTANT INFORMATION
FOR MS-DOS USERS WITH 256 KB MEMORY

Do not use the /HIGH switch when linking a program with MS-LINK on systems with 256 KB memory. (The switch can never be used with PASCAL and FORTRAN programs, regardless of memory size.) While MS-LINK performs successfully, the computer "goes down" when MS-DOS attempts to load the .EXE program into memory.

If using interpretive BASIC, you can store a subroutine in high memory, but you should only use this technique if you are familiar with debugger:

1. Debug subroutine
2. Move subroutine to higher memory
3. Record register values (segment address of subroutine)
4. Load BASIC
5. Load BASIC program
6. Change DEF SEG to refer to subroutine
7. Save subroutine (BSAVE)

Finally, use the following sequence in programs that compute the size of the user area. This sequence ensures a successful program load.

1. Load the application program.
2. Save segment address of program (in CS).
3. Issue the System Call, "Modify allocated memory block":
   - move CS to ES
   - move length of program to BX
   - move 4AH to AH
   - INT 21H

4. Issue the System Call "Allocate memory":
   - move FFFFH to BX
   - move 48H to AH
   - INT 21H

5. Multiply the contents of BX by 16 to determine the size of the user area.

(For detailed information see the MS-DOS Programmer's Manual, "System Calls" chapter.)

KIT INSTALLATION

for

MEMORY EXPANSION
(K 202)

The attached pages provide information for installing this kit into an NCR DECISION MATE V. These pages should be filed in your NCR DECISION MATE V User Information Manual.

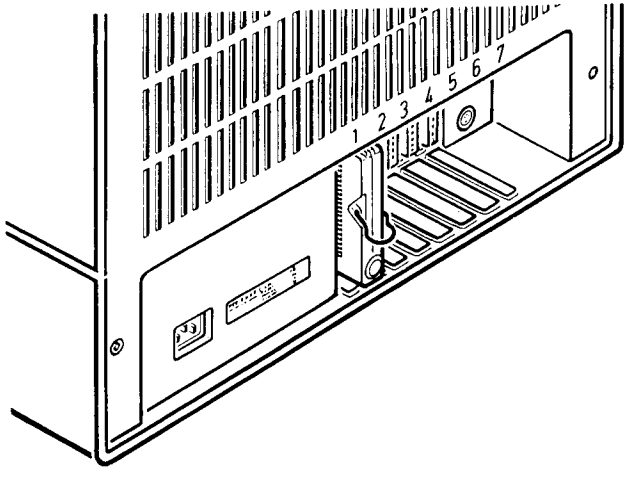
KIT INSTALLATION

MEMORY EXPANSION
(K 202)

1. Install this kit into slot 1 at the rear of the NCR DECISION MATE V.

   NOTE: No other slot position may be used for this kit.

You have now increased the memory capacity of your system to 256 Kilobytes.



NCR DECISION MATE V

SPEICHERERWEITERUNG (K202)

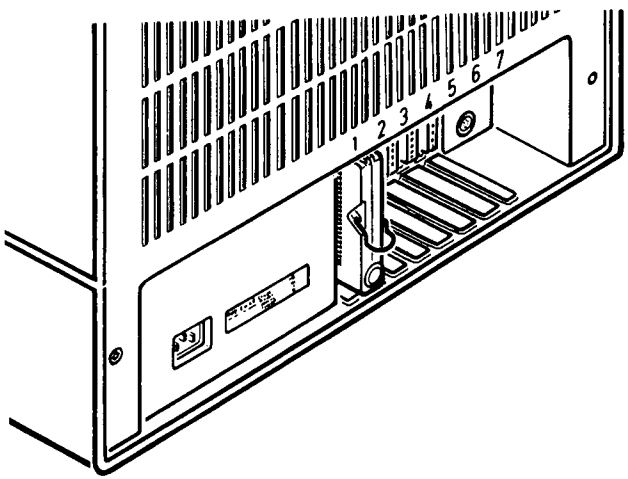
Die beiliegende Seite erklärt Ihnen, wie Sie diese Leistungserweiterung an Ihren NCR DECISION MATE V anschließen können. Bitte ordnen Sie diese Beschreibung in Ihre Bedienungsanleitung für den NCR DECISION MATE V ein.

NCR ist ständig bemüht, die Produkte im Zuge der Entwicklung von Technologie, Bauteilen, Soft- und Firmware dem neuesten Stand anzupassen. NCR behält sich deshalb das Recht vor, Spezifikationen ohne vorherige Ankündigung zu ändern.
Nicht alle hier beschriebenen Leistungen werden von NCR in allen Teilen der Welt vertrieben. Nähere Informationen bezüglich eventueller Einschränkungen oder Erweiterungen sowie den aktuellen Stand erfahren Sie von Ihrem Händler oder der nächstgelegenen NCR-Geschäftsstelle.

SPEICHERERWEITERUNG (K202)

Setzen Sie dieses Erweiterungsmodul in die dafür vorgesehene Steckfassung 1 an der Rückseite Ihres NCR DECISION MATE V ein.

Beachten Sie bitte, daß diese Erweiterung in keine andere Steckfassung gesteckt werden darf.



Sie haben damit die Speicherkapazität Ihres NCR DECISION MATE V auf 256 Kilobyte erweitert.

NCR

NCR DECISION MATE V

MS-BASIC®

MS-BASIC

INHALT

1. ALLGEMEINE INFORMATIONEN
Initialisierung 1-1
Operations Modi 1-1
Zeilenformat 1-1
Zeilennummern 1-2
Zeichensatz 1-2
Steuerzeichen 1-3
Konstanten 1-3
Einfache und doppelte Genauigkeit bei numerischen Konstanten 1-4
Variable 1-5
Namen für Variable und Kennzeichnungssymbole 1-5
Gruppen-Variable 1-6
Umwandlung von numerischen Daten 1-6
Ausdrücke und Operatoren 1-8
Arithmetische Operatoren 1-8
Ganzzahl-Division und Modulus-Arithmetic 1-9
Überlauf und Division durch Null 1-9
Vergleichende Operatoren 1-10
Logische Operatoren 1-10
Funktionale Operatoren 1-13
Text Operationen 1-13
Korrigierende Eingabe 1-14
Fehlermeldungen 1-14

2. MS-BASIC, BEFEHLE UND ANWEISUNGEN
REGELN DES INSTRUKTIONSFORMATES 2-1
AUTO 2-3
CALL 2-4
CHAIN 2-5
CLEAR 2-7
CLOAD 2-8
CLOSE 2-9
COMMON 2-10
CONT 2-12
CSAVE 2-13

DATA 2-14
DEF FN 2-15
DEFINT/SNG/DBL/STR 2-17
DEF USR 2-18
DELETE 2-19
DIM 2-20
EDIT 2-21
END 2-25
ERASE 2-26
DIE VARIABLEN ERR und ERL 2-27
ERROR 2-28
FIELD 2-30
FOR ... NEXT 2-33
GET 2-36
GOSUB ... RETURN 2-37
GOTO 2-38
IF ... THEN [...ELSE] und IF ... GOTO 2-39
INPUT 2-41
INPUT # 2-43
KILL 2-44
LET 2-45
LINE INPUT 2-46
LINE INPUT# 2-47
LIST 2-48
LLIST 2-50
LOAD 2-51
LPRINT und LPRINT USING 2-52
LSET und RSET 2-53
MERGE 2-54
MID$. 2-55
NAME 2-56
NEW 2-57
NULL 2-58
ON ERROR GOTO 2-59
ON ... GOSUB und ON ... GOTO 2-60
OPEN 2-61
OPTION BASE 2-62
OUT 2-63
POKE 2-64
PRINT 2-65
PRINT USING 2-68
PRINT# UND PRINT# USING 2-73
PUT 2-75
RANDOMIZE 2-76

READ 2-77
REM 2-79
RENUM 2-80
RESTORE 2-82
RESUME 2-83
RUN 2-84
SAVE 2-85
STOP 2-86
SWAP 2-87
TRON/TROFF 2-88
WAIT 2-89
WHILE ... WEND 2-90
WIDTH 2-91
WRITE 2-92
WRITE# 2-93

3. MS-BASIC FUNKTIONEN

ABS 3-2
ASC 3-2
ATN 3-3
CDBL 3-3
CHR$ 3-4
CINT 3-4
COS 3-5
CSNG 3-5
CVI, CVS, CVD 3-6
EOF 3-7
EXP 3-7
FIX 3-8
FRE 3-8
HEX$ 3-9
INKEY$ 3-10
INP 3-10
INPUT$ 3-11
INSTR 3-12
INT 3-12
LEFT$ 3-13
LEN 3-13
LOC 3-14
LOG 3-14
LPOS 3-14
MID$ 3-15
MKI$, MKS$, MKD$ 3-16
OCT$ 3-16

PEEK 3-17
POS 3-17
RIGHT$ 3-18
RND 3-18
SGN 3-19
SIN 3-19
SPACE$ 3-20
SPC 3-20
SQR 3-21
STR$ 3-21
STRING$ 3-22
TAB 3-22
TAN 3-23
USR 3-23
VAL 3-24
VARPTR 3-25

A ÜBERSICHT ÜBER FEHLERCODES UND FEHLERMELDUNGEN A-1

B MATHEMATISCHE FUNKTIONEN B-1

C ASCII ZEICHEN-CODES C-1

D RESERVIERTE WORTE FÜR MS-BASIC D-1

KAPITEL 1

ALLGEMEINE INFORMATIONEN ÜBER MS-BASIC

INITIALISIERUNG

Die Initialisierung variiert für die verschiedenen Implementationen von MS-BASIC. Bitte lesen Sie aufmerksam dieses Handbuch, um die richtige Vorgehensweise für Ihr Betriebssystem zu wählen.

OPERATIONS MODI

Nach dem Initialisieren von MS-BASIC meldet es sich mit „OK". Dies bedeutet, daß sich MS-BASIC im Befehlsmodus befindet und bereit ist, Befehle entgegenzunehmen. Zu diesem Zeitpunkt kann MS-BASIC in zwei Weisen benutzt werden. im Direktmodus oder im Indirektmodus.

Im Direktmodus werden vor BASIC-Befehlen und -Anweisungen keine Zeilennummern verwendet. Sie werden direkt nach der Eingabe ausgeführt. Ergebnisse von arithmetischen und logischen Operationen können sofort angezeigt und abgespeichert werden. Die Instruktionen selbst können im Direktmodus nicht gespeichert werden. Der Direktmodus ist sinnvoll bei der Fehlersuche und wenn BASIC als „Rechner" für kleine Berechnungen benutzt wird, die kein komplettes Programm benötigen.

Der Indirektmodus wird zur Eingabe von Programmen verwendet. Programmzeilen sind mit Zeilennummern versehen und werden abgespeichert. Das Programm, das sich im Speicher befindet, wird durch Eingabe des Befehles RUN ausgeführt.

ZEILENFORMAT

Programmzeilen eines BASIC-Programmes haben folgendes Format (eckige Klammern zeigen an, daß der darin befindliche Teil wahlweise verwendet werden kann):

nnnnn BASIC-Anweisung [: BASIC-Anweisung...] <Eingabetaste>

Eine Zeile kann mehrere BASIC-Anweisungen enthalten. Jede Anweisung muß von der nächsten durch einen Doppelpunkt getrennt werden.

Eine BASIC Programmzeile beginnt immer mit der Zeilennummer, endet mit der Eingabetaste und kann maximal 255 Zeichen enthalten.

Es kann eine logische Zeile über eine physische Zeile des Bildschirmes hinaus ausgedehnt werden. Dazu wird die Taste <Zeilenschaltung> benutzt, die es ermöglicht, die Eingabe der logischen Zeile auf der nächsten physischen Zeile fortzusetzen, ohne die <Eingabetaste> zu verwenden.

Zeilennummern

Jede BASIC-Programmzeile beginnt mit einer Zeilennummern. Die Zeilennummern geben die Reihenfolge an, in der die Zeilen im Speicher abgestellt sind. Außerdem dienen sie als Bezugsadressen für Programmsprünge und beim Editieren. Zeilennummern müssen im Bereich von 0 bis 65529 liegen. Es kann ein Punkt (.) benutzt werden, um bei den Befehlen EDIT, LIST, AUTO und DELETE auf die momentane Zeile zu verweisen.

ZEICHENSATZ

Der Zeichensatz von MS-BASIC setzt sich aus Buchstaben, Ziffern und Sonderzeichen zusammen.

Der Buchstabensatz umfaßt Groß- und Kleinbuchstaben.

Der Ziffernsatz besteht aus den Ziffern 0 bis 9.

Nachfolgend sind die von MS-BASIC erkannten Sonderzeichen aufgeführt:

Zeichen	Bedeutung
	Leerschritt
=	Gleichheitszeichen oder Zuordnungssymbol
+	Plus
–	Minus
*	Stern oder Multiplikationssymbol
/	Schrägstrich oder Divisionssymbol
^	Pfeil nach oben bzw. Exponentierungssymbol
(	linke Klammer
)	rechte Klammer
%	Prozent
#	Nummernzeichen (oder Pfund Sterling)
$	Dollarzeichen
!	Ausrufezeichen
[	linke eckige Klammer
]	rechte eckige Klammer

Zeichen	Bedeutung
,	Komma
.	Punkt (Dezimalpunkt)
'	Apostroph
;	Strichpunkt (Semikolon)
:	Doppelpunkt (Kolon)
&	kaufmännisches „und" (Ampersand)
?	Fragezeichen
<	kleiner als
>	größer als
\	Schrägstrich links oder Divisionssymbol für Integer-Division
@	„at"-Zeichen
_	Unterstreichung
<rubout>	löscht das zuletzt eingegebene Zeichen (auch Rücktaste)
<escape>	dient dazu, um Unterbefehle im EDIT-Modus abzubrechen. Siehe Seite 2-19
<tab>	Bewegt den Zeiger am Bildschirm auf die nächste Tabulator-Position (Tabulator-Positionen sind nach je 8 Stellenpositionen gesetzt).
<line feed>	Zeilenschaltung zur nächsten physischen Zeile
<carriage return>	Eingabetaste. Zur Beendigung einer Eingabe.

Steuerzeichen

MS-BASIC kennt die folgenden Steuerzeichen:

Control-A	Beginnt den EDIT-Modus für die momentane Zeile.
Control-C	Unterbricht einen Programmablauf und bringt MS-BASIC in den Direktmodus.
Control-G	Bringt ein akustisches Signal zum Ertönen.
Control-H	Rücktaste. Löscht das zuletzt eingegebene Zeichen.
Control-I	Tabulation. Bewegt den Zeiger am Bildschirm auf die nächste Tabulator-Position.
Control-O	Unterbricht die Ausgabe in einem Programm, alle anderen Funktionen werden fortgeführt. Eine weitere Betätigung von Control-O aktiviert die Ausgabe wieder.
Control-R	Wiederholt die momentane Zeile.
Control-S	Unterbricht den Ablauf eines Programmes.
Control-Q	Führt den Programmablauf fort (nach Control-S).
Control-U	Löscht die momentane Zeile.

KONSTANTEN

Konstanten sind vorgegebene Werte, die BASIC während des Programmablaufes benutzt. Es gibt zwei Arten: alphanumerische Konstanten und numerische Konstanten.

Eine alphanumerische Konstante (wir nennen sie von nun an String-Konstante (ist eine Folge von bis zu 255 alphanumerischen Zeichen, die in Anführungsstrichen (") eingebettet sind. Beispiele:

"HALLO"
"$25,000.00"
"Anzahl der Mitarbeiter"

Numerische Kosntanten sind positive oder negative Zahlen. Numerische Konstanten in BASIC dürfen keine Kommas enthalten.

Es gibt fünf Typen von numerischen Konstanten

Integre Konstanten	Ganzzahlige Zahlen zwischen −32768 und +32767. Sie haben keinen Dezimalpunkt.
Festkomma Konstanten	Positive oder negative Realzahlen. D.h. Zahlen mit Dezimalpunkt.
Gleitkomma-Konstanten	Positive oder negative Zahlen in Exponentialdarstellung (ähnlich der wissenschaftlichen Darstellung). Eine Gleitkomma-Konstante besteht aus einer ganzzahligen oder realen Zahl, der Mantisse, wahlweise mit Vorzeichen versehen, gefolgt vom Buchstaben E und einer ganzen Zahl, dem Exponenten, der ebenfalls wahlweise Vorzeichen haben kann. Der zulässige Bereich für Gleitkomma-Konstanten ist 10 hoch −38 bis 10 hoch +38. Beispiele: 234.988E−7 = .0000235988 2359E6 = 2359000000 (Gleitkomma-Konstanten mit doppelter Genauigkeit benutzen den Buchstaben D statt E. Siehe nächsten Abschnitt).
Hexadezimale Konstanten	Hexadezimale Zahlen mit dem Vorspann &H. Beispiele: &H76 &H32F
Oktale Konstanten	Oktalzahlen mit dem Vorspann &O oder &. Beispiele: &0347 &1234

Bemerkung: Die 8K-Version von MS-BASIC unterstützt keine hexadezimalen oder oktalen Konstanten.

Einfache und doppelte Genauigkeit bei numerischen Konstanten

Numerische Konstanten können einfache oder doppelte Genauigkeit aufweisen. Solche mit einfacher Genauigkeit werden 7-stellig gespeichert und mit bis zu 6 Stellen ausgegeben.

Bei doppelter Genauigkeit werden die Zahlen 16-stellig gespeichert und können mit bis zu 16 Stellen ausgegeben werden.

Einfache Genauigkeit hat jede numerische Konstante, die:

- aus sieben oder weniger Ziffern besteht, oder
- in der Gleitkommadarstellung den Buchstaben E benutzt, oder
- als Nachspann ein Ausrufezeichen (!) hat.

Doppelte Genauigkeit hat jede numerische Konstante, die

- aus acht oder mehr Ziffern besteht, oder
- in der Gleitkommadarstellung den Buchstaben D benutzt, oder
- als Nachspann das Nummernzeichen (#) hat.

Beispiele von Konstanten

Einfache Genauigkeit	Doppelte Genauigkeit
46.8	345692811
−1.09E-06	−1.09432D-06
3489.0	3489.0#
22.5!	1654321.1234

VARIABLE

Als Variable bezeichnet man Namen, die in einem BASIC-Programm Werte repräsentieren. Der Wert einer Variablen kann entweder ausdrücklich vom Programmierer bestimmt werden, oder das Ergebnis von Berechnungen in einem Programm sein. Bevor eine Variable einen Wert zugeordnet bekommt, wird ihr Wert mit Null angenommen.

Namen für Variable und Kennzeichnungsymbole

Namen für Variable in MS-BASIC können beliebig lang sein. Es werden bis zu 40 Zeichen eines Variablennamens benutzt. Variablennamen können aus Buchstaben und Ziffern bestehen. Das erste Zeichen muß ein Buchstabe sein. Spezielle Kennzeichnungssymbole sind ebenfalls erlaubt. - - Siehe unten.

Die reservierten Wörter dürfen nicht als Variablennamen verwendet werden. Variablennamen, die mit den Buchstaben FN beginnen, werden als vom Benutzer definierte Funktionen behandelt. Die reservierten Wörter umfassen alle Befehle, Anweisungen, Funktions- und Operatorennamen von MS-BASIC.

Variable können entweder numerische Werte oder alphanumerische Texte enthalten. Namen für Textvariablen werden am Ende mit einem Dollar-Zeichen ($) gekennzeichnet, zum Beispiel: A$ = "Umsatzbericht". Das Dollar-Zeichen ist ein Kennzeichnungssymbol für einen Typ von Variablen. Dies bedeutet, es kennzeichnet die Variable als Text-Variable.

Es können numerische Variablennamen mit Kennzeichnungssymbolen für ganzzahlige Werte sowie einfache und doppelte Genauigkeit versehen werden. Die Kennzeichnungssymbole sind wie folgt:

%	Ganzzahlige Variable (Integer)
!	Variable mit einfacher Genauigkeit
#	Variable mit doppelter Genauigkeit.

Falls keine Kennzeichnung vorgenommen wurde, nimmt MS-BASIC Variable mit einfacher Genauigkeit an.

Beispiele für Variablennamen folgen:
In den Versionen "Erweitert" und "Platte":

PI#	bezeichnet eine Variable mit doppelter Genauigkeit
MINIMUM!	bezeichnet eine Variable mit einfacher Genauigkeit
LIMIT%	bezeichnet eine ganzzahlige Variable.

Es gibt eine weitere Methode, um den Typ einer Variablen festzulegen. Die MS-BASIC Anweisungen DEFINT, DEFSTR, DEFSNG und DEFDBL können innerhalb eines Programmes den Typ für bestimmte Variablen festlegen. Diese Anweisungen sind in Kapitel 2 näher beschrieben.

Gruppen-Variable

Hierbei handelt es sich um eine Gruppe oder Tabelle von Variablen, die mit demselben Namen bezeichnet werden. Dieser Name wird mit einem ganzzahligen Wert oder einem ganzzahligen Ausdruck unterdefiniert. Eine Gruppen-Variable hat ebensoviele Unterdefinitionen (Indizes) wie Dimensionen. V(10) bezeichnet z.B. einen Wert einer eindimensionalen Gruppen-Variablen, T(1,4) einen Wert in einer zweidimensionalen, usw. Die maximale Anzahl von Dimensionen für eine Gruppen-Variable ist 255, die maximale Anzahl von Elementen (= Werten) pro Dimension ist 32767.

UMWANDLUNG VON NUMERISCHEN DATEN

Falls notwendig, kann BASIC numerische Konstante von einem Typ in einen anderen umwandeln. Dabei sollten Sie folgende Regeln kennen:

- Wenn eine numerische Konstante einer numerischen Variablen anderen Types gleichgesetzt wird, wird der Wert so gesetzt, daß er dem Typ der Variablen entspricht. Werden dabei alphanumerische mit numerischen Daten gemischt, erscheint die Fehlermeldung "TYPE MISMATCH" (Typenvermischung). Beispiel:

```
10 A% = 23.42
20 PRINT A%
RUN
23
```

- Während der Auswertung eines mathematischen oder vergleichenden Ausdruckes werden alle Operanden dem mit der höchsten Genauigkeit angepaßt. Auch das Ergebnis einer arithmetischen Operation entspricht dieser Genauigkeit.
  Beispiele:

```
10 D# = 6#/7
20 PRINT D#
RUN
 .8571428571428571
```

Die arithmetische Operation wird mit doppelter Genauigkeit ausgeführt und das Ergebnis in D# wird mit derselben Genauigkeit ausgegeben.

```
10 D = 6#/7
20 PRINT D
RUN
 .857143
```

Die arithmetische Operation wird mit doppelter Genauigkeit ausgeführt. Das Ergebnis wird nach D gespeichert, gerundet, und mit einfacher Genauigkeit ausgegeben.

- Logische Operatoren (siehe Seite 1—11) wandeln die Operanden in ganzzahlige Werte um und ergeben ein ganzzahliges Resultat. Falls die Operanden außerhalb des Bereiches —32768 bis +32767 liegen, erscheint die Fehlermeldung "OVERFLOW" (Überlauf).

- Wenn ein Wert mit Gleitkomma in einen ganzzahligen Wert umgewandelt werden muß, werden die Dezimalstellen gerundet.
  Beispiel:

```
10 C% = 55.88
20 PRINT C%
RUN
 56
```

- Falls einer Variablen mit doppelter Genauigkeit ein Wert mit einfacher Genauigkeit zugeordnet wird, sind nur die ersten sieben Ziffern der übertragenen Zahl genau. Das kommt davon, daß die Zahl mit einfacher Genauigkeit nur 7 Ziffern hatte. Der absolute Wert beider Zahlen weicht jedoch nur um weniger als 6.3E—8 multipliziert mit dem Wert einfacher Genauigkeit, ab.
  Beispiel:

```
10 A = 2.04
20 B# = A
30 PRINT A;B#
RUN
 2.04   2.039999961853027
```

AUSDRÜCKE UND OPERATOREN

Ein Ausdruck kann eine Konstante oder eine Variable sein. Er kann aber auch Konstante und Variable mit Operatoren so verbinden, daß sie einen einzigen Wert ergeben.

Operatoren dienen dazu, mathematische oder logische Operationen mit Werten auszuführen. Die Operatoren in MS-BASIC können in vier Kategorien eingeteilt werden:

- arithmetische
- vergleichende
- logische
- funktionelle.

Arithmetische Operatoren

Die arithmetischen Operatoren in ihrer Rangfolge

Operator	Operation	Beispiel eines Ausdruckes
^	Exponentation (hoch)	X^Y
–	Negierung	–X
*,/	Multiplikation, Gleitkomma-Division	X*Y X/Y
+,–	Addition, Subtraktion	X+Y, X–Y

Um die Reihenfolge, in der Operationen durchgeführt werden, zu ändern, sind Klammern zu benutzen. Klammerinhalte werden zuerst ermittelt. Innerhalb einer Klammer gilt die normale Rangfolge.

Hier einige Beispiele algebraischer Ausdrücke und ihre BASIC-Schreibweise:

Algebraischer Ausdruck	BASIC-Ausdruck
X+2Y	X+Y*2
$X-\frac{Y}{Z}$	X–Y/Z
$\frac{XY}{Z}$	X*Y/Z
$\frac{X+Y}{Z}$	(X+Y)/Z
$(X^2)^Y$	(X^2)^Y
X^{Y^Z}	X^(Y^Z)
X(–Y)	X*(–Y) Zwei aufeinanderfolgende Operatoren müssen durch eine Klammer getrennt werden.

Ganzzahl-Division und Modulus-Arithmetic – Die "Erweiterte" und die "Platten"-Version von MS-BASIC kennen zwei weitere Operationen: die Ganzzahl-Division und die Modulus-Arithmetic.

Die Ganzzahl-Division wird durch den Operator Schrägstrich links (\) erreicht. Die Operanden werden vor Ausführung der Division zu ganzen Zahlen gerundet (sie müssen im Bereich von −32768 bis +32767 sein) und der Quotient wird ebenfalls als ganze Zahl ausgegeben (ungerundet).

Beispiele:

```
10\4 = 2
25.68\6.99 = 3
```

Die Ganzzahl-Division steht in der Rangfolge direkt hinter Multiplikation und Gleitkomma-Division.

Die Modulus-Arithmetik wird durch den Operator MOD eingeleitet. Sie ermittelt den ganzzahligen Rest einer Ganzzah-Division.

Beispiele:

```
10.4 MOD 4 = 2 (10/4=2, Rest 2).
25.68 MOD 6.99 = 5 (26/7=3, Rest 5)
```

Die Modulus-Arithmetik steht in der Rangfolge direkt hinter der Ganzzahl-Division.

Überlauf und Division durch Null – Falls während der Berechnung eines Ausdruckes eine Division durch Null auftaucht, wird die Fehlermeldung "DIVISION BY ZERO" angezeigt. Der "Unendlich-Wert" des Systems mit dem Vorzeichen des Dividenden wird als Ergebnis der Division festgelegt und der Programmablauf wird fortgesetzt. Falls die Berechnung einer Exponentierung eine Null hoch eine negative Zahl ergibt, wird ebenfalls "DIVISION BY ZERO" angezeigt und der positive "Unendlich-Wert" des Systems wird als Ergebnis der Exponentierung angenommen, bevor das Programm fortgesetzt wird.

Falls ein Datenüberlauf erfolgt, wird die Fehlermeldung "OVERFLOW" angezeigt, der "Unendlich-Wert" des Systems mit dem algebraisch richtigen Vorzeichen wird als Ergebnis festgelegt und der Programmablauf wird fortgesetzt.

Vergleichende Operatoren

Diese dienen dazu, zwei Werte zu vergleichen. Das Ergebnis des Vergleichs ist entweder "richtig" (—1) oder "falsch" (0). Dieses Ergebnis kann dann zur Entscheidung über den weiteren Programmablauf benutzt werden (siehe IF, Seite 2-35).

Operator	Vergleichstext	Ausdruck
=	Gleichheit	X=Y
<>	Ungleichheit	X<>Y
<	kleiner als	X<Y
>	größer als	X>Y
<=	kleiner oder gleich	X<=Y
>=	größer oder gleich	X>=Y

(Das Gleichheitszeichen wird auch zur Zuweisung von Werten an Variable benutzt. Siehe LET, Seite 2—40).

Wenn arithmetische und vergleichende Operatoren in einem Ausdruck kombiniert verwendet werden, so werden immer zuerst die arithmetischen Operationen ausgeführt. Als Beispiel, der Ausdruck

```
X+Y<(T—1)/Z
```

ist dann "richtig", wenn der Wert von X plus Y kleiner ist, als der Wert von T minus 1 geteilt durch Z.

Weitere Beispiele:

```
IF SIN(X) < 0 GOTO 1000
IF I MOD J<>0 THEN K=K+1
```

Logische Operatoren

Sie führen Mehrfachvergleiche, Bit-Manipulationen oder Boole'sche Operationen aus. Der logische Operator bewirkt ein Bit-Ergebnis, das entweder "richtig" (ungleich Null) oder "falsch" (gleich Null) ist. In Ausdrücken werden logische Operationen nach arithmetischen und vergleichenden Operationen ausgeführt. Das Ergebnis von logischen Operationen wird wie nachfolgend beschrieben, ausgeführt. Die Operatoren sind gemäß ihrer Rangordnung gelistet.

NOT

X	NOT X
1	0
0	1

AND

X	Y	X AND Y
1	1	1
1	0	0
0	1	0
0	0	0

OR

X	Y	X OR Y
1	1	1
1	0	1
0	1	1
0	0	0

XOR

X	Y	X XOR Y
1	1	0
1	0	1
0	1	1
0	0	0

IMP

X	Y	X IMP Y
1	1	1
1	0	0
0	1	1
0	0	1

EQV

X	Y	X EQV Y
1	1	1
1	0	0
0	1	0
0	0	1

Genauso wie vergleichende Operatoren für Programmablauf-Entscheidungen benutzt werden, können logische Operatoren zwei oder mehrere Vergleiche verbinden, und einen "richtigen" oder "falschen" Wert für eine Entscheidung ermitteln (siehe IF, Seite 2-35).

Beispiele:

```
IF D<200 AND F<4 THEN 80
IF I>10 OR K<0 THEN 50
IF NOT P THEN 100
```

Logische Operatoren wandeln die Operanden in 16 Bit große Binärkomplemente zwischen —32768 und +32767 mit Vorzeichen um. (Wenn die Operanden außerhalb dieses Bereiches liegen, wird ein Fehler angezeigt). Falls beide Operanden 0 oder —1 als Wert ergeben, ermitteln die logischen Operatoren als Ergebnis 0 oder —1. Die logische Operation wird mit den umgewandelten Operatoren bitweise durchgeführt; d.h. jedes Bit des Ergebnisses wird durch die entsprechenden Bits der beiden Operatoren bestimmt.

Dadurch ist es möglich, logische Operatoren zum Suchen nach besonderen Bitmustern in Bytes zu benutzen. Der AND-Operator kann z.B. dazu benützt werden, um alle Bits eines I/O-Ausganges bis auf eines zu setzen. Der OR-Operator kann benutzt werden, um zwei Bytes zu einem bestimmten Binärwert zu mischen. Die nachfolgenden Beispiele sollen aufzeigen, wie logische Operatoren arbeiten.

63 AND 16 = 16	63 = binär 111111 und 16 = binär 10000, daher ist 63 AND 16 = 16
15 AND 14 = 14	15 = binär 1111 und 14 = binär 1110, also 15 AND 14 = 14 (binär 1110)
—1 AND 8 = 8	—1 = binär 1111111111111111 und 8 = binär 1000, daher —1 AND 8 = 8
4 OR 2 = 6	4 = binär 100 und 2 = binär 10, daher ist 4 OR 2 = 6 (binär 110)
10 OR 10 = 10	10 = binär 1010, daher 1010 OR 1010 = 1010(10)
—1 OR —2 = —1	—1 = binär 1111111111111111 und —2 = binär 1111111111111110, daher ist —1 OR —2 = —1. Der Komplementärwert von 16 Nullen im Bitimage ist 16mal die Eins, welches das Zweierkomplement der Zahl —1 ergibt.
NOT X = —(X+1)	Das Zweierkomplement jeder ganzen Zahl ist das Bit-Komplement plus Eins.

Funktionale Operatoren

Funktionen werden benutzt, um in einem Ausdruck eine vorbestimmte Operation für einen Operanden aufzurufen. MS-BASIC hat integrierte Funktionen wie z.B. SQR (square root = Quadratwurzel) oder SIN (Sinus). Alle integrierten Funktionen von MS-BASIC sind in Kapitel 3 beschrieben.

MS-BASIC ermöglicht dem Programmierer auch, selbst Funktionen zu definieren, siehe DEF FN, Seite 2—13.

Text Operationen (alphanumerische Operationen)

Texte können durch + zusammengefügt werden, z.B.:

```
10 A$ = "DATEI": B$= "NAME"
20 PRINT A$ + B$
30 PRINT "NEUER"+A$+ B$
RUN
DATEINAME
NEUER DATEINAME
```

Texte können mit Hilfe der gleichen Operatoren verglichen werden wie numerische Werte:

```
=<>  <   >   <=   >=
```

Texte werden zeichenweise verglichen. Dabei wird der ASCII-Kode als Vergleichsbasis benützt. Sind alle ASCII-Kodes gleich, so sind die beiden Operanden gleich. Falls sie differieren, wird der niedrigere Kode vor den höheren eingeordnet. Ist ein Operand kürzer als der andere, wird er als kleiner angenommen. Beim Vergleich werden alle Leerzeichen mit verglichen.

Beispiele:

```
"AA" < "AB"
"DATEINAME" = "DATEINAME"
"X&" > "X#"
"CL" < "cl"
"kg" > "KG"
"SCHMID" < "SCHMIED"
B$< "9.12.1982" (B$ sei "8.12.1982")
```

Textvergleiche können benützt werden, um alphanumerische Werte zu prüfen oder Texte alphabetisch zu ordnen. Alle Text-Konstanten, die in Vergleichen benützt werden, müssen in Anführungszeichen eingebettet sein.

KORRIGIERENDE EINGABE

Falls beim Eingeben einer Zeile eine falsche Taste bestätigt wurde, kann das Zeichen durch Betätigung der Taste RUBOUT (DEL) oder Control-H (Rücktaste) gelöscht werden. Während RUBOUT das Zeichen mit zwei Schrägstrichen umrahmt, wird es durch Control-H auch optisch gelöscht. Anschließend kann einfach weiter eingegeben werden.

Um eine ganze Zeile zu löschen, die gerade eingegeben wird, muß Control-U betätigt werden. Anschließend wird automatisch die Funktion <Carriage Return (Eingabetaste)> ausgeführt.

Um eine Zeile für ein Programm zu korrigieren, das gerade im Speicher ist, genügt es, diese Zeile mit der gleichen Zeilennummer neu einzugeben. Dadurch wird die alte Programmzeile überschrieben.

Weitere Korrekturmöglichkeiten gibt es in den Versionen "Erweitert" und "Platte" von MS-BASIC. Siehe dazu EDIT, Kapitel 2.

Um das gesamte Programm im Speicher zu löschen, genügt es, den Befehl NEW einzugeben (siehe Kapitel 2). NEW wird benutzt, um den Speicher vor der Eingabe eines neuen Programmes zu löschen.

FEHLERMELDUNGEN

Wenn MS-BASIC einen Fehler entdeckt, der eine Programmunterbrechung bewirkt, wird eine Fehlermeldung ausgegeben. Im Anhang A ist eine komplette Übersicht über die Fehlerkodes und Fehlermeldungen von MS-BASIC zu finden.

KAPITEL 2

MS-BASIC; BEFEHLE UND ANWEISUNGEN

In diesem Kapitel werden alle in MS-BASIC verfügbaren Befehle und Anweisungen beschrieben. Jede Beschreibung folgt demselben Schema:

Format: Hier wird das korrekte Format der Instruktion gezeigt. Siehe dazu die untenstehenden Regeln des Instruktionsformates.

Zweck: Beschreibt, wozu die Instruktion dient.

Bemerkung: Beschreibt genau, wie die Instruktion anzuwenden ist.

Beispiel: Zeigt Beispiele, die den Gebrauch der Instruktion erläutern.

REGELN DES INSTRUKTIONSFORMATES

Das Format jeden Befehles und jeder Anweisung folgt den nachstehenden Regeln.

GROSS — Positionen in Großbuchstaben sind einzugeben wie aufgezeigt.

< > — Positionen in Kleinbuchstaben innerhalb von Spitzklammern (< >) werden vom Programmierer definiert.

[ ] — Positionen in eckigen Klammern ([ ]) können wahlweise benutzt werden.
Alle Satzzeichen außer eckigen und Spitzklammern sind wie vorgegeben einzusetzen (z.B. Kommas, Klammern, Strichpunkte, Doppelpunkte, Bindestriche, Gleichheitszeichen).

... — Positionen, hinter denen eine Punktreihe ist (...), können beliebig oft innerhalb einer logischen Zeile wiederholt werden.

{ } — Geschweifte Klammern schließen eine von mehreren Wahlmöglichkeiten ein: Für die MS-BASIC-Anweisung müssen Sie sich für mindestens eine der in geschweiften Klammern enthaltenen Möglichkeiten

entscheiden, es sei denn, die Wahlmöglichkeiten sind zusätzlich in eckigen Klammern angegeben. In diesem Fall dürfen Sie auf diesen Teil der Anweisung verzichten.

| Senkrechte Trennungsstriche dienen der Trennung von Wahlmöglichkeiten, die in geschweiften Klammern angegeben sind.

Mit Ausnahme der Spitzklammern und der eckigen Klammern sind alle in der Beschreibung einer MS-BASIC-Anweisung angegebenen Satzzeichen (Komma, Semikolon, Bindestrich, Gleichheitszeichen, runde Klammern) als unerläßliche Bestandteile der Anweisung zu betrachten.

AUTO

Format: AUTO [<zeilennummer>[,<abstand>]]

Zweck: Dient zur automatischen Generierung von Zeilennummern jeweils nach Betätigen der Eingabetaste.

Bemerkung: AUTO beginnt mit der Numerierung bei <zeilennummer> und erhöht jede nachfolgende Zeile um <abstand>. Werden die Werte nicht eingegeben, so nimmt MS-BASIC jeweils den Wert 10 an. Wird auf <zeilennummer> zwar ein Komma, jedoch kein <abstand> eingegeben, so wird automatisch der Wert <abstand> vom zuletzt eingegebenen AUTO-Befehl übernommen.

Falls AUTO eine Zeilennummer generiert, die bereits vorhanden ist, wird nach der Zeilennummer ein Stern angezeigt, um den Programmierer vor dem versehentlichen Überschreiben einer bestehenden Programmzeile zu warnen. Wird nur die Eingabetaste betätigt, ohne weitere Eingabe, so bleibt die ursprüngliche Programmzeile erhalten und die nächste Zeilennummer wird generiert.

AUTO wird durch Control-C abgebrochen. Die Zeile, in der Control-C betätigt wurde, geht dabei verloren. Nach Abbruch von AUTO kommt MS-BASIC in den Direktmodus.

Beispiel:

AUTO 100,50	generiert Zeilennummern 100, 150, 200 ...
AUTO	generiert Zeilennummern 10, 20, 30, 40 ...

CALL

Format: CALL <variable> [(<argumente>)]

Zweck: CALL dient dazu, Unterprogramme aufzurufen, die in Assemblersprache geschrieben wurden.

Bemerkung: Der CALL-Befehl ist eine der Möglichkeiten, den Programmablauf einem Unterprogramm, das in Assemblersprache geschrieben wurde, zu übertragen. (Siehe auch die USR-Funktion, Seite 3-23).

<variable> enthält die Speicheradresse, an der das Unterprogramm beginnt. <variable> darf keine Gruppenvariable sein. <argumente> enthält die Argumente, die an das Unterprogramm übergeben werden. <argumente> darf keine alphanumerischen Konstanten enthalten.

Der Befehl CALL generiert dieselbe Aufruffolge wie die Microsoft Compiler für FORTRAN, COBOL und BASIC.

Beispiel:
```
110 UNTERPROG1 = &HD000
120 CALL UNTERPROG1 (I,J,K)
 .
 .
 .
```

Hinweis: In einem MS-BASIC Compiler-Programm ist die Zeile 110 nicht erforderlich, weil die Adresse von MYROUT durch den "Linking Loader" während der Ladezeit zugeordnet wird.

CHAIN

Format: CHAIN [MERGE]<programmname>
[,[<zeilennummer/ausdruck>][,ALL][, DELETE
<bereich>]]

Zweck: CHAIN dient zum Aufruf anderer Programme mit Übergabe von Daten aus dem laufenden Programm

Bemerkung: <programmname> bezeichnet das aufzurufende Programm. Beispiel: Beispiel:

CHAIN "PROG1"

<zeilennummer/ausdruck> ist eine Zeilennummer oder ein Ausdruck, der eine Zeilennummer ergibt. Sie bezieht sich auf das aufzurufende Programm. Sie gibt den Anfangspunkt für den Programmablauf an. Wird sie nicht angegeben, so beginnt der Programmablauf mit der ersten Zeile. Beispiel:

CHAIN "PROG1", 1000

<zeilennummer/ausdruck> wird durch den Befehl RENUM nicht verändert.

Mit der Angabe ALL wird erreicht, daß alle Variablen des laufenden Programmes an das aufgerufene Programm übergeben werden. Wenn ALL nicht angegeben wird, muß im laufenden Programm in einer COMMON-Anweisung die Liste der zu übergebenden Variablen enthalten sein. Siehe Seite 2-9. Beispiel:

CHAIN "PROG1", 1000, ALL

Falls MERGE angegeben wird, kann ein Unterprogramm in Überlagerungstechnik ins laufende Programm eingebunden werden. D.h., die MERGE-Funktion wird sowohl im laufenden, als auch im aufgerufenen Programm ausgeführt. Das aufzurufende Programm muß im ASCII-Format gespeichert sein, um mit MERGE aufgerufen zu werden. Beispiel:

CHAIN MERGE "UNTERPROG1", 1000

Nachdem eine Überlagerung geladen ist, ist es üblich, sie wieder zu löschen, um Platz für andere Überlagerungen zu schaffen. Um dies zu erreichen, wird die DELETE-Angabe benutzt. Beispiel:

CHAIN MERGE "UNTERPROG2", 1000,
DELETE 1000-5000

Die Zeilennummern in der Angabe <bereich> werden vom Befehl RENUM verändert.

Anmerkung: Der BASIC-Compiler von Microsoft unterstützt die Angaben ALL, MERGE und DELETE im Befehl CHAIN *nicht*. Sollen die erstellten Programme kompatibel zum BASIC-Compiler sein, so empfiehlt es sich, Variablen mit COMMON zu übergeben und auf die Überlagerungstechnik zu verzichten.

Anmerkung: Der Befehl CHAIN mit dem Eintrag MERGE läßt die Datei geöffnet und behält den laufenden Status von OPTION BASE bei.

Anmerkung: Falls die MERGE-Option nicht benutzt wird, behält CHAIN die vorgegebenen Variablentypen bzw. Anwenderfunktionen im aufgerufenen Programm nicht bei. D.h. jede DEFINT, DEFSNG, DEFDBL, DEFSTR oder DEFFN-Anweisung, die eine gemeinsame Variable betrifft, muß im aufgerufenen Programm neu aufgeführt werden.

CLEAR

Format: CLEAR [,[<ausdruck1>][,<ausdruck2>]]

Zweck: Dieser Befehl dient dazu, alle numerischen Variablen auf Null zu setzen und alle Stringvariablen zu löschen. Wahlweise kann das Speicherende gesetzt werden und die Größe des Verkettungsbereiches (stack space) definiert werden.

Bemerkung: <ausdruck 1> ist eine Speicherstelle, die, falls angegegeben, die höchste für MS-BASIC zur Verfügung stehende Speicherposition definiert.

<ausdruck 2> setzt den Verkettungsbereich für MS-BASIC. Automatisch werden entweder 256 Bytes oder ein Achtel des verfügbaren Speichers (jeweils der kleinere Wert) angenommen.

Anmerkung: In früheren Ausgaben von MS-BASIC wurde durch <ausdruck1> der verfügbare Textbereich und durch <ausdruck2> das Speicherende definiert. Ab Ausgabe 5.0 von MS-BASIC wird der Textbereich dynamisch definiert. Die Fehlermeldung "Out of string space" (Ende des Textbereiches) erscheint nur, wenn kein Speicherplatz mehr für BASIC zur Verfügung steht.

Beispiel:
```
CLEAR
CLEAR, 32768
CLEAR,,2000
CLEAR,32768,2000
```

CLOAD

Format: CLOAD <programmname>
CLOAD? <programmname>
CLOAD* <name einer gruppen-variablen>

Zweck: Dient dazu, ein Programm oder eine Gruppen-Variable von der Kassette in den Speicher zu laden.

Bemerkung: CLOAD führt den Befehl NEW aus, bevor es ein Programm von der Kassette lädt. <programmname> ist der String-Ausdruck oder das erste Zeichen desselben, der bei der Ausgabe des Programmes auf Kassette mittels CSAVE spezifiziert wurde.
CLOAD? vergleicht das Programm im Speicher mit dem Programm auf Kassette, das den gleichen Programmnamen hat. Wenn sie übereinstimmen, zeigt MS-BASIC OK an, andernfalls wird NO GOOD angezeigt.

CLOAD* lädt eine numerische Gruppen-Variable, die auf Kassette gespeichert wurde. Die Werte von der Kassette werden in die Tabelle mit dem <namen der gruppenvariablen> eingelesen, falls sie mit dem Befehl CSAVE auf die Kassette geschrieben wurden.

CLOAD und CLOAD? werden immer im Direktmodus als Befehl eingegeben. CLOAD* kann sowohl als Befehl im Direktmodus, als auch als Anweisung in einem Programm vorkommen. Es ist sicherzustellen, daß die Tabelle vorher mit DIM definiert wurde. MS-BASIC kehrt nach den Befehlen CLOAD, CLOAD? und CLOAD* immer in den Direktmodus. Vor der Eingabe des CLOAD-Befehles muß beachtet werden, das Kassettengerät richtig anzuschließen, die Abspielfunktion anzuschalten und die Kassette richtig zu positionieren.

Siehe auch CSAVE, Seite 2-11.

Anmerkung: CLOAD und CSAVE sind nicht in allen Implementierungen von MS-BASIC enthalten.

Beispiel: CLOAD "MAX2"
lädt das Programm "M" in den Speicher.

CLOSE

Format: CLOSE[[#]<dateinummer>[,[#]<dateinummer...>]]

Zweck: Abschluß der Ein-/Ausgabe einer Plattendatei

Bemerkung: <dateinummer> ist diejenige Nummer, unter welcher die Ein-/Ausgabe für diese Datei eröffnet wurde (OPEN). Eine CLOSE-Anweisung ohne Zusätze schließt alle eröffneten Dateien ab.

Die Zuordnung einer bestimmten Datei zu einer Dateinummer wird durch Ausführung von CLOSE aufgehoben. Die Datei kann anschließend mit einer beliebigen Dateinummer wiedereröffnet werden. Gleichermaßen kann die Dateinummer einer beliebigen Datei zugeordnet werden.

Eine CLOSE-Anweisung für eine sequentielle Ausgabe auf eine Datei schreibt zuerst noch den Pufferinhalt in die Datei.

Die Befehle END und NEW führen immer eine CLOSE-Anweisung für alle Plattendateien automatisch aus. (STOP schließt keine Dateien).

Beispiel: Siehe Anhang B.

COMMON

Format: COMMON <liste von variablen>

Zweck: Dient der Übergabe von Variablen an ein Programm, das mit CHAIN aufgerufen wird.

Bemerkung: Die Anweisung COMMON wird im Zusammenhang mit der Anweisung CHAIN benutzt. Anweisungen können an beliebiger Position in einem Programm stehen. Es wird jedoch empfohlen, sie am Beginn anzuordnen. Eine Variable darf nicht in mehreren COMMON-Anweisungen aufgeführt werden. Gruppen-Variablen werden durch das Anhängen von Klammern "()" an den Variablennamen gekennzeichnet. Sollen alle Variablen eines Programmes übergeben werden, empfiehlt es sich, CHAIN mit dem Zusatz ALL zu verwenden und COMMON nicht zu verwenden.

Beispiel:
```
100 COMMON A,B,C,D (),G$
110 CHAIN "PROGR3",10
      .
      .
      .
```

Hinweis: Der MS-BASIC Compiler unterstützt eine modifizierte Version der COMMON Anweisung. Die COMMON Anweisung muß in einem Programm vor irgendwelchen durchführbaren Anweisungen erscheinen. Die derzeit nichtbenutzbaren Anweisungen sind:

COMMON
DEFDBL, DEFINT, DEFSNG, DEFSTR
DIM
OPTION BASE
REM
$INCLUDE

In einer COMMON-Anweisung verwendete Gruppen-Variablen sind durch eine vorangestellte DIM-Anweisung zu kennzeichnen.

Die Standard Form der COMMON Anweisung ist als leeres COMMON zu verstehen.

Namentliche "Compiler-Style" COMMON-Bereiche von Microsoft FORTRAN werden ebenso unterstützt; aber auch hier sind die Variablen nicht quer über CHAIN$ erhältlich. Der Syntax für namentliche COMMON ist:

COMMON [<Name>] <Liste der Variablen>

wobei (Name) aus 1 bis 6 alphanumerischen Zeichen besteht, beginnend mit einem Buchstaben. Dies ist anwendbar für die Kommunikation mit dem Microsoft FORTRAN Compiler sowie Assembliersprachen Routinen ohne daß Parameter ausdrücklich in der CALL Anweisung zu erklären sind.

Die leere Common Größe und Reihenfolge der Variablen muß in CHAIN-Programmen gleich sein. Dies läßt sich mit dem MSBASIC Compiler am besten sicherstellen, indem alle leeren COMMON Erklärungen in einem Einzel "Include-File" angeordnet werden, unter Verwendung der $INCLUDE Anweisung in jedem Programm.

Beispiel: MENU.BAS

```
10 $INCLUDE COMDEF
 .
 .
 . 1000 CHAIN "PROG1"
```

PROG1.BAS

```
10$INCLUDE COMDEF
 .
 .
 . 2000 CHAIN "MENU"
```

COMDEF.BAS

```
100 DIM A(100),B$(200)
110 COMMON J,J,K,A()
120 COMMON A$,B$(),X,Y, Z
```

CONT

Format: CONT

Zweck: Dient zur Fortsetzung eines Programmes, das mit Control-C unterbrochen wurde bzw. eine STOP- oder END-Anweisung ausführte.

Bemerkung: Das Programm wird an der Stelle fortgesetzt, an der es unterbrochen wurde. Falls die Unterbrechung an einer Stelle stattfand, an der die Aufforderung zur Dateneingabe in einer INPUT-Anweisung erschien, wird das Programm mit dem erneuten Aufzeigen dieser Aufforderung (? oder Text) fortgesetzt.

CONT wird normalerweise zusammen mit STOP bei der Fehlersuche verwendet. Nach der Programmunterbrechung können Zwischenwerte überprüft und geändert werden. Dies geschieht im Direktmodus. Die Programmausführung wird mit CONT oder einem GOTO im Direktmodus fortgesetzt. Mit GOTO kann die Fortsetzung bei einer definierten Zeilennummer gestartet werden. Bei den Versionen "Erweitert" und "Platte" kann CONT zur Programmfortsetzung nach Fehlern benützt werden.

CONT ist nicht möglich, wenn das Programm während der Unterbrechung verändert wurde.

Beispiel: Siehe Beispiel auf Seite 2-82 (STOP)

CSAVE

Format: CSAVE <string ausdruck>
CSAVE*<gruppen-variable>

Zweck: Dient zum Speichern von Programmen oder Tabellen aus dem Systemspeicher auf eine Kassette.

Bemerkung: Jedes Programm oder jede Tabelle hat auf der Kassette einen eigenen Dateinamen. Wenn der Befehl CSAVE <string ausdruck> ausgeführt wird, nimmt MS-BASIC das sich momentan im Speicher befindliche Programm, schreibt es auf Kassette und nimmt den ersten Buchstaben aus <string ausdruck> als Dateinamen. Auch wenn <string ausdruck> aus mehreren Zeichen besteht, wird nur das erste als Dateiname zugeordnet.

Wenn der Befehl CSAVE* <gruppen-variable> ausgeführt wird, nimmt MS-BASIC die spezifizierte Tabelle und schreibt sie auf Kassette. Die Tabelle muß numerisch sein. Mehrdimensionale Tabellen werden so abgespeichert, daß der jeweils linke Subskriptionswert am schnellsten variiert.

CSAVE kann sowohl direkt, als auch im Programm erscheinen.

Vor Ausführung von CSAVE bzw. CSAVE* ist sicherzustellen, daß das Kassettengerät richtig angeschlossen ist und die Aufzeichnungsfunktion angeschaltet ist.

Siehe auch CLOAD, Seite 2-7.

Anmerkung: CSAVE und CLOAD sind nicht in allen Implementierungen von MS-BASIC enthalten.

Beispiel: CSAVE "TAGABSCHL"

Schreibt das Programm, das sich im Speicher befindet, auf die Kassette und ordnet den Dateinamen "T" zu.

DATA

Format: DATA < liste von konstanten>

Zweck: Dient zum Speichern von numerischen und alphanumerischen (string-) Konstanten, die durch die READ-Anweisung(en) eines Programmes abgefragt werden. (Siehe READ, Seite 2-73).

Bemerkung: DATA-Anweisungen sind nicht ausführbar und können an beliebiger Position innerhalb eines Programmes stehen. Eine DATA-Anweisung kann soviele Konstanten enthalten, als in eine Programmzeile passen (Trennung durch Kommas). Eine beliebige Anzahl von DATA-Anweisungen kann in einem Programm vorkommen. Die READ-Anweisungen greifen auf die DATA-Anweisungen sequentiell in der Reihenfolge der Zeilennummern zu. Die Konstanten aller DATA-Anweisungen kann man sich auch als eine einzige ununterbrochene Reihe vorstellen. Unabhängig davon ist, wieviele Konstanten in einer Anweisung stehen und wo diese Anweisungen innerhalb des Programmes plaziert sind.

< liste von konstanten > kann numerische Konstanten aller Formate enthalten (d.h. Festpunkt-, Fließkomma- oder Ganzzahl-Konstanten). Numerische Ausdrücke sind nicht erlaubt. String-Konstanten in DATA-Anweisungen müssen dann in Anführungszeichen stehen, wenn die Kommas, Doppelpunkte oder führende bzw. nachfolgende Leerzeichen enthalten. Andernfalls können die Anführungszeichen entfallen.

Der Typ der Variablen (numerisch oder alphanumerisch) in der READ-Anweisung muß mit der entsprechenden Konstanten in der DATA-Anweisung übereinstimmen.

DATA-Anweisungen können durch die RESTORE-Anweisung (Seite 2-78) von vorne gelesen werden.

Beispiel: Siehe Beispiele auf den Seiten 2-73 und 2-74 (READ).

DEF FN

Format: DEF FN <name> [(<parameter-liste>)] =
= <funktionsdefinition>

Zweck: Um eine vom Benutzer geschriebene Funktion zu definieren und benennen.

Bemerkung: <name> muß ein gültiger Variablenname sein. Dieser Name, unter Voranstellung von FN, wird der Name der Funktion. <parameter-liste> definiert jene Variablen in einer Funktion, die eingesetzt werden, wenn die Funktion aufgerufen wird. Die Parameter werden durch Kommas getrennt. <funktions-definition> ist ein Ausdruck, der die Funktion ausführt. Dieser Ausdruck ist auf eine Zeile beschränkt. Variablennamen, die in diesem Ausdruck erscheinen, dienen nur der Definition der Funktion. Sie berühren nicht Programmvariablen mit demselben Namen. Eine Variable der Funktionsdefinition kann wahlweise in der Parameter-Liste erscheinen. Wenn dies der Fall ist, so wird der Wert der Variablen beim Aufruf der Funktion eingesetzt, andernfalls wird der momentane Wert der Variablen genommen.

Die Variablen in der Parameter-Liste repräsentieren, auf der Basis eins zu eins, die Argumente (Variablen oder Werte), die beim Aufruf der Funktion einzusetzen sind. (In der 8K Version ist nur ein Argument in einem Funktionsaufruf erlaubt. Die DEF FN-Anweisung kann deshalb auch nur eine Variable enthalten.)

In den Versionen "Erweitert" und "Platte können vom Benutzer definierte Funktionen sowohl numerisch als auch alphanumerisch sein. In der 8K-Version sind alphanumerische Funktionen nicht möglich. Wenn eine Type im Funktionsnamen definiert ist, wird das Ergebnis des Ausdruckes in diesen Typ gebracht bevor es an die aufrufende Anweisung übergeben wird. Wird eine Type im Namen spezifiziert, die nicht mit derjenigen im Argument übereinstimmt, so erscheint die Fehlermeldung "Type mismatch".

Die Anweisung DEF FN muß ausgeführt werden, bevor die entsprechende Funktion aufgerufen wird. Wird eine Funktion aufgerufen, bevor sie definiert wurde, so erscheint die Fehleranzeige "Undefined user function".

DEF FN kann nicht im Direktmodus benützt werden.

Beispiel:

```
410 DEF FNAB(X,Y)=S^3/Y^2
420 T=FNAB(I,J)
```

Zeile 410 definiert die Funktion FNAB, welche in Zeile 420 aufgerufen wird.

DEFINT/SNG/DBL/STR

Format: DEF<type> <bereich(e) von buchstaben>
wobei <type> gleich INT, SNG, DBL oder STR ist.

Zweck: Um Variablen als ganzzahlig, mit einfacher oder doppelter Genauigkeit, bzw. als String (alphanumerisch) zu deklarieren.

Bemerkung: Eine DEF-(Typen)-Anweisung erklärt alle Variablen, deren Name mit dem/den spezifizierten Buchstaben beginnen, zu dem angegebenen Typ. Immer hat jedoch eine direkte Typenerklärung beim Variablennamen Vorrang über eine Typenerklärung in einer DEF-Anweisung.

Wenn keine Typenerklärungen angegeben sind, nimmt MS-BASIC für alle Variablen den Typ einfache Genauigkeit an.

Beispiel: 10 DEFDBL L-P
Alle Variablen, deren Namen mit den Buchstaben L, M, N, O und P beginnen, haben doppelte Genauigkeit.

10 DEFSTR A
Alle Variablen, deren Namen mit A beginnen, sind String-Variable.

10 DEFINT I-N, W-Z
Alle Variablen, deren Namen mit den Buchstaben I, J, K, L, M, N, W, X, Y und Z beginnen, sind ganzzahlige Variable.

DEF USR

Format: DEF USR [<ziffer>] = <ganzzahliger ausdruck>

Zweck: Dient zur Festlegung der Anfangsadresse einer Unterroutine, die in Assemblersprache geschrieben wurde.

Bemerkung: <ziffer> kann den Wert 0 bis 9 haben. Die Ziffer bezieht sich auf die Nummer der USR-Routine, deren Anfangsadresse spezifiziert wird. Wenn <ziffer> nicht angegeben ist, wird DEF USR0 angenommen. Der Wert des <ganzzahligen ausdrucks> ist die Anfangsadresse der USR-Routine. Siehe Anhang C, Unterroutinen in Assemblersprache.

In einem Programm kann eine beliebige Anzahl von DEF USR-Anweisungen vorkommen, um Anfangsadressen von Unterroutinen festzulegen. Dadurch können alle benötigten Unterroutinen aufgerufen werden.

Beispiel:

```
.
.
.
200 DEF USR0=24000
210 X=USR0(Y^2/2.89)
.
.
.
```

DELETE

Format: DELETE [<zeilennummer>] [—<zeilennummer>]

Zweck: Dient dazu, Programmzeilen zu löschen.

Bemerkung: MS-BASIC kehrt nach Ausführung von DELETE immer in den Direktmodus zurück. Wenn die <zeilennummer> nicht vorhanden ist, wird der Fehler "illegal function call" (illegaler Funktionsaufruf) angezeigt.

Beispiel: DELETE 40
Löscht die Programmzeile 40

DELETE 40-100
Löscht die Zeilen 40 bis einschließlich 100

DELETE —40
Löscht alle Zeilen bis einschließlich Zeile 40.

DIM

Format: DIM <liste von indizierten variablen>

Zweck: DIM dient zur Angabe der maximalen Anzahl von Elementen von Gruppenvariablen und der Bereitstellung des entsprechenden Speicherplatzes.

Bemerkung: Wird eine Gruppenvariable ohne DIM-Anweisung benutzt, so wird als maximale Anzahl von Elementen 10 angenommen. Wird eine Gruppenvariable mit einem Index angesprochen, der größer als das spezifizierte Maximum ist, so erscheint die Fehlernachricht "Subscript our of range" (Index außerhalb des Bereiches). Der niedrigste mögliche Index ist immer 0, falls nicht mit der Anweisung OPTION BASE anders festgelegt (siehe Seite 2-58).

Die DIM-Anweisung setzt alle Elemente der spezifizierten Gruppenvariablen auf den Anfangswert Null.

Beispiel:

```
10 DIM A(20)
20 FOR I = 0 TO 20
30 READ A(I)
40 NEXT I
 .
 .
 .
```

EDIT

Format: EDIT <zeilennummer>

Zweck: Dient zum Einschalten des Korrekturmodus (EDIT-Mode) auf der spezifischen Programmzeile.

Bemerkung: Im Korrekturmodus ist es möglich, Teile einer Programmzeile zu korrigieren, ohne die Zeile neu eingeben zu müssen. Nach dem Einschalten des Korrekturmodus zeigt MS-BASIC die Nummer der zu korrigierenden Zeile an, macht ein Leerzeichen und wartet dann auf eine Korrekturanweisung.

Anweisungen im Korrekturmodus

Die Anweisungen im Korrekturmodus dienen zur Positionierung des Zeigers, zum Einsetzen, Löschen, Ersetzen von Zeichen oder zum Suchen von Text in einer Zeile. Die Anweisungen im Korrekturmodus werden nicht angezeigt. Von den meisten dieser Anweisungen kann ein ganzzahliger Wert eingegeben werden, der angibt, wie oft die Anweisung auszuführen ist. Wird keine Zahl angegeben, so wird die Anweisung einmal ausgeführt.

Die Anweisungen des Korrekturmodus können entsprechend den folgenden Funktionen eingeteilt werden:

Bewegen des Zeigers
Einsetzen von Text
Löschen von Text
Ersetzen von Text
Beenden und Wiederbeginn des Korrekturmodus

Anmerkung: In der nachfolgenden Beschreibung werden folgende Symbole verwendet:

<z>	steht für ein beliebiges Zeichen
<text>	steht für eine Zeichenfolge (string) beliebiger Länge
[i]	steht für eine wahlweise einzusetzende ganze Zahl (automatisch wird 1 angenommen)
$	steht für die Taste "Escape" (oder "Anderer Modus").

Bewegen des Zeigers

Leertaste: Die Leertaste bewegt den Zeiger nach rechts. [i]-Leertaste bewegt den Zeiger um "i" Positionen nach rechts. Die Zeichen in der Zeile werden so angezeigt, wie der Zeiger nach rechts bewegt wird.

Rubout: Im Korrekturmodus bewegt [i]-Rubout den Zeiger, um "i" Positionen nach links (Rückschritt). Die Zeichen der Zeile werden so angezeigt, wie der Zeiger nach links bewegt wird.

Einsetzen von Text

I I<text>**$** setzt <text> an der augenblicklichen Zeigerposition ein. Die eingesetzten Zeichen werden angezeigt. Um das Einsetzen zu beenden, wird die Escape-Taste betätigt. Falls die Eingabetaste in einer Einsetz-Anweisung betätigt wird, ist das Ergebnis genauso, als ob die Escape-Taste und anschließend die Eingabetaste betätigt worden wären. Während einer Einsetz-Anweisung können mit der Rubout- oder Delete-Taste Zeichen links vom Zeiger gelöscht werden. Wird versucht, ein Zeichen einzusetzen, das die Zeilenlänge über 255 bringt, so ertönt der Fehlerton (control-G) und das Zeichen wird nicht angezeigt.

X Die X-Anweisung dient zum Anfügen an die Zeile. X bewegt den Zeiger an das Ende der Zeile und geht dann in die Einsetz-Anweisung über. Jetzt kann Text eingesetzt werden wie bei der Einsetz-Anweisung. Nach Beendigung des Anfügens wird die Escape- oder die Eingabetaste betätigt.

Löschen von Text

D [i] D löscht "i" Zeichen rechts vom Zeiger. Die gelöschten Zeichen werden zwischen zwei "Schrägstrichen links" angezeigt. Der Zeiger steht rechts neben dem letzten gelöschten Zeichen. Ist der Rest der Zeile kleiner als "i", so wird bis zum Zeilenende gelöscht.

H — H löscht alle Zeichen rechts vom Zeiger und geht dann in die Einsetz-Anweisung. H wird benützt, um Anweisungen am Ende einer Zeile zu überschreiben.

Aufsuchen von Text

S — Die Anweisung [i] S <z> sucht "i"-mal das Zeichen <z> und positioniert den Zeiger links davon. Das Zeichen, auf dem der Zeiger steht, wird nicht in den Suchvorgang eingeschlossen. Wird <z> nicht gefunden; bleibt der Zeiger am Ende der Zeile stehen. Alle Zeichen der Zeile, die beim Suchen passiert werden, kommen zur Anzeige.

K — Die Anweisung [i] K <z> ist ähnlich der Anweisung [i] S <z> mit der Ausnahme, daß alle Zeichen zwischen der momentanen Zeigerposition und dem aufgesuchten Zeichen gelöscht werden. Der Zeiger wird vor dem Zeichen <z> positioniert und die gelöschten Zeichen werden zwischen zwei 'Schrägstrichten links' angezeigt.

Ersetzen von Text

C — Die Korrekturanweisung C <z> ersetzt das nächste Zeichen mit <z>. Sollen mehrere Zeichen ersetzt werden, ist die Anweisung iC gefolgt von "i" Zeichen zu verwenden. Sobald "i" Zeichen eingegeben sind, ist die Anweisung beendet.

Beenden und Wiederbeginn des Korrekturmodus

<Eingabetaste> — Die Betätigung der Eingabetaste zeigt den Rest der Zeile an, speichert alle Änderungen und beendet den Korrekturmodus.

E — Die Anweisung E hat das gleiche Ergebnis wie die Eingabetaste, jedoch ohne den Rest der Zeile anzuzeigen.

Q — Die Anweisung Q beendet den Korrekturmodus, ohne die Änderungen zu speichern.

L — Die Anweisung L zeigt den Rest der Zeile an (sie speichert alle bis dahin gemachten Änderungen) und positioniert den Zeiger wieder am Zeilenanfang. Das System bleibt im Korrekturmodus. L wird normalerweise benutzt, um die Zeile anzuzeigen, nachdem man den Korrekturmodus angewählt hat.

A Die Anweisung A dient zum Neubeginn einer Korrektur. Sie bringt die ursprüngliche Zeile zurück und stellt den Zeiger wieder an den Anfang der Zeile.

Anmerkung: Wird im MS-BASIC eine nicht identifizierbare oder unerlaubte Anweisung im Korrekturmodus eingegeben, so ertönt der Fehlerton (Control-G), und die Anweisung wird ignoriert.

Syntaxfehler (Satzaufbaufehler)

Wird während des Programmablaufes ein Syntaxfehler erkannt, so schaltet MS-BASIC automatisch auf der fehlerhaften Zeile in den Korrekturmodus um. Zum Beispiel:

```
10 K = 2 (4)
RUN
? Syntax error in 10 (Syntaxfehler in Zeile 10)
10
```

Wird nach der Korrektur die Eingabetaste betätigt, oder die Anweisung E eingegeben, setzt MS-BASIC die Zeile wieder ein. Dies bedeutet, daß alle Werte von Variablen verloren gehen. Sollen diese Werte jedoch überprüft werden, muß der Korrekturmodus mit der Anweisung Q beendet werden. MS-BASIC geht dann in den Direktmodus und alle Werte von Variablen bleiben erhalten.

Control-A

Um den Korrekturmodus auf der im Moment einzugebenden Zeile einzuschalten, ist Control-A zu betätigen. MS-BASIC antwortet mit einer Zeilenschaltung, einem Ausrufezeichen und einem Leerzeichen. Der Zeiger steht auf dem ersten Zeichen der Zeile. Nun kann eine Korrekturanweisung eingegeben werden.

Anmerkung: Soll eine soeben eingegebene Zeile korrigiert werden, so ist die Anweisung "EDIT" zu benützen, um in den Korrekturmodus zu kommen. (Das Zeilennummernsymbol "." bezieht sich jeweils auf die momentane Zeile).

END

Format: END

Zweck: Beendet ein Programm, schließt alle Dateien und bringt das System in den Direktmodus.

Bemerkung: Die END-Anweisung kann an jeder beliebigen Stelle in einem Programm stehen, um dessen Ausführung zu beenden. Im Gegensatz zur STOP-Anweisung, bewirkt END keine Anzeige einer Unterbrechungsmeldung. Am Ende eines Programmes kann die END-Anweisung auch entfallen. MS-BASIC kehrt nach einer END-Anweisung immer in den Direktmodus zurück.

Beispiel: 520 IF K > 100 THEN END ELSE GOTO 20

ERASE

Format: ERASE <liste von gruppen-variablen>

Zweck: Entfernen von Gruppen-Variablen aus einem Programm.

Bemerkung: Gruppen-Variable können nach einer ERASE-Anweisung neu dimensioniert werden. Der durch ERASE freigewordene Speicherplatz kann jedoch auch für andere Zwecke genutzt werden. Wird versucht, eine Gruppen-Variable neu zu dimensionieren, ohne daß vorher ERASE durchgeführt wurde, so wird die Fehlermeldung "Redimensioned array" (Gruppen-Variable bereits dimensioniert) angezeigt.

Anmerkung: Die Anweisung ERASE wird vom Microsoft BASIC Compiler nicht unterstützt.

Beispiel:

```
.
.
.
450 ERASE A, B
460 DIM B(99)
.
.
.
```

DIE VARIABLEN ERR UND ERL

Wenn eine Unterroutine zur Abarbeitung von Fehlern angelaufen wird, enthält die Variable ERR den Fehlerkode für diesen Fehler und die Variable ERL enthält die Nummer der Zeile, in der der Fehler entdeckt wurde. Die beiden Variablen ERR und ERL werden üblicherweise in IF ... THEN-Anweisungen verwendet, um den Programmablauf in der Fehlerroutine zu steuern.

Falls der Fehler von einem Befehl im Direktmodus verursacht wurde, enthält ERL den Wert 65535. Um festzustellen, ob der Fehler im Direktmodus verursacht wurde, kann IF 65535 = ERL THEN ...

Andernfalls

IF ERR = <fehlerkode> THEN ...
IF ERL = <zeilennummer> THEN ...

Wenn die Zeilennummer nicht auf der rechten Seite des vergleichenden Operators ist, wird sie beim Befehl RENUM nicht mit verändert. DA ERL und ERR reservierte Variablen sind, können sie nicht links vom Gleichheitszeichen in einer LET-(Zuordnungs-) Anweisung stehen. Die Fehlerkodes von MS-BASIC sind im Anhang in A aufgeführt.

ERROR

Format: ERROR <ganzzahliger ausdruck>

Zweck: 1) Zur Simulation von Fehlerfällen in MS-BASIC;
2) um dem Anwender die Definition von Fehlerkodes zu ermöglichen.

Bemerkung: Der Wert des <ganzzahligen ausdrucks> muß größer 0 und kleiner 255 sein. Entspricht der Wert des Ausdrucks einem bereits benutzten Fehlercode von MS-BASIC (Siehe Anfang F), dann simuliert die ERROR-Anweisung diesen Fehler und zeigt die entsprechende Fehlermeldung an (Siehe Beispiel 1).

Um einen eigenen Fehlerkode zu definieren, muß ein Wert über dem höchsten MS-BASIC Fehlerkode gewählt werden. (Es empfiehlt sich, die höchsten verfügbaren Werte anzusetzen, um kompatibel zu bleiben, falls MS-BASIC zusätzliche Fehlerkodes erhält.) Dieser vom Benutzer definierte Fehlerkode kann anschließend gut in einer Fehlerroutine abgehandelt werden (siehe Beispiel 2).

Wenn eine ERROR-Anweisung einen Kode spezifiziert, für den keine Fehlermeldung definiert wurde, zeigt MS-BASIC die Nachricht UNPRINTABLE ERROR (Fehlermeldung nicht vorhanden) an. Falls für eine ERROR-Anweisung keine Fehlerroutine vorhanden ist, wird eine Fehlermeldung angezeigt und das Programm abgebrochen.

Beispiel 1:

```
LIST
10 S = 10
20 T = 5
30 ERROR S+T
40 END
Ok
RUN
String too long in line 30
```

Oder, im Direktmodus

```
Ok
ERROR 15      (diese Zeile geben Sie ein)
String too long (diese Zeile kommt von MS-BASIC)
Ok
```

Beispiel 2:

```
.
.
.
110 ON ERROR GOTO 400
120 INPUT "WIEVIEL SETZEN SIE";B
130 IF B>5000 THEN ERROR 210
.
.
.
400 IF ERR = 210 THEN PRINT "MAXIMAL-
    EINSATZ IST 5000"
410 IF ERL = 130 THEN RESUME 120
```

FIELD

Format: FIELD [#]<dateinummer>,<feldlänge>
AS<string-variable>...

Zweck: Speicherplatzzuordnung für Variable in einem Puffer für Dateien mit wahlfreiem Zugriff.

Bemerkung: Um Daten aus einem "wahlfreien" Puffer nach einer GET-Anweisung herauszuholen oder sie von einer PUT-Anweisung in den Puffer reinzuschreiben, muß eine FIELD-Anweisung ausgeführt werden.

<dateinummer> ist die Nummer, unter der die Datei in der OPEN-Anweisung eröffnet wurde. <feldlänge> bezeichnet die Anzahl der Zeichen, die der <string-variablen> zugeordnet werden. Das Beispiel:

```
FIELD 1,20 AS N$, 10 AS ID$, 40 AS ADD$
```

ordnet die ersten 20 Positionen (= Stellen, Zeichen, Bytes) im "wahlfreien" Puffer der String-Variablen N$, die nächsten 10 Stellen ID$ und die nächsten 40 Stellen ADD$ zu. FIELD bringt KEINE Daten in den Puffer (siehe LSET/RSET und GET).

Die Gesamtzahl der in der FIELD-Anweisung zugeordneten Stellen darf die bei der Eröffnung der Datei spezifizierte Satzlänge nicht überschreiten. Andernfalls wird ein "Field overflow" (Feldüberlauf) als Fehler angezeigt. (Bei nicht spezifizierter Satzlänge wird automatisch 128 angesetzt.)

Eine beliebige Anzahl von FIELD-Anweisungen können für dieselbe Datei ausgeführt werden. Alle FIELD-Anweisungen, die angeführt wurden, sind gleichzeitig gültig.

Anmerkung: Eine Variable, die in einer FIELD-Anweisung benutzt wurde, darf in keiner INPUT- oder LET-Anweisung vorkommen. Sobald eine Variable in einer FIELD-Anweisung vorkam, wird sie einem Pufferbereich zugeordnet. Wird diese Variable anschließend in einer INPUT- oder LET-Anweisung benutzt, so wird sie dem Stringbereich zugeordnet.

Beispiel 1:
```
FIELD 1,20 AS N$,10 AS ID$,40 AS ADD$
```

Der Puffer für die mit 1 angesprochene Datei wird unter den String-Variablen N$ (die ersten 20 Bytes), ID$ (10 Bytes) und ADD$ (die letzten 40 Bytes) aufgeteilt. Der Puffer erhält allerdings erst bei der nächsten GET- oder LSET/RSET-Anweisung Daten.

Beispiel 2:
```
10 OPEN "R",#1, "A:TELEFON",35
15 FIELD #1,2 AS SATZNR$,33 AS ANDERE$
20 FIELD #1,25 AS NAME$,10 AS RUFNR$
25 GET #1
30 GESAMT=CVI(SATZNR$)
35 FOR I%=2 TO GESAMT
40 GET #1,I%
45 PRINT NAME$,RUFNR$
50 NEXT I%
```

Die Datensätze der in Zeile 10 eröffneten Datei können anhand zweier FIELD-Anweisungen aufgeteilt werden: In Zeile 15 sind nur die zwei ersten Bytes des 35-Byte- Datensatzes von Bedeutung. Sie enthalten die Anzahl der Einträge in diesem Telefonverzeichnis. Die zweite FIELD-Anweisung ist für die verbleibenden Datensätze vorgesehen. Die zwei FIELD-Anweisungen dürfen gleichzeitig bestehen, obwohl sie sich auf dieselbe Datei beziehen.

Beispiel 3:
```
10 FOR MEHRFACH%=0 TO 7
20 FIELD# 1,(MEHRFACH%*16) AS OFFSET$ AS
   A$(MEHRFACH%)
30 NEXT MEHRFACH%
```

Die gleichartigen Elemente einer Gruppenvariablen dienen dem Aufbau einer FIELD-Anweisung. Die folgende Anweisung bewirkt eine identische Aufteilung des Puffers:

```
FIELD# 1,16 AS A$(0),16 AS A$(1), ...,16 AS
A$(6),16 AS A$(7)
```

Beispiel 4:
```
10 DIM GRÖSSE%(ANZAHL%): REM Gruppe von
   FIELD-Größen
20 FOR MEHRFACH%=0 TO ANZAHL%: READ
   GRÖSSE%(MEHRFACH%): NEXT MEHRFACH%
```

```
 30 DATA 9,10,12,21,41
  .
  .
  .
120 DIM A$(ANZAHL%):REM  Die zu den FIELD-
    Bereichen gehörenden Variablen
130 OFFSET%=0
140 FOR MEHRFACH%=0 TO ANZAHL %
150 FIELD#1,OFFSET% AS OFFSET$,SIZE%
    MEHRFACH%) AS A$(MEHRFACH%)
160 OFFSET%=OFFSET%+GROESSE%MEHRFACH%)
170 NEXT MEHRFACH%
```

Die auf diese Weise eingerichteten FIELD-Bereiche entsprechen denen des Beispiels 3. Der wichtigste Unterschied besteht darin, daß die in Beispiel 4 eingerichteten FIELD-Bereiche unterschiedliche Größen aufweisen. Die folgende Anweisung bewirkt eine identische Aufteilung des Puffers:

```
FIELD#1,GROESSE(0) AS A$(0),GROESSE%(1) AS A$(1),
...,GROESSE%(ANZAHL%) AS A$(ANZAHL%)
```

FOR ... NEXT

Format: FOR <variable> = x TO y [STEP z]
.
.
.
NEXT [<variable>] [,<variable> ...]
(x, y und z sind numerische Ausdrücke)

Zweck: Ermöglicht es, eine Reihe von Anweisungen in einer Schleife mit einer gegebenen Anzahl von Durchläufen auszuführen.

Bemerkung: <variable> wird als Zähler benützt. Der erste numerische Ausdruck (x) ist der Ausgangswert des Zähler, der zweite (y) ist der Endwert des Zählers. Die auf die FOR-Anweisungen folgenden Programmzeilen werden ausgeführt, bis die Anweisung NEXT entdeckt wird. Dann wird der Zähler um den Wert (z) erhöht. Nun wird überprüft, ob der Zähler jetzt größer ist, als der Endwert (y). Falls nicht, verzweigt das MS-BASIC Programm zu der unmittelbar auf die FOR-Anweisung folgende Anweisung und der Vorgang wird wiederholt. Falls der Zähler jedoch größer ist, als der Endwert, wird das Programm mit der unmittelbar auf NEXT folgenden Anweisung fortgesetzt. Das ist eine FOR ... NEXT-Schleife. Wenn STEP nicht angegeben wird, nimmt MS-BASIC den Wert 1 an. Wenn STEP negativ angegeben wird, ist der Endwert kleiner als der Ausgangswert. Der Zähler wird um STEP reduziert, sooft die Schleife durchlaufen wird und der Vorgang wird solange wiederholt, bis der Zähler kleiner ist, als der Endwert.

Die Schleife wird dann nicht durchlaufen, wenn der Anfangswert multipliziert mit dem Vorzeichen von STEP den Endwert mulitpliziert mit dem Vorzeichen von STEP überschreitet.

Geschachtelte Schleifen

FOR ... NEXT-Schleifen können geschachtelt werden. D.h. eine solche Schleife kann innerhalb einer anderen solchen Schleife sein. Wenn Schleifen geschachtelt werden, muß jeder ein eindeutiger Variablenname für

den Zähler gegeben werden. Die NEXT-Anweisung für die innere Schleife muß vor der NEXT-Anweisung für die äußere Schleife verarbeitet werden. Wenn geschachtelte Schleifen denselben Endpunkt haben, können sie mit einer einzigen NEXT-Anweisung beendet werden.

Die Variablen in einer NEXT-Anweisung können weggelassen werden. In diesem Falle wird das NEXT dem zuletzt abgearbeiteten FOR zugeordnet. Wird eine NEXT-Anweisung entdeckt, bevor die entsprechende FOR-Anweisung bearbeitet wurde, so erscheint die Fehlernachricht "NEXT without FOR" (NEXT ohne FOR), und das Programm wird abgebrochen.

Beispiel 1:

```
10 K = 10
20 FOR I = 1 TO K STEP 2
30 PRINT I;
40 K = K + 10
50 PRINT K
60 NEXT
RUN
 1  20
 3  30
 5  40
 7  50
 9  60
Ok
```

Beispiel 2:

```
10 J = 0
20 FOR I = 1 TO J
30 PRINT I
40 NEXT J
```

In diesem Beispiel wird die Schleife nicht ausgeführt, weil der Ausgangswert den Endwert der Schleife übersteigt.

Beispiel 3:

```
10 I = 5
20 FOR I = 1 TO + I + 5
30 PRINT I;
40 NEXT
RUN
 1  2  3  4  5  6  7  8  9  10
Ok
```

In diesem Beispiel wird die Schleife zehnmal durchlaufen. Der Endwert für die Schleife wird immer vor dem Anfangswert gesetzt! (Achtung: Frühere Ausgaben von MS-BASIC haben den Anfangswert vor dem Endwert gesetzt. Das bedeutet, daß die obige Schleife sechsmal durchlaufen worden wäre.

GET

Format: GET [#]<dateinummer>[,<satznummer>]

Zweck: Dient zum Lesen eines Satzes einer wahlfreien Datei in einen wahlfreien Puffer.

Bemerkung: < dateinummer> ist die Nummer, unter der diese Datei in der OPEN-Anweisung eröffnet wurde. Wird <satznummer> nicht angegeben, so wird automatisch der nächste Satz (nach der letzten GET-Anweisung) eingelesen. Die höchste Satznummer ist 32767.

Beispiel: Siehe Anhang B.

Anmerkung: Nach einer GET-Anweisung können Zeichen aus dem Puffer mit den Anweisungen INPUT# und LINE INPUT# gelesen werden.

GOSUB ... RETURN

Format: GOSUB <zeilennummer>
.
.
.
RETURN

Zweck: Verzweigung zu einer und Rückkehr aus einer Unterroutine

Bemerkung: <zeilennummer> ist die Nummer der ersten Zeile der Unterroutine.

Eine Unterroutine kann in einem Programm beliebig oft aufgerufen werden. Sie kann auch von einer anderen Unterroutine aufgerufen werden. Solche geschachtelten Unterroutinen werden nur durch den verfügbaren Speicherbereich begrenzt.

Die RETURN-Anweisung(en) in einer Unterroutine bewirkt (bewirken), daß MS-BASIC auf die der zuletzt ausgeführten GOSUB-Anweisung unmittelbar folgende Anweisung zurückverzweigt. Eine Unterroutine kann auch mehrere RETURN-Anweisungen beinhalten. Dies ist dann nötig, wenn der logische Ablauf eine Rückverzweigung von verschiedenen Punkten der Unterroutine aus verlangt. Unterroutinen können an jeder beliebigen Stelle des Programmes plaziert werden. Es ist jedoch empfehlenswert, die Unterprogramme klar vom Hauptprogramm abzugrenzen. Um zu vermeiden, daß Unterroutinen unbeabsichtigt angelaufen werden, kann jeweils vorher eine STOP-, END- oder GOTO-Anweisung stehen. Letztere sollte den Programmfluß um die Unterprogramme herum leiten.

Beispiel:

```
10 GOSUB 40
20 PRINT "RÜCKKEHR AUS DER UNTERROUTINE"
30 END
40 PRINT "UNTERROUTINE";
50 PRINT "IN";
60 PRINT "BEARBEITUNG"
70 RETURN
RUN
UNTERROUTINE IN BEARBEITUNG
RÜCKKEHR AUS DER UNTERROUTINE
Ok
```

GOTO

Format: GOTO <zeilennummer>

Zweck: Zur bedingungslosen Verzweigung aus dem normalen Programmablauf zu einer bestimmten Programmzeile

Bemerkung: Wenn <zeilennummer> auf eine ausführbare Anweisung verweist, werden diese und die nachfolgenden ausgeführt, Ist es eine nicht ausführbare Zeile, wird der Programmablauf bei der nächsten ausführbaren Zeile fortgesetzt, die <zeilennummer> folgt.

Beispiel:

```
LIST
10 READ R
20 PRINT "R=";R,
30 A = 3.14*R^2
40 PRINT "FLÄCHE=";A
50 GOTO 10
60 DATA 5, 7, 12
Ok
RUN
R = 5          Fläche = 78.5
R = 7          Fläche = 153.86
R = 12         Fläche = 452.16
?out of data in 10
Ok
```

IF ... THEN [...ELSE] und IF ... GOTO

Format: IF <ausdruck> THEN <anweisung(en)> |
<zeilennummer> [ELSE <anweisung(en)> |
<zeilennummer>]

Format: IF <ausdruck> GOTO <zeilennummer>
[ELSE <anweisung(en)> |<zeilennummer>]

Anmerkung: der Nebensatz ELSE ... ist nur in den Versionen "Erweitert" und "Platte" erlaubt.

Zweck: Um eine Entscheidung bezüglich des Programmablaufes aufgrund des Resultates eines Ausdruckes herbeizuführen.

Bemerkung: Wenn das Ergebnis des <ausdruckes> nicht Null ist, wird die THEN- bzw. GOTO-Klausel ausgeführt. Auf THEN kann entweder eine Zeilennummer als Verzweigugungsadresse oder eine oder mehrere Anweisungen folgen, die auszuführen ist/sind. Auf GOTO muß immer eine Zeilennummer folgen. Wenn das Ergebnis des <ausdruckes> jedoch Null ist, wird der THEN- bzw. GOTO-Teil übersprungen und die ELSE-Klausel — falls vorhanden — wird ausgeführt. Das Programm wird mit der nächsten ausführbaren Anweisung fortgesetzt. (Die ELSE-Klausel ist nur in den Versionen "Erweitert" und "Platte" vorhanden). Bei den Versionen "Erweitert" und "Platte" darf vor THEN ein Komma eingefügt werden.

Schachtelung von IF-Anweisungen

In den Versionen "Erweitert" und "Platte" können IF ... THEN ... ELSE-Anweisungen geschachtelt werden. Schateln ist nur durch die maximale Zeilenlänge limitiert. Zum Beispiel:

```
IF X>Y THEN PRINT "GROESSER ALS"
ELSE IF Y>X THEN PRINT "KLEINER ALS"
ELSE PRINT "GLEICH"
```

stellt eine legale Anweisung dar. Falls eine Anweisung ungleich viele ELSE- und THEN-Klauseln enthält, wird jedes ELSE dem nächstliegenden "freien" THEN zugeordnet. Z.B.:

```
IF A=B THEN IF B=C THEN PRINT "A=C"
ELSE PRINT "A<>C"
```

ergibt nicht "A<>C" wenn A<>B ist.

Wird eine IF ... THEN-Anweisung im Direktmodus mit einer Zeilennummer nach THEN eingegeben, erscheint die Fehlermeldung "Undefined Line" (nicht definierte Zeile), falls nicht vorher eine Zeile mit der entsprechenden Nummer im indirekten Modus eingegeben wurde.

Anmerkung: Wenn IF benutzt wird, um die Gleichheit eines Wertes zu überprüfen, der das Ergebnis einer Gleitkomma-Berechnung ist, muß beachtet werden, daß die interne Darstellung etwas ungenau sein kann. Der Vergleich sollte deshalb auf den Bereich beschränkt werden, in dem die Genauigkeit des Wertes variieren kann. Um zum Beispiel eine berechnete Variable A mit dem Wert 1.0 zu vergleichen, dient:

```
IF ABS (A—1.0) < 1.0E—6 THEN ...
```

Dieser Vergleich ergibt "wahr", wenn der Wert von A gleich 1.0 mit einer relativen Abweichung kleiner 1.0E—6 ist.

Beispiel 1:

```
200 IF I THEN GET # 1,I
```

Diese Anweisung liest den Satz mit der Nummer I, wenn I nicht Null ist.

Beispiel 2:

```
100 IF (I<20)*(I>10) THEN DB = 1979—1:
    GOTO 300
110 PRINT "AUSSERHALB DES BEREICHES"
 .
 .
 .
```

In diesem Beispiel wird getestet, ob I größer 10 und kleiner 20 ist. Wenn ja, wird DB errechnet und das Programm verzweigt zur Zeile 300. Falls nicht, wird das Programm mit der Zeile 110 fortgesetzt.

Beispiel 3:

```
210 IF IOFLAG THEN PRINT A$ ELSE LPRINT A$
```

Diese Anweisung veranlaßt die Ausgabe einer Variablen auf der Anzeige oder auf dem Drucker abhängig vom Wert der Variablen IOFLAG. Wenn IOFLAG Null ist, wird auf dem Drucker, andernfalls auf der Anzeige ausgegeben.

INPUT

Format: INPUT [;] [<"bedienernachricht">;]
<liste von variablen>

Zweck: Um Eingaben über die Tastatur während der Ausführung des Programmes zu ermöglichen.

Bemerkung: Wenn eine INPUT-Anweisung angelaufen wird, wird der Programmablauf unterbrochen und ein Fragezeichen zeigt an, daß das Programm auf Daten wartet. Falls eine <"bedienernachricht"> angegeben ist, wird diese vor dem Fragezeichen angezeigt. Die entsprechenden Daten sind dann über die Tastatur einzugeben.

Statt des Strichpunktes kann ein Komma nach der Bedienernachricht angegeben werden. Dadurch wird das Fragezeichen unterdrückt. Die Anweisung INPUT "GEBE GEBURTSDATUM EIN", B$ zeigt diese Nachricht ohne Fragezeichen an.

Wird unmittelbar nach INPUT ein Strichpunkt angegeben, so bewirkt die Betätigung der Eingabetaste keine Zeilenende-/Zeilenschaltung auf der Anzeige.

Die eingegebenen Daten werden der/den Variablen in der <liste von variablen> zugeordnet. Die Anzahl der eingegebenen Datenpositionen muß der Anzahl der Variablen in dieser Liste entsprechen. Daten werden durch Kommas voneinander getrennt.

Die Variablennamen in der Liste können Namen für numerische oder String-Variablen (einschl. indizierten Variablen) sein. Der Typ jeder Datenposition muß mit dem Typ des Variablennamen übereinstimmen. (Strings müssen bei der Eingabe nicht mit Anführungszeichen versehen werden.)

Wird auf INPUT mit zu vielen, zu wenigen oder den falschen Daten (vom Typ her) geantwortet, so wird die Nachricht "?Redo from start" (Eingabe von vorne beginnen) angezeigt. Die Daten werden erst dann zugeordnet, wenn die Eingabe akzeptiert wurde.

Beispiel:

```
10 INPUT X
20 PRINT X "ZUM QUADRAT IST" X^2
30 END
RUN
?5                    (Die 5 wurde vom Bediener auf das
                      Fragezeichen hin eingegeben)
 5 ZUM QUADRAT ist 25
Ok

LIST
10 PI = 3.14
20 INPUT "GEBEN SIE DEN RADIUS EIN";R
30 A = PI*R^2
40 PRINT "DIE KREISFLAECHE ist";A
50 PRINT
60 GOTO 20
Ok
RUN
GEBEN SIE DEN RADIUS EIN? 7.4    (Eingabe:7.4)
Die KREISFLAECHE IST 171.946

GEBEN SIE DEN RADIUS EIN?
usw.
```

INPUT

Format: INPUT# <dateinummer>,<liste von variablen>

Zweck: Dient dazu, Daten von einer sequentiellen Datei auf der Platte zu lesen und Variablen im Programm zuzuordnen.

Bemerkung: <dateinummer> ist die Nummer, mit der die Datei in der OPEN-Anweisung eröffnet wurde. <liste von variablen> enthält die Variablennamen, denen die Daten der Datei zugeordnet werden. (Die Typen von Daten und Variablennamen müssen übereinstimmen). Die Anweisung INPUT# zeigt im Gegensatz zu INPUT kein Fragezeichen an.

Die Datenpositionen in der Datei müssen genauso angeordnet sein als wären sie von Hand bei einer INPUT-Anweisung eingegeben worden. Bei numerischen Werten werden führende Leerstellen,"Eingabetasten"- und "Zeilenschalt"-Symbole ignoriert. Das erste Zeichen außer den eben erwähnten wird als Beginn der Zahl angenommen. Das Ende der Zahl wird durch ein Leerzeichen, "Eingabetasten"-Symbol oder Komma bestimmt.

Wenn MS-BASIC eine sequentielle Datei nach einem alphanumerischen Datenfeld durchsucht, werden führende Leerzeichen, "Eingabetasten-" und "Zeilenschalt-"Symbole ebenfalls ignoriert. Das erste Zeichen ungleich den eben erwähnten wird als Beginn des Datenfeldes angenommen. Ist dieses Zeichen ein Anführungszeichen, besteht das alphanumerische Datenfeld aus allen Zeichen zwischen diesem und dem nächsten Anführungszeichen. Dies bedeutet, daß ein Datenfeld in Anführungszeichen innerhalb des Textes kein Anführungszeichen enthalten kann. Ist das erste Zeichen kein Anführungszeichen, so wird das String-Feld durch ein Komma, ein „Eingabetasten"-Symbol oder "Zeilenschalt"-Symbol oder nach 255 Zeichen beendet. Wird das Ende der Datei erreicht, während ein Datenfeld (numerisch oder alphanumerisch) eingelesen wird, so ist damit auch das Ende des Datenfeldes erreicht.

KILL

Format: KILL <dateiname>

Zweck: Dient dazu, eine Datei auf der Platte zu löschen.

Bemerkung: Wird mit der KILL-Anweisung eine eröffnete Datei angesprochen, so wird "File already open" (Datei bereits geöffnet) angezeigt.

KILL kann für alle Arten von Dateien, Programme, wahlfreie Dateien und sequentielle Dateien, benützt werden.

Beispiel: 200 KILL "DATEI1"

LET

Format: [LET] <variable> = <ausdruck>

Zweck: Dient dazu, Variablen den Wert eines Ausdruckes zuzuordnen.

Bemerkung: Das Wort LET kann wahlweise weggelassen werden. Das Gleichheitszeichen reicht aus, um einer Variablen einen Wert zuzuordnen.

Beispiel:

```
110 LET D=12
120 LET E=12^2
130 LET F=12^4
140 LET SUM=D+E+F
      .
      .
      .
```

oder

```
110 D=12
120 E=12^2
130 F=12^4
140 SUM=D+E+F
      .
      .
      .
```

LINE INPUT

Format: LINE INPUT [;][<"bedienernachricht">;]
<string-variable>

Zweck: Zur Eingabe einer ganzen Zeile (bis zu 254 Zeichen) in eine String-Variable ohne Begrenzungssymbole

Bemerkung: <"bedienernachricht"> ist eine alphanumerische Meldung, die an der Anzeige erscheint, bevor die Eingabe erwartet wird. Ein Fragezeichen wird nur dann angezeigt, wenn es Bestandteil der Bedienernachricht ist. Die gesamte Eingabe bis zur Eingabetaste wird der <string-variablen> zugeordnet.

Wird unmittelbar nach LINE INPUT ein Strichpunkt angegeben, so bewirkt die Betätigung der Eingabetaste keine Zeilenende-/Zeilenschaltung auf der Anzeige.

Eine LINE INPUT-Anweisung kann durch Control-C abgebrochen werden. MS-BASIC kehrt dann in den Direktmodus zurück und zeigt Ok an. Wenn man CONT eingibt, wird das Programm mit dieser LINE INPUT-Anweisung fortgesetzt.

Beispiel: Siehe Beispiel unter LINE INPUT#

LINE INPUT#

Format: LINE INPUT# <dateinummer> , <string-variable>

Zweck: Dient dazu, eine gesamte Zeile (bis zu 254 Zeichen) ohne Begrenzungssymbole von einer sequentiellen Plattendatei in eine alphanumerische Variable zu lesen.

Bemerkung: <dateinummer> ist die Nummer, unter der die Datei eröffnet wurde. <string-variable> ist der Name der Variablen, der die Zeile zugeordnet wird. LINE INPUT# liest alle Zeichen einer sequentiellen Datei bis zu einem „Eingabetaste"-Symbol. Diese Symbolsequenz wird dann übersprungen und die nächste LINE INPUT#-Anweisung liest alle Zeichen bis zum nächsten Symbol "Eingabetaste". (Wird eine Sequenz von "Zeilenschalt"-/"Eingabetaste"-Symbolen entdeckt, so wird sie konserviert).

LINE INPUT# ist besonders sinnvoll, wenn jede Zeile einer Datei in Felder unterteilt wurde oder wenn ein MS-BASIC Programm in ASCII-Modus ausgeschrieben wurde und von einem anderen Programm als Datei benutzt wird.

Beispiel:

```
10 OPEN "0",1,"LISTE"
20 LINE INPUT "KUNDENDATEN?";C$
30 PRINT# 1,C$
40 CLOSE 1
50 OPEN "I",1,"LISTE"
60 LINE INPUT# 1,C$
70 PRINT C$
80 CLOSE 1
RUN
KUNDENDATEN? LINDA MEIER 234.4
HAMBURG | LINDA MEIER 234.4 HAMBURG
Ok
```

LIST

Format 1: LIST [<zeilennummer>]

Format 2: LIST [<zeilennummer>[-[<zeilennummer]]]

Zweck: Dient dazu, ein Programm, das sich im Speicher befindet, oder Teile davon, auf der Anzeige zu listen.

Bemerkung: MS-BASIC kehrt nach der Ausführung von LIST immer in den Direktmodus zurück.

Format 1: Wenn <zeilennummer> nicht angegeben ist, beginnt die Auflistung mit der niedrigsten Zeilennummer. (Die Auflistung endet beim Programmende oder durch Eingabe von Control-C). Ist <zeilennummer> angegeben, so beginnt die Auflistung bei der 8K-Version mit dieser Zeile. Bei den Versionen "Erweitert" und "Platte" wird nur diese eine spezifizierte Zeile gelistet.

Format 2: Dieses Format erlaubt folgende Möglichkeiten:

- Ist nur die erste Zeilennummer angegeben, so werden diese Zeile und alle nachfolgenden Zeilen gelistet.
- Ist nur die zweite Zeilennummer angegeben, so werden alle Zeilen vom Programmanfang bis einschließlich dieser Zeile gelistet.
- Sind beide Nummern angegeben, so wird der gesamte spezifizierte Bereich gelistet.

Beispiel: Format 1:

LIST	listet das gesamte Programm, das momentan im Speicher ist.
LIST 500	listet in der 8K-Version alle Zeilen von Zeile 500 bis zum Programmende. Listet Zeile 500 in "Erweitert" und "Platte".

Format 2:

LIST 150—	Listet alle Zeilen von Zeile 150 bis zum Programmende.
LIST —1000	Listet alle Zeilen vom Programmanfang bis einschließlich Zeile 1000.
LIST 150—1000	Listet alle Zeilen von 150—1000 einschließlich der beiden genannten Zeilen.

LLIST

Format: LLIST [<zeilennummer>[−[<zeilennummer]]]

Zweck: Dient dazu, ein Programm, das sich im Speicher befindet, oder Teile davon, auf dem Systemdrucker zu listen.

Bemerkung: LLIST nimmt eine Druckbreite von 132 Zeichen pro Zeile an.

MS-BASIC kehrt immer nach Ausführung der Anweisung LLIST in den Direktmodus zurück. LLIST erlaubt die gleichen Möglichkeiten wie Format 2 der LIST-Anweisung.

Anmerkung: LLIST und LPRINT sind nicht in allen Implementationen von MS-BASIC enthalten.

Beispiel: Siehe Beispiele von LIST, Format 2.

LOAD

Format: LOAD <dateiname>[,R]

Zweck: Dienst dazu, eine Programmdatei von der Platte in den Speicher zu laden.

Bemerkung: <dateiname> ist der Name, mit dem das Programm in der SAVE-Anweisung auf die Platte geschrieben wurde. (Unter CP/M wird dabei, falls nicht anders angegeben, der Anhang .BAS dazugesetzt).

Nach dem Laden der R-Option läuft das Programm automatisch ab.

LOAD schließt alle geöffneten Dateien und löscht alle VAriablen sowie alle im Speicher befindlichen Programmzeilen, bevor es das angegebene Programm lädt. Ist jedoch der Wahlzusatz "R" angegeben, wird das Programm sofort nach dem Laden ausgeführt (RUN— und alle offenen Dateien bleiben geöffnet. LOAD mit "R" kann somit benutzt werden, um mehrere Programme (oder Segmente eines Programmes) zu verketten. Informationen werden dabei mittels Dateien von einem Programm ans andere übergeben.

Beispiel:
```
LOAD "STRTRK",R
LOAD "B;MYPROG"
```

LPRINT und LPRINT USING

Format: LPRINT [<liste von ausdrücken>]

LPRINT USING <string-ausdruck>;
<liste von ausdrücken>

Zweck: Dient zur Ausgabe von Daten über den Drucker

Bemerkung: Diese Anweisung ist gleich der Anweisung PRINT und PRINT USING mit der Ausnahme, daß die Ausgabe über den Systemdrucker erfolgt.

LPRINT nimmt an, daß die Druckbreite 132 Zeichen beträgt.

Anmerkung: LPRINT und LLIST sind nicht in allen Implementationen von MS-BASIC enthalten.

LSET und RSET

Format: LSET <string-variable> = <string-ausdruck>
RSET <string-variable> = <string-ausdruck>

Zweck: Dient zur Übertragung von Daten aus dem Speicher in den Puffer für Dateien mit wahlfreiem Zugriff (Vorbereitung für eine PUT-Anweisung).

Bemerkung: Hat der <string-ausdruck> weniger Stellen, als in der FIELD-Anweisung für die <string-variable> definiert, so wird der Text durch LSET linksbündig und durch RSET rechtsbündig gepuffert. (Die restlichen Stellen werden mit Leerzeichen aufgefüllt). Ist der Text länger als das Feld, so werden die Zeichen rechts abgeschnitten. Numerische Werte müssen in Strings umgewandelt werden, bevor sie mit LSET oder RSET übertragen werden. Siehe hierzu die Funktionen MKI$, MKS$ und MKD$, in Kapitel 3.

Beispiel:
```
150 LSET A$= MKS$ (AMT)
160 LSET D$= DESC ($)
```

Anmerkung: LSET und RSET können auch für nicht durch FIELD definierte String-Variable benutzt werden, um den Text links- bzw. rechtsbündig anzuordnen. Die Programmzeilen

```
110 A$= SPACE$(20)
120 RSET A$= N$
```

setzen den Text aus N$ rechtsbündig im 20-Zeichen-Feld A$. Diese Leistung kann bei der Formattierung von Druckzeilen sehr nützlich sein.

MERGE

Format: MERGE <programmname>

Zweck: Wird gebraucht, um eine bestimmte Programmdatei von der Platte mit dem zur Zeit im Speicher befindlichen Programm zu mischen.

Bemerkung: <programmname> ist der Name, der dem Programm in der SAVE-Anweisung gegeben wurde. (Mit CP/M wird dabei, falls nicht anders angegeben, der Anhang .BAS dazugesetzt). Die Datei muß vorher im ASCII-Format abgespeichert worden sein (SAVE,A). Andernfalls wird "Bad file mode" (Falscher Dateimodus) angezeigt.

Falls irgendwelche Zeilennummern identisch sind, werden die Zeilen im Speicher von den entsprechenden Zeilen von der Platte überschrieben. (MERGE kann als "Einsetzen" der Programmzeilen von der Platte in den Speicher betrachtet werden.)

MS-BASIC kehrt nach der Ausführung des Befehles MERGE immer in den Direktmodus zurück.

Beispiel: MERGE "NUMMERN"

MID$

Format: MID$ (<string-ausdr.1>,n[,m])=<string-ausdr.2>, wobei n und m ganzzahlige Ausdrücke und <string-ausdr. 1> sowie <string-ausdr.2> Text-Ausdrücke sind.

Zweck: Dient dazu, Teile eines Textes durch einen anderen Text zu ersetzen.

Bemerkung: Die Zeichen in <string-ausdr. 1>, beginnend mit dem Zeichen an Position n werden durch die Zeichen von <string-ausdr.2> ersetzt. Das wahlweise anzugebende m gibt an, wieviele Zeichen von <string-ausdr.2> maximal benutzt werden dürfen. Wird m nicht angegeben, so steht der ganze Text von <string-ausdr.2> zur Verfügung. In keinem Falle werden jedoch mehr Zeichen ersetzt, als <string-ausdr.1> ursprünglich hatte.

Beispiel:
```
10 A$ = "KANSAS CITY,MO"
20 MID$(A$,14) = "KS"
30 PRINT A$
RUN
KANSAS CITY, KS
```

MID$ wird auch als Funktion benutzt, die einen Teil eines Textes extrahiert.

NAME

Format: NAME <alter dateiname> AS <neuer dateiname>

Zweck: Wird zum Ändern eines Dateinamen auf der Platte benützt.

Bemerkung: <alter dateiname> muß vorhanden sein, <neuer dateiname> darf noch nicht vorhanden sein; widrigenfalls ein Fehler entsteht. Nach einer NAME-Anweisung ist die Datei auf der gleichen Platte am gleichen Platz unter neuem Namen abgespeichert.

Beispiel:
```
Ok
NAME "KUNDEN" AS "KUNDEN-1"
Ok
```

In diesem Beispiel wurde der Datei KUNDEN der neue Name KUNDEN-1 gegeben.

NEW

Format: NEW

Zweck: Löscht das im Speicher befindliche Programm, sowie sämtliche Variablen.

Bemerkung: NEW wird im Direktmodus verwendet, bevor ein neues Programm eingegeben wird. MS-BASIC kehrt nach Ausführung der Anweisung NEW immer in den Direktmodus zurück.

Beispiel: NEW

NULL

Format: NULL < ganzzahliger ausdruck >

Zweck: Festsetzen der Anzahl Nullen, die am Ende einer jeden Zeile ausgegeben werden.

Bemerkung: Der < ganzzahlige ausdruck > sollte sein bei:

Lochstreifenstanzern mit 10 Zeilen/Sekunde	> = 3
Fernschreibern	0 oder 1
fernschreiberverträglichen Bildschirmen	0 oder 1
Druckern (30 Zeichen/sek.)	2 oder 3.

Ohne Angabe wird 0 eingesetzt.

Beispiel:

```
Ok
NULL 2
Ok
100 INPUT X
200 IF X< 50 GOTO 800
 .
 .
 .
```

Zwei Null-Zeichen werden nach jeder Zeile ausgegeben.

ON ERROR GOTO

Format: ON ERROR GOTO <zeilennummer>

Zweck: Zur Fehlersuche und Festlegung der ersten Zeile der Fehlerroutine.

Bemerkung: Sobald eine ON ERROR GOTO-Anweisung die Verzweigung bei Fehlern aktiviert hat, bewirken alle Fehler einschließlich jener im Direktmodus (z.B. Satzaufbaufehler) eine Programmverzweigung zur Fehlerroutine. Wenn <zeilennummer> nicht existiert, wird der Fehler "Undefined Line" (nicht definierte Zeile) angezeigt. Um die Verzweigung zu deaktivieren, ist die Anweisung ON ERROR GOTO 0 auszuführen. Anschließend auftretende Fehler zeigen eine Fehlernachricht an und bewirken eine Programmunterbrechung. Eine ON ERROR GOTO 0-Anweisung in einer Fehlerroutine veranlaßt MS-BASIC zur Unterbrechung und Anzeige der Fehlermeldung des verursachenden Fehlers. Es wird empfohlen, in allen Fehlerroutinen diese Anweisung bei Fehlerfällen, die nicht vom Programm abgehandelt werden, eine Anweisung ON ERROR GOTO 0 auszuführen.

Anmerkung: Wird ein Fehler innerhalb der Fehlerroutine erzeugt, so wird die BASIC-Fehlermeldung angezeigt und die Verarbeitung abgebrochen. Eine Verzweigung innerhalb der Fehlerroutine wird im Falle eines dort auftretenden Fehlers nicht ausgeführt.

Beispiel: 10 ON ERROR GOTO 1000

ON ... GOSUB und ON ... GOTO

Format: ON <ausdruck> GOSUB <liste von zeilennummern>
ON <ausdruck> GOTO <liste von zeilennummern>

Zweck: Zur Verzweigung auf eine von mehreren angegebenen Zeilennummern abhängig vom Wert eines Ausdruckes.

Bemerkung: Der Wert von <ausdruck> bestimmt, zu welcher Zeilennummer aus der Liste verzweigt wird. Ist der Wert z.B. drei, so wird die dritte Zeilennummern aus der Liste zur Verzweigungsadresse. (Ergibt der Ausdruck einen Dezimalbruch, so werden die Dezimalstellen gerundet).

In der ON ... GOSUB-Anweisung gibt jede Zeilennummer in der Liste den Anfang einer Unterroutine an.

Er gibt <ausdruck> den Wert Null oder einen Wert größer als die Anzahl der Zeilennummern in der Liste (jedoch kleiner als 256), setzt BASIC das Programm mit der nächstfolgenden ausführbaren Anweisung fort. Ergibt sich aus <ausdruck> jedoch ein negativer Wert oder ein Wert über 255, wird "Illegal function call" (unerlaubte Funktion) angezeigt.

Beispiel: 100 ON L GOTO 150, 300, 320, 390.

OPEN

Format: OPEN <modus>,[#]<dateinummern>,
<dateiname>,[<satzlänge>]

Zweck: Eröffnen einer Plattendatei für Ein-/Ausgabe.

Bemerkung: Eine Plattendatei muß mit OPEN eröffnet werden, bevor irgenwelche Ein-/Ausgabe-Operationen auf diese Datei vorgenommen werden können. OPEN stellt den Pufferbereich für diese Datei bereit und bestimmt den Zugriffsmodus dafür.

<modus> ist ein String-Ausdruck, dessen erster Buchstabe einer der nachfolgenden sein kann.

O spezifiziert den Ausgabe-Modus (Output)
I spezifiziert den Eingabe-Modus (Input)
R spezifiziert Ein- und Ausgabe mit wahlfreiem (random) Zugriff.

<dateinummer> ist ein ganzzahliger Ausdruck zwischen 1 und 15. Diese Nummer wird der betreffenden Datei zugeordnet und bleibt gültig, solange diese eröffnet bleibt. Diese Nummer benutzen andere Plattenein-/Ausgabe-Anweisungen als Bezug auf diese Datei.

<dateiname> ist ein String-Ausdruck, der einen Dateinamen entsprechend den Regeln des benutzten Betriebssystems beinhaltet.

<satzlänge> ist ein ganzzahliger Ausdruck, der, sofern angegegen, die Satzlänge für Dateien mit wahlfreiem Zugriff definiert. Falls nicht angegeben, wird 128 angenommen.

Anmerkung: Eine Datei kann für sequentielle Eingabe oder wahlfreien Zugriff gleichzeitig unter mehreren Dateinummern eröffnet sein. Zur Ausgabe kann sie jedoch nur unter einer Nummer zu einer bestimmten Zeit eröffnet sein.

Beispiel: 10 OPEN "I", 2,"INVEN"

OPTION BASE

Format: OPTION BASE n
wobei n 1 oder 0 ist.

Zweck: Dient zur Festlegung des Minimalwertes für Elemente von indizierten Variablen.

Bemerkung: Automatisch wird als Minimalwert 0 eingesetzt. Wird jedoch die Anweisung

OPTION BASE 1

ausgeführt, so wird der niedrigste Wert jeden Elementes auf 1 festgelegt.

OUT

Format: OUT I,J
wobei I und J ganzzahlige Ausdrücke zwischen 0 und 256 sind.

Zweck: Dient dazu, ein Byte (Stelle) an einen Ausgabekanal zu senden.

Bemerkung: Der ganzzahlige Ausdruck I bezeichnet den Ausgabekanal, J ist der auszugebende Wert.

Beispiel: 100 OUT 32, 100

POKE

Format: POKE I,J
wobei I und J ganzzahlige Ausdrücke sind.

Zweck: Wird dazu benutzt, eine Speicherstelle (Byte) zu beschreiben.

Bemerkung: Der Ausdruck I gibt die Adresse der zu beschreibenden Speicherstelle an. Der Ausdruck J ist der Wert, der in diese Speicherstelle geschrieben werden soll. J muß im Bereich von 0 bis 255 sein. In der 8K-Version muß I kleiner als 32768 sein. In den Versionen "Erweitert" und "Platte" kann I im Bereich von 0 bis 65536 sein.

In der 8K-Version können die Speicherstellen mit Adressen über 32768 beschrieben werden, indem I einen negativen Wert erhält. Der Wert wird ermittelt, indem man den Wert 65536 von der gewünschten Speicheradresse subtrahiert; z.B. um Daten in die Adresse 45000 zu schreiben: I = 45000 — 65536 = = -- 20536.

Die Umkehrfunktion von POKE ist PEEK. Das Argument von PEEK ist eine Speicheradresse, deren Inhalt gelesen werden soll. Siehe Seite 3-17.

POKE und PEEK werden vor allem eingesetzt für effiziente Datenspeicherung, Laden von Unterprogrammen in Assembler-Sprache, und Übermitteln von Argumenten von und zu Assembler-Unterprogrammen.

Beispiel: 10 POKE &H5A00, &HFF

PRINT

Format: PRINT [<liste von ausdrücken>]

Zweck: Zur Anzeige von Daten auf dem Bildschirm.

Bemerkung: Wird <liste von ausdrücken> weggelassen, so wird eine Leerzeile angezeigt. Andernfalls werden die Werte der Ausdrücke angezeigt. Die Ausdrücke in der Liste können numerisch und/oder alpanumerisch sein. (Letztere müssen in Anführungszeichen stehen).

Anzeigepositionen

Die Position jedes angezeigten Ausdruckes wird durch das Trennsymbol in der Liste der Ausdrücke bestimmt. MS-BASIC unterteilt eine Bildschirmzeile in Zonen zu je 14 Positionen. Ein Komma als Trennsymbol bewirkt, daß der nächste Wert am Beginn der nächsten Zone angezeigt wird. Ein Strichpunkt bewirkt, daß der nächste Wert unmittelbar am Ende des vorherigen angezeigt wird. Ein (oder mehrere) Leerzeichen zwischen den Ausdrücken wirken wie der Strichpunkt.

Ist ein Komma oder Strichpunkt am Ende der Liste nach dem letzten Ausdruck, so wird die Position für die nächste PRINT-Anweisung entsprechend diesen Regeln definiert. Steht kein Komma oder Strichpunkt am Ende der Liste, so wird das Symbol <Eingabetaste> ausgegeben. Ist die anzuzeigende Zeile länger als eine Bildschirmzeile, so bewirkt MS-BASIC, daß die Anzeige auf der nächsten Bildschirmzeile fortgesetzt wird.

Nach numerischen Werten erscheint immer ein Leerzeichen. Positive Zahlen haben vorneweg ein Leerzeichen, negative ein Minuszeichen. MS-BASIC versucht immer, Zahlen in Normalform anzuzeigen. Ist jedoch die Exponentialdarstellung genauer, so wird diese Form gewählt:

Beispiele:

Einfache Genauigkeit	Darstellungsform
10^(−6)	.000001
10^(−7)	1E−7

Doppelte Genauigkeit	Darstellungsform
10^(−16)	.0000000000000001
10^(−17)	1D−17

Statt des Wortes PRINT kann in einer PRINT-Anweisung auch ein Fragezeichen gesetzt werden.

Beispiel 1:

```
10 X = 5
20 PRINT X+5,X−5,X*(−5),X^5
30 END
RUN
10        0        −25        3125
Ok
```

In diesem Beispiel bewirken die Kommas, daß jeder Wert am Anfang der nächsten Zone angezeigt wird.

Beispiel 2:

```
LIST
10 INPUT X
20 PRINT X "ZUM QUADRAT IST" X^2 UND";
30 PRINT X "HOCH DREI IST" X^3
40 PRINT
50 GOTO 10
Ok
RUN
?9
 9 ZUM QUADRAT IST 81 und 9 HOCH DREI IST
 729

?21
 21 ZUM QUADRAT IST 441 und 21 HOCH DREI
 IST 9261

 ?
```

Hier bewirkt der Strichpunkt am Ende der Zeile 20, daß die Anzeige von Zeile 30 auf derselben Bildschirmzeile erfolgt und Zeile 40 bewirkt eine Leerzeile auf dem Bildschirm.

Beispiel 3:

```
10 FOR X = 1 TO 5
20 J = J + 5
30 K = K + 10
40 ? J;K;
50 NEXT X
Ok
RUN
 5   10   10   20   15   30   20   40   25   50
Ok
```

In diesem Beispiel bewirken die Strichpunkte in der PRINT-Anweisung, daß die Zahlen unmittelbar aneinander gereiht werden. (Dabei folgt auf jede Zahl ein Leerzeichen und jede positive Zahl hat vorneweg ein Leerzeichen). In der Zeile 40 wird ein Fragezeichen für PRINT verwendet.

PRINT USING

Format: PRINT USING <string-ausdruck>;
<liste von ausdrücken>

Zweck: Um Texte oder Zahlen in einem bestimmten Format anzuzeigen.

Bemerkung und Beispiel: <liste von ausdrücken> setzt sich aus den Texten und/oder Zahlen zusammen, die anzuzeigen sind. Sie werden durch Strichpunkte getrennt. <string-ausdruck> ist ein Text (oder eine Variable, bestehend aus Formatierungszeichen. Diese Zeichen (siehe unten) bestimmen Platzbedarf und Format der anzuzeigenden Texte und/oder Zahlen.

Textfelder

Wird PRINT USING benutzt, um String (Texte) anzuzeigen, kann eines von drei Formatierungszeichen angewandt werden:

"!" spezifiziert, daß nur das erste Zeichen des Textes angezeigt wird.

"\Leerzeichen\" bedeutet, daß 2+n Zeichen angezeigt werden. Werden die Schrägstriche-links ohne Leerzeichen eingegeben, so werden zwei Zeichen des Textes angezeigt, mit einem Leerzeichen werden drei Zeichen angezeigt; usw. Ist der Text länger als dieses Feld, so werden die verbleibenden Zeichen einfach ignoriert. Ist das Feld länger als der Text, so wird der Text linksbündig angezeigt und die restlichen Stellen rechts werden mit Leerzeichen gefüllt.

Beispiel:

```
10 A$ = "SIEH": B$ = "HER"
30 PRINT USING "!";A$;B$
40 PRINT USING "\  \";A$;B$
50 PRINT USING "\   \";A$,B$;"! !"
RUN
SH
SIEHHER
SIEH HER!!
Ok
```

"&" spezifiziert ein Textfeld mit variabler Länge. Wenn das Feld mit "&" definiert wird, wird der Text so angezeigt, wie er eingegeben wurde.

Beispiel:

```
10 A$ = "SIEH";B$ = "HER"
20 PRINT USING "!";A$;
30 PRINT USING "&";B$
RUN
SHER
Ok
```

Numerische Felder

Wird PRINT USING zur Anzeige von Zahlen benutzt, können folgende Spezialzeichen zur Formatierung des numerischen Feldes angewandt werden.

\# Ein Nummernzeichen steht für jede Ziffernposition. Zifferpositionen werden immer aufgefüllt. Hat die anzuzeigende Zahl weniger Ziffern als Positionen im Feld vorgesehen sind, so wird die Zahl rechtsbündig (mit Leerzeichen lins) im Feld angezeigt.

Ein Dezimalpunkt kann an jeder beliebigen Position im Feld eingesetzt werden. Wird ein Nummernzeichen links vom Dezimalpunkt gesetzt, so wird diese Stelle immer angezeigt (im Bedarfsfall als 0). Zahlen werden bei Bedarf gerundet.

```
PRINT USING "# #.# #";.78
 0.78

PRINT USING "# # #.# #";987.654
 987.65

PRINT USING "##.##"; 10.2,5.3,66,789,.234
 10.20  5.30  66.79  0.23
```

Im letzten Beispiel sind die Werte jeweils durch drei Leerzeichen getrennt.

\+ Ein Pluszeichen vor oder hinter dem Formatfeld bestimmt, daß immer und wo das Vorzeichen der Zahl angezeigt wird. (Plus oder Minus, vor oder hinter der Zahl).

— Ein Minuszeichen hinter dem Formatfeld ergibt, daß bei negativen Zahlen hinter der letzten Ziffer ein Minuszeichen angezeigt wird.

```
PRINT USING "+# # # . # #"; —68.95, 2.4, 55.6, —.9
—68.95  +2.40 +55.60 —0.90
```

```
PRINT USING "# # . # # — ";—68.95, 22.499, —7.01
68.95—  22.45 7.01—
```

** Zwei Sterne am Anfang des Formatstrings bewirken, daß freie Stellen vor der Zahl mit Sternen gefüllt werden. Die zwei Sterne stehen außerdem für zwei weitere Ziffernpositionen.

```
PRINT USING "**# # "; 12.39, —0.9, 765.1
*12.4 *—0.9 765.1
```

$$ Zwei Dollarzeichen an Anfang des Formatstrings bewirken, daß unmittelbar links von der Zahl ein Dollarzeichen ausgegeben wird. Die zwei Zeichen spezifizieren zwei zusätzliche Ziffernpositionen, von denen eine als Dollarzeichen angezeigt wird. Diese Funktion kann nicht bei der Exponentialdarstellung benutzt werden. Negative Zahlen können nicht angezeigt werden, es sei denn, das Formatfeld hat am Ende ein Minuszeichen.

```
PRINT USING "$$# # # .# # "; 456.78
$456.78
```

**$ Steht **$ am Anfang des Formatfeldes, so bewirkt dies eine Kombination der beiden oben genannten Funktionen. Führende Leerstellen werden mit Sternen gefüllt und unmittelbar vor der Zahl steht ein Dollarzeichen. Durch **$ werden drei zusätzliche Ziffernpositionen definiert, von denen eine als Dollarzeichen angezeigt wird.

```
PRINT USING "**# # # .# # "; 2.34
***$2.34
```

, Ein Komma links vom Dezimalpunkt im Formatstring bewirkt, daß ein Komma links von jeder dritten Ziffer links vom Dezimalpunkt angezeigt wird. Ein Komma am Ende des Formatstrings wird als Teil dieses Strings mit angezeigt. Das Komma spezifiziert eine Ziffernposition. Das Komma hat bei der Exponentialdarstellung (^^^^) keine Auswirkung.

```
PRINT USING "####,.##"; 1234.5
1,234.50

PRINT USING "####.##,";1234.5
1234.50,
```

^^^^ Vier Pfeile nach oben hinter den Nummernzeichen spezifizieren die Darstellung im Exponentialformat. Die vier Pfeile reservieren den Platz für die Anzeige von "E+xx". Der Dezimalpunkt kann an beliebiger Stelle stehen. Die signifikanten Stellen werden linksbündig angeordnet und der Exponent wird angeglichen. Falls das Vorzeichen nicht angegeben wird, (durch vor- oder hintenangestelltes + oder —), wird eine Position links vom Dezimalpunkt für eine Leerstelle oder das Minuszeichen benutzt.

```
PRINT USING "##.##^^^^". 234.56
 2.35 E+02

PRINT USING ".####^^^^—":888888
 .8889E+06

PRINT USING "+.##^^^^":123
 +.12E+03
```

— Ein Unterstreichungsstrich im Formatstring bedeutet, daß das nächste Zeichen anzuzeigen ist.

```
PRINT USING "_!##.##_!";12.34
 !12.34!
```

Soll ein Unterstreichungsstrich angezeigt werden, so ist er in Anführungsstriche zu setzen. ("_").

% Hat die anzuzeigende Zahl mehr Ziffern als das definierte Feld zuläßt, so wird vor der Zahl ein Prozentzeichen angezeigt. Wird die Feldgröße durch Rundung überschritten, so erscheint das Prozentzeichen vor der gerundeten Zahl.

```
PRINT USING "##.##";111.22
% 111.22

PRINT USING ".##";.999
%1.00
```

PRINT# UND PRINT# USING

Format: PRINT#<dateinummer>,
[USING<string-ausdruck>;]<liste von ausdrücken>

Zweck: Um Daten auf eine sequentielle Plattendatei zu schreiben.

Bemerkung: <dateinummer> ist die Nummer, unter der die Datei in der OPEN-Anweisung eröffnet wurde. <string-ausdruck> setzt sich aus Formatierungszeichen wie unter PRINT USING (Seite 2-64) beschreiben, zusammen. Die <liste von ausdrücken> beinhaltet numerische und/oder Textausdrücke, die auf die Datei geschrieben werden sollen.

PRINT# komprimiert die Daten auf der Platte nicht. Sie werden genauso auf die Platte geschrieben, wie sie am Bildschirm mit der PRINT-Anweisung angezeigt würden. Aus diesem Grunde ist es ratsam, auf die richtigen Trennsymbole in der Liste zu achten, um die Daten später wieder richtig einlesen zu können.

In der Liste von Ausdrücken sollten numerische Werte durch Strichpunkte getrennt werden. Z.B.:

```
PRINT#1,A;B;C;X;Y;Z
```

(Werden Kommas als Trennsymbole benutzt, werden auf die Platte auch die zusätzlichen Leerzeichen zwischen den Feldern ausgegeben.)

Textausdrücke müssen durch Strichpunkte in der Liste getrennt werden. Um Textausdrücke korrekt auf der Platte zu formatieren, sollten zusätzliche Trennsymbole vorgegeben werden:

Als Beispiel sei A$= "KAMERA" und B$= "93604-1". Die Anweisung

```
PRINT#1.A$;B$
```

schreibt KAMERA93604-1 auf die Platte. Da kein Trennsymbol vorhanden ist, kann diese Information nicht in zwei Variablen eingelesen werden. Das Problem kann folgendermaßen behoben werden:

```
PRINT #1,A$;",";B$
```

Dadurch wird KAMERA, 93604-1 auf die Platte geschrieben. Diese Information kann beim Einlesen zwei Text-Variablen zugeordnet werden.

Wenn die Texte selbst Zeichen wie Kommas, Strichpunkte, führende Leerzeichen, Eingabetaste oder Zeilenschaltung enthalten, sind sie in Anführungszeichen gesetzt auf die Platte zu schreiben. Dazu benutzt man CHR$(34).

Z.B. bei A$ = "KAMERA, AUTOMATIK" und B$ = "93604-1". Die Anweisung

```
PRINT #1,A$;B$
```

würde KAMERA, AUTOMATIK 93604-1 ausgegeben und die Anweisung

```
INPUT #1,A$,B$
```

würde "KAMERA" der Variablen A$ und "AUTOMATIK 93604-1" der Variablen B$ zuordnen. Um die beiden Textfelder korrekt zu trennen, sind sie durch CHR$(34) mit Anführungsstrichen zu versehen. Die Anweisung

```
PRINT #1,CHR$(34);A$;CHR$(34);CHR$(34):
B$;CHR$(34)
```

schreibt "KAMERA, AUTOMATIK" "93604-1" auf die Platte und die Anweisung

```
INPUT #1,A$,B$
```

ordnet A$ den Text "KAMERA, AUTOMATIK" und B$ den Text "93604-1" zu.

Die Anweisung PRINT # kann wahlweise mit der Klausel USING benutzt werden, um das Format der Daten auf der Platte zu bestimmen. Zum Beispiel:

```
PRINT #1,USING "$$###.## ";J;K;L
```

Siehe auch WRITE# , Kapitel 2.

PUT

Format: PUT [#]<dateinummer>[,<satznummer>]

Zweck: Um einen Satz aus dem Pufferbereich auf eine Plattendatei mit wahlfreiem Zugriff zu schreiben.

Bemerkung: <dateinummer> ist die Nummer, unter der die Datei in der OPEN-Anweisung eröffnet wurde. Wird <satznummer> nicht angegeben, wird der Satz die nächstfolgende Satznummer bekommen (nach der letzten PUT-Anweisung). Die Satznummern können von 1 bis 32767 gehen.

Anmerkung: Der Pufferbereich für Dateien mit wahlfreiem Zugriff kann durch die Anweisungen PRINT#, PRINT# USING und WRITE# beschrieben werden, bevor eine PUT-Anweisung erfolgt.

Im Falle von WRITE# füllt MS-BASIC den Puffer mit Leerzeichen bis zum Symbol <Eingabetaste> auf. Jeder Versuch, über den Pufferbereich hinaus zu lesen oder zu schreiben, ergibt die Fehlermeldung "Field Overflow" (Feld-Überlauf).

RANDOMIZE

Format: RANDOMIZE[<ausdruck>]

Zweck: Setzt den Zufallsgenerator zurück.

Bemerkung: Wenn kein <ausdruck> angegeben ist, wird der Programmablauf von MS-BASIC unterbrochen und nach einem Ausgangswert durch die Anzeige

Random Number Seed (−32768 to 32767)?

gefragt, bevor RANDOMIZE ausgeführt wird.

Wird der Zufallsgenerator nicht zurückgesetzt, so ergibt die RND-Funktion die gleiche Folge von Zufallszahlen, sooft das Programm aufgerufen wird. Um dies zu vermeiden, sollte eine RANDOMIZE-Anweisung am Programmanfang stehen und bei jedem Aufruf mit einem anderen Argument beantwortet werden.

Beispiel:

```
10 RANDOMIZE
20 FOR I = 1 TO 5
30 PRINT RND;
40 NEXT I
RUN
Random Number Seed (−32768 to 32767)? 3
(Eingabe: 3)
 .88598  .484668  .586328  .119426  .709225
Ok
RUN
Random Number Seed (−32768 to 32767)?4
(Eingabe: 4, um eine neue Nummernfolge zu erhalten)
 .803506  .162462  .929364  .292443  .322921
Ok
RUN
Random Number Seed (−32768 to 32767)?3
(ergibt dieselbe Nummernfolge wie im ersten Durchlauf)
 .88598  .484668  .586328  .119426  .709225
Ok
```

READ

Format: READ <liste von variablen>

Zweck: Dient zum Lesen von Werten in einer DATA-Anweisung und Zuordnung an Variable.

Bemerkung: Eine READ-Anweisung muß immer im Zusammenhang mit einer DATA-Anweisung verwendet werden. READ ordnet den Werten in der DATA-Anweisung Variable auf einer eins-zu-eins Basis zu. Die Variablen in einer READ-Anweisung können numerisch oder alphanumerisch sein. Die gelesenen Werte müssen im Typ mit der jeweiligen Variablen übereinstimmen. Andernfalls wird ein "Syntax error" (Satzaufbaufehler) angezeigt.

Eine einzige READ-Anweisung kann auf eine oder mehrere DATA-Anweisungen zugreifen (entsprechend deren Reihenfolge), oder mehrere READ-Anweisungen können auf eine DATA-Anweisung zugreifen. Ist die <liste von variablen> größer als die Anzahl von Elementen in der/den DATA-Anweisung(en), so wird der Fehler "OUT OF DATA" (Ende der Daten) angezeigt. Ist die Anzahl der angegebenen Variablen kleiner als die Elemente der DATA-Anweisung(en), so greift die folgende READ-Anweisung auf das erste noch nicht gelesene Element zu. Folgt keine READ-Anweisung, dann werden die übrigen Daten ignoriert.

Um DATA-Anweisungen wieder von vorne zu lesen, ist die RESTORE-Anweisung zu nehmen. (Siehe RESTORE, Seite 2-78).

Beispiel 1:

```
 .
 .
 .
 80 FOR I = 1 TO 10
 90 READ A(I)
100 NEXT I
110 DATA 3.08,5.19,3.12,3.98,4.24
120 DATA 5.08,5.55,4.00,3.16,3.37
 .
 .
 .
```

Dieser Programmteil liest die Daten aus den DATA-Anweisungen und ordnet sie der Tabelle A zu. Anschließend ist der Wert von A(1) = 3.08, A(2) = 5.19, usw.

Beispiel 2:

```
LIST
10 PRINT "PLZ", "STADT", "VORWAHL"
20 READ  C,S$,Z$
30 DATA 8000, "MUENCHEN 2" , 089
40 PRINT C, S$, Z$
Ok
RUN
PLZ           STADT          VORWAHL
8000          MUENCHEN 2     089
Ok
```

Dieses Programm liest numerische und alphanumerische Daten von der DATA-Anweisung in Zeile 30.

REM

Format: REM < bemerkung>

Zweck: Um erklärende Bemerkungen in ein Programm einzufügen.

Bemerkung: REM-Anweisungen werden nicht ausgeführt, sie werden jedoch genauso ausgegeben, wie sie eingegeben werden, wenn das Programm gelistet wird.

REM-Anweisungen können als Verzweigungsadresse benutzt werden (von GOTO oder GOSUB). Die Programmausführung wird dann mit der ersten auf REM folgenden ausführbaren Anweisung fortgesetzt.

Außer in die 8K-Version können Bemerkungen ans Ende einer Zeile mit einem einfachen Anführungsstrich (Apostroph) statt mit :REM angehängt werden.

Beispiel:

```
.
.
.
120 REM BERECHNUNG DER DURCHSCHNITTS-
          GESCHWINDIGKEIT
130 FOR I = 1 TO 20
140 SUM = SUM + V(I)
.
.
.
```

oder in den Versionen "Erweitert" und "Platte"

```
.
.
.
120 FOR I = 1 TO 20 'BERECHNUNG DER
                      DURCHSCHNITTS-
                      GESCHWINDIGKEIT'
130 SUM = SUM + V(I)
140 NEXT I
.
.
.
```

RENUM

Format: RENUM [[<neue nummer>][,[<alte nummer>][,<abstand>]]]

Zweck: Zur Neu-Numerierung der Progammzeilen

Bemerkung: <neue nummer> bezeichnet die erste Zeilennummer in der neuen Sequenz. Der Wert 10 wird eingesetzt, falls nicht angegeben, <alte nummer> bezeichnet die Zeilennummer des momentanten Programmes, bei der die Neu-Numerierung beginnen soll. Ist keine Angabe gemacht, so wird die erste Programmzeile genommen. <abstand> bezeichnet den gewünschten Abstand der Zeilennummern. Der Wert 10 wird genommen, falls keine Angabe gemacht.

RENUM ändert auch alle Zeilennummer-Angaben in den Anweisungen GOTO, GOSUB, THEN, ON ... GOTO, ON ... GOSUB sowie ERL so um, daß sie den neu vergebenen Zeilennummern entsprechen. Sollte dies eine nicht mehr existierende Zeilennummer ergeben, wird die Fehlermeldung "Undefined line xxxxx in yyyyy" (Undefinierte Zeile xxxxx in yyyyy) angezeigt. Die unkorrekte Zeilennummer (xxxxx) wird durch RENUM nicht verändert. Die Zeilennumer yyyyy kann geändert werden.

Anmerkung: Mit RENUM kann nicht die Reihenfolge der Programmzeilen verändert werden (z.B. RENUM 15,30, wenn das Programm die drei Zeilen 10, 20 und 30 hat.) Es können auch keine Programmzeilennummern über 65529 erzeugt werden. In diesem Fall wird der Fehler "Illegal function call" (Illegaler Funktionsaufruf) angezeigt.

Beispiel:		
	RENUM	Numeriert das gesamte Programm. Die erste neue Zeilennummer ist 10. Der Abstand der Zeilennummern ist auch 10.

RENUM 300,,50	Numeriert das gesamte Programm. Die erste neue Zeilennummer ist 300. Der Abstand der Zeilennummern ist 50.
RENUM 1000,900,20	Numeriert die Zeilen ab Zeile 900 neu, so daß sie mit 1000 beginnen und einen Abstand von 20 haben.

RESTORE

Format: RESTORE [<zeilennummer>]

Zweck: Erlaubt, DATA-Anweisungen von einer spezifizierten Zeile an neu zu lesen.

Bemerkung: Nach Ausführung von RESTORE greift die nächste READ-Anweisung auf das erste Element der ersten DATA-Anweisung im Programm zu. Ist <zeilennummer> angegeben, so greift die nächste READ-Anweisung auf das erste Element in der angegebenen DATA-Anweisung zu.

Beispiel:

```
10 READ A,B,C
20 RESTORE
30 READ D,E,F
40 DATA 57,68,79
 .
 .
 .
```

RESUME

Format:
```
RESUME
RESUME 0
RESUME NEXT
RESUME <zeilennummer>
```

Zweck: Zur Programmfortsetzung nach der Ausführung einer Fehlerkorrekturroutine.

Bemerkung: Jedes der vier Formate kann, abhängig davon, wo das Programm fortgesetzt werden soll, benutzt werden.

RESUME oder RESUME 0	Die Fortsetzung beginnt bei der Zeile, die den Fehler verursacht hat.
RESUME NEXT	Die Fortsetzung beginnt bei der Zeile nach der Zeile, die den Fehler verursacht hat.
RESUME <zeilennummer>	Die Fortsetzung beginnt bei <zeilennummer>

Eine RESUME-Anweisung außerhalb einer Fehlerroutine bewirkt die Anzeige "RESUME without error" (RESUME ohne Fehler).

Beispiel:
```
10 ON ERROR GOTO 900
 .
 .
 .
900 IF (ERR=230) AND (ERL=90) THEN PRINT
    "BITTE NOCHMALS VERSUCHEN":
    RESUME 80
 .
 .
 .
```

RUN

Format 1: RUN [<zeilennummer>]

Zweck: Start des Programmes, das im Speicher ist.

Bemerkung: Ist <zeilennummer> angegeben, beginnt die Programmausführung an dieser Stelle, andernfalls mit der ersten Programmzeile. MS-BASIC kehrt anschließend immer in den Direktmodus zurück.

Beispiel: RUN

Format 2: RUN <programmname> [,R]

Version: Platte

Zweck: Laden einer Programmdatei von der Platte in den Speicher und Ausführung des Programmes.

Bemerkung: <programmname> ist der name, mit dem das Programm in der SAVE-Anweisung auf die Platte geschrieben wurde. (Mit CP/M wird — falls nicht anders angegeben — .BAS angehängt).

RUN schließt alle eröffneten Dateien und löscht den Speicherinhalt bevor das angegebene Programm geladen wird. Mit der Angabe "R" bleiben die Dateien jedoch eröffnet.

Beispiel: RUN "NEUPROG",R

Bemerkung: Der MS-BASIC Compiler unterstützt die RUN und RUN <Zeilennummer>-Formen der RUN-Anweisung, aber nicht die "R"-Option mit RUN. Falls Sie diese Leistung benötigen, ist die CHAIN-Anweisung zu benutzen.

SAVE

Format: SAVE <programmname> [,A|,P]

Zweck: Speichert ein Programm auf der Platte ab.

Bemerkung: <programmname> ist ein Text in Anführungszeichen. Er muß den Vorschriften für Dateinamen des jeweiligen Betriebssystems entsprechen. (CP/M fügt — falls nicht anders angegeben — den Anhang .BAS zu). Falls bereits eine Datei <programmname> existiert, wird sie überschrieben.

Mit dem Anhang A wird das Programm im ASCII-Format ausgegeben. Andernfalls wird bei BASIC ein komprimiertes Binärformat verwendet. ASCII benötigt mehr Platz auf der Platte, aber einige Plattenzugriffe verlangen ASCII-Format. Z.B. der Befehl MERGE und einige Betriebssystembefehle wie LIST können ASCII-Format verlangen.

Mit dem Anhang P wird das Programm in einem verschlüsselten Binärformat geschrieben. Wird dieses Programm mit RUN oder LOAD aufgerufen, so ist es nicht möglich, es zu listen oder korrigieren.

Beispiel:
```
SAVE "COM2",A
SAVE "PROG",P
```

STOP

Format: STOP

Zweck: Um ein Programm abzubrechen und in den Direktmodus zu gehen.

Bemerkung: STOP-Anweisungen können an jeder beliebigen Stelle ein Programm abbrechen. Sobald eine STOP-Anweisung gefunden wird, erscheint die Anzeige.

Break in line nnnnn (Abbruch in Zeile nnnnn)

Im Gegensatz zu END schließt STOP keine Dateien.
MS-BASIC kehrt bei STOP immer in den Direktmodus zurück. Das Programm kann durch CONT fortgesetzt werden.

Beispiel:

```
10 INPUT A,B,C
20 K=A^2*5.3:L=B^3/.26
30 STOP
40 M=C*K+100:PRINT M
RUN
?1,2,3
BREAK in 30
Ok
PRINT L
 30.7692
Ok
CONT
 115.9
Ok
```

SWAP

Format: SWAP <variable>,<variable>

Zweck: Zum Austausch der Werte zweier Variablen.

Bemerkung: Jeder Typ von Variablen kann bei SWAP verwendet werden (ganzzahlig, einfache und doppelte Genauigkeit, Texte), aber die beiden Variablen müssen vom selben Typ sein, sonst wird "Type mismatch" (Typenfehler) angezeigt.

Beispiel:

```
LIST
10 A$ = "ZWEI": B$= "DREI" ;C$= "MAL"
20 PRINT A$ C$ B$
30 SWAP A$ B$
40 PRINT A$ C$ B$
Ok
RUN
 ZWEI MAL DREI
 DREI MAL ZWEI
Ok
```

TRON/ TROFF

Format: TRON

TROFF

Zweck: Zur Verfolgung des Programmablaufes

Bemerkung: Als Hilfsmittel bei der Fehlersuche bewirkt TRON (entweder direkt eingegeben oder als Anweisung im Programm), daß jeweils die Nummer der ausgeführten Zeile angezeigt wird. Die Zeilennummern werden in eckigen Klammern angezeigt. Diese Leistung wird durch TROFF (oder NEW) wieder abgeschaltet.

Beispiel:

```
TRON
Ok
LIST
10 K = 10
20 FOR J = 1 TO 2
20 L = K + 10
40 PRINT J;K;L
50 K = K+10
60 NEXT
70 END
Ok
RUN
[10][20][30][40]   1   10   20
[50][60][30][40]   2   20   30
[50][60][70]
Ok
TROFF
Ok
```

WAIT

Format: WAIT <eingabekanal>,I[,J]
wobei I und J ganzzahlige Ausdrücke sind.

Zweck: Zur Unterbrechung des Programmes während der Zustand eines Eingabekanals des Systemes beobachtet wird.

Bemerkung: WAIT unterbricht das Programm solange, bis ein bestimmter Eingabekanal einen bestimmten Binärwert meldet. Jeder auf diesem Kanal eingehende Wert wird mit dem Ausdruck J verglichen. Ist das Vergleichsergebnis falsch (0), so wird der nächste Wert auf diesem Kanal ausgelesen und überprüft. Sobald das Vergleichsergebnis wahr (1) ist, wird das Programm mit der nächsten Zeile fortgesetzt. Falls J nicht angegeben wurde, wird als Vergleichswert 0 angenommen.

ACHTUNG: Es ist möglich, mit der WAIT-Anweisung eine unendlicht Schleife zu beginnen. In diesem Falle muß das System von Hand neu gestartet werden.

Beispiel: 100 WAIT 32,2

WHILE ... WEND

Format:
```
WHILE <ausdruck>
  .
  .
  .
[<schleifen-anweisungen>]
  .
  .
WEND
```

Zweck: Dient dazu, eine Reihe von Anweisungen in einer Schleife auszuführen, solange eine gegebene Bedingung erfüllt ist.

Bemerkung: Wenn der <ausdruck> nicht Null (d.h. wahr) ist, werden die <schleifen-anweisungen> bis WEND ausgeführt. BASIC kehrt dann zur WHILE-Anweisung zurück und überprüft den <ausdruck>. Falls nicht wahr, wird das Programm mit der Anweisung, die der WEND-Anweisung folgt, fortgesetzt.

WHILE/WEND-Schleifen können beliebig oft geschachtelt werden. Jedes WEND wird dem jeweils zuletzt gelesenen WHILE zugeordnet. Bleibt ein WHILE übrig, wird "WHILE without WEND" (WHILE ohne WEND) angezeigt, bleibt ein WEND übrig, wird "WEND without WHILE" (WEND ohne WHILE) angezeigt.

Beispiel:
```
 90 'SORTIEREN DER TABELLE A$
100 FLIPS = 1 'UM EINEN DURCHGANG ZU
    ERZWINGEN
110 WHILE FLIPS
115             FLIPS = 0
120             FOR I=1 TO J—1
130                     IF A$ (I)>A$ (I+1) THEN
                                SWAP A$ (I),A$ (I+1)
                                : FLIPS = 1
140             NEXT I
150 WEND
```

WIDTH

Format: WIDTH [LPRINT] <ganzzahliger ausdruck>

Zweck: Bestimmung der maximalen Zeilenlänge am Systemdrucker.

Bemerkung: Wird der Zusatz LPRINT weggelassen, so bezieht sich die Zeilenbreite auf den Bildschirm. Mit LPRINT wird die Zeilenbreite am Systemdrucker definiert.

<ganzzahliger ausdruck> muß einen Wert zwischen 15 und 255 haben. Automatisch wird 72 angenommen.

Wenn 255 eingesetzt wird, ist die Zeilenbreite „unendlich", d.h. BASIC setzt kein Symbol für <eingabetaste> ein. Die Position des Zeigers oder Druckkopfes, wenn sie mit den Funktionen POS oder LPOS abgefragt wird, gibt nach Position 255 den Wert 0 an.

Beispiel:
```
10 PRINT "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
RUN
ABCDEFGHIJKLMNOPQRSTUVWXYZ
Ok
WIDTH 18
Ok
RUN
ABCDEFGHIJKLMNOPQR
STUVWXYZ
Ok
```

WRITE

Format: WRITE [<liste von ausdrücken>]

Zweck: Datenausgabe am Bildschirm

Bemerkung: Sind keine Ausdrücke angegeben, wird eine Leerzeile angezeigt. Ist die <liste von ausdrücken> vorhanden, werden die Werte der Ausdrücke am Bildschirm angezeigt. Die Ausdrücke können numerisch und/oder alphanumerisch sein. Sie müssen durch Kommas getrennt werden.

Wenn die Daten angezeigt werden, werden sie durch Kommas voneinander getrennt. Texte werden in Anführungszeichen gesetzt. Nach der Anzeige des letzten Wertes fügt BASIC ein Symbol <eingabetaste/zeilenschaltung> ein.

WRITE zeigt numerische Werte in derselben Formatierung an wie die PRINT-Anweisung, Seite 2-61.

Beispiel:
```
10 A = 80:B = 90:C$ = "DAS IST ALLES"
20 WRITE A,B,C$
RUN
 80, 90, "DAS IST ALLES"
Ok
```

WRITE#

Format: WRITE# <dateinummer>,<liste von ausdrücken>

Zweck: Schreiben von Daten auf sequentielle Dateien.

Bemerkung: <dateinummern> ist die Nummer, unter der die Datei mit "0" in OPEN eröffnet wurde. Die Ausdrücke in der Liste können numerisch oder alphanumerisch sein und müssen durch Kommas getrennt werden.

Der Unterschied zwischen WRITE# und PRINT# ist der, daß WRITE# automatisch Kommas zwischen die Daten setzt und Texte in Anführungsstriche setzt. Deshalb ist es nicht erforderlich, ausdrücklich Trennsymbole einzufügen. Eine Folge von <eingabetaste/zeilenschaltung> wird hinter den letzten Wert auf die Platte geschrieben.

Beispiel: Bei A$ = "KAMERA" und B$ = "93604-1". Die Anweisung:

```
WRITE# 1, A$, BS
```

schreibt folgendes auf die Platte:

"KAMERA","93604-1".

Eine nachfolgende INPUT#-Anweisung, wie z.B.:

```
INPUT# 1,A$,B$
```

würde "KAMERA" in A$ und "93604-1" in B$ einsetzen.

KAPITEL 3

MS-BASIC FUNKTIONEN

In diesem Kapitel werden die Funktionen, die MS-BASIC bietet, behandelt. Die Funktionen können von jedem Programm ohne weitere Definitionen aufgerufen werden.

Argumente zu Funktionen werden immer in Klammern angegeben. In der Formatbeschreibung der Funktionen in diesem Kapitel werden folgende Abkürzungen gewählt:

X und Y	stehen für numerische Ausdrücke allgemein
I und J	stehen für ganzzahlige Ausdrücke
X$ und Y$	stehen für Text- (String-) Ausdrücke

Wird ein Gleitkommawert angegeben, wo ein ganzzahliger Wert gebraucht wird, rundet MS-BASIC die Nachkommastellen und benutzt die entstandene ganze Zahl.

ACHTUNG: Der MS-BASIC Interpreter ermittelt in Funktionen nur Ergebnisse, die ganzzahlig sind oder einfache Genauigkeit haben. Nur der BASIC-Compiler bringt bei Funktionen Ergebnisse mit doppelter Genauigkeit.

ABS

Format: ABS(X)

Aktion: Ergibt den absoluten Wert des Ausdruckes X.

Beispiel:
```
PRINT ABS(7*(-5))
 35
Ok
```

ASC

Format: ASC(X$)

Aktion: Ergibt einen numerischen Wert, der dem ASCII-Kode des ersten Zeichens des Textes X$ entspricht. (Siehe Anhang I, ASCII-Kodes).

Wenn X$ "Null" ist, wird ein "Illegal function call" (illegaler Funktionsaufruf) angezeigt.

Beispiel:
```
10 X$="TEST"
20 PRINT ASC(X$)
RUN
 84
Ok
```

Siehe auch Funktion CHR$ zur Umwandlung von ASCII-Kode in Text.

ATN

Format: ATN(X)

Aktion: Ergibt den Arcustangens von X in Bogengraden. Das Ergebnis ist im Bereich $-\pi/2$ bis $\pi/2$. Der Ausdruck X kann von beliebigem numerischem Typ sein. Das Ergebnis wird immer mit einfacher Genauigkeit ermittelt.

Beispiel:
```
10 INPUT X
20 PRINT ATN(X)
RUN
? 3
 1.24905
Ok
```

CDBL

Format: CDBL(X)

Aktion: Wandelt X in eine Zahl mit doppelter Genauigkeit um.

Beispiel:
```
10 A = 454.67
20 PRINT A;CDBL(A)
RUN
 454.67  454.6700134277344
Ok
```

CHR$

Format: CHR$(I)

Aktion: Ergibt einen Text, dessen einziges Zeichen den ASCII-Kode I hat. (ASCII-Kodes sind aus Anhang I ersichtlich). CHR$ wird allgemein benutzt, um ein einzelnes Zeichen an den Bildschirm zu senden. Zum Beispiel kann der Fehlerton (BEL=CHR$(7)) vor einer Fehlernachricht ausgegeben werden, oder ein Symbol für Formularaufschaltung (Form Feed = CHR$(12)) wird gesandt, um den Bildschirm zu löschen und den Zeiger in die Ausgangsposition zu bringen. Darüber hinaus dient CHR$ zur Positionierung des Zeigers, wie im nachfolgenden Beispiel gezeigt.

Beispiel:
```
PRINT CHR$(27) + CHR$(61) + CHR$
      (32+Zeilennummer)
      (Escape) (Zeigerfunktion) (Zeilennummer)
      + CHR$(32+Kolonne)
      (Kolonne)
```

CINT

Format: CINT(X)

Aktion: Wandelt X in einen ganzzahligen Wert um, indem die Nachkommastellen gerundet werden. Falls X nicht im Bereich −32768 bis 32767 ist, wird ein "Overflow" (Überlauf) als Fehler angezeigt.

Beispiel:
```
PRINT CINT(45.67)
 46
Ok
```

Siehe die Funktionen CDBL und CSNG, welche Zahlen in solche mit doppelter bzw. einfacher Genauigkeit umwandeln. Siehe auch die Funktionen FIX und INT, die ebenfalls ganzzahlige Ergebnisse bringen.

COS

Format: COS(X)

Aktion: Ergibt den Kosinus von X in Bogengraden. Die Berechnung von COS(X) erfolgt mit einfacher Genauigkeit.

Beispiel:
```
10 X=2*COS(.4)
20 PRINT X
RUN
 1.84212
Ok
```

CSNG

Format; CSNG(X)

Aktion: Wandelt X in eine Zahl mit einfacher Genauigkeit um.

Beispiel:
```
10 A# = 975.3421#
20 PRINT A#;CSNG(A#)
RUN
 975.3421  975.342
Ok
```

Siehe die Funktionen CINT und CDBL zur Umwandlung in ganzzahlige Zahlen und Zahlen mit doppelter Genauigkeit.

CVI, CVS, CVD

Format: CVI (< 2-byte-text)
CVS (< 4-byte-text)
CVD (< 8-byte-text)

Aktion: Wandelt Textwerte in Zahlen um. Numerische Werte, die von einer Plattendatei mit wahlfreiem Zugriff gelesen wurden, müssen von Texten in Zahlen zurückgewandelt werden. CVI wandelt einen 2-stelligen Text in einen ganzzahligen Wert, CVS wandelt einen 4-stelligen Text in eine Zahl mit einfacher Genauigkeit und CVD wandelt einen 8-stelligen Text in eine Zahl mit doppelter Genauigkeit.

Beispiel:

```
.
.
.
10 FIELD#1,4 AS N$, 12 AS B$,...
80 GET#1
90 Y=CVS(N$)
.
.
.
```

Siehe auch MKI$, MKS$ und MKD$, Seite 3—16.

EOF

Format: EOF <dateinummer>

Aktion: Ergibt −1 (wahr) sobald das Ende einer sequentiellen Datei erreicht wird. EOF sollte bei der Eingabe (INPUT) aus einer sequentiellen Datei benutzt werden, um "Input past end" (Eingabe nach Dateiende)-Fehler zu vermeiden.

Beispiel:
```
10 OPEN "I",1,"DATEN"
20 C=0
30 IF EOF(1) THEN 100
40 INPUT#1,M(C)
50 C=C+1:GOTO 30
 .
 .
 .
```

EXP

Format: EXP(X)

Aktion: Ergibt e hoch X. X muß <=87.3365 sein. Ergibt EXP einen Überlauf, so wird dieser angezeigt ("Overflow"), das Unendlichzeichen des Systems wird als Ergebnis ausgegeben und das Programm wird fortgesetzt.

Beispiel:
```
10 X=5
20 PRINT EXP(−1)
RUN
 54.5982
Ok
```

FIX

Format: FIX(X)

Aktion: Ergibt den abgeschnittenen ganzzahligen Teil von X. FIX(X) ist gleich SGN(X)*INT(ABS(X)). Der Hauptunterschied zwischen INT und FIX ist der, daß FIX bei negativem X nicht die nächstniedrigere Zahl ergibt.

Beispiel:
```
PRINT FIX(58.75)
 58
Ok

PRINT FIX(—58.75)
—58
Ok
```

FRE

Format: FRE(0)
FRE(X$)

Aktion: Die Argumente bei FRE sind ohne Bedeutung. FRE ergibt die Anzahl der Stellen im Speicher, die nicht von MS-BASIC belegt sind.

FRE (" ") erzwingt eine Speicherbereinigung bevor die Anzahl der freien Stellen ermittelt wird. BITTE GEDULD: die Speicherbereinigung kann 1 bis 1 1/2 Minuten dauern. BASIC leitet keine Speicherbereinigung ein, solange nicht der ganze freie Speicher aufgebraucht ist. Deshalb wird die Zeit für Speicherbereinigungen durch periodische Anwendung von FRE (" ") reduziert.

Beispiel:
```
PRINT FRE(0)
 14542
Ok
```

HEX$

Format: HEX$(X)

Aktion: Ergibt einen Text, der den hexadezimalen Wert des dezimalen Argumentes darstellt. X wird zu einem ganzzahligen Wert gerundet, bevor HEX$(X) ermittelt wird.

Beispiel:

```
10 INPUT X
20 A$=HEX$(X)
30 PRINT "DEZIMAL:"
   X "IST HEXADEZIMAL:" A$
RUN
? 32
DEZIMAL: 32 IST HEXADEZIMAL 20
Ok
```

Siehe Funktion OCT$ für Umwandlung in Oktalzahlen.

INKEY$

Format: INKEY$

Aktion: Ergibt entweder ein Textfeld mit einem Zeichen von der Tastatur oder ein leeres Textfeld, wenn kein Zeichen eingegeben wurde. Mit Ausnahme von Control-C, welches das Programm unterbricht, werden alle Zeichen direkt dem Programm übergeben. Die Zeichen werden nicht am Bildschirm angezeigt. (Beim BASIC-Compiler wird auch Control-C ans Programm übergeben.)

Beispiel:
```
1000 'UNTERPROGRAMM FUER EINGABE MIT
     ZEITLIMIT
1010 ANTWORT$=" "
1020 FOR I%=1 TO ZEITLIMIT%
1030 A$=INKEY$: IF LEN(A$)=0 THEN 1060
1040 IF ASC(A$)=13 THEN ZEITLIMIT%=0:
     RETURN
1050 ANTWORT$=ANTWORT$+A$
1060 NEXT I%
1070 ZEITLIMIT%=1:RETURN
```

INP

Format: INP(I)

Aktion: Ergibt das Byte, das vom Kanal I gelesen wurde. I muß im Bereich von 0-255 sein. INP ist die Komplementärfunktion zur OUT-Anweisung, Seite 2-57.

Beispiel:
```
100 A = INP(255)
```

INPUT$

Format: INPUT$(X[,[#]Y])

Aktion: Ergibt einen Text mit X Zeichen von der Tastatur oder von Datei Nummer Y. Erfolgt die Eingabe über die Tastatur, so werden keine Zeichen am Bildschirm angezeigt. Alle Kontrollzeichen außer Control-C werden ans Programm übergeben. Control-C dient zur Unterbrechung der INPUT$-Funktion.

Beispiel 1:

```
 5 'HEXADEZIMALE LISTE EINER
   SEQUENTIELLEN DATEI
10 OPEN "I",1,"DATEN"
20 IF EOF(1) THEN 50
30 PRINT HEX$(ASC(INPUT$(1,#1)));
40 GOTO 20
50 PRINT
60 END
```

Beispiel 2:

```
  .
  .
  .
100 PRINT "EINGABE: W = WEITER, H = HALT"
110 X$ = INPUT$(1)
120 IF X$ = "W" THEN 500
130 IF X$ = "H" THEN 700 ELSE 100
  .
  .
  .
```

INSTR

Format: INSTR ([I,] X$, Y$)

Aktion: Sucht nach dem ersten Text Y$ innerhalb des Textes X$ und ergibt die Stellenposition, wo Y$ gefunden wurde. Mit I kann wahlweise die Startposition für die Suche innerhalb X$ definiert werden. I muß im Bereich 1 bis 255 sein. Wenn I>LEN(X$), oder wenn X$ leer ist, oder wenn Y$ nicht gefunden wird, ergibt INSTR den Wert 0. Wenn Y$ leer ist, ergibt INSTR den Wert 1 oder I. X$ und Y$ können Variable, Ausdrücke oder Texte in Anführungsstrichen sein.

Beispiel:
```
10 X$ = "ABCDEB"
20 Y$ = "B"
30 PRINT INSTR(X$, Y$);INSTR (4,X$, Y$)
RUN
 2   6
Ok
```

Anmerkung: Wird I=0 angegeben, so erscheint die Fehlermeldung "Illegal Argument in nnnnn" (Unerlaubtes Argument in nnnnn), wobei nnnnn die Zeilennummer ist.

INT

Format: INT(X)

Aktion: Ergibt den größten ganzzahligen Wert< =X.

Beispiel:
```
PRINT  INT (99.98)
 99
Ok

PRINT  INT (—12.11)
—13
Ok
```

Siehe auch die Funktionen FIX und CINT, die auch ganzzahlige Werte ergeben.

LEFT$

Format: LEFT$(X$, I)

Aktion: Ergibt einen Text, der den Zeichen am linken Rand vom Text X$ in der Anzahl I entspricht. I muß im Bereich von 0 bis 255 sein. Ist I > LEN(X$), so ergibt das den ganzen Text von X$. Ist I = 0, so ergibt es den Leertext (mit der Länge 0).

Beispiel:
```
10 A$ = "BASIC-80"
20 B$ = LEFT$(A$,5)
30 PRINT B$
BASIC
Ok
```

Siehe auch die Funktionen MID$ und RIGHT$.

LEN

Format: LEN(X$)

Aktion: Ergibt die Anzahl der Zeichen in X$. Auch nichtdruckende Zeichen und Leerzeichen werden gezählt.

Beispiel:
```
10 X$= "HANNOVER, NIEDERSACHSEN"
20 PRINT LEN (X$)
 23
Ok
```

LOC

Format: LOC (<dateinummer>)

Aktion: Bei Dateien mit wahlfreiem Zugriff ergibt LOC die Nummer des nächsten Satzes, der bei den Anweisungen GET oder PUT (ohne Satznummer) benutzt wird. Bei sequentiellen Dateien ergibt LOC die Anzahl der Sektoren (128 Byte lange Blöcke), die von der Datei gelesen oder auf die Datei geschrieben wurden seit der Eröffnung in OPEN.

Beispiel: 200 IF LOC(1) > SO THEN STOP

LOG

Format: LOG(X)

Aktion: Ergibt den natürlichen Logarithmus von X. X muß größer Null sein.

Beispiel:
```
PRINT LOG(45/7)
 1.86075
Ok
```

LPOS

Format: LPOS(X)

Aktion: Ergibt die augenblickliche Druckposition des Systemdruckers bzw. des Druckerpuffers. Ergibt nicht zwingenderweise die physische Position des Druckkopfes. X ist ein Füllargument ohne Bedeutung.

Beispiel: 100 IF LPOS(X) > 60 THEN LPRINT CHR$(13)

MID$

Format: MID$(X$,I[,J])

Aktion: Ergibt einen Textausschnitt aus X$ mit der Länge J, beginnend mit Zeichen I von X$. I und J müssen im Bereich zwischen 1 und 255 sein. Wird J weggelassen, oder ist J größer als X$ Stellen rechts von I hat, so ergeben sich alle Zeichen von X$ beginnend bei Stelle I. Falls I > LEN(X$), ergibt MID$ einen Leertext.

Beispiel:
```
LIST
10 A$ = "GUTEN"
20 B$ = "MORGEN  ABEND  TAG"
30 PRINT A$; MID$(B$,8,5)
Ok
RUN
GUTEN ABEND
Ok
```

Siehe auch die Funktionen LEFT$ und RIGHT$.

Anmerkung: Wird I = 0 spezifiziert, so erscheint die Fehlermeldung "Illegal argument in nnnnn". (Unerlaubtes Argument in nnnnn), wobei nnnnn die Zeilennummer ist.

MKI$, MKS$, MKD$

Format: MKI$ (<ganzzahliger Ausdruck>)
MKS$(<ausdruck mit einfacher genauigkeit>)
MKD$(<ausdruck mit doppelter genauigkeit>)

Aktion: Wandelt numerische Werte in Texte um. Jeder numerische Wert, der mittels der Anweisungen LSET oder RSET in den Pufferbereich einer wahlfreien Datei übergeben werden soll, muß vorher in einen Text umgewandelt werden. MKI$ wandelt einen ganzzahligen Ausdruck in 2 Textstellen um, MKS$ einen Ausdruck mit einfacher Genauigkeit in 4 Textstellen und MKD$ einen Ausdruck mit doppelter Genauigkeit in 8 Textstellen.

Beispiel:
```
 90 WERT = (K+T)
100 FIELD# 1, 8 AS D$,20 AS N$
110 LSET D$ = MK$ (WERT)
120 LSET N$ = A$
130 PUT # 1
 .
 .
 .
```

Siehe auch CVI, CVS, CVD, Seite 3-6

OCT$

Format: OCT$(X)

Aktion: Ergibt einen Text, der den Oktalwert des dezimalen Argumentes darstellt. X wird zu einem ganzzahligen Wert gerundet, bevor OCT$(X) ermittelt wird.

Beispiel:
```
PRINT OCT$(24)
 30
Ok
```

Siehe Funktion HEX$ für Umwandlung in hexadezimale Werte

PEEK

Format: PEEK(I)

Aktion: Ergibt den Inhalt (Byte = ganzzahliger Wert von 0 bis 255) der Speicherstelle I. Bei der 8K-Version muß I kleiner 32768 sein. Um eine Speicherstelle über 32768 abzufragen, wird 65536 von der Speicheradresse subtrahiert. Bei den Versionen "Erweitert" und "Platte" muß I im Bereich 0 bis 65536 sein. PEEK ist die Komplementärfunktion zur Anweisung POKE.

Beispiel: A = PEEK (&H5A00)

POS

Format: POS(I)

Aktion: Ergibt die momentane Position des Zeigers am Bildschirm. Die äußerst linke Position ist 1. X ist ein Füllargument ohne Bedeutung.

Beispiel: IF POS(X) > 60 THEN PRINT CHR$(13)

Siehe auch LPOS.

RIGHT$

Format: RIGHT$(X$,I)

Aktion: Ergibt einen Text, der den Zeichen am rechten Rand vom Text X$ in der Anzahl I entspricht. I muß im Bereich von 0 bis 255 sein. Ist I> LEN(X$), so ergibt das den ganzen Text von X$. Ist I = 0, so ergibt es den Leertext (mit der Länge 0).

Beispiel:
```
10 A$ = "PLATTE MS-BASIC"
20 PRINT RIGHT$(A$, 8)
RUN
MS-BASIC
Ok
```

Siehe auch die Funktionen MID$ und LEFT$.

RND

Format: RND[(X)]

Aktion: Ergibt eine Zufallszahl zwischen 0 und 1. Die Reihenfolge der Zufallszahlen ist bei jedem Programmablauf dieselbe, falls nicht der Zufallsgenerator zurückgesetzt wurde (siehe RANDOMIZE, Seite 2-72). Wird X< 0 angegeben, so wird für jedes gegebene X die gleiche Reihenfolge begonnen.

X>0 oder kein X ermittelt die nächste Zufallszahl in der Reihe. X = 0 wiederholt die letzte Zufallszahl.

Beispiel:
```
10 FOR I = 1 TO 5
20 PRINT INT (RND*100)
30 NEXT
RUN
 24  30  31  51  5
Ok
```

SGN

Format: SGN(X)

Aktion: Bei X>0 ergibt SGN(X) den Wert 1.
Bei X=0 ergibt SGN(X) den Wert 0.
Bei X<0 ergibt SGN(X) den Wert −1.

Beispiel: ON SGN(X)+2 GOTO 100,200,300 ergibt eine Verzweigung nach 100, wenn X negativ ist, nach 200, wenn X Null ist und nach 300, wenn X positiv ist.

SIN

Format: SIN(X)

Aktion: Ergibt den Sinus von X in Bogengraden. SIN(X) wird mit einfacher Genauigkeit ermittelt. COS(X) = SIN (X+3.14159/2).

Beispiel:
```
PRINT SIN(1.5)
 .997495
Ok
```

SPACE$

Format: SPACE$(X)

Aktion: Ergibt einen Text mit der Länge X, bestehend aus Leerzeichen. Der Ausdruck X wird zu einem ganzzahligen Wert gerundet und muß im Bereich 0 bis 255 sein.

Beispiel:
```
10 FOR I = 1 TO 5
20 X$ = SPACE$(I)
30 PRINT X$;I
40 NEXT I
RUN
 1
  2
   3
    4
     5
Ok
```

Siehe auch Funktion SPC.

SPC

Format: SPC(I)

Aktion: Gibt eine Anzahl von I Leerzeichen aus. SPC kann nur in den Anweisungen PRINT und LPRINT verwendet werden. I muß im Bereich 0 bis 255 sein.

Beispiel:
```
PRINT "DORT" SPC(15) "HINTEN"
DORT                 HINTEN
Ok
```

Siehe auch Funktion SPACE$.

SQR

Format: SQR(X)

Aktion: Ergibt die Quadratwurzel von X. X muß >=0 sein.

Beispiel
```
10 FOR X = 10 TO 25  STEP 5
20 PRINT X,SQR(X)
30 NEXT
RUN
 10                    3.16228
 15                    3.87298
 20                    4.47214
 25                    5
Ok
```

STR$

Format: STR$(X)

Aktion: Ergibt die Textdarstellung des Wertes von X.

Beispiel:
```
 5 'RECHNEN FUER KINDER
10 INPUT "GEBE EINE ZAHL EIN";N
20 ON LEN(STR$(N)) GOSUB 30,100,200,300,
                         400,500
      .
      .
      .
```
Siehe auch Funktion VAL.

STRING$

Format: STRING$(I,J)
STRING$(I,X$)

Aktion: Ergibt einen Text mit der Länge I, der aus dem Zeichen mit dem ASCII-Kode J oder dem ersten Zeichen des Textes X$ besteht.

Beispiel:
```
10 X$ = STRING$(10,45)
20 PRINT X$ "MONATSBERICHT" X$
RUN
----------MONATSBERICHT----------
Ok
```

TAB

Format: TAB(I)

Aktion: Tabuliert auf Position I. Ist die augenblickliche Position bereits über I, so wird auf Position I der nächsten Zeile tabuliert. Die äußerst linke Position hat die Nummer 1, die äußerst rechte Position ist die Breite (Bildschirm- oder Druckerzeile) minus Eins. I muß im Bereich von 1 bis 255 sein. TAB kann nur in den Anweisungen PRINT und LPRINT vorkommen.

Beispiel:
```
10 PRINT "NAME" TAB(25) "BETRAG":PRINT
20 READ A$,B$
30 PRINT A$ TAB(25) B$
40 DATA "F.HUBER", "DM 25.00"
RUN
NAME                    BETRAG

F. HUBER                DM  25.00
Ok
```

TAN

Format: TAN(X)

Aktion: Ergibt den Tangens von X in Bogengraden. Der Tangens wird mit einfacher Genauigkeit ermittelt. Falls TAN überläuft, wird "Overflow" angezeigt und das Unendlichzeichen des Systems mit dem entsprechenden Vorzeichen wird als Ergebnis eingesetzt. Das Programm wird fortgesetzt.

Beispiel:
```
10 Y = Q*TAN(X)/2
```

USR

Format: USR[<ziffer>](X)

Aktion: Ruft die Assembler Unterroutine des Benutzer auf, die das Argument X hat. <ziffer> kann nur in den Versionen "Erweitert" und "Platte" benutzt werden. <ziffer> muß im Bereich von 0 bis 9 sein und bezieht sich auf die Ziffer, die der betreffenden Routine mit der Anweisung DEF USR zugeordnet wurde. Wird <ziffer> nicht angegeben, so wird USR0 angenommen.

Beispiel:
```
40 B = T*SIN(Y)
50 C = USR(B/2)
60 D = USR (B/3)
 .
 .
 .
```

VAL

Format: VAL(X$)

Aktion: Ergibt den numerischen Wert des Textes X$. VAL streicht dabei die führenden Leerzeichen, Tabs und Zeilenschaltsymbole des Textes X$.
Zum Beispiel:

VAL (" −3")

ergibt −3.

Beispiel:
```
10 READ NAME$, STRASSE$, PLZ$, ORT$
20 IF VAL (PLZ$) < 8000 OR VAL(PLZ$)
   > 8500 THEN PRINT NAME$ TAB(25)
   "AUSSERHALB"
30 IF VAL (PLZ$) > = 8900 AND VAL (PLZ$)
   < = 8999'THEN PRINT NAME$ TAB(25)
   "SCHWABEN"
 .
 .
 .
```

Siehe Funktion STR$ für die Umwandlung von numerischen Werten in Texte.

VARPTR

Format 1: VARPTR (< variablenname>)

Format 2: VARPTR (# < dateinummer>)

Aktion: Format 1: Gibt die Adresse des ersten Zeichens des Feldes mit der Bezeichnung < variablenname> an. Vor Ausführung von VARPTR muß dem < variablennamen> ein Wert zugewiesen werden. Andernfalls erfolgt die Fehlermeldung "Illegal function call" (Unerlaubter Funktionsaufruf). Jede Variablentype (Numerisch, Text, Tabelle) kann verwendet werden. Die Adresse ist im Bereich von 32767 bis —32768. Bei negativen Antworten addiert man den Wert 65536, um die tatsächliche Adresse zu ermitteln.

VARPTR benutzt man normalerweise, um die Adresse einer Variablen oder Tabelle zu erhalten, um sie an eine Assembler-Unterroutine übergeben zu können. Bei einer indizierten Variablen (z.B. A) gibt man normalerweise die Funktion VARPTR (A(0)) an, um die niedrigste Adresse der Variablen zu ermitteln.

Anmerkung: Vor der Ausführung von VARPTR für indizierte Variablen sollten alle einfachen Variablen Werte zugewiesen haben, da die Adressen für indizierte Variablen mit jeder neuen einfachen Variablen verändert werden.

Format 2: Gibt bei sequentiellen Dateien die Anfangsadresse des Pufferbereiches der Datei mit < dateinummer> an. Bei Dateien mit wahlfreiem Zugriff wird die Adresse des FIELD-PUFFERS der Datei mit < dateinummer> angegeben.

Beispiel: 100 X = USR (VARPTR(Y))

ANHANG A

ÜBERSICHT ÜBER FEHLERCODES UND FEHLERMELDUNGEN

Code	Nummer	Nachricht
NF	1	NEXT without FOR NEXT ohne FOR Eine Variable in einer NEXT-Anweisung stimmt nicht mit einer vorher ausgeführten, nicht abeschlossenen FOR-ANWEISUNG überein.
SN	2	Syntax Error Satzaufbaufehler Eine Zeile mit unkorrekter Zeichenfolge wurde entdeckt. (Z.B. unpaarige Klammern, falsch buchstabierte Befehle oder Anweisungen, falsche Satzzeichen, usw.).
RG	3	Return without GOSUB RETURN ohne GOSUB Eine RETURN-Anweisung wurde angelaufen, ohne daß eine GOSUB-Anweisung offensteht.
OD	4	Out of data Ende der Daten Eine READ-Anweisung wird ausgeführt, ohne daß weitere DATA-Anweisungen mit nicht gelesenen Daten vorhanden sind.
FC	5	Illegal function call Illegaler Funktionsaufruf Ein Parameter außerhalb des gültigen Bereiches wird an eine Text- oder Mathematik-Funktion übergeben. Ein FC-Fehler kann auch auftreten bei: • einem negativen oder übergroßen Index • einem negativen oder NULL-Argument bei LOG • einem negativen Argument bei SQR

Code	Nummer	Nachricht
		• einem Aufruf einer USR-Funktion, für die die Startadresse nicht definiert wurde • einem falschen Argument bei MID$, LEFT$, RIGHT$, INP, OUT, WAIT, PEEK, POKE, TAB, SPC, STRING$, SPACE$, INSTR oder ON ... GOTO.
OV	6	Overflow Überlauf Das Ergebnis einer Berechnung ist zu groß, um im Zahlenformat von MS-BASIC dargestellt zu werden. Bei Unterlauf wird das Ergebnis Null und die Ausführung wird fortgesetzt.
OM	7	Out of memory Kein Speicherplatz Ein Programm ist zu groß, hat zu viele FOR-Schleifen bzw. GOSUBs, zu viele Variablen oder Ausdrücke, die zu kompliziert sind.
UL	8	Undefined Line Nicht definierte Zeile Ein GOTO, GOSUB, IF ... THEN ... ELSE oder DELETE spricht eine nicht vorhandene Zeile an.
BS	9	Subscript out of range Index außerhalb des Bereiches Ein Tabellenelement wird mit entweder einem Index außerhalb des Bereiches oder mit der falschen Anzahl von Indizes angesprochen.
DD	10	Redimensioned array Gruppenvariable bereits dimensioniert Zwei DIM-Anweisungen für dieselbe Gruppenvariable sind vorhanden oder eine DIM-Anweisung wird entdeckt, nachdem das System bereits 10 Elemente zugeordnet hat.
/O	11	Division by zero Division durch Null Eine Division durch Null wurde entdeckt, oder die Ausführung einer Potenzierung ergibt NULL hoch eine negative Zahl. Im Falle der Division wird der Unendlich-Wert des Systems mit dem Vorzeichen des Dividenden versehen, beim

Code	Nummer	Nachricht
		Potenzieren wird der Enendlich-Wert mit positivem Vorzeichen als Ergebnis angenommen. In beiden Fällen wird das Programm fortgesetzt.
ID	12	Illegal direkt — Unerlaubte Direkteingabe Eine Anweisung, die im Direktmodus nicht erlaubt ist, wurde als direkter Befehl eingegeben.
TM	13	Type mismatch — Typenverwechslung Einer Textvariablen wurde ein numerischer Wert zugeordnet oder umgekehrt; eine Funktion, die ein numerisches Argument erwartet, empfing ein alphanumerisches oder umgekehrt.
OS	14	Out of string space — Kein Textspeicherplatz Textvariablen in BASIC haben den zur Verfügung stehenden freien Speicherplatz überschritten. BASIC ordnet Textspeicherplatz dynamisch zu, solange Speicherplatz überhaupt vorhanden ist.
LS	15	String too long — Text zu lange Ein Versuch wurde gemacht, einen Text mit mehr als 255 Zeichen zu erstellen.
ST	16	String formula too complex — Textformel zu komplex Ein Textausdruck ist zu lang oder zu komplex. Der Ausdruck sollte in kleinere Ausdrücke zerlegt werden.
CN	17	Can't continue — Fortsetzung nicht möglich Es wurde versucht, ein Programm fortzusetzen, das • aufgrund eines Fehlers unterbrochen wurde, • während der Unterbrechung modifiziert wurde oder • nicht existiert.

Code	Nummer	Meldung	
UF	18	Undefined user function	Benutzer-Funktion nicht definiert

Eine USR-Funktion wurde aufgerufen, bevor die Funktion definiert wurde (DEF-Anweisung).

Nur Version "Erweitert" und "Platte"

19	NO RESUME	RESUME-Anweisung fehlt

Eine Fehlerroutine ohne RESUME-Anweisung wurde angelaufen.

20	RESUME without error	RESUME ohne Fehler

Eine RESUME-Anweisung wurde außerhalb einer Fehlerroutine entdeckt.

21	Unprintable error	Fehler ohne Meldung

Für den aufgetretenen Fehler gibt es keine Fehlermeldung. Dies ist normalerweise der Fall bei einem Fehler mit indefiniertem Fehlerkode.

22	Missing operand	Ein Operand fehlt

Ein Ausdruck enthält nur einen Operanden.

23	Line buffer overflow	Überlauf des Zeilenpuffers

Es wurde versucht, eine Zeile mit zu vielen Zeichen einzugeben.

26	FOR without NEXT	FOR ohne NEXT

Ein FOR ohne dazugehöriges NEXT wurde entdeckt.

29	WHILE without WEND	WHILE ohne WEND

Einer WHILE-Anweisung fehlt das dazugehörige WEND.

Code	Nummer	Meldung

30 WEND without WHILE — WEND ohne WHILE
Ein WEND ohne dazugehöriges WHILE wurde angelaufen.

Plattenfehler

50 Field overflow — Feldüberlauf
In einer FIELD-Anweisung sollen mehr Stellen zugeordnet werden, als für die Satzlänge einer wahlfreien Datei definiert wurden.

51 Internal error — Interner Fehler
Eine interne Fehlfunktion trat in der Plattenversion von MS-BASIC auf.

52 Bad file number — Falsche Dateinummer
Eine Anweisung oder ein Befehl sprechen eine Dateinummer für eine Datei an, die nicht geöffnet ist oder außerhalb der bei der Initialisierung spezifizierten Anzahl von Dateien ist.

53 File not found — Datei nicht gefunden
Eine LOAD-, KILL- oder OPEN-Anweisung spricht eine Datei an, die auf der laufenden Platte nicht existiert.

54 Bad file mode — Falscher Dateimodus
Es wird versucht, PUT, GET oder LOF bei einer sequentiellen Datei anzuwenden, LOAD bei einer wahlfreien Datei auszuführen oder OPEN mit einem Dateimodus außer "I", "O" oder "R" anzuwenden.

55 File already open — Datei bereits geöffnet
Ein OPEN mit sequentieller Ausgabe oder KILL wird für eine Datei, die bereits geöffnet ist, angelaufen.

Code	Nummer	Meldung	
	57	Disk I/O error	Plattenein-/-ausgabe-fehler

Ein Ein-/Ausgabefehler wurde während einer Ein-/Ausgabetätigkeit entdeckt. Dies ist ein katastrophaler Fehler, aus dem das Betriebssystem keinen Ausweg findet.

	58	File already exists	Datei bereits vorhanden

Der neue Dateiname in einer NAME-Anweisung existiert bereits auf der Platte.

	61	Disk full	Platte voll

Der gesamte Plattenspeicherplatz ist aufgebraucht.

	62	Input past end	Eingabe hinter Dateiende

Eine INPUT-Anweisung wird ausgeführt, nachdem alle Daten in der Datei gelesen wurden bzw. bei einer leeren Datei. Um diesen Fehler zu vermeiden, kann die EOF-Funktion genutzt werden, die das Dateiende angibt.

	63	Bad record number	Falsche Satznummer

In einer PUT oder GET-Anweisung ist die Satznummer entweder größer als das Maximum (32767) oder Null.

	64	Bad file name	Falscher Dateiname

Eine unerlaubte Form des Dateinamens (z.B. zuviele Zeichen) taucht in einer LOAD-, SAVE-, KILL- oder OPEN-Anweisung auf.

	66	Direct statement file	Direktbefehl in der Datei

Ein Direktbefehl wird entdeckt während ein Programm im ASCII-Format geladen wird. LOAD wird abgebrochen.

	67	Too many files	Zu viele Dateien

Es wird versucht, eine neue Datei zu eröffnen (mit SAVE oder OPEN), obwohl alle 255 Verzeichniseinträge voll sind.

ANHANG B

MATHEMATISCHE FUNKTIONEN

ABGELEITETE FUNKTIONEN

Funktionen, die nicht schon in MS-BASIC enthalten sind, können wie folgt berechnet werden:

FUNKTION	MS-BASIC AUSDRUCK
SEKANTE	SEC(X) = 1/COS(X)
KOSEKANTE	CSC(X) = 1/SIN(X)
KOTANGENS	COT(X) = 1/TAN(X)
INVERTIERTER SINUS	ARCSIN(X) = ATN(X/SQR(–X*X+1))
INVERTIERTER KOSINUS	ARCCOS(X)=–ATN(X/SQR(–X*X+1)) +1.5708
INVERTIERTE SEKANTE	ARCSEC(X) = ATN(X/SQR(X*X–1)) + SGN(SGN(X)–1)*1.5708
INVERTIERTE KOSEKANTE	ARCCSC(X) = ATN(X/SQR(X*X–1)) +(SGN(X)–1)*1.5708
INVERTIERTER KOTANGENS	ARCCOT(X) = ATN(X)+1.5708
HYPERBOLISCHER SINUS	SINH(X) = (EXP(X)–EXP(–X))/2
HYPERBOLISCHER KOSINUS	COSH(X) = (EXP(X)+EXP(–X))/2
HYPERBOLISCHER TANGENS	TANH(X) = EXP(–X)/(EXP(X)+EXP(–X))*2+1
HYPERBOLISCHE SEKANTE	SECH(X) = 2/(EXP(X)+EXP(–X))
HYPERBOLISCHE KOSEKANTE	CSCH(X) = 2/(EXP(X)–EXP(–X))
HYPERBOLISCHER KOTANGENS	COTH(X)=(EXP(–X)/(EXP(X)–EXP(–X))*2+1
INVERTIERTER HYPERBOLISCHER SINUS	ARCSINH(X) = LOG(X+SQR(X*X+1))
INVERTIERTER HYPERBOLISCHER KOSINUS	ARCCOSH(X) = LOG(X+SQR(X*X–1))
INVERTIERTER HYPERBOLISCHER TANGENS	ARCTANH(X) = LOG((1+X)/(1–X))/2
INVERTIERTE HYPERBOLISCHE SEKANTE	ARCSECH(X)=LOG((SQR(–X*X+1)+1)/X)
INVERTIERTE HYPERBOLISCHER KOSEKANTE	ARCCSCH(X)=LOG(SGN(X)*SQR (X*X+1)/X)
INVERTIERTER HYPERBOLISCHER KOTANGENS	ARCCOTH(X) = LOG ((X+1)/(X–1))/2

ANHANG C

ASCII-ZEICHENKODES

ASCII CODE	Zeichen allg. deutsch	ASCII Code	Zeichen allg. deutsch	ASCII Code	Zeichen allg. deutsch
000	NUL	043	+	086	V
001	SOH	044	,	087	W
002	STX	045	–	088	X
003	ETX	046	.	089	Y
004	EOT	047	/	090	Z
005	ENQ	048	0	091	[Ä
006	ACK	049	1	092	\ Ö
007	BEL	050	2	093	] Ü
008	BS	051	3	094	^
009	HT	052	4	095	<
010	LF	053	5	096	'
011	VT	054	6	097	a
012	FF	055	7	098	b
013	CR	056	8	099	c
014	SO	057	9	100	d
015	SI	058	:	101	e
016	DLE	059	;	102	f
017	DC1	060	<	103	g
018	DC2	061	=	104	h
019	DC3	062	>	105	i
020	DC4	063	?	106	j
021	NAK	064	@ §	107	k
022	SYN	065	A	108	l
023	ETB	066	B	109	m
024	CAN	067	C	110	n
025	EM	068	D	111	o
026	SUB	069	E	112	p
027	ESCAPE	070	F	113	q
028	FS	071	G	114	r
029	GS	072	H	115	s
030	RS	073	I	116	t
031	US	074	J	117	u
032	Leerschritt	075	K	118	v
033	!	076	L	119	w
034	"	077	M	120	x
035	#	078	N	121	y
036	$	079	O	122	z
037	%	080	P	123	{ ä
038	&	081	Q	124	| ö
039	'	082	R	125	} ü
040	(	083	S	126	~
041	)	084	T	127	DEL
042	*	085	U		

ASCII-Codes sind dezimal angegeben.
LF = Zeilenschaltung, FF = Formularaufschaltung, CR = Eingabetaste,
DL = Löschen.

ANHANG D

RESERVIERTE WORTE FÜR MS-BASIC

MS-BASIC verwendet nachstehende reservierte Worte:

ABS	ERASE	LOF	RIGHT$
AND	ERL	LOG	RND
ASC	ERR	LPOS	RSET
ATN	ERROR	LPRINT	RUN
AUTO	END	LSET	SAVE
CALL	EXP	MERGE	SBN
CDBL	FIELD	MID$	SIN
CHAIN	FILES	SMKD$	SPACE
CHR$	FIX	MKI$	SPC
CINT	FOR	MKS$	SQR
CLEAR	FRE	MOD	STOP
CLOSE	GET	NAME	STR$
COMMON	GOSUB	NEW	STRING$
CONT	HEX$	NOT	SWAP
COS	IF	OCT$	SYSTEM
CSNG	IMP	ON	TAB
CVD	INP	OPENON	TAN
CVI	INPUT	OPTION	THEN
CVS	INKEY$	OR	TO
DATA	INPUT=	PEEK	TROFF
DEFDBL	INPUT$	POKE	TRON
DEFINT	INSTR	POS	
DEFSNG	INT	PPRINT	USR
DEFSTR	KILL	PRINT=USING	VAL
DEF FN	LEFT$	PUT	VARPTR
DEF USR	LEN	RANDOMIZE	WAIT
DELETE	LET	READ	WEND
DIM	LINE	REM	WHILE
EDIT	LIST	RENUM	WRITE
ELSE	LLIST	RESET	WRITE =
END	LOAD	RESTORE	XOR
EOF	LOC	RESUME	

NCR

NCR DECISION MATE V

MS™-DOS - Erweiterung

MS-DOS ERWEITERUNG

INHALT

1. MS-BASIC MIT DEM MS-DOS BETRIEBSSYSTEM

Starten von MS-BASIC 1-1
Beenden von MS-BASIC 1-3
Diskettendateien 1-4

2. UNTERSCHIEDE DER MS-BASIC SPRACHE BEI MS-DOS 2-1

BLOAD-ANWEISUNG 2-3
BSAVE-ANWEISUNG 2-5
CALL-ANWEISUNG 2-7
CALLS-ANWEISUNG 2-9
CLOAD-ANWEISUNG 2-10
COMMON-ANWEISUNG 2-11
CSAVE-ANWEISUNG 2-12
DATE$-FUNKTION 2-13
DEF SEG-ANWEISUNG 2-14
EOF-FUNKTION 2-15
FILES-ANWEISUNG 2-17
INP-FUNKTION 2-18
LOF-FUNKTION 2-19
OPEN-ANWEISUNG 2-20
OUT-ANWEISUNG 2-22
RESET-ANWEISUNG 2-23
TIME$-FUNKTION 2-24
TIME$-ANWEISUNG 2-25
USR-FUNKTION 2-26

3. UMWANDLUNG VON PROGRAMMEN IN MS-BASIC

Dimensionen der Zeichenfolgen 3-1

Mehrfachanweisungen 3-2
MAT-Funktionen 3-2

4. HANDHABUNG VON DISKETTENDATEIEN

Programmdatei-Befehle 4-1
Geschützte Dateien 4-3
Datendateien auf Diskette 4-3
Sequentielle Dateien 4-3
Erstellen einer sequentiellen Datei 4-4
Zugriff zu einer sequentiellen Datei 4-6
Hinzufügen von Daten zu einer sequentiellen Datei 4-6
Direktzugriffsdateien 4-7
Erstellen einer Direktzugriffsdatei 4-7
Zugriff auf eine Direktzugriffsdatei 4-9

5. SUBROUTINEN IN DER ASSEMBLERSPRACHE

Zuweisung des Speicherplatzes 5-1
CALL-Anweisung 5-2
CALLS-Anweisung 5-6
USR-Funktion 5-6

A. MS-DOS FEHLERMELDUNGEN A-1

KAPITEL 1

MS-BASIC MIT DEM MS-DOS BETRIEBSSYSTEM

Die MS-DOS Version von MS-BASIC wird auf einer Diskette mit doppelter Schreibdichte geliefert. Der Name der Datei lautet BASIC86. Die Mindestspeicherkapazität für das System beträgt 128K. Dabei wird von 36K für MS-BASIC und 30K für MS-DOS ausgegangen, so daß 62K für Benutzerprogramme übrig bleiben.

HINWEIS: Vor der Benutzung von MS-BASIC müssen Sicherheitskopien der MS-BASIC Diskette mit dem DISKCOPY-Befehl von MS-DOS angefertigt werden. (Für weitere Informationen über den DISKCOPY-Befehl wird auf Kapitel 5 „MS-DOS Befehle" in dem MS-DOS Benutzerhandbuch verwiesen). Die Master-Diskette wird an einem sicheren Platz aufbewahrt. Gearbeitet wird mit der Sicherheitskopie.

STARTEN VON MS-BASIC

Für das Starten von MS-BASIC wird zuerst MS-DOS gestartet und danach der folgende Befehl eingegeben:

BASIC86< Rückführtaste>

Das System antwortet mit folgender Anzeige:

Darüber hinaus kann ein Satz von Optionen in der Befehlszeile angegeben werden. Die Syntax lautet:

BASIC86 [<Dateiname>] [/F:<Anzahl von Dateien>]
[/M:<Höchste Speicheradresse>] [/S:<Maximale Satzgröße>]

Wird <Dateiname> angegeben, so fährt MS-BASIC so fort, als wäre nach Beendigung des Startverfahrens ein RUN <Dateiname> Befehl eingegeben worden. Die Standarderweiterung .BAS wird benutzt, wenn keine andere Erweiterung angegeben wird, und wenn der Dateiname eine Länge von weniger als neun Zeichen aufweist.

Mit der Option <Dateiname> können MS-BASIC Programme im Batch-Modus ausgeführt werden, wobei die BATCH-Einrichtung in MS-DOS benutzt wird. Derartige Programme müssen eine SYSTEM-Anweisung enthalten, um nach Beendigung in MS-DOS zurückzukehren, so daß das nächste Programm in dem Batch-Strom ausgeführt werden kann. Für weitere Informationen über die Benutzung der BATCH-Einrichtung wird auf die Kapitel 4 und 5 des MS-DOS Benutzerhandbuchs verwiesen.

Mit /F: <Anzahl von Dateien> wird die Anzahl von Diskettendateien festgelegt, die während der Ausführung eines MS-BASIC Programms gleichzeitig eröffnet sein können. Jeder in dieser Gruppe zugeordnete Datenblock der Datei wird mit /S: festgelegt (die Standardgröße für /S: beträgt 128 Bytes). Wird die Option /F weggelassen, so beträgt die Anzahl von Dateien standardmäßig 3.

/M: <Höchste Speicheradresse> legt die höchste Speicheradresse fest, die von MS-BASIC benutzt wird.

/S: <Maximale Satzgröße> legt die maximale Satzgröße zur Benutzung mit Direktzugriffsdateien fest. Die Standardsatzgröße beträgt 128 Bytes.

HINWEIS: <Anzahl von Dateien>, <Höchste Speicheradresse> und <Maximale Satzgröße> können in Dezimalform, in Oktalform oder Hexadezimalform angegeben werden, wobei bei der Oktalform &0 und bei der Hexadezimalform &H vorangestellt wird.

Beispiele für Befehlszeilen beim Starten:

BASIC86 PAYROLL.BAS	Benutzung des gesamten Speicherplatzes und Benutzung von drei Dateien, Laden und Ausführen von PAYROLL.BAS.

BASIC 86 INVENT/F:6	Benutzung des gesamten Speicherplatzes und Benutzung von sechs Dateien, Laden und Ausführen von INVENT.BAS.
BASIC86/M:32768	Benutzung der ersten 32K des Speicherplatzes und Benutzung von drei Dateien.
BASIC86 DATACK/F:2/M:&H9000	Benutzung der ersten 36K der Speicherkapazität und von zwei Dateien, sowie Ausführung von DATACK.BAS.

BEENDEN VON MS-BASIC

Für das Beenden von MS-BASIC und die Rückkehr zu dem MS-DOS Betriebssystem wird der Befehl:

SYSTEM

eingegeben, der sämtliche Dateien abschließt und MS-DOS wieder in den Speicher lädt, ohne bestehende Programme oder Daten zu löschen (d.h. es wird ein sogenannter „Warmstart" ausgeführt). Wird Control-C mit MS-BASIC benutzt, so wird zu der Befehlsebene von MS-BASIC und nicht zu MS-DOS zurückgekehrt.

DISKETTENDATEIEN

Diskettendateien unterliegen den normalen MS-DOS Regeln für die Benennung von Dateien. Sämtliche Dateinamen können mit einer Laufwerkbezeichnung von <Buchstabe>: (z.B. A:) beginnen. Wird kein Laufwerk angegeben, so wird vom aktuellen Laufwerk ausgegangen. Die Standarderweiterung .BAS wird mit LOAD, SAVE, MERGE und RUN <Dateiname> Befehlen benutzt, wenn in dem Dateinamen kein Punkt (.) steht und wenn der Dateiname eine Länge von weniger als neun Zeichen hat.

Beispiele:

RUN "NEWFILE.EXE"
RUN "A:NEWFILE.EXE"
SAVE "NEWFILE" (die Datei wird mit der Erweiterung .BAS gesichert).

Umfangreiche Direktzugriffsdateien werden unterstützt. Die Höchstzahl von logischen Sätzen beträgt 32767. Wird eine Satzgröße von 256 angegeben, so kann auf Dateien mit einer Kapazität von bis zu 8 MB zugegriffen werden.

KAPITEL 2

UNTERSCHIEDE DER MS-BASIC SPRACHE BEI MS-DOS

Die folgenden Einrichtungen unterscheiden sich in dieser Implementierung von der Beschreibung in dem MS-BASIC Handbuch. Die Schreibweise der Syntax bleibt dieselbe:

[ ] Eckige Klammern geben an, daß der in ihnen stehende Eintrag wahlweise ist.

< > Spitze Klammern geben vom Benutzer eingegebene Daten an. Wird in den spitzen Klammern ein Text in Kleinbuchstaben angegeben, so muß der Benutzer einen von dem Text definierten Eintrag eingeben, z.B. <Dateiname>. Enthalten die spitzen Klammern einen Text in Großbuchstaben, so muß der Benutzer die von dem Text angegebene Taste betätigen, z.B.<RETURN> (Abschlußtaste).

{ } Geschweifte Klammern geben an, daß der Benutzer zwischen zwei oder mehr Eingaben wählen kann. Mindestens eine der in geschweiften Klammern stehenden Eingaben muß ausgewählt werden, es sei denn, die Eingaben stehen selbst in eckigen Klammern.

| Senkrechte Trennstriche trennen die in geschweiften Klammern stehenden Auswahlmöglichkeiten. Mindestens einer der von senkrechten Trennstrichen getrennten Einträge muß ausgewählt werden, es sei denn, die Einträge stehen selbst in eckigen Klammern.

... Auslassungszeichen geben an, daß ein Eintrag beliebig oft wiederholt werden kann.

GROSS Großbuchstaben geben die Teile von Anweisungen oder Befehlen an, die genau wie dargestellt eingegeben werden müssen.

Alle anderen Satzzeichen wie Kommas, Doppelpunkte, Schrägstriche und Gleichzeichen müssen genau wie dargestellt eingegeben werden.

BLOAD-ANWEISUNG

Mit der BLOAD-Anweisung können ein Programm oder Daten, die als Speicherabbilddatei gesichert wurden, an einer beliebigen Stelle im Speicher geladen werden. Eine Speicherabbilddatei ist eine byteweise Kopie des ursprünglichen Speicherinhalts. Für Informationen über das Sichern von Speicherabbilddateien wird auf die „BSAVE-Anweisung" verwiesen.

BLOAD wird häufig für das Laden von Programmen in der Assemblersprache benutzt, ist jedoch nicht auf diesen Einsatz beschränkt. Mit ihr können beispielsweise auch kompilierte Microsoft Pascal- oder Microsoft FORTRAN-Routinen geladen werden.

Syntax BLOAD<Dateiname> [,<Abstand>]

Wobei <Dateiname> einen Zeichenfolgenausdruck mit der Einheitenbezeichnung und dem Dateinamen darstellt. (Die Einheitenbezeichnung ist wahlweise).

<Abstand> ist ein numerischer Ausdruck, mit dem eine vorzeichenlose Ganzzahl in dem Bereich zwischen 0 und 65.535 zurückgegeben wird. Hier handelt es sich um die Abstandsadresse in dem von der letzten DEF SEG-Anweisung deklarierten Segment, bei der das Laden beginnen muß (siehe „DEF SEG-Anweisung").

Zweck Laden der angegebenen Speicherabbilddatei in den Speicher.

Bemerkungen BLOAD unterliegt den folgenden Regeln:

1. Wird die Einheit weggelassen, so wird vom aktuellen Laufwerk ausgegangen.
2. Wird der Abstand weggelassen, so werden die in der Datei angegebene Segmentadresse und der angegebene Abstand benutzt (d.h. die Adresse, die beim Erstellen der Datei von der BSAVE-Anweisung angegeben wurde). Deshalb wird die Datei in dieselbe Adresse geladen, aus der sie gesichert wurde.
3. Wird der Abstand angegeben, so handelt es sich bei der benutzten Segmentadresse um die in der

zuletzt ausgeführten DEF SEG-Anweisung angegebene Adresse. Deshalb muß eine DEF SEG-Anweisung vor der BLOAD-Anweisung ausgeführt werden.

ACHTUNG: BLOAD nimmt keine Überprüfung des Adressenbereichs vor. Deshalb kann eine Datei an eine beliebige Stelle im Speicher geladen werden. Der Benutzer muß darauf achten, daß er die Datei nicht über MS-BASIC oder dem Betriebssystem lädt.

Beispiele:

```
10 'Load subroutine at 6000:F000
20 DEF SEG =&H600 'Set segment to 6000 Hex
30 BLOAD "PROG1",&HF000 'Load PROG1
```

Dieses Beispiel setzt die Segmentadresse auf 6.000 Hex und lädt PROG1 bei F000.

Da sämtliche Versionen von MS-BASIC die BLOAD-Anweisung unterstützen, wird sie in dem MS-BASIC Handbuch nicht besprochen.

BSAVE-ANWEISUNG

Mit der BSAVE-Anweisung können Daten oder Programme als Speicherabbilddateien auf Diskette gesichert werden. Eine Speicherabbilddatei ist eine byteweise Kopie des Speicherinhalts. BASVE wird häufig für das Sichern von Programmen in der Assemblersprache benutzt, kann jedoch auch mit in anderen Sprachen geschriebenen Daten oder Programmen benutzt werden.

Syntax BSAVE<Dateiname>,<Abstand>,<Länge>

Wobei <Dateiname> einen Zeichenfolgenausdruck mit der Einheitenbezeichnung und dem Dateinamen darstellt. (Die Einheitenbezeichnung ist wahlweise).

<Abstand> ist ein numerischer Ausdruck, der eine vorzeichenlose Ganzzahl in dem Bereich von 0 bis 65.535 zurückgibt. Hier handelt es sich um die Abstandsadresse, ab der mit dem Sichern in dem Segment begonnen wird, das von der zuletzt ausgeführten DEF SEG-Anweisung deklariert wurde.

<Länge> ist eine numerischer Ausdruck, der eine vorzeichenlose Ganzzahl in dem Bereich von 1 bis 65.535 zurückgibt. Hier handelt es sich um die Länge der zu sichernden Speicherabbilddatei in Bytes.

Zweck Sichern des Inhalts des angegebenen Speicherbereichs als Diskettendatei.

Bemerkungen <Dateiname>,<Abstand> und <Länge> müssen in der Syntax angegeben werden.

Vor der BASVE-Anweisung muß eine DEF SEG-Anweisung ausgeführt werden. Die in der DEF SEG-Anweisung angegebene Adresse wird für das Sichern benutzt.

Beispiel:

```
10 'Save PROG1
20 DEF SEG=&H6000
30 BSAVE "PROG1",&HF000,256
```

In diesem Beispiel werden 256 Bytes gesichert, wobei mit der Adresse 6000:F000 in der Datei PROG1 begonnen wird.

Die BSAVE-Anweisung wird in dem MS-BASIC Handbuch nicht besprochen.

CALL-ANWEISUNG

Die CALL-Anweisung wird für die Verbindung von Programmen in der Maschinensprache mit MS-BASIC benutzt. Die nachfolgende Beschreibung stellt einen Zusatz zu der Beschreibung in dem MS-BASIC Handbuch dar.

Syntax CALL<Variablenname> [(<Parameterliste>)]

Wobei <Variablenname> Die Segmentposition enthält, die den Anfangspunkt der aufgerufenen Subroutine im Speicher darstellt. Hier muß beachtet werden, daß die Segmentposition der Variablen zugeordnet werden muß, bevor die CALL-Anweisung ausgegeben wird (siehe nachfolgendes Beispiel).

<Parameterliste> enthält die durch Kommas voneinander getrennten Variablen oder Konstanten, die an die Routine übergeben werden müssen.

Enthält diese Liste eine Matrixvariable, auf die eine nicht definierte einfache Variable folgt, so kommt es zu einer Fehlermeldung "Illegal Function Call" (Unzulässiger Funktionsaufruf). Dieser Fehler kann vermieden werden, indem sämtlichen einfachen Variablen vor der Ausführung der CALL-Anweisung ein Wert zugeordnet wird.

Zweck Aufruf einer Subroutine in der Assemblersprache.

Bemerkungen Die Benutzung der CALL-Anweisung wird für den Aufruf von 8086 Programmen in der Maschinensprache mit MS-BASIC empfohlen. Wir empfehlen, daß Sie die USR-Funktion nicht benutzen. Für einen Vergleich der beiden Methoden und für eine genaue Beschreibung der Benutzung der CALL-Anweisung für Subroutinen in der Assemblersprache wird auf Kapitel 5 „Subroutinen in der MS-BASIC Assemblersprache" verwiesen.

Bei der Ausführung einer CALL-Anweisung wird die Kontrolle über die in der letzten DEF SEG-Anweisung angegebene Segmentadresse und über

die in dem Teil <Variablenname> der CALL-Anweisung angegebenen Segmentposition an die Benutzerroutine übergeben. Die Werte werden an BASIC zurückgegeben, indem der Variablenname, der das Ergebnis aufnimmt, in die <Parameterliste> eingefügt wird.

Beispiel

```
100 DEF SEG=&H8000
110 F00=&H7FA
120 CALL F00(A,B$,C)
 .
 .
 .
```

Mit Zeile 100 wird die Segmentadresse auf 8000 Hex festgelegt. Die Variable F00 wird auf &H7FA festgelegt, so daß durch Aufruf von F00 die Subroutine in der Adresse 8000:7FA ausgeführt wird (dies entspricht der absoluten Hexadezimaladresse 807FA).

CALLS-ANWEISUNG

Die CALLS-Anweisung entspricht genau der CALL-Anweisung, nur daß die segmentierten Adressen sämtlicher Parameter übergeben werden. (CALL übergibt nicht segmentierte Adressen). CALLS muß beim Zugriff auf MS-FORTRAN Routinen benutzt werden, da es sich bei sämtlichen MS-FORTRAN Parametern um sogenannte segmentierte "Call-by-reference" Adressen handelt.

Wie bei der CALL-Anweisung benutzt CALLS die Segmentadresse, die von der zuletzt ausgeführten DEF SEG-Anweisung definiert wurde, um die aufgerufene Routine zu finden.

Die CALLS-Anweisung ist in dem MS-BASIC Handbuch nicht enthalten.

CLOAD-ANWEISUNG

Die CLOAD-Anweisung wird bei MS-BASIC mit MS-DOS nicht unterstützt.

COMMON-ANWEISUNG

Wie unter „COMMON" in dem MS-BASIC Handbuch angegeben, werden Matrixvariablen in einer COMMON-Anweisung angegeben, indem an den Variablennamen Klammern angehängt werden. Da bei anderen Microsoft-Produkten die Anzahl von Dimensionen in der Matrix in der Anweisung aufgenommen werden können, akzeptiert der MS-BASIC Interpreter diese Syntax, ignoriert jedoch den numerischen Ausdruck selbst. So sind beispielsweise die folgenden Anweisungen beide gültig und werden als gleichbedeutend angesehen:

```
COMMON A( )
COMMON A(3)
```

Die in Klammern angegebene Zahl entspricht der Anzahl von Dimensionen und nicht den Dimensionen selbst. So könnte die Variable A(3) in diesem Beispiel einer DIM-Anweisung mit DIM A(5,8,4) entsprechen.

CSAVE-ANWEISUNG

Die SCAVE-Anweisung wird bei MS-BASIC mit MS-DOS nicht unterstützt.

DATE$-FUNKTION

Mit der DATE$-Funktion wird das aktuelle Datum wieder aufgerufen. Diese Funktion ist in dem MS-BASIC Handbuch nicht enthalten.

Syntax DATE$

Zweck Wiederaufrufen des aktuellen Datums. (Für das Festlegen des Datums wird die im nächsten Abschnitt beschriebene DATE$-Anweisung benutzt.)

Bemerkungen Die DATE$-Funktion gibt eine aus zehn Zeichen bestehende Zeichenfolge in dem Format mm-dd-yyyy zurück, wobei mm dem Monat (01 bis 12), dd dem Tag (01 bis 31) und yyyy dem Jahr (1980 bis 2099) entspricht.

Beispiel 10 PRINT DATE$

Die DATE$-Funktion druckt das Datum aus, das anhand des mit der DATE$-Anweisung festgelegten Datums berechnet wird.

DEF SEG-ANWEISUNG

Syntax DEF SEG[=<Adresse>]

Wobei Adresse einen numerischen Ausdruck darstellt, der eine vorzeichenlose Ganzzahl in dem Bereich von 0 bis 65.535 zurückgibt.

Zweck Zuweisung der aktuellen Segmentadresse, auf die von einer nachfolgenden CALL-, CALLS- oder POKE-Anweisung oder von einer USR- oder PEEK-Funktion bezug genommen werden soll.

Bemerkungen Die angegebene Adresse wird zur Benutzung als Segment gesichert, das von CALL, CALLS, POKE, usr und PEEK benötigt wird.

Wird ein Wert außerhalb des Adressenbereichs von 0 bis 65.535 eingegeben, so kommt es zu einer Fehlermeldung "Illegal Function Call" (Unzulässiger Funktionsaufruf), und der vorhergehende Wert wird festgehalten.

Wird die Option <Adresse> weggelassen, so wird das zu benutzende Segment auf das MS-BASIC Datensegment (DS) festgesetzt. Hier handelt es sich um den ursprünglichen Standardwert.

Wird die Option <Adresse> angegeben, so muß sie auf einer 16-Byte-Grenze beruhen. MS-BASIC überprüft die Gültigkeit der angegebenen Adresse nicht.

HINWEIS DEF und SEG müssen durch ein Leerzeichen voneinander getrennt werden. Ansonsten interpretiert MS-BASIC die Anweisung DEFSEG=100 als „Zuweisung des Wertes 100 an die Variable DEFSEG".

Beispiel
```
10 DEF SEG=&HB800 'Set segment to & 800 Hex
20 DEF SEG 'Restore segment to MS-BASIC data
                segment
```

Die DEF SEG-Anweisung ist in dem MS-BASIC Handbuch nicht enthalten.

EOF-FUNKTION

Die EOF-Funktion kann mit Direktzugriffsdateien und sequentiellen Dateien gleichermaßen benutzt werden. Die nachfolgende Beschreibung stellt eine Ergänzung der Beschreibung in dem MS-BASIC Handbuch dar.

Syntax EOF (< Anzahl von Dateien>)

Zweck Bei sequentiellen Dateien gibt die EOF-Funktion „wahr" (-1) zurück, wenn keine weiteren Daten mehr in der Datei stehen. Die Datei ist leer, wenn die nächste Eingabeoperation (z.B. INPUT, LINE INPUT, INPUT$) zu einer Fehlermeldung "Input past end" (Eingabe über das Ende hinaus) führen würde.

Bei Direktzugriffsdateien gibt die EOF-Funktion „wahr" (-1) zurück, wenn die zuletzt ausgeführte GET-Anweisung versucht, über das Dateiende hinaus zu lesen.

Bemerkungen Da MS-BASIC einer Datei jeweils 128 Bytes zuordnet, ist es möglich, daß EOF das Ende einer Direktzugriffsdatei, die mit einer Satzlänge von weniger als 128 Bytes eröffnet wurde, nicht genau erkennt. Wird eine Datei beispielsweise mit einer Satzlänge von 64 Bytes eröffnet und wird ein Satz in die Datei geschrieben (z.B. PUT# 1,1), so gibt EOF „falsch" zurück, wenn versucht wird, eine GET-Anweisung mit dem zweiten Satz der Datei auszuführen (z.B. GET# 1,2). Dies geschieht, obwohl in diesem Satz nicht wirklich geschrieben wurde.

Beispiel

```
10 REM
20 REM Eröffnen der Bibliotheksverzeichnisdatei
30 REM LIBRARY.DAT.
40 OPEN "R", #1, "LIBRARY.DAT"
50 REM Die ersten 35 Bytes des
60 REM Satzes enthalten den Titel,
70 REM die restlichen 93 Bytes enthalten
80 REM zusätzliche Informationen, die
90 REM von diesem Programm nicht benutzt wer-
       den.
```

```
100 FIELD+1,35 AS TITLE$,93 AS G$
110 REM
120 REM Initialisierung der festgestellten Anzahl
        von Büchern
130 REM
140 NBOOKS=0
150 REM Versuch, den nächsten Satz zu holen.
160 REM Beachten, daß der Parameter mit der
170 REM Satznummer von GET nicht enthalten ist,
180 REM so daß der nächste Satz der
190 REM Datei aufgerufen wird.
200 GET#1
210 REM
220 REM Ist dies das Ende der Datei?
230 REM
240 IF EOF(1) THEN 1000
250 REM Nein, Anzahl von Büchern erhöhen,
260 REM den aktuellen Titel ausdrucken und
270 REM zurückgehen, um den nächsten
280 REM Satz zu lesen.
290 NBOOKS=NBOOKS+1
300 PRINT TITLE$
310 GOTO 200
1000 REM Hier, wenn das Ende der Datei
1010 REM erreicht wurde. Eine Leerzeile
1020 REM und die Anzahl von Büchern drucken,
1030 REM die Datei abschließen und das Programm
         beenden.
1040 PRINT "Es sind ";NBOOKS;" Bücher in";
1050 PRINT "Ihrer Bibliothek vorhanden."
1060 CLOSE
1070 END
```

Dieses Beispielprogramm listet die Titel der Bücher auf, die in der Datei LIBRARY.DAT verzeichnet sind. Außerdem zählt es die Bücher in der Bibliothek, indem es die Anzahl von Sätzen zählt, die aus der Datei LIBRARY.DAT gelesen werden, bevor das Dateiende angetroffen wird.

Jeder Satz von LIBRARY.DAT enthält Informationen über ein Buch in der Bibliothek. Die Satzlänge beträgt 129 Bytes. Die ersten 35 Bytes enthalten den Titel des Buches; die restlichen 93 Bytes enthalten zusätzliche Informationen über das Buch (z.B. Autor, Verleger, Ort usw.). Diese Informationen werden in diesem Beispiel nicht benutzt.

FILES-ANWEISUNG

Syntax FILES[<Dateiname>]

Wobei <Dateiname> einen Dateinamen und eine wahlweise Einheitenbezeichnung enthält.

Zweck Ausdrucken der Namen der Dateien, die auf der angegebenen Diskette stehen.

Bemerkungen Wird <Dateiname> weggelassen, so werden sämtliche Dateien in dem gerade benutzten Laufwerk aufgelistet. Dateiname ist ein Zeichenfolgenausdruck, der Fragezeichen (?) oder Sternchen (*) als Dateigruppensymbole enthalten kann. Ein Fragezeichen entspricht jedem einzelnen Zeichen in dem Dateinamen oder der Erweiterung. Ein Sternchen entspricht einem oder mehreren Zeichen ab dieser Position. Das Sternchen ist eine Abkürzung für eine Reihe von Fragezeichen.

Beispiele
```
FILES
FILES "*,BAS"
FILES "B:*.*"
FILES "B: (entspricht "B:*.*")
FILES "TEST?.BAS"
```

Die FILES-Anweisung ist in dem MS-BASIC Handbuch nicht enthalten.

INP-FUNKTION

Wird MS-BASIC mit DM-DOS benutzt, so geht der Bereich der gültien Anschlußnummern von 0 bis 65.535 und nicht von 0 bis 255, wie in dem MS-BASIC Handbuch angegeben.

Syntax INP(I)

Wobei I eine gültige Anschlußnummern in dem Bereich von 0 bis 65.535 darstellt.

Zweck Rückgabe des von Anschlußposition I gelesenen Bytes.

Bemerkungen INP ist die Ergänzungsfunktion zu der OUT-Anweisung.

Beispiel 100 A=INP(54321)

In der Assemblersprache entspricht dies:

```
MOV  DX,54321
IN   AL,DX
```

LOF-FUNKTION

Da die LOF-Funktion nicht in sämtlichen Implementierungen von MS-BASIC enthalten ist, ist sie auch in dem MS-BASIC Handbuch nicht enthalten.

Syntax LOF (<Dateinummer>)

Zweck Rückgabe der Länge der Datei in Bytes.

Beispiel

```
110 IF REC*RECSIZ>LOF(1)
    THEN PRINT "INVALID ENTRY"
```

In diesem Beispiel enthalten die Variablen REC und RECSIZ die Anzahl von Sätzen bzw. die Satzlänge. Mit der Berechnung wird bestimmt, ob der angegebene Satz über das Dateiende hinausgeht.

OPEN-ANWEISUNG

Bei dieser Implementierung von MS-BASIC ist eine zusätzliche Form der OPEN-Anweisung verfügbar. Neben der Unterstützung der sequentiellen Eingabe, der sequentiellen Ausgabe und des Direktzugriffsmodus unterstützt diese neue Form der OPEN-Anweisung den APPEND-Modus, mit dem Daten an das Ende einer sequentiellen Datei angehängt werden können. Die neue Syntax wird nachfolgend erläutert. Die in dem MS-BASIC Handbuch beschriebene Syntax bleibt weiterhin gültig.

Syntax

OPEN <Dateiname> [FOR<Modus>] AS [#] <Dateinummer>
[LEN=<Satzlänge> /

Wobei <Modus> einer der folgenden Angaben entsprechen kann:

OUTPUT Gibt den sequentiellen Ausgabemodus an.

INPUT Gibt den sequentiellen Eingabemodus an.

APPEND Gibt den sequentiellen Ausgabemodus an, setzt den Dateizeiger an das Ende der Datei und gibt die Anzahl von Sätzen als letzten Satz der Datei ein. Mit einer PRINT# - oder WRITE# -Anweisung wird die Datei dann erweitert.

<Satzlänge> legt die Satzlänge für Direktzugriffsdateien fest. Diese Option darf nicht mit sequentiellen Dateien benutzt werden.

Die Satzlänge darf die mit /S: beim Starten festgelegte Grenze nicht überschreiten (die Standardvorgabe für /S: beträgt 128 Bytes). Wird die Option für die Satzlänge nicht benutzt, so beträgt die Standardlänge 128 Bytes.

Bemerkungen Wird <Modus> weggelassen, so wird standardmäßig vom Direktzugriffsmodus ausgegangen.

Beispiel 10 OPEN "MAILING.DAT" FOR APPEND AS 1

Nun können Daten an das Ende der Datei MAILING.DAT angehängt werden.

Alle anderen Parameter und Regeln entsprechen den Angaben in "OPEN" in dem MS-BASIC Handbuch.

OUT-ANWEISUNG

Bei dieser Implementierung von MS-BASIC geht der Bereich von gültigen Eingaben für die Anschlußposition von 0 bis 65.535 und nicht von 0 bis 255, wie in dem MS-BASIC Handbuch angegeben. Der Bereich für die Daten geht immer noch von 0 bis 255.

Syntax OUT I,J

Wobei I einen ganzzahligen Ausdruck in dem Bereich von 0 bis 65.535 darstellt. J ist ein ganzzahliger Ausdruck in dem Bereich von 0 bis 255. Bei I handelt es sich um die Nummer einer Anschlußposition und bei J um die zu übetrragenden Daten.

Zweck Senden eines Bytes an die Ausgabeanschlußposition des Rechners.

Bemerkungen OUT ist die Ergänzungsanweisung zu der INP-Funktion.

Beispiel 100 OUT 12345,255

In der Assemblersprache entspricht dies:

```
MOV DX,12345
MOV AL,255
OUT DX,AL
```

RESET-ANWEISUNG

Da RESET nicht für alle Implementierungen auf dieselbe Art und Weise benutzt wird, ist diese Anweisung in dem MS-BASIC Handbuch nicht enthalten.

Syntax RESET

Zweck Abschließen sämtlicher Dateien in allen Laufwerken.

Bemerkungen RESET schließt alle eröffneten Dateien in sämtlichen Laufwerken ab, und schreibt die Verzeichnisinformationen auf jede Diskette mit eröffneten Dateien.

Sämtliche Dateien müssen abgeschlossen werden, bevor eine Diskette aus dem Laufwerk herausgenommen wird.

TIME$-FUNKTION

Da diese Funktion nicht in allen Implementierungen enthalten ist, ist sie auch in dem MS-BASIC Handbuch nicht enthalten.

Syntax TIME$

Zweck Wiederaufrufen der aktuellen Zeit. (Für das Festlegen der Zeit wird die im nächsten Abschnitt beschriebene TIME$-Anweisung benutzt.)

Bemerkungen Die TIME$-Funktion gibt eine aus acht Zeichen bestehende Zeichenfolge in der Form hh:mm:ss zurück, wobei hh den Stunden (00 bis 23), mm den Minuten (00 bis 59) und ss den Sekunden (00-59) entspricht. Ein 24-Stunden-Takt wird benutzt, so daß die Eingabe von 8:00 p.m. als 20:00 angezeigt wird.

Beispiel 10 PRINT TIME$

Druck die Zeit aus, die aufgrund der mit der TIME$-Anweisung festgelegten Zeit berechnet wurde.

TIME$-ANWEISUNG

Die TIME$-Anweisung legt die aktuelle Zeit fest. Diese Anweisung ist in dem MS-BASIC Handbuch nicht enthalten.

Syntax TIME$=<Zeichenfolgenausdruck>

Wobei <Zeichenfolgenausdruck> eine Zeichenfolge in einer der folgenden Formen zurückgibt:

hh	(Legt die Stunden fest; für Minuten und Sekunden wird der Standardwert 00 benutzt).
hh:mm	(Legt die Stunden und Minuten fest; für die Sekunden wird der Standardwert 00 benutzt).
hh:mm:ss	(Legt die Stunden, Minuten und Sekunden fest).

Zweck Festlegen der Zeit für die TIME$-Funktion.

Bemerkungen Ein 24-Stunden-Takt wird benutzt, so daß die Eingabe von 8:00 p.m. als 20:00:00 angezeigt wird.

Beispiel 10 TIME$="08:00:00"

Die aktuelle Zeit wird auf 8:00 festgelegt.

USR-FUNKTION

Die USR-Funktionen können für den Aufruf von Subroutinen in der Assemblersprache benutzt werden, obwohl wir hierfür die CALL-Anweisung empfehlen. Diese Funktion wird hier für die Benutzer beschrieben, die mit vorher geschriebenen Programmen arbeiten, die USR-Funktionen enthalten.

Für einen Vergleich zwischen CALL und USR und für eine genaue Beschreibung des Aufrufs von Subroutinen in der Assemblersprache wird auf Kapitel 5 "Subroutinen in der MS-BASIC Assemblersprache" verwiesen.

USR wird auch in dem MS-BASIC-Handbuch beschrieben. Die vorliegende Beschreibung ist jedoch etwas genauer und enthält darüber hinaus einige Informationen, die für diese Implementierung von MS-BASIC spezifisch sind.

Syntax USR [<Ziffer>] [(Parameter)]

Wobei <Ziffer> die gerade aufgerufene USR-Routine angibt. Für die Regeln, denen <Ziffer> unterliegt, wird auf "DEF USR" in dem BASIC Handbuch verwiesen. Wird <Ziffer> weggelassen, so wird von USR0 ausgegangen. <Parameter> entspricht dem an die Subroutine übergebenen Wert. Hier kann es sich um einen beliebigen numerischen oder Zeichenfolgenausdruck handeln.

Zweck Aufruf einer Subroutine in der Assemblersprache.

Bemerkungen Wird bei dieser Implementierung ein anderes Segment als das Standardsegment (MS-BASIC Datensegment, DS) benutzt, so muß vor einem USR-Funktionsraufruf eine DEF SEG-Anweisung ausgeführt werden. Mit der in der DEF SEG-Anweisung angegebenen Adresse wird die Segmentadresse der Subroutine bestimmt.

Für jede USR-Funktion muß eine entsprechende DEF USR-Anweisung ausgeführt werden, um die USR-Aufrufposition zu definieren. Diese Position und die gerade aktive DEF SEG-Segmentadresse bestimmen die Anfangsadresse der Subroutine.

Beispiel

```
100 DEF SEG=&H8000
110 DEF USR0=0
120 X=5
130 Y=USR0(X)
140 PRINT Y
```

Der Typ der Variablen (numerisch oder Zeichenfolge), die den Funktionsaufruf aufnimmt, muß mit dem übergebenen Parameter konsistent sein.

KAPITEL 3

UMWANDLUNG VON PROGRAMMEN IN MS-BASIC

Bei Programmen, die in einer anderen BASIC-Version als MS-BASIC geschrieben sind, sind einige geringfügige Änderungen erforderlich, bevor sie mit MS-BASIC ausgeführt werden können. In diesem Kapitel werden einige grundlegende Dinge dargelegt, die beachtet werden müssen, wenn Programme in MS-BASIC umgewandelt werden.

DIMENSIONEN DER ZEICHENFOLGEN

Sämtliche Anweisungen, die zur Deklaration der Länge von Zeichenfolgen benutzt wurden, müssen gelöscht werden. Eine Anweisung wie DIM A$(I,J), mit der eine Zeichenfolgematrix für J Elemente mit der Länge I dimensioniert wird, muß in die MS-BASIC Anweisung DIM A$(J) umgewandelt werden.

Einige BASIC-Befehle benutzen ein Komma (,) oder kommerzielles Und (&) für die Verkettung von Zeichenfolgen. Jedes dieser Zeichen muß in ein Pluszeichen (+) geändert werden, den Operator für die Zeichenfolgenverkettung bei MS-BASIC.

In MS-BASIC nehmen die Funktionen MID$, RIGHT$ und LEFT$ Teilfolgen aus Zeichenfolgen heraus. Formen wie A$(I) (für den Zugriff zu dem I-ten Zeichen in A$) oder A$(I,J) (um eine Teilfolge von A$ von Position I in Position J zu bewegen) müssen folgendermaßen geändert werden:

Andere BASIC-Versionen	Microsoft BASIC
X$=A$(I)	M$=MID$(A$,I,1)
X$=A$(I,J)	X$=MID$(A$,I,J—I+1)

Steht die Bezugnahme auf eine Teilfolge auf der linken Seite einer Zuweisung und ersetzt X$ die Zeichen in A$, so muß die Referenz folgendermaßen umgewandelt werden:

Andere BASIC-Versionen	Microsoft BASIC
A$(I)=X$	MID$(A$,I,1)=X$
A$(I,J)=X$	MID$(A$,I,J—I+1)=X$

MEHRFACHZUWEISUNGEN

Einige BASIC-Versionen lassen Anweisungen in der Form:

```
10 LET B=C=0
```

zu, um B und C gleich Null zu setzen. MS-BASIC interpretiert das zweite Gleichzeichen als boolschen Operator und setzt B gleich -1, wenn C gleich Null ist. Diese Anweisung muß stattdessen in zwei Zuweisungsanweisungen umgewandelt werden:

```
10 C=0:B=0
```

MEHRFACHANWEISUNGEN

Einige BASIC-Versionen benutzen einen umgekehrten Schrägstrich (\), um mehrere Anweisungen auf einer Zeile voneinander zu trennen. Bei MS-BASIC müssen sämtliche Anweisungen auf einer Zeile durch einen Doppelpunkt (:) voneinander getrennt werden.

MAT-FUNKTIONEN

Programme, die die MAT-Funktionen benutzen, die in einigen BASIC-Versionen vorhanden sind, müssen mit FOR ... NEXT-Schleifen neu geschrieben werden, um ordnungsgemäß ausgeführt werden zu können.

KAPITEL 4

HANDHABUNG VON DISKETTENDATEIEN

In diesem Kapitel werden die Eingabe- und Ausgabeprozeduren auf Diskette für den BASIC-Anfänger erläutert. Ein BASIC-Anfänger oder ein Benutzer, der diskettenbezogene Fehlermeldungen erhält, sollte diese Verfahren und die Programmbeispiele durchlesen, um sicherzugehen, daß er sämtliche Diskettenanweisungen richtig benutzt.

Ist ein Dateiname in einem Diskettenbefehl oder einer Diskettenanweisung erforderlich, so muß ein Name benutzt werden, der den Regeln für die Benennung von Dateien bei dem jeweiligen Betriebssystem gerecht wird. Das MS-DOS Betriebssystem hängt an Dateinamen, die in einem SAVE-, RUN-, MERGE- oder LOAD-Befehl zugewiesen wurden, die Standarderweiterung .BAS an.

PROGRAMMDATEI-BEFEHLE

In der nachfolgenden Liste werden die Befehle und Anweisungen zusammengefaßt, die bei der Verarbeitung von Programmdateien benutzt werden.

HINWEIS: Ab der MS-BASIC Version 5.27 können das Sternchen (*) und Fragezeichen (?) als Dateigruppensymbole mit den FILES-, KILL- und NAME-Befehlen benutzt werden.

Dateigruppensymbole sollten stets mit größter Vorsicht benutzt werden. Es darf nicht außer Acht gelassen werden, daß der Befehl KILL *.* beispielsweise sämtliche Dateien in dem Verzeichnis löscht, und nicht nur die Datei mit dem Namen *.*. In Kapitel 3 des MS-DOS Benutzerhandbuchs wird die Benutzung von Dateigruppensymbolen beschrieben.

SAVE <Dateiname> [,A]

Schreibt das gerade im Speicher stehende Programm

auf Diskette. Durch die Option A wird das Programm als eine Folge von ASCII-Zeichen geschrieben. (Ansonsten benutzt MS-BASIC ein komprimiertes Binärformat.)

(Für eine Beschreibung des Sicherns von geschützten Dateien wird auf Abschnitt 4.2 "Geschützte Dateien" verwiesen.)

LOAD <Dateiname> [,R]

Lädt das Programm von der Diskette in den Speicher. Durch die Option R wird das Programm sofort ausgeführt. LOAD löscht stets den aktuellen Speicherinhalt und schließt sämtliche Dateien vor dem Laden ab. Wird jedoch R benutzt, so bleiben eröffnete Datendateien eröffnet. Auf diese Weise können Programme verkettet oder in Abschnitten geladen werden. Außerdem kann auf dieselben Datendateien zugegriffen werden. (LOAD <Dateiname>, R und RUN <Dateiname>, R sind gleichbedeutend).

RUN <Dateiname> [,R]

Durch RUN <Dateiname> wird das Programm von der Diskette in den Speicher geladen und ausgeführt. RUN löscht den aktuellen Speicherinhalt und schließt sämtliche Dateien vor dem Laden des Programms ab. Wird jedoch die Option R angegegebne, so bleiben sämtliche eröffneten Datendateien eröffnet. (RUN <Dateiname> ,R und LOAD <Dateiname> ,R sind gleichbedeutend).

MERGE <Dateiname>

Lädt das Programm von der Diskette in den Speicher, löscht jedoch den aktuellen Speicherinhalt nicht. Die Programmzeilennummern auf der Diskette werden mit den Zeilennummern im Speicher gemischt. Haben zwei Zeilen dieselbe Nummer, so wird nur die Zeile von dem Diskettenprogramm gesichert. Nach einem MERGE-Befehl bleibt das "gemischte" Programm im Speicher und MS-BASIC kehrt zur Befehlsebene zurück.

KILL <Dateiname>

Löscht die Datei von der Diskette. <Dateiname>

kann eine Programmdatei, eine sequentielle oder Direktzugriffs-Datendatei bezeichnen.

NAME <Alter Dateiname>
AS <Neuer Dateiname>

Zur Änderung des Namens einer Diskettendatei wird die NAME-Anweisung, NAME <Alte Datei> AS <Neue Datei> ausgeführt. NAME kann mit Programmdateien, Direktzugriffsdateien oder sequentiellen Dateien benutzt werden.

GESCHÜTZTE DATEIEN

Soll ein Programm in einem codierten Binärformat gesichert werden, so muß die Protect-Option mit dem SAVE-Befehl benutzt werden. Zum Beispiel:

```
SAVE "MYPROG",P
```

HINWEIS: Da ein auf diese Art und Weise gesichertes Programm nicht aufgelistet oder editiert werden kann, wird man wahrscheinlich noch eine nicht geschützte Kopie des Programms zur Auflistung und Editierung sichern.

DATENDATEIEN AUF DISKETTE

Zwei Arten von Datendateien auf Diskette können von einem BASIC-Programm erstellt und benutzt werden: sequentielle Dateien und Direktzugriffsdateien.

Sequentielle Dateien

Sequentielle Dateien sind einfacher zu erstellen als Direktzugriffsdateien. Beim Zugriff zu den Daten ergibt sich jedoch eine Einschränkung bei Flexibilität und Geschwindigkeit. Bei den in eine sequentielle Datei geschriebenen Daten handelt es sich um ASCII-Zeichen. Diese Zeichen werden nacheinander (sequentiell) in der Reihenfolge gespeichert, in der sie gesendet werden. Sie werden auch auf dieselbe Art und Weise zurückgelesen.

Folgende Anweisungen und Funktionen werden mit sequentiellen Dateien benutzt:

CLOSE
EOF
INPUT#
LINE INPUT#
LOF
OPEN
PRINT#
PRINT# USING
WRITE#

Für weitere Informationen über diese Anweisungen und Funktionen wird auf das MS-BASIC Handbuch verwiesen. Für weitere Informationen über die LOF-Funktion wird auf den Abschnitt "LOF-Funktion" an früherer Stelle in diesem Handbuch verwiesen.

Erstellen einer sequentiellen Datei –
Für das Erstellen einer sequentiellen Datei und den Zugriff zu den Daten in der Datei sind folgende Programmschritte erforderlich:

1. Die Datei wird in dem "0"-Modus eröffnet.

   ```
   OPEN "0", #1, "DATA"
   ```

2. Die Daten werden mit der WRITE#-Anweisung in die Datei geschrieben. (Stattdessen kann der PRINT#-Befehl benutzt werden, in diesem Fall muß jedoch zuerst das MS-BASIC Handbuch herangezogen werden.)

   ```
   WRITE# 1,A$;B$;C$
   ```

3. Für den Zugriff zu den Daten in der Datei muß die Datei abgeschlossen und im "I"-Modus erneut eröffnet werden.

   ```
   CLOSE# 1
   OPEN"I",#1,"DATA"
   ```

4. Mit der INPUT#-Anweisung werden Daten von der sequentiellen Datei in das Programm gelesen.

   ```
   INPUT# 1,X$,Y$,Z$
   ```

Ein Programm, das eine sequentielle Datei erstellt, kann auch mit der PRINT# USING-Anweisung formatierte Daten auf die Diskette schreiben. So könnte beispielsweise die Anweisung

```
PRINT# 1,USING"###.##,";A,B,C,D
```

dazu benutzt werden, numerische Daten ohne besondere Abgrenzungszeichen auf die Diskette zu schreiben. Das Komma (,) am Ende der Formatzeichenfolge wird zur Trennung der einzelnen Elemente in der Diskettendatei benutzt.

Wird die LOC-Funktion mit einer sequentiellen Datei benutzt, so gibt sie die Anzahl von Sektoren zurück, die seit dem Eröffnen in die Datei geschrieben oder von der Datei gelesen wurden.

```
100 IF (LOC(1) > 50 THEN STOP
```

würde beispielsweise die Programmausführung beenden, wenn mer als 50 Sektoren in die Datei # 1 geschrieben oder von dieser Datei gelesen würden, seit sie eröffnet wurde.

Programm 1 ist ein kurzes Programm, das eine sequentielle Datei namens "DATA" anhand der am Termianl eingegebenen Informationen erstellt.

Programm 1 — — Erstellen einer sequentiellen Datendatei

```
10 OPEN "0",#1,"DATA"
20 INPUT "NAME";N$
25 IF N$="DONE" THEN END
30 INPUT "DEPARTMENT";D$
40 INPUT "DATE HIRED";H$
50 PRINT# 1,N$;",";D$;",";H$
60 PRINT: GOTO 20
RUN
NAME? MICKEY MOUSE
DEPARTMENT? AUDIO/VISUAL AIDS
DATE HIRED? 01/12/72

NAME: SHERLOCK HOLMES
DEPARTMENT? RESEARCH
DATE HIRED? 12/03/65

NAME? EBENEEZER SCROOGE
DEPARTMENT? ACCOUNTING
DATE HIRED? 04/26/78
```

```
NAME: SUPER MANN
DEPARTMENT? MAINTENANCE
DATE HIRED? 08/16/78

NAME? etc.
```

Zugriff zu einer sequentiellen Datei –
Programm 2 greift auf die Datei "DATA" zu, die in Programm 1 erstellt wurde, und zeigt die Namen sämtlicher im Jahre 1980 neu eingestellten Mitarbeiter an.

Programm 2 — — Zugriff zu einer sequentiellen Datei

```
10 OPEN "I",#,1,"DATA"
20 INPUT# 1,N$,D$,H$
30 IF RIGHT$(H$,2)="78" THEN PRINT N$
40 GOTO 20
RUN
EBENEEZER SCROOGE
SUPER MANN
Input past end in 20
OK
```

Programm 2 liest sequentiell jedes Element in der Datei. Nachdem sämtliche Daten gelesen wurden, führt Zeile 20 zu einer Fehlermeldung "Input past end". Um diese Fehlermeldung zu vermeiden, muß Zeile 15 eingefügt werden, die die EOF-Funktion benutzt, um eine Überprüfung auf das Dateiende vorzunehmen.

```
15 IF EOF(1) THEN END
```

Dann wird Zeile 40 in GOTO 15 geändert.

Hinzufügen von Daten zu einer sequentiellen Datei
Steht eine sequentielle Datei auf Diskette und sollen weitere Daten an das Ende der Datei angefügt werden, so kann die Datei nicht einfach im "0"-Modus eröffnet und mit dem Schreiben von Daten begonnen werden. Sobald eine sequentielle Datei im "0"-Modus eröffnet wird, wird ihr aktueller Inhalt zerstört. Stattdessen wird die OPEN-Anweisung mit dem APPEND-Modus benutzt, wie in Kapitel 2 "Unterschiede der MS-BASIC-Sprache bei MS-DOS" beschrieben.

Direktzugriffsdateien

Beim Erstellen und Zugreifen auf Direktzugriffsdateien sind mehr Programmschritte als bei sequentiellen Dateien erforderlich. Es gibt jedoch auch Vorteile bei der Benutzung von Direktzugriffsdateien. Einer dieser Vorteile besteht darin, daß Direktzugriffsdateien weniger Platz auf der Diskette erfordern, da MS-BASIC sie in einem gepackten Binärformat speichert. (Eine sequentielle Datei wird als eine Folge von ASCII-Zeichen gespeichert).

Der größte Vorteil der Direktzugriffsdateien besteht darin, daß auf die Daten direkt, d.h. an beliebiger Stelle auf der Diskette, zugegriffen werden kann. Bei Direktzugriffsdateien müssen nicht sämtliche auf der Diskette stehenden Informationen gelesen werden, wie bei sequentiellen Dateien. Dies ist möglich, da die Informationen in bestimmten Einheiten, die als Sätze bezeichnet werden, gespeichert und wieder aufgerufen werden. Jeder Satz ist numeriert.

Folgende Anweisungen und Funktionen werden mit Direktzugriffsdateien benutzt:

CLOSE
CVD
DVI
CVS
FIELD
GET
LSET
LOC
MKD$
MKI$
MKS$
OPEN
PUT
RSET

Erstellen einer Direktzugriffsdatei

Beim Erstellen einer Direktzugriffsdatei sind die folgenden Programmschritte erforderlich.

1. Die Datei wird für den Direktzugriff ("R"-Modus) mit OPEN eröffnet. In diesem Beispiel wird eine Satzlänge von 32 Bytes angegeben. Wird die Satzlänge weggelassen, so beträgt die Standardvorgabe 128 Bytes.

   OPEN "R",#1, "FILE",32

2. Mit der FIELD-Anweisung wird der Platz im Direktzugriffspuffer für die Variablen zugeordnet, die in die Direktzugriffsdatei geschrieben werden.

```
FIELD# 1,20 AS N$
4 AS A$, 8 AS P$
```

3. Mit dem LSET-Befehl werden die Daten in den Direktzugriffspuffer übertragen. Numerische Werte müssen in Zeichenfolgenwerte umgewandelt werden, wenn sie in den Puffer gesetzt werden. Zu diesem Zweck werden die "Make"-Funktionen benutzt. Durch MKI$ wird ein ganzzahliger Wert in eine Zeichenfolge umgewandelt. Durch MKS$ wird ein ganzzahliger Wert in einen Wert mit einfacher Genauigkeit umgewandelt. Durch MKD$ wird ein ganzzahliger Wert in einen Wert mit doppelter Genauigkeit umgewandelt. Für weitere Informationen über diese Funktionen wird auf das MS-BASIC Handbuch verwiesen.

```
LSET N$=M$
LSET A$=MKS$(AMT)
P$=TEL$
```

4. Mit der PUT-Anweisung werden die Daten von dem Puffer auf die Platte geschrieben.

```
PUT# 1,CODE%
```

Wird die LOC-Funktion mit Direktzugriffsdateien benutzt, so wird die "aktuelle Satznummer" zurückgegeben. Die aktuelle Satznummer entspricht der letzten Satznummer, die in einer GET- oder PUT-Anweisung benutzt wurde, plus 1. So beendet die Anweisung

```
IF LOC(1) > 50 THEN END
```

beispielsweise die Programmausführung, wenn die aktuelle Satznummer in Datei # 1 größer ist als 50.

Programm 3 schreibt Informationen, die an dem Terminal eingegeben werden, in eine Direktzugriffsdatei.

Programm 3 — — Erstellen einer Direktzugriffsdatai

```
10 OPEN "R",#1,"FILE",32
20 FIELD#1,20 AS N$, 4 AS A$, 8 AS P$
30 INPUT "2-DIGIT CODE";CODE%
40 INPUT "NAME";X$
50 INPUT "AMOUNT";AMT
60 INPUT "PHONE";TEL$:PRINT
70 LSET N$=X$
80 LSET A$=MKS$(AMT)
90 LSET P$=TEL$
100 PUT#1, CODE%
110 GOTO 30
```

Bei jeder Ausführung der PUT-Anweisung wird ein Satz in die Datei geschrieben. Der zweistellige Code, der in Zeile 30 eingegeben wird, wird zur Satznummer.

HINWEIS: Eine mit FIELD zugewiesene Zeichenfolgenvariable darf in einer INPUT- oder LET-Anweisung nicht benutzt werden. Dadurch zeigt der Zeiger für diese Variable in die Zeichenfolge anstatt in den Puffer der Direktzugriffsdatei.

Zugriff auf eine Direktzugriffsdatei

Beim Erstellen einer Direktzugriffsdatei sind die folgenden Schritte erforderlich.

1. Die Datei wird im "R"-Modus eröffnet.

   ```
   OPEN"R",#1,"FILE",32
   ```

2. Mit der FIELD-Anweisung wird Platz im Direktzugriffspuffer für die Variablen zugewiesen, die von der Datei gelesen werden.

   ```
   FIELD#1,20 AS N$
   4 AS A$, 8 AS P$
   ```

HINWEIS: In einem Programm, das sowohl die Eingabe als auch die Ausgabe bei derselben Direktzugriffsdatei ausführt, kann häufig eine einzige OPEN-Anweisung und eine einzige FIELD-Anweisung benutzt werden.

3. Mit der GET-Anweisung wird der gewünschte Satz in den Direktzugriffspuffer übertragen.

   GET# 1,CODE%

4. Auf die Daten in dem Puffer kann nun von dem Programm zugegriffen werden. Numerische Werte müssen wieder mit den Umwandlungsfunktionen in Zahlen umgewandelt werden. CVI wandelt numerische Werte in Ganzzahlwerte um. CVS wandelt numerische Werte in Werte mit einfacher Genauigkeit um. CVD wandelt numerische Werte in Werte mit doppelter Genauigkeit um.

   PRINT N$
   PRINT CVS (A$)

Programm 4 greift auf die Direktzugriffsdatei "FILE" zu, die in Programm 3 erstellt wurde. Wird der zweistellige Code an dem Terminal eingegeben, so werden die mit diesem Code verknüpften Informationen aus der Datei gelesen und angezeigt.

Programm 4 — — Zugriff zu einer Direktzugriffsdatei

```
10 OPEN "R",# 1,"FILE",32
20 FIELD# 1, 20 AS N$' 4 AS A$, 8 AS P$
30 INPUT "2-DIGIT CODE";CODE%
40 GET# 1, CODE%
50 PRINT N$
60 PRINT USING "$$# # # . # # ";CVS(A$)
70 PRINT P$:PRINT
80 GOTO 30
```

Programm 5 ist ein Bestandsprogramm, das den Zugriff zu der Direktzugriffsdatei darstellt. In diesem Programm wird die Satznummer als Teilnummer benutzt. Es wird davon ausgegangen, daß das Bestandsverzeichnis nicht mehr als 100 verschiedene Teilnummern enthält. Die Zeilen 900 bis 960 initialisieren die Datendatei, indem CHR$ (255) als erstes Zeichen jedes Satzes geschrieben wird. Mit diesem Zeichen wird später (Zeile 270 und Zeile 500) bestimmt, ob schon für diese Teilnummer ein Eintrag vorhanden ist.

Die Zeilen 140 bis 210 zeigen die verschiedenen Bestandsfunktionen an, die das Programm ausführt. Wird die gewünschte Funktionsnummer eingegeben, so wird in Zeile 230 eine Verzweigung zu der entsprechenden Subroutine vorgenommen.

Programm 5 — — Bestandsprogramm

```
120 OPEN"R",#1,"INVEN.DAT",39
130 FIELD#1,1 AS F$,30 AS D$,2 AS Q$2 AS R$,4 AS P$
140 PRINT: PRINT "FUNCTIONS:": PRINT
150 PRINT 1,"INITIALIZE FILE"
160 PRINT 2,"CREATE A NEW ENTRY"
170 PRINT 3,"DISPLAY INVENTORY FOR ONE PART"
180 PRINT 4,"ADD TO STOCK"
190 PRINT 5,"SUBTRACT FROM STOCK"
200 PRINT 6,"DISPLAY ALL ITEMS BELOW REORDER
    LEVEL"
210 PRINT: PRINT: INPUT"FUNCTION";FUNCTION
220 IF (FUNCTION<1)OR(FUNCTION>6) THEN PRINT
    "BAD FUNCTION NUMBER": GOTO 140
230 ON FUNCTION GOSUB 900,250,390,380,560,680
240 GOTO 210
250 REM BUILD NEW ENTRY
260 GOSUB 840
270 IF ASC(F$) < > 255 THEN INPUT "OVERWRITE";A$:
    IF A$< >"Y" THEN RETURN
280 LSET F$= CHR$(0)
290 INPUT "DESCRIPTION";DESC$
300 LSET D$=DESC$
310 INPUT "QUANTITY IN STOCK";Q%
320 LSET Q$=MKI$(Q%)
330 INPUT "REORDER LEVEL";R%
340 LSET R$=MKI$(R%)
350 INPUT "UNIT PRICE";P
360 LSET P$=MKS$(P)
370 PUT# 1,PART%
380 RETURN
390 REM DISPLAY ENTRY
400 GOSUB 840
410 IF ASC(F$)=255 THEN PRINT "NULL ENTRY";RETURN
420 PRINT USING "PART NUMBER###";PART%
430 PRINT D$
440 PRINT USING "QUANTITY ON HAND #####";CVI(Q$)
450 PRINT USING "REORDER LEVEL#####";CVIR$)
460 PRINT USING "UNIT PRICE $$###.##";CVS(P$)
470 RETURN
480 REM ADD TO STOCK
490 GOSUB 840
500 IF ASC(F$)=255 THEN PRINT "NULL ENTRY";RETURN
```

```
510 PRINT D$: INPUT "QUANTITY TO ADD ";A%
520 Q%=CVI(Q$)+A%
530 LSET Q$=MKI$(Q%)
540 PUT# 1,PART%
550 RETURN
560 REM REMOVE FROM STOCK
570 GOSUB 840
580 IF ASC(F$)=255 THEN PRINT "NULL ENTRY": RETURN
590 PRINT D$
600 INPUT "QUANTITY TO SUBTRACT";S%
610 Q%=CVI(Q$)
620 IF (Q%—S%) < 0 THEN PRINT "ONLY";Q%;" IN STOCK":
    GOTO 600
630 Q%=Q%—S%
640 IF Q%=< CVI(R$) THEN PRINT "QUANTITY NOW";
    Q%;"REORDER LEVEL";CVI(R$)
650 LSET Q$=MKI$(Q%)
660 PUT# 1,PART%
670 RETURN
680 REM DISPLAY ITEMS BELOW REORDER LEVEL
690 FOR I=1 TO 100
710 GET# 1,I
720 IF CVI(Q$)< CVI(R$) THEN PRINT D$;"QUANTITY";
    CVI(Q$) TAB(50) "REORDER LEVEL";CVI(R$)
730 NEXT I
740 RETURN_
840 INPUT "PART NUMBER".PART%
850 IF(PART%< 1) OR (PART%> 100) THEN PRINT
    "BAD PART NUMBER": GOTO 840 ELSE GET# 1,
    PART%: RETURN
890 END
900 REM INITIALIZE FILE
910 INPUT "ARE YOU SURE";B$: IF B$< >"Y" THEN
    RETURN
920 LSET F$= CHR$(255)
930 FOR I=1 TO 100
940 PUT# 1,I
950 NEXT I
960 RETURN
```

KAPITEL 5

SUBROUTINEN IN DER ASSEMBLERSPRACHE

Dieses Kapitel ist für die Benutzer gedacht, die Subroutinen in der Assemblersprache aus den MS-BASIC Programmen aufrufen. Die Benutzer, die keine Subroutinen in der Assemblersprache benutzen, benötigen diese Informationen nicht.

Sämtliche Versionen von MS-BASIC verfügen über Verbindungsmöglichkeiten mit Subroutinen in der Assemblersprache über die USR-Funktion und die CALL-Anweisung. Die CALLS-Anweisung wird auch unter den 8086 Implementierungen unterstützt.

Durch die USR-Funktion können Subroutinen in der Assemblersprache auf dieselbe Art und Weise aufgerufen werden, wie interne MS-BASIC Funktionen aufgerufen werden. Für die Verbindung zwischen 8086 Programmen in der Maschinensprache und MS-BASIC werden jedoch CALL- oder CALLS-Anweisungen empfohlen. Diese Anweisungen erzeugen einen einfacher zu lesenden Quellencode und können mehrere Parameter übergeben. Darüber hinaus ist die CALL-Anweisung mit mehr Sprachen kompatibel als die USR-Funktion.

ZUWEISUNG DES SPEICHERPLATZES

Für eine Subroutine in der Assemblersprache muß Speicherplatz reserviert werden, bevor sie geladen werden kann. Zu diesem Zweck wird der /M:-Schalter während des Startens benutzt. (Für einen Überblick über das Startverfahren wird auf Kapitel 1 verwiesen.) Mit dem /M:-Schalter wird die höchste Speicheradresse festgelegt, die von MS-BASIC benutzt werden kann.

Neben dem Codebereich für den MS-BASIC Interpretierer benutzt MS-BASIC eine Speicherkapazität von bis zu 64K, die mit ihrem Datensegment (DS) beginnt.

Wird beim Aufruf einer Subroutine in der Assemblersprache mehr Stapelplatz benötigt, so kann der MS-BASIC Stapel gesichert und ein neuer Stapel zur Benutzung durch die Subroutine in der Assemblersprache bereitgestellt werden. Bevor jedoch aus der Subroutine gesprungen wird, muß der MS-BASIC Stapel wieder hergestellt werden.

Die Subroutine in der Assemblersprache kann mit dem Betriebssystem oder mit der POKE-Anweisung in den Speicher geladen werden. Wird das Microsoft-Dienstprogramm-Softwarepaket benutzt, so können die Routinen mit dem MACRO Assembler (MS-MACRO) assembliert und mit Link Lead (MS-LINK) gebunden (jedoch nicht geladen) werden. Beim Laden der Programmdatei müssen folgende Richtlinien beachtet werden:

1. Die Subroutinen dürfen keine langen Referenzen enthalten.
2. Die ersten 512 Bytes der MS-LINK Ausgabedatei müssen übersprungen werden. Danach wird der Rest der Datei eingelesen.

CALL-ANWEISUNG

Die CALL-Anweisung wird als Verbindung zwischen 8086 Programmen in der Maschinensprache und MS-BASIC empfohlen. Die USR-Funktion sollte nur benutzt werden, wenn vorher geschriebene Programme ausgeführt werden, die schon USR-Funktionen enthalten. (Für eine Beschreibung der Funktion wird auf "USR-Funktion" in diesem Kapitel verwiesen.)

Syntax CALL <Variablenname> [(<Parameterliste)>]

Wobei <Variablenname> die Segmentposition enthält, die dem Anfangspunkt der aufgerufenen Subroutine im Speicher entspricht.

<Parameterliste> enthält die durch Kommas (,) voneinander getrennten Variablen oder Konstanten, die an die Routine übergeben werden müssen.

Beim Aufruf der CALL-Anweisung kommt es zu folgenden Ereignissen:

1. Für jeden Parameter in der Parameterliste wird der aus 2 Bytes bestehende Offset der Parameteradresse innerhalb des Datensegments (DS) in den Stapel bewegt.
2. Das Codesegment (CS) für die Rücksprungadresse von MS-BASIC und das Offset (IP) werden in den Stapel bewegt.
3. Die Steuerung wird über einen 8086 CALL FAR-Befehl zu der in der letzten DEF SEG-Anweisung angegebenen Segmentadresse und zu dem in <Variablenname> angegebenen Offset an die Benutzerroutine übertragen.

In den nachfolgenden Diagrammen wird der Status des Stapels bei der Ausführung der CALL-Anweisung, sowie die Bedingung des Stapels während der Ausführung der aufgerufenen Subroutine dargestellt.

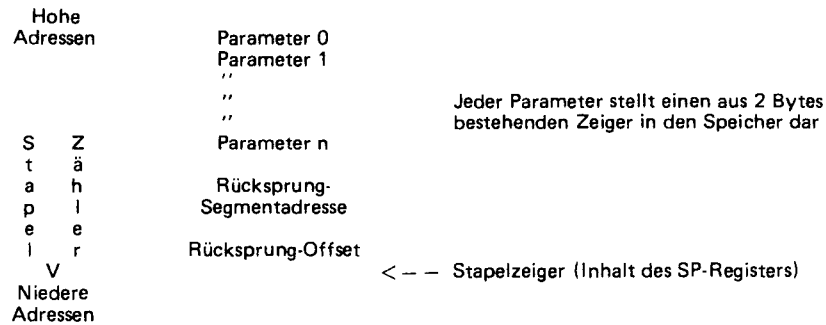


Abbildung 1 Status des Stapels bei der Aktivierung der CALL-Anweisung

Nun hat die Benutzerroutine die Kontrolle. Auf die Parameter kann durch Bewegen des Stapelzeigers (SP) und durch Hinzufügen eines positiven Relativzeigers zu (BR) Bezug genommen werden.

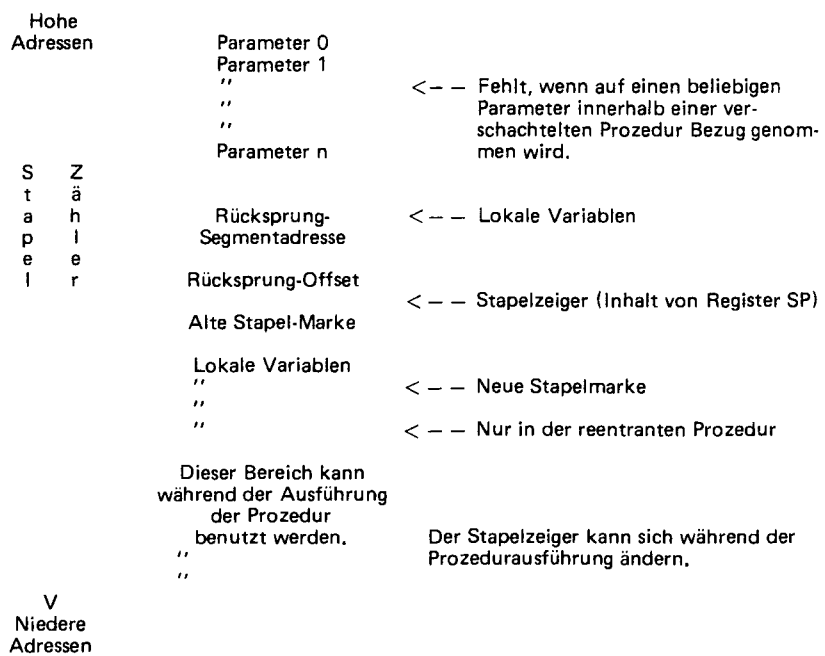


Abbildung 2 Status des Stapels während der Ausführung einer CALL-Anweisung

Bei Codieren einer Subroutine müssen die folgenden Regeln eingehalten werden:

1. Die aufgerufene Routine kann die Register AX, BX, CX, DX, SI, DI und BP zerstören.
2. Dem aufgerufenen Programm müssen die Anzahl und Länge der übergebenen Parameter bekannt sein. Die Bezugnahmen auf Parameter sind positive Offsets, die zu (BP) hinzugefügt werden (wobei davon ausgegangen wird, daß die aufgerufene Routine den aktuellen Stapelzeiger in BP bewegt hat, d.h. MOV, BP,SP). Dies bedeutet, daß die Adresse von p1 bei 8(BP), p2 bei 6(BP), von p3 bei 4(BP) liegt usw.
3. Die aufgerufene Routine muß eine RET<n> Anweisung ausführen (wobei <n> der Anzahl von Parametern in der Parameterliste mal zwei entspricht), um den Stapel auf den Anfang der aufrufenden Routine auszurichten.
4. Die Werte werden an MS-BASIC zurückgegeben, indem der Variablenname, der das Ergebnis aufnimmt, in die Parameterliste eingefügt wird.
5. Handelt es sich bei dem Parameter um eine Zeichenfolge, so zeigt der Relativzeiger des Parameters auf 3 Bytes, die — als Einheit — als der "Zeichenfolgenbeschreiber" bezeichnet werden. Byte 0 des Zeichenfolgenbeschreibers enthält die Länge der Zeichenfolge (0 bis 255). Bei den Bytes 1 bzw. 2 handelt es sich um die unteren und oberen 8 Bits der Anfangsadresse der Zeichenfolge innerhalb der Zeichenfolge.

WARNUNG

Ist der Parameter ein Zeichenfolgenliteral in dem Programm, so zeigt der Zeichenfolgenbeschreiber zu dem Programmtext. Es muß darauf geachtet werden, daß das Programm nicht auf diese Art und Weise geändert oder zerstört wird. Um unvorhersehbare Ergebnisse zu vermeiden, wird +" " zu dem Zeichenfolgenliteral in dem Programm hinzugefügt. Zum Beispiel:

```
20 A$ = "BASIC"+" "
```

Dadurch wird das Zeichenfolgenliteral in die Zeichenfolge kopiert. Nun kann die Zeichenfolge geändert werden, ohne daß das Programm betroffen wird.

6. Zeichenfolgen können von Benutzerroutinen geändert werden, die Länge darf jedoch **nicht** geändert werden. MS-BASIC kann Zeichenfolgen nicht richtig verarbeiten, wenn deren Länge durch externe Routinen geändert wird.

Beispiel für eine CALL-Anweisung:

```
100 DEF SEG=&H8000
110 F00=&H7FA
120 CALL F00(A,B$,C)
 .
 .
 .
```

In Zeile 100 wird das Segment auf H8000 gesetzt. Der Wert der Variablen F00 wird als unterstes Wort zu der Adresse hinzugefügt, nachdem der DEF SEG-Wert um 8 Bits nach links geschoben wurde. (Hier handelt es sich um eine Funktion des Mikroprozessors und nicht um eine Funktion von MS-BASIC). Hier wird F00 auf &H7FA gesetzt, so daß durch Aufruf von F00 die Subroutine in der Adresse 8000:7FA Hex (absolute Adresse 807FA Hex) ausgeführt wird.

Die folgende Routine in der 8086 Assemblersprache zeigt den Zugriff zu den übergebenen Parameters. Das Rückgabeergebnis wird in der Variablen "C" gespeichert.

```
MOV   BP,SP      ; Aktuelle Stapelposition in BP ermitteln.
MOV   BX,6[BP]   ; Adresse von B$ ermitteln.
MOV   CL,[BS]    ; Länge von B$ in CL ermitteln.
MOV   DX,1[BX]   ; Adresse des B$ Textes in DX ermitteln.
          .
          .
          .
MOV   SI,8[BP]   ; Adresse von 'A' in SI ermitteln.
MOV   DI,4[BP]   ; Zeiger zu 'C' in DI ermitteln.
MOV   WORD       ; Variable 'A' in 'C' speichern.
RET   6          ; Stapel wieder herstellen, Rücksprung.
```

HINWEIS: Das aufgerufene Programm muß den Variablentyp für die übergebenen numerischen Parameter kennen. In dem vorhergehenden Beispiel kopiert der Befehl

```
MOVS WORD
```

nur zwei Bytes. Dies ist in Ordnung, wenn es sich bei den Variablen A und C um Ganzzahlen handelt. Es müssen jedoch

vier Bytes kopiert werden, wenn die Variablen die einfache Genauigkeit aufweisen, und 8 Bytes, wenn sie die doppelte Genauigkeit aufweisen.

CALLS-ANWEISUNG

Wie schon vorher erwähnt, muß die CALLS-Anweisung für den Zugriff zu den MS-FORTRAN Routinen benutzt werden. CALLS arbeitet wie CALL, nur daß bei CALLS die Parameter als segmentierte Adressen und nicht als unsegmentierte Adressen übergeben werden.

USR-FUNKTION

Auch wenn die CALL-Anweisung für den Aufruf von Subroutinen in der Assemblersprache empfohlen wird, steht die USR-Funktion immer noch zur Verfügung, um die Kompatibilität mit vorher geschriebenen Programmen aufrecht zu erhalten.

Syntax USR[<Ziffer<] [(<Parameter>)]

Wobei <Ziffer> von 0 bis 9 geht. <Ziffer> gibt die aufgerufene USR-Routine an. Wird <Ziffer> weggelassen, so wird von USR0 ausgegangen.
<Parameter> ist ein beliebiger numerischer oder Zeichenfolgenauseruck. Die Parameter werden im einzelnen in den folgenden Abschnitten beschrieben.

Bei dieser Implementierung von MS-BASIC muß eine DEF SEG-Anweisung vor einem USR-Funktionsaufruf ausgeführt werden, um zu gewährleisten, daß das Codesegment zu der aufgerufenen Subroutine zeigt. Die in der DEF SEG-Anweisung angegebene Segmentadresse bestimmt das Anfangssegment der Subroutine.

Für jede USR-Funktion muß eine entsprechende DEF USR-Anweisung ausgeführt werden, um das Offset des USR-Funktionsaufrufs zu definieren. Dieses Offset und die gerade aktive DEF SEG-Adresse bestimmen die Anfangsadresse der Subroutine.

Wird der USR-Funktionsaufruf vorgenommen, so enthält das Register AL einen Wert, der den Typ des übergebenen Parameters angibt. Einer der folgenden Werte kann in AL angegeben werden:

Wert in AL	Parametertyp
2	Aus zwei Bytes bestehende Ganzzahl (Zweier-Komplement)
3	Zeichenfolge
4	Gleitpunktzahl mit einfacher Genauigkeit
8	Gleitpunktzahl mit doppelter Genauigkeit

Handelt es sich bei dem Parameter um eine Zahl, so zeigt das BX-Registerpaar zu dem Gleitpunktakkumulator (FAC), in dem der Parameter gespeichert ist:

FAC stellt den Exponenten minus 128 dar. Der Binärpunkt stehts links von dem signifikantesten Bit der Mantisse.

FAC-1 enthält die sieben höchstwertigen Bits der Mantisse, wobei die führende 1 unterdrückt (impliziert) wird. Bit 7 gibt das Vorzeichen der Zahl an (0=positiv, 1=negativ).

Handelt es sich bei dem Parameter um eine Ganzzahl:
so enthält FAC minus 2 die 8 oberen Bits des Parameters,
so enthält FAC-3 die niederen Bits des Parameters.

Handelt es sich bei dem Parameter um eine Gleitpunktzahl mit einfacher Genauigkeit:
so enthält FAC-2 die 8 mittleren Bits der Mantisse,
so enthält FAC-3 die 8 niederen Bits der Mantisse.

Handelt es sich bei dem Parameter um eine Gleitpunktzahl mit doppelter Genauigkeit:
so enthalten FAC-7 bis FAC-4 4 weitere Bytes der Mantisse
(FAC-7 enthält die 8 niederwertigsten Bits).

Handelt es sich bei dem Parameter um eine Zeichenfolge, so zeigt das DX-Registerpaar zu den 3 Bytes, die — als eine Einheit — als "Zeichenfolgenbeschreiber" bezeichnet werden. Byte 0 des Zeichenfolgenbeschreibers enthält die Länge der Zeichenfolge (0 bis 255). Die Bytes 1 bzw. 2 stellen die 8 nieder- bzw. hochwertigen Bits der Anfangsadresse der Zeichenfolge in dem MS-BASIC Datensegment dar.

WARNUNG

Handelt es sich bei dem Parameter um ein Zeichenfolgenliteral in dem Programm, so zeigt der Zeichenfolgenbeschreiber auf den Programmtext. Hier muß darauf geachtet werden, daß das Programm auf diese Weise nicht geändert oder zerstört wird.

Normalerweise weist der von einer USR-Funktion zurückgegebene Wert denselben Typ (Ganzzahl, Zeichenfolge, einfache oder doppelte Genauigkeit) auf, wie der in ihn übertragene Parameter.

Beispiel:

```
110 DEF USR0=&H8000 'Geht davon aus, daß
    der Benutzer /M:32767 angegeben hat
120 X=5
130 Y=USR0(X)
140 PRINT Y
```

Der Typ der Variablen (numerische Variable oder Zeichenfolgenvariable), die den Funktionsaufruf empfängt, muß mit dem übergebenen Parameter konsistent sein.

ANHANG A

MS-DOS FEHLERMELDUNGEN

Die folgenden Fehlermeldungen können bei MS-BASIC mit MS-DOS angezeigt werden.

Nummer	Meldung
70	Disk write protected Die Diskette verfügt über einen Schreibschutz, oder es handelt sich um eine Diskette, auf die nicht geschrieben werden kann.
71	Disk not ready Diese Fehlermeldung könnte von einer Reihe von Problemen verursacht werden. Die wahrscheinlichste Fehlerursache besteht darin, daß die Diskette nicht richtig eingelegt wurde.
72	Disk media error Während des Schreibens auf eine Diskette oder während des Lesens von der Diskette ist es zu einem Hardware- oder Diskettenproblem gekommen. So könnte beispielsweise die Diskette beschädigt sein oder das Diskettenlaufwerk nicht richtig arbeiten.
74	Rename across disks Es wurde versucht, eine Datei mit einer neuen Laufwerkbezeichnung umzubenennen. Dies ist nicht zulässig.

NCR

NCR DECISION MATE V

GW™-BASIC - Erweiterung

GW-BASIC ERWEITERUNG

INHALT

1. ALLGEMEINE INFORMATIONEN

GW-BASIC LEISTUNGEN 1-1
Bildschirmmodus 1-1
  Textmodus 1-1
  Grafikmodus 1-2
  X- und Y-Koordinaten 1-3
Farbauswahl 1-4
Musikauswahl 1-7
Datenübertragung 1-7
BENUTZEN VON GW-BASIC 1-7
Vorprogrammierte Funktionstasten 1-7
Gesamtbildschirmeditor 1-8
Konfiguration für GW-BASIC 1-9
Laden von GW-BASIC 1-13
Auswahl des Speicherausdrucks innerhalb des Programms ... 1-14

2. ANWEISUNGEN UND FUNKTIONEN

BEEP 2-3
CIRCLE 2-4
CLS 2-7
COLOR 2-8
CSRLIN 2-10
DRAW 2-11
EDIT 2-14
GET und PUT 2-15
KEY 2-21
LCOPY 2-24
LINE 2-25
LIST 2-28
LOCATE 2-30
ON COM(n) 2-32
ON KEY 2-34

OPEN (COM) 2-37
PAINT 2-41
PLAY 2-43
POINT 2-46
PRESET 2-47
PSET 2-48
RETURN 2-49
SCREEN 2-50
SOUND 2-51
WIDTH 2-54

3. DATENÜBERTRAGUNG

ERÖFFNEN DER DATENÜBERTRAGUNGSDATEI 3-1
EIN-/AUSGABE BEI DER DATENÜBERTRAGUNG 3-1
E/A-Funktionen 3-2
INPUT$-Funktion 3-2
GET- und PUT-Anweisungen für die Datenübertragung 3-3
STEUERSIGNALE 3-3
Ausgabesignale 3-4
Eingabesignale 3-4
BEISPIELPROGRAMM 3-4
Hinweise zu dem Beispielprogramm 3-5

A. GW-BASIC FEHLERMELDUNGEN A-1

B. ERWEITERUNG DES TERMINAL-FUNKTIONSCODES

POSITIONIEREN DES CURSORS B-1
LÖSCHEN DES BILDSCHIRMS B-1
GRAFIK-ATTRIBUTE B-2
DER LAUTSPRECHER B-2
DIE FUNKTIONSTASTEN B-2

ALLGEMEINE INFORMATION

NCR GW-BASIC erweitert die Möglichkeiten von MS-BASIC bei MS-DOS, indem bei der Programmierung des NCR DECISION MATE V Grafik-, Ton- und Kommunikationseinrichtungen zur Verfügung gestellt werden. Dies gilt sowohl für den Schwarz/Weiß- als auch für den Farbbildschirm. Damit diese Einrichtungen einfach in das jeweilige Benutzerprogramm eingebaut werden können, umfaßt die GW-BASIC Software einfach zu benutzende Anweisungen, vordefinierte Funktionstasten, einen Gesamtbildschirmeditor und eine einfache GW-BASIC Konfigurier-Routine, die zur Definition der Übertragungsumgebung und für das Ausdrucken des Bildschirmformats erforderlich ist.

Diese Beschreibung von GW-BASIC ist in drei Kapitel unterteilt, plus einem Anhang für Fehlermeldungen. Das erste Kapitel enthält einen Überblick über die Einrichtungen in dieser Erweiterung. Außerdem wird beschrieben, welche Schritte zur Benutzung von BW-BASIC erforderlich sind. Das zweite Kapitel definiert jede der Anweisungen in der Erweiterung. In dem letzten Kapitel wird die Datenübertragung beschrieben.

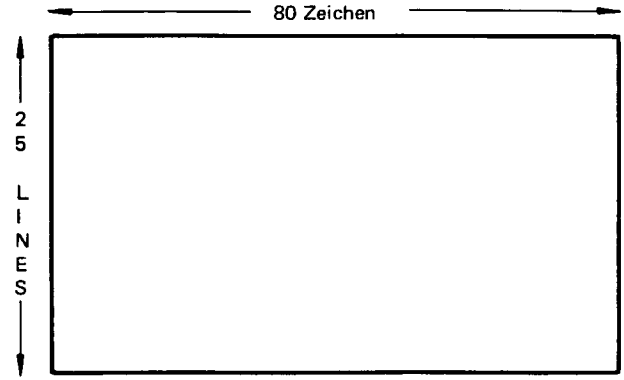
GW-BASIC LEISTUNGEN

BILDSCHIRM-MODUS

Zur Benutzung einer der Anweisungen in der GW-BASIC Erweiterung muß zuerst GW-BASIC geladen werden. GW-BASIC kann in eier von zwei Betriebsarten ausgeführt werden. Modus 0 entspricht dem *Textmodus*. Hier handelt es sich um den Standardmodus (gebräuchlichster Modus). Modus 1 entspricht dem *Grafikmodus*. Das System muß (mit der Screen-Anweisung) in diesen Modus geschaltet werden, wenn bestimmte Anweisungen in der GW-BASIC Erweiterung benutzt werden. Warum dies erforderlich ist, wird deutlich, wenn erst einmal bekannt ist, wie GW-BASIC die Bildschirmeingabe und Bildschirmausgabe handhabt.

Textmodus

Im Textmodus geht die Software davon aus, daß der Bildschirm 25 Zeilen und 80 Zeichen pro Zeile umfaßt. (Zeile 25 ist reserviert; siehe "Vordefinierte Funktionstasten" in diesem Kapitel).

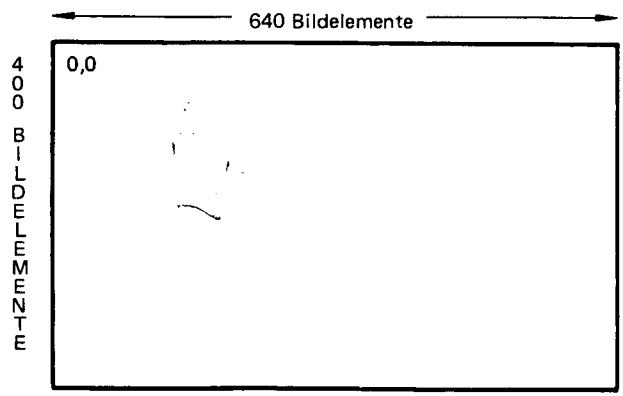


Bildschirm im Textmodus

Zeigt die Software ein vom Benutzer über die Tastatur eingegebenes Zeichen an, so übersetzt sie die vom Benutzer betätigte Taste und zeigt ihr Abbild an der Position der Schreibmarke an. Im Textmodus wird mit den auf der Tastatur abgebildeten Zeichen gearbeitet.

Grafikmodus

Der Grafikmodus ist schon etwas subtiler. In ihm können Bilder und andere Formen gezeichnet werden. Die Software geht davon aus, daß der Bildschirm aus Bildelementen (Pixels) besteht. Ein Bildelement ist einfach ein Punkt auf dem Bildschirm. Der NCR DECISION MATE V verfügt über 640 Bildelemente in der Breite und 400 Bildelemente in der Länge.



Bildschirm im Grafikmodus

Unter den in GW-BASIC zur Verfügung stehenden Anweisungen und Funktionen müssen die folgenden im Grafikmodus benutzt werden. ("Sich im Grafikmodus befinden" bedeutet einfach, daß eine Bildschirmanweisung für Modus 1 eingegeben wurde).

CIRCLE	PAINT	PRESET	GET	POINT (Funktion)
DRAW	LINE	PSET	PUT	

Selbstverständlich können im Grafikmodus auch andere BASIC-Anweisungen benutzt werden. Der Benutzer *muß* sich jedoch im Grafikmodus befinden, um Grafikanweisungen benutzen zu können. Da die Bildschirmbearbeitung bei Grafiken komplexer ist als bei Texten, muß immer wieder in den Textmodus zurückgeschaltet werden, wenn der Grafikteil des Programms beendet ist. Das Editieren des Programms ist immer im Textmodus auszuführen.

X- und Y-Koordinaten

Die Grafikanweisungen erfordern sowohl eine X- als auch eine Y-Koordinate, welche beschreiben, an welcher Stelle auf dem Bildschirm mit der Zeichnung begonnen werden soll. Die X-Koordinate stellt die Horizontalposition auf dem Bildschirm dar, während die Y-Koordinate die Vertikalposition darstellt. 0,0 ist die Position des ersten Bildelements in der oberen linken Ecke des Bildschirms.

Bei den meisten Grafikanweisungen können die Koordinaten in einer von zwei Formen angegeben werden: mit einer absoluten Form, bei der X,Y die genaue Position angeben, oder einer verschobenen Form, bei der X,Y die Abstandswerte von dem letzten Punkt darstellen, auf den Bezug genommen wurde. Werden die Koordinaten in verschobener Form angegeben, so muß das Wort STEP aufgenommen werden, damit die Software weiß, daß von einem vorher festgelegten Punkt weitergegangen wird.

In diesem Zusammenhang sollten die beiden folgenden Beispiele betrachtet werden:

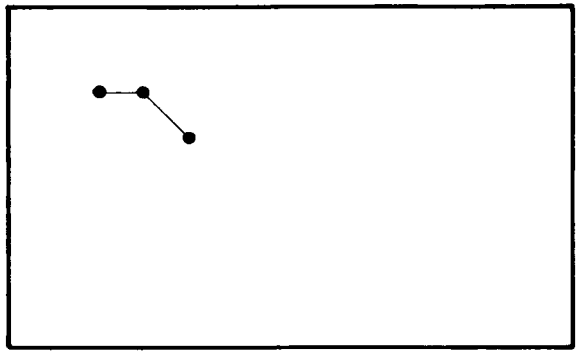
```
10 SCREEN 1
20 LINE (100,100)-(150,100)
30 LINE (150,100)-(200,150)
```

Beispiel 1 Absolute Form

```
10 SCREEN 1
20 LINE (100,100)-STEP(50,0)
30 LINE-STEP (50,50)
```

Beispiel 2 Verschobene Form

Beide Beispiele führen zur Anzeige der folgenden Zeilen auf dem Bildschirm. Beide geben eine Anfangsposition des Bildelements in 100,100 an.



FARBAUSWAHL

Bei einem Farbbildschirm können verschiedene Farben für den Vordergrund (die Zeichen oder das Grafikabbild) und den Hintergrund (den Bildschirm selbst) gewählt werden. Bei dem Schwarz/Weiß-Modell werden grüne Zeichen auf schwarzem Hintergrund angezeigt. Die gewünschten Farben werden mit der Color-Anweisung oder beim Zeichnen einer Grafik mit der Graphics-Anweisung angegeben.

Die bei dem NCR DECISION MATE V zur Verfügung stehende Farben zeigt die nachfolgende Liste. (Mit den Zahlen wird die Farbe in den Grafikanweisungen definiert).

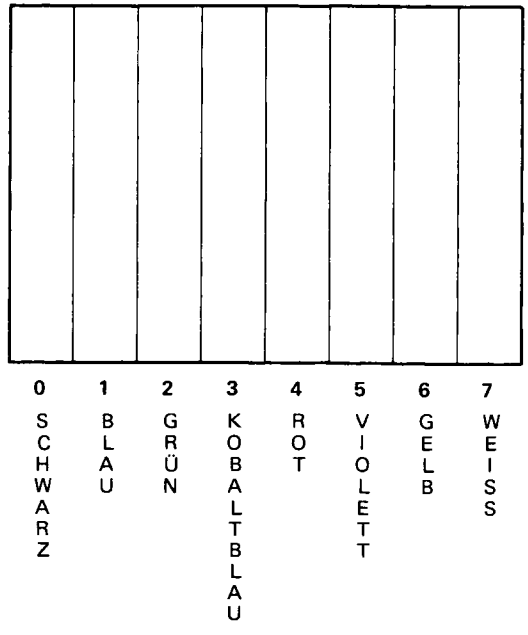
0 = Schwarz	4 = Rot
1 = Blau	5 = Violett
2 = Grün	6 = Gelb
3 = Kobaltblau	7 = Weiß

Bei der Benutzung von Farben muß der Benutzer wissen, wie diese Farben abgespeichert sind, insbesondere wenn er ein Bildschirmformat ausdrucken möchte. Diese Information kann die Entscheidung für die bei den Abbildungen zu benutzenden Farben beeinflussen.

Die verschiedenen Farben sind in verschiedenen Speichern abgelegt und nur ein Speicherinhalt kann gleichzeitig ausgedruckt werden. Deshalb kann je nach Benutzung der Farben ein vollständiges Abbild ausgedruckt werden oder auch nicht. In der nachfolgenden Tabelle wird angegeben, in welchem Speicher eine Farbe gespeichert ist. Hierbei ist zu beachten, daß die Farben nach Primärfarben unterteilt sind, und daß einige Farben in mehr als einem Speicher abgelegt sind.

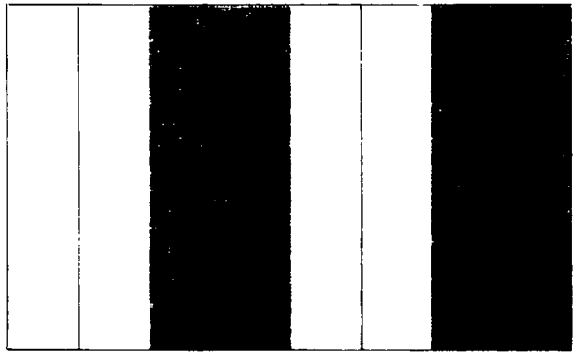
Speicher 1	Speicher 2	Speicher 4
Blau	Grün	Rot
Kobaltblau	Kobaltblau	
Violett		Violett
		Gelb
Weiß	Gelb	Weiß
	Weiß	

Das Bildschirmformat läßt sich ausdrucken, indem der auszudruckende Speicher entweder mit der GW-BASIC Konfigurier-Routine oder durch Aufnahme einer besonderen Codierung in dem Benutzerprogramm angegeben wird. (Beide Methoden sind später in diesem Kapitel noch beschrieben). Nun soll von folgender Abbildung auf dem Bildschirm ausgegangen werden. Jede der acht Farben wird durch einen senkrechten Strich dargestellt.

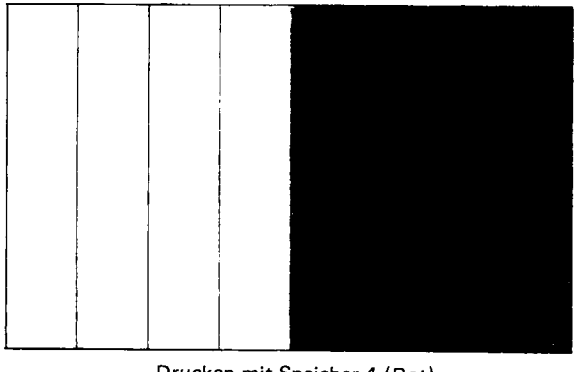


Nun wollen wir betrachten, wie das Abbild je nach angegebenem Speicher ausgedruckt wird.

Drucken mit Speicher 1 (Blau)



Drucken mit Speicher 2 (Grün)



Drucken mit Speicher 4 (Rot)

MUSIKAUSWAHL

Die GW-BASIC Erweiterung umfaßt auch Anweisungen, mit denen Musik gespielt (oder einfach ein Geräusch produziert) werden kann. Hier handelt es sich um die Beep-, Sound- und Play-Anweisungen. Unter den Anweisungen ist die Play-Anweisung die leistungsfähigste, da mit ihr ein ganzes Musikstück mit einer Anweisung erstellt werden kann. Durch Sound wird dagegen eine einzelne Note erzeugt, während Beep entsprechend seinem Namen fungiert, d.h. einen Piepton erzeugt.

DATENÜBERTRAGUNG

Mit GW-BASIC kann mit anderen Computern, einem Drucker oder einer anderen Einheit kommuniziert werden, welche eine asynchrone RS-232 Schnittstelle benutzt. Zur Implementierung der Datenübertragung muß die Übertragungseinheit zuerst mit der GW-BASIC Konfigurier-Routine beschrieben werden (siehe "Benutzen von GW-BASIC"). Danach wird eine neue Form der Open-Anweisung benutzt. Die anderen Eingabe- und Ausgabeprozeduren entsprechen im allgemeinen den Prozeduren für eine Datei, sind jedoch im einzelnen noch in Kapitel 3, Datenübertragung, beschrieben.

BENUTZEN VON GW-BASIC

Dieser Abschnitt beschreibt die Benutzung von GW-BASIC. Im einzelnen wird erläutert, wie das System konfiguriert, das Programm geladen und wie ein Ausdruck des Bildschirmformats erhalten wird. Die GW-BASIC Software enthält auch Hilfen, mit denen der Benutzer seine Programme schneller erstellen und editieren kann. Diese Hilfseinrichtungen umfassen vorprogrammierte Funktionstasten und einen Gesamtbildschirmeditor.

VORPROGRAMMIERTE FUNKTIONSTASTEN

Beim Laden von GW-BASIC wird am unteren Ende des Bildschirms in Zeile 25 eine Anzeige ausgegeben. Die Anzeige kennzeichnet die programmierbaren Funktionstasten, die GW-BASIC benutzt. Diese Funktionstasten befinden sich in der obersten Tastenreihe der Tastatur und tragen die Kennzeichen F1 bis F20.

F1 LOAD"	F6 EDIT	F11 GOTO	F16 CHAR$(
F2 RUN	F7 TRON	F12 GOSUB	F17 STRING$(
F3 CONT	F8 TROFF	F13 IF	F18 LINE(
F4 SAVE"	F9 PRINT	F14 THEN	F19 CIRCLE(
F5 LIST	F10 PRINT USING	F15 ELSE	F20 DRAW

Durch Benutzung der programmierbaren Funktionstasten werden sich wiederholende Tastatureingaben vermieden. So kann beispielsweise ein Programm einfach durch Betätigung der Funktionstaste F5 aufgelistet werden.

Die Definition der programmierbaren Funktionstasten ist veränderbar. Außerdem kann die Anzeige auf Zeile 25 unterdrückt werden. (Für weitere, genauere Informationen sei auf die Key-Anweisung verwiesen).

GESAMTBILDSCHIRMEDITOR

Wie BASIC arbeitet GW-BASIC entweder im direkten oder indirekten Modus und benutzt dieselben Programmiervereinbarungen. So gelten beispielsweise die BASIC-Regeln für Datentypen, Dateneingabe und Programmzeilen auch in GW-BASIC. (Allgemeine Regeln und Informationen über BASIC stehen in Kapitel 1 des MS BASIC BENUTZERHANDBUCHS).

Bei GW-BASIC steht jedoch ein Gesamtbildschirmeditor zur Verfügung. Diese Einrichtung bedeutet einfach, daß jede Textzeile *an beliebiger Stelle auf dem Bildschirm* schnell editiert werden kann.

In Tabelle 1 sind die Tasten angeführt, mit denen die Bewegung der Schreibmarke gesteuert werden kann. In einigen Fällen kann zwischen verschiedenen Tasten gewählt werden, je nachdem, welche Taste für die Eingabe am bequemsten ist. Muß eine Tastenkombination benutzt werden (wie bei CONTROL-J), so muß die Ctl-Taste gedrückt gehalten und die zweite Taste betätigt werden.

Neben der uneingeschränkten Bewegung auf dem Bildschirm läßt der Editor auch ein effizienteres Editieren zu. Zur Änderung bestehender Programmzeilen wird die LIST-Anweisung benutzt, wobei unbedingt die Abschlußtaste zu betätigen ist, um die geänderte Zeile in dem Programm zu speichern.

- Gelegentlich kann GW-BASIC in den Direktmodus zurückkehren, wobei die Schreibmarke auf einer Zeile mit einer Meldung, wie beispielsweise OK, steht. In diesem Fall wird die Zeile automatisch gelöscht. Würde sie nicht gelöscht und würde die Abschlußtaste betätigt, so würde die Meldung an GW-BASIC zur Interpretation übergeben, und es käme zu einem Syntaxfehler. BASIC-Meldungen sind mit dem hexadezimalen FF zu beenden, um sie vom Benutzertext zu unterscheiden.
- Nach Änderung einer Zeile braucht die Schreibmarke nicht zum Ende der logischen Zeile bewegt zu werden, bevor die Abschlußtaste betätigt wird. Der Editor weiß, wo jede logi-

sche Zeile endet und überträgt die gesamte Zeile, selbst wenn die Abschlußtaste am Anfang einer Zeile betätigt wird.

Der Editor kann auch während der Programmausführung eingesetzt werden. Kommt es zu einem Syntaxfehler, so geht GW-BASIC automatisch bei der Zeile, die den Fehler verursacht hat, in den Editier-Modus. Zum Beispiel:

```
10 A = 2$12
RUN

Y Syntax Error in 10
10 A = 2$12
```

Der Editor zeigt die fehlerhafte Zeile an und setzt die Schreibmarke unter die Ziffer 1. Der Benutzer würde dann die Schreibmarke zu dem Dollarzeichen ($) bewegen und dieses in ein Pfeilzeichen nach oben (^) ändern, gefolgt von der Betätigung der Abschlußtaste. Die korrigierte Zeile wird nun wieder in das Programm zurückgespeichert.

In diesem Beispiel führt das Zurückspeichern der Zeile in das Programm zu einem Verlust sämtlicher Variablen. Soll der Inhalt einiger Variablen vor Durchführung der Änderung überprüft werden, so wird die UNTERBRECHUNGS-Taste betätigt, um in den Direktmodus zurückzukehren. Die Variablen würden gesichert, da sich keine Programmzeile geändert hat. Nachdem der Benutzer mit dem Ergebnis zufrieden ist, könnte die Zeile editiert und das Programm erneut ausgeführt werden.

KONFIGURATION FÜR GW-BASIC

Bevor GW-BASIC aufgerufen wird, muß unter Umständen zuerst die GWCONF-Routine ausgeführt werden. Mit der GWCONF-Routine, die auf der GW-BASIC Diskette steht, werden die Konfigurationsinformationen für GW-BASIC definiert. GWCONF ist vor allem unter einem der folgenden Umstände auszuführen:

- Für das Ausdrucken von Grafiken wird ein Drucker benutzt.
- Entweder der rote oder der blaue Speicher soll für das Ausdrucken benutzt werden.
- Die Datenübertragung wird benutzt.

Der Drucker ist zu definieren, selbst wenn der schon mit der Configure-Routine von MS-DOS definiert wurde: GW-BASIC benötigt zusätzliche Informationen.

TASTE(N) (Hexadezimal-/ Dezimalwert)	NAME	BESCHREIBUNG
↵ oder Ctl-M (0D/13)	ABSCHLUSS	Sendet die Zeile (bis zur Position der Schreibmarke) zur Interpretation von GW-BASIC.
Ctl-J (0A/10)	ZEILEN-VORSCHUB	Bewegt die Schreibmarke zu der ersten Stelle auf der nächsten Zeile. (Steht die Schreibmarke in Zeile 24, so wird weitergeblättert.)
↖ oder Ctl-K (0B/11)	SCHREIBMARKE IN AUSGANGS-POSITION	Bewegt die Schreibmarke zu der oberen linken Ecke des Bildschirms
↑ oder Ctl-^ (1E/30)	SCHREIBMARKE NACH OBEN	Bewegt die Schreibmarke um eine Zeile nach oben.
oder Ctl-0 (1F/31)	SCHREIBMARKE NACH UNTEN	Bewegt die Schreibmarke um eine Zeile nach unten.
← oder Ctl-] (1D/29)	SCHREIBMARKE NACH LINKS	Bewegt die Schreibmarke um eine Stelle nach links. Wird die Schreibmarke über den linken Rand des Bildschirms hinausbewegt, so springt sie an das rechte Ende der vorhergehenden Zeile.
→ oder Ctl-\ (1C/28)	SCHREIBMARKE NACH RECHTS	Bewegt die Schreibmarke um eine Stelle nach rechts. Wird die Schreibmarke über den rechten Rand des Bildschirms hinausbewegt, so springt sie an das linke Ende der nächsten Zeile.
Ctl-→ oder Ctl-F (06/06)	NÄCHSTES WORT	Bewegt die Schreibmarke nach rechts zu dem nächsten Wort. "Nächstes Wort" ist das nächste Zeichen rechts von der Schreibmarke (oder auf der nächsten Zeile) auf dem Zeichenvorrat A—Z oder 0—9.
Ctl-← oder Ctl-B (02/02)	VORHER-GEHENDES WORT	Bewegt die Schreibmarke nach links zu dem vorhergehenden Wort. "Vorhergehendes Wort" ist das nächste Zeichen links von der Schreibmarke (oder auf der vorhergehenden Zeile) auf dem Zeichenvorrat A—Z oder 0—9.
← oder Ctl-H (08/08)	RÜCKSCHRITT	Löscht das zuletzt eingegebene Zeichen oder das Zeichen links von der Schreibmarke. (Steht das Zeichen in der ersten Spalte, so läuft die Schreibmarke vom Bildschirm). Sämtliche Zeichen rechts von der Schreibmarke werden um eine Stelle nach links bewegt. Geht eine logische Zeile über eine physische Zeile hinaus, so werden die nachfolgenden Zeichen nach links und oben geschoben, um die Zeile aufzufüllen.

TASTE(N) (Hexadezimal-/ Dezimalwert)	NAME	BESCHREIBUNG
Ctl-E (05/05)	LÖSCHEN BIS ZUM ENDE	Löscht von der aktuellen Position der Schreibmarke bis zum Ende einer logischen Zeile.
CTL-↖ oder Ctl-L (0C/12)	BILDSCHIRM LÖSCHEN	Löscht den Bildschirm und setzt die Schreibmarke in die obere linke Ecke des Bildschirms.
Ctl-U (15/21)	ABGABE	Löscht die gesamte logische Zeile
Ctl-C (03/03)	UNTER-BRECHUNG	Kehrt in den Direktmodus zurück, ohne eventuell an der aktuellen Zeile vorgenommene Änderungen zu sichern.
Ctl-T (14/20)	ANZEIGE DER FUNKTIONS-TASTEN	Schiebt die Anzeige der Funktionstasten in Zeile 25.
Ctl-Q (11/17)	ZEILEN-MARKIERUNG	Markiert eine Zeile für das Löschen.
Ctl-R (12/18)	EINFÜGEN	Schaltet den Einfügemodus ein oder aus. (Mit dem Einfügemodus werden Zeichen zwischen die Zeichen auf einer Zeile gesetzt.) Ist der Einfügemodus ausgeschaltet, so wird er durch Betätigung dieser Taste eingeschaltet. Ist der Einfügemodus eingeschaltet, so wird er durch Betätigung dieser Taste ausgeschaltet. Wird im Einfügemodus gearbeitet, so werden die Zeichen neben der Schreibmarke nach rechts geschoben, wenn eingegebene Zeichen in der aktuellen Position eingefügt werden. Bei jeder Tastenbetätigung wird die Schreibmarke um eine Stelle nach rechts bewegt. Laufen Zeichen (oder Leerzeichen) von der rechten Seite des Bildschirms, so werden sie auf der linken Seite der nachfolgenden Zeile wieder eingefügt. Wird nicht im Einfügemodus gearbeitet, so ersetzen die eingegebenen Zeichen die bestehenden Zeichen auf der Zeile.
Ctl-I (09/09)	TAB	Wird nicht im Einfügemodus gearbeitet, so wird die Schreibmarke durch Betätigung dieser Taste über die Zeichen hinwegbewegt, bis der nächste Tabulatorstopp erreicht ist. Tabulatorstopps sind in Abständen von 8 Zeichen gesetzt. Wird im Einfügemodus gearbeitet, so werden durch Betätigun dieser Taste Leerzeichen von der aktuellen Position der Schreibmarke bis zum nächsten Tabulatorstopp eingefügt.
Ctl-N (0E/14)	ENDE	Bewegt die Schreibmarke zum Ende der logischen Zeile. Ab dieser Position eingegebene Zeichen werden an die Zeile angehängt.

Tabelle 1 Editiertasten

Die folgende Tabelle zeigt, welche Informationen mit GWCONF angegeben werden können. Hier ist zu beachten, daß bei der Auswahl eines Druckers mit serieller Schnittstelle bestimmte Definitionen automatisch erstellt werden.

	Ursprüngliche Definition	**mit GWCONF**
Drucker	Keiner	EPSON FX80 ITOH M8510A (bidirektional) ITOH M8510A (eindirektional) Keiner
Serielle Druckerschnittstelle:		
– Stoppbits	1	1 1/2 oder 2
– Parität	Gerade	Deaktiviert oder ungerade
– Zeichenlänge	7 Bits	5, 6 oder 8
– Baudrate	9.600	50 bis 19.200
Speicher ausdruck	Grün	Rot oder Blau
Anschlußadressen für die Datenübertragung	Anschluß 1 = 00 hex Anschluß 2 = 00 hex	30, 38, 60, 68, 70, 78 B0, B8, C0, C8

Zur Benutzung von GWCONF wird die BW-BASIC Diskette eingelegt. (MS-DOS muß schon geladen sein).

HINWEIS: Die GW-BASIC Diskette kann entweder in Laufwerk A oder in Laufwerk B eingelegt werden. Bei Systemen mit einem einzigen Diskettenlaufwerk sind die Disketten auszutauschen.

Nach Eingabe von GWCONF wird folgende Bildschirmanzeige ausgegeben:

```
1)  Select Printer
2)  Modify Color for Screendump
3)  Modify Communication Ports
4)  Exit Program
* Enter Your Selection
```

Bei jeder Eingabe führen weitere Bildschirmanzeigen den Benutzer durch die Definition. Obwohl sich die Bildschirmanzeigen selbst erläutern, sind einige Benutzungsregeln unter Umständen hilfreich. (In dieser Beschreibung wird jedoch davon ausgegangen, daß der Benutzer die Anschlußadressen für die Konfiguration kennt. Diese Information wird in dem "System Technical Manual, Teil 1, Hardware" näher erläutert).

Für den Bildschirmauszug kann jeder der drei Speicher ausgewählt werden, es kann jedoch nur einer gleichzeitig ausgedruckt werden. Sollen alle drei Abbilder gedruckt werden, so muß deshalb vor jeder Druckoperation GWCONF benutzt werden.

Es gibt noch eine Alternativmethode für die Angabe des auszudruckenden Speichers. Bei dieser Technik übernimmt das Programm die Auswahl der Farben. (Siehe "Auswahl des Speicherausdrucks innerhalb des Programms").

Die Funktion für das Beenden des Programms wird nach Beendigung der Änderungen benutzt. Wird diese Funktion angegeben, so nimmt die Software folgende Anzeige vor:

```
1) Update O.S. disk in drive A
2) Return to main program
3) Exit CONFIG

*  Enter Function
```

Funktion 1 wird benutzt, wenn die neuen Konfigurationsinformationen auf die MS-DOS Master-Diskette geschrieben werden sollen. In diesem Fall muß die MS-DOS Diskette unbedingt in Laufwerk A eingelegt sein. Sind die Änderungen nur vorübergehender Natur (für die aktuelle Ausführung) bestimmt), so wird Funktion 3 benutzt. Die Änderungen werden nur im Speicher vorgenommen.

Nach Ausführung von GWCONF kann GW-BASIC geladen werden.

LADEN VON GW-BASIC

Für das Laden von GW-BASIC wird die GW-BASIC Diskette eingelegt und

```
GW-BASIC
```

eingegeben. Um wieder in das MS-DOS Betriebssystem zurückzukehren, ist als Antwort auf die BASIC-Eingabeaufforderung SYSTEM einzugeben.

```
OK?
SYSTEM
A >
```

AUSWAHL DES SPEICHERAUSDRUCKS INNERHALB DES PROGRAMMS

Die GW-BASIC Diskette enthält ein Objektmodul, mit dem die Steuerung über das Ausdrucken von Farbgrafiken an das Programm übergeben wird. Das Modul trägt den Namen DUMPCL (dump color). Mit ihm kann innerhalb eines Programms ein auszudruckender Farbspeicher angegeben werden (1, 2, 4).

Bevor DUMPCL benutzbar ist, muß es als separate Datei auf der GW-BASIC Diskette erstellt werden. Dies wird über MS-LINK (siehe MS-DOS Benutzerhandbuch) und DEBUG erreicht, mit denen die Adresse des DUMPCL-Moduls gelesen werden kann. Diese beiden Befehle stehen auf der MS-DOS Diskette. Steht ein einziges Diskettenlaufwerk zur Verfügung, so ist die nachfolgend beschriebene Reihenfolge einzuhalten, um DUMPCL als Datei zu erstellen. Sind zwei Diskettenlaufwerke vorhanden, so müssen die in jedem Schritt in Klammern angegebenen Instruktionen befolgt werden.

1. Die MS-DOS Diskette einlegen. (Bei zwei Diskettenlaufwerken wird die MS-DOS Diskette und die GW-BASIC Diskette eingelegt.)
2. Das System zeigt A> an. LINK EINGEBEN. (Bei zwei Diskettenlaufwerken werden die Operationen mit Laufwerk B ausgeführt. B: eingeben und nach Anzeige von B> durch das System wird A:LINK eingegeben.)
3. Das System zeigt:

   Microsoft Object Linker V2.00
   (c) Copyright 1982 by Microsoft Inc.

   Object Modules [.OBJ]:

   an. Die GW-BASIC Diskette einlegen. DUMPCL eingeben. (Bei zwei Diskettenlaufwerken wird einfach DUMPCL eingegeben.)
4. Das System zeigt:

   Run File [DUMPCL.EXE]:

   an. /H eingeben, um die höchste Adresse im Speicher anzugeben. (Bei zwei Diskettenlaufwerken wird B:/H eingegeben.)
5. Das System zeigt:

   List File [NUL.MAP]:

   an. Abschlußtaste betätigen.

6. Das System zeigt:

   Libraries [.LIB]:

   an. Abschlußtaste betätigen.
7. Das System zeigt:

   Warning: No STACK segment

   There was 1 error detected.

   A>

   an. Die Warnung und der entdeckte Fehler beeinflussen die Prozedur nicht. Die MS-DOS Diskette einlegen und DEBUG eingeben. (Bei zwei Diskettenlaufwerken lautet die Eingabeaufforderung B>. A: DEBUG DUMPCL.EXE eingeben. Die DUMPCL.EXE-Datei wird direkt in DEBUG geladen.
8. Das System zeigt einen Bindestrich (—) an. Die GW-BASIC Diskette einlegen und NDUMPCL.EXE eingeben. Dadurch wird die neue DUMPCL.EXE-Datei für DEBUG angegeben. (Bei zwei Diskettenlaufwerken wird R eingegeben, um sämtliche Register anzuzeigen. Danach wird zu Schritt 11 gegangen.)
9. Das System zeigt einen Bindestrich (—) an. L. eingeben, um DUMPCL.EXE in DEBUG zu laden.
10. Das System zeigt einen Bindestrich (—) an. R eingeben, um sämtliche Register anzuzeigen.
11. Das System gibt folgende Anzeige aus:

```
AX=FFFF  BX=0000  CX=0026  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0AA0  ES=0AA0  SS=1FE0  CS=1FE0  IP=0000  NV UP DI PL NZ NA PO NC
1FE0:0000 55              PUSH     BP
```

    An dieser Stelle muß die Adresse CS=1FE0 beachtet werden. CS stellt die Adresse des DUMPCL-Moduls dar. Die hier angegebene Adresse für CS muß für den jeweiligen Rechner festgehalten werden. Sie muß für den späteren Gebrauch aufgeschrieben werden.
12. Das System zeigt einen Bindestrich (—) an. NGWBASIC.EXE eingeben. Dadurch wird die GWBASIC.EXE-Datei angegeben.
13. Das System zeigt einen Bindestrich (—) an. L eingeben, um GWBASIC.EXE in den Speicher zu laden.
14. Das System zeigt einen Bindestrich (—) an. G eingeben, um in GWBASIC.EXE zu gehen.

Nun befinden wir uns in GW-BASIC.

15. Die folgenden Zeilen eingeben:

```
DEF SEG=&H1FE0
BSAVE "DUMPCL.COM",&H0,&H30
```

Hier muß beachtet werden, daß es sich bei dem eingegebenen DEF SEG-Wert um die Adresse des DUPMCL-Moduls handelt. An dieser Stelle muß der Wert für CS eingegeben werden, der in dem Rechner angezeigt wurde (Schritt 11). DUMPCL.COM in der zweiten Zeile ist der neue, zu sichernde Dateiname. Die letzten beiden Einträge der zweiten Zeile stellen die Abstandsadresse von der in der DEF SEG-Anweisung angegebenen Adresse und die Länge der zu sichernden Datei in Bytes dar (die DUMPCL-Datei hat immer die hier angegebene Länge). In diesem Zusammenhang wird auf die BSAVE-Anweisung in MS-DOS Erweiterung, Abschnitt 2, verwiesen.

Die Datei ist nun auf der Diskette gesichert. GW-BASIC wird beendet, indem SYSTEM eingegeben wird. DEBUG wird beendet, indem Q eingegeben wird. Nun kann GW-BASIC geladen werden.

Nachdem die DUMPCL-COM-Datei auf der Diskette steht, kann sie für die Steuerung des Ausdruckens von Farbgrafiken durch das Programm benutzt werden. Zur Benutzung dieser Datei müssen die folgenden BASIC-Anweisungen in das Programm aufgenommen werden:

```
10 DEF SEG=&H1FE0
20 BLOAD"DUMPCL.COM",&H0
30 A%=1
40 CALL &H0(A%)
50 LCOPY
```

In Zeile 10 wird die Adresse des DUMPCL-Moduls eingegeben, die bei Schritt 11 angezeigt wurde. In Zeile 30 wird der Farbspeicher angegeben, der ausgedruckt werden soll. Für einen blauen Vordergrund wird 1, für einen grünen Vordergrund 2 und für einen roten Vordergrund 4 eingegeben.

ANWEISUNGEN UND FUNKTIONEN

Die folgenden Anweisungen und Funktionen für GW-BASIC unterscheiden sich in der Implementierung von der Beschreibung in dem MS-BASIC Benutzerhandbuch. Jede Beschreibung weist folgendes Format auf:

Format: Zeigt das richtige Format für die Instruktion an. Die Formatschreibweise wird nachfolgend angegeben.

Zweck: Erläutert die Benutzung der Instruktion.

Bemerkungen: Zeigt Beispielprogramme oder Programmsegmente, die die Benutzung der Instruktion verdeutlichen.

Die Formatschreibweise ist dieselbe wie in MS-BASIC und MS-DOS Erweiterung:

[ ] Eckige Klammern geben an, daß der Eintrag wahlweise ist.

< > Spitze Klammern geben vom Benutzer eingegebene Daten an. Steht in den spitzen Klammern ein Text in Kleinbuchstaben, so muß der Benutzer eine vom Text definierte Eingabe vornehmen, z.B. <Dateiname>. Steht in den spitzen Klammern ein Text in Großbuchstaben, so muß der Benutzer die in dem Text angegebene Taste betätigen, z.B. <RETURN>-Taste (Abschlußtaste).

Geschweifte Klammern geben an, daß der Benutzer zwischen zwei oder mehr Einträgen wählen kann. Mindestens einer der in geschweiften Klammern stehenden Einträge muß ausgewählt werden, es sei denn, die Einträge selbst stehen in eckigen Klammern.

| Senkrechte Trennstriche trennen die Auswahlmöglichkeit innerhalb der geschweiften Klammern.

Mindestens einer der durch senkrechte Trennstriche getrennten Einträge muß ausgewählt werden, es sei denn, die Einträge stehen selbst in eckigen Klammern.

. . . Auslassungszeichen geben an, daß ein Eintrag beliebig oft wiederholt werden kann.

GROSS Großbuchstaben geben Teile von Anweisungen oder Befehlen an, die genau wie dargestellt eingegeben werden müssen.

Alle anderen Satzzeichen, wie Kommas, Doppelpunkte, Schrägstriche und Gleichzeichen müssen genau wie dargestellt eingegeben werden.

BEEP ANWEISUNG

Format: BEEP

Zweck: Läßt den Lautsprecher mit 830 Hz für die Dauer von 250 ms ertönen.

Bemerkungen: Nichtgrafikversionen von MS-BASIC benutzen PRINT CHR$ (7), um ein ASCII-Klingel-Zeichen zu senden.

Beispiel: 2430 IF X < 20 THEN BEEP 'X liegt außerhalb des Bereichs, entsprechenden Hinweis machen.

CIRCLE-ANWEISUNG

Format: CIRCLE (x,y), Radius [,Farbe [,Anfang,Ende[,Aspekt]]]

Zweck: Zeichnen einer Ellipse nach den folgenden Definitionen:

x,y
Gibt die Koordinaten der Mitte der Ellipse an.

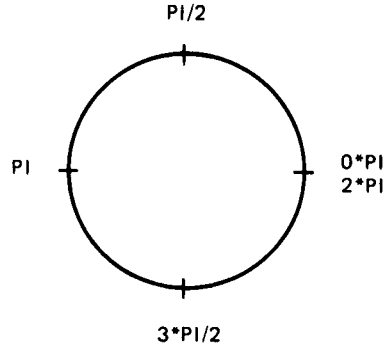
Radius
Gibt den Radius (Hauptachse) in Punkten an.

Farbe
Gibt die Farbe des Kreises (0—7, siehe Color-Anweisung) an. Werden keine Angaben gemacht, so ist die Farbe die Vordergrundfarbe.

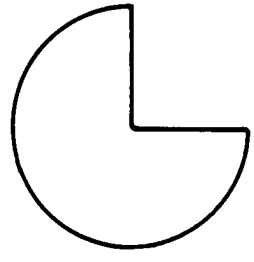
Anfang,Ende
Gibt in Radiant an, wo die Zeichnung anfangen und aufhören soll. Die Werte können von -2*PI bis 2*PI gehen, wobei PI=3.141593 ist. (Siehe auch Bemerkungen.)

Aspekt
Gibt das Verhältnis des X-Radius zum Y-Radius an. (Werden keine Angaben gemacht, so wird von einem Verhältnis von 1/1, einem Kreis, ausgegangen.) Ist das Verhältnis kleiner als 1, so ist der Radius der X-Radius. Ist das Verhältnis größer als 1, so ist der Radius der Y-Radius.

Bemerkungen: Die beiden ersten Parameter (x,y Koordinaten und Radius) sind die einzigen für das Zeichnen eines Kreises erforderlichen Parameter. Die beiden letzten Parameter werden für das Zeichnen anderer "kurvenförmiger" Formen benutzt. Mit Anfang und Ende kann der Benutzer beispielsweise steuern, wieviel des Kreises gezeichnet werden soll. Die Werte von Anfang und Ende werden in Radiant in der mathematischen Standardart angegeben.



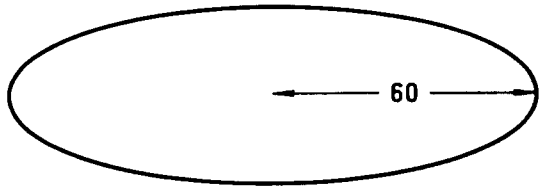
Entweder der Anfangs- oder der Entwert kann negativ sein (-0 ist jedoch nicht zulässig). In diesem Fall wird der Winkel mit einer Linie mit dem Mittelpunkt verbunden. Durch die Anfangs- und Endwerte von -PI/2, -PI würde ein Teil eines Kreises gezeichnet.



Mit dem Aspekt-Parameter kann eine andere Ellipse als ein Kreis gezeichnet werden. Ist das Aspekt-Verhältnis kleiner als 1, so ist r der X-Radius. Ist das Aspekt-Verhältnis größer als 1, so ist r der Y-Radius. Zum Beispiel zeichnet:

```
10 SCREEN 1
20 CIRCLE (160m100), 60 , , ,5/18
```

eine Ellipse wie die folgende:



Punkte, die nicht mehr auf dem Bildschirm liegen, werden von der Circle-Anweisung nicht gezeichnet.

CLS-ANWEISUNG

Format: CLS

Zweck: Löscht die Bildschirmanzeige auf die gerade ausgewählte Hintergrundfarbe.

Bemerkungen: Die Bildschirmanzeige kann auch mit den Tasten Ctl-L oder Ctl-HOME gelöscht werden. (Mit der Screen-Anweisung kann der Bildschirm ebenfalls gelöscht werden.)

Ist die KEY ON-Anweisung wirksam, während die CLS-Anweisung benutzt wird, so wird der Bildschirm gelöscht; die Funktionszeile am unteren Ende des Bildschirms wird jedoch mit den gerade aktiven Vordergrund-/Hintergrundfarben erneuert.

COLOR-ANWEISUNG

Format: COLOR [Vordergrund] [,Hintergrund]

Vordergrund
Gibt die Farbe für die Zeichen an. An dieser Stelle wird eine vorzeichenlose Ganzzahl in dem Bereich von 0—7 angegeben (siehe nachfolgende Tabelle).

Hintergrund
Gibt die Farbe für den Bildschirm an. An dieser Stelle wird eine vorzeichenlose Ganzzahl in dem Bereich von 0—7 eingegeben (siehe nachfolgende Tabelle).

Zweck: Ändert entweder die Vordergrund- oder Hintergrundfarbe (oder beide). Die Farben werden durch Codes angegeben:

0 = Schwarz
1 = Blau
2 = Grün
3 = Kobaltblau
4 = Rot
5 = Violett
6 = Gelb
7 = Weiß

Bemerkungen: Einer der Parameter kann weggelassen werden. In diesem Fall wird von dem Wert einer vorhergehenden COLOR-Anweisung (sofern vorhanden) ausgegangen.

Die Vordergrund- und Hintergrundfarben können identisch sein. In diesem Fall sind die Zeichen unsichtbar.

Bei der Umwandlung von Programmen muß beachtet werden, daß NCR GW-BASIC einen dritten Parameter und die Codes 8—31 zuläßt, ohne eine Fehlermeldung anzuzeigen. Außerdem unterscheidet sich auch die Syntax für Text- und Grafikmodus nicht.

Ein unzulässiger Parameterwert führt zu der Fehlermeldung "Illegal Function Call". Die Bildschirmfarben bleiben so, wie sie vor Eingabe der Anweisung waren.

Beispiele:

10 COLOR 4,7	Benutzt rote Zeichen auf weißem Hintergrund.
20 COLOR ,,4	Ändert die Hintergrundfarbe auf rot; die Zeichen sind unsichtbar.
30 COLOR 2,0	Benutzt grüne Zeichen auf schwarzem Hintergrund.

CSRLIN-FUNKTION

Format: X = CSRLIN

X
Gibt eine beliebige numerische Variable an, für die die Software einen Wert in dem Bereich von 1 bis 24 zurückgibt.

Zweck: Gibt die aktuelle Position der Schreibmarke auf der Zeile an.

Beispiel: In dem nachfolgenden Beispiel gibt die Anweisung in Zeile 10 die aktuelle Zeilenposition an. In Zeile 20 gibt die Anweisung die aktuelle Spaltenposition an. In Zeile 30 wird HELLO in der Mitte des Bildschirms geschrieben. Und in Zeile 40 wird die Position der Schreibmarke auf die vorhergehende Zeile und Spalte zurückgesetzt. Die Software gibt für X = POS(O) einen Wert in dem Bereich von 1 bis 80 zurück.

```
10 Y = CSRLIN
20 X = POS(O)
30 LOCATE 12,40:PRINT "HELLO"
40 LOCATE Y,X
```

DRAW-ANWEISUNG

Format: DRAW <Zeichenfolgenausdruck>

Zweck: Zeichnet ein Objekt, so wie es von dem Zeichenfolgenausdruck angegeben wird.

Bemerkungen: Mit der DRAW-Anweisung kann ein Objekt mit Befehlen in der Objektdefinitionssprache gezeichnet werden. Ein Sprachbefehl besteht aus einem einzigen Zeichen innerhalb einer Zeichenfolge, auf das wahlweise einer oder mehrere Parameter folgen. Der Zeichenfolgenausdruck definiert ein Objekt, das auf dem Bildschirm aufgezeichnet wird, wenn BASIC die DRAW-Anweisung ausführt.

Die folgenden Bewegungsbefehle beginnen die Bewegung ab den Koordinaten des letzten Punktes, der mit einem anderen Sprachbefehl, mit einer LINE-Anweisung oder einer PSET-Anweisung gezeichnet wurde. Wird ein Programm ausgeführt, so beginnt die Bewegung ab der Mitte des Bildschirms (320, 200).

U [<n>] Nach oben bewegen
D [<n>] Nach unten bewegen
L [<n>] Nach links bewegen
R [<n>] Nach rechts bewegen
E [<n>] Diagonal nach oben und rechts bewegen
F [<n>] Diagonal nach unten und rechts bewegen
G [<n>] Diagonal nach unten und links bewegen
H [<n>] Diagonal nach oben und links bewegen

Das n in den vorhergehenden Befehlen gibt die bei der Bewegung einzuhaltende Entfernung an. Die Anzahl von bewegten Punkten entspricht dem Skalierungsfaktor mal n (siehe S unten). Wird n nicht angegeben, so wird die Zeichnung durch die Befehle um eine Einheit bewegt.

M <x,y>
Absolute oder relative Bewegung (für eine Beschreibung der x- und y-Koordinaten wird auf Kapitel 1 verwiesen). Steht vor dem x ein + oder —, so wer-

den x und y zu den Koordinaten des zuletzt gezeichneten Punktes hinzugefügt und mit dem aktuellen Punkt durch eine Linie verbunden. Wird kein + oder — hinzugefügt, so wird eine Linie von dem aktuellen Punkt zu dem Punkt (x,y) gezeichnet.

Die folgenden Vorspannbefehle können vor einem der obigen Bewegungsbefehle stehen:

B
Bewegen, jedoch keine Punkte zeichnen.

N
Bewegen, nach Ausführung jedoch zu der Originalposition zurückkehren.

A <n>
Winkel n festlegen. n kann von 0 bis 3 gehen, wobei 0 0^{o}, 1 90^{o}, 2 180^{o} und 3 270^{o} entspricht.

C <n>
Farbe n festlegen. n kann von 0 bis 7 gehen (siehe Color-Anweisung).

S <n>
Skalierungsfaktor festlegen. n kann von 1 bis 255 gehen. Der Skalierungsfaktor, multipliziert mit der in den U-, D-, L-, R-, E-, F-, G-, H- und M-Befehlen angegebenen Entfernung ergibt die tatsächliche Entfernung.

X <Zeichenfolge>
Ausführung einer Teilfolge. Mit diesem Befehl kann eine zweite Teilfolge einer Zeichenfolge ausgeführt werden, wie GOSUB in BASIC. Bei den Parametern kann es sich um Konstanten wie 123 oder =Variable handeln, wobei Variable den Namen einer Variablen darstellt.

Beispiele: Für das Zeichen eines Dreiecks werden folgende Befehle eingegeben:

```
10 SCREEN 1
20 DRAW "E60;F60;L120"
```

Für das Zeichen eines Kästchens werden folgende Befehle eingegeben:

```
10 SCREEN 1
20 V = 100
30 DRAW "U=v;R=V;D=V;L=v"
```

EDIT-ANWEISUNG

Format: EDIT <Zeilennummer>

Zeilennummer
Gibt die Zeilennummer einer Zeile in dem Programm an. Ist keine derartige Zeile vorhanden, so wird eine Fehlermeldung "Undefined Line Number" angezeigt.

Zweck: Anzeige einer Zeile für das Editieren.

Bemerkungen: Mit der EDIT-Anweisung wird einfach die angegebene Zeile angezeigt. Die Schreibmarke wird unter die erste Ziffer der Zeilennummer bewegt. Danach kann die Zeile mit den in dem Abschnitt "Gesamtbildschirmeditor" in Kapitel 1 beschriebenen Tasten geändert werden.

Ein Punkt (.) bezieht sich immer auf die aktuelle Zeile. Wurde gerade eine Zeile eingegeben und soll zurückgegangen werden, um die Zeile zu editieren, so kann EDIT. eingegeben werden, um die Zeile erneut anzuzeigen.

GET- UND PUT-ANWEISUNGEN

Formate: GET (x1,y1)—(x1,y2),<Matrixname>
PUT (x1,y1),<Matrixname> [,<Funktionsverb>]

Funktion: Die GET- und PUT-Anweisungen übertragen Grafikabbilder zu und von dem Bildschirm. Mit den Anweisungen können auch Trickbilder und eine schnelle Bildbewegung durchgeführt werden.

Bemerkungen: GET (x1,Y1)—(x2,y2),<Matrixname>

(x1,y1) und (x2,y2)
Koordinaten in der absoluten oder relativen Form (siehe Kapitel 1) der gegenüberliegenden Ecken eines Rechtecks.

Matrixname
Der vom Benutzer festgelegte Name der Matrix, die die Abbildinformationen aufnimmt.

GET liest die Farben der Punkte in dem Bildschirmformat, das auf die Rechteckgrenzen ausgerichtet ist, in eine Matrix. Das Rechteck wird auf dieselbe Art und Weise definiert, wie das von der LINE-Anweisung mit der Option ",b" gezeichneten Rechteck.

Die Matrix wird einfach für die Aufnahme des auf die Rechteckgrenzen ausgerichteten Abbilds benutzt. Die Matrix muß numerisch und groß genug sein, um das gesamte Abbild aufnehmen zu können. Die benötigte Matrixgröße in Bytes kann mit folgender Formel bestimmt werden.

$$\text{INT}\left(\frac{x+7}{8}\right) \frac{*\text{bits}*}{\text{pixel}}\ y+4$$

Wobei x die Länge einer Breitseite des Rechtecks und y die Länge einer Längsseite des Rechtecks darstellt. Bits pro Bildelement entspricht im Farbmodus 2 und im Schwarz/Weiß-Modus 1. Die Bytes pro Element einer Matrix sind:

2 für Ganzzahl
4 für einfache Genauigkeit
8 für doppelte Genauigkeit.

Soll beispielsweise die GET-Anweisung dazu benutzt werden, ein 10x12 Abbild in eine Matrix zu setzen, so ist folgende Anzahl von Bytes erforderlich:

INT $(\frac{10+7}{8})*3*12+4$ oder 76 bytes.

Legende:

Es wäre also eine ganzzahlige Matrix $(\frac{76}{2})$ von mindestens 38 Bytes erforderlich.

Das Speicherformat in der Matrix sieht folgendermaßen aus:

2 Bytes für Dimension x in Bits
2 Bytes für Dimension y in Bits
Matrixdaten.

Die Daten für jede Punktreihe sind linksbündig auf einer Bytegrenze ausgerichtet. Sind weniger als ein Vielfaches von 8 Bits gespeichert, so wird der Rest des Bytes mit Nullen aufgefüllt.

PUT (x1,y1),<Matrix>[,<Funktionsverb>]

(x1,y1)
Koordinaten der oberen linken Ecke des auf den Bildschirm zu übertragenden Abbilds. Ist das Abbild für den Bildschirm zu groß, so wird eine Fehlermeldung "Illegall Function Call" ausgegeben.

Matrix
Name der numerischen Matrix, die das zu übertragende Abbild enthält.

Funktionsverb
Wird für die Verbindung des übertragenen Abbilds mit dem Bildschirm benutzt. Gültige Einträge sind: PSET, PRESET, AND, OR oder XOR. Die Standardvorgabe ist XOR.

Mit der PUT-Anweisung wird das in der Matrix gespeicherte Abbild auf den Bildschirm übertragen.

PSET
Überträgt Daten aus der Matrix Wort für Wort auf den Bildschirm.

PRESET
Wie PSET, nur daß ein negatives Abbild erzeugt wird.

AND
AND darf nur benutzt werden, wenn das Abbild auf den Bildschirm übertragen werden soll und wenn auf dem Bildschirm schon ein Abbild vorhanden ist. Nur die Punkte, die in beiden Abbildern vorhanden sind, werden auf dem Bildschirm angezeigt.

OR
OR wird benutzt, um ein schon bestehendes Abbild durch das Abbild zu überlagern.

XOR
XOR ist die Standardvorgabe. Durch sie werden die Punkte auf dem Bildschirm an der Stelle umgekehrt, an der in dem Matrixabbild ein Punkt vorhanden ist. Mit XOR können auch Trickbilder erzeugt werden. Wird ein Abbild mit PUT zweimal gegen einen komplexen Hintergrund gesetzt, so bleibt der Hintergrund unverändert. Dadurch kann das Objekt auf dem Bildschirm hin- und herbewegt werden, ohne daß der Hintergrund gelöscht wird.

Trickbilder können in folgender Reihenfolge erstellt werden:

1. Mit XOR wird das Abbild auf den Bildschirm gesetzt.
2. Die neue Adresse des Abbilds wird berechnet.
3. Mit XOR wird das Abbild ein zweites Mal bei der ersten Adresse auf den Bildschirm gesetzt. Dadurch wird das Abbild von der ersten Adresse gelöscht.

4. Nun wird zurück zu Schritt 1 gegangen. Mit XOR wird das Abbild an die neue Adresse gesetzt.

Durch eine Bewegung auf diese Art und Weise bleibt der Hintergrund unverändert. Das Flimmern kann durch Verkürzen der Zeit zwischen den Schritten 4 und 1 und durch ausreichend Zeit zwischen den Schritten 1 und 3 reduziert werden.

Sollen Trickbilder von mehr als einem Abbild erstellt werden, so muß jedes Abbild separat Schritt für Schritt erstellt werden.

Ist die Sicherung des Hintergrunds nicht von Bedeutung, so kann ein Abbild mit PSET in ein Trickbild verwandelt werden. Es muß jedoch ein ausreichend großes Rechteck vorhanden sein, das sowohl das erste als auch das neue Abbild aufnehmen kann. Ist der Bereich groß genug, so löscht der zusätzliche Bereich das erste Abbild. Diese Methode ist an sich schneller als die Methode mit XOR, da nur eine PUT-Anweisung erforderlich ist, obwohl mit einem größeren Bereich gearbeitet werden muß. In dem nachfolgenden Beispiel legt Zeile 20 die Dimensionen des zu benutzenden Bildschirmbereichs fest. Mit Zeile 30 wird ein aufgefülltes Kästchen in der Farbe 6 gezeichnet. Mit Zeile 40 wird dieses Kästchen in eine Matrix gelesen. Die Zeilen 50 bis 90 setzen das Kästchen wieder auf den Bildschirm und verschieben es nach links.

```
10 SCREEN 1
20 DIM M(1000)
30 LINE (0,0)-(30,30),6,BF
40 GET (0,0)-(60,30),M
50 FOR I=579 TO 10 STEP -1
60 PUT (I,200),M,PSET
70 NEXT I
80 GOTO 60
```

AND

		Bildschirmfarbe							
		0	1	2	3	4	5	6	7
Matrixfarbe	0	0	0	0	0	0	0	0	0
	1	0	1	0	1	0	1	0	1
	2	0	0	2	2	0	0	2	2
	3	0	1	2	3	0	1	2	3
	4	0	0	0	0	4	4	4	4
	5	0	1	0	1	4	5	4	5
	6	0	0	2	2	4	4	6	6
	7	0	1	2	3	4	5	6	7

OR

Matrixfarbe	Bildschirmfarbe							
	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	1	3	3	5	5	7	7
2	2	3	2	3	6	7	6	7
3	3	3	3	3	7	7	7	7
4	4	5	6	7	4	5	6	7
5	5	5	7	7	5	5	7	7
6	6	7	6	7	6	7	6	7
7	7	7	7	7	7	7	7	7

XOR

Matrixfarbe	Bildschirmfarbe							
	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	3	2	5	4	7	6
2	2	3	0	1	6	7	4	5
3	3	2	1	0	7	6	5	4
4	4	5	6	7	0	1	2	3
5	5	4	7	6	1	0	3	2
6	6	7	4	6	2	3	0	1
7	7	6	5	4	3	2	1	0

KEY-ANWEISUNG

Format: KEY<Tastennummer>,<Zeichenfolgenausdruck>
KEY LIST
KEY ON
KEY OFF

Tastennummer
Gibt die Nummer der programmierbaren Funktionstaste in dem Bereich von 1 bis 20 an (siehe nachfolgende Liste).

Zeichenfolgenausdruck
Gibt den Zeichenfolgenausdruck an, der der programmierbaren Funktionstaste zugewiesen wird.

Zweck: Ermöglicht die Zuweisung eines Zeichenfolgenausdrucks an die programmierbaren Funktionstasten. Jeder oder einer der Tasten kann eine Zeichenfolge aus bis zu 15 Zeichen zugewiesen werden. Wird die Taste betätigt, so wird die entsprechende Zeichenfolge in BASIC eingegeben.

Bemerkungen: Ursprünglich sind den programmierbaren Funktionstasten die folgenden Werte zugeordnet:

F1 LOAD"	F11 GOTO⎵
F2 RUN ↵	F12 GOSUB⎵
F3 CONT ↵	F13 IF⎵
F4 SAVE"	F14 THEN⎵
F5 LIST⎵	F15 ELSE⎵
F6 EDIT⎵	F16 CHR$(
F7 TRON ↵	F17 STRING$(
F8 TROFF ↵	F18 LINE⎵ (
F9 PRINT⎵.	F19 CIRCLE⎵ (
F10 PRINT⎵USING⎵	F20 DRAW⎵

KEY ON
Hier handelt es sich um die ursprüngliche Einstellung, durch die die Tasten F1 bis F7 in Zeile 25 angezeigt werden. Zur Anzeige der sieben nächsten Tasten wird Ctl—T betätigt. Zur Anzeige der sechs letzten Tasten wird Ctl-T erneut betätigt. Um die Folge erneut zu starten, wird Ctl-T betätigt.

KEY OFF
Löscht die Anzeige der programmierbaren Funktionstasten aus Zeile 25, deaktiviert die Funktionstasten jedoch nicht.

KEY LIST
Listet alle 20 programmierbaren Funktionstastenwerte auf dem Bildschirm auf. Sämtliche 15 Zeichen jedes Wertes werden angezeigt.

KEY <Tastennummer>,<Zeichenfolgenausdruck>.
Weist der angegebenen Tasten den Zeichenfolgenausdruck zu. Der Zeichenfolgenausdruck kann eine Länge von 1 bis 15 Zeichen haben. Ist er länger als 15 Zeichen, so werden nur die ersten 15 Zeichen zugewiesen.

Wird ein Wert für <Tastennummer> angegeben, der nicht in dem Bereich von 1 bis 20 liegt, so wird die Fehlermeldung "Illegal Function Call" angezeigt. Die vorhergehende Tastenzuweisung wird beibehalten.

Wird einer programmierbaren Funktionstaste eine Zeichenfolge mit der Länge 0 zugewiesen, so wird die Taste deaktiviert. Sie bleibt deaktiviert, bis ihr ein anderer Zeichenfolgenausdruck zugewiesen wird.

Wird eine programmierbare Funktionstaste zugewiesen, so gibt die INKEY$-Funktion bei jedem Aufruf ein Zeichen der Zeichenfolge zurück. Ist die programmierbare Funktionstaste deaktiviert, so gibt INKEY$ eine Zeichenfolge mit der Länge 2 zurück. Das erste Zeichen ist eine binäre 0 und das zweite stellt den Abtastcode der Taste dar.

Beispiele:

In dem nachfolgenden Beispiel weist die Anweisung in Zeile 10 der Taste F1 die Zeichenfolge 'MENU' <Abschluß> zu. Diese Zuweisung kann in einem Programm dazu benutzt werden, ein Menü anzuzeigen, wenn der Benutzer in dieses Menü geht. Mit Zeile 20 wird die Taste deaktiviert.

```
10 KEY 1,"MENU"+CHR$(13)
20 KEY 1," "
```

Die nachfolgende Routine initialisiert die fünf ersten programmierbaren Funktionstasten:

```
10 KEY OFF
20 DATA KEY1,KEY2,KEY3,KEY4,KEY5
30 FOR I=1 to 5: READ FUNCTIONKEYS$(I)
40 KEY I,FUNKTIONKEYS$(I)
50 NEXT I
60 KEY ON
```

LCOPY-ANWEISUNG

Format: LCOPY

Zweck: Druckt den mit der GW-BASIC Konfigurationsroutine festgelegten Speicher (1, 2, 4) aus (siehe "Konfiguration für GW-BASIC" in Kapitel 1).

Bemerkungen: LCOPY kann für das Ausdrucken sowohl von Grafiken als auch von Texten benutzt werden, wenn sich der Text ebenfalls im Grafikmodus befindet.

Bei einem Schwarz/Weiß-Bildschirm druckt LCOPY das gesamte Grafikabbild (und Texte im Grafikmodus).

Beispiel: Für ein Beispiel des Ausdruckens von Bildschirmabbildern wird auf "Farbauswahl" in Kapitel 1 verwiesen.

LINE-ANWEISUNG

Format: LINE[(x1,y1)]-(x2,y2) [,[Farbe] [,b[f]]]

Zweck: Zeichnet eine Linie, ein Kästchen oder ein ausgefülltes Kästchen auf dem Bildschirm.

(x1,y1),(x2,y2)
Gibt die Koordination in absoluter oder verschobener Form an (siehe "X- und Y-Koordinaten" in Kapitel 1). Werden die Koordinaten für den (x1, y1) Punkt nicht angegeben, so entspricht der Anfangspunkt der Linie dem letzten von (x2,y2) in einer vorhergehenden Anweisung angegebenen Punkt.

Farbe
Gibt die Farbe der Linie, des Kästchens oder ausgefüllten Kästchens an (0—7, siehe COLOR-Anweisung). Wird keine Angabe gemacht, so entspricht die Farbe der Vordergrundfarbe.

b oder bf
Gibt ein Kästchen oder ein ausgefülltes Kästchen an. Mit b wird BASIC angewiesen, ein Rechteck zu zeichnen, wobei die Punkte (x1,y1) und x2,y2) gegenüberliegende Ecken darstellen. Dadurch brauchen keine vier LINE-Befehle eingegeben zu werden, die dieselbe Funktion ausführen:

```
LINE (x1,y1)-(x2,y1)
LINE (x1,y1)-(x1,y2)
LINE (x2,y1)-(x2,y2)
LINE (x1,y2)-(x2,y2)
```

Mit bf wird BASIC angewiesen, dasselbe Rechteck wie b zu zeichnen und außerdem die Innenpunkte mit derselben Farbe wie bei b auszufüllen.

Bemerkungen: Die verschobene Koordinatenform kann überall dort benutzt werden, wo eine Koordinate benutzt wird. Hier muß beachtet werden, daß sämtliche Grafikanweisungen und Grafikfunktionen den zuletzt benutzten Punkt aktualisieren. Wird die ver-

schobene Form mit der zweiten Koordinate benutzt, so wird die Koordinate von der ersten Koordinate in der Anweisung abgesetzt. Mit dem folgenden Befehl wird beispielsweise eine Linie von (60,40) bis (70,50) gezeichnet:

LINE (60,40)-STEP(10,10)

Wird eine Koordinate angegeben, die außerhalb des zulässigen Bereiches liegt, so wird der Koordinate der nächstgültige Wert zugewiesen. Anders ausgedrückt, negative Werte werden zu 0, y-Werte, die größer sind als 399 werden zu 399, und x-Werte, die größer sind als 639, werden zu 639.

Beispiele: Zeichnen einer Line von dem letzten angegebenen Punkt zu dem Punkt (x2,y2):

LINE -(x2,y2)

Aufnahme eines Anfangspunktes für eine auf dem Bildschirm diagonal nach unten laufende Linie:

LINE (0,0)-(639,399)

Zeichnen einer Linie quer über den Bildschirm:

LINE (0,200)-(639,200)

Zeichnen einer Linie in der Farbe Nr. 2:

LINE (10,0)-(20,20),2

Zeichnen eines Kästchens in der Vordergrundfarbe:

LINE (0,0)-(100,100),,b

Zeichnen eines Kästchens und Ausfüllen dieses Kästchens mit der Farbe Nr. 2:

LINE (0,0)-(200,200),2,bf

Kontinuierliches Zeichnen von Linien mit beliebigen Farben:

```
10 SCREEN 1
20 CLS
30 LINE -(rnd*639,rnd*399),rnd*7
40 GO TO 20
```

Zeichnen eines sich abwechselnden Musters — Linie ein, Linie aus:

```
10 SCREEN 1
20 FOR X=0 TO 639
30 LINE (X,0)-(X,399),X AND 1
40 NEXT
```

Zeichnen beliebig ausgefüllter Kästchen in beliebigen Farben:

```
10 SCREEN 1
20 CLS
30 LINE -(rnd*639,rnd*399),rnd*7,bf
40 GO TO 20
```

LIST-ANWEISUNG

Format: LIST [[<Zeilennummer>[-[<Zeilennummer>]]] [,<Einheit>]]

Zeilennummer
Gibt die Zeilennummer in dem Bereich von 0 bis 65.529 an.

Einheit
Zeichenfolgenausdruck für eine der folgenden Einheiten:

"SCRN:" Bildschirm
"LPT1:" Zeilendrucker

Zweck: Ermöglicht das Auflisten eines Programms auf dem Bildschirm oder einem Zeilendrucker.

Bemerkungen: Wird der Einheitenparameter weggelassen, so wird der Bildschirm als Standardeinheit benutzt.

Wird der Zeilenparameter weggelassen, so listet die Software das gesamte Programm auf.

Jede Auflistung auf dem Bildschirm oder dem Drucker kann durch Betätigung von CTl-S unterbrochen werden.

Wird der Gedankenstrich (-) in dem Zeilenparameter benutzt, so stehen die drei folgenden Optionen zur Verfügung:

Wird nur die erste Zeilennummer angegeben, so listet die Software diese Zeile und alle Zeilen mit höheren Nummern auf.

Wird nur die zweite Zeilennummer angegeben, so listet die Software sämtliche Zeilen ab dem Anfang des Programms bis zur angegebenen Zeile auf.

Werden beide Zeilennummern angegeben, so listet die Software den angegebenen Zeilenbereich auf.

LIST, "LPT1:" entspricht LLIST in MS-BASIC. Wird "LPT1:" angegeben, so wird der letzte gültige Breitenbefehl, der für den Drucker eingegeben wurde, benutzt.

LIST, "SCRN:" entspricht LIST in MS-BASIC.

Beispiele: Auflisten des Programms auf dem Zeilendrucker:

LIST ,"LPT1:"

Auflisten der Zeilen 10 bis 20 auf dem Bildschirm:

LIST 10-20

Auflisten von Zeile 10 bis zur letzten Zeile auf dem Bildschirm:

LIST 10- ,"SCRN:"

Auflisten von der ersten Zeile bis zur Zeile 200 auf dem Zeilendrucker:

LIST -200,"LPT1:"

Auflisten der Zeilen 35 bis 65 auf dem Bildschirm:

LIST 35-65,"SCRN:"

LOCATE-STATEMENT

Format: LOCATE [Zeile] [,[Spalte] [,[Schreibmarke]]]

Zeile
Gibt die Zeilennummer auf dem Bildschirm an. Wahlweise. An dieser Stelle muß ein numerischer Ausdruck angegeben werden, der zu einer vorzeichenlosen Ganzzahl in dem Bereich von 1-24 führt.

Spalte
Gibt die Spaltennummer auf dem Bildschirm an. Wahlweise. An dieser Stelle muß ein numerischer Ausdruck eingegeben werden, der zu einer vorzeichenlosen Ganzzahl in den Bereich von 1-80 führt.

Schreibmarke
Ein boolescher Wert, mit dem angegeben wird, ob die Schreibmarke sichtbar ist. Wahlweise. 0 für nicht sichtbar und eine von Null abweichende Zahl für sichtbar angeben.

Zweck: Bewegt die Schreibmarke zu der angegebenen Position auf dem aktiven Bildschirm. Durch nachfolgende PRINT-Anweisungen werden Zeichen an diese Position gesetzt. Durch den wahlweisen Parameter für die Schreibmarke wird die blinkende Schreibmarke ein- und ausgeschaltet.

Bemerkungen: Werden für die Zeile und die Spalte Werte außerhalb der zulässigen Bereiche eingegeben, so wird eine Fehlermeldung "Illegal Function Call" angezeigt. In diesem Fall werden die vorhergehenden Werte beibehalten.

Jeder dieser Parameter kann weggelassen werden. Werden die Parameter weggelassen, so wird von dem vorhergehenden Wert ausgegangen.

Die Häufigkeit des Aufblinkens der Schreibmarke kann nicht ausgewählt werden.

Die 25. Zeile ist für die Anzeige der programmierbaren Funktionstasten reserviert. Es wird empfoh-

len, nicht über diese Zeile zu schreiben, selbst wenn die Anzeige ausgeschaltet ist.

Beispiel: In dem nachfolgenden Beispiel bewegt die Anweisung in Zeile 10 die Schreibmarke zu der Ausgangsstellung in der oberen linken Ecke. Die Anweisung in Zeile 20 macht die blinkende Schreibmarke sichtbar, ohne ihre Position zu ändern.

```
10 LOCATE 1,1
20 LOCATE ,,1
```

ON COM(n)-ANWEISUNG

Format: ON COM(n) GOSUB <Zeile>

n
Nummer des Übertragungskanals (1 oder 2).

Zeile
Zeilennummer des Anfangs der Unterbrechnungs-Routine. Durch eine Zeilennummer 0 wird die Unterbrechung für den angegebenen Kanal deaktiviert.

Funktion: Mit dieser Anweisung kann die Software eine Zeilennummer unterbrechen, wenn Informationen in den Datenübertragungspuffer gesetzt werden.

Bemerkungen: Die folgenden Anweisungen steuern die Aktivierung oder Deaktivierung der Unterbrechungsroutine:

COM(n) ON
Muß ausgeführt werden, um die ON COM(n)-Anweisung zu aktivieren. Wird in der ON COM(n)-Anweisung eine von 0 abweichende Zeilennummer angegeben, so prüft die Software jedesmal, wenn das Programm eine neue Anweisung startet, ob Zeichen in den angegebenen Kanal gesetzt wurden. Sind keine Zeichen vorhanden, so führt die Software einen GOSUB-Befehl zu der angegebenen Zeile aus.

COM(n) OFF
Wird diese Anweisung ausgeführt, so findet für den Kanal keine Unterbrechung statt. Selbst wenn Datenübertragungen ausgeführt werden, werden die in dem Kanal empfangenen Zeichen nicht im Speicher gesichert.

COM(n) STOP
Wird diese Anweisung ausgeführt, so findet für den Kanal keine Unterbrechung statt. Eventuell in dem Kanal empfangene Zeichen werden jedoch im Speicher gesichert, so daß bei Ausführung von COM(n) ON sofort eine Unterbrechung stattfindet.

Kommt es zu einer Unterbrechung, so veranlaßt die Unterbrechung automatisch eine COM(n) STOP für diese Routine, so daß es niemals zu sich wiederholenden Unterbrechungen kommt. Beim Rücksprung aus der Unterbrechungsroutine wird automatisch eine COM(n) ON-Anweisung ausgeführt, es sei denn, eine explizite COM(n) OFF-Anweisung wurde innerhalb der Unterbrechungsroutine ausgeführt.

Eine Unterbrechung findet nur dann statt, wenn die Software ein Programm ausführt.

Kommt es zu einer Unterbrechung wegen eines Fehlers, so werden sämtliche Unterbrechungen automatisch deaktiviert.

Typischerweise liest die Unterbrechungs-Routine einer Datenübertragung eine gesamte Meldung von dem Übertragungskanal, bevor sie sie zurückgibt. Es wird nicht empfohlen, die Datenübertragungsunterbrechung bei aus einem einzigen Zeichen bestehenden Meldungen zu benutzen, da der Aufwand für die Unterbrechung und das Lesen jedes einzelnen Zeichens bei hohen Baudraten zu einem Überlauf des Unterbrechungspuffers für die Datenübertragung führen kann.

RETURN <Zeile>
Diese Form von RETURN ist wahlweise. Sie wird benutzt, um bei einer festen Zeilennummer in das Softwareprogramm zurückzugehen. Durch diese Maßnahme wird der GOSUB-Eintrag überflüssig, den die Unterbrechung erstellt hat. RETURN <Zeile> muß mit größter Vorsicht benutzt werden! Jede andere zum Zeitpunkt der Unterbrechung aktive GOSUB-, WHILE- oder FOR-Anweisung bleibt aktiv. Beim Rücksprung aus der Subroutine führt jeder Versuch, die Schleifen außerhalb der Subroutine fortzusetzen, zu der Fehlermeldung "NEXT without FOR".

ON KEY(n)-ANWEISUNG

Format: ON KEY(n) GOSUB <Zeile>

n
Gibt eine von 1 bis 24 numerierte Funktionstaste wie folgt an:

1-20 Programmierbare Funktionstasten F1 bis F20
21 Schreibmarke nach oben
22 Schreibmarke nach links
23 Schreibmarke nach rechts
24 Schreibmarke nach unten

Zeile
Gibt die Zeilennummer an, bei der BASIC die Unterbrechungsroutine für die angegebene Taste beginnt. Durch eine Zeilennummer 0 wird die Unterbrechung für diese Taste deaktiviert.

Zweck: Ermöglicht der Software die Unterbrechung einer Zeilennummer, wenn die entsprechende Funktionstaste oder Schreibmarkentaste betätigt wird.

Bemerkungen; Die folgenden Anweisungen steuern die Aktivierung oder Deaktivierung der Unterbrechungsroutine:

KEY(n) ON
Muß ausgeführt werden, um die ON KEY(n)-Anweisung zu aktivieren. Wird für die Unterbrechung mit ON KEY(n) eine von Null abweichende Zeilennummer angegeben, so prüft die Software jedesmal, wenn das Programm eine neue Anweisung startet, ob die angegebene Taste betätigt wurde. Wurde die Taste betätigt, so führt die Software eine GOSUB-Anweisung zu der angegebenen Zeile aus.

KEY(n) OFF
Wird diese Anweisung ausgeführt, so findet für die angegebene Taste keine Unterbrechung statt. Selbst wenn die Taste betätigt wird, wird die Unterbrechungsroutine nicht berücksichtigt.

KEY(n) STOP
Wird diese Routine ausgeführt, so findet keine Unterbrechung für die angegebene Taste statt. Wird jedoch die angegebene Taste betätigt, so findet bei Ausführung von KEY(n) ON sofort eine Unterbrechung statt.

Kommt es zu einer Unterbrechung, so verursacht sie automatisch eine KEY(n) STOP-Anweisung bei dieser Routine, so daß es niemals zu sich wiederholenden Unterbrechungen kommen kann. Beim Rücksprung aus der Unterbrechungsroutine wird automatisch eine KEY(n) ON-Anweisung ausgeführt, es sei denn, innerhalb der Unterbrechungsroutine wurde eine explizite KEY(n) OFF-Anweisung ausgeführt.

Die Unterbrechung findet nur statt, wenn die Software ein Programm ausführt.

Kommt es zu einer Unterbrechung wegen eines Fehlers, so werden sämtliche Unterbrechungen automatisch deaktiviert.

Keine Art der Unterbrechung wird deaktiviert, wenn sich die Software im Direktmodus befindet. Insbesondere die Funktionstasten nehmen ihre Standardfunktion während der Eingabe wieder auf.

Eine Taste, die zu einer Unterbrechung führt, kann nicht mit den INPUT- oder INKEY$-Anweisungen getestet werden, so daß verschiedene Unterbrechungsroutinen für jede Taste benutzt werden müssen, wenn eine unterschiedliche Funktion gewünscht wird.

RETURN <Zeile>
Diese Art von RETURN ist wahlweise. Mit RETURN <Zeile> wird bei einer festen Zeilennummer zu dem Softwareprogramm zurückgegangen. Durch diese Maßnahme wird der GOSUB-Eintrag überflüssig, der von der Unterbrechung erstellt wurde. RETURN <Zeile> muß mit Vorsicht benutzt werden! Zum Zeitpunkt der Unterbrechung

aktive andere GOSUB-, WHILE- oder FAR-Anweisungen bleiben aktiv. Beim Rücksprung aus einer Subroutine führt jeder Versuch, zur Fortsetzung der Schleifen außerhalb der Subroutine zu der Fehlermeldung "NEXT without FOR".

OPEN COM-ANWEISUNG

Format: OPEN "<Einheit>:[<Geschwindigkeit>],
[<Parität>],[<Daten>],[<Stopp>] [,RS]
[,CS[<n>]] [,DS [<n>]] [,CD[<n>]] [,LF]"
AS [#] <Dateinummer>

Zweck: Eröffnet eine Dateiübertragungsdatei. Weist einen Puffer auf dieselbe Art und Weise wie OPEN bei Diskettendateien für die Ein-/Ausgabe zu. Unterstützt die asynchrone Datenübertragung mittels RS-232 zu anderen Computern und Peripheriegeräten.

Bemerkungen: Einheit
Gibt eine der folgenden Datenübertragungseinheiten an: COM1 oder COM2.

Geschwindigkeit
Eine ganzzahlige Konstante, die die Baudrate beim Senden oder Empfangen angibt. Gültige Geschwindigkeiten sind 50, 75, 100, 134, 150, 300, 600, 1200, 1800, 2400, 3600, 4800, 7200, 9600 und 19200. Die Baudrate von 134 umfaßt 134,5. Die Standardvorgabe beträgt 300 bps.

Parität
Eine aus einem Zeichen bestehende Konstante, die die Parität für das Senden und Empfangen wie folgt angibt:

S LEERZEICHEN: Das Paritätsbit wird immer in Form eines Leerzeichens (0 Bit) gesendet und empfangen.

O UNGERADE: Parirätsprüfung und ungerade Parität beim Senden und Empfangen.

M MARKE: das Paritätsbit wird immer als eine Marke gesendet und empfangen (1 Bit).

E GERADE: Paritätsprüfung auf gerades Paritätsbit beim Senden und Empfangen.

N KEINE: keine Paritätsprüfung beim Senden oder Empfangen.

Die Standardvorgabe für die Parität ist die gerade Parität (E).

Daten
Eine ganzzahlige Konstante, die die Anzahl von Sende- oder Empfangs-Datenbits angibt. Gültige Werte sind: 4, 5, 6, 7 und 8. Die Standardvorgabe ist 7. Wird 4 angegeben, so muß gleichzeitig die Parität in Form einer Marke (M) oder eines Leerzeichens (S) angegeben werden. Wird M oder S nicht angegeben, so kommt es zu der Fehlermeldung "Bad File Name". Werden 8 Bits angegeben, so muß die Parität N (keine) angegeben werden.

Stop
Eine ganzzahlige Konstante, die die Anzahl von Stoppbits angibt. Gültige Werte sind 1 und 2. Das Standardstoppbit für 50, 75 und 100 bps ist 2. Das Standardstoppbit für alle anderen Werte ist 1. Wird 4 oder 5 für <Daten> angegeben, so bedeutet eine für <Stop> eingegebene 2 11/2 Stoppbits.

RS
Unterdrückt das Leitungssignal "Sendeanforderung" (RTS). Wird RS eingegeben, so wird die RTS-Leitung nicht eingeschaltet, wenn eine OPEN COM-Anweisung ausgeführt wird.

CS <n>
Steuert das Leitungssignal "Sendebereitschaft" (CTS). Wird CS eingegeben, so wartet das System auf das Leitungssignal, ohne einen Fehler zurückzugeben. Wird CSn eingegeben, so gibt n die Zeit an, während der gewartet wird, bevor das System eine Fehlermeldung "Device Timeout" zurückgibt. Wird n gleich Null gesetzt, so entspricht dies der Eingabe von CS. Wird die Option weggelassen, so beträgt die Standardvorgabe eine Sekunde.

DS <n>
Steuert Das Leitungssignal "Betriebsbereitschaft" (DSR). Wird DS eingegeben, so wartet das System auf das Leitungssignal, ohne einen Fehler zurückzugeben. Wird DSn eingegeben, so gibt n die Zeit an, während der gewartet wird, bevor das System eine Fehlermeldung "Device Timeout" zurückgibt. Wird n auf Null gesetzt, so entspricht dies der Eingabe von DS. Wird die Option weggelassen, so beträgt die Standardvorgabe eine Sekunde.

CD<n>
Steuert das CD-Leitungssignal, das auch unter dem Namen Empfangssignalpegel (RLSD) bekannt ist. Wird CD eingegeben, so wird das Leitungssignal nicht überprüft. Wird CDn eingegeben, so gibt n die Zeit an, während der gewartet wird, bevor das System eine Fehlermeldung "Device Timeout" zurückgibt. Wird n auf Null gesetzt oder wird die Option weggelassen, so wird das Leitungssignal nicht überprüft.

n
Gibt die Anzahl von ms an, während der das System wartet, bevor es eine Fehlermeldung „Device Timeout" zurückgibt. n kann von 0 bis 65.535 gehen.

LF
Sendet einen Zeilenvorschub im Anschluß an jede Zeilenschaltung. LF muß angegeben werden, wenn die Datenübertragungsdateien auf einem seriellen Zeilendrucker ausgedruckt werden. Hier muß beachtet werden, daß die INPUT# - und LINE INPUT# -Anweisungen, wenn sie für das Lesen aus einer Datenübertragungsdatei benutzt werden, die mit der LF-Option eröffnet wurde, den Zeilenvorschub und den Stopp ignorieren, wenn sie eine Zeilenschaltung feststellen.

Dateinummer
Gibt einen ganzzahligen Ausdruck an, der eine gültige Dateinummer zurückgibt. Die Nummer ist mit der Datei während der Eröffnungszeit ver-

knüpft und wird von anderen E/A-Anweisungen der Datenübertragung für die Bezugnahme auf die Datei benutzt.

Eventuelle Codierfehler innerhalb des Zeichenfolgenausdruckes von <Geschwindigkeit> bis zu LF führen zu einer Fehlermeldung "Bad File Name". Es wird nicht angegeben, welcher Parameter fehlerhaft ist.

Ist der Modemanschluß nicht richtig aufgestellt, so kommt es zu einer Fehlermeldung "Device Timeout", wenn DSR nicht erkannt wird. Für richtige Verdrahtungsinstruktionen wird auf die Hardware-Dokumentation verwiesen.

Für Informationen über die Ein-/Ausgabe bei der Datenübertragung wird auf das Kapitel "Datenübertragung" verwiesen.

Die Fehlermeldungen bei der Datenübertragung werden in dem Anhang "Fehlermeldungen" angegeben.

Beispiel: In dem folgenden Beispiel ist Dateinummer 1 für die Datenübertragung mit allen Standardvorgaben eröffnet: 300 bps, gerade Parität und 7 Datenbits mit einem Stoppbit.

```
10 OPEN "COM1:" AS #1
```

In dem folgenden Beispiel wird Dateinummer 2 für die Datenübertragung mit einer Geschwindigkeit von 2.400 bps eröffnet. Die Standardvorgaben sind: gerade Parität, 7 Datenbits und ein Stoppbit.

```
10 OPEN "COM1:2400" AS #2
```

In dem folgenden Beispiel wird Dateinummer 1 für eine asynchrone Ein-/Ausgabe mit einer Geschwindigkeit von 1.200 bps eröffnet. Eine Parität wird weder erstellt noch überprüft. 8-Bit-Bytes werden gesendet und empfangen. Der Standardwert für das Stoppbit lautet 1.

```
10 OPEN "COM2:1200,N,8" AS #
```

PAINT-ANWEISUNG

Format: PAINT (x,y) [,Zeichenfarb[,Grenzfarbe]]

Zweck: Füllt einen Bereich auf dem Bildschirm mit der angegebenen Farbe auf.

x,y
Gibt die Koordinaten des Punktes an, an dem mit dem Zeichnen begonnen wird. Dieser Punkt, der in einer absolutlen oder verschobenen Form angegeben werden kann (für eine Erläuterung der x- und y-Koordinaten wird auf Kapitel 1 verwiesen), kann innerhalb oder außerhalb einer Zeichnung, jedoch nicht auf einer Grenze liegen.

Zeichenfarbe
Gibt die Farbe an, mit der gezeichnet wird (die "Füll"-Farbe). An dieser Stelle wird ein Wert von 01—7 eingegeben (siehe COLOR-Anweisung).

Grenzfarbe
Gibt die Grenzfarbe der Zeichnung an. An dieser Stelle wird ein Wert von 0—7 eingegeben.

Bemerkungen: Der Anfangspunkt kann innerhalb oder außerhalb einer Zeichnung liegen. Liegt der Punkt außerhalb, so wird der Bildschirm mit der Farbe gekennzeichnet. Nur der Bereich innerhalb der Grenzen wird nicht mit dieser Farbe gezeichnet.

Wird keine Zeichenfarbe angegeben, so wird die momentan aktive Vordergrundfarbe benutzt. Wird keine Grenzfarbe angegeben, so wird der ganze Bildschirm gezeichnet.

Bei Schwarz/Weiß-Bildschirmen kann nur schwarz und grün angegeben werden. Wird eine andere Farbe angegeben, so wird automatisch grün benutzt.

Beispiel: In dem folgenden Beispiel wird ein Kreis gezeichnet, der dann zuerst mit Gelb und danach mit Schwarz ausgefüllt wird. Mit der GOTO-Anweisung wird die Folge wiederholt, so daß es zu einem Trickbildeffekt kommt.

```
10 COLOR 7,0
20 CIRCLE (300,200),10,1
40 PAINT (300,191),6,1
50 PAINT (300,209),0,0
60 GOTO 20
```

Hier muß beachtet werden, daß die Grenzfarbe bei der zweiten Anweisung schwarz ist, wodurch die vorherige blaue Grenze überdeckt und der Kreis "gelöscht" wird.

PLAY-ANWEISUNG

Format: PLAY <Zeichenfolgenausdruck>

Zweck: Spielt Musik, wie durch den Zeichenfolgenausdruck definiert.

Bemerkungen: Mit der PLAY-Anweisung kann ein Ton erzeugt werden, indem seine Eigenschaften in dem Zeichenfolgenausdruck definiert werden. Der Ausdruck kann aus einem der folgenden Befehle bestehen, die in beliebiger Reihenfolge angegeben werden können, es sei denn, in der Beschreibung werden ausdrücklich andere Angaben gemacht.

A-G [#,+,-] — Musikskala
Spielt die angegebenen Noten, A—G. A# oder + im Anschluß an eine Note gibt einen hohen Ton (einen halben Schritt höher) an. Ein — im Anschluß an eine Note gibt einen tiefen Ton (einen halben Schritt niedriger) an.

L <n> — Länge
Legt die Länge der Note (oder der Noten) fest, wobei n von 1—64 gehen kann. So gibt L1 beispielsweise eine ganze Note an, L2 eine halbe Note ... und L64 eine 64stel Note. Die Länge kann vor einer Gruppe von Noten oder im Anschluß an eine einzelne Note angegeben werden, um nur deren Länge zu ändern. Im letzteren Fall entspricht beispielsweise A16 der Definition von L16A.

MB — Vordergrundmusik
Legt die Musik oder den Ton fest, der im Vordergrund gespielt werden soll. Jede nachfolgende Note oder jeder nachfolgende Ton wird erst gestartet, nachdem die vorhergehende Note oder der vorhergehende Ton beendet ist. MF ist der ursprüngliche Standardwert.

MN — Musik normal
Spielt jede Note mit 7/8 der Zeit, die in L (Länge) angegeben wird.

ML — Musik Legato
Spielt jede Not mit der vollen Länge (wie in L angegeben).

MS — Musik Staccato
Spielt jede Note mit 3/4 in der Zeit, die in L (Länge) angegeben wurde.

N <n> — Note
Spielt die von n angegebene Note. n kann von 22—63 gehen. (Siehe Notentabelle unter der Sound-Anweisung). n kann gleich Null sein, um eine Pause anzugeben. Mit diesem Befehl kann die Note auf einem anderen Weg als durch den Namen (A—G) und die Oktave angegeben werden.

O <n> — Oktave
Legt die Oktave fest, wobei n von 1—5 gehen kann.

P <n> — Pause
Legt die Länge der Pause fest, wobei n von 1—64 gehen kann. Der Wert von n entspricht dem Wert von n in dem Längen-Befehl. So kommt es beispielsweise bei P1 zu einer Pause während der Länge einer gesamten Note, bei P2 zu einer Pause während der Länge einer halben Note usw.

T <n> — Tempo
Legt die Anzahl von Viertelnoten (n) fest, die in einer Minute gespielt werden können. n kann von 32—255 gehen; der Standardwert beträgt 120.

. — Punkt
Wird der Punkt im Anschluß an eine Note benutzt, so wird die Note als eine mit Punkt versehene Note gespielt, d.h., ihre Länge wird mit 3/2 multipliziert. Im Anschluß an eine Note kann mehr als ein Punkt benutzt werden. In diesem Fall wird die Länge entsprechend angepaßt. Durch A.. wird beispielsweise 9/4 mal so lang wie durch L angegeben gespielt, durch A.... wird 27/8 gespielt usw. Punkte können auch im Anschluß an eine Pause (P) benutzt werden, um die Pausenlänge auf dieselbe Art und Weise zu skalieren.

X Variable;
führt die angegebene Zeichenfolge aus.

In allen Befehlen kann der Wert n eine Konstante oder eine = Variable darstellen, wobei Variable dem Namen der Variablen entspricht. Das Semikolon (;) ist erforderlich, wenn eine Variable auf diese Art und Weise benutzt wird. Außerdem ist es erforderlich, wenn der X-Befehl benutzt wird. Ansonsten ist ein Semikolon zwischen Befehlen wahlweise. Allerdings ist es im Anschluß an MF, MB, MN, ML oder MS nicht zulässig. Leerzeichen in einer Zeichenfolge werden ignoriert.

Variablen können auch in der Form VARPTR$ (Variable) anstelle von = Variable; angegeben werden. Diese Methode ist in Programmen, die später kompiliert werden, von besonderem Nutzen.

Mit X kann auch ein "Unterton" in einer Zeichenfolge gespeichert und wiederholt mit verschiedenen Tempi oder Oktaven aus einer anderen Zeichenfolge aufgerufen werden.

Beispiel:

```
10 MARY$="GFE-FGGG"
20 PLAY "MB T100 03 L8; XMARY$; P8 FFF4"
30 PLAY "GB-B-4; XMARY$; GFFGFE-."
```

POINT-FUNKTION

Format: V=POINT(x,y)

V
Gibt die Farbe an. Gültige Werte gehen von 0—7 (siehe COLOR-Anweisung).

x,y
Geben die Koordinaten eines Punktes an. Die Koordinaten müssen die absolute Form aufweisen (für eine Beschreibung der x- und y-Koordinaten wird auf Kapitel 1 verwiesen).

Zweck: Mit dieser Funktion kann der Benutzer die Farbe eines Punktes vom Bildschirm lesen.

Bemerkungen: Wird ein Punkt angegeben, der au-erhalb des Bereiches —32768 bis 32767 liegt, so wird —1 zurückgegeben.

Beispiele:

```
10 SCREEN 1
20 FOR C=0 TO 7
30 PSET (10,10),C
40 IF POINT(10,10)<>C THEN PRINT "Broken
   Basic!"
50 NEXT C
```

```
10 SCREEN 1
20 IF POINT(i,i) <>0 THEN PRESET (i,i) ELSE
   PSET (i,i)
```

In dem zweiten Beispiel überprüft BASIC die Farbe eines Punktes. Ist der Punkt nicht schwarz, so wird die Farbe in schwarz geändert. Ist der Punkt schwarz, so wird die Farbe in die Vordergrundfarbe geändert. Dies kann auch noch auf andere Art und Weise ausgeführt werden:

```
10 SCREEN 1
20 PSET (i,1i),1-POINT(i,i)
```

PRESET-ANWEISUNG

Format: PRESET (x-Koordinate,y-Koordinate) [,Farbe]

x-Koordinate, y-Koordinate
Legt die Punktkoordinaten in absoluter oder verschobener Form fest (für eine Erläuterung der x- und y-Koordinaten wird auf Kapitel 1 verwiesen).

Farbe
Wahlweise. Gibt die Farbe des Punktes (0—7, siehe COLOR-Anweisung) an. Wird die Farbe nicht angegeben, so entspricht die Farbe der Hintergrundfarbe.

Zweck: Legt einen Punkt auf dem Bildschirm fest, von dem an mit dem Zeichnen begonnen wird.

Bemerkungen: PRESET und PSET verfügen über eine identische Syntax. Der einzige Unterschied besteht darin, daß nun bei PRESET keine Farbe angegeben wird, die Hintergrundfarbe 0 ausgewählt wird. Wird bei PSET keine Farbe angegeben, so entspricht die Farbe der Vordergrundfarbe. Zeile 60 in dem Beispiel unter PSET könnte folgendermaßen aussehen:

60 PRESET (i,i)

Bei BASIC können die Koordinatenwerte hinter dem Rand des Bildschirms liegen, ohne daß eine Maßnahme ergriffen oder eine Fehlermeldung angezeigt wird. Werte außerhalb des Ganzzahlbereichs —32768 bis 32767 führen jedoch zu einem Überlauffehler.

PSET-ANWEISUNG

Format: PSET (x-Koordinate,y-Koordinate)[,Farbe]

x-Koordinate, y-Koordinate
Gibt die Koordinaten des Punktes entweder in absoluter oder verschobener Form an (für eine Erläuterung der x- und y-Koordinaten wird auf Kapitel 1 verwiesen).

Farbe.
Wahlweise. Gibt die Farbe des Punktes an (0—7, siehe COLOR-Anweisung). Wird keine Farbe angegeben, so entspricht die Farbe der Vordergrundfarbe.

Zweck: Legt einen Punkt auf dem Bildschirm fest, von dem an mit dem Zeichnen begonnen wird.

Bemerkungen: Bei BASIC können Koordinatenwerte hinter dem Rand des Bildschirms liegen, ohne daß eine Maßnahme ergriffen oder eine Fehlermeldung angezeigt wird. Werte außerhalb des Ganzzahlbereiches von —32768 bis 32767 führen jedoch zu einem Überlauffehler.

Beispiel: Wird Schwarz als Vordergrundfarbe festgelegt, so zeichnen die Zeilen 10 bis 40 des Beispiels eine diagonale Linie von Punkt (0,0) zu Punkt (100,100). Die Zeilen 50 bis 70 löschen die Linie, indem jeder Punkt auf 0, schwarz, gesetzt wird.

```
10 SCREEN 1
20 FOR i=0 TO 100
30 NEXT
40 FOR i=100 TO 0 STEP -1
50 PSET (i,i),0
60 NEXT
```

RETURN-ANWEISUNG

Format: RETURN [<Zeile>]

Zeile
Gibt die Zeilennummer des Programms an, in die zurückgegangen werden soll.

Zweck: Rücksprung aus einer Subroutine.

Bemerkungen: Der wahlweise Zeilenparameter wurde zu der RETURN-Anweisung hinzugefügt, um nichtlokale Rücksprünge aus Unterbrechungsroutinen zu ermöglichen. Unter Umständen soll bei einer festen Zeilennummer in das BASIC-Programm zurückgegangen werden, während der von der Unterbrechung erzeugte GOSUB-Eintrag eliminiert wird. RETURN <Zeile> muß mit Vorsicht benutzt werden! Jede andere GOSUB-, WHILE- oder FOR-Anweisung, die während der Unterbrechung aktiv war, bleibt aktiv.

SCREEN-FUNKTION

Format: SCREEN-Modus

Der Modus muß entweder mit 0 oder 1 angegeben werden:

0 = Textmodus
1 = Grafikmodus.

Zweck: Wählt die Software für die Arbeit im Text- oder Grafikmodus aus. (Für eine Erläuterung des Bildschirmmodus wird auf Kapitel 1 verwiesen).

Bemerkungen: Ist der Wert des Modusparameters gültig, so speichert die Software den neuen Bildschirmmodus und löscht den Bildschirm. Die Vordergrund- und Hintergrundfarben werden nicht geändert. Ist der Parameterwert ungültig (nicht 0, 1 oder 2 — 2 für Kompatibilität und Umwandlung), so wird die Fehlermeldung "Illegal Function Call" angezeigt. Der Bildschirmmodus bleibt derselbe wie vor der Eingabe der Anweisung.

Ist der neue Bildschirmmodus derselbe wie der vorhergehende Mosud, so löscht die Software nur den Bildschirm.

Beispiele:

```
10 SCREEN 0    Wählt den Textmodus aus.
20 SCREEN 1    Schaltet in den Grafikmodus um.
```

SOUND-ANWEISUNG

Format: SOUND <Frequenz,Dauer>

Frequenz
Gibt die gewünschte Frequenz in Hertz (Zyklen pro Sekunde) an. An dieser Stelle wird die gewünschte Zahl zwischen 220 und 32767 eingegeben. (Siehe auch Tabelle mit Noten und Frequenzen).

Dauer
Gibt die gewünschte Länge des Tons in Taktschritten an. (Ein Taktschritt = 55 ms). An dieser Stelle wird die Anzahl von Taktschritten eingegeben. (Siehe auch Tabelle mit den typischen Tempi).

Zweck: Generiert einen Ton über den Lautsprecher.

Note	Frequenz	Nr.*	Note	Frequenz	Nr.*
Pause	32767		F#	740	43
A	220	22	G	784	44
A#	233	23	G#	830	45
B	247	24	A	880	46
C	262	25	A#	930	47
C#	277.2	26	B H	987.8	48
D	293.6	27	C	1046.4	49
D#	311.6	28	C#	1106	50
E	329.6	29	D	1174.6	51
F	349.2	30	D#	1244	52
F#	370	31	E	1318.6	53
G	392	32	F	1397	54
G#	416	33	F#	1480	55
A	440	34	G	1568	56
A#	466	35	G#	1660	57
B	493.2	36	A	1760	58
** C	523.2	37	A#	1864	59
C#	554.8	38	B	1975.6	60
D	587.4	39	C	2093	61
D#	622	40	C#	2217.4	62
E	659.2	41	D	2349.4	63
F	698.4	42			

* Siehe PLAY-Anweisung für die Benutzung dieser Nummern.
** Mittleres C.

Format: KKK.K

In der vorangegangenen Tabelle werden die Noten mit den jeweiligen Frequenzen angegeben. Die Abstimmnote A hat eine Frequenz von 440.

Bemerkungen: Die SOUND-Anweisung erzeugt einen Ton, der beibehalten wird, bis eine andere SOUND-Anweisung angetroffen wird. Wird eine SOUND-Anweisung mit einer Dauer von 0 angetroffen, so wird jede gerade ausgeführte SOUND-Anweisung ausgeschaltet. (Wird keine SOUND-Anweisung ausgeführt), so hat SOUND freq,0 keine Auswirkung.)

Die Töne können gepuffert werden, so daß die Programmausführung nicht angehalten wird, wenn eine neue SOUND-Anweisung angetroffen wird. (Siehe MB-Befehl, der unter der PLAY-Anweisung erläutert wird.)

Um Ruhepausen zu erzeugen, wird SOUND 32767, Dauer benutzt.

	Tempo	Takte/ Minute	Schritte/ Takt (Dauer
Sehr langsam	Larghissimo		
↓	Largo	40-60	27.3-18.2
↓	Larghetto	60-66	18.2-16.55
↓	Grave		
↓	Lento		
↓	Adagio	66-76	16.55-14.37
Langsam	Adagietto		
↓	Andante	76-108	14.37-10.11
Mittel	Andantino		
↓	Moderato	108-120	10.11-9.1
Schnell	Allegretto		
↓	Allegro	120-168	9.1-6.5
↓	Vivace		
↓	Veloce		
↓	Presto	168-208	6.5-5.25
Sehr schnell	Prestissimo		

Die Dauer eines Taktes wird anhand der Takte pro Minute berechnet. Die Takte pro Minute müssen in 1.092 geteilt werden (die Anzahl von Taktschritten in einer Minute). In der vorangegangenen Tabelle werden typische Tempi je nach Taktschritten (Dauer) dargestellt.

Beispiel: Das nachfolgende Programm erstellt ein Glissando.

```
10 FOR I=220 TO 2200 STEP 20
20 SOUND I,0.5
30 NEXT
40 FOR I=1200 TO 220 STEP -20
50 SOUND I,0.5
60 NEXT
```

WIDTH-ANWEISUNG

Format: WIDTH <Dateinummer>,<Größe>
WIDTH <Einheit>,<Größe>
WIDTH <Größe>

Dateinummer
Numerischer Ausdruck in dem Ganzzahlbereich von 1—255. Dieser Ausdruck gibt die Nummer einer eröffneten Datei an.

Größe
Numerischer Ausdruck in dem Ganzzahlbereich von 1—255. Dieser Ausdruck gibt die neue Breite an.

Einheit
Zeichenfolgenausdruck, der die Einheit kennzeichnet. Gültige Einheiten sind SCRN: und LPT1:.

Zweck: Legt die Zeilenbreite einer Druckzeile in Anzahl von Zeichen fest.

Bemerkungen; WIDTH <Dateinummer>,<Größe>
Ist die Datei für LPT1: eröffnet, so wird die Druckzeilenbreite bei dem Zeilendrucker sofort auf die neue angegebene Größe geändert. Mit dieser Anweisung kann die Breite beliebig geändert werden, während die Datei eröffnet ist. Diese Form der WIDTH-Anweisung ist nur für LPT1: gültig.

WIDTH "LPT1:",<Größe>
Wird als verzögerte Breitenzuweisung für den Drucker benutzt. Bei dieser Form der WIDTH-Anweisung wird der neue Breitenwert gespeichert, ohne tatsächlich die aktuelle Breiteneinstellung zu ändern. Eine nachfolgende OPEN"LPT1:"FOR OUTPUT AS <Nummer> benutzt die neue angegebene Größe, während die Datei eröffnet ist.

WIDTH <Größe>
oder
WIDTH "SCRN:",<Größe>
Dieser Befehl hat keine Auswirkung, da der NCR DECISION MATE V stets eine Bildschirmbreite von 80 Zeichen benutzt. NCR GW-BASIC läßt diesen Befehl jedoch ohne Anzeige einer Fehlermeldung zu.

Wird ein beliebiger Wert außerhalb des Bereichs von 1—255 für die Breite oder die Dateinummer angegeben, so wird die Fehlermeldung "Illegal Function Call" angezeigt. Die Breite oder Dateinummer bleibt so bestehen, wie sie vor Eingabe des unzulässigen Wertes war.

Durch Benutzung des WIDTH-Befehls gehen keine Daten verloren. Die Software fügt einfach eine Zeilenschaltung hinzu, nachdem die angegebene Anzahl von Zeichen gesendet wurde. Wird beispielsweise eine 60-Zeichen-Zeile und ein 40-Zeichen-Drucker benutzt und wird WIDTH 40 ausgegeben, so werden die ersten 40 Zeichen auf einer Zeile und die nächsten 20 Zeichen auf der nächsten Zeile ausgedruckt.

DATENÜBERTRAGUNG

In diesem Kapitel werden die BASIC-Anweisungen beschrieben, die zur Unterstützung der synchronen Datenübertragung gemäß RS-232 mit anderen Computern und Peripheriegeräten (mit oder ohne XON-XOFF Protokoll) erforderlich sind.

ERÖFFNEN EINER DATENÜBERTRAGUNGSDATEI

Mit der OPEN COM-Anweisung wird ein Puffer für die Ein-/Ausgabe auf dieselbe Art und Weise wie durch die OPEN-Anweisung bei Diskettendateien zugeordnet. In diesem Zusammenhang wird auf die OPEN COM-Anweisung in dem Kapitel "Anweisungen und Funktionen" verwiesen.

EIN-/AUSGABE BEI DER DATENÜBERTRAGUNG

Da der Datenübertragungspuffer als Datei eröffnet wird, sind sämtliche sequentiellen Ein-/Ausgabeanweisungen, die für Diskettendateien gültig sind, auch für die Datenübertragung gültig.

Sequentielle Eingabeanweisungen für die Datenübertragung sind identisch mit den sequentiellen Eingabeanweisungen für Diskettendateien. Hier handelt es sich im einzelnen um die folgenden Anweisungen:

INPUT#
LINE INPUT#
INPUT$

Die sequentiellen Ausgabeanweisungen für die Datenübertragung sind ebenfalls identisch mit den Ausgabeanweisungen für die Diskettendateien. Hier handelt es sich um folgende Anweisungen:

PRINT
PRINT USING
WRITE#

Für Einzelheiten über das Format und die Benutzung der oben angegebenen Anweisungen und Funktionen wird auf das MS-BASIC Benutzerhandbuch verwiesen.

E/A-Funktionen

Der schwierigste Aspekt der asynchronen Datenübertragung ist die Verarbeitung der Zeichen mit der Geschwindigkeit, mit der sie empfangen werden. Bei Geschwindigkeiten von mehr als 2.400 bps muß die Zeichenübertragung von dem zentralen Rechner solange suspendiert werden, bis schon empfangene Zeichen verarbeitet wurden. Dies kann durch Senden eines XOFF-Befehls (Ctl-S) und XON-Befehls (Ctl-Q) an den zentralen Rechner geschehen. Mit XOFF wird der zentrale Rechner angewiesen, das Senden zu stoppen. Mit XON wird er aufgefordert, das Senden wieder aufzunehmen.

Mit drei Funktionen kann festgestellt werden, wann es zu einer Überlaufbedingung kommen kann:

LOC(x) Gibt die Anzahl von Zeichen in dem Eingabepuffer zurück, die auf das Lesen warten. Stehen mehr als 255 Zeichen in dem Puffer, so gibt LOC(x) 255 zurück. (Der Eingabepuffer kann mehr als 255 Zeichen aufnehmen, wie durch die /C:-Option in dem BASIC-Befehl angegeben.) Bleiben weniger als 255 Zeichen in dem Puffer, so gibt LOC(x) den tatsächlichen Wert zurück.

LOF(x) Gibt den freien Platz im Eingabepuffer zurück. Hier Handelt es sich um dieselbe Angabe wie /C: <Größe> - LOC(x), wobei Größe der Größte des Datenübertragungspuffers wie von der Option /C: festgelegt, entspricht. Die Standardgröße des Puffers beträgt 256.

EOF(x) Gibt "wahr" (-1) zurück, wenn der Eingabepuffer leer ist; gibt "falsch" (0) zurück, wenn Zeichen auf das Lesen werten.

INPUT$-Funktion

Wir empfehlen, die INPUT$-Funktion anstelle der INPUT#- und LINE INPUT#-Anweisungen beim Lesen von Datenübertragungsdateien zu benutzen, da mit dieser Funktion sämtliche gelesenen Zeichen einer Zeichenfolge zugewiesen werden können. Mit INPUT# wird die Eingabe gestoppt, wenn ein Komma oder eine Zeilenschaltung erkannt wird.

Durch INPUT$ wird eine Zeichenfolge mit x Zeichen zurückgegeben, die aus der Datei mit der Nummer Y gelesen wurde. Beim Lesen eines Datenübertragungs-Puffers werden insbesondere die folgenden Anweisungen benutzt:

```
10 WHILE NOT EOF(1)
20 A$=INPUT$(LOC(1),#1
30 . . .
40 . . .
50 . . .
60 WEND
```

Stehen Zeichen in dem Eingabepuffer, so geben die obigen Anweisungen die Zeichen in dem Puffer in A$ zurück und verarbeiten sie (Zeilen 30, 40, 50 usw.). Sind mehr als 255 Zeichen vorhanden, so werden nur 255 Zeichen gleichzeitig zurückgegeben, um einen Überlauf der Zeichenfolge zu vermeiden. Sind mehr als 255 Zeichen vorhanden, so ist darüber hinaus EOF (1) falsch und die Eingabe in A$ wird fortgesetzt, bis der Puffer leer ist.

GET- und PUT-Anweisungen für die Datenübertragung

Die GET- und PUT-Anweisungen unterscheiden sich bei den Datenübertragungsdateien nur geringfügig von den Anweisungen bei Diskettendateien (siehe MS-BASIC Benutzerhandbuch).

Format: GET <Dateinummer>,<nBytes>
PUT <Dateinummer>,<nBytes>

Dateinummer
Gibt die Nummer an, unter der die Datei eröffnet wurde.

nBytes
Gibt die Anzahl von Bytes an, die in die bzw. aus der Datenübertragungsdatei übertragen werden müssen.

Zweck: Ermöglicht eine E/A mit fester Länge zu bzw. von der Datenübertragungsdatei.

Bemerkungen: Aufgrund der niedrigen Leistung bei der Datenübertragung über Telefonleitungen wird empfohlen, daß GET und PUT nicht in derartigen Anwendungen benutzt werden.

STEUERUNGSSIGNALE

Dieser Abschnitt enthält Informationen über die Steuerungssignale, die unter Umständen erforderlich sind, um mit einem anderen Computer oder einem anderen Peripheriegerät zu kommunizieren.

Ausgabesignale

Beim Starten von BASIC in dem NCR DECISION MATE V werden die Signalleitungen "Sendeanforderung" (RTS) und "Datenendeinrichtung betriebsbereit" (DTR) erst eingeschaltet, wenn eine OPEN COM-Anweisung ausgeführt wird. Das RTS-Signal kann durch Angabe der RS-Option in der OPEN COM-Anweisung unterdrückt werden (hier wird auf das Kapitel über die Anweisungen und Funktionen verwiesen). Es sei denn, sie wird unterdrückt, bleibt die Leitung eingeschaltet, bis die Datenübertragungsdatei mit CLOSE, END, NEW, RESET, SYSTEM oder RUN ohne die R-Option abgeschlossen wird. Wird eine OPEN COM-Anweisung nicht erfolgreich ausgeführt, so bleibt die Leitung eingeschaltet. In diesem Fall kann die OPEN COM-Anweisung ohne eine CLOSE-Anweisung erneut ausgeführt werden.

Eingabesignale

Ist die Signalleitung "Sendebereitschaft" (CTS) oder "Betriebsbereitschaft" (DSR) ausgeschaltet, so kann keine OPEN COM-Anweisung ausgeführt werden. BASIC gibt nach einer Sekunde eine Fehlermeldung "Device Timeout" aus. In diesem Fall kann jedoch angegeben werden, ob und wie diese Leitungen getestet werden sollen, indem die CS- und DS-Optionen in der OPEN COM-Anweisung angegeben werden (siehe Kapitel "Anweisungen und Funktionen").

Sind die CTS- oder DSR-Leitungssignale ausgeschaltet, während ein Programm ausgeführt wird, so können die mit der Datenübertragungsdatei verknüpften E/A-Anweisungen nicht ausgeführt werden. Außerdem wird eine Fehlermeldung "Device Fault" oder Device Timeout" angezeigt.

Führt der zentrale Rechner ein Programm aus und sendet der Satellitenrechner Zeichen, so wird nur ein Zeichen in der Hardwareschnittstelle des zentralen Rechners gesichert. Bei der nächsten Ausführung einer Datenübertragungsanweisung durch den zentralen Rechner wird dann eine Fehlermeldung "Device I/O" angezeigt. Mit dieser Fehlermeldung wird ein Überlauf in der Hardwareschnittstelle des zentralen Rechners angegeben.

BEISPIELPROGRAMM

Mit dem folgenden Programm kann der NCR DECISION MATE V als Dialogterminal benutzt werden. Neben der Vollduplex-Übertragung mit einem zentralen Rechner ermöglicht das Programm das Fernladen (Schreiben) von Daten in eine Datei, und umgekehrt das Fernladen (Senden) zu einem anderen Rechner.

Neben der Verdeutlichung der einzelnen Elemente der asynchronen Datenübertragung soll dieses Programm die Übertragung von BASIC-Programmen und Daten zu und von dem NCR DECISION MATE V erläutern.

Hinweise zu dem Beispielprogramm

Zeilennummer	Kommentare
	Beim Starten von GW-BASIC muß der /F: Schalter auf 3 gestellt werden.
10	Versetzt den Bildschirm in den normalen Alpha-Modus.
20	Schaltet die Anzeige der programmierbaren Funktionstasten aus, löscht den Bildschirm und prüft, ob sämtliche Dateien abgeschlossen sind.

HINWEIS: Asynchron impliziert die Zeichen-E/A im Gegensatz zur Zeilen- oder Block-E/A. Deshalb werden sämtliche PRINT-Befehle (zu der Datenübertragungsdatei, dem Bildschirm oder einer Diskettendatei) mit einem Semikolon (;) beendet. Dadurch wird die normalerweise am Ende einer PRINT-Anweisung ausgegebene Zeilenschaltung gestoppt.

30	Definiert sämtliche numerischen Variablen als Ganzzahlen. Dies ist im wesentlichen zur Benutzung in der Subroutine in den Zeilen 500 bis 660 gedacht. Jedes Programm, bei dem es auf eine Optimierung der Geschwindigkeit ankommt, sollte wo immer möglich, Ganzzahlen in Schleifen benutzen.
35-40	Löscht die 23ste Zeile ab Spalte 1.
50	Definiert die booleschen Werte "wahr" und "falsch".
70	Definiert die ASCII-Zeichen XON und XOFF.
100-130	Druck die Programmkennzeichnung aus und fordert die Baudrate (Geschwindigkeit) an. Eröffnet die Übertragung zu der Datei Nr. 1 mit gerader Parität, sieben Datenbits und einem Zeilenvorschub (LF) im Anschluß an jede Zeilenschaltung.

200-280 In diesem Abschnitt erhält der Benutzer ein Menü für den Empfang von Daten auf dem Bildschirm oder in einer Datei oder für das Senden von Daten von der Tastatur oder von einer der Dateien.

1. Der Benutzer wird gefragt, wieviele Zeichen auf der Übertragungsleitung empfangen werden sollen, bevor sie auf dem Bildschirm angezeigt werden.
2. Liest eines oder mehrere Zeichen von der Tastatur in A$ und sendet A$. Mit dem Menü wird der Benutzer durch die weitere Operation geleitet.
3. Wird nur ein Leerzeichen eingegeben, so wird auf n Zeichen gewartet und nach Empfang dieser Zeichen gedruckt.
4. Wurde nur das Zeichen M eingegeben, so ist der Benutzer für das Fernladen einer Datei bereit. Deshalb muß der Dateiname ermittelt werden.
5. Wurde nur ein E eingegeben, so stoppt das Programm bei 9000-9040.
6. Handelt es sich bei der Eingabe (A$) nicht um M, E oder ein Leerzeichen, so wird die Eingabe durch Schreiben in die Datenübertragungsdatei (PRINT #1 . . . .) wie in Schritt 2 beschrieben, gesendet. In Zeile 230 wird zu dem Menü zurückgegangen.
7. In den Zeilen 250 bis 260 wird der Inhalt des Datenübertragungspuffers (wie durch n ausgewählt) auf dem Bildschirm gelesen und angezeigt. Nun wird mit 1 fortgefahren.

300-310 Ermitteln des zu benutzenden Diskettendateinamens.

400-430 Fragt, ob der Dateiname gesendet oder empfangen werden soll und eröffnet die Datei.

490-540 Die empfangenen Daten füllen eine Matrix mit 126 Positionen auf, es sei denn, ein Dateiendezeichen (Zeile 530) wurde empfangen. In diesem Fall wird die Datei abgeschlossen.

550-620 Bevor in die ausgewählte Diskettendatei geschrieben wird, wird XOFF an den Sender gesendet. Zwei zusätzliche Zeichen (Zeilen 560-590) können gelesen werden, nachdem die 126 Positionen aufgefüllt wurden und bevor der Sender XOFF erhält.

625 Nachdem die Matrix vollständig in die Diskettendatei geschrieben und XON an den Sender gesendet wurde, fährt der Sender mit dem Senden fort.

630 Fortsetzung des Empfangs wie in Zeile 500.

640-680 Beim Dateiende werden die letzten Zeichen in die Datei geschrieben und die Datei abgeschlossen. Nun wird wieder imit dem Menü fortgefahren.

800-880 Hier handelt es sich um eine Warteroutine, die benutzt wird, wenn der Sender auch Zeichen empfängt. Empfängt der Sender XOFF, so muß vor Fortsetzung des Sendens gewartet werden, bis XON empfangen wird.

1000-1060 Hier handelt es sich um eine Senderoutine. Bis zum Ende der Diskettendatei wird folgendermaßen fortgefahren:

Einlesen eines Zeichens in A$ mit der INPUT$-Anweisung. Senden des Zeichens an die Datenübertragungseinheit in 1015. (Wird ein Zeichen empfangen, so wird die Warteroutine für XON anstelle von XOFF aufgerufen, Zeile 1015.) Ctl-Z wird beim Dateiende in Zeile 1040 gesendet, wenn die Empfangseinheit ein derartiges Zeichen für das Abschließen der Datei benötigt. Schließlich wird in den Zeilen 1050 und 1060 die Diskettendatei abgeschlossen, die Beendigungsmeldung ausgedruckt und zum Dialogmodus in Zeile 200 zurückgegangen.

9000-9040 Diese Zeilen werden ausgeführt, wenn E als Antwort auf das Menü eingegeben wird. Diese Zeilen schließen die Datenübertragungsdatei und die Ausgabedatei auf dem Bildschirm ab, stellen die Anzeige mit den programmierbaren Funktionstasten wieder her und beenden das Programm.

```
10 SCREEN 0
20 KEY OFF:CLS:CLOSE
30 DEFINT A-Z
35 LOCATE 23,1
40 PRINT STRING$(60," ")
50 FALSE=0:TRUE= NOT FALSE
70 XOFF$=CHR$(19):XON$=CHR$(17)
100 LOCATE 23,1:PRINT "Async TTY Program          ";
110 LOCATE 1,1:LINE INPUT "speed? ";SPEED$
120 COMFIL$="com1:"+SPEED$+",e,7,,LF
130 OPEN COMFIL$ AS #1
140 OPEN "scrn:" FOR OUTPUT AS #2
200 LOCATE 1,1:LINE INPUT "on receiving, wait for n characters, n = ";N$
203 N%=VAL(N$)
205 LOCATE 3,1:PRINT "press any keys for transmittion"
206 PRINT "except: M    for file i/o"
207 PRINT "or       space for receiving"
208 PRINT "or       E     for ending program"
209 LINE INPUT;A$
210 IF A$=" " THEN 250
211 IF A$="M" THEN 300
212 IF A$="E" THEN 9000
220 PRINT #1,A$;
230 GOTO 200
250 A$=INPUT$(N%,#1)
260 PRINT #2,A$;
280 GOTO 200
300 LOCATE 8,1
310 LINE INPUT"file? ";DSKFIL$
400 LOCATE 9,1
410 LINE INPUT"(T)ransmit or(R)eceive? ";TXRX$
420 IF TXRX$="T" THEN OPEN DSKFIL$ FOR INPUT AS #3:GOTO 1000
430 OPEN DSKFIL$ FOR OUTPUT AS #3
490 DIM BUF$(128)
500 FOR J=1 TO 126
520 BUF$(J)=INPUT$(1,#1)
530 IF BUF$(J)=CHR$(26) THEN GOTO 640
540 NEXT J
550 PRINT #1,XOFF$;
560 IF LOC(1)=0 THEN K=126:GOTO 600
570 BUF$(127)=INPUT$(1,#1)
580 IF LOC(1)=0 THEN K=127:GOTO 600
585 BUF$(128)=INPUT$(1,#1)
590 K=128
600 FOR I=1 TO K
610 PRINT #3,BUF$(I);
620 NEXT I
625 PRINT #1,XON$;
630 GOTO 500
640 FOR I=1 TO J
650 PRINT #3,BUF$(I);
660 NEXT I
670 CLOSE #3:CLS:LOCATE 24,10:PRINT "* download complete *"
680 GOTO 200
```

```
800 B$=INPUT$(1,#1)
810 IF B$=XOFF$ THEN GOTO 850
820 PRINT #2,B$;
830 IF LOC(1)=0 THEN RETURN
840 GOTO 800
850 B$=INPUT$(1,#1)
860 IF B$=XON$ THEN RETURN
870 PRINT #2,B$;
880 GOTO 850
1000 WHILE NOT EOF(3)
1010 A$=INPUT$(1,#3)
1015 PRINT #1,A$;
1020 IF LOC(1)>0 THEN GOSUE 800
1030 WEND
1040 PRINT #1,CHR$(26); 'ctrl-Z to make close file.
1050 CLOSE #3:CLS:LOCATE 23,10:PRINT "** upload complete ***;
1060 GOTO 200
9000 CLOSE #1
9010 CLOSE #2
9030 KEY ON
9040 END
```

HINWEIS: In dem obigen Beispiel muß folgende Zeile für Baudraten von 4.800 bps und mehr aufgenommen werden:

1014 FOR I=1 TO 10:NEXT

Wie schon vorher erwähnt, muß bei der Entwicklung eines Datenübertragungsprogramms sowohl die Baudrate des zentralen Rechners als auch die Baudrate des Satellitenrechners berücksichtigt werden. Kommt es zu einer Fehlermeldung "Device I/O", so gibt sie normalerweise einen Überlauf bei der Hardwareschnittstelle an. In diesem Fall muß das Programm entsprechend angepaßt werden.

ANHANG A

GW-BASIC FEHLERMELDUNGEN

Bei GW-BASIC werden folgende Fehlermeldungen angezeigt:

Nummer	Meldung
24	Device Timeout Wird angezeigt, wenn eines der zu testenden Signale (CTS, DSR oder CD) beim Eröffnen einer Datei fehlt, oder wenn der zentrale Rechner CTS, DSR oder CD verliert, während er darauf wartet, Daten in den Ausgabepuffer zu setzen.
25	Device Fault Diese Meldung wird angezeigt, wenn der zentrale Rechner DSR oder CD verliert.
57	Device I/O Error Diese Fehlermeldung wird angezeigt, wenn es zu einem Überlauf-, Paritäts- oder Rahmenfehler bei der Datenübertragung kommt. Hier handelt es sich um einen schwerwiegenden Fehler, d.h. das Betriebssystem kann den Fehler nicht beheben.
68	Device Unvailable Es wurde versucht, eine Datei in einer nicht vorhandenen Einheit zu eröffnen. Unter Umständen war die Hardware zur Unterstützung der Einheit, wie beispielsweise LPT2: oder LPT3: nicht vorhanden oder war deaktiviert. Zu dieser Fehlermeldung kommt es, wenn eine OPEN COM1-Anweisung ausgeführt wird, während die Unterstützung von RS-232 über den /C:0 Schalter auf der Befehlszeile deaktiviert war.
69	Communication Buffer Overflow Zu dieser Fehlermeldung kommt es, wenn eine Eingabeanweisung für die Datenübertragung ausge-

Nummer	Meldung
	führt wird, während der Eingabepuffer schon voll ist. Kommt es zu dieser Fehlerbedingung, so wird die ON ERROR GOTO-Anweisung benutzt, um die Eingabe erneut vorzunehmen. Nachfolgende Eingaben versuchen, diesen Fehler zu löschen, es sei denn, die Zeichen werden weiterhin schneller empfangen, als das Programm sie verarbeiten kann. In diesem Fall stehen verschiedene Optionen zur Verfügung: Erweitern der Größe des Empfangspuffers für die Datenübertragung über den /C:-Schalter. Implementieren des XON/XOFF-Protokolls in dem zentralen Rechner/Satellitenrechner, um das Senden so lange auszuschalten, bis die Zeichen in dem Eingabepuffer verarbeitet sind. Benutzung einer kleineren Baudrate für das Senden und Empfangen.
70	Disk Write Protect Diese Fehlermeldung wird angezeigt, wenn versucht wird, auf eine schreibgeschützt Diskette zu schreiben. Für die Fehlerbehebung wird eine ON ERROR GOTO-Anweisung benutzt.
72	Disk Media Error Diese Fehlermeldung wird angezeigt, wenn die FDC-Steuereinheit einen Hardware- oder Datenträgerfehler feststellt. Diese Fehlermeldung gibt im allgemeinen einen fehlerhaften Datenträger an. Eventuell vorhandene Dateien müssen auf eine neue Diskette kopiert und die beschädigte Diskette muß neu formatiert werden. Durch FORMAT werden die fehlerhaften Spuren gekennzeichnet und in eine Datei für fehlerhafte Spuren gesetzt. Der Rest der Diskette kann nun benutzt werden.

ANHANG B

ERWEITERUNG DER TERMINAL-FUNKTIONSCODES

GW-BASIC bietet Ihnen erweiterte Zugriffsmöglichkeiten zu den Funktionen an, die den Bildschirm, den Lautsprecher und die Funktionstasten Ihres NCR DECISION MATE V steuern. Aufgrund dieser Erweiterungen entfallen die im MS-DOS Benutzer-Handbuch (Anhang C) angegebenen Terminal-Funktionscodes, die mit ESC beginnen. Nachstehend finden Sie Beispiele der Benutzung der entsprechenden GW-BASIC-Anweisungen.

POSITIONIEREN DES CURSORS

Hierzu sollten Sie die LOCATE-Anweisung verwenden.

Beispiele: LOCATE 12,40 setzt den Cursor ungefähr in die Mitte des Bildschirms; nach der Ausführung von LOCATE 1,1 befindet sich der Cursor in der oberen linken Ecke des Bildschirms.

Wenn Sie in Ihrem Programm die gegenwärtige Position des Cursors in zwei Variablen, z.B. Zeile% und SAPLTE%, laufend festhalten, können Sie ihn auch relativ zu dieser Position bewegen.

LÖSCHEN DES BILDSCHIRMS

Für das Löschen des Bildschirms steht Ihnen die CLS-Anweisung zur Verfügung.

Wenn Sie die Zeile, in der der Cursor sich befindet, ab der Cursor-Position löschen wollen, können Sie die PRINT-Anweisung mit einer entsprechenden Anzahl von Leerstellen verwenden.

Beispiel: PRINT " ";

Die folgenden Anweisungen löschen die Zeile ab der in SPALTE% festgehaltenen Position:

```
FOR I%=1 TO 81-SAPLTE%:PRINT CHR$(32);:NEXT I%
```

Die GW-BASIC-Anweisung CSRLIN ergibt die Nummer der Bildschirmzeile, in der sich der Cursor befindet. Mit Hilfe dieser Anweisung können Sie den ganzen Bildschirm ab der gegenwärtigen Cursor-Position löschen. Hierzu müssen Sie zuerst die laufende Zeile gemäß der oben beschriebenen Verfahrensweise löschen. Der Cursor steht nun am Anfang der nachfolgenden Zeile. Löschen Sie diese Zeile mit

```
FOR I%=1 TO 80:PRINT CHR$(32);:NEXT I%
```

Bevor Sie eine Zeile löschen, müssen Sie anhand der CSRLIN-Anweisung die Nummer der Zeile ermitteln, an deren Anfang der Cursor sich nun befindet. Wiederholen Sie diesen Vorgang für das Löschen ganzer Zeilen, bis die 24. Zeile gelöscht worden ist.

GRAFIK-ATTRIBUTE

Die COLOR-Anweisung gibt Ihnen die Möglichkeit, die Farben für Vordergrund (Schrift) und Hintergrund zu bestimmen.

Die Inversion der Darstellung am Bildschirm wird ebenfalls mit Hilfe der COLOR-Anweisung erreicht. Wenn Ihr NCR DECISION MATE V einen einfarbigen Bildschirm besitzt, benutzen Sie die Anweisung COLOR 0,2. Die Anweisung 2,0 macht diese Inversion rückgängig: Die Darstellung erfolgt wieder als grüne Zeichen auf schwarzem Hintergrund.

Bei einem Farbbildschirm ist es zweckmäßig, die gegenwärtige Zusammenstellung der Vordergrund- und Hintergrund-Farben in zwei Variablen (z.B. SCHRIFT% und PAPIER%) laufend festzuhalten. Die Inversion der Darstellung am Bildschirm wird dadurch erreicht, daß Sie die Inhalte dieser zwei Variablen gegeneinander austauschen und die Anweisung COLOR SCHRIFT%,PAPIER% ausführen.

DER LAUTSPRECHER

Die BEL-Funktion (ASCII 7) wird von der BEEP-Anweisung erfüllt. Ferner können Sie mit PLAY und SOUND Musik programmieren.

DIE FUNKTIONSTASTEN

Die Ausführung der KEY-Anweisung bewirkt die Programmierung einer beliebigen Funktionstaste Ihres NCR DECISION MATE V.

Lizenzprogramme für den Personalcomputer NCR - DM V

Bitte füllen Sie diese Karte aus und senden Sie an umstehende Adresse.

Ich bestätige hiermit, das (die) nachfolgende(n) aufgezählte(n) Programm(e) erhalten zu haben:

```
MS-PRODUCT       : GW (TM)-BASIC
MS VERSION NO.   : 1.4
SERIAL NO.       : 08386
```

Für die Benutzung der vorstehenden Programme gelten die Allgemeinen Lizenzbedingungen der NCR GmbH, die ich erhalten habe und mir bekannt sind.

Name: _____

Vorname: _____

Anschrift: _____

Datum: _____ _____

Unterschrift

Absender

(Postleitzahl) (Ort)

Gebühr
bezahlt
Empfänger

Antwort-
Postkarte

An die
NCR GmbH
Ulmer Straße 160 b

8900 Augsburg