Computersystem

P 6060

Bedienungshandbuch

olivetti

Computersystem

P 6060

Bedienungshandbuch

olivetti

INHALT

1. COMPUTERSYSTEM P6060

1. DAS COMPUTERSYSTEM P 6060

1.1 DIE STANDARDKONFIGURATION

Die Standardkonfiguration (Abbildung 1.1) besteht aus folgenden Komponenten:

- ZENTRÄLEINHEIT
- TASTATUR
- KONSOLE
- DISPLAY
- FLOPPY-DISK-EINHEIT MIT 2 LAUFWERKEN
- THERMODRUCKER

Abb. 1.1 - Das System P 6060: Die Standardkonfiguration

1.1.1 Die Zentraleinheit

Die Zentraleinheit besteht aus:

- dem Leitwerk
- dem Rechenwerk (Arithmetic-Logic-Unit: ALU)
- dem Hauptspeicher (RAM).

Das Leitwerk

- koordiniert und kontrolliert die Operationen des Systems.

Das Rechenwerk (ALU)

- führt die arithmetischen Operationen durch (Addition, Subtraktion, usw.).
- führt logische Operationen durch, die es dem Leitwerk ermöglichen, aus verschiedenen Alternativen in der Ausführung eine bestimmte auszuwählen.

Der Hauptspeicher besteht aus integrierten MOS-Bauelementen und enthält:

- Mikroprogramme, die aus einer Folge von Mikroanweisungen bestehen und die bei Aktivierung durch das Leitwerk die entsprechenden Instruktionen ausführen.
- Basis-Softwareprogramme zur Erstellung, Ausführung und Modifikation der Anwenderprogramme.
- Anwenderprogramme.
- die zu verarbeitenden Daten und die Ergebnisse der Operationen.

Man kann daher im Hauptspeicher zwei Teile unterscheiden:

Ein Teil ist für das System reserviert und enthält die Betriebssoftware;
der andere Teil steht dem Anwender zur Verfügung und hat eine Mindestkapazität von 8 K (8192 byte), wobei ein Byte aus 8 Bit besteht.

1.1.2 Die Tastatur

Die Tastatur besteht aus 96 monostabilen Tasten und dient dazu, dem System Daten, Befehle und Instruktionen über den Tastaturpuffer mitzuteilen. Der Tastaturpuffer kann bis zu 80 Zeichen aufnehmen.

Die Tastatur (Abbildung 1.2) ist in Tastenfelder unterteilt:

- Basic- u. Alphanumerische Tasten
- Editing-Tasten
- Algebraische Tasten
- End-of-Line Tasten
- Befehls-Tasten
- Funktions-Tasten

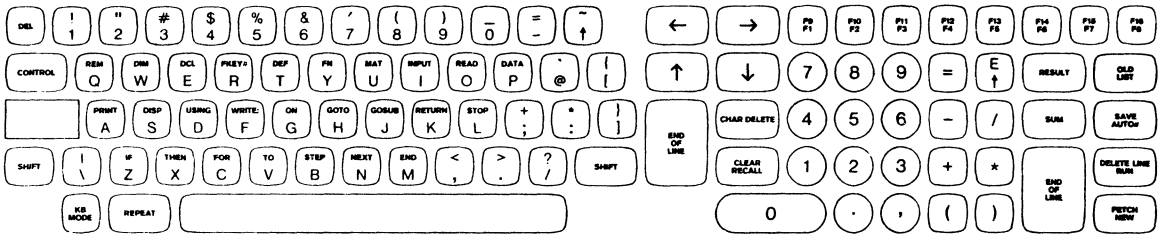


Abb. 1.2 - Die Tastatur

- BASIC- u. ALPHANUMERISCHES TASTENFELD

Dieses Feld besteht aus den in Abbildung 1.3 dargestellten Tasten:

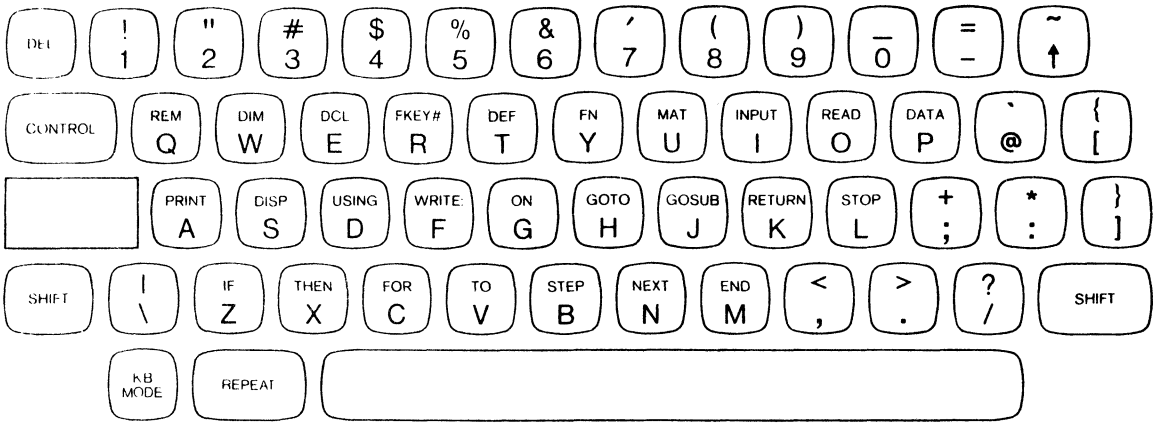


Abb. 1.3 - Basic- u. alphanumerische Tasten

Mit diesen Tasten kann man dem System eingeben:

- Programmanweisungen wie: 100 IF A = B3 THEN 120
- Rechnen im Calculator-Mode wie: ⧗1 = (3*⧗↑2.5 - PI)/4
- Numerische Daten wie: -135.2
- Strings (Zeichenketten) wie: MAX MEIER
- Systemanweisungen wie: SAVE U, PROG9
- Befehle zum Aufrufen und Ausführen von Dienstprogrammen wie: EXEC FDCOPY, U

Die Basic- u. alphanumerischen Tasten erzeugen die 128 ISO-Code-Zeichen und die Codes für 26 Schlüsselwörter der Basic-Instruktionen.
Die ISO-Code-Zeichen sind im Anhang 1 dargestellt.

Die alphanumerische Tastatur enthält außerdem die folgenden Tasten:

(DELETE) erzeugt das ISO-Zeichen DEL, das im Terminalbetrieb gebraucht wird. Wird dieses Zeichen ausgedruckt oder im Display angezeigt, so entspricht ihm das Symbol ❄ .

(CONTROL) ist im Terminal-Mode notwendig und ergibt zusammen mit einer alphanumerischen Basic-Taste gedrückt, die in Anhang 1 angegebenen Zeichen der Spalten 0 und 1 der ISO-Tabelle,

(SHIFT): Wird Shift zusammen mit einer Taste mit zwei Bedeutungen gedrückt, so wird das obere Zeichen dargestellt. Auf diese Weise erfolgt auch der Zugriff zu den Basic-Schlüsselworten. Die Shifttaste entspricht also der Umschalttaste bei Schreibmaschinen.

(KEYBOARD MODE): Wird diese Taste gedrückt, so gestattet dies den Gebrauch des alphanumerischen Basic-Teiles wie bei einer Schreibmaschine.

Es ist bei gedrückter Taste KB-MODE möglich, Groß- und Kleinbuchstaben zu erzeugen, je nachdem, ob die SHIFT-Taste zusätzlich zum jeweiligen Zeichen gedrückt wird oder nicht.
Ist die KB-MODE Taste gedrückt, so leuchtet die Anzeige am linken Rand der Tastatur.
Soll der KB-MODE wieder ausgeschaltet werden, so ist wieder die Taste KB-MODE zu drücken.

SONDER ZEICHEN

Folgende Zeichen können generiert werden:

- die Ziffern 0 bis 9
- die Interpunktionszeichen: , . ; : ! ? "
- die Klammern: ( ) [ ] { }
- die Akzente: ´ `
- die Vergleichs- und Zuweisungszeichen: > < =
- die arithmetischen Operationszeichen: + - * / ↑
- die Zeichen: $ % ~ - / \ # &

(REPEAT) zusammen mit einer anderen Taste gedrückt, bewirkt eine Wiederholung dieses Zeichens, bis eine der beiden Tasten gelöst wird.

BLANK TASTE

Mit dieser Taste können Leerstellen erzeugt werden.

- TASTENFELD EDITING

(ausführliche Beschreibung siehe Kap. 7)

Der Editing-Teil besteht aus den 7 Tasten, die in Abbildung 1.4 dargestellt sind.

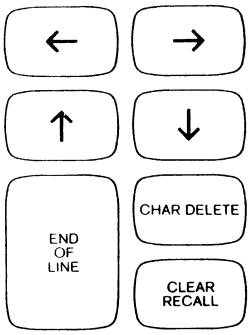
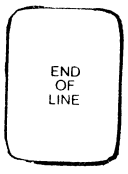


Abb. 1.4 - Editing



(EOL) beendet die Eingabe über die Tastatur.

Verschieben des Pointers im Display nach links bzw. rechts

Anzeige der nächstfolgenden bzw. vorangehenden Programm- oder Textzeile

(CHARACTER DELETE) löscht das Zeichen links vom Pointer und verschiebt etwaige Zeichen rechts davon um eine Stelle nach links.

(CLEAR/RECALL)

. zusammen mit SHIFT gedrückt, bewirkt dies das Löschen aller Zeichen, die sich im Tastaturpuffer befinden.

. ohne SHIFT wird im Display anstelle des jetzt ausgegebenen Textes der Inhalt des Displaypuffers angezeigt.

- ALGEBRAISCHE TASTEN

Der algebraische Teil besteht aus 21 Tasten, die in Abbildung 1.5 dargestellt sind.

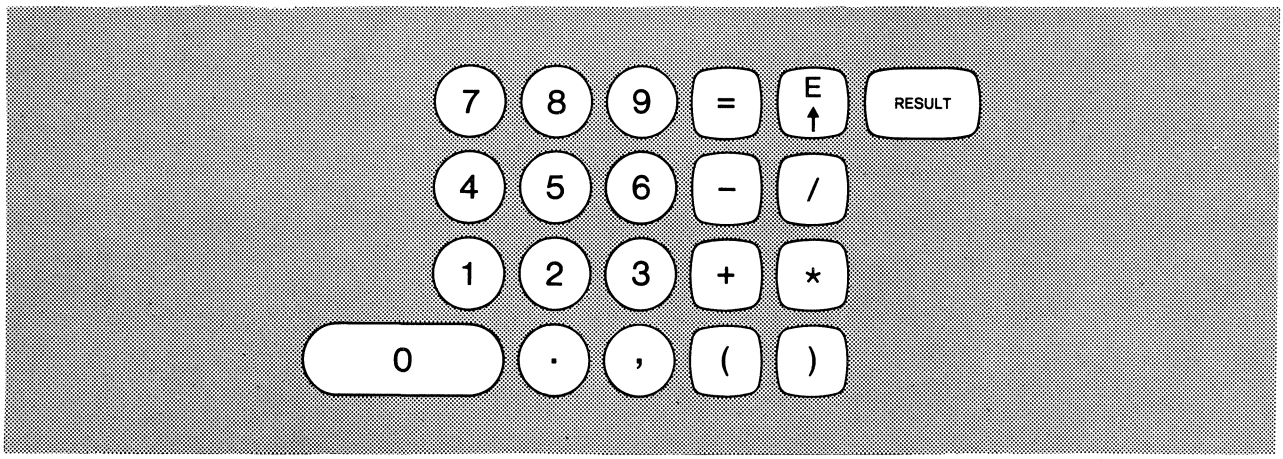


Abb. 1.5 - Algebraisches Tastenfeld

- NUMERISCHE TASTEN

Sie erzeugen die Dezimalziffern 0 bis 9

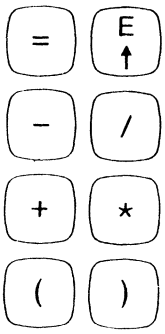
Dezimalpunkt : Ersetzt das Komma in der Dezimaldarstellung einer Zahl.

Diese Taste dient zur Eingabe von Zahlen in Exponential-Darstellung; die nach E eingegebene Zahl wird vom System als Exponent von 10 interpretiert. Dagegen gilt bei Verwendung des Zeichens ↑: Die auf ↑ folgende Zahl wird als Exponent interpretiert.

Werden der Reihe nach die Tasten 1 2 E 5 gedrückt, so wird die Zahl 12×10^5 gebildet. Drückt man hingegen die Tasten 1 2 ↑ 5 , so wird die Zahl 12^5 gebildet.

- ARITHMETISCHE TASTEN

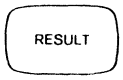
Das System führt folgende arithmetische Rechenoperationen aus:



Addition + Subtraktion - Multiplikation *
Division / Potenzierung E ↑ Zuweisung =

(und) : Durch Setzen von Klammern "(" und ")" kann die Reihenfolge der Operationen in arithmetischen Ausdrücken geändert werden.

Anmerkung: Alle vorhin beschriebenen Tasten sind auch im alphanumerischen Tastenfeld enthalten und haben dieselbe Wirkung.



(RESULT) Diese Taste wird verwendet, wenn man im Calculator-Mode oder im Debugging-Mode arbeitet (siehe RECHNEN IM CALCULATOR MODE Kapitel 3 und 11).

- ABSCHLIESSEN EINER EINGABEZEILE

Dieser Teil enthält die 2 Tasten aus Abbildung 1.6

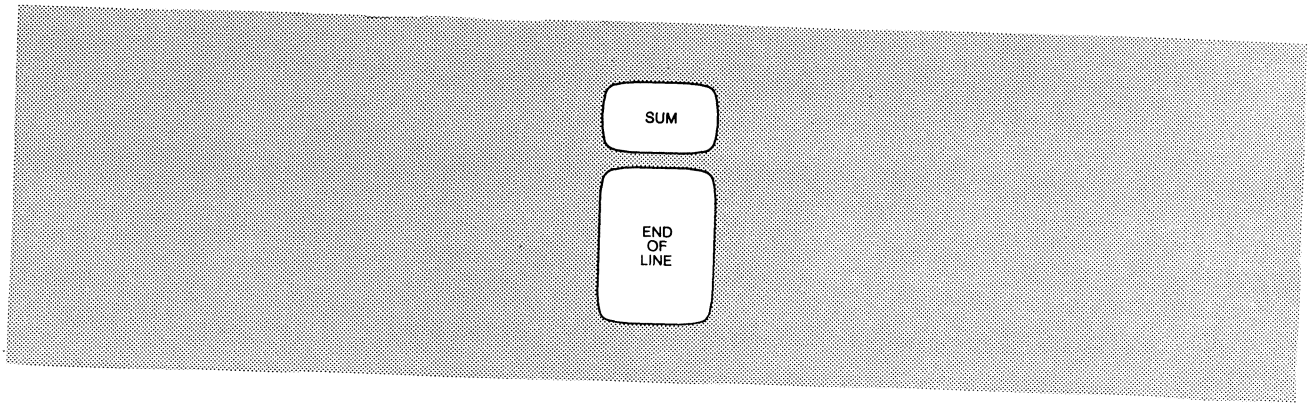
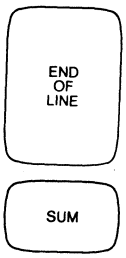


Abb. 1.6 - Abschließen einer Eingabezeile



(EOL) hat die gleiche Wirkung wie die im EDITING-Feld beschriebene Taste.

(SUM) dient zur Summierung von Ergebnissen im Calculator-Mode (Kap. 11)

- BEFEHLS-TASTEN

Dieses Feld besteht aus 4 Tasten, wie sie in Abbildung 1.7 dargestellt sind:

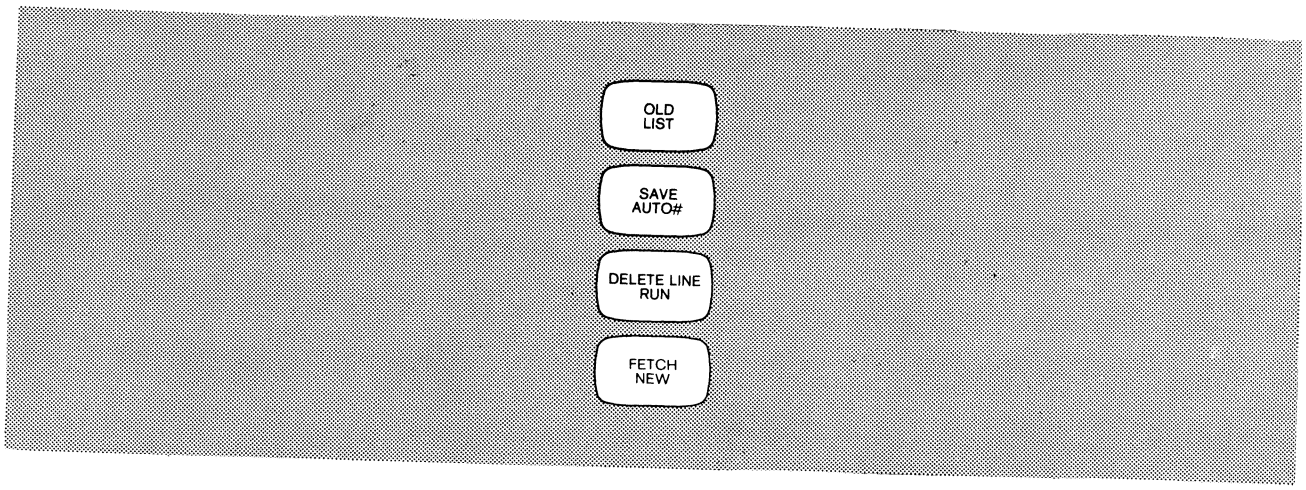


Abb. 1.7 - Befehls-Tasten

Damit können die 8 am häufigsten benötigten System-Befehle aufgerufen werden. Ihre Bedeutung ist in Kap. 6 ausführlich beschrieben.

- DIE FUNKTIONSTASTEN

Dieses Feld besteht aus 8 Tasten, wie sie in Abbildung 1.8 dargestellt sind:

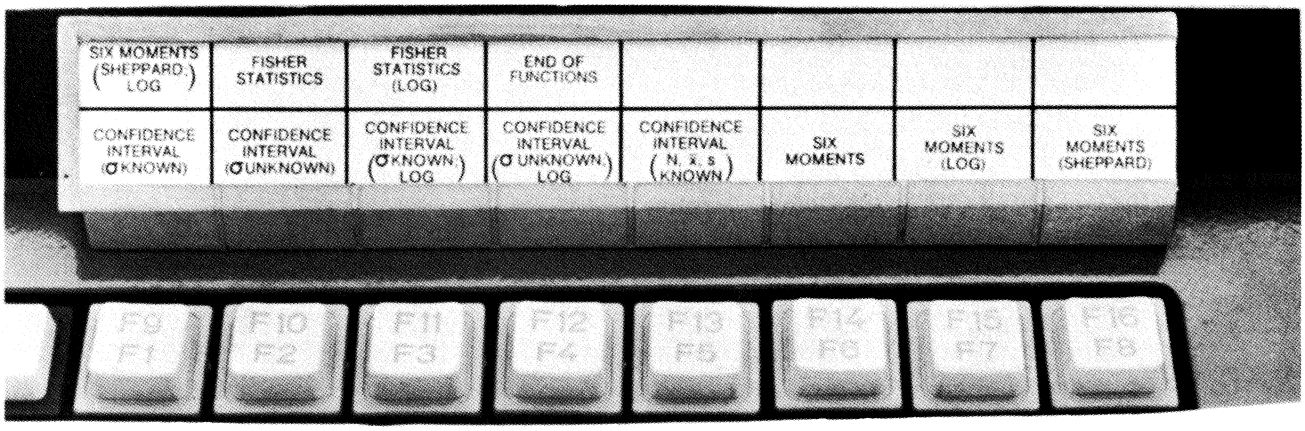


Abb. 1.8 - Funktionstasten

Man kann den 16 Funktionstasten F1 bis F 16 (siehe Abb. 1.8) selbstgewählte Zeichenfolgen zuordnen, sodaß jedesmal, wenn eine solche Taste gedrückt wird, die entsprechende Zeichenfolge in den Tastaturpuffer geladen wird. Jeder Taste sind zwei Funktionen zugeordnet, die obere Funktion wird mit der Taste SHIFT aktiviert. (Siehe Kapitel 9 und 10 für die Verwendung der Funktionstasten)

1.1.3 Die Konsole

Die Konsole enthält 7 Tasten, 4 Leuchtanzeigen und ein Dezimalstellenrad, wie dies aus Abbildung 1.9 ersichtlich ist. Die 7 Tasten leuchten auf, wenn sie aktiviert sind.

Die Aktivierung der Tasten wird durch nochmaliges Drücken aufgehoben.

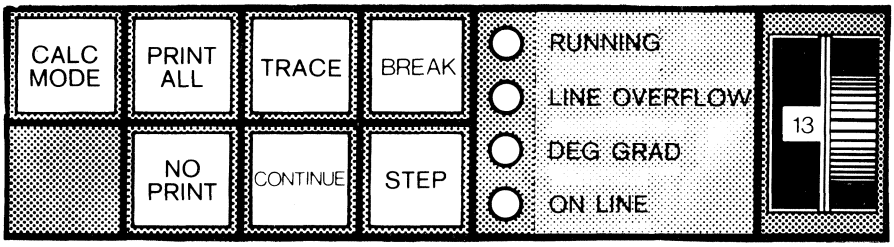


Abb. 1.9 - Die Konsole

(CALCULATOR MODE) drückt man diese Taste, so kann manuell gerechnet werden; dieser Zustand wird durch das Aufleuchten der Taste angezeigt.

(PRINT ALL) Diese Taste bewirkt, daß alle Texte, die im Display erscheinen, auch gedruckt werden; dies wird durch das Aufleuchten der Taste angezeigt.

(NO PRINT) Drückt man diese Taste, so werden alle vom Programm oder System bewirkten Ausdrucke unterdrückt; Dieser Zustand wird durch das Aufleuchten der Taste angezeigt.

(BREAK) Wird diese Taste gedrückt, so unterbricht man die Abarbeitung des laufenden Programmes. Das System geht in den Command-Mode über.

Die folgenden 3 Tasten werden für das Debugging verwendet:

(TRACE) Wird diese Taste gedrückt, so werden die Zeilennummern der ausführbaren Anweisungen eines Programmes während der Abarbeitung gedruckt. Der Ausdruck der Nummern erfolgt entsprechend der Reihenfolge der Ausführung.

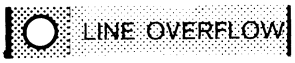
Anmerkung : Die Taste NO PRINT dominiert die Wirkung der Tasten PRINT ALL und TRACE.

Dieser Zustand wird durch das Aufleuchten der Taste angezeigt. Diese Funktion wird durch eine im Programm vorkommende Basic-Anweisung TRACE OFF automatisch wieder aufgehoben.

(STEP) Drückt man diese Taste, so wird die Ausführung des Programmes unterbrochen. Mit jedem weiteren Tastendruck wird ein einzelner Schritt des Programmes durchgeführt. Im aktivierten Zustand leuchtet die Taste.

(CONTINUE) Drückt man die Taste, wird die Abarbeitung des unterbrochenen Programmes wieder aufgenommen.

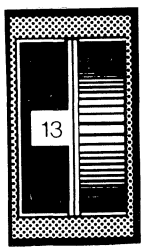
Während der Ausführung von Programmen blinkt diese Lampe, in allen anderen Betriebszuständen leuchtet diese Lampe konstant.



Leuchtet die Lampe auf, so bedeutet dies, daß versucht wird, mehr als 80 Zeichen in den Tastatur-Puffer einzugeben.

Bei Eingabe der Anweisungen SDEG oder SGRAD (im Calculator-Mode) leuchtet diese Lampe auf. Sie gibt an, daß für die Winkelfunktionen das Argument in Altgrad oder Neugrad vorliegt bzw. errechnet wird.

Leuchtet diese Lampe, so arbeitet das System als Terminal.



(DEZIMALSTELLENRAD) Beim Rechnen im Calculator-Mode kann das Format im Display und im Ausdruck damit beliebig eingestellt werden.

1.1.4 Das Display

Das Display ermöglicht die Anzeige von 32 beliebigen ISO-Zeichen laut Tabelle im Anhang D.

Im Display werden angezeigt:

- über die Tastatur eingegebene Zeilen
- Mitteilungen vom Programm
- Mitteilungen vom System
- Fehlermeldungen
- Aufforderungen, Daten einzugeben (durch die Anweisungen INPUT, MAT INPUT, RKB)
- über die Tastatur eingegebene Daten
- Zeichen, die über die Funktionstasten eingegeben wurden

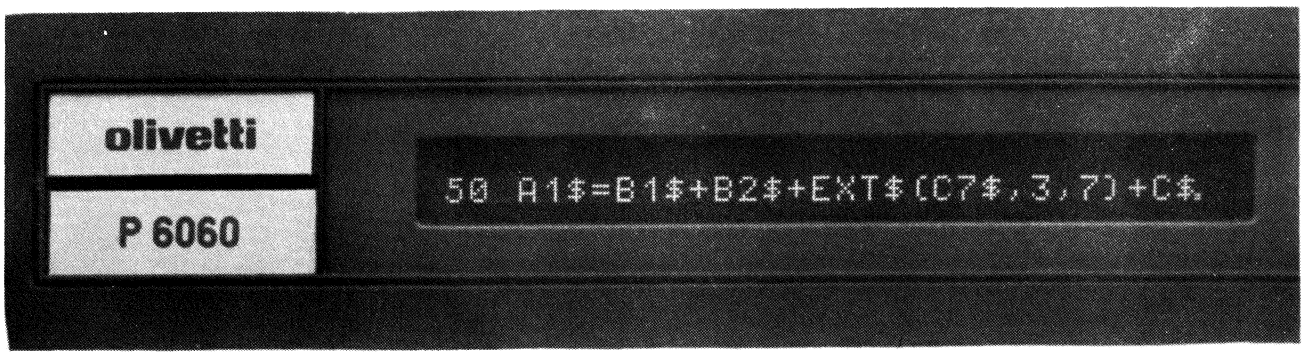


Abb. 1.10 - Das Display

Wird eine Zeile über die Tastatur eingegeben, so erscheint im Display ein Leuchtpunkt, der Pointer genannt wird. Er gibt an, in welche Position der Zeile das nächste Zeichen gesetzt wird. Vor der Eingabe einer Zeile zeigt der Pointer auf die erste Position links und wandert bei jeder weiteren Eingabe eines Zeichens jeweils um eine Stelle nach rechts. Werden mehr als 31 Zeichen eingegeben, so sind nur die letzten 31 im Display sichtbar. Während des Programmlaufs sind dagegen 32 Stellen im Display sichtbar.

1.1.5 Die Floppy-Disk-Einheit

Die Floppy-Disk - Einheit ist mit beweglichen Schreib-/Leseköpfen ausgestattet. Als Datenträger werden austauschbare Platten (Disketten) verwendet. (Siehe Abbildung 1.11)
Die Einheit enthält 1 oder 2 Laufwerke, in die jeweils 1 Diskette eingelegt werden kann.

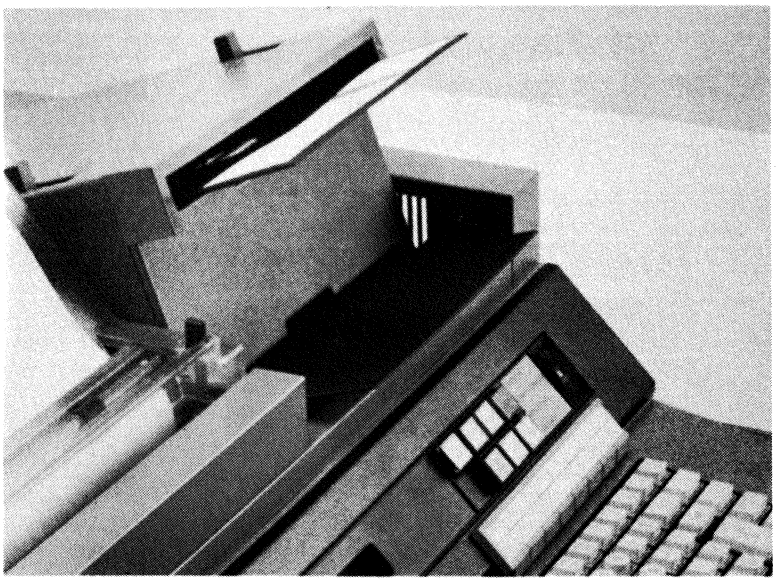


Abb. 1.11 - Die Floppy-Disk-Einheit

Die Diskette besteht aus Mylar und ist mit einem Plastiküberzug versehen. Sie wird von einer Papierhülle umgeben und kann darin frei rotieren.

Die Floppy-Disk - Einheit ist ein Speicher mit Direktzugriff. Bei einer bestimmten Umdrehungsgeschwindigkeit ist ein Lesen oder Schreiben von Informationen durch einen magnetischen Schreib-/Lesekopf möglich.

Die Informationen werden auf konzentrischen Kreisen abgespeichert, die man als Spuren bezeichnet.

Jede Diskette enthält 77 Spuren:

- 73 Spuren für Programme und Daten
- 4 Spuren werden vom System benutzt.

Jede Spur ist in 26 Sektoren zu je 128 byte unterteilt. Die Kapazität einer Platte beträgt somit 242.944 byte. (256.256 byte, wenn man auch den vom System benutzten Teil dazuzählt)
Die System-Diskette enthält die Mikroprogramme der Firmware, Basissoftwareprogramme und die Programme der Anwender-Software der Olivetti-Programmbibliothek. Im freien Teil kann der Anwender Datenfiles (sequentiell oder mit Direktzugriff) und Text- oder Programmfiles abspeichern, wie im folgenden noch gezeigt wird.

1.1.6 Der integrierte alphanumerische Drucker

Der integrierte Drucker arbeitet auf thermographischer Basis und liefert Druckbuchstaben (der vollständige Zeichenvorrat ist in Anhang D angeführt), die mit einer 5x7-Matrix dargestellt werden. In jeder Zeile können maximal 80 Zeichen mit einer Geschwindigkeit von 80 Zeichen/sec. gedruckt werden.

Dieser Drucker ermöglicht mit den entsprechenden Programmbefehlen

- das Plotten von Funktionen
- das Darstellen von Bildern im Punktraster

Abb. 1.12 - System P 6060 mit dem integrierten Drucker

1.2 ERWEITERTE KONFIGURATION

Ausgehend von der Standardkonfiguration, wie sie im vorigen Abschnitt beschrieben wurde, gelangt man durch modulares Ergänzen zu erweiterten Konfigurationen.

Die Erweiterungen betreffen:

GRUNDEINHEIT:

- Erweiterungen des Hauptspeichers von einem Minimum von 8 K byte (1 K = 1024 byte) bis maximal 48 K byte und zwar mit folgenden Ausbaustufen: 8, 16, 24, 32, 40, 48 K byte.

- Ein oder zwei IPSO-Interfaces (Interfaccia Periferiche Standard Olivetti) zum Anschluß von kompatiblen peripheren Einheiten: an jedes Interface können gleichzeitig bis zu 4 periphere Ein-/Ausgabe-Geräte angeschlossen werden.

- Interface DCC 6609 zum Anschluß eines externen Massenspeichers mit beweglichem Schreib-/Lesekopf vom Typ DCU 7292.

- Interface C CITT V 24 (EIA RS 232 C) zum Anschluß an Datenfernübertragungseinrichtungen (Time-Sharing) und/oder kompatibler Peripherie.

EXTERNE EINHEITEN

- Periphere Olivetti-Einheiten, die mit dem IPSO Interface kompatibel sind:
  - Schnelldrucker
  - Lochstreifenleser
  - Lochstreifenstanzer
  - Lochkartenleser
  - Magnetbandkassetten-Einheit (ECMA-NORM)
  - Adapter für Messinstrumente
  - Plotter
  - Magnetbandeinheiten

- DCU: Platte mit beweglichen Köpfen (siehe Abb. 1.13)
  Die Einheit hat bewegliche Schreib/Leseköpfe und besteht aus 2 Platten, einer Fest- und einer Wechselplatte.

  Jede Platte ist auf beiden Seiten speicherfähig, wobei jede Seite 2,45 Megabyte (Mega = eine Million) aufnehmen kann (Gesamtkapazität beider Platten: 9,8 Megabyte).

  Der Zugriff zu Daten kann sowohl sequentiell als auch direkt erfolgen; die mittlere Zugriffszeit beträgt 50,5 m sec, wobei die Wartezeit und die Übertragungszeit von 300 K byte/sec. schon inbegriffen sind.

- EXTERNE Verarbeitung in Time-sharing über die asynchrone Steuereinheit CCITT V 24 (EIA RS 232)

- SERIELLE PERIPHERIE beliebiger Hersteller mit der Schnittstelle CCITT V 24 (EIA RS 232 C).
  (z.B. 8 bit-Fernschreiber, Lochstreifenstanzer, Lochstreifenleser, Magnetbandkassetten, digitale Messgeräte, Video-Display, Plotter usw.)
  In diese Kategorie fallen auch periphere Geräte, die von fremden Firmen angeboten werden.

Abb. 1.13 - DCU-

2. BEDIENUNG DES P6060

2. BEDIENUNG DES P 6060

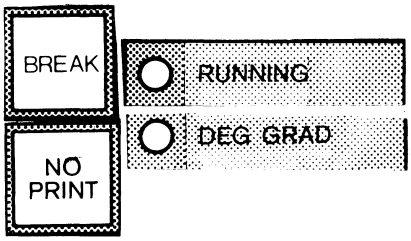
In diesem Kapitel werden die Bedienungsgrundlagen erläutert, die für das Arbeiten mit den integrierten Einheiten des Systems P 6060 notwendig sind.

2.1 EIN-/AUSSCHALTEN DES SYSTEMS

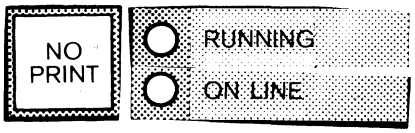
2.1.1 Einschalten

- Hauptschalter (OFF/ON) in Stellung ON bringen (siehe Abb. 1.1)
- Unmittelbar nach dem Einschalten des Systems leuchten alle Lampen an der Konsole auf: es wird die Funktionsfähigkeit der Hardware überprüft. Arbeiten alle Teile korrekt, leuchtet nur noch die Taste NO PRINT und es ertönt ein Signal.
- Nach einigen Sekunden können die Lampen der Konsole in verschiedenen Kombinationen aufleuchten. Dies hat folgende Bedeutung:

Aufleuchtende KONSOL-LAMPEN	BEDEUTUNG
BREAK, RUNNING, NO PRINT, ON LINE	Vor dem Schließen der Einschübe wurde keine Diskette eingelegt.
TRACE, RUNNING, NO PRINT, ON LINE	Statt der System-Disk ist eine User-Disk eingelegt worden.
NO PRINT, RUNNING, DEG GRAD	Die System-Disk hat einen Fehler.



Fehlerhafte Diskette



Einschub einer Floppy-Disk-Einheit ist offen.

Anmerkung:
Die hier angegebenen Kombinationen aufleuchtender Lampen betreffen nur jene Anzeigen, die vom Anwender unmittelbar behoben werden können. Darüberhinaus können andere Kombinationen aufleuchtender Lampen auftreten, die auf Fehler in der Hardware hinweisen.

Treten nach dem Einschalten der Maschine solche Kombinationen auf, die sich nicht durch die Korrektur von Bedienungsfehlern oder einen Wechsel der Disketten beheben lassen, so empfiehlt es sich, unter Angabe der auf der Konsole aufleuchtenden Lampen, den technischen Kundendienst zu verständigen.

2.1.2 Ausschalten

- Läuft ein Programm, das Files auf Disketten verarbeitet, so muß vor dem Ausschalten des Systems die Konsoltaste BREAK gedrückt werden.

- OFF/ON-Schalter in Stellung OFF bringen.

Anmerkung:
Nach dem Ausschalten des Systems muß ca. 10 Sekunden gewartet werden, bis das System erneut eingeschaltet werden kann.

2.2 ARBEITSBEGINN

Es kann vorkommen, daß der Benutzer vor dem Arbeitsbeginn noch einige Vorbereitungen treffen muß, z.B. im integrierten Drucker Papier einlegen, oder die Floppy-Disks in die entsprechenden Einheiten einlegen.

2.2.1 Das Auswechseln der Papier-Rolle

Um die Papierrolle aus dem Drucker zu entfernen und eine neue einzulegen, sind die in Abbildung 2.1 und 2.2 dargestellten Schritte auszuführen:

- den Plastik-Deckel heben
- den Druckkopf vom Papier entfernen, indem man den entsprechenden Heben leicht nach vorn zieht.
- die Papierrolle aus der Halterung nehmen und den Metallzylinder aus der Rolle ziehen
- den Metallzylinder in die neue Rolle einschieben und die Rolle in die entsprechende Halterung setzen
- das Papier, von Hand, unter der Druckrolle hindurchziehen und dann mit dem Vorschubhebel (Abbildung 2.1) das Papier hochziehen
- wenn das Papier ein paar Zentimeter hochsteht, senkt man den Papierhalter
- den Druckkopf zurückstellen, indem man den entsprechenden Hebel wieder nach hinten drückt
- die gewünschte Druckstärke kann mit einem Einstellrad gewählt werden; sie variiert zwischen 0 und 9: 0 erzeugt Fettdruck und 9 liefert die hellsten Zeichen
- den Plastikdeckel schließen, wobei man das Papier durch den Spalt hindurchzieht; man betätigt nochmals den Vorschubhebel um zu kontrollieren, ob das Papier störungsfrei geführt wird.

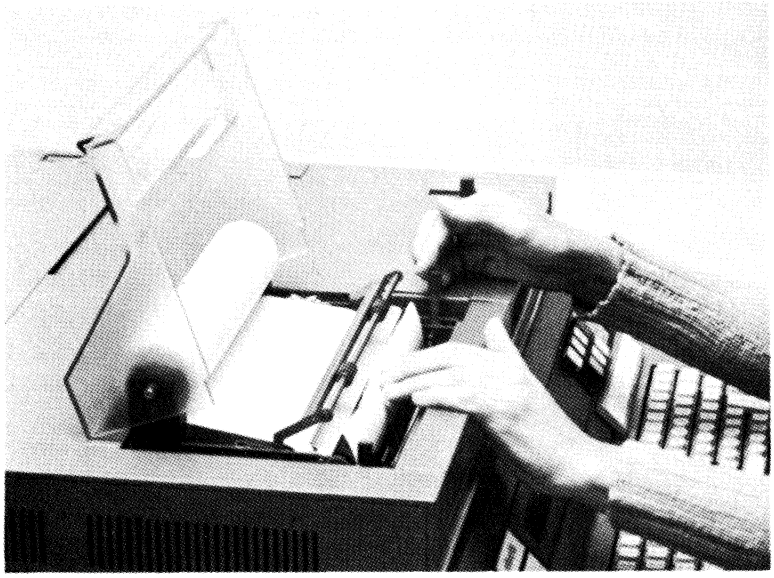


Abb. 2. 1 - Auswechseln der Papierrolle

2. 2. 2 Einlegen der Disketten

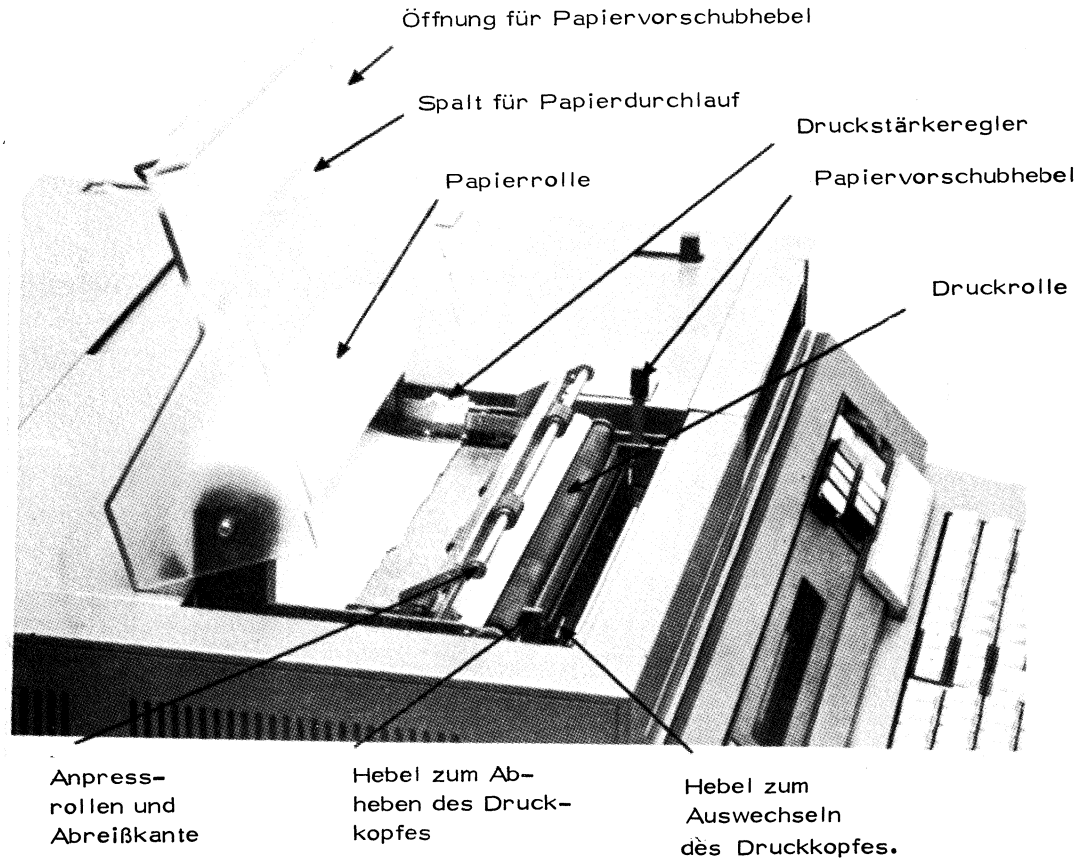
. Das System einschalten.

. Mit dem entsprechenden Hebel die Platteneinheit entriegeln. Die ganze Einheit hochklappen und mit den entsprechenden Hebeln die Einschübe öffnen. Muß nur in den oberen Einschub eine Diskette eingelegt werden, so erübrigt sich das Hochklappen der Einheit.

. In den oberen Einschub ist die Diskette mit dem Etikett nach oben, in den unteren mit dem Etikett nach unten einzulegen, bis sie hörbar einrastet.
Siehe Abbildung 2. 3.

. Die Einschübe schließen, indem man die entsprechenden Hebel nach vorne zieht.

Abbildung 2.2: Komponenten des Thermodruckers:



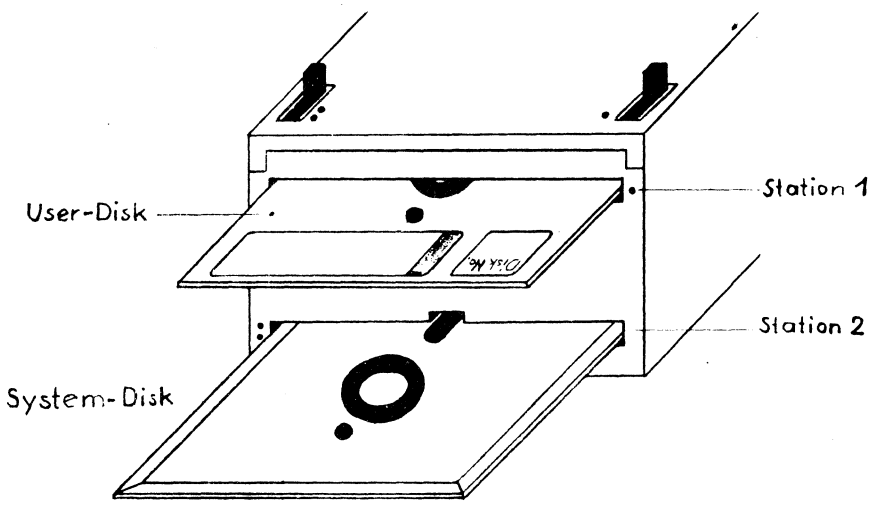


Abb.: 1.3 Einlegen der Disketten

ACHTUNG: Vor dem Schließen muß man sich vergewissern, ob das System eingeschaltet ist; ist es ausgeschaltet, kann die Diskette zerstört werden.

. Die ganze Einheit wieder senken und mit dem entsprechenden Hebel absichern. Das ist zwar nicht unbedingt erforderlich, es gewährleistet aber eine bessere Einhaltung der Betriebstemperatur.

2.2.3 Entnehmen der Disketten

. Öffnen der Verschlußklappen der Diskettenstationen
. Disketten entnehmen und in die Schutzhülle legen

Wichtig !

. Disketten sind staub- und druckempfindlich !

Bitte beachten Sie:

. Niemals Disketten ohne Schutzhülle aufbewahren
. Disketten senkrecht in der mitgelieferten Schachtel aufstellen
. Disketten-Etiketten vor dem Aufkleben beschreiben
. Die sichtbaren Teile der Magnetplatte nicht berühren

2.3 LADEN DES BETRIEBSSYSTEMS

Nachdem man die Disketten eingelegt hat, beginnt das System mit einer Initialisierungsphase.
Während der Initialisierung wird der Teil des Betriebssystems, der die Interpretation der Tastatureingaben durchführt, von der System-Diskette in den Hauptspeicher geladen. Während dieser Phase blinkt die Kontroll-Lampe RUNNING.

Nach der Initialisierungsphase erscheint auf dem Display die Meldung READY. Die Lampe RUNNING hört auf zu blinken und leuchtet konstant. Das System ist nun bereit, Befehle und BASIC-Anweisungen aufzunehmen.

(Erscheint anstelle von READY eine Fehlermeldung, so ist der Fehler nach den Anweisungen der Fehlerliste zu beheben).

2.4 DATIERUNG

Bei der Erstellung von Programmen bzw. Files ist es nach der Initialisierung sinnvoll, mit der Anweisung DATE das Datum einzugeben. Das allgemeine Format ist :

DATE Datum

"Datum" besteht aus 6 Zeichen, mit denen in beliebigem Format das Datum eingegeben werden kann, jedoch darf das Leerzeichen nicht verwendet werden.

Beispiel : 061176
OKT.76 falsch : AUG 76

Beim Ausdrucken des Kataloges wird das eingegebene Datum in 3 Gruppen zu je 2 Zeichen, getrennt durch "-", ausgegeben.

Beispiel : 06-11-76
OK-.T-76

Jedes File, das auf Disketten gespeichert wird, wird dann mit dem jeweiligen Datum versehen. Damit ist bei jedem File ersichtlich, wann es erstellt oder modifiziert wurde.

Das eingegebene Datum wird zudem auf der System-Diskette festgehalten und wird erst durch eine neue DATE-Anweisung verändert.

Durch Ändern von Programm- oder Textfiles (REPLACE) oder Beschreiben von Datenfiles wird im Katalog in der Spalte LAST MOD (letzte Modifikation) das aktuelle Datum ausgegeben. In der Spalte CREAT bleibt das Datum erhalten.

2.5 EINGABE ÜBER DIE TASTATUR

Über die Tastatur können eingegeben werden :

- Systembefehle
- BASIC-Anweisungen
- Strings
- Numerische Daten
- Textzeilen
- Berechnungen im Calculator Mode

Die über die Tastatur eingegebenen Zeichen werden in einem Register, "Tastatur-Puffer" genannt, abgespeichert. Dieser Puffer hat eine Kapazität von 80 Zeichen.

Das eingetastete Zeichen erscheint unmittelbar im Display und zwar rechts von der letzten Stellung des Pointers.

Die numerische Eingabe kann in Fest- oder Gleitkomma (exponentielle Darstellung) erfolgen. Maximal werden 13 signifikante Stellen und bei Gleitkommadarstellung 2 Stellen für den Exponenten sowie die negativen Vorzeichen akzeptiert.

Zum Beispiel : .9999999999999E99 / -127.5698 / -31.55E-12

In gewissen Programmen wird aus Kapazitätsgründen für numerische Werte nur mit halber Kapazität gearbeitet. Entsprechende Hinweise finden sich in den zugehörigen Bedienungsanleitungen (siehe auch Kapitel 10).

Das Display kann maximal 32 Zeichen und den Pointer anzeigen.

Drückt man die Tasten EOL oder SUM , so werden alle Zeichen außer dem Pointer vom Tastatur-Puffer in den Hauptspeicher übertragen. Die Zeichen im Display werden gelöscht und der Pointer weist auf die erste Stelle im Display. Die Eingabe über die Tastatur ist auch möglich, während das System arbeitet, während des Druckens und der I/O-Operationen auf Floppy-Disk; die Befehle EOL oder SUM werden jedoch ignoriert und es ertönt ein Signal; END OF LINE und SUM werden erst dann angenommen, wenn die Lampe RUNNING nicht mehr blinkt.

Anmerkung : Gibt man mehr als 80 Zeichen über die Tastatur ein, wird das letzte Zeichen nicht akzeptiert; dies wird durch ein akustisches Signal angegeben und gleichzeitig leuchtet die Konsol-Lampe LINE OVERFLOW auf.

Werden zwei Zeichen gleichzeitig eingegeben, so werden beide ignoriert und es ertönt ein Signal.

Mitteilungen vom Programm oder vom System werden vom Hauptspeicher in den "Display-Puffer" übertragen. Da bei diesen Mitteilungen der Pointer nicht aufleuchtet, können bis zu 32 Zeichen im Display stehen.

Das System ist mit dem Display-Puffer automatisch verbunden, sobald eine Anweisung DISP ausgeführt wird oder das Programm auf Dateneingabe über die Tastatur wartet.

Es erscheint bei den Anweisungen INPUT, MAT INPUT oder RKB als Anfrage ? oder ??.

Sobald ein Zeichen eingetastet wird, erlischt diese Mitteilung am Display und es wird der Inhalt des Tastatur-Puffers angezeigt. Wenn das Display den Inhalt des Tastatur-Puffers anzeigt, ist es möglich, den Inhalt des Display-Puffers wieder anzuzeigen, indem man die Taste CLEAR RECALL drückt, umgekehrt wird durch dieselbe Taste der Inhalt des Tastatur-Puffers ins Display gebracht, in dem sich zuvor der Inhalt des Display-Puffers befand.

Das Display kann sowohl die über die Tastatur eingegebenen Zeichen, als auch Mitteilungen von Programmen und vom System anzeigen.

2.5.1 Korrektur von Eingaben über die Tastatur

Bevor man EOL oder SUM drückt, ist es möglich, Zeichen im Tastatur-Puffer :

- zu löschen
- zu modifizieren
- einzufügen

- Löschen

. Löschen eines Zeichens :

Man setzt den Pointer an die Stelle unmittelbar rechts vom Zeichen, das gelöscht werden soll. Dazu verwendet man die Tasten [←] , [→] , SHIFT , REPEAT . Man drückt CHAR DELETE.

Das Zeichen wird gelöscht und der Pointer und alle Zeichen rechts davon werden um eine Stelle weiter nach links verschoben.

. Löschen aller Zeichen :

Man drückt gleichzeitig die Tasten SHIFT und CLEAR / RECALL unabhängig von der Position des Pointers. Es werden im Puffer alle Zeichen gelöscht und der Pointer weist auf die erste Stelle im Display.

- Modifizieren

. Modifizieren eines Zeichens :

Man löscht das zu modifizierende Zeichen (siehe Löschen eines Zeichens) und gibt das neue Zeichen ein. Das eingetastete Zeichen ersetzt das gelöschte Zeichen im Puffer; der Pointer zeigt auf die erste Stelle rechts nach dem neuen Zeichen.

- Einfügen

. Einfügen eines Zeichens :

Man setzt den Pointer an jene Stelle im Display, an der neue Zeichen eingefügt werden sollen. Dazu benutzt man die Tasten

[←] , [→] , SHIFT , REPEAT .

Man tastet die neuen Zeichen ein. Im Display werden diese Zeichen an der Stelle angezeigt, an der sie eingefügt wurden und der Pointer zeigt auf die erste Stelle rechts vom letzten neuen Zeichen.

3. ARBEITEN MIT ANWENDERPROGRAMMEN

3. DAS ARBEITEN MIT ANWENDERPROGRAMMEN

Dieses Kapitel soll vorwiegend dem Anwender dienen, der nur mit bestehenden Programmen arbeitet und die Programmierung nicht kennt. Es vermittelt ihm das Vorgehen beim Starten der Programme, dem Vorbereiten und Sichern der Disketten und erlaubt ihm, den P 6060 als einfachen Tischrechner zu verwenden.

3.1. STARTEN EINES PROGRAMMES

- Inbetriebsetzung der Maschine und Einlegen der Disketten laut Beschreibung in Kapitel 2

- Falls in der Programmbeschreibung nicht anders angegeben, Programmstart mit der Systemanweisung RUN nach folgendem Format:

RUN Programmname EOL

Der Programmname muß der entsprechenden Programmbeschreibung entnommen werden.

Beispiel:

RUN STRESS EOL
RUN *PROG EOL

Erscheint im Display die Fehlermeldung ERROR 187 (Programm bzw. File nicht gefunden), ist zu prüfen:

Wurde der Programmname richtig eingetippt ?

Zur Kontrolle den eingegebenen Namen mit RECALL ins Display zurückrufen.

Ist der verwendete Name richtig, ist zu prüfen : wurden die richtigen Disketten eingelegt.

3.2 INITIALISIEREN UND DUPLIZIEREN VON DISKETTEN

3.2.1 Initialisieren neuer Disketten

Neue Disketten, d.h. Disketten, die nicht beschrieben sind, müssen initialisiert werden. Erst dann können Programm- oder Text-Files gespeichert sowie Daten-Files kreiert werden.

Zur Initialisierung ist folgende Prozedur anzuwenden :

- Maschine in Betrieb nehmen gemäß Kapitel 2 und zu initialisierende Diskette in Laufwerk 1 legen.
- Systemanweisung eingeben :
  EXEC LBC, U EOL
  (näheres siehe unter LBC, Kapitel 6)

Wenn die Konsol-Lampe RUNNING nicht mehr blinkt, ist die Initialisierung beendet und im Display erscheint die Meldung READY.

Bereits beschriebene Disketten können durch EXE LBC, U wieder gelöscht werden (User- und Systemdisketten). Der Befehl EXE LBC, S löscht den Inhalt einer Systemdiskette, nicht aber das sich darauf befindende Betriebssystem. Dabei muß sich die zu löschende Systemdiskette im Laufwerk 2 befinden.

3.2.2 Duplizieren von Disketten

- Kopieren von User-Disks :

  Vor Beginn des Kopierens sind die Original-User-Diskette und eine System-Diskette eingelegt.

  . Nach Eingabe von :
  EXEC FDCOPY, U EOL
  erscheint im Display die Meldung :
  INSERT DISK > RECEIVING DISK ON DRIVE 
  (der zweite Teil der Meldung erscheint nach Drücken der Tasten SHIFT und )

  . Empfängerdiskette in das genannte Laufwerk einlegen - Konsoltaste CONTINUE drücken.

. Nach dem Kopieren erscheint im Display :
END - ILLEGAL STATUS REARRANGE DISKS

Erscheint anstelle dieser Meldung :
ERROR n - ILLEGAL STATUS REARRANGE DISKS

so ist nach den Anweisungen der Fehlerliste vorzugehen. Die Kopie ist nicht vollständig.

Die Kopie ist dem Laufwerk zu entnehmen und durch die System-Disk zu ersetzen. Nach Drücken der Taste CONTINUE ist das System wieder arbeitsfähig.

- Kopieren von System-Disks :

Vor Beginn des Kopierens sind eine leere Diskette und eine System-Diskette eingelegt.

. Nach Eingabe von :
EXEC FDCOPY, S EOL

beginnt der Kopiervorgang.

. Nach dem Kopieren erscheint im Display :
END - ILLEGAL STATUS REARRANGE DISKS

Erscheint anstelle dieser Meldung :
ERROR n - ILLEGAL STATUS REARRANGE DISKS

so ist nach den Anweisungen der Fehlerliste vorzugehen. Die Kopie ist nicht vollständig.

Die Kopie ist dem Laufwerk zu entnehmen und durch eine User-Disk zu ersetzen. Nach Drücken der Taste CONTINUE ist das System wieder arbeitsfähig.(Näheres siehe unter FDCOPY, Kapitel 6).

3.3 MANUELLES RECHNEN (Calculator Mode)

Im folgenden sind nur die wichtigsten Rechenbefehle beschrieben. Alle Möglichkeiten des Calculator Mode enthält das Kapitel 11.

3.3.1 Umschalten in Calculator Mode :

Drücken der Konsoltaste CALC MODE, die nun aufleuchtet.

Die Rückkehr aus dem Calculator Mode erfolgt wieder durch Drücken derselben Taste, oder durch Eingabe eines Systembefehles wie NEW oder RUN.

3.3.2 Mathematische Operationen :

+	Addition	*	Multiplikation	↑	Potenzieren
-	Subtraktion	/	Division		

3.3.3 Rechenvorgang :

Es können beliebige mathematische Formeln unter Verwendung dieser Operatoren, sowie der runden Klammern zur Prioritätensetzung, eingegeben werden (max. 80 Zeichen). Es können die Tasten der speziellen Rechnertastatur oder der alphanumerischen Tastatur verwendet werden. Das Resultat wird nach EOL gebildet und erscheint im Display bzw. wird gedruckt, sofern die PRINT ALL-Taste aktiviert wurde. Das Format des Resultats kann mit dem Dezimalstellenrad gesteuert werden.

Das Ergebnis der jeweils letzten Rechenoperation wird gleichzeitig im Register RESULT gespeichert und kann durch Drücken der entsprechenden Taste RESULT jederzeit wieder in eine neue Berechnung aufgenommen werden. Dieser Aufruf wird mit dem Symbol Φ dargestellt.

Beispiele :

a) 2 * (1.5 + 2 5) EOL (Dez. Rad auf 3)
im Display erscheint : 67.ØØØ

b) Φ * 1.Ø352 E-3 + 1ØØ EOL (Φ = vorhergehendes Resultat =67)
Resultatanzeige : 1ØØ.Ø69

- Meldungen :

  Wenn die Empfängerdiskette schon als System- bzw. User-Disk initialisiert wurde, gibt das System vor dem Kopiervorgang eine Warnmeldung aus :

System-Disk	User-Disk
FILE "P6FWRZ" VALID	FILE "P6FSYS" VALID
FILE "P6SW " VALID	
FILE "P6FWO " VALID	
FILE "P6FSYS" VALID	

  Im Display erscheint die Frage "CONTINUE ?". Wird die Konsoltaste CONTINUE gedrückt, so wird der Kopiervorgang gestartet. Soll nicht kopiert werden, ist die Taste BREAK zu drücken. Es erscheint die Meldung :

  BREAK OCCURED - CHECK DISK STATUS.

  Nach Herstellen einer gültigen Diskettenkonfiguration (System-Disk und gegebenenfalls User-Disk) ist die Taste CONTINUE zu drücken.

3.3 MANUELLES RECHNEN (Calculator Mode)

Im folgenden sind nur die wichtigsten Rechenbefehle beschrieben. Alle Möglichkeiten des Calculator Mode enthält das Kapitel 11.

3.3.1 Umschalten im Calculator Mode :

Drücken der Konsoltaste CALC MODE, die nun aufleuchtet.

Die Rückkehr aus dem Calculator Mode erfolgt wieder durch Drücken derselben Taste, oder durch Eingabe eines Systembefehls wie NEW oder RUN.

3.3.2 Mathematische Operationen :

+	Addition	*	Multiplikation	↑	Potenzieren
-	Subtraktion	/	Division		

3.3.3 Rechenvorgang :

Es können beliebige mathematische Formeln unter Verwendung dieser Operatoren, sowie der runden Klammern zur Prioritätensetzung, eingegeben werden (max. 80 Zeichen). Es können die Tasten der speziellen Rechnertastatur oder der alphanumerischen Tastatur verwendet werden. Das Resultat wird nach EOL gebildet und erscheint im Display bzw. wird gedruckt, sofern die PRINT ALL-Taste aktiviert wurde. Das Format des Resultats kann mit dem Dezimalstellenrad gesteuert werden.

Das Ergebnis der jeweils letzten Rechenoperation wird gleichzeitig im Register RESULT gespeichert und kann durch Drücken der entsprechenden Taste RESULT jederzeit wieder in eine neue Berechnung aufgenommen werden. Dieser Aufruf wird mit dem Symbol ⧖ dargestellt.

Beispiele :

a) 2 ✱ (1.5 + 2 ↑ 5) EOL (Dez. Rad auf 3)
im Display erscheint : 67.ØØØ

b) ⧖ ✱ 1.Ø352 E-3 + 1ØØ EOL (⧖ = vorhergehendes Resultat = 67)
Resultatanzeige : 1ØØ.Ø69

4. BETRIEBSARTEN DES SYSTEMS

4. BETRIEBSARTEN DES SYSTEMS

Das System P6ø6ø kennt folgende Betriebsarten :

- COMMAND MODE
- RUNNING MODE
- DEBUGGING MODE
- CALCULATOR MODE

Der Zusammenhang der einzelnen Betriebsarten wird am Ende dieses Kapitels in einer Graphik verdeutlicht.

4.1 COMMAND MODE

Der Command-Mode erlaubt :

- Die Eingabe und Ausführung von Systembefehlen (Kapitel 6),
- den Aufruf von Dienstprogrammen (Abschnitt 6.4),
- die Eingabe von Programmen und Texten, sowie Editing-Operationen (Kapitel 9),
- den Übergang in den Calculator-Mode (Kapitel 11).

Das System ist im Command-Mode :

- nach dem Laden des Systems,
- nach der Ausführung eines Anwenderprogrammes,
- nach der Eingabe von Systembefehlen im Calculator-Mode, oder nach seiner expliziten Aufhebung durch Drücken der Konsoltaste,
- nach dem Drücken der Konsoltaste BREAK im Running-Mode oder Debugging-Mode.

Im Command-Mode leuchtet die Konsollampe RUNNING konstant. Nach dem Laden des Systems bzw. nach einer Rückkehr in den Command-Mode aus dem Running-Mode oder Debugging-Mode erscheint im Display zusätzlich die Meldung "READY", falls im Befehl SAVE des laufenden Programmes kein MSG = ø Parameter angegeben wurde.

4.2. RUNNING MODE

Werden vom System Dienstprogramme oder Anwenderprogramme ausgeführt, so befindet sich das System in Running Mode. Bei der Ausführung von Dienstprogrammen richten sich die Eingriffsmöglichkeiten nach der Art und den Erfordernissen des jeweils aufgerufenen Dienstprogrammes.
Die folgende Beschreibung des Running-Modes befaßt sich nur mit der Ausführung von Anwenderprogrammen.

Der Aufruf der Ausführung eines Programmes erfolgt mit einem der Befehle:

RUN

PREPARE

Wird eine Programmausführung mit PREPARE gestartet, so wird der Running-Mode nur über den Debugging-Mode erreicht.
Wird die Ausführung des Programmes mit RUN gestartet, so werden sämtliche Funktionen von PREPARE mitausgeführt, es wird jedoch (ohne Übergang in den Debugging-Mode) unmittelbar mit der Ausführung des Programmes begonnen.
Die genauen Funktionen der Befehle PREPARE und RUN werden in Kapitel 6 beschrieben.

Möglichkeiten des Verlassens der Programmausführung:

4.2.1. Rückkehr in den Command-Mode

- Wird die Ausführung eines Programmes durch Erreichen der END-Anweisung regulär beendet, so erfolgt die Rückkehr in den Command-Mode.

- Nach dem Drücken der Konsoltaste BREAK wird die Abarbeitung des laufenden Programmes abgebrochen, wobei das laufende Statement regulär beendet wird, eventuell geöffnete Files geschlossen werden und das Programm wieder für Editing-Operationen zur Verfügung gestellt wird.

4.2.2 Übergang in den Debugging-Mode

Der Übergang aus dem Running-Mode in den Debugging-Mode erfolgt durch:

- Drücken der Taste STEP
- Durch einen STOP - Befehl im Programm
- Durch die Ausführung eines STOP - Befehls, der im Debugging-Mode eingegeben wurde
- Durch einen behebbaren Fehler
- Nach nicht behebbaren Fehlern ohne Möglichkeit der Programmfortsetzung

Wird während der Ausführung eines Programmes die Konsoltaste STEP gedrückt, so wird am Ende der laufenden Operation die Abarbeitung des Programmes unterbrochen. Es stehen danach alle Möglichkeiten des Debugging-Modes zur Verfügung (siehe Kapitel 10). Tritt bei der Abarbeitung eines Programmes ein Fehler auf, so meldet das System den Fehlercode und bleibt im Debugging-Mode.

Mit den im Debugging-Mode möglichen Operationen können die Ergebnisse des Statements, das den Fehler verursachte, erzeugt werden und die Abarbeitung mit dem nächsten oder einem beliebigen anderen Befehl fortgesetzt werden. Die Rückkehr in den Running-Mode erfolgt entweder mit der Consoltaste CONTINUE oder mit dem Befehl START. (START ist nur möglich, wenn für das mit RUN aufgerufene Programm bereits die Preexecution durchgeführt war oder das Programm mit PREPARE gestartet wurde.)
Bei Drücken der Taste STEP wird jeweils ein Statement verarbeitet und in den Debugging-Mode zurückgekehrt.

4.2.3 Inputanforderung

Tritt im Ablauf der Programmausführung eine der Anweisungen:
RKB, INPUT, MAT INPUT oder RECEIVE# auf, so wartet das System auf die Eingabe der Daten in richtiger Anzahl und Format.

- Nach dem Abschluß der Eingabe kehrt das System in den Running-Mode zurück.

- Wird die Konsoltaste BREAK gedrückt, während das System auf eine Eingabe wartet, wird der Programmlauf abgebrochen.

- Wird die Konsoltaste STEP gedrückt, geht das System in den Debugging-Mode. Nach Drücken der Taste CONTINUE erscheint im Display ein Fragezeichen; die Eingabe kann nun erfolgen.

Wartet das System auf eine Eingabe, so hört die Lampe RUNNING zu blinken auf und leuchtet konstant.

4.2.4 Operator Call

Bei der Ausführung von Dienstprogrammen werden von einzelnen Programmen bestimmte Tätigkeiten des Benutzers verlangt, wie etwa den Wechsel von Disketten durchzuführen.

Die Aufforderung zum Wechseln einer Diskette erfolgt im Display. Nach dem durchgeführten Diskettentausch wird durch Drücken der Konsoltaste CONTINUE die Abarbeitung des Dienstprogrammes fortgesetzt.

Das Drücken der Konsoltaste BREAK wird nur akzeptiert, wenn eine arbeitsfähige Konstellation vorhanden ist.

4.2.5 Behebbare Fehler (Fehlercode 1-16)

Tritt während eines Programmlaufes ein behebbarer Fehler auf, z.B. eine fehlende Wertzuweisung, geht das System unter Meldung des Fehlercodes in den DEBUGGING-MODE über. Eine Abfrage von Variablen und gegebenenfalls eine Wertzuweisung ist möglich. Der Programmlauf kann durch Drücken der Konsoltaste CONTINUE fortgesetzt werden.

4.2.6 Nicht behebbare Fehler

4.2.6.1 Nicht behebbare Fehler der Preexecution (Fehlercode 40-55)

Tritt während der Preexecution eines Programmes (nach PREPARE oder RUN) ein nicht behebbarer Fehler auf, z.B. ein unerlaubter Sprungbefehl, erfolgt eine Auflistung der Fehler und das System geht in den COMMAND-MODE über.

4.2.6.2 Nicht behebbare Fehler der Ausführungsphase (Fehlercode 65-97, 162-170)

Tritt während eines Programmlaufes ein nicht behebbarer Fehler auf, z.B. RETURN ohne GOSUB, geht das System unter Meldung des Fehlercodes in eine Quasi-DEBUGGING-MODE über, d.h. eine Abfrage von Variablen ist möglich, aber der Programmlauf kann nicht fortgesetzt werden. Es muß die Konsoltaste BREAK gedrückt werden, danach geht das System in den COMMAND-MODE über.

4.3 DEBUGGING MODE

Das System befindet sich im Debugging-Mode :

- Nach Drücken der Taste STEP
- Nach Ausführung der Anweisung PREPARE
- Bei einer STOP-Anweisung im Programmablauf
- Nach Meldung eines behebbaren Fehlers
- Nach dem Auftreten eines nicht behebbaren Fehlers.

Der Debugging-Mode erlaubt das Austesten eines Programmes und die Behebung von Fehlern.

Die genaue Arbeitsweise wird im Kapitel 10 beschrieben.

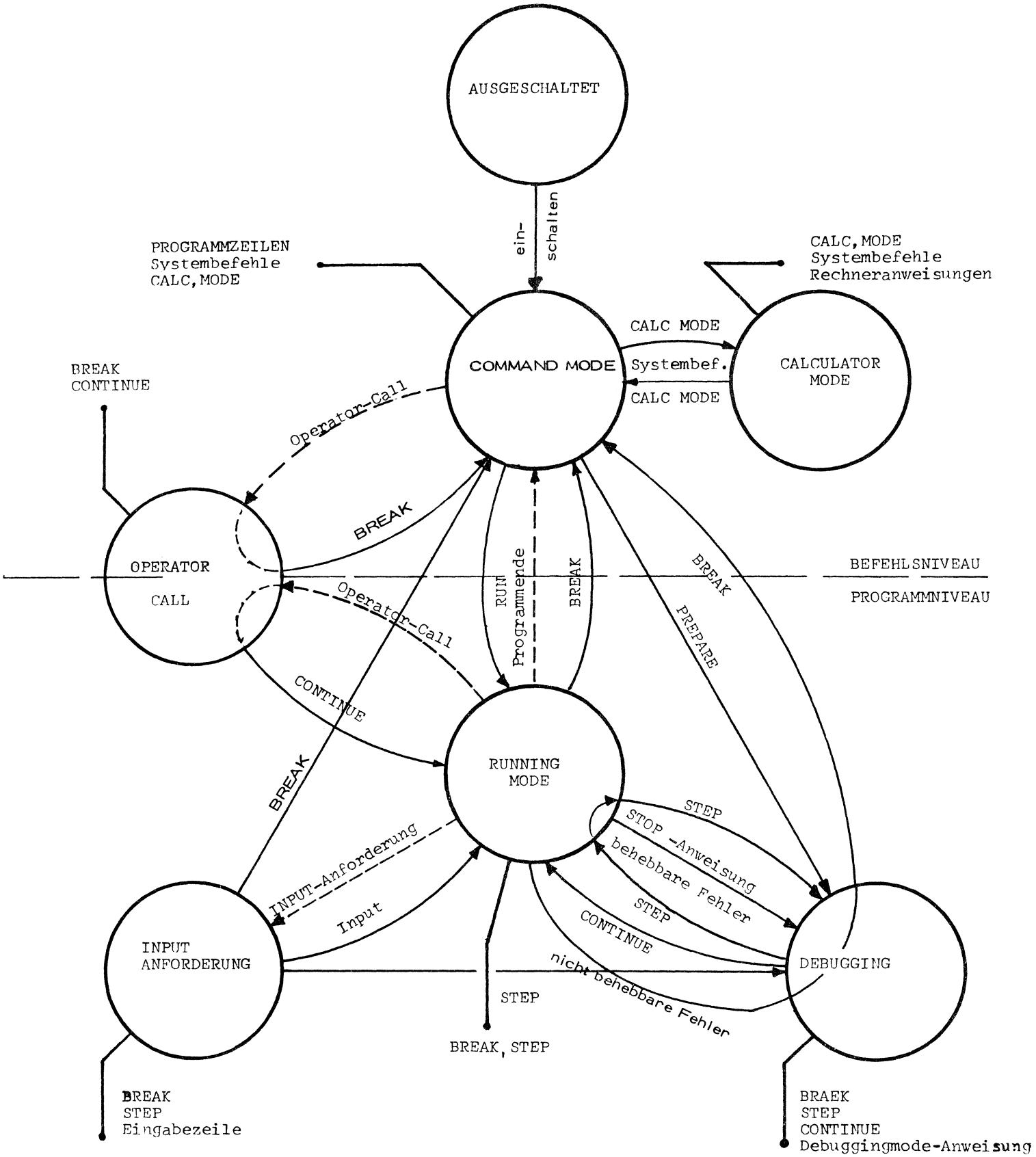
4.4 CALCULATOR MODE

Durch die Aktivierung der Calculator-Mode-Taste (leuchtet auf) ist der P6060 als Tischrechner zu verwenden.

Der Debugging-Mode enthält ebenfalls den Calculator-Mode.

Die genaue Beschreibung der Möglichkeiten im Calculator-Mode erfolgt in Kapitel 11.

4.5 VOLLSTÄNDIGES ZUSTANDSDIAGRAMM



5. ALLGEMEINE INFORMATIONEN ÜBER EXTERNE FILES

5. ALLGEMEINE INFORMATIONEN ÜBER EXTERNE FILES

5. 1. ALLGEMEINES

Unter Files versteht man eine problemorientierte organisatorische Zusammenfassung gleichartiger oder gleichbehandelter Daten in einem Speichermedium.
Im System P 6060 werden drei Filetypen unterschieden, und zwar

1) Programmfiles
2) Textfiles
3) Datenfiles

Die Files heißen "extern", da sie auf dem externen Speicher "Floppy-Disk" gespeichert sind.

5. 1. 1. Programm- und Textfiles

Programm- bzw. Textfiles erlauben es, Programme bzw. Texte unter einem Namen zu speichern.
Ein Programmfile enthält eine mit Zeilennummern versehene Folge ausführbarer BASIC - Anweisungen.
Ein Textfile besteht aus einer Folge numerierter Zeilen, die beliebige Folgen von ISO - Zeichen enthalten.

Ein Programmfile kann daher nur ein Programm, ein Textfile nur einen Text enthalten. Es haben daher die Begriffe Programm und Programmfile (bzw. Text und Textfile) die gleiche Bedeutung.
Für die Behandlung von Programm-und Textfiles ist im System P6060 eine Reihe von Systembefehlen vorgesehen. Textfiles können außerdem in einem Basic-Programm wie sequentielle Datenfiles gelesen werden.

5. 1. 2. Datenfiles

Datenfiles werden für die programmunabhängige permanente Speicherung numerischer und alphanumerischer Daten verwendet.

5. 2. FILE - ORGANISATION

Auf den Disketten werden Bibliotheken eingerichtet, die eine gemeinsame Verwaltung der in ihnen gespeicherten Files erlauben. Jedes File gehört einer Bibliothek an, wobei 3 verschiedene Arten von Bibliotheken unterschieden werden

- Package - Bibliothek
- Common - Bibliothek
- USER - Bibliothek

Der wesentliche Unterschied zwischen den einzelnen Bibliotheken besteht im Ausmaß, in dem ihre Files vor verschiedenen Operationen geschützt sind.
Die Erstellung der Bibliotheken erfolgt mit Hilfe des Dienstprogrammes LBCREATE, das im Kapitel 7 detailliert beschrieben ist.

Die Package - Bibliothek enthält entweder
1) ein von Olivetti erstelltes Programmpaket oder
2) vom Anwender erstellte Programm-, Text- und Datenfiles.

Eine Package - Bibliothek kann als Ganzes mit dem Dienstprogramm LBPROTECT gegen ihre Veränderung durch die Systembefehle

CREATE	TRUNCATE
MODIFY	REPLACE
PURGE	TRANSCODE
SAVE	FLCOPY

geschützt werden.

- COMMON-Bibliothek

Eine Common-Bibliothek kann sowohl Programme als auch Text- und Datenfiles enthalten.
Nach der Ausführung des Dienstprogrammes LBPROTECT ist eine Common-Bibliothek gegen folgende Befehle geschützt

MODIFY
PURGE

- USER - Bibliothek

Die User-Bibliothek kann Programme, Text- und Datenfiles enthalten.
Die User-Bibliothek ist nicht geschützt, man kann mit den entsprechenden Befehlen Files modifizieren, löschen und hinzufügen.

5.2.1. Disketten

Es gibt zwei Arten von Disketten: System- und Benutzer-Diskette (im folgenden als System-Disk und User-Disk bezeichnet).

- die System-Disk

Die System-Disk enthält das Betriebssystem sowie alle Dienstprogramme. Sie kann aber auch eine Package-, eine Common- oder eine USER-Bibliothek enthalten.
Enthält die Konfiguration nur ein Floppy-Disk-Laufwerk, steht nur die System-Disk für die Speicherung von Files zur Verfügung.

- die User-Disk

Die User-Disk kann eine Package-,eine Common- und eine USER-Bibliothek enthalten.

5.3. FILENAMEN

Jedes File auf einer Diskette wird mit seinem Namen identifiziert. Über den Namen kann ein File gesucht oder vor unerlaubtem Zugriff geschützt werden.
Ein Filename setzt sich zusammen aus einem Bibliothekskennzeichen (wenn das File der Package- oder Common-Bibliothek angehört) und dem eigentlichen Namen.

- Der eigentliche Name besteht aus maximal 6 alphanumerischen Zeichen, wobei das erste Zeichen ein Buchstabe sein muß.
- Für das Bibliothekskennzeichen gilt:
  " * ": Das File gehört zur Packagebibliothek
  " + ": Das File gehört zur Commonbibliothek

Files, die zum Bestandteil der USER-Bibliothek gehören, haben kein Bibliothekskennzeichen, d. h. ihre Namen bestehen nur aus dem eigentlichen Namen.

	richtige Namen	falsche Namen
Packagebibliothek	* SINES	* (kein Alphazeichen)
Commonbibliothek	+ G	+8G (das 2-te Zeichen ist kein Buchstabe)
Userbibliothek	GRAPH 2	GRAPH66 (mehr als 6 Zeichen)

5.4. ERSTELLEN VON FILES

5.4.1. Erstellen von Programmen und Textfiles

Programme und Texte werden entweder unmittelbar im Arbeitsspeicher erstellt oder ergeben sich durch die Umwandlung aus einem File eines anderen Typs.
Soll ein Programm im Arbeitsspeicher über die Tastatur erstellt werden, so ist vor der Erstellung der Befehl [1)]

NEW

einzugeben.
Soll ein Textfile erstellt werden, so lautet der entsprechende Befehl:

TEXT

Wird nach der Erstellung eines Programmes oder Textes der Befehl

SAVE

gegeben, so wird das Programm (Text) im Arbeitsspeicher unter diesem Namen in der durch den Namen angegebenen Bibliothek auf einer Diskette gespeichert.

Beispiel:

```
TEXT
10 DAS IST EIN TEXT
20 DER IN DER COMMON BIBLIOTHEK
30 DER USER-DISK GESPEICHERT
40 WERDEN SOLL

SAVE U, + TEXT1
```

Nach der Ausführung des Befehles
SAVE U, + TEXT1
ist der Text in der COMMON-Bibliothek der USER-DISK enthalten und kann unter diesem Namen wieder angesprochen werden.
(z.B. mit OLD +TEXT 1).

---

1) Alle in der Folge erwähnten Systembefehle sind in Kapitel 6 beschrieben.

5.4.2 Erstellen von Datenfiles

Für die vollständige Vereinbarung eines Datenfiles sind folgende Angaben erforderlich:

1) Die Angabe, ob es auf der System-Disk oder User-Disk erstellt werden soll
2) Ein Name (der auch die Bibliothek festlegt)
3) Die Zugriffsart
4) Der maximale Speicherbedarf .

Ein Datenfile wird mit dem Befehl CREATE erstellt.

5.4.2.1 Die Zugriffsarten

Ein Datenfile kann sequentiellen oder random (direkten) Zugriff erlauben.

- Sequentielle Datenfiles

Die Daten sind im File in unmittelbarer Aufeinanderfolge gespeichert. Das Schreiben von Daten in das File erzeugt eine physische Reihenfolge, die nicht unterbrochen werden kann, und die auch die Reihenfolge festlegt, in der die Daten aus dem File gelesen werden.
Sequentielle Datenfiles können nur auf zwei Arten beschrieben werden:

- Von Beginn an.
  Der vorherige Inhalt wird dabei vollständig zerstört.
- Durch Anhängen der Daten an den alten Inhalt, der dabei erhalten bleibt.
  Es ist dabei nicht möglich, ein Datenelement direkt anzusprechen und gegen ein anderes auszutauschen.

- Random Files

Wird für ein Datenfile die Zugriffsart Random festgelegt, so kann auf jedes Datenelement im File direkt zugegriffen werden, das heißt, es können Daten an jede beliebige Stelle im File geschrieben werden bzw. kann beim Lesen auf jedes Datum im File Bezug genommen werden.
Die Festlegung der Position des Datums erfolgt unabhängig von der Schreib- oder Leseanweisung in einer eigenen Anweisung.

5.4.2.2 Platzbedarf

Die Daten eines Files werden in binärem Format gespeichert. Die kleinste Einheit, auf die in einem Datenfile zugegriffen werden kann, ist ein Wort, das aus 4 Bytes besteht.
Ein numerisches Datum belegt in einfacher Genauigkeit in einem Datenfile ein Wort, in doppelter Genauigkeit belegt es zwei Worte.
Für alphanumerische Daten (Strings) errechnet sich der Platzbedarf wie folgt:
Die Anzahl der Worte ist gleich INT ((n-1)/4 + 2), wenn n die Anzahl der Zeichen im String angibt.

Um im Befehl CREATE die Anzahl der Bytes festzulegen, die für ein Datenfile reserviert werden sollen, ist es daher notwendig, aus der Art der Daten und der maximalen Anzahl von Daten, die dieses File enthalten soll, zunächst die Anzahl der Worte und daraus die erforderliche Byteanzahl zu errechnen.

Anmerkungen:
- Das Wort ist die kleinste Einheit, auf die in einem File mit random Zugriff zugegriffen werden kann.
- Ist die Anzahl der im CREATE-Befehl angegebenen Bytes kein Vielfaches von 128, so wird die Anzahl der Bytes automatisch auf das nächstgrößere Vielfache von 128 erhöht. Erfolgt keine Angabe der Länge, so werden 4096 Bytes reserviert.
- Bei Random Files ist die aktuelle Länge unabhängig von der tatsächlichen Belegung mit Daten immer gleich der maximalen Länge.

Beispiel:
Auf einer User-Disk soll unter dem Namen SFILE ein File mit sequentiellem Zugriff erstellt werden.
Es soll möglich sein, 1000 Strings mit je zehn Zeichen abzuspeichern.

Der erforderliche Platzbedarf beträgt daher in Bytes:
1000 * 4 * INT ((10-1)/4 +2) = 4000 * 4 = 16 000.
Der CREATE Befehl zur Erstellung dieses Files lautet daher:

```
CRE U, SFILE, S, 16 000.
```

5.5 ÖFFNEN VON DATEN-FILES

Bevor ein Programm auf ein externes Datenfile zugreifen kann, ist es erforderlich, das File durch Angabe seines Namens in einer FILES oder FILE: Anweisung zu öffnen. Jedem angeführten File wird ein logischer Kommunikationskanal zwischen Hauptspeicher und Floppy-Disk zugeordnet. Jeder Kanal hat eine Nummer, die im weiteren Ablauf des Programmes das File identifiziert, das mit seinem Namen dem Kanal zugeordnet wurde. Die Zuordnung der Kanäle zu den Files wird durch die Position des Filenamens im FILES-Statement bestimmt. Die daraus entstehende Ordnungszahl heißt "Filedesignator".

Nach dem Öffnen des Files sind Ein- und Ausgabeoperationen mit diesem File möglich, es muß jedoch zwischen Randomfiles und sequentiellen Files unterschieden werden.

5.5.1 Sequentielle Files

Nach der Eröffnung eines sequentiellen Files in der FILES oder in einer FILE: Anweisung kann im File von Beginn an gelesen werden. Soll auf das File geschrieben werden, so ist das erst nach der Anweisung SCRATCH: möglich, falls von Beginn des Files an geschrieben werden soll, oder nach APPEND: falls an ein bestehendes File Daten angehängt werden sollen.
Soll aus dem Schreib-Mode wieder in den Lese-Mode übergegangen werden, bzw. mit dem Lesen erneut begonnen werden, so ist nach der Anweisung RESTORE: ein Lesen des Files von Beginn an möglich.

5.5.2 Random Files

Neben allen Möglichkeiten, die sequentielle Files in einem Programm bieten, gibt es bei Files mit random-Zugriff folgende Möglichkeiten: Nach seiner Öffnung in einer FILES oder FILE: Anweisung kann sowohl gelesen als auch geschrieben werden. Die Positionierung auf ein bestimmtes Wort im File erfolgt mit einer SETW: Anweisung.

Es gibt zwei Arten von Random-Files :

- R -File

  Wird das File mit dem Parameter R vereinbart, so enthält es nach Ausführung von CREATE numerische Daten in einfacher Genauigkeit, die den Wert "nicht initialisiert" haben.

- Z -File

  Wird das File mit dem Parameter Z vereinbart, so enthält es nach Ausführung von CREATE numerische Daten in einfacher Genauigkeit, die den Wert ø (Null) haben.

5.5.3 Text - Files

Text-Files können in einem Programm wie sequentielle Datenfiles im Lese-Mode behandelt werden. Nach dem Öffnen des Files oder nach Ausführung der Anweisung RESTORE : können Textfiles mit READ: gelesen werden.
Jede Zeile des Textfiles entspricht einem String, wobei die Zeilennummer Bestandteil dieses Strings ist.

5.6 SCHÜTZEN VON FILES

Das System P6060 ermöglicht es, Programm- Text- und Daten-Files vor bestimmten Operationen zu schützen. Es ist dadurch z.B. möglich, Programme zu erstellen, die zwar ausgeführt, aber nicht gedruckt oder modifiziert werden können.
Der Schutz einzelner Files ist mit dem Befehl SECURE möglich.

6. SYSTEMBEFEHLE

6. SYSTEMBEFEHLE

Der Rechner P 6060 erlaubt es, technische und wissenschaftliche Probleme mit Programmen in der problemorientierten Sprache BASIC zu lösen.
Auch die Kommunikation mit dem System erfordert eine Sprache; diese Sprache besteht aus Systembefehlen, die das Erstellen und die Ausführung eines BASIC-Programmes rasch und bequem ermöglichen. Mit den verfügbaren Befehlen kann man unter anderem

- Programme erstellen
- Programme ausführen
- Programme modifizieren
- Programme speichern
- Daten- und Textfiles erstellen und abspeichern
- Bibliotheken verwalten.

Diese Befehle ermöglichen außerdem noch den Austausch von Information zwischen dem Hauptspeicher und den externen Einheiten der P6060: der Floppy-Disk und dem Drucker.

In diesem Kapitel soll der Aufbau und die Anwendung dieser Systembefehle besprochen werden.
Bevor die einzelnen Systembefehle besprochen werden, ist es notwendig, die allgemeinen Regeln und die Elemente der Systembefehle kennenzulernen. Diese Elemente zu besprechen, ist die Aufgabe des nächsten Kapitels.

6.1. DIE EINGABE EINES BEFEHLES

Ein Befehl besteht aus einem Schlüsselwort, dem meistens noch ein oder mehrere Operanden folgen. Die Schlüsselwörter sind englische Begriffe, die die Funktion des Befehles beschreiben.
Die Operanden liefern die zusätzlichen Informationen zur Ausführung der Operation. Für die Schlüsselwörter der am häufigsten verwendeten Befehle sind eigene Tasten vorhanden.
Das beschleunigt die Eingabe der Befehle und hilft, Eingabefehler zu vermeiden. Alle Schlüsselwörter können auf die ersten 3 Buchstaben abgekürzt werden; das System analysiert nur die ersten 3 Zeichen zur Bestimmung eines Befehles.

Wird nach dem dritten Zeichen eines Schlüsselwortes ein falscher Buchstabe eingegeben, so wird dieser Name trotzdem richtig interpretiert.
Nach Eingabe des Schlüsselwortes sind die Operanden zeichenweise einzutasten; die Anweisung wird dann durch das Drücken der END OF LINE-Taste abgeschlossen. Zwischen dem Schlüsselwort und dem ersten Operanden müssen eine oder mehrere Leerstellen stehen.
Nach Drücken der END OF LINE-Taste wird der Befehl analysiert. Wenn der Befehl vom System interpretiert werden kann, wird er unmittelbar ausgeführt, andernfalls wird eine Fehlermeldung ausgegeben. Der Befehl kann dann mit den Möglichkeiten der Editing-Operationen im Display korrigiert werden.

6. 2. NOTATIONEN

Die folgenden Symbole werden nur dazu verwendet, um das Format eines Befehles zu definieren; sie sind kein Bestandteil des Befehls:

- \- Bindestrich
- _ Unterstreichung
- {} geschwungene Klammer
- [] eckige Klammer
- ... Folge von Punkten

{ } schließt eine Folge von Parametern ein, von denen mindestens ein Parameter vorkommen muß.

[ ] schließt eine Folge von optionalen Parametern ein. Ein solcher Parameter kann, muß aber nicht angegeben werden.

_ Stehen mehrere Parameter zur Auswahl und wird kein Parameter angegeben, so wird vom System der unterstrichene Parameter eingesetzt.
Ein unterstrichener Parameter muß also nicht explizit eingegeben werden.

Beispiel :

Die Befehle SAVE S, PROG
und SAVE ,PROG
sind äquivalent.

.... der vorherige Operand kann mehrmals wiederholt werden.

, trennt die Operanden eines Systembefehls voneinander.

Beachten Sie, daß das Format eines Befehles genau eingehalten werden muß.

Beispiel :

CAT F ist falsch, richtig ist : CAT ,,,F

6.3 LISTE UND FUNKTION DER SYSTEMBEFEHLE

Die Systembefehle und ihre Funktionen sind in alphabetischer Reihenfolge angeführt.

NAME	FUNKTION
AUTO #	bewirkt die automatische Vorgabe von Zeilennummern bei der Eingabe von Text- oder Programmzeilen.
CATALOG	druckt das Inhaltsverzeichnis der Bibliotheken aus.
COMPILE	wandelt ein Textfile in ein ausführbares BASIC-Programm um.
CONFIGURE	ermöglicht die Verkleinerung des in der Hardware vorhandenen Anwenderspeichers und erlaubt die Festlegung der IPSO-Peripherie, die anstelle des Thermodruckers als Ausgabeeinheit verwendet werden soll.
CREATE	reserviert für ein Datenfile den Speicherplatz auf einer Diskette.
DATE	Das über die Tastatur eingegebene Datum wird auf der System-Disk gespeichert und dient zur Datierung der Operationen auf externen Files.
DCHANGE	Eine Diskette kann gegen eine andere ausgetauscht werden, während das System eingeschaltet bleibt . Dadurch wird das Programm im Hauptspeicher nicht gelöscht.

DECOMPILE	Ein ausführbares Programm im Arbeitsspeicher wird in ein Textfile umgewandelt.
DELETE LINE	löscht eine oder mehrere Zeilen eines Programmes oder eines Textfiles.
EXEC	Ein Dienstprogramm (siehe Kapitel 7) wird in den Arbeitsspeicher geladen und ausgeführt.
FETCH	Eine Zeile eines Programmes oder Textfiles aus dem Arbeitsspeicher wird in den Tastaturpuffer übertragen.
LDKEYS	Den Funktionstasten wird der auf der System-Disk gespeicherte Standardinhalt zugewiesen.
LINK	Ein als Textfile auf einer Diskette gespeichertes Unterprogramm (oder eine mehrzeilige Funktion) wird in ein Programm im Hauptspeicher integriert.
LIST	Es werden eine oder mehrere Zeilen eines im Arbeitsspeicher befindlichen Programmes oder Textfiles ausgedruckt.
MODIFY	Der Name und/oder der reservierte Speicherplatz eines Files auf einer Diskette wird geändert.
NEW	Ein Programm kann über die Tastatur eingegeben werden.
OLD	Ein Programm oder ein Textfile wird von einer Diskette in den Arbeitsspeicher geladen.
OPTIONS	Zusätzliche Teile des Betriebssystems werden in den Hauptspeicher geladen. Der für den Anwender verfügbare Teil des Hauptspeichers wird dadurch verkleinert.
PREPARE	Syntaktische Analyse eines Programmes und Umwandlung in ein (lauffähiges) Objekt-Programm. Nach Ausführung des Befehls befindet sich das System im DEBUGGING-Mode.
PURGE	Ein File in einer nicht geschützten Bibliothek wird gelöscht.
REPLACE	Ein Programm (oder ein Textfile) auf einer Diskette wird durch das Programm (oder den Text) gleichen Namens ersetzt, das sich im Arbeitsspeicher befindet.
RESEQUENCE	ermöglicht die Neu- bzw. Umnumerierung eines Programmes oder eines Textes im Arbeitsspeicher.

RUN	startet die Ausführung eines Programmes im Arbeitsspeicher oder von Floppy-Disk.
SAVE	Ein Programm oder ein Textfile im Arbeitsspeicher wird unter einem Namen auf einer Diskette gespeichert.
SECURE	schützt ein Programm oder einen Datenfile ganz oder teilweise vor dem Ausdruck oder Display und vor Änderungen.
SHIFT	ermöglicht das Addieren einer Konstanten zu den im Arbeitsspeicher vorhandenen Zeilennummern ab einer bestimmten Stelle.
SPACE	Der verfügbare Speicherplatz auf den beiden Disketten wird im Display angezeigt.
START	ermöglicht die Abarbeitung eines Programmes ab einer angegebenen Zeilennummer. (Eingabe im DEBUGGING-MODE).
STOP	Das Programm geht an der durch die Zeilennummer angegebenen Stelle in den DEBUGGING-MODE über (Eingabe im DEB.-MODE).
STKEYS	der aktuelle Inhalt der Funktionstasten wird auf die System-Disk als Standardbelegung gespeichert.
TEXT	Ein Textfile kann über Tastatur eingegeben werden.
TRANSCODE	Ein Datenfile wird in ein Textfile konvertiert und umgekehrt.
TRUNCATE	der reservierte Speicherplatz eines sequentiellen Datenfiles wird auf die aktuelle Länge des Files reduziert.
VALIDATE	Ein offen gebliebenes File wird geschlossen.

AUTO#

Der Befehl AUTO# (Automatic)

Funktion: Für die Anweisungen eines Programmes oder die Zeilen eines Textfiles werden automatisch Zeilennummern vergeben.

Format: AUT [O#][Zeilennr.] [, Schrittweite]

Zeilennummer ist eine positive ganze Zahl zwischen 1 und 9999 und gibt die Nummer der nächstfolgenden Anweisung oder Textzeile an.
Schrittweite ist eine positive ganze Zahl und gibt die Schrittweite zur nächsten Zeilennummer an.

Wirkung:

- Sobald eine Anweisung oder Textzeile mit der Taste EOL abgeschlossen und in den Arbeitsspeicher übertragen wird, wird die Nummer der nächsten Zeile im Tastaturpuffer generiert und im Display angezeigt.
- Fehlt die Option-Angabe, so wählt das System als nächste Zeilennummer die um 10 erhöhte, bisher höchste Zeilennummer und erhöht jeweils um 10. Steht im Arbeitsspeicher noch keine Programm- oder Textzeile, so beginnt die Numerierung bei 10.
- Fehlt nur der Parameter Zeilennr., so wählt das System als nächste Zeilennummer die um die Schrittweite erhöhte bisher größte Zeilennummer im Hauptspeicher. Steht im Hauptspeicher noch keine Programm- oder Textzeile, so beginnt die Numerierung mit der Zahl Schrittweite.
- Ohne Parameter-Angabe Schrittweite beginnt die Numerierung bei Zeilennr. und die Schrittweite beträgt 1o.

Bemerkung:

- Um die automatische Zeilennumerierung zu unterbrechen, drückt man die Taste CLEAR / RECALL mit der Taste SHIFT
  Das System geht in den COMMAND-MODE.
  Will man die automatische Zeilennumerierung wieder fortsetzen, muß neuerlich der Befehl AUTO# eingegeben werden.
- Das Schlüsselwort des Befehles kann durch die Taste SAVE / AUTO# auf der Befehls-Tastatur eingegeben werden.

Beispiele: Beispiel 1: Automatische Vergabe von Zeilennummern bei der Programmerstellung.

Automatisch: Anfangswert 10

Schrittweite 10

```
NEW
AUTO#
10 DIM A(20)
20 FOR I=1 TO 20
30 INPUT A(I)
40 NEXT I
50 END
```

Beispiel 2: Zeilennumerierung bei der Eingabe eines Textfiles mit vorgewähltem Anfangswert und vorgewählter Schrittweite.

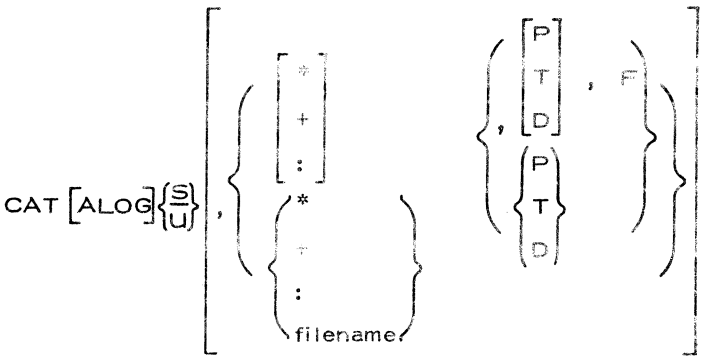
```
TEXT
AUTO# 3,2
3 Das ist eine Texteingabe,
5 die automatisch bei 3 beginnend
7 mit einer Schrittweite von 2
9 numeriert wird.
```

CATALOG

Der Befehl CATALOG

Funktion: Das Inhaltsverzeichnis von Bibliotheken wird ausgedruckt.

Format:



"S" gibt die System-Disk an

"U" gibt die User-Disk an

"*" gibt die Package-Bibliothek an

"+" gibt die Common-Bibliothek an

":" steht für alle Bibliotheken

"Filename" Falls nur die Informationen über ein bestimmtes File gedruckt werden sollen, wird "Filename" durch den Namen des Files ersetzt.

"P" Es soll nur der Katalog der Programmfiles gedruckt werden.

"T" Es soll nur der Katalog der Textfiles gedruckt werden.

"D" Es soll nur der Katalog der Datenfiles gedruckt werden.

"F" gibt an, daß die vollständige Information über die Files gedruckt werden soll.

Wirkung: Der vollständige Befehl mit allen Operanden (Bsp. CATALOG U, : , P , F) bewirkt den Ausdruck der Informationen über alle Files, die durch die ersten 3 Operanden spezifiziert werden:

- Filename
- Art des Files
- Erstellungsdatum des Files
- Datum der letzten Modifikation des Files

- Für das File reservierter Speicherplatz (in Byte)
- Aktuelle Länge des Files (in Byte)
- Identifikationscode des Files

Bei Release 3.Ø :

- Anzahl der Abschnitte, in die das File auf der Floppy-Disk zerlegt ist (max. 4)

Fehlt der 4. Operand, so werden folgende verkürzte Informationen über die Files, die durch die anderen Operanden spezifiziert wurden, ausgegeben :

. Filename (ohne Bibliotheksangabe * oder +)
. Art des Files

Fehlt der 3. Operand, so erfolgt der Ausdruck der Informationen über Programm-, Text- und Datenfiles.

Fehlt der 2. Operand, so erfolgt der Ausdruck der Informationen über die Files der Benutzerbibliothek.

Datenfiles, die nicht durch ein reguläres Programmende geschlossen wurden, werden im Katalog durch das Wort OPEN gekennzeichnet.

Bemerkung :

- Steht als zweiter Operand Filename, so ist der dritte Operand bedeutungslos.
- Die Ausführung des Befehls kann mit der Konsol-Taste BREAK beendet werden.

Beispiele :

CAT U	Es werden die Namen und der Typ aller Files der User-Library der User-Disk gelistet.
CAT U, +, T, F	Es wird ein Katalog der Textfiles der Common-Library der User-Disk gedruckt.
CAT, * , , F	Es wird ein vollständiger Katalog aller Files der Package-Library der System-Disk gedruckt.

Beispiele : Ausdruck aller Bibliotheken der System-Disk mit vollständiger Information.

1. Release 2.Ø :

```
CAT S,:,,F

  * R E L E A S E  2.0  *          VOLLABEL =  K01179
FILE    TYPE  CREAT   LAST MOD  MAX SIZE  USED SIZE  CODE NUMBER


*FILE     S  070777   070777     10112       0

+FILE     S  070777   070777     10112       0

 SEQ      S  070777   070777     10112       0
 PROG1    P  070777   070777     256         256
 RAN      R  070777   070777     10112       10112
 PROG2    P  070777   070777     256         256
 ZER      Z  070777   070777     10112       10112
```

2. Release 3.Ø :

```
CAT S,:,,F

       *  K0FDS0-R 3.0  *          VOLLABEL =  K01281
FILE    TYPE  CREAT     LAST MOD   MAX SIZE  USED SIZE  CODE NUMBER  EXT


*FILE     S  07-07-77  07-07-77    10112       0                      1

+FILE     S  07-07-77  07-07-77    10112       0                      1

 SEQ      S  07-07-77  07-07-77    10112       0                      2
 PROG1    P  07-07-77  07-07-77    256         256                    1
 RAN      R  07-07-77  07-07-77    10112       10112                  2
 PROG2    P  07-07-77  07-07-77    256         256                    1
 ZER      Z  07-07-77  07-07-77    10112       10112                  2
```

COMPILE

Der Befehl COMPILE

Funktion: Dieser Befehl übersetzt ein Textfile im Arbeitsspeicher in ein ausführbares BASIC-Programm.

Format: COM[PILE]

Wirkung: Ein Textfile im Arbeitsspeicher wird in ein ausführbares BASIC-Programm umgewandelt.

Bemerkung: Das Textfile im Arbeitsspeicher muß ein BASIC-Quellenprogramm sein. Ist das nicht der Fall, so erfolgt eine Fehlermeldung.
Sind im Textfile syntaktische Fehler, so werden die entsprechenden Fehlermeldungen nach der vollständigen Übersetzung des Files gedruckt.
Die Zeilennummern der fehlerhaften Zeilen bleiben erhalten; diese können dadurch für die notwendigen Korrekturen in den Tastatur-Puffer geholt werden (Befehl FETCH).

Beispiel 1: Eingabe eines Programmes als Text mit nachfolgender Compilierung und Ausführung

```
TEX
AUTO# 5,5
5 INPUT I
10 X=PI+I
15 PRINT X
20 END
COM
RUN
**** FORMALLY CORRECT PROGRAM ****
?
14
 17.141593
```

Beispiel 2: Fehlerhafte Programmeingabe mit Korrektur:

```
TEXT
AUTO#
10 INPUT I
20 DIEP I
30 GOTO 10
40 END

COM
ERROR 102 IN LINE 20

FETCH 20
0020 DIEP I
0020 DISP I
RUN
**** FORMALLY CORRECT PROGRAM ****
?
10000
 10000
?
  :
  :
```

Zeile 20 : falsches Schlüsselwort DIEP.

Es erfolgt eine Syntax-Kontrolle.

Korrektur der Zeile 20.

Nachfolgende Ausführung ohne neuerliche

Eingabe von COM.

CONFIGURE

Der Befehl CONFIGURE

Funktion : Neufestlegung der Größe des Anwenderspeichers und/oder der Standardausgabeeinheit über das Betriebssystem.

Format : CON [FIGURE] [{ EP = n_1 [, MS = n_2] / MS = n_2 }]

n_1 und n_2 sind ganze Zahlen, für die gilt : $\emptyset \leq n_1 \leq 31$

$8 \leq n_2 \leq M$

mit M = in der Hardware vorhandene Anzahl K-Byte.

Wirkung : Der Inhalt des Hauptspeichers wird gelöscht und das System neu initialisiert. Die Parameter werden auf der Systemdiskette gespeichert und bestimmen so lange die Konfiguration, bis sie durch einen neuen CONFIGURE-Befehl geändert werden.

- Parameter EP (External Printer) ist angegeben :

  Die periphere Einheit, die unter der Adresse n_1 über IPSO-Interface angeschlossen ist, wird statt des Thermodruckers als Ausgabeeinheit verwendet. Das System arbeitet mit dem gesamten Hauptspeicher.

- Parameter MS (Memory Size) ist angegeben :

  Die Größe des Anwenderspeichers wird auf n_2 K-Byte festgelegt, Ausgabeeinheit ist der Thermodrucker.

- Beide Parameter sind angegeben :

  Die periphere Einheit, die unter der Adresse n_1 über IPSO-Interface angeschlossen ist, wird als Ausgabeeinheit verwendet und die Größe des Anwenderspeichers wird auf n_2 K-Byte festgelegt.

- Beide Parameter fehlen :

  Ausgabeeinheit ist der Thermodrucker, die Größe des Anwenderspeichers entspricht der in der Hardware vorhandenen Anzahl K-Byte.

Bemerkungen :

- Anstelle des Thermodruckers kann ein Schnelldrucker wie der 1220, 1230, 1240 oder eine andere Einheit mit dem gleichen Steuerzeichen verwendet werden. Es steht nur der Zeichensatz des gewählten Gerätes zur Verfügung.

- Zeichnungen der Option PLOT werden in jedem Falle über den Thermodrucker ausgeführt.

- Durch die Angabe des Parameters EP werden im Arbeitsspeicher 312 Byte belegt.

- Treten bei der Übertragung zum peripheren Drucker Fehler auf, so erfolgt im Display die Meldung ABN PRT. Nach Beheben der Fehlerursache kann mit CLEAR und CONTINUE fortgesetzt werden.

Beispiel :

```
NEW
10END
PRE
**** FORMALLY CORRECT PROGRAM ****
:: ROOM=26346 ::
```

Das System hat 32 K Anwenderspeicher, es sind alle Options geladen.

```
CON MS=16
```

```
NEW
10END
PRE
**** FORMALLY CORRECT PROGRAM ****
:: ROOM=9962  ::
```

Durch den Befehl CON wird der Anwenderspeicher auf 16 K verkleinert.

Um wieder den vollen Speicher zu erhalten, wird

```
CON
```

eingegeben.

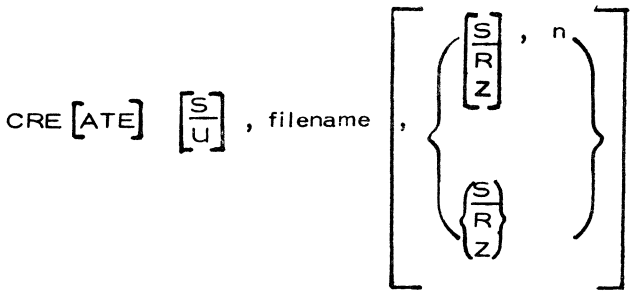
```
NEW
10END
PRE
**** FORMALLY CORRECT PROGRAM ****
:: ROOM=26346 ::
```

CREATE

Der Befehl CREATE

Funktion : Für ein Datenfile wird in einer Bibliothek auf einer Diskette der angegebene Speicherplatz reserviert.

Format :



"S" gibt an, daß der Platz auf der System-Disk reserviert werden soll.

"U" gibt an, daß der Platz auf der User-Disk reserviert werden soll.

"filename" wird ersetzt durch den Namen des Files, das erstellt werden soll.

"S" legt fest, daß das File vom Typ "SEQUENTIELL" ist.

"R" legt fest, daß das File vom Typ "RANDOM" ist. Es enthält numerische Daten in einfacher Genauigkeit, die den Wert "nicht initialisiert" haben.

"Z" legt fest, daß das File vom Typ "RANDOM" ist. Es enthält numerische Daten in einfacher Genauigkeit, die den Wert Ø (Null) haben.

"n" wird ersetzt durch die Anzahl der Bytes, die vom System für das File reserviert werden sollen.

n ist eine ganz positive Zahl mit : $1 \leq n \leq 242\,120$

Wirkung :

- Der vollständige Befehl mit allen Parametern reserviert für das File mit den Namen Filename n Bytes auf der angegebenen Diskette.
- Fehlt der vierte Operand, so werden für das File 4096 Bytes auf der angegebenen Diskette reserviert.
- Die Anzahl n von Bytes wird vom System automatisch auf Vielfache von 128 aufgerundet.

Bemerkung : In einer Bibliothek dürfen zwei Files nicht den gleichen Namen haben.

- Ist eine Package-Bibliothek geschützt, so können darin keine weiteren Files gespeichert werden.
- Man kann in einer Bibliothek nicht mehr Files erstellen, als im Dienstprogramm LBCREATE bei der Erstellung der Bibliothek festgelegt wurde (siehe Abschnitt 6.4).

Beispiel:

```
CRE U,*NAME,R,8192
```

In der Package-Bibliothek der User-Disk wird ein Randomfile mit 8K-Byte erstellt.

```
CRE ,USFIL,S
```

In der User-Bibliothek der System-Disk wird ein sequentielles Datenfile mit der Standardlänge von 4096 Byte erstellt.

DATE

Der Befehl DATE

Funktion : Die Operationen mit den Files auf einer Floppy-Disk werden datiert.

Format : DAT [E] Datum

"Datum" ist eine Folge von 6 beliebigen ISO-Zeichen mit Ausnahme des Leerzeichens.

Wirkung : Bei jedem Laden des Systems (nach dem Einschalten oder nach der Ausführung eines OPTIONS- oder CONFIGURE-Befehls) wird die Zeichenfolge, die zuletzt als Datum auf der System-Disk gespeichert wurde, in den Hauptspeicher geladen.

Bemerkung : Die Zeichenfolge wird im allgemeinen folgendermaßen aussehen :

TTMMJJ (Tag, Monat, Jahr)

Es ist zu beachten, daß die Ausgabe der Zeichenfolge in drei Gruppen zu je zwei Zeichen getrennt durch "-", erfolgt : die eingegebene Zeichenfolge "JULI77" wird mit "JU-LI-77" ausgegeben.

Beispiele :

```
DATE JULI77
FILE    TYPE  CREAT     LAST MOD  MAX SIZE  USED SIZE  CODE NUMBER  EXT

 FILE     S  JU-LI-77  JU-LI-77    4096        0                     1
DATE 070777
FILE    TYPE  CREAT     LAST MOD  MAX SIZE  USED SIZE  CODE NUMBER  EXT

 FILE     S  JU-LI-77  07-07-77    4096        0                     1
```

DCHANGE

Der Befehl DCHANGE (Disk change)

Funktion: Dieser Befehl ermöglicht das Austauschen von Floppy-Disks, während das System geladen ist, ohne dabei den Inhalt des Arbeitsspeichers zu löschen.

Format: DCH [ANGE] [S / U]

"S" gibt an, daß die System-Disk gewechselt wird.
"U" gibt an, daß die User-Disk gewechselt wird.

Wirkung:

- Bei einer Konfiguration mit zwei Platten kann man die durch den Operanden angegebenen Floppy-Disk austauschen.
- Wurde bis zu diesem Befehl nur mit der System-Disk gearbeitet und wird der Befehl DCH U eingegeben, kann danach mit beiden Disketten gearbeitet werden.
- Nach Eingabe des Befehles gibt das System die Meldung
  INSERT DISK > NEW {USDIS / SYSDIS} ON DRIVE {* / **}
  aus, je nachdem, ob der Operand S oder U eingegeben wurde.
  Nachdem man die gewünschte Platte eingelegt hat, muß man die Konsol-Taste CONTINUE drücken.

Bemerkung: Wechselt man eine Systemplatte aus, so muß die neue Platte demselben Release angehören.

Wird eine Disk ohne vorherige Eingabe des Befehls DCHANGE gewechselt, so erfolgt im Display die Meldung

ABN FD - DCH OMITTED

Sobald ein Systembefehl eingegeben wird. Durch Drücken der Taste CLEAR (oder der Tasten RECALL und CONTINUE) kehrt das System in den Command-Mode zurück.

Beispiele:

Wechseln der System-Disk

```
DCH
INSERT DISK                    >NEW SYSDIS ON DRIVE **
```

Wechseln der User-Disk

```
DCH U
INSERT DISK                    >NEW USDIS ON DRIVE *
```

DECOMPILE

Der Befehl DECOMPILE

Funktion : Ein BASIC-Programm im Arbeitsspeicher wird in ein Textfile umgewandelt.

Format : DEC [OMPILE]

Wirkung : Das BASIC-Programm im Hauptspeicher wird in ein Textfile umgewandelt.

Bei der Decompilierung werden Variablennamen oder Zeilennummern die noch in den Tabellen enthalten sind, aufgrund von Änderungen aber nicht mehr verwendet bzw. angesprochen werden aus den Tabellen entfernt. Durch Eingabe der Befehle DEC-COM kann ein Programm dadurch verkürzt und im Ablauf beschleunigt werden.

Bemerkung : Durch den Befehl SECURE geschützte Programme können nicht in ein Textfile konvertiert werden.

Es können nur Zeilen decompiliert werden, die außer der Zeilennummer höchstens 76 signifikante Zeichen enthalten.

Beispiel :

```
LIST
FILE
```

Programmeingabe

```
0010 RANDOMIZE
0020 PRINT USING 30,RND,RND,RND,RND
0030  ##.#####    ##.#####    ##.#####    ##.#####
0040 END

END OF LISTING
```

```
DEC
LIST ,,X
FILE
```

Umwandlung des Programmes in einen Text

```
 RANDOMIZE
'PRINT USING 30,RND,RND,RND,RND
  ##.#####    ##.#####    ##.#####    ##.#####
 END

END OF LISTING
```

Ausdruck als Text ohne Zeilennummer

DELETE LINE

Der Befehl DELETE LINE

Funktion: Eine oder mehrere Programm- oder Textzeilen im Arbeitsspeicher werden gelöscht.

Format: DEL[ETE LINE][[Zeilennummer$_1$][, Zeilennummer$_2$]]

"Zeilennummer$_1$" gibt die Nummer der Zeile an, (oder die erste Zeile eines Abschnittes), die (der) gelöscht werden soll.

"Zeilennummer$_2$" gibt die letzte Zeile eines Abschnittes an, der gelöscht werden soll.

Wirkung:

- Der vollständige Befehl mit allen Operanden gibt dem System an, daß alle Zeilen des durch Zeilennummer$_1$ und Zeilennummer$_2$ bestimmten Abschnittes des Programmes oder des Textfiles im Arbeitsspeicher gelöscht werden sollen.
- Fehlt der zweite Operand, so wird nur die Zeile, deren Zeilennummer durch den ersten Operanden angegeben wird, gelöscht.
- Fehlen beide Operanden, so wird eine der folgenden Zeilen gelöscht:
  1) Die letzte über Tastatur eingegebene Zeile
  2) Die letzte Zeile, die mit dem Befehl FETCH oder mit Hilfe der Tasten ↓ oder ↑ in den Tastaturpuffer übertragen wurde.
  3) Die letzte Zeile, die mit dem Befehl LIST ausgedruckt wurde.
  4) Die letzte Zeile eines Programmes, das mit dem Befehl RUN ausgeführt wird.

Bemerkung: Zeilen von Programmen oder Textfiles, die mit dem Befehl SECURE geschützt wurden, können nicht gelöscht werden.

- das Schlüsselwort DELETE LINE kann mit der Taste DELETE LINE / RUN zusammen mit der Taste SHIFT eingegeben werden.
- im Befehl angegebene Zeilennummern müssen im Programm vorhanden sein.

Beispiel :

```
OLD TEST
LIST
FILE     TEST


0010 REM BEISPIEL"DELETE LINE"
0020 REM
0030 FOR I=1 TO 5 STEP 1
0040 PRINT I,
0050 FOR J=1 TO 4 STEP 1
0060 PRINT J;
0070 NEXT J
0080 PRINT
0090 NEXT I
0100 END

END OF LISTING


RUN

 1                1  2  3  4
 2                1  2  3  4
 3                1  2  3  4
 4                1  2  3  4
 5                1  2  3  4




DELETE LINE 30,40
DELETE LINE 90
LIST
FILE     TEST


0010 REM BEISPIEL"DELETE LINE"
0020 REM
0050 FOR J=1 TO 4 STEP 1
0060 PRINT J;
0070 NEXT J
0080 PRINT
0100 END

END OF LISTING


RUN

 1  2  3  4
```

EXEC

Der Befehl EXEC (Execute)

Funktion : Lädt ein Dienstprogramm in den Arbeitsspeicher und führt es aus.

Format : EXE C Dienstprogramm , Parameter , Parameter ...

"Dienstprogramm" gibt den Namen des Dienstprogrammes an.

"Parameter" kennzeichnet die für das jeweilige Dienstprogramm spezifischen Operanden.

Wirkung : EXEC fordert vom System das angegebene Dienstprogramm an. Dieses wird nach dem Laden in den Arbeitsspeicher ausgeführt.

Bemerkung : Durch den Befehl EXEC wird der Inhalt der Arbeitsspeicher gelöscht. Es stehen die folgenden Dienstprogramme zur Verfügung :

FDCOPY
FLCOPY
LBCREATE
LIBCOPY
LBPROTECT

Der Befehl FETCH

Funktion: Eine Programmzeile oder eine Textzeile im Hauptspeicher wird in den Tastatur-Puffer übertragen.

Format: FET [CH] [Zeilennummer]

"Zeilennummer" ist die Nummer der Zeile, die in den Tastatur-Puffer übertragen werden soll.

Wirkung:

. Die Programm- oder Textzeile mit der angegebenen Zeilennummer wird in den Tastatur-Puffer übertragen und im Display angezeigt.

. Fehlt der Operand, so wird je nachdem, welche Operation unmittelbar zuvor durchgeführt wurde, eine der folgenden Zeilen in den Tastatur-Puffer übertragen und im Display angezeigt:

- Die letzte über Tastatur eingegebene Zeile;
- Die letzte Zeile eines Files, das mit dem Befehl OLD in den Arbeitsspeicher geladen wurde;
- Die letzte mit dem Befehl LIST ausgedruckte Zeile;
- Die letzte Anweisung eines Programmes, das gerade ausgeführt wurde und noch im Arbeitsspeicher steht;
- Die letzte Zeile, die mit Hilfe der Tasten ↓ oder ↑ in den Tastatur-Puffer übertragen wurde.

Bemerkung: Gibt man im Befehl FETCH eine Zeilennummer ein, die im Arbeitsspeicher nicht existiert, so wird die Zeile mit der nächst kleineren Zeilennummer in den Tastatur-Puffer übertragen. Ist die eingegebene Zeilennummer kleiner als die kleinste Zeilennummer im Arbeitsspeicher, so wird die Zeile mit der größten Zeilennummer in den Tastatur-Puffer übertragen.

. Die im Display angezeigte Zeile hat nicht immer das gleiche Format, in dem die Zeile über die Tastatur eingegeben wurde; sie wird vorher vom System modifiziert. Gibt man zum Beispiel die Anweisung

10 A = B

ein, so wird sie mit dem Befehl FETCH im Display als

0010 LET A = B

angezeigt.

Ergibt sich beim Editing durch das System eine Zeile mit mehr als 80 Zeichen, so werden alle Leerstellen eliminiert.

. FETCH kann nicht angewendet werden, wenn ein Programm oder Text mit SECURE geschützt wurde.

. Das Schlüsselwort kann durch die Taste FETCH / NEW mit SHIFT eingegeben werden.

Beispiele:

```
 OLD TEST
 LIST
 FILE      TEST


 0010 REM BEISPIEL"DELETE LINE"
 0020 REM
 0030 FOR I=1 TO 5 STEP 1
 0040 PRINT I,
 0050 FOR J=1 TO 4 STEP 1
 0060 PRINT J;
 0070 NEXT J
 0080 PRINT
 0090 NEXT I
 0100 END

 END OF LISTING


 FETCH 30
 0030 FOR I=1 TO 5 STEP 1


 LIST 50,60
 FILE      TEST


 0050 FOR J=1 TO 4 STEP 1
 0060 PRINT J;

 END OF LISTING


 FETCH
 0060 PRINT J;



 FETCH 5
 0100 END


 FETCH 71
 0070 NEXT J
```

LDKEYS

Der Befehl LDKEYS (Load Keys)

Funktion : Die Standardbelegung der Funktionstasten wird wieder hergestellt.

Format : LDK [EYS]

Wirkung : Die aktuelle Belegung der Funktionstasten wird durch die Standardbelegung ersetzt. (s.a. STKEY)

Beispiel 1 : (Die Standardbelegung sei :

F1 : STANDARD
F2 : BELEGUNG)

```
CALC
MODE
FKEY#1  MODIFIZIERTE
```

Übergang in den CALC.-Mode
Die Belegung "STANDARD" der Taste F1 wird durch den String "MODIFIZIERTE" ersetzt.

drücken von F1 und F2 : Im Display steht : MODIFIZIERTE BELEGUNG

und Eingabe von
LDKEYS

Es wird die Standardbelegung wiederhergestellt.

```
F1  F2
STANDARDBELEGUNG
```

Beispiel 2 :

```
NEW

10 FKEY#1, AENDERUNG
20 FKEY#2, DURCH PROGRAMM
30 END
```

Angabe eines Programmes zur Modifikation.

```
RUN
**** FORMALLY CORRECT PROGRAM ****
AENDERUNG DURCH PROGRAMM
```

Im Display erscheinender Text

LDKEYS

Wiederherstellung der Standard-belegung.

LINK

Der Befehl LINK

Funktion : Ein Unterprogramm oder eine Funktionsdefinition, die als Textfile auf einer Diskette gespeichert ist, wird in das Programm im Hauptspeicher integriert.

Format : LIN [K] filename, Zeilennummer [,∝]

"filename" ist der Name des Textfiles, das in das Programm im Hauptspeicher eingefügt werden soll.

"Zeilennummer" gibt die Zeilennummer an, ab der ein Unterprogramm eingefügt werden soll.

"∝ " ist ein Großbuchstabe, der den Namen angibt, unter dem eine Funktionsdefinition in das Programm eingefügt werden soll.

Wirkung : Wird der Operand ∝ angegeben, so wird die Funktionsdefinition, die auf einer Diskette als Textfile unter dem Namen "filename" gespeichert ist, in das Programm im Hauptspeicher eingefügt. Der ersten Zeile der Funktionsdefinition wird die im LINK-Befehl angegebene "Zeilennummer" zugewiesen. Alle folgenden Zeilennummern werden in der Schrittweite der Funktionsdefinition umnumeriert. Das System weist der Funktion den Namen FN∝ zu, oder, bei einer Stringfunktion, den Namen FN∝$.

Wird nur der Operand "Zeilennummer" angegeben, so wird das Unterprogramm, das als Textfile mit dem Namen "filename" auf einer Diskette gespeichert ist, in das bestehende Programm eingefügt. Die erste Zeile des Unterprogrammes erhält die im LINK-Befehl angegebene Zeilennummer, die folgenden Zeilen werden in der Schrittweite des Unterprogrammes umnumeriert.

Bemerkung : Soll ein Unterprogramm oder eine Funktionsdefinition in ein Programm eingefügt werden, so muß durch den Befehl SHIFT an entsprechender Stelle Platz für die einzufügenden Zeilen geschaffen werden.

Nach der Ausführung des Befehls LINK steht im Arbeitsspeicher ein neues, ausführbares Programm zur Verfügung.

Es ist darauf zu achten, daß das Programm nach der Ausführung des Befehles LINK nur ein END-Statement enthält.

Beispiel 1 : Einfügen einer Funktionsdefinition in ein Programm im Hauptspeicher

```
FILE      LINK                          Funktionsdefinition (Textfile)

0090 DEF FNB
0100 PRINT "  I","SQR(I)"
0110 PRINT
0120 FOR I=1 TO N
0130 IF INT(SQR(I))<>SQR(I) THEN 150
0140 PRINT I,SQR(I)
0150 NEXT I
0160 FOR I=1 TO 10
0170 PRINT
0180 NEXT I
0190 FN*=0
0200 FNEND

END OF LISTING


FILE      LINK1                         Hauptprogramm

0100 DISP "N=";
0110 INPUT N
0120 LET Y=FNA
0130 GOTO 150
0150 DISP "WEITER";
0160 INPUT X
0170 IF X=1 THEN 100
0180 END

END OF LISTING


OLD LINK1

SHIFT 150,1000

LINK LINK,140,A
```

```
FILE     LINK1
```

Neues Hauptprogramm (nach Ausführung des Befehles LINK)

```
0100 DISP "N=";
0110 INPUT N
0120 LET Y=FNA
0130 GOTO 1150
0140 DEF FNA
0150 PRINT "  I","SQR(I)"
0160 PRINT
0170 FOR I=1 TO N STEP 1
0180 IF INT(SQR(I))<>SQR(I) THEN 200
0190 PRINT I,SQR(I)
0200 NEXT I
0210 FOR I=1 TO 10 STEP 1
0220 PRINT
0230 NEXT I
0240 LET FN*=0
0250 FNEND
1150 DISP "WEITER";
1160 INPUT X
1170 IF X=1 THEN 100
1180 END

END OF LISTING
```

Beispiel 2: Einfügen eines Unterprogrammes in ein Programm im Hauptspeicher

```
FILE     *LINK
```

Unterprogramm (Textfile)

```
0100 PRINT "  I","SQR(I)"
0110 PRINT
0120 FOR I=1 TO N
0130 IF INT(SQR(I))<>SQR(I) THEN 150
0140 PRINT I,SQR(I)
0150 NEXT I
0160 FOR I=1 TO 10
0170 PRINT
0180 NEXT I
0190 RETURN

END OF LISTING
```

```
FILE     LINK2


0100 DISP "N=";
0110 INPUT N
0120 GOSUB 140
0130 GOTO 150
0150 DISP "WEITER";
0160 INPUT X
0170 IF X=1 THEN 100
0180 END

END OF LISTING



OLD LINK2

SHIFT 150,1000

LINK *LINK,140


FILE     LINK2
```

Neues Hauptprogramm (nach Ausführung des Befehles LINK)

```
0100 DISP "N=";
0110 INPUT N
0120 GOSUB 140
0130 GOTO 1150
0140 PRINT "  I","SQR(I)"
0150 PRINT
0160 FOR I=1 TO N STEP 1
0170 IF INT(SQR(I))<>SQR(I) THEN 190
0180 PRINT I,SQR(I)
0190 NEXT I
0200 FOR I=1 TO 10 STEP 1
0210 PRINT
0220 NEXT I
0230 RETURN
1150 DISP "WEITER";
1160 INPUT X
1170 IF X=1 THEN 100
1180 END

END OF LISTING
```

LIST

Der Befehl LIST

Funktion: Eine oder mehrere Zeilen eines Programmes oder eines Textfiles im Arbeitsspeicher werden ausgedruckt.

Format: LIS [T] [Zeilennr.$_1$] [, {[Zeilennr.$_2$] ,X / Zeilennr.$_2$}]

"Zeilennummer$_1$" gibt die Zeile an, die ausgedruckt werden soll, beziehungsweise gibt die erste Zeile eines Absatzes an, der ausgedruckt werden soll.

"Zeilennummer$_2$" gibt die letzte Zeile eines Absatzes an, der ausgedruckt werden soll.

"X" unterdrückt den Ausdruck der Zeilennummern bei der Ausgabe eines Textfiles.

Wirkung:

- Programm- oder Textzeilen im Arbeitsspeicher, deren Zeilennummern nicht kleiner als Zeilennummer$_1$ und nicht größer als Zeilennummer$_2$ sind, werden ausgedruckt.
- Fehlen die beiden letzten Operanden, so werden die Programm- oder Textzeilen im Arbeitsspeicher beginnend mit der Zeile, deren Nummer als erster Operand angegeben wird, ausgedruckt.
- Fehlt der erste und der dritte Operand, so werden alle Zeilen mit einer Nummer, die kleiner oder gleich der angegebenen Nummer ist, gedruckt.
- Fehlen sämtliche Operanden, so wird das gesamte Programm bzw. das gesamte Textfile im Arbeitsspeicher gedruckt.
- Der Operand X bewirkt, daß die Zeilen eines Textfiles ohne Zeilennummern gedruckt werden.

Bemerkung: Die mit dem Befehl LIST ausgegebenen Zeilen werden in einem Standardformat gedruckt; dabei wird die Zeilennummer vierstellig mit führenden Nullen ausgegeben.
Wurde zum Beispiel
40 A = B
eingegeben, so erfolgt der Ausdruck:
0040 LET A = B.

Ergibt sich dabei eine Zeile mit mehr als 80 Zeichen, so wird sie komprimiert, indem nicht signifikante Leerstellen weggelassen werden.

- Die gleichen Editing-Operationen erfolgen auch, wenn ein Programm mit dem Befehl DECOMPILE in ein Textfile umgewandelt wird.
- Ist ein Programm oder Textfile mit SECURE geschützt, so kann das Programm oder das Textfile nicht mit LIST ausgedruckt werden.
- Das Schlüsselwort des Befehles LIST kann durch die Taste OLD LIST eingegeben werden.

Beispiele: Beispiel 1: Ausdruck von Textfiles und Programmen

```
LIST
FILE     TEST

0010 REM BEISPIEL"DELETE LINE"
0020 REM
0030 FOR I=1 TO 5 STEP 1
0040 PRINT I,
0050 FOR J=1 TO 4 STEP 1
0060 PRINT J;
0070 NEXT J
0080 PRINT
0090 NEXT I
0100 END

END OF LISTING


LIST  70
FILE     TEST

0070 NEXT J
0080 PRINT
0090 NEXT I
0100 END

END OF LISTING


LIST 30,50
FILE     TEST

0030 FOR I=1 TO 5 STEP 1
0040 PRINT I,
0050 FOR J=1 TO 4 STEP 1

END OF LISTING


LIST ,25
FILE     TEST

0010 REM BEISPIEL"DELETE LINE"
0020 REM

END OF LISTING
```

Beispiel 2:

```
LIST ,,X
FILE     TEXT


 BEISPIEL 2:
 **********

 DAS BEISPIEL ZEIGT DEN AUSDRUCK
 EINES TEXTFILES OHNE ZEILENNUMMERN.
 IM ENTSPRECHENDEN LIST-BEFEHL
 IST ALS DRITTER PARAMETER "X" ANZUGEBEN

END OF LISTING
```

MODIFY

Der Befehl MODIFY

Funktion: Der Name eines Files und/oder der für ein Datenfile reservierte Speicherplatz auf einer Diskette werden geändert.

Format: MOD [IFY] $filename_1$, { $filename_2$ [,n] / n }

"$filename_1$" ist der Name des zu modifizierenden Files.
"$filename_2$" gibt den neuen Namen für das File an.
"n" ist eine positive ganze Zahl und legt den neuen Speicherplatz fest, der für das File auf der Floppy-Disk reserviert werden soll.

Wirkung: Der vollständige Befehl mit allen drei Operanden bewirkt, daß der alte Name des Files in der Bibliothek durch den neuen Namen ersetzt wird und daß der Speicherbereich auf der Platte auf n byte geändert werden soll.

- Werden nur die Operanden $filename_1$ und $filename_2$ angegeben, so erhält das Programm-, Text- oder Datenfile den neuen Namen $filename_2$.
- Sind die Operanden $filename_1$ und $filename_2$ angegeben, wird zunächst geprüft, ob beide Filenamen zum gleichen Bibliothek-Typ gehören. Wenn nein, erfolgt eine Fehlermeldung. Andernfalls wird geprüft, ob in der entsprechenden Bibliothek ein File mit Namen $filename_2$ vorhanden ist. Falls ja, erfolgt wieder Fehlermeldung, andernfalls wird die Ausführung des Befehles fortgesetzt.
- Wird n als zweiter Operand angegeben, so behält das File seinen Namen, es werden aber n byte auf der Diskette für das File reserviert.

Bemerkung: Durch den Befehl MODIFY kann der für Datenfiles reservierte Platz nur bis zum aktuell belegten Speicherplatz verkleinert werden, um zu verhindern, daß Daten verloren gehen.

Da für Random-Files das logische Ende und physische Ende zusammenfallen, kann der für ein Random-File reservierte Speicherplatz nicht verkleinert werden.
In einer Bibliothek dürfen zwei Files nicht den gleichen Namen haben.
Für Files in Package- und Commonbibliotheken, die mit dem Dienstprogramm LB PROTECT geschützt wurden, kann nur die Größe modifiziert werden.

Beispiel: Ändern des Namens und der reservierten Länge für ein Randomfile

```
CAT S,,D,F

  * R E L E A S E  2.0  *          VOLLABEL =  K01179

FILE    TYPE  CREAT   LAST MOD  MAX SIZE  USED SIZE  CODE NUMBER

 SS       S  250277   250277     4096        400
 RFILE    R  231176   231176     128         128
 SFILE    S  260976   260976     4096        92
 SPIEL    R  260976   260976     10112       10112
```

```
MOD SS,SSNEU,500
```
Das File SS erhält den Namen SSNEU und wird verkleinert.

```
MOD SFILE,8000
```
Das File SFILE wird vergrößert.

```
MOD RFILE,RANDOM
```
Das File RFILE erhält den Namen RANDOM.

```
CAT S,,D,F

  * R E L E A S E  2.0  *          VOLLABEL =  K01179

FILE    TYPE  CREAT   LAST MOD  MAX SIZE  USED SIZE  CODE NUMBER

 SSNEU    S  250277   250277     512         400
 RANDOM   R  231176   250277     128         128
 SFILE    S  260976   260976     8064        92
 SPIEL    R  260976   260976     10112       10112
```

NEW

Der Befehl NEW

Funktion: Das System wird auf die Eingabe eines Programmes über die Tastatur vorbereitet.

Format: NEW

Wirkung: Der aktuelle Inhalt des Arbeitsspeichers wird gelöscht. Das System faßt alle im folgenden eingegebenen Zeilen als Zeilen eines neuen BASIC-Programmes auf und kontrolliert die Zeilen auf syntaktische Richtigkeit.

Bemerkung: Da der aktuelle Inhalt des Arbeitsspeichers bei der Eingabe des Befehles NEW gelöscht wird, muß zuvor einer der Befehle SAVE oder REPLACE eingegeben werden, wenn dieser Inhalt des Arbeitsspeichers gespeichert werden soll.

Nach dem Laden (z.B. nach einem OPTIONS-Befehl) des Systems kann ein Programm ohne den Befehl NEW eingegeben werden.

OLD

Der Befehl OLD

Funktion: Ein Programm oder Textfile wird von der Diskette in den Arbeitsspeicher geladen.

Format: OLD filename

"filename" ist der Name des Programmes oder des Textfiles, das in den Arbeitsspeicher geladen werden soll.

Wirkung: Es wird geprüft, ob ein Programm oder Textfile mit dem Namen "filename" auf einer Diskette gespeichert ist, wenn ja, wird das Programm (das Textfile) in den Arbeitsspeicher geladen. Der bisherige Inhalt des Arbeitsspeichers wird gelöscht.

Bemerkung: Enthält das System zwei Floppy-Disks, so wird das File zuerst auf der Systemplatte gesucht.
Hat man also zwei Programme unter demselben Namen auf der System- und auf der User-Disk gespeichert und soll das Programm von der User-Disk in den Hauptspeicher geladen werden, so muß die System-Disk gegen eine andere, die kein Programm oder Textfile dieses Namens enthält, ausgewechselt werden.

Das Schlüsselwort des Befehles kann durch Drücken der Taste SHIFT zusammen mit der Taste OLD / LIST eingegeben werden.

Beispiel:

```
OLD TEXT
LIST ,10,X
FILE     TEXT

 BEISPIEL 2:

END OF LISTING



OLD TEST
FETCH 50
0050 FOR J=1 TO 4 STEP 1
```

OPTIONS

Der Befehl OPTIONS

Funktion : Das Betriebssystem im Hauptspeicher wird um die angegebenen zusätzlichen Routinen erweitert; dabei wird der Anwenderbereich des Hauptspeichers eingeschränkt.

Format : OPT [IONS] [option] [, option] ....

"option" kann sein : MAT [RIX] , STR [ING], PLO [TTER], RS2 [32] .
Jede Option darf nur einmal im Befehl angegeben werden.

Wirkung : Es werden die im Befehl angeführten Module (= Teile) des Betriebssystems in den Anwenderteil des Hauptspeichers geladen.

Der Parameter MAT bewirkt, daß der zur Ausführung von Matrizenoperationen benötigte Modul geladen wird; es werden 1,5 K-Byte im Anwenderteil des Hauptspeichers benötigt.

Der Parameter STR bewirkt, daß der zur Ausführung von Stringoperationen benötigte Modul geladen wird. Es werden 2 K-Byte im Anwenderteil des Hauptspeichers benötigt.

Der Parameter PLO bewirkt, daß der zur Ausführung von Plottanweisungen benötigte Modul geladen wird. Die Option PLO belegt im Arbeitsspeicher einen Platz von 2 K-Byte.

Der Parameter RS2 bewirkt, daß der Modul des Betriebssystems geladen wird, der das Arbeiten mit peripheren Einheiten über ein serielles Interface (V.24, EIA RS232-C) ermöglicht. Die Option RS2 belegt im Hauptspeicher 3 K-Byte.

Der Befehl ohne Parameter bewirkt, daß die Markierung der bisher geladenen Options auf der System-Disk gelöscht wird. Das System wird ohne Options neu initialisiert.

Bemerkung : Auf der System-Disk ist gespeichert, welche Options bei der Initialisierung des Systems mitzuladen sind. Bei jeder Neuinitiallisierung - d.h. nach dem Einschalten, nach CON, OPT oder einem Fehler mit Systemabbruch (ERROR n * A) - konfiguriert sich das System entsprechend dem zuletzt ausgeführten OPTIONS-Befehl.

- Durch den OPTIONS-Befehl wird der Inhalt des Arbeitsspeichers gelöscht.
- Die entsprechenden Options werden nur bei der Ausführung der Anweisungen benötigt. Die Eingabe und syntaktische Kontrolle kann auch bei nicht geladener Option erfolgen.
- Die Option STR muß für die Ausführung folgender Anweisungen und Funktionen geladen sein :

  BPAD, CHR$, DEPAD, EXT$, PAD, REP$, SCN.

  Stringverkettung (+), Stringvektoren, Mehrfachzuweisungen (A$ = B$ = C$), Stringvergleich.

PREPARE

Der Befehl PREPARE

Funktion: Es wird die vollständige syntaktische Analyse des Programmes im Arbeitsspeicher durchgeführt und die Ausführbarkeit des Programmes überprüft. Ist das Programm formal korrekt, geht das System in den DEBUGGING MODE über.

Format: PRE [PARE] [filename]

"filename" ist der Name eines Programmes auf einer Diskette.

Wirkung:

- Das Programm mit dem Namen filename wird in den Hauptspeicher geladen und auf die Durchführbarkeit der Anweisungen überprüft.
- Ohne Angabe des Operanden wird die syntaktische Analyse des Programmes, das sich im Arbeitsspeicher befindet, durchgeführt.

Bemerkung: Erfolgt nach Eingabe des Befehles keine Fehlermeldung, so steht das Programm im Arbeitsspeicher in ausführbarer Form und das System befindet sich im DEBUGGING MODE.

Ist die Konsoltaste PRINT ALL aktiviert, so erfolgt die Meldung :

* * * * FORMALLY CORRECT PROGRAM * * * *

Außerdem wird die Meldung : : ROOM = X : : ausgedruckt.
X gibt die Anzahl der freien Bytes im Anwenderteil des Hauptspeichers an. Ferner erscheint im Display die Meldung:

PROGRAM Programmname READY TO RUN

- Erfolgte nach der Ausführung des Befehles PREPARE keine Fehlermeldung, so kann die Durchführung des Programmes im Arbeitsspeicher mit der Konsol-Taste CONTINUE gestartet werden; durch die Taste STEP kann das Programm Schritt für Schritt abgearbeitet werden.

- Treten bei der Durchführung des Befehles PREPARE Fehler im Programm auf, so werden die dazugehörigen Fehlermeldungen ausgedruckt.
  Das System ist nun im COMMAND MODE und es können die nötigen Korrekturen vorgenommen werden.

- Nicht ausführbare Anweisungen wie z.B. DCL, DIM oder BUFFER# werden bereits bei der Ausführung des Befehles berücksichtigt. Im Programm angesprochene Files müssen bereits vor der Ausführung von PREPARE kreiert werden.

- Der bei ROOM = X angegebene freie Platz wird bei der Programmausführung für die Auflösung numerischer Ausdrücke für FOR/NEXT - Schleifen , Funktionen und zur Speicherung der Rücksprungadressen bei Unterprogrammen verwendet.

- Wird nach der Ausführung von PREPARE das Programm ohne weitere Änderung abgespeichert, so entfällt bei weiteren Aufrufen durch RUN, CHAIN oder PREPARE die Vorausführung. (Näheres siehe unter Systembefehl RUN.)

Beispiel :

```
PRE VAHI
**** FORMALLY CORRECT PROGRAM ****
 :: ROOM=26346 ::
```

PURGE

Der Befehl PURGE

Funktion: Ein File in einer Bibliothek der Diskette wird gelöscht.

Format: PUR [GE] filename

"filename" ist der Name des Files, das gelöscht werden soll.

Wirkung: Das System prüft, ob auf einer Diskette ein File mit dem Namen "filename" vorhanden ist. Wenn ja, wird dieses File gelöscht.

Bemerkung: Sind bei einer Konfiguration mit zwei Disketten zwei Programme mit demselben Namen auf den Disketten gespeichert, so wird durch PURGE nur das File auf der System-Disk gelöscht.

- Bei Files in Packagebibliotheken oder in Commonbibliotheken, die vorher mit dem Dienstprogramm LBPROTECT geschützt worden sind, ist der Befehl unwirksam.

Beispiel:

```
CAT .,P,F

  * R E L E A S E  2.0  *          VOLLABEL =  K01179

FILE    TYPE  CREAT   LAST MOD   MAX SIZE   USED SIZE   CODE NUMBER

 LUNA     P  260976   260976     3200        3200
 AUTO     P  260976   260976     4352        4352
 SCHACH   P  260976   260976     512         512
 FREQU    P  260976   260976     2688        2688

PUR AUTO

CAT .,P,F

  * R E L E A S E  2.0  *          VOLLABEL =  K01179

FILE    TYPE  CREAT   LAST MOD   MAX SIZE   USED SIZE   CODE NUMBER

 LUNA     P  260976   260976     3200        3200
 SCHACH   P  260976   260976     512         512
 FREQU    P  260976   260976     2688        2688
```

REPLACE

Der Befehl REPLACE

Funktion: Ein Programm oder ein Textfile wird gegen ein anderes ausgetauscht, wobei der Name erhalten bleibt.

Format: REP [LACE]

Wirkung: Es wird geprüft, ob das Programm oder das Textfile im Arbeitsspeicher einen Namen besitzt und ob auf einer Diskette ein Programm oder Textfile unter demselben Namen abgespeichert ist. Sind beide Bedingungen erfüllt, wird das gespeicherte Programm (Textfile) durch das Programm oder den Text im Arbeitsspeicher ersetzt.

Bemerkung:

- Der Befehl ermöglicht, daß ein gespeichertes Programm (Textfile) durch eine beliebig modifizierte Fassung ersetzt werden kann.
  Nach Ausführung des Befehles ist unter dem Namen des ursprünglichen Programmes (Textfiles) die modifizierte Fassung gespeichert.
- Ein neu eingegebenes Programm (Textfile) kann nur mit dem Befehl SAVE abgespeichert werden. Soll ein unter dem Namen filename gespeichertes Programm (Textfile) durch ein neu eingegebenes Programm ersetzt werden, ist die Befehlsfolge
  PUR filename
  SAVE n , filename
- Wird mit zwei Disketten gearbeitet und ist auf jeder ein Programm (Textfile) mit Namen filename gespeichert, so wird durch REPLACE nur das Programm (das Textfile) auf der System-Disk ersetzt.
- Das Programm oder der Text im Arbeitsspeicher hat nur dann einen Namen, wenn es (er) von der Diskette geladen wurde, d.h. nach Ausführung der Befehle OLD filename, RUN filename und PREPARE filename.

Beispiel :

```
OLD TEST
LIST
FILE     TEST


0010 REM
0020 FOR I=1 TO 10 STEP 1
0030 PRINT I
0040 NEXT I
0050 END

END OF LISTING



FETCH 20
0020 FOR I=1 TO 10 STEP 1
0020 FOR I=1 TO 15 STEP 1


REP
OLD TEST
LIST
FILE     TEST


0010 REM
0020 FOR I=1 TO 15 STEP 1
0030 PRINT I
0040 NEXT I
0050 END

END OF LISTING
```

RESEQUENCE

Der Befehl RESEQUENCE

Funktion: Umnumerierung der Zeilen eines Programmes oder eines Textfiles im Arbeitsspeicher.

Format: RES [EQUENCE] [Zeilennummer][, Schrittweite]

"Zeilennummer" gibt die Zeilennummer an, die der ersten Zeile des Programmes (Textfiles) im Arbeitsspeicher zugewiesen wird.

"Schrittweite" ist eine positive ganze Zahl, die die Schrittweite der Numerierung festlegt.

Wirkung: Das System weist der ersten Zeile eines Programmes oder Textfiles im Arbeitsspeicher die Zeilennummer zu, die durch den ersten Operanden angegeben wird; jeder folgenden Zeile wird eine jeweils um die Schrittweite erhöhte Zeilennummer zugewiesen.

- Wird nur der erste Operand angegeben, so wird die durch den Operanden spezifizierte Zeilennummer der ersten Zeile des Programmes oder Textfiles zugewiesen und als Schrittweite 10 angenommen.
- Wird nun der zweite Operand angegeben, so wird die Zahl Schrittweite der ersten Zeile des Programmes oder Textfiles zugewiesen, jede folgende Zeile erhält dann eine jeweils um die Schrittweite erhöhte Zeilennummer zugewiesen.

Fehlen beide Operanden, erhält die erste Zeile die Zeilennummer 10, die Schrittweite ist ebenfalls 10.

Bemerkung: Im Programm modifiziert das System automatisch die Sprungziele bei bestimmten Anweisungen (GOTO, GOSUB, ON GOTO, ON GOSUB) im Arbeitsspeicher.

Beispiele :

```
LIST
FILE


0050 DISP "EINGABE EINER ZAHL ZWISCHEN 1 UND 10";
0060 INPUT A
0070 IF (A<1)OR (A>10) THEN 50
0080 FOR I=1 TO A STEP 1
0090 PRINT I;
0100 NEXT I
0105 PRINT
0110 GOTO 50
0120 END

END OF LISTING



RES 1,2
LIST
FILE


0001 DISP "EINGABE EINER ZAHL ZWISCHEN 1 UND 10";
0003 INPUT A
0005 IF (A<1)OR (A>10) THEN 1
0007 FOR I=1 TO A STEP 1
0009 PRINT I;
0011 NEXT I
0013 PRINT
0015 GOTO 1
0017 END

END OF LISTING



RES
LIST
FILE


0010 DISP "EINGABE EINER ZAHL ZWISCHEN 1 UND 10";
0020 INPUT A
0030 IF (A<1)OR (A>10) THEN 10
0040 FOR I=1 TO A STEP 1
0050 PRINT I;
0060 NEXT I
0070 PRINT
0080 GOTO 10
0090 END

END OF LISTING



RES 5
LIST
FILE


0005 DISP "EINGABE EINER ZAHL ZWISCHEN 1 UND 10";
0015 INPUT A
0025 IF (A<1)OR (A>10) THEN 5
0035 FOR I=1 TO A STEP 1
0045 PRINT I;
0055 NEXT I
0065 PRINT
0075 GOTO 5
0085 END

END OF LISTING
```

```
RES ,4
LIST
FILE


0004 DISP "EINGABE EINER ZAHL ZWISCHEN 1 UND 10";
0008 INPUT A
0012 IF (A<1)OR (A>10) THEN 4
0016 FOR I=1 TO A STEP 1
0020 PRINT I;
0024 NEXT I
0028 PRINT
0032 GOTO 4
0036 END

END OF LISTING
```

```
TEX
AUTO#
10 PRINT"Olivetti"
20 GOTO 10
30 END


LIST
FILE


0010 PRINT"Olivetti"
0020 GOTO 10
0030 END

END OF LISTING


RES 17,3
LIST
FILE


0017 PRINT"Olivetti"
0020 GOTO 10
0023 END

END OF LISTING
```

RUN

Der Befehl RUN

Funktion : Veranlaßt die Ausführung eines Programmes.

Format : RUN [{ Filename / Zeilennr. }]

Wirkung : Wird der Befehl RUN Filename eingegeben, wird überprüft, ob auf einer Diskette ein File mit dem Namen Filename gespeichert ist. Die Suche nach dem File beginnt immer auf der System-Diskette. Ist das File ein Programm- oder Textfile, wird es in den Arbeitsspeicher geladen. Ist das File ein Programmfile,wird es ausgeführt, andernfalls erfolgt eine Fehlermeldung.

Wird der Befehl RUN oder RUN Zeilennummer eingegeben, wird überprüft, ob der Arbeitsspeicher ein Programm enthält; wenn nein, erfolgt eine Fehlermeldung, wenn ja, wird das Programm ausgeführt.

Bemerkung : Wird RUN Zeilennummer eingegeben, beginnt die Ausführung in der angegebenen Zeile, sonst wird mit der 1. Programmzeile begonnen.

- Nach Eingabe eines RUN-Befehles wird vom System geprüft, ob eine Preexecution durchzuführen ist. Unter "Preexecution" versteht man dabei eine Prüfung des Programmes auf formale Richtigkeit (Syntaxprüfung) durch das System.

  Eine Preexecution wird nur durchgeführt, wenn

  . ein neu eingegebenes Programm zum ersten Mal ausgeführt werden soll.

  . bei der letzten Preexecution ein Fehler festgestellt wurde.

  . ein Programm nach dem letzten RUN oder PREPARE - Befehl geändert worden ist.

  . ein Programm durch RUN Filename gestartet wird, das gespeichert wurde, ohne daß zuvor eine fehlerfreie Preexecution durchgeführt worden ist.

Eine erneute Preexecution eines solchen Programmes bei Aufruf RUN Filename unterbleibt, wenn folgende Befehlsfolge ausgeführt wird :

- OLD Filename
- PREPARE
- Drücken der Taste BREAK nach fehlerfreier Preexecution
- REPLACE.

Tritt während der Preexecution ein Fehler auf, wird eine Fehlermeldung angegeben, das Programm wird nicht ausgeführt.

Ist die Konsoltaste PRINT ALL aktiviert, wird nach einer fehlerfreien Preexecution die Meldung

**** FORMALLY CORRECT PROGRAM ****

ausgedruckt.

Wird ein Programm, für das eine fehlerfreie Preexecution durchgeführt wurde, erneut gestartet, unterbleibt die Preexecution.

- Zum Zweck der Optimierung wird das Programm vor der eigentlichen Ausführung in eine Form überführt, die keine Anwendung der Befehle LIST, FETCH u.ä. erlaubt. Daher wird vom System eine Kopie der ursprünglichen Version auf der User-Diskette zwischengespeichert und nach der Ausführung oder einem Abbruch durch BREAK wieder in dem Arbeitsspeicher geladen. Wird nur mit der System-Diskette gearbeitet, erfolgt diese Zwischenspeicherung auf der System-Diskette.

  Ist auf der entsprechenden Diskette der dafür erforderliche Platz nicht verfügbar, erfolgt die Fehlermeldung ERROR 188.

Die Zwischenspeicherung entfällt, wenn der Befehl in der Form RUN Filename eingegeben wird. In diesem Fall lädt das System nach der Ausführung des Programms die gespeicherte Version erneut in den Arbeitsspeicher.

- Die Programmausführung kann mit der Konsoltaste BREAK abgebrochen werden. Begonnene Input/Output - Operationen werden regulär beendet und eventuell geöffnete Files werden geschlossen. Das System geht in den Command-Mode über.

- Mit der Konsoltaste STEP kann die Programmausführung unterbrochen werden, das System geht in den Debugging-Mode über.

Beispiel :

```
NEW
AUTO

10 PRINT "PROGRAMMBEGINN"
20 REM WIRD PREEXECUTION
30 REM DRUCHGEFÜHRT ODER NICHT ?
40 PRINT "PROGRAMMENDE"
50 END

RUN                                   Bei der ersten Ausführung wird die
                                      Preexecution durchgeführt.
**** FORMALLY CORRECT PROGRAM ****

PROGRAMMBEGINN
PROGRAMMENDE


RUN                                   Bei der zweiten Ausführung entfällt
                                      die Preexecution.
PROGRAMMBEGINN
PROGRAMMENDE

SAVE S, PROGM                         Abspeichern des Programms mit
                                      durchgeführter Preexecution.
                                      (D.h. bei weiteren Programmläufen
                                      entfällt die Preexecution).
```

SAVE

Der Befehl SAVE

Funktion: Abspeichern eines Programms oder Textfiles in einer Bibliothek.

Format: SAV[E][S/U] , filename [,MSG=n]

"filename": Name, unter dem das Programm oder Textfile abgespeichert werden soll.

"S": gibt die System-Disk an.

"U": gibt die User-Disk an.

"n": 0 oder 1

Wirkung: Es wird geprüft, ob auf der angegebenen Diskette ein File mit dem Namen "filename" gespeichert ist; wenn nicht, wird geprüft, ob in der durch das Format des Namens "filename" bestimmten Bibliothek genügend Speicherplatz vorhanden ist, wenn ja, wird das Programm bzw. das Textfile unter diesem Namen abgespeichert.
Die Angabe des Parameters "S" kann entfallen.

Bedeutung des Parameters "MSG = n" (MSG = Message) :

Er legt fest, ob und gegebenfalls welche Nachrichten während eines jeden Programmlaufs vom Betriebssystem ausgegeben werden.

- Wird für "n" 1 eingegeben, wird nur die Meldung PROGRAM filename RUNNING unterdrückt.
- Wird für "n" 0 eingesetzt, wird zusätzlich die Meldung READY nach Ende der Ausführung unterdrückt. Dies ermöglicht die Ausgabe einer Display-Anzeige durch das Programm, die auch im COMMAND-MODE erhalten bleibt.
- Ist der Parameter im Befehl nicht ausgeführt, werden alle Meldungen angegeben.

Bemerkungen:

- Der Aufbau des Namens "filename" bestimmt die Bibliothek, in die gespeichert wird.
  Im einzelnen gilt für das Format der Filenamen:

  1) *name Speicherung in Package-Bibliothek
  2) +name Speicherung in Common-Bibliothek
  3) name Speicherung in User-Bibliothek

  "name": String, der bis zu 6 alphanumerische Zeichen enthalten kann. Der String muß mit einem Buchstaben beginnen, zwischen "*" bzw. "+" und dem ersten Buchstaben des Strings darf kein Leerzeichen stehen. Der String darf ferner kein Sonderzeichen enthalten.

- Ist die Package-Bibliothek durch das Dienstprogramm LBPROTECT geschützt, kann kein weiteres File in dieser Bibliothek gespeichert werden, ein Save-Befehl

  SAVE S/U , * NAME

  bewirkt eine Fehlermeldung.

Beispiel:

Ein Programm soll unter dem Namen +PROG in die Commonbibliothek der System-Disk, und unter dem Namen *PROG in die Package-Bibliothek der User-Disk gespeichert werden.

Die Befehle lauten:

```
SAVE ,  +PROG
SAVE U, *PROG
```

SECURE

Der Befehl SECURE

Funktion: Der Befehl verhindert, daß Anweisungen oder Elemente eines Programmes über den Drucker oder das Display ausgegeben oder durch Editingoperationen verändert werden und verhindert das Neubeschreiben von Datenfiles.

Format: SEC [URE] filename [, n]

"filename": Name des Files, das geschützt werden soll.
"n": positive, ganze Zahl

Wirkung: Das System prüft, ob ein Programm oder Datenfile unter dem Namen "filename" auf einer Diskette gespeichert ist.
Wenn ja, wird dieses gesichert und es gilt daß die Befehle

-DECOMPILE
-RESEQUENCE
-MODIFY
-TRANSCODE
-TRUNCATE
-LINK

nicht ausgeführt werden.
Für die Befehle

- DELETE LINE
- FETCH
- [↓]
- [↑]
- LIST

gilt:

1) Ist im Befehl kein Parameter "n" angegeben, wird keiner dieser Befehle ausgeführt.
2) Ist ein Wert für "n" eingegeben, werden die Befehle nur ausgeführt, sofern sie sich auf eine Anweisung mit einer Zeilennummer kleiner als "n" beziehen.

Für gesicherte Datenfiles gilt :

Der Befehl

TRANSCODE

wird nicht ausgeführt.

Bemerkung : Der geschützte Bereich eines Programmes oder Datenfiles kann durch einen weiteren SECURE-Befehl erweitert, nicht aber verkleinert werden. Daten von geschützten Datenfiles können nur gelesen werden.

Beispiele :

```
NEW

10 DATA 5, 7, 10, 20, 10, 3
20 READ N
30 S = Ø
40 FOR I = 1 TON
50 READ A
60 S = S + A
70 NEXT I
80 PRINT "MITTELWERT"; S/N
90 END

SAVE U, MITTEL
SECURE MITTEL, 20
```

Das so gesicherte Programm erlaubt die Berechnung des arithmetischen Mittels einer Zahlenfolge. Der Rechenteil, der mit Zeile 20 beginnt, ist gesichert, der Inhalt der Data-Anweisung kann verändert werden, um z. B. verschiedene Datensätze auszuwerten.

```
OLD MITTEL
LIST

10 DATA 5, 7, 10, 20, 10, 2
```

Es wird nur der ungesicherte Teil des Programms gedruckt.

```
SECURED FILE

FETCH 20
ERROR 205
```

Die Zeile 20 kann nicht ins Display geholt werden.

```
20 FOR I = 1 TO 20
ERROR 205
```

Zeile 20 kann nicht verändert werden.

SHIFT

Der Befehl SHIFT

Funktion : Der Befehl erlaubt die Erhöhung der Zeilennummern ab einer beliebigen Stelle eines Programmes oder Textes.

Format : SHI[FT] Zeilennummer, Konstante

- Zeilennummer : Nummer der ersten Zeile, die geshiftet wird.
- Konstante : Positive ganze Zahl, um die die Zeilennummern erhöht werden.

Wirkung : Die Zeilennummern eines im Arbeitsspeicher vorhandenen Textes oder Programmes werden ab der Zeilennummer um den Betrag Konstante erhöht.

Bemerkung :

- Befindet sich im Arbeitsspeicher ein Programm, so werden alle Sprünge entsprechend der neuen Numerierung modifiziert.
- Die mit Zeilennummer angesprochene Zeile muß existieren.
- Die Schrittweite der Numerierung wird durch SHIFT nicht verändert.
- Ist die Summe von größter Zeilennummer und angegebener Konstante größer als 9999, so erfolgt eine Fehlermeldung.

Beispiel :

```
LIST
FILE

0005 FILES SDAT
0010 SCRATCH :1
0015 DISP "Ihr String";
0020 RKB A$
0025 IF A$="" THEN 40
0030 WRITE :1,A$ EOF 40
0035 GOTO 15
0040 DISP "ENDE"
0045 END

END OF LISTING
```

Programm vor Ausführung von SHIFT.

```
SHIFT 30,12

LIST
FILE

0005 FILES SDAT
0010 SCRATCH :1
0015 DISP "Ihr String";
0020 RKB A$
0025 IF A$="" THEN 52
0042 WRITE :1,A$ EOF 52
0047 GOTO 15
0052 DISP "ENDE"
0057 END

END OF LISTING
```

Erhöhung der Zeilennummern ab Zeile 30 um 12.

Programm nach Ausführung des Befehls.

SPACE

Der Befehl SPACE

Funktion: Mit SPACE kann der freie Platz auf den Disketten abgefragt werden.

Format: SPA [CE]

Wirkung: Getrennt für System-Disk und User-Disk wird der freie Platz auf den Disketten ausgegeben; die Angabe erfolgt in Bytes.

Beispiel:

```
SPA
SYSDIS   68980
```

Monodisksystem

```
SPA
SYSDIS   68980 ,USDIS   237770
```

Bei zwei Disketten erfolgt die Anzeige im Display getrennt für beide Disketten.

START

Der Befehl START

Funktion — Startet einen unterbrochenen Programmlauf ab einer bestimmten Zeilennummer.

Format : STA [RT] Zeilennummer

"Zeilennummer" ist eine positive ganze Zahl zwischen 1 und 9999

Wirkung : Der Befehl START bewirkt die Abarbeitung eines unterbrochenen Programmlaufes ab der durch den Parameter Zeilennummer angegebenen Programmzeile.

Bemerkung :

- Für die Ausführung des Befehls muß sich das System im DEBUGGING-MODE befinden.
- Der Parameter Zeilennummer ist so zu wählen, daß eine sinnvolle Abarbeitung des Programmes möglich ist. Das Sprungziel darf nicht innerhalb von
  - . FOR / NEXT - Schleifen
  - . Unterprogrammen
  - . mehrzeiligen Funktionsdefinitionen liegen.
- Die als Parameter angegebene Zeilennr. muß eine im Programm vorhandene Zeilennr. sein.

Beispiel :

```
LIST
FILE     START

0010 DISP "'STA  line-number' eingeben";
0020 STOP
0030 PRINT "  ZEILE 30"
0040 PRINT "  ZEILE 40"
0050 PRINT "  ZEILE 50"
0060 PRINT "  ZEILE 60"
0070 END

END OF LISTING


RUN START
'STA  line-number' eingeben

STA 50
  ZEILE 50
  ZEILE 60
```

STKEYS

Der Befehl STKEYS (Store Keys)

Funktion : Die aktuelle Belegung der Funktionstasten wird als Standardinhalt gespeichert.

Format : STK [EYS]

Wirkung : Die in den Funktionstasten F1 bis F16 gespeicherten Strings werden als neuer Standardinhalt auf der System-Disk gespeichert. Der Standardinhalt wird bei jedem Neuladen des Systems oder mit dem Befehl LDKEYS wieder in die Funktionstasten geladen. Der Standardinhalt wird durch eine zwischenzeitliche andere Belegung der Funktionstasten nicht geändert.

Beispiel :

```
LIST
FILE


0010 FKEY #1,STANDARD
0020 FKEY #2,BELEGUNG
0030 END

END OF LISTING
```

Belegung der Funktionstasten durch ein Programm.

Nach

RUN

enthält die Taste F1 den Text STANDARD,
die Taste F2 den Text BELEGUNG.

Nach

STKEYS

wird diese Belegung auf der System-Disk als Standardbelegung gespeichert. Nach Drücken der Tasten F1 und F2 erscheint der Text STANDARDBELEGUNG im Display.

Konsoltaste CALC MODE aktivieren

Belegung der Funktionstasten mittels Calculator-Mode

```
0010 FKEY #1,RUN HELP:
0020 FKEY #5,LIST:
```

Das Setzen eines Doppelpunktes nach den Systembefehlen LIST oder RUN HELP ersetzt das Drücken der "End of Line"-Taste.

Nach dem Drücken von F1 wird das Programm HELP gestartet und nach dem Drücken von F2 wird ein sich im Arbeitsspeicher befindendes Programm gelistet.

Nach STKEYS wird diese Belegung auf der System-Disk als Standardbelegung gespeichert.

Wird im Calculator-Mode oder innerhalb eines Programmes F1 durch

```
FKEY#1, AKTUELL
```

geändert, so erhält man die Standardbelegung zurück, wenn entweder der Befehl LDKEYS eingegeben oder das System neu initialisiert wird.

D.h. nach Eingabe von

```
LDKEYS
```

wird durch Drücken von F1 das Programm HELP gestartet.

STOP

Der Befehl STOP

Funktion : Unterbricht die Ausführung des Prpgrammes im Arbeitsspeicher bei einer bestimmten Zeile (vor deren Ausführung).

Format : STO [P] Zeilennummer

"Zeilennummer" ist eine positive ganze Zahl zwischen 1 und 9999

Wirkung : Die Abarbeitung des Programmes wird bei der angegebenen Zeile unterbrochen und das System geht in den DEBUGGING-MODE.

Bemerkung :

- Für die Ausführung des Befehls muß sich das System im DEBUGGING-MODE befinden.
- Die als Parameter angegebene Zeilennummer muß eine im Programm vorhandene Zeilennummer sein.

Beispiel :

```
LIST
FILE      STOSTA

0010 DISP "STO line-number EOL, STA line-number EOL eingeben";
0020 STOP
0030 PRINT "  ZEILE 30"
0040 PRINT "  ZEILE 40"
0050 PRINT "  ZEILE 50"
0060 PRINT "  ZEILE 60"
0070 END

END OF LISTING


RUN STOSTA
STO line-number EOL, STA line-number EOL eingeben

STO 50
STA 40
  ZEILE 40
```

TEXT

Der Befehl TEXT

Funktion: Das System wird auf die Eingabe eines Textfiles über die Tastatur vorbereitet.

Format: TEX [T]

Wirkung: Der aktuelle Inhalt des Arbeitsspeichers wird gelöscht. Das System faßt alle im folgenden eingegebenen Zeilen als Zeilen eines Textfiles auf. Es erfolgt keine Syntaxkontrolle.

Bemerkung: Die Zeilennummer einer Textzeile ist kein Bestandteil des Textes; sie kann durch RESEQUENCE verändert werden.

Beispiel:

```
TEXT
AUTO#
10 Beispiel
20 ********
30
40 Ein Textfile wird zunaechst mit dem Befehl SAVE
50 gespeichert, und - um es mit REPLACE nach der Aenderung wieder
60 zurueckstellen zu koennen - sofort mit OLD geladen.
70 Anschliessend wird der Text erweitert und mit REPLACE zurueckgestellt.


SAVE ,TEXT
OLD TEXT


90 Die Erweiterung des Textfiles auf der Diskette,
100 wird vom System automatisch durchgefuehrt.


REP


OLD TEXT
LIST ,,X
FILE     TEXT


 Beispiel
 ********

 Ein Textfile wird zunaechst mit dem Befehl SAVE
 gespeichert, und - um es mit REPLACE nach der Aenderung wieder
 zurueckstellen zu koennen - sofort mit OLD geladen.
 Anschliessend wird der Text erweitert und mit REPLACE zurueckgestellt.
 Die Erweiterung des Textfiles auf der Diskette
 wird vom System automatisch durchgefuehrt.

END OF LISTING
```

TRANSCODE

Der Befehl TRANSCODE

Funktion: Der Befehl konvertiert Datenfiles in Textfiles und umgekehrt.

Format: TRA [NSCODE] { T, filename / D, [S/U] , filename } [, #]

"filename": Name eines Datenfiles

"S": System-Disk

"U": User-Disk

Wirkung:

- Parameter T:

Es wird geprüft, ob auf einer Diskette ein Datenfile mit Namen "filename" existiert. Ist das File geschützt, erfolgt eine Fehlermeldung, andernfalls wird geprüft, ob alle Datenelemente alphanumerisch (Strings) sind. Ist dies der Fall, wird jedes einzelne Datenelement in eine Textzeile umgewandelt.

- Parameter # ist angegeben:
  Es wird geprüft, ob das Datenfile ein konvertiertes Textfile ist und ob bei dieser Umformung die Zeilennummer mit abgespeichert wurde. Wenn ja, werden die ursprünglichen Zeilennummern beibehalten.

- Ohne Parameter # :
  Das Textfile wird mit erster Zeilennummer 1 und mit Schrittweite 1 numeriert, eventuell mit abgespeicherte Zeilennummern werden ignoriert.
  Das Textfile befindet sich nach der Umwandlung im Arbeitsspeicher.

- Parameter D:

Es wird geprüft, ob der Arbeitsspeicher ein Textfile enthält. Es wird weiter geprüft, ob auf einer Diskette ein File mit Namen "filename" existiert;
wenn ja, erfolgt Fehlermeldung, andernfalls wird geprüft, ob auf der durch den Parameter S bzw. U bestimmten Diskette genügend Speicherplatz frei ist.

Wenn ja, wird ein sequentielles Datenfile mit dem Namen "filename" kreiert. Für den Speicherbedarf des Datenfiles gilt:
kleinstes Vielfaches von 128, das größer oder gleich der Länge des Textfiles (in Bytes) ist.
In dem so kreierten Datenfile wird jede Zeile des Textfiles als ein alphanumerisches Datenelement (String) abgespeichert.

- Parameter # ist angegeben:
  Die Nummer einer jeden Zeile wird mitgespeichert. Die gesamte Zeile wird als ein String gespeichert.

- Parameter # fehlt:
  Die Zeilennummern werden nicht mitgespeichert.

Bemerkung: Ist der Arbeitsspeicher zu klein, um ein komplettes konvertiertes Datenfile aufzunehmen, werden nur soviel Datenelemente in Textzeilen umgewandelt, wie der Arbeitsspeicher aufnehmen kann.

Enthält ein Datenelement mehr als 76 Zeichen (ohne mitgespeicherte Zeilennummern), so wird dieses Datenelement nicht konvertiert.

Während der Ausführung des Befehles (Parameter T) kann die Tastatur nicht verwendet werden.

Beispiele :

Beispiel 1 :

```
TEX
AUTO#
10 P6060 fuer
20 TEXTVERARBEITUNG
30 ideal !!
```

Eingabe des Textfiles.

```
TRA D,U,TEXT
```

Umwandlung des Textes in ein Datenfile.

```
EXE FLPRINT,TEXT
```

Ausdruck des Datenfiles mit Dienstprogramm FLPRINT

```
 P6060 fuer       TEXTVERARBEITUNG                    ideal !!
READY
```

```
TRA T,TEXT
```

Umwandlung des Datenfiles in einen Text im Arbeitsspeicher.

```
LIST
FILE

0001 P6060 fuer
0002 TEXTVERARBEITUNG
0003 ideal !!

END OF LISTING
```

Ausdruck des Textes im Arbeitsspeicher.

Beispiel 2 :

Das Programm PTRANS liegt als Textfile vor. Durch den Befehl TRANSCODE werden die Programmzeilen in Elemente des Datenfiles DPROG umgewandelt. Die ausführbare Form des Programmes PTRANS (nach COMPILE) druckt sich selbst aus.

```
LIST
FILE     PTRANS

0010 FILES DPROG
0020 DCL 80A$
0030 READ :1,A$ EOF 70
0040 PRINT USING 60,A$
0050 GOTO 30
0060 :'LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL
0070 PRINT
0080 END

END OF LISTING
```

Listing des Textfiles.

```
TRA D,U,DPROG,#
```

Umwandlung des vorliegenden Textfiles in das Datenfile DPROG.

```
COM
RUN
**** FORMALLY CORRECT PROGRAM ****
0010 FILES DPROG
0020 DCL 80A$
0030 READ :1,A$ EOF 70
0040 PRINT USING 60,A$
0050 GOTO 30
0060 :'LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL
0070 PRINT
0080 END
```

Umwandlung des vorliegenden Textfiles in ein ausführbares Programm. Das Programm druckt den Inhalt des Files DPROG.

TRUNCATE

Der Befehl TRUNCATE

Funktion: Der für ein sequentielles Datenfile reservierte Speicherplatz wird auf den aktuell belegten Platz verkleinert.

Format: TRU [NCATE] filename

Wirkung: Es wird geprüft, ob ein nicht geschütztes Datenfile mit dem Namen filename existiert. Ist das der Fall, und ist das Datenfile vom Typ "sequentiell", wird der reservierte Speicherplatz auf den aktuell belegten Platz verkleinert. (Das logische Ende des Files gilt auch als physisches.)

Bemerkung: Für Random-Files ist der Befehl unwirksam, da für Random-Files immer logisches = physisches Ende gilt.
Die maximale Länge des Datenfiles ist nach Ausführung des Befehles TRUNCATE immer ein ganzzahliges Vielfaches von 128.

Beispiel:

```
CAT U,,D,F

  * R E L E A S E  2.0  *          VOLLABEL =

FILE    TYPE   CREAT   LAST MOD   MAX SIZE   USED SIZE   CODE NUMBER

 ADRESS   R   JULI76    JULI76     10112        10112
 SDAT     S   JULI76    JULI76     10112        0
```

```
TRU ADRESS
```

TRUNCATE hat für RANDOM-Files keine Wirkung.

```
TRU SDAT
```

Das File SDAT wird auf die aktuelle Länge reduziert.

```
CAT U,,D,F

  * R E L E A S E  2.0  *          VOLLABEL =

FILE    TYPE   CREAT   LAST MOD   MAX SIZE   USED SIZE   CODE NUMBER

 ADRESS   R   JULI76    JULI76     10112        10112
 SDAT     S   JULI76    JULI76     0            0
```

VALIDATE

Der Befehl VALIDATE

Funktion : Der Befehl schließt ein offen gebliebenes Datenfile.

Format : VAL [IDATE] filename

Wirkung : Das Datenfile mit dem Namen filename, das aufgrund eines nicht regulären Programmendes (z.B. nach einem Stromausfall) offen geblieben ist, wird geschlossen.

Bemerkung :

- Ein nach Leseoperationen offen gebliebenes File ist unverändert. Wurde das File beschrieben, so kann bei Randomfiles das zuletzt geschriebene Datum zerstört sein. Bei sequentiellen Files ist nach dem Schließen die aktuelle Länge gespeichert, die es vor dem Öffnen hatte. Die Daten des Files sind nur entsprechend der gespeicherten aktuellen Länge des Files verfügbar.
- Offen gebliebene Files werden im Catalog durch 'OPEN' gekennzeichnet.

Beispiel :

```
CAT U,+,D,F

  * R E L E A S E  2.0  *          VOLLABEL =  K01179
FILE    TYPE  CREAT   LAST MOD  MAX SIZE  USED SIZE  CODE NUMBER

+RD       R  250277   250277    80110       80110
+SS       S  250277   250277    4096        408                    OPEN

VAL +SS

CAT U,+,D,F

  * R E L E A S E  2.0  *          VOLLABEL =  K01179
FILE    TYPE  CREAT   LAST MOD  MAX SIZE  USED SIZE  CODE NUMBER

+RD       R  250277   250277    80110       80110
+SS       S  250277   250277    4096        408
```

6.4 LISTE DER DIENSTPROGRAMME

Im folgenden Abschnitt wird die Funktion der im Betriebssystem enthaltenen Dienstprogramme beschrieben.

Der Aufruf eines Dienstprogrammes erfolgt unter Angabe des Namens und der erforderlichen Parameter mit dem Befehl EXEC.

Im Gegensatz zur Ausführung von Systembefehlen wird bei der Ausführung von Dienstprogrammen der Arbeitsspeicher benötigt. Soll der bisherige Inhalt des Arbeitsspeichers erhalten bleiben, so ist vor Aufruf eines Dienstprogrammes der Inhalt des Arbeitsspeichers mit SAVE oder REPLACE abzuspeichern.

FDCOPY	Kopieren von Disketten
FLCOPY	Kopieren von Files
LBCREATE	Initialisierung einer Diskette
LIBCOPY	Kopieren von Bibliotheken
LBPROTECT	Sichern einer Bibliothek

FDCOPY

Das Dienstprogramm FDCOPY (Floppy-Disk Copy)

Funktion : Das Dienstprogramm ermöglicht das Kopieren von Disketten.

Format : EXE [C] FDC [OPY] [,{S/U}]

Wirkung :

- Parameter "S"
  Es soll die System-Disk (auf die zweite Disk) kopiert werden.
- Parameter "U"
  Es soll die User-Disk (auf eine noch einzulegende Disk) kopiert werden.
- Kein Parameter
  Es wird der Parameter "S" angenommen.

Bemerkungen :

- FDCOPY kann nur für Zweidisketten-Stationen verwendet werden.
- Das System überprüft, ob die Empfängerdiskette gültige Bibliotheken des Betriebssystems oder des Anwenders enthält und druckt in diesem Fall eine entsprechende Meldung. Im Display erscheint die Meldung CONTINUE. Soll dennoch kopiert werden, so ist die Konsoltaste CONTINUE zu drücken, soll nicht kopiert werden, ist BREAK zu drücken.
- Soll eine User-Disk kopiert werden, so kommt nach dem Aufruf die Meldung

  INSERT DISK > RECEIVING DISK ON DRIVE {* / * *}

  Die System-Disk ist dann durch die Empfängerdisk zu ersetzen. Die Fortsetzung erfolgt mit CONTINUE.
- Da am Ende des Kopierens das System entweder zwei System-Disks oder zwei User-Disks enthält, erfolgt die Meldung

  END - ILLEGAL STATUS > REARRANGE DISKS
- Wird das Kopieren aufgrund eines Fehlers oder Stromausfalls abgebrochen, so enthält die Empfängerdisk keine vollständige Kopie. Der Kopiervorgang ist von Beginn an zu wiederholen.

Meldungen :

- CONTINUE ?

  Die Ausführung des Kopierens wird nicht gestartet, da die Empfängerdiskette eine gültige Diskette ist. Ist die Empfängerdisk eine gültige System-Disk, wird am Thermodrucker (oder externen Drucker)

  FILE "P6FWRZ" VALID
  FILE "P6SW " VALID
  FILE "P6FWO " VALID
  FILE "P6FSYS" VALID

  gedruckt.

  Ist die Empfängerdiskette eine gültige User-Disk, so wird

  FILE "P6FSYS" VALID

  gedruckt.

  Ist die Empfängerdiskette eine gültige Diskette eines anderen Systems (aber keine P6060-Diskette), so werden ebenfalls Meldungen gedruckt.

  Soll der Kopiervorgang gestartet werden, ist die Konsoltaste CONTINUE zu drücken, soll - da beim Einlegen der Disketten eine Verwechslung erfolgte - nicht kopiert werden, so ist die Taste BREAK zu drücken.

- BREAK OCCURED > CHECK DISK STATUS

  Diese Meldung wird angezeigt, wenn die Konsoltaste BREAK entweder nach der Meldung "CONTINUE?" aktiviert wird (Empfängerdiskette bleibt unversehrt) oder durch Drücken dieser Taste der bereits begonnene Kopiervorgang abgebrochen wird (Empfängerdiskette enthält meist keine gültige Kopie).

  Enthält das System zwei System-Disks, so ist eine zu entnehmen und gegebenenfalls durch eine User-Disk zu ersetzen. Enthält das System zwei User-Disks, so ist eine System-Disk einzulegen. Die System-Disk muß nicht in der Einheit eingelegt werden, an der sie sich vor dem Kopieren befand.

  Wurde eine gültige Diskettenkonfiguration eingelegt, so ist mit CONTINUE fortzusetzen.

- END - ILLEGAL STATUS > REARRANGE DISKS

  Das Kopieren wurde regulär beendet. Die eingelegte Diskettenkombination ist ungültig, da entweder zwei System- oder zwei User-Disks eingelegt sind. Es ist eine gültige Diskettenkombination herzustellen, wobei die System-Disk nicht an der Stelle eingelegt werden muß, wo sie vor dem Kopieren war. Dann ist mit CONTINUE fortzufahren.

- ERROR n

  Das Kopieren wurde aufgrund eines Fehlers abgebrochen. Der Fehler ist nach den Anweisungen der Fehlerliste zu beheben und das Kopieren erneut zu starten.

- ERROR n - ILLEGAL STATUS > REARRANGE DISKS

  Das Kopieren wurde aufgrund des angezeigten Fehlers abgebrochen. Das System hat eine ungültige Diskettenkombination (z. B. keine System-Disk). Die Empfängerdiskette ist nach der Behebung der Fehlerursache durch die ursprünglich eingelegte Diskette zu ersetzen und das Kopieren erneut zu starten.

- INSERT DISK > RECEIVING DISK ON DRIVE {*, **}

  Beim Kopieren von User-Disks ist die eingelegte System-Disk durch die Empfängerdisk zu ersetzen. Die Meldung gibt an, in welche Einheit (* - obere Einheit, ** - untere Einheit) die Empfängerdiskette einzulegen ist.
  Nach dem Austausch der Disketten ist mit CONTINUE fortzusetzen.

- READY

  Die Ausführung des Dienstprogrammes ist beendet, das System hat eine arbeitsfähige Diskettenkonfiguration. Das System befindet sich im Command-Mode.

Anmerkung : Ist das letzte Zeichen im Display " >", so erscheint die Meldung nicht vollständig. Der zweite Teil der Meldung wird nach Drücken der Tasten SHIFT und [⟶] sichtbar.

Beispiele : 1. Kopieren einer System-Disk

Es sind die zu kopierende System-Disk und eine beliebige User-Disk eingelegt. Die System-Disk soll auf eine neue Diskette kopiert werden.

a) Die User-Disk ist mit DCH U durch die Empfängerdiskette zu tauschen.
Eingabe von :

DCH U

Im Display wird

INSERT DISK > NEW USDIS ON DRIVE {* / **}

angezeigt. Die Empfängerdiskette ist in die entsprechende Einheit einzulegen. Nach Drücken von CONTINUE erscheint die Meldung

USDIS NOT INITIALIZD

b) Es kann sofort mit Eingabe von

EXE FDCOPY, S

fortgesetzt werden. Es wird unmittelbar mit dem Kopieren begonnen. Wird es erfolgreich beendet, erscheint die Meldung

END - ILLEGAL STATUS > REARRANGE DISKS

c) Das System enthält nun zwei System-Disks. Es ist eine Diskette - meist die Kopie - wieder durch eine User-Disk zu ersetzen. Nach Drücken von CONTINUE erscheint im Display

READY

und das System ist im Command-Mode.

2. Kopieren einer User-Disk

Das System enthält eine System- und eine User-Disk. Es wird im Beispiel angenommen, daß die Empfängerdisk bereits als User-Disk verwendet wurde.

a) Eingabe von

EXE FDCOPY, U

Im Display kommt die Meldung

INSERT DISK > RECEIVING DISK ON DRIVE {* / **}

Die Empfängerdiskette ist statt der System-Disk in die entsprechende Einheit einzulegen und das Programm ist mit CONTINUE fortzusetzen.

b) Da die soeben eingelegte Empfängerdiskette bereits als User-Disk verwendet wurde, wird

FILE "P6FSYS" VALID

gedruckt und im Display

CONTINUE ?

angezeigt. Da bewußt auf eine bereits verwendete Diskette kopiert werden soll, wird das Kopieren mit Drücken von CONTINUE gestartet.

c) Nach dem regulären Ende des Kopierens wird

END - ILLEGAL STATUS > REARRANGE DISKS

angezeigt. Es ist eine User-Disk (im allgemeinen wieder die Kopie) durch die System-Disk zu ersetzen. Nach Drücken von CONTINUE wird

READY

angezeigt und das System befindet sich im Command-Mode.

FLCOPY

DIENSTPROGRAMM FLCOPY (File Copy)

Funktion: Kopieren eines Files in eine beliebige Bibliothek auf derselben oder einer anderen Diskette.

Format: EXE[C]FLC [OPY], IN=[SYSLIB / USLIB], filename 1, OUT=[SYSLIB / USLIB],{ filename 2 / * / + }

"filename 1", "filename 2": Filenamen

Wirkung: Die Bedeutung der einzelnen Parameter ist die folgende:

- IN = [SYSLIB / USLIB] : gibt die Diskette an, auf der das zu kopierende File gespeichert ist:

  "SYSLIB": System-Disk

  "USLIB": User-Disk

  ohne Angabe: Es wird die System-Disk angenommen

- "filename 1": Name des Files, das kopiert werden soll

- OUT = [SYSLIB / USLIB]:

  "SYSLIB": Es wird auf eine System-Disk kopiert, und zwar:
  gilt IN = USLIB , wird auf die eingelegte System-Disk,
  gilt IN = SYSLIB oder nur IN = , wird auf eine andere, an Stelle der User-Disk einzulegende System-Disk kopiert.

"USLIB": Es wird auf eine User-Disk kopiert, und zwar:

gilt

IN = SYSLIB , wird auf die eingelegte,

gilt

IN = USLIB , wird auf eine andere, an Stelle der System-Disk einzulegende User-Disk kopiert.

"ohne Angabe": Es wird auf dieselbe Diskette kopiert, auf der sich das Originalfile befindet.

- Wird als vierter Parameter "*" angegeben, wird in die Package-Bibliothek kopiert.

- Wird "+" angegeben, wird in die Common-Bibliothek kopiert.

- Wird keine Angabe gemacht, wird die Kopie in der User-Bibliothek abgelegt.

- "filename 2": Name, den die Kopie erhalten soll

Es wird geprüft, ob in der Empfängerbibliothek ein File mit dem gewünschten Namen vorhanden ist; wenn ja, wird der Kopiervorgang abgebrochen, andernfalls wird gemäß den obigen Erläuterungen kopiert.
Ist die Empfänger-Bibliothek eine Package-Bibliothek, die durch das Dienstprogramm LBPROTECT geschützt worden ist, wird der Kopiervorgang abgebrochen.

Bemerkung: Ist das zu kopierende File durch den Befehl SECURE geschützt worden, ist auch die Kopie in gleicher Weise geschützt.
Ist auf der Empfängerdiskette für die Kopie nicht genügend Platz, so wird nicht kopiert.
Sequentielle Files werden entsprechend ihrer maximalen Länge kopiert.

Meldungen :

- BREAK OCCURED >CHECK DISK STATUS

  Das Dienstprogramm wurde nach dem Drücken der Taste BREAK abgebrochen. Es ist sicherzustellen, daß eine arbeitsfähige Diskettenkonfiguration eingelegt ist. Die System-Disk muß nicht in der Einheit eingelegt werden, an der sie vor dem Kopieren war. Während des eigentlichen Kopiervorgangs wird BREAK nicht akzeptiert. Die Fortsetzung erfolgt mit CONTINUE.

- END - ILLEGAL STATUS >REARRANGE DISKS

  Wurde von einer System-Disk auf eine andere System-Disk, oder von einer User-Disk auf eine andere kopiert, so enthält das System nach dem Kopieren keine gültige Diskettenkonfiguration. Es ist eine arbeitsfähige Diskettenkonfiguration herzustellen, und mit CONTINUE fortzusetzen.

- ERROR n

  Das Kopieren wurde aufgrund eines Fehlers nicht ausgeführt oder abgebrochen. Das System enthält eine arbeitsfähige Diskettenkonfiguration. Nach Beheben der Fehlerursache kann das Kopieren erneut gestartet werden.

- ERROR n - ILLEGAL STATUS >REARRANGE DISKS

  Das Kopieren wurde aufgrund eines Fehlers nicht ausgeführt oder abgebrochen. Das System enthält keine gültige Diskettenkonfiguration. Es ist eine entsprechende Diskettenkonfiguration einzulegen und mit CONTINUE fortzusetzen. Nach Beheben der Fehlerursache kann mit dem Kopieren erneut begonnen werden.

- INSERT DISK >RECEIVING {SYSDIS / USDIS} ON DRIVE {* / **}

  Im Aufruf des Dienstprogrammes stimmen die beiden Parameter nach IN= und OUT= überein. Für die Ausführung des Kopierens ist dann entsprechend als Empfängerdiskette eine zweite System- oder zweite User-Disk einzulegen. Ist diese Diskettenkonfiguration hergestellt, wird mit CONTINUE das Kopieren gestartet. Das Kopieren führt dann immer zu einer Meldung, die ILLEGAL STATUS enthält.

- READY

  Nach dem Ende des Kopierens mit gültiger Diskettenkonfiguration oder nach Herstellen einer gültigen Konfiguration wird die Ausführung des Dienstprogrammes beendet und das System befindet sich im Command-Mode.

Anmerkung : Ist das letzte Zeichen im Display das Zeichen > , so kann der zweite Teil durch Drücken der Tasten SHIFT und [→] sichtbar gemacht werden.

Beispiele :

Beispiel 1 : **Kopieren mit den eingelegten Disketten :**
Ein File der Common-Bibliothek der User-Disk wird in die User-Bibliothek der System-Disk kopiert.

```
CAT U,+,,F

  * R E L E A S E  2.0  *          VOLLABEL =  K01179

FILE    TYPE  CREAT   LAST MOD   MAX SIZE  USED SIZE  CODE NUMBER

+RD       R  250277   250277     80110       80110
+DATEN    S  250277   250277     4096        408

EXEC FLCOPY,IN=USLIB,+DATEN,OUT=SYSLIB,DAT1

READY

CAT S,:,,F

  * R E L E A S E  2.0  *          VOLLABEL =  K01179

FILE    TYPE  CREAT   LAST MOD   MAX SIZE  USED SIZE  CODE NUMBER


 DAT1     S  250277   250277     4096        408
```

Beispiel 2 : Kopieren mit Austausch der System-Disk gegen eine zweite User-Disk. Nach dem Kopieren wurde die Original-User-Disk gegen die System-Disk getauscht.

```
CAT U,,D,F

  * R E L E A S E  2.0  *          VOLLABEL =

FILE    TYPE  CREAT   LAST MOD   MAX SIZE   USED SIZE   CODE NUMBER

 PLO      S  240177   240177     17024       16384
 SGN      S  040277   040277     4096        3072
 SIGN     S  040277   040277     4096        804
 RANPLO   Z  040277   020777     40064       40064
 PLOT     S  020777   020777     24064       12288
 SPLO     S  020777   020777     10112       5120


EXE FLC,IN=USLIB,RANPLO,OUT=USLIB,*
INSERT DISK                      >RECEIVING USDIS ON DRIVE **
END - ILLEGAL STATUS             >REARRANGE DISKS
READY

CAT U,*,,F

  * R E L E A S E  2.0  *          VOLLABEL =  K01179

FILE    TYPE  CREAT   LAST MOD   MAX SIZE   USED SIZE   CODE NUMBER

*RANPLO  Z  040277   020777     40064       40064
```

LBCREATE

DIENSTPROGRAMM LBCREATE (Library Create)

Funktion : Initialisierung des File-Systems auf einer Diskette.

Format : EXE [C] LBC [REATE] [,{[S/U] [[,* = n_1], [+ = n_2] [,NP = n_3]] {S/U}}]

"n_1", "n_2", "n_3", : positive ganze Zahlen mit

$n_1 + n_2 + n_3 \leq 14$

"S" : Initialisierung der System-Disk

"U" : Initialisierung der User-Disk

"*" : Package-Library

"+" : Common-Library

"NP" : User-Library

Wirkung : Auf der angegebenen Diskette werden die Bibliotheken für die Anwender-Programme eingerichtet. Die Zahlen n_1, n_2 und n_3 legen fest, wieviele Directory-Sektoren den einzelnen Bibliotheken zur Verfügung gestellt werden. Da in jedem dieser Sektoren 13 Einträge von Filenamen gemacht werden können, wird dadurch implizit die Zahl der Files festgelegt, die in den einzelnen Bibliotheken gespeichert werden können. Werden keine Werte für n_1, n_2 oder n_3 angegeben, werden folgende Werte vom System angenommen :

$n_1 = 5$
$n_2 = 4$
$n_3 = 5$

Wird mindestens ein n_i angegeben, so werden allenfalls nicht angegebene n_i gleich Ø gesetzt.

Meldungen :

- ACTION ON UNIT {* / **} ?

  Vor Ausführung der Initialisierung wird der Benutzer darauf hingewiesen, auf welcher Diskettenstation das Dienstprogramm ausgeführt wird. Durch Drücken der Taste CONTINUE wird das Dienstprogramm gestartet; durch Drücken der Taste BREAK wird die Initialisierung nicht durchgeführt.

- READY

  Nach regulärem Ende der Initialisierung erscheint im Display die Meldung READY; das System befindet sich im COMMAND-MODE.

Bemerkungen : Nach Ausführung des Dienstprogrammes LBCREATE sind vorher gespeicherte Files gelöscht.

Beispiele :

```
EXEC LBCREATE,U
ACTION ON UNIT *  ?
READY
SPA
SYSDIS  70000 ,USDIS  240590
```

Die Bibliotheken der User-Disk werden mit den Standardwerten ertellt (*=5,+=4,NP=5)

```
EXE LBC,U,+=1
ACTION ON UNIT *  ?
READY
SPA
SYSDIS  70000 ,USDIS  242250
```

Es wird auf der User-Disk nur eine Common-Bibliothek für max. 13 Files erstellt

```
EXE LBC,S,*=2,+=3,NP=9
ACTION ON UNIT ** ?
READY
SPA
SYSDIS  70000 ,USDIS  242250
```

Die System-Disk enthält:
- eine Package-Bibliothek für 26 Files
- eine Common-Bibliothek für 39 Files
- eine User-Bibliothek für 117 Files

```
EXE LBC,,NP=1
ACTION ON UNIT ** ?
READY
SPA
SYSDIS  71670 ,USDIS  242250
```

Die System-Disk enthält eine User-Bibliothek für 13 Files

LIBCOPY

DIENSTPROGRAMM LIBCOPY (Library Copy)

Funktion: Kopieren einer oder aller Bibliotheken von einer Diskette auf eine andere.

Format: EXE [C] LIB[COPY] , IN = {SYSLIB / USLIB} , [* / + / :] , OUT = {SYSLIB / USLIB}

Wirkung: Die einzelnen Parameter haben folgende Bedeutung:

- IN = {SYSLIB / USLIB} : Gibt die Diskette an, auf der sich die zu kopierende Bibliothek befindet.

- "SYSLIB": System-Disk
- "USLIB": User-Disk
- {* / + / :} Spezifiziert die Bibliothek

" " Es ist die User-Bibliothek zu kopieren.
(kein Parameter)

"*" Es ist die Package-Bibliothek zu kopieren.

"+" Es ist die Common-Bibliothek zu kopieren.

":" Es sind alle Bibliotheken zu kopieren.

- OUT = {SYSLIB / USLIB} : gibt den Typ der Diskette an, auf die die Bibliothek kopiert werden soll

"SYSLIB": Es wird auf eine System-Disk kopiert, und zwar,
gilt IN = SYSLIB, ist an Stelle der User-Disk eine weitere System-Disk einzulegen,
gilt IN = USLIB, werden die Bibliotheken auf die eingelegte System-Disk kopiert.

"USLIB" : Es wird auf eine User-Disk kopiert, und zwar :
gilt IN = SYSLIB, wird auf die eingelegte User-Disk kopiert,
gilt IN = USLIB, ist an Stelle der System-Disk eine User-Disk einzulegen.

Meldungen :

- BREAK OCCURED > CHECK DISK STATUS

  Wird die Ausführung des Dienstprogrammes durch BREAK abgebrochen, so erscheint diese Meldung. Es ist sicherzustellen, daß eine gültige Diskettenkonfiguration eingelegt ist und es ist mit CONTINUE fortzusetzen.

  Wird das Kopieren durch BREAK unterbrochen, so wird das zu kopierende File zu Ende kopiert und dann das BREAK ausgeführt. Die System-Disk muß nicht an der Stelle eingelegt sein, an der sie vor Aufruf des Kopierens eingelegt war.

- END - ILLEGAL STATUS > REARRANGE DISKS

  Nach dem regulären Ende des Kopierens wird bei nicht arbeitsfähiger Diskettenkonfiguration diese Meldung ausgegeben. Es ist eine arbeitsfähige Diskettenkonfiguration einzulegen, wobei die System-Disk nicht an der Stelle eingelegt werden muß, an der sie vor dem Kopieren eingelegt war. Nach dem Herstellen einer arbeitsfähigen Konfiguration ist das Dienstprogramm mit CONTINUE fortzusetzen.

- ERROR n

  Das Kopieren wurde aufgrund des angezeigten Fehlers abgebrochen. Das System enthält eine arbeitsfähige Diskettenkonfiguration. Das System befindet sich im Command-Mode.

- ERROR n - ILLEGAL STATUS > REARRANGE DISKS

  Das Kopieren wurde aufgrund des angezeigten Fehlers abgebrochen. Das System enthält keine gültige Diskettenkonfiguration. (Zwei System- oder zwei User-Disks.) Es ist eine arbeitsfähige Konfiguration herzustellen und mit CONTINUE fortzusetzen. Das System befindet sich im Command-Mode.

- INSERT DISK >RECEIVING {SYSDIS / USDIS} ON DRIVE {* / **}

  Stimmen im Aufruf des Dienstprogrammes die Parameter bei IN= und OUT= überein, so wird auf eine zweite Diskette gleichen Typs kopiert. Entsprechend der Meldung ist eine System- bzw. User-Disk in die angegebene Einheit einzulegen und mit CONTINUE fortzusetzen. Nach Ausführung der Kopie kommt immer eine Meldung, die ILLEGAL STATUS enthält.

- READY

  Nach regulärem Ende des Kopierens bei gültiger Diskettenkonfiguration oder nach Herstellen einer gültigen Konfiguration und CONTINUE wird das Dienstprogramm beendet und das System befindet sich im Command-Mode.

Anmerkung : Ist das letzte Zeichen im Display das Zeichen " > ", so kann der zweite Teil der Meldung durch Drücken der Tasten SHIFT und [⟶] sichtbar gemacht werden.

Bemerkungen :

- Das Dienstprogramm ist nur verfügbar, wenn mit System-Disk und User-Disk gearbeitet wird.
- Die Kopien werden neu organisiert abgespeichert : die einzelnen Files werden zusammenhängend gespeichert, auch wenn Files der Original-Bibliothek in segmentierter Form abgespeichert waren.
- Ist zur Ausführung der Kopie nicht genügend Platz auf der Empfängerdiskette, so wird - in der Reihenfolge des Kataloges der entsprechenden Bibliothek - nur die Anzahl von Files kopiert, die vollständig kopiert werden können.

Beispiel : Der Inhalt der User-Library der System-Disk wird in die User-Library der User-Disk übertragen.

```
EXEC LIBCOPY,IN=SYSLIB,,OUT=USLIB
```

LBPROTECT

DIENSTPROGRAMM LBPROTECT (Library Protect)

Funktion: Die angegebene Bibliothek wird gesichert.

Format: EXE [C] LBP[ROTECT] [, { [SYSLIB/USLIB][, {*/+}] / {SYSLIB/USLIB} }]

Wirkung: Die einzelnen Parameter haben folgende Bedeutung:

"SYSLIB" Die zu schützende Bibliothek liegt auf der System-Disk

"USLIB" Die zu schützende Bibliothek liegt auf der User-Disk.

Wird keiner dieser beiden Parameter angegeben, wird "SYSLIB" angenommen.

"*" Es soll die Package-Bibliothek geschützt werden.

"+" Es soll die Common-Bibliothek geschützt werden.

Wird keiner dieser Parameter eingegeben, werden die Package- und die Common-Bibliothek geschützt.

Bemerkungen: Ist die Package-Bibliothek geschützt, werden folgende Befehle nicht mehr akzeptiert, soweit sie sich auf Files in dieser Bibliothek beziehen:

SAVE	TRANSCODE
CREATE	TRUNCATE
PURGE	REPLACE
MODIFY	FLCOPY

- Ist die Common-Library geschützt, werden folgende Befehle nicht durchgeführt:

  PURGE
  MODIFY

- Der Schutz bleibt beim Kopieren einer Diskette erhalten.
- User-Bibliotheken können nicht geschützt werden.
- Ist eine Bibliothek geschützt, kann dies nicht mehr rückgängig gemacht werden.

- Mit dem Dienstprogramm FLCOPY ist es nur möglich, Files aus einer geschützten Bibliothek zu kopieren.

Beispiel : Schützen der Package-Bibliothek der User-Disk :

```
EXEC LBPROTECT,USLIB,*
READY
```

```
OLD HEXA
SAVE U,*HEXA,MSG=0
ERROR  200
```

Der Versuch, ein Programm zu speichern, bewirkt die Fehlermeldung : "Unerlaubte Operation, da Bibliothek geschützt ist."

7. EINFÜHRUNG IN DIE SPRACHE BASIC

7. EINFÜHRUNG IN DIE SPRACHE BASIC

7.1 BASICZEICHEN

Die BASIC-Sprache hat einen Zeichenvorrat, einen Wortvorrat und bestimmte Regeln der Syntax, aus denen man Befehle, Daten und Variable bilden kann.
Die Zeichen unterteilt man in

- alphabetische
- numerische
- Sonderzeichen.

7.1.1 Alphabetische Zeichen

Zu den alphabetischen Zeichen gehören die Großbuchstaben des lateinischen Alphabetes und folgende 3 Zeichen:

@ kommerzielles a (Klammeraffe)
# Nummernzeichen
$ Dollarzeichen

7.1.2 Numerische Zeichen

Numerische Zeichen sind die Ziffern von 0 - 9.

7.1.3 Sonderzeichen

Die Sonderzeichen sind in folgender Tabelle zusammengefaßt:

	NAME		NAME
	Leerzeichen	;	Strichpunkt
=	Gleichheitszeichen oder Zuweisung	.	Schlußpunkt oder Dezimalpunkt
+	Additionszeichen	:	Doppelpunkt
-	Subtraktionszeichen	&	kommerzielles "und"
*	Sternchen oder Multiplikationszeichen	?	Fragezeichen
/	Schrägstrich oder Divisionszeichen	>	größer als
↑	Potenzierung	<	kleiner als
(	öffnende Klammer	,	Komma
)	schließende Klammer	"	Anführungszeichen

7.1.4 Die Leerzeichen (Blanks)

Leerzeichen können in einem Programm beliebig gesetzt werden, um die Lesbarkeit zu erhöhen.
Man beachte aber, daß Leerzeichen in den folgenden Fällen signifikant sind:

- Innerhalb von alphanumerischen Konstanten (Strings)
- In Formatanweisungen, die das Format der Ausgabedaten auf Drucker und Display spezifizieren.

Nicht zugelassen sind Leerzeichen an folgenden Stellen:

- Innerhalb einer Zeilen-Nummer
- Innerhalb von BASIC-Wörtern
- Innerhalb von Variablennamen und Funktionen
- Innerhalb numerischer Konstanten

7.2 ZAHLENDARSTELLUNG

Der numerische Wert der Zahlen wird im Dezimalsystem dargestellt.

7.2.1 Zahlenbereich

Unter der Größe einer Zahl versteht man den Absolutwert der Zahl.
Der Zahlenbereich des P6060 BASIC umfaßt alle Zahlen, die größer oder gleich 10^{-99} und kleiner oder gleich $9.999999999999 * 10^{99}$ sind.

7.2.2 Genauigkeit

Unter der Genauigkeit einer Zahl versteht man die maximale Anzahl der signifikanten Ziffern, aus denen die Zahl bestehen kann.
Der P6060 kann mit einfacher oder doppelter Genauigkeit arbeiten.
Wird für die Darstellung der Zahl die allgemeine Form $n\ 10^m$ verwendet, so gilt:

einfache Genauigkeit	doppelte Genauigkeit
$1 \leq n \leq 999999$	$1 \leq n \leq 9999999999999$
$-63 \leq m \leq +63$	$-99 \leq m \leq +99$

Das System arbeitet generell mit doppelter Genauigkeit. Für die Festlegung einfacher Genauigkeit siehe die Anweisung DCL in Kapitel 8.

7.2.3 Externe Darstellung von Zahlen

Arithmetische Zahlen können als:

- ganze Zahlen
- Dezimalzahlen in Festkommadarstellung
- Dezimalzahlen in Gleitkommadarstellung

ein- oder ausgegeben werden. Die Wahl der Darstellung ist abhängig von der Größe der Zahlen und der erforderlichen Genauigkeit. Die Zahlen können positiv oder negativ sein. Negative Zahlen werden durch ein Minuszeichen vor der Zahl dargestellt, positiven Zahlen kann ein Pluszeichen vorangesetzt werden.

7.2.3.1 Ganze Zahlen

Ganze Zahlen können bis zu 13 Ziffern aufweisen. Bei positiven Zahlen braucht das Vorzeichen nicht gesetzt zu werden.

Beispiele:

```
Ø
+4
-4
```

7.2.3.2 Dezimalzahlen in Festkommadarstellung

Dezimalzahlen in Festkommadarstellung können bis zu 13 Stellen lang und mit einem Vorzeichen versehen sein. Dem ganzzahligen Teil folgt ein Dezimalpunkt und anschließend bis zu 12 Nachkommastellen.

Beispiele:

```
9.
99.
-.99
+.99
+99.99
```

-	99999999999999	0.000000000001
+	0.0000000000001	9999999999999

7.2.3.3 Dezimalzahlen in Gleitkommadarstellung

Zahlen in Gleitkommadarstellung bestehen aus einem Vorzeichen, gefolgt von einer ganzen Zahl oder einer Dezimalzahl (Mantisse genannt), der der Buchstabe E angehängt wird. Die Zahl hinter dem Buchstaben E gibt die Zehnerpotenz an, mit der multipliziert wird und besteht aus maximal zwei Ziffern. Sie kann mit positivem oder negativem Vorzeichen versehen sein. Die Mantisse kann max. 13 signifikante Ziffern enthalten.

Beispiele: Gleitkomma/Festkomma

.99E-5	Ø.ØØØØØ99
+1.0E+1Ø	1ØØØØØØØØØØ
9E-1Ø	Ø.ØØØØØØØØØ9

7.2.4 Wahl der Zahlendarstellung

Eine Zahl kann in jeder der 3 beschriebenen Zahlendarstellungen über die Tastatur eingegeben werden. Die Zahl 9 Millionen kann z.B. auf folgende Arten dargestellt werden:

9 ØØØ ØØØ
9 ØØØ ØØØ . ØØØ
9E+6

Der Zahlenbereich des Absolutbetrages liegt zwischen 1 E -99 und 9.999999999999E+99.
Werden mehr als 13 signifikante Stellen eingegeben, so erscheint im Display eine Fehlermeldung.
Die Zahlendarstellung der Daten auf dem Display und auf dem Ausdruck kann im Programm festgelegt werden (siehe die Befehle DISP USING, PRINT USING).

7.3 NUMERISCHE KONSTANTE

Eine numerische Konstante ist eine ganze Zahl oder eine Dezimalzahl in Fest- oder Gleitkomma, deren Wert während der Programmausführung unverändert bleibt.

Beispiele:

```
X = Y/95
X = Y-99.9
X = Z*9.9E-10
```

95, 99.9 und 9.9E - 10 sind Konstante

7.4. INTERNE KONSTANTE π

Die Zahl π = 3,141592 654590 ist als interne Konstante in doppelter Genauigkeit vorhanden. Sie kann mit dem Namen PI aufgerufen werden.
Interne Konstante können im Zusammenhang mit numerischen Ausdrücken verwendet werden. Sie können mit den Vorzeichen Plus und Minus versehen werden.

Beispiele: PI*2

7.5. NUMERISCHE VARIABLE

Variable sind Größen, die mit Namen bezeichnet werden und deren Wert während der Programmausführung verändert werden kann. Variable werden durch einen beliebigen Großbuchstaben dargestellt, dem eine Ziffer (Ø-9) folgen kann.

Beispiele: Z, Z 9, Z Ø

Eine Variable, der noch kein Wert zugewiesen wurde, hat einen "nicht definierten" Wert. Wird eine solche Variable in einem arithmetischen Ausdruck verwendet, so wird an den Operator eine Fehlermeldung ausgegeben. Wird ohne Eingabe eines anderen Wertes weitergerechnet, so wird der Variablen vom System der Wert Ø zugewiesen. Die Variable gilt weiterhin als "nicht definiert".

Maximal können 128 numerische Variable im gleichen Programm verwendet werden.

7.6 STRINGKONSTANTE

Eine Stringkonstante besteht aus einer Folge von Zeichen, die von Anführungszeichen eingeschlossen werden. Das Anführungszeichen selbst ist kein Bestandteil der Konstanten. Zugelassen sind alle Zeichen der ISO-Code Tabelle.

Beispiele: "DIE FLÄCHE DES TRAPEZES BETRÄGT"
"Ø 123456789"
" VOLUMEN "

Unter der Länge einer Stringkonstanten versteht man die Anzahl der Zeichen innerhalb der Anführungszeichen. Die maximale Länge ist gleich der max. Länge einer Zeile und beträgt 75 Zeichen.

7.7 STRINGVARIABLE

Stringvariable enthalten eine Folge von zulässigen Zeichen und können im Laufe des Programmes verändert werden. Namen von Stringvariablen bestehen aus einem Großbuchstaben des Alphabetes (A – Z) gefolgt von einem Dollarzeichen ($) , oder aus einem Großbuchstaben gefolgt von einer Ziffer und dem Dollarzeichen ($).

Die Zeichen können über Tastatur eingegeben werden (siehe die Anweisungen INPUT, RKB) oder werden im Programm zugewiesen (siehe READ/DATA).

Eine Variable, der kein Wert zugewiesen wird, wird mit dem Wert "nicht definiert" initialisiert und erhält bei der Ausführung des Programmes den Wert "Nullstring"; außerdem wird eine Fehlermeldung an den Operator ausgegeben.

Stringvariable können bis zu 1023 Zeichen enthalten. Die Länge einer Stringvariablen kann mit dem Befehl DCL festgelegt werden (siehe Kapitel 8).

Wird die Länge einer Stringvariablen nicht angegeben, so wird sie automatisch auf 16 Zeichen limitiert. Wird einer Stringvariablen ein String zugewiesen, der länger ist als die deklarierte Länge für die Variable, so erfolgt eine Fehlermeldung. Wird das Programm trotzdem mit CONTINUE fortgesetzt, so wird der String rechts abgeschnitten.

7.8 FELDER (INDIZIERTE VARIABLE)

Indizierte Variable bezeichnen ein Feld von Variablen (numerische Variable oder String-Variable).

Ein Feld kann ein- oder zweidimensional sein.

Ein eindimensionales Feld (Vektor) kann als eine natürliche Folge von Elementen gedacht werden.

Ein zweidimensionales Feld hingegen ist eine Matrix, bestehend aus Zeilen und Spalten.
Ein Feldname besteht aus einem Großbuchstaben (A - Z). Der Name eines Feldes kann z.B. A sein, A2 ist nicht erlaubt.

Ein Element des Feldes wird bestimmt durch den Namen des Feldes, zusammen mit einem Index, wenn es sich um einen Vector handelt, und zusammen mit zwei Indices bei einer Matrix. Die Indices geben die Position des Elementes im Feld an.

Beispiele:

Wenn A der Name eines Vectors ist, so heißen die Elemente A(1), A(2), A(3), A(4), A(5), A(6), A(7), A(8), A(9), A(10), ...
A(4) ist also die vierte Komponente des Vectors A.

Bezeichnet Z eine Matrix, so sind die Elemente:

Z (1, 1)	Z (1, 2)	Z (1, 3)	Z (1, 4)
Z (2, 1)	Z (2, 2)	Z (2, 3)	Z (2, 4)
Z (3, 1)	Z (3, 2)	Z (3, 3)	Z (3, 4)

Das Element Z (2, 3) ist also das dritte Element der zweiten Zeile.

Die Indices können beliebige arithmetische Ausdrücke sein, die einen ganzzahligen Wert zwischen eins und der oberen Feldgrenze ergeben, wobei bei einem nicht ganzzahligen Ergebnis gerundet wird.

Die Dimension eines Feldes (die Anzahl der Elemente) wird durch den Befehl DIM festgelegt.

In einem Programm kann man einer Variablen und einem Feld denselben Namen geben; so kann B eine Variable, aber auch ein Feld angeben. Es ist jedoch nicht erlaubt, in einem Programm einem Vector und einer Matrix denselben Namen zu geben. Vor der Ausführung eines Programmes sind die Elemente "nicht definiert", erst wenn im Laufe des Programmes ein Feldelement verwendet wird, erfolgt eine Fehlermeldung. Wird ohne eine Wertzuweisung weitergerechnet, wird der Variablen vom System der Wert Ø zugewiesen; die Variable gilt weiterhin als "nicht definiert".

7.8.1 Feldvereinbarung

Unter Feldvereinbarung versteht man die Angabe, ob ein Feld ein- oder zweidimensional aufgebaut ist, sowie die Angabe der Anzahl der Elemente.
Man unterscheidet explizite und implizite Feldvereinbarungen, je nachdem, ob das Feld durch den Dimensionierungsbefehl DIM deklariert wurde, oder ob man sich im Programm ohne DIM-Anweisung auf ein Feldelement bezieht.
Die implizite Deklaration weist einem Feld pro Dimension 10 Elemente zu. Wird der Vector Z nicht explizit durch den Befehl DIM deklariert, so bedeutet das, daß der Vector Z 10 Elemente besitzt. Der Befehl Z (5) = 99 gibt an, daß dem 5. Element des Vector Z die Zahl 99 zugewiesen wird.
Ebenso bedeutet Y (5, 7) = -990. 5, daß die Matrix Y (ohne Dimensionierungsbefehl DIM) 100 Elemente (10 Zeilen und 10 Spalten) besitzt. Analog wird hier dem 7. ten Element der 5. ten Zeile die Zahl -990. 5 zugewiesen.
Bei einer Matrix für Stringvariable ist die größtmögliche Dimension ohne DIM-Anweisung 5 x 5.

7.8.2 Veränderung der Dimensionen einer Matrix

Die Dimensionen einer Matrix mit numerischen oder Stringelementen kann verändert werden, jedoch muß die Anzahl der Elemente kleiner sein als anfangs explizit oder implizit festgelegt wurde. Siehe dazu die Beschreibung der Matrix Befehle in Kapitel 8.

7.9 NAMEN VON VARIABLEN

In Abb. 1 sind die Regeln zur Bildung von Variablen-Namen nochmals zusammengefaßt:

Bedeutung	Name	Beispiel
Numerische Variable	a[Ziffer]	Z, K 3
Stringvariable	a[Ziffer]$	Z3$, Z$
Numerisches Feld	a({n / n,m})	Z(3) Z(3,4)
Feld mit Stringelementen	a$({n / n,m})	Z$(3) Z$(3,4)

a bedeutet: bel. Großbuchstabe

richtig	falsch
Z1 = 10 einfache Variable	Z (1) = 10
Z (1) = 10 indizierte Variable	Z 3 (1,2) = 100
Z$ = " WECHSELSPANNUNG" (einfache Stringvariable)	Z (5)$ = "GESCHWINDIGKEIT"
Z$(1,5) = "LEISTUNG" (indizierte Stringvariable)	Z (1,3) $ = "RAUM"

Abb. 1: Regeln zur Bildung von Variablen-Namen

7.10 STANDARDFUNKTIONEN

OLIVETTI P6060 BASIC enthält folgende Funktionen für numerische Operationen und die Verarbeitung von Zeichenketten (Strings).

7.10.1 Numerische Funktionen

Eine numerische Funktion kann in jedem arithmetischen Ausdruck mit Konstanten, Variablen und Feldelementen verwendet werden.

```
Z = SIN (X  PI) - A
A1 = SQR (SIN (X) )
```

Alle numerischen Funktionen haben nur ein Argument und ergeben einen Wert in doppelter Genauigkeit. Das Argument kann auch ein arithmetischer Ausdruck sein (siehe 7.11.1).

Ist ein Argument unzulässig, wird ein Fehler gemeldet.

Abb. 2 gibt eine Übersicht über alle numerischen Funktionen.

Name	Bezeichnung
ACS (X)	Arcuscosinus von X (Bogenmaß)
ASN (X)	Arcussinus von X (Bogenmaß)
ATN (X)	Arcustangens von X (Bogenmaß)
ABS (X)	Absolutbetrag von X
COS (X)	Cosinus von X (Bogenmaß)
COT (X)	Cotangens von X (Bogenmaß)
DEG (X)	Umwandlung von X (Bogenmaß) in Grad
EXP (X)	Exponentialfunktion von X (e^X)
HCN (X)	Cosinus hyperbolicus von X
HSN (X)	Sinus hyperbolicus von X
HTN (X)	Tangens hyperbolicus von X
INT (X)	Abrundung auf nächste kleinere ganze Zahl
IOC (X)	Abfrage des Zustandes einer peripheren Einheit
LEN (X$)	Länge der Variablen X$

Name	Bezeichnung
LGT (X)	dekadischer Logarithmus von X
LOG (X)	natürlicher Logarithmus von X
RAD (X)	Umwandlung von X (Grad) in Bogenmaß
RND	Zufallszahl zwischen 0 und 1
SGN (X)	Signum (Vorzeichen) von X
SIN (X)	Sinus von X (Bogenmaß)
SCN (A$, B$, X, Y)	Aufsuchen von B$ innerhalb A$
SQR (X)	Quadratwurzel von X
TAN (X)	Tangens von X (Bogenmaß)

Abb. 2 : Numerische Standardfunktionen
siehe ausführliche Beschreibung in Abschnitt 8.3

7.10.2 STRING-FUNKTIONEN

Eine Stringfunktion kann in allen Stringausdrücken verwendet werden und liefert als Ergebnis einen String. Die Argumente einer Stringfunktion können sowohl numerisch als auch alphanumerisch sein.

Name	Beschreibung
BLN$(X, A$, B$)	Vergleich zweier Strings gemäß Regeln der Boole'schen Algebra.
CHR$ (X)	Wandelt eine Zahl zwischen 0 und 255 in das entsprechende Zeichen der ISO-Tabelle um.
EXT $ (A$, I, L)	Teilstring von A$: Vom I -ten Zeichen beginnend bis zum L-ten Zeichen einschließlich.
REP $ (A$, B$, C$, N, I)	Ändert den String A$, indem B$ durch C$ N-mal ersetzt wird, beginnend beim I-ten Zeichen.

Abb. 4 - Standards der STRING-FUNKTIONEN
Ausführliche Beschreibung in Abschnitt 8.3

7.11 AUSDRÜCKE

Ein Ausdruck kann ein beliebiger arithmetischer Ausdruck sein oder ein Stringausdruck. Ausdrücke sind also alle Konstanten, Variablen, Felder, indizierte Variable, die Standardfunktionen und Anwenderfunktionen und alle Kombinationen davon, die man durch Verknüpfung mit Operatoren erhält.

7.11.1 Arithmetische Ausdrücke und Operatoren

Arithmetische Ausdrücke sind z.B.:

```
A 1
X/Y + 11.5 - 73 * Z 1
(X - 1Ø) * Z 1
SIN (X * Y)
-97.4
A ↑ B
```

ARITHMETISCHE OPERATOREN:

Symbol	Operatoren
↑	Potenzierung
*	Multiplikation
/	Division
+	Addition oder Pluszeichen
-	Subtraktion oder Minuszeichen
=	Zuweisung

7.11.1.1 Regeln zu den arithmetischen Operatoren

Potenzierung:

B ↑ E — Basis B wird zur Potenz von E erhoben

Voraussetzung	Ergebnis
B = E = 0	B ↑ E = 1
B = 0, E < 0	Fehlermeldung (Overflow)
B < 0, E ≠ INT (E)	Fehlermeldung (Overflow)
B ≠ 0, E = 0	B ↑ E = 1
B = 0, E > 0	B ↑ E = 0

Multiplikation und Addition:

A * B ist gleichbedeutend mit B * A

A + B ist gleichbedeutend mit B + A

es gilt also das kommutative Gesetz.

A * (B * C) ist nicht immer gleich mit (A * B) * C

A + (B + C) ist nicht immer gleich mit (A + B) + C

Das Ergebnis in der Klammer kann nämlich gerundet oder abgeschnitten werden.

Division und Subtraktion:

A / B bedeutet A dividiert durch B

ist B = 0 so erfolgt eine Fehlermeldung (Overflow)

A - B bedeutet A minus B

Vorzeichen erlaubt:

- B + (- A) + C - (- Z)

A + - B oder A * - B	Der zweite Operator (+ od. -)
A++B (wird als A+B ge-	wird hier jeweils als Vorzeichen
speichert)	von B interpretiert.

Die Zeichen + und - können zwischen einer öffnenden Klammer und einem arithmetischen Ausdruck gesetzt werden. Außerdem können sie auch noch vor dem ersten Zeichen eines arithmetischen Ausdruckes stehen.

7.11.1.2 Prioritätsregeln

Die numerischen Operatoren haben folgende Priorität:

Operation	Priorität
↑	höchste
*, /	↓
+, -	niedrigste

Operationen innerhalb der Klammern werden zuerst ausgeführt: im Beispiel (A + B - C) * D wird zuerst die Klammer berechnet und das Ergebnis mit D multipliziert. Ausdrücke mit Operatoren gleicher Priorität werden, wenn nicht Klammern es anders bestimmen, von links nach rechts ausgewertet.

7.11.2 Stringausdrücke und Operatoren

Ein Stringausdruck besteht aus Stringkonstanten, Stringvariablen, indizierten Stringvariablen oder Stringfunktionen, die eventuell auch durch den Verknüpfungsoperator verbunden sein können.

Der Verknüpfungsoperator wird durch das Symbol + dargestellt.

Stringkonstante müssen immer zwischen Anführungszeichen stehen.

Beispiele:

A$ + " = " + B$ + " * " + C$

stand in A$ VOLUMEN

in B$ GRUNDFLAECHE

in C$ HOEHE

so ergibt der Ausdruck:

VOLUMEN = GRUNDFLAECHE * HOEHE

7.11.3 Vergleichsoperatoren

Hier wird der Inhalt zweier Stringausdrücke oder der Wert von arithmetischen Ausdrücken verglichen. Das Ergebnis ist ein Wahrheitswert, "wahr" oder "falsch" (Boole'sche Variable).

Vergleichsoperatoren sind:

OPERATOR			BEDEUTUNG
	=		gleich
<>	oder	><	ungleich
>=	oder	=>	größer gleich
<=	oder	=<	kleiner gleich
	>		größer
	<		kleiner

Die allgemeine Form des Ausdrucks von Vergleichsoperationen ist:

Ausdruck 1 Vergleichsoperator Ausdruck 2

Ausdruck 1 und 2 dürfen beliebige Ausdrücke sein, jedoch ohne Vergleichsoperatoren. Es können nur 2 Ausdrücke verglichen werden.

Die zu vergleichenden Ausdrücke müssen beide numerisch oder alphanumerisch sein.

Werden Zeichenketten verglichen, so erfolgt der Vergleich Zeichen für Zeichen von links nach rechts, entsprechend der Reihenfolge der Zeichen in der ISO-Code-Tabelle.

Vergleicht man zwei Zeichenketten verschiedener Länge, so wird der kürzere String mit dem ISO-Zeichen NUL bis zur Länge der längeren Zeichenkette aufgefüllt.

Wird bis zur signifikanten Länge des kürzeren Strings Gleichheit festgestellt, so gilt der längere Ausdruck als der größere.

BEISPIEL

AB	IST GROESSER ALS	AAAAAAAAAAAAAAA
2500	IST KLEINER ALS	3
123456	IST GLEICH	123456

7.11.4 BOOLE'sche Operatoren

Ein Vergleich liefert als Ergebnis einen Wahrheitswert "wahr" oder "falsch". Die Wahrheitswerte von zwei Vergleichen können durch die boole'schen Operatoren

AND

OR

verknüpft werden und liefern wieder einen Wahrheitswert. Das allgemeine Format lautet :

(Vergleich 1) $\left\{\begin{matrix}\text{AND}\\ \text{OR}\end{matrix}\right\}$ (Vergleich 2)

Die beiden Vergleichsoperatoren "Vergleich 1" und "Vergleich 2" müssen in Klammern eingeschlossen sein.

Die Verknüpfung der beiden Vergleiche mit AND liefert genau dann den Wahrheitswert "wahr", wenn sowohl "Vergleich 1" als auch "Vergleich 2" den Wahrheitswert "wahr" hat.

Die Verknüpfung der beiden Vergleiche mit OR liefert genau dann den Wahrheitswert "wahr", wenn zumindest einer der beiden Vergleiche "Vergleich 1" oder "Vergleich 2" den Wahrheitswert "wahr" hat.

Wird für "wahr" 1 und für "falsch" Ø geschrieben, kann man die Wahrheitstafel für AND und OR bilden :

AND

Vergleich 1	1	1	0	0
Vergleich 2	1	0	1	0
(1) AND (2)	1	0	0	0

OR

Vergleich 1	1	1	0	0
Vergleich 2	1	0	1	0
(1) OR (2)	1	1	1	0

Beispiel : (A > B) OR (C > D) Der Ausdruck ist "wahr" wenn A größer als B und/oder C größer als D ist.

(A > B) AND (C > D) Der Ausdruck ist "wahr", wenn sowohl A größer als B als auch C größer als D ist.

7.12 STRUKTUR EINES BASICPROGRAMMES

Ein BASIC-Programm besteht aus einer Folge von Zeilen (Statements). Jede Zeile hat die folgende allgemeine Form:

Zeilennummer Schlüsselwort Operanden

7.12.1 Die Zeilennummer

Jede Zeile beginnt mit einer Zeilennummer, einer ganzen Zahl zwischen 1 und 9999.
Ein Programm wird in aufsteigender Reihenfolge der Zeilennummern abgearbeitet, wobei der direkte Ablauf durch spezielle Programmanweisungen wie Sprungbefehle, Unterprogrammaufrufe etc. unterbrochen werden kann.
Werden bei der Programmeingabe die Zeilen nicht in aufsteigender Reihenfolge eingegeben, so werden sie automatisch entsprechend geordnet. Lücken beliebiger Größe in der fortlaufenden Zeilennumerierung sind erlaubt.
Das genaue Vorgehen bei der Eingabe eines Programmes ist in Kapitel 9 ausführlich beschrieben.

7.12.2 Schlüsselwort

Jede Basic-Anweisung beginnt (nach der Zeilennummer) mit einem ein- bis dreiteiligen Schlüsselwort.

7.12.3 Operanden

Nach jedem Schlüsselwort, bzw. dessen Teilwort, können ein oder mehrere Operanden stehen, wie Konstanten, Variable, Ausdrücke oder Vergleichs- und Zuweisungsoperanden.

Beispiele:

a) ein- bis dreiteilige Schlüsselwörter

```
PRINT
IF ..... THEN ...
FOR ..... TO ..... STEP ...
```

b) Einfügen von Operanden

```
PRINT A * B
IF A = B THEN 200
FOR K = 1 TO 10 STEP 2/A
```

Über die Verwendung der Operanden und der möglichen Trennzeichen bestehen für jedes Schlüsselwort genaue Formatanweisungen, die in Kapitel 8 für jedes BASIC-Wort genau beschrieben sind.

8. BASIC - ANWEISUNGEN

8. BASIC - ANWEISUNGEN

Ein BASIC-Programm besteht aus einer Folge von Anweisungen, wobei man ausführbare und nicht ausführbare Anweisungen unterscheidet.

Die ausführbaren Anweisungen bewirken eine Aktion des Programmes, wie zum Beispiel die Zuweisung eines Wertes an eine Variable oder das Ausdrucken der Ergebnisse.

Nicht ausführbare Anweisungen beschreiben die für das Programm und für den Benutzer notwendigen Informationen, sie bewirken jedoch keine sichtbare Aktion. Sie können im Laufe des Programmes zwischen die ausführbaren Anweisungen gesetzt werden.

Die Anzahl der Anweisungen in einem Programm ist einerseits beschränkt durch die Speicherkapazität des OLIVETTI P6060 und andererseits durch die Art der Befehle selbst. Jede Anweisung eines BASIC-Programmes nennt man eine Zeile und muß mit einer Nummer, "der Zeilennummer", beginnen. Die Zeilennummer legt die Reihenfolge in der Abarbeitung der Befehle fest.

Alle Anweisungen werden entsprechend der Zeilennummer nacheinander ausgeführt, unabhängig von der Reihenfolge der Eingabe.

Ausnahmen davon verursachen Sprünge, Wiederholungen und Subroutinen.

Als Zeilennummer können alle ganzen Zahlen zwischen 1 und 9999 verwendet werden.

Eine Programmzeile kann maximal 80 Zeichen inklusive 4 Stellen der Zeilen-Nummer enthalten und wird über Tastatur eingegeben. Jede Zeile wird durch EOL abgeschlossen.

Das Zeichen EOL (End of Line) ist jedoch nicht Bestandteil der Programmzeile selbst.

Beispiel einer BASIC-Anweisung :

Zeilennummer	BASIC-Anweisung
10	PRINT A, B

In diesem Kapitel werden alle Anweisungen der Sprache BASIC zunächst nach Funktionen geordnet und kurz erläutert; anschließend folgt die ausführliche Beschreibung der BASIC-Anweisungen in alphabetischer Reihenfolge.

Die Beschreibung der Option MAT, der Standardfunktionen und der Darstellung von Ausgabeelementen im Standardformat beschließen dieses Kapitel.

Die Anweisungen der Option PLOT und Programmierung von IPSO-Peripherie werden in eigenen Kapiteln behandelt.

8.1 LISTE DER BASIC-ANWEISUNGEN

1. Vereinbarungen für Speicherzuweisungen

DCL	Legt fest, welche Variablen mit einfacher Genauigkeit verarbeitet werden sollen und bestimmt die maximale Länge von Strings
DIM	Legt für numerische und alphanumerische Felder die Anzahl der Elemente fest

2. Zuweisung

LET	Weist das Ergebnis eines Ausdruckes einer numerischen Variablen oder einer Stringvariablen zu

3. Programmverzweigungen

FOR/NEXT	Dient zur Bildung von Schleifen in Abhängigkeit von einer Laufvariablen
GOTO	Unbedingter Sprung zu einer Zeilennummer
IF/THEN	Bedingter Sprung zu einer Zeilennummer
ON....GOTO	Bedingter Sprung zu einer Zeilennummer in Abhängigkeit vom Wert eines Ausdruckes

4. Unterprogramme und definierte Funktionen

DEF	Dient zur Vereinbarung einzeiliger Funktionen
DEF/FNEND	Dient zur Vereinbarung mehrzeiliger Funktionen
GOSUB	Aufruf eines Unterprogrammes
ON....GOSUB	Aufruf von Unterprogrammen in Abhängigkeit vom Wert eines Ausdruckes
RETURN	Markiert das logische Ende eines Unterprogrammes

5. Anweisungen zur Ein/Ausgabe und für interne Files

DATA	Baut ein internes Datenfile auf
DISP	Ausgabe von Daten über Display im Standardformat
DISP USING	Ausgabe von Daten über Display in definiertem Format
: (IMAGE)	Festlegung eines Formates in einer Zeile, die dann mit ....USING Zeilen Nr. aufgerufen wird
INPUT	Ermöglicht die Eingabe von numerischen und/oder alphanumerischen Daten über die Tastatur
PRINT	Ausgabe über den Drucker im Standardformat
PRINT USING	Ausgabe über den Drucker im definierten Format
READ	Lesen von Daten aus einem internen File
RESTORE	Setzt den Pointer im internen Datenfile auf das erste Element
RKB	Ermöglicht die Eingabe einer beliebigen Zeichenfolge über die Tastatur, wobei alle Zeichen akzeptiert werden

6. Anweisungen für Strings

ASSIGN	Die in einem String durch einen Delimiter begrenzten Teilstrings werden einer Liste von numerischen oder alphanumerischen Variablen zugeordnet.
BASSIGN	Daten, die in einem String in internem Format enthalten sind, werden auf eine Liste von numerischen Variablen oder Stringvariablen aufgeteilt.
BBUILD	Die Ergebnisse einer Liste von numerischen oder Stringausdrücken werden einer Stringvariablen in internem Format zugewiesen.
BPAD	Eine Stringvariable wird mit binären Zeichen bis zur deklarierten Länge aufgefüllt.
BUILD	Die Werte einer Liste von Ausdrücken werden im Standardformat einer Stringvariablen zugewiesen, wobei ein Trennzeichen eingefügt werden kann.
BUILD USING	Die Werte einer Liste von Ausdrücken werden in definiertem Format einer Stringvariablen zugewiesen.
CONVERT	1.) Die ISO-Codes der Zeichen eines Stringausdruckes werden der Reihe nach den Elementen eines Vektors zugewiesen. 2.) Die Werte eines Vektors werden als ISO-Zeichen interpretiert und der Reihe nach einer Stringvariablen zugewiesen.
DEPAD	Entfernt die Füllzeichen aus einer Stringvariablen.
PAD	Eine Stringvariable wird bis zur deklarierten Länge mit ISO-Zeichen aufgefüllt.

7. Anweisungen für externe Files

APPEND:	Erlaubt das Anfügen von Daten an ein bestehendes sequentielles Datenfile
FILES	Legt die Maximalzahl der Files fest, die bei Ausführung des Programmes gleichzeitig geöffnet sein können und öffnet die, deren Namen in der Anweisung aufgeführt sind
FILE:	Schließt das File, das dem Filedesignator zugeordnet ist und erlaubt das Öffnen eines anderen Files unter Beibehaltung des Filedesignators

READ:	Erlaubt das Lesen von Daten aus einem Text- oder Datenfile
RESTORE:	Setzt in einem Text- oder sequentiellen Datenfile den Pointer auf das erste Element und erlaubt das nachfolgende Lesen von Beginn an
SCRATCH:	Setzt in einem sequentiellen Datenfile den Pointer an den Beginn und versetzt das File in den Schreib-Mode
SETW:	Der Pointer eines Random Files (R-File oder Z-File) wird auf das angegebene Wort gesetzt
WHERE:	Die aktuelle Position, der Typ des adressierten Datenelementes und die Länge eines Strings können abgefragt werden
WRITE:	Schreibt Daten auf das angegebene Datenfile

8. Spezielle Anweisungen

BEEP	Programmierbares akustisches Signal
CHAIN	Beendet die Ausführung des laufenden Programmes und startet die Ausführung eines anderen Programmes
DELAY	Bewirkt die Unterbrechung der Programmausführung für eine bestimmte Zeit
END	Gibt das physische Ende eines Programmes an
FKEY	Erlaubt die Belegung einer Funktionstaste mit einer Zeichenfolge
RANDOMIZE	Bei Aufruf der Funktion RND wird eine von der Standardfolge verschiedene Folge von Zufallszahlen erzeugt
REMARK	Ermöglicht das Einfügen von Kommentaren in ein Programm
STOP	Unterbricht die Ausführung eines Programmes und bewirkt den Übergang in den Debugging-Mode
TRACE ON	Bewirkt den Ausdruck der Zeilennummern in der Reihenfolge ihrer Verarbeitung
TRACE OFF	Hebt die Wirkung der Anweisung TRACE ON und der Konsoltaste TRACE auf

9. Anweisungen der Option MAT

MAT...=	Elementweise Zuweisung der Werte einer Matrix an eine andere
MAT...±	Matrizenaddition oder Subtraktion
MAT...*	Multiplikation einer Matrix mit einem Skalar oder Multiplikation zweier Matrizen
MAT...CON	Setzt alle Elemente einer Matrix auf 1
MAT...IDN	Erzeugt eine Einheitsmatrix
MAT INPUT	Zeilenweise Eingabe der Werte eines Feldes über Tastatur
MAT...INV	Bildet die Inverse einer Matrix
MAT READ	Elementweise Zuweisung von Daten aus einem internen File an ein oder mehrere Felder
MAT READ:	Elementweise Zuweisung von Daten aus einem externen File an ein oder mehrere Felder
MAT PRINT	Ausdruck eines oder mehrerer Felder im Standardformat
MAT PRINT USING	Ausdruck eines oder mehrerer Felder im definierten Format
MAT...TRN	Bildet die Transponierte einer Matrix
MAT WRITE:	Schreibt die Elemente eines oder mehrerer Felder auf ein Datenfile
MAT...ZER	Erzeugt eine Nullmatrix

10. Anweisungen der Option PLOT

CPLOT	Ausgabe der Ergebnisse von Stringausdrücken über den Plotter
CSIZE	Definition der Größe und Schreibrichtung für die Ausgabe von Strings über Plotter
CTAB	Verschiebung um eine bestimmte Zeichen- und Zeilenanzahl für die Ausgabe von Strings

DOT	Zeichnen eines Punktes, der durch Koordinaten definiert ist
DRAW	Ausgabeanweisung für das Bild über den integrierten Printer
EXTERNAL PLOTTER	Vorwahl für einen externen peripheren Plotter
FRAME	Festlegung der Größe der Zeichenfläche, des Rahmens für das Bild
IDOT	Zeichnen eines Punktes durch Angaben von Inkrementen dx und dy
INIMAGE	Angabe des Namens des Datenfiles, in dem das Bild gespeichert werden soll, Aufbauen des Puffers im Arbeitsspeicher und Laden der Parameter, die notwendig für die Plotoperationen sind
IPLOT	Ziehen einer Verbindungslinie zu dem Punkt, der durch Inkremente dx und dy erreicht wird
LDIMAGE	Definition eines Plotfiles, Übernahme des letzten gespeicherten Bildes mit den Parametern in den Arbeitsspeicher
MOVE	Positionieren auf einen durch Koordinaten definierten Punkt
OFFSET	Verschiebt den Koordinatenursprung im Koordinatensystem
PLOT	Ziehen einer Linie zu dem durch Koordinaten definierten Punkt
SCALE	Festlegen der Maßstäbe im Koordinatensystem durch Minimum und Maximum der beiden Achsen, implizit werden dadurch Maßstabfaktoren und der Koordinatenursprung definiert
XAXIS	Zeichnen einer Strecke parallel zur X-Achse
YAXIS	Zeichnen einer Strecke parallel zur Y-Achse

11. Anweisungen zur Programmierung von peripheren Einheiten

BUFFER	Für einen I/O-Kanal wird im Arbeitsspeicher ein Puffer für den Datenaustausch mit einer peripheren Einheit reserviert.
CMD	Kommandos, Steuerbefehle an die Peripherie
RECEIVE	Eingabebefehl, String-Input von der peripheren Eingabeeinheit in den Arbeitsspeicher
SEND	Ausgabebefehl, String-Output aus dem Arbeitsspeicher an Ausgabeeinheit
TEST	Der Status der peripheren Einheit wird in das Arbeitsregister übertragen, laufende I/O-Operationen mit dieser Einheit werden nicht unterbrochen.
WAIT	Ende der I/O-Operationen wird abgewartet und Status der Peripherie in das Arbeitsregister übertragen.
Funktion IOC (X)	Ermöglicht die Abfrage des Arbeitsregisters im Programm

APPEND:

ANWEISUNG APPEND:

Funktion : Diese Anweisung erlaubt das Hinzufügen von Daten an ein sequentielles File.

Format : APPEND: Filedesignator

Filedesignator ist ein num. Audruck.

Wirkung : Bei dem durch den Filedesignator angegebenen externen File wird der Pointer auf das erste freie Element im File gesetzt. Bei einer nachfolgenden WRITE: Anweisung mit dem gleichen Filedesignator werden die Daten an die bereits bestehenden angefügt. Ergibt die Berechnung des Ausdruckes für den Filedesignator keinen ganzzahligen Wert, so wird der Wert gerundet.

Bemerkung : Der Wert des Filedesignators muß größer als Null sein und kleiner oder gleich der Anzahl der Filenamen in der Anweisung "FILES".

Beispiel :

```
FILE     APP

0010 FILES SDAT
0020 REM DIESER TEIL BESCHREIBT DAS FILE VON BEGINN AN
0030 SCRATCH :1
0040 FOR I=1 TO 10 STEP 1
0050 WRITE :1,I
0060 NEXT I
0070 REM ES WERDEN DIE ERSTEN 5 DATEN GELESEN
0080 RESTORE :1
0090 FOR I=1 TO 5 STEP 1
0100 READ :1,I
0110 PRINT I;
0120 NEXT I
0130 PRINT
0140 REM NEUE DATEN WERDEN HINZUGEFUEGT.
0150 REM ******************************
0160 REM
0170 APPEND :1
0180 FOR I=11 TO 15 STEP 1
0190 WRITE :1,I
0200 NEXT I
0210 REM LESEN DES GESAMTEN FILES
0220 RESTORE :1
0230 READ :1,I EOF 260
0240 PRINT I;
0250 GOTO 230
0260 PRINT
0270 PRINT "FILEENDE"
0280 END

END OF LISTING


 1  2  3  4  5
 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
FILEENDE
```

ASSIGN

ANWEISUNG ASSIGN

Funktion: Diese Anweisung bewirkt die Zuweisung von Werten aus einem Stringausdruck an eine oder mehrere numerische oder alphanumerische Variable. Als Datentrennungszeichen dient ein fixer Delimiter.

Format:

ASSIGN Stringausdruck , { num. Variable / Stringvar. } [, { num. Var. / Stringvar. } ....] ; Delimiter d

Delimiter d: Numerischer Wert eines beliebigen ISO-Zeichens, $0 \leq d \leq 255$

Wirkung: Das System berechnet den Stringausdruck. Entsprechend der Zahl d wird die entstandene Zeichenkette in verschiedene Strings aufgeteilt, die mit ihrem numerischen oder alphanumerischen Wert den Variablen der Variablenliste zugewiesen werden.

Bemerkung: Einer numerischen Variablen in der Variablenliste muß ein numerischer Wert als Ergebnis zugewiesen werden. Die Anzahl der Strings, in die die Zeichenkette aufgeteilt wird, und die durch den Delimiter d getrennt sind, muß größer oder gleich sein der Anzahl der Variablen in der Variablenliste.

Beispiel:

```
FILE     ASS

0010 DCL 30(A$,V$,Z$)
0020 DISP "BITTE VOR- UND ZUNAME EINGEBEN";
0030 RKB A$
0040 REM TRENNUNG DES NAMENS IN VOR- UND ZUNAME
0050 REM
0060 ASSIGN A$,V$,Z$;32
0070 PRINT "VORNAME: ";V$;TAB(35);"ZUNAME: ";Z$
0080 GOTO 20
0090 END

END OF LISTING


RUN
VORNAME: HELMUT                    ZUNAME: SCHMIDT
VORNAME: FRANZ-JOSEF               ZUNAME: STRAUSS
VORNAME: HELMUT                    ZUNAME: KOHL
VORNAME: FRIEDRICH                 ZUNAME: GENSCHER
```

BASSIGN

ANWEISUNG BASSIGN (Binary Assign)

Funktion : Dieser Befehl weist einer oder mehreren Variablen, in internem Format, den Inhalt eines String-Ausdruckes zu.

Format : BASSIGN Stringausdruck, {num. Var. / Stringvar.} [, {num. Var. / Stringvar.}] .....

Wirkung : Es wird der Stringausdruck berechnet. Die entstandene Zeichenkette wird als eine Liste von Daten in internem Format aufgefaßt. Die Daten werden den entsprechenden Variablen in der Variablenliste zugewiesen.

Bemerkung : Die Anzahl der Daten im "Stringausdruck" muß größer oder gleich sein der Anzahl der Variablen in der Variablenliste. Einer numerischen Variablen muß ein numerischer Wert zugewiesen werden.

Beispiel : FILE BASSIG

```
0010 REM BEISPIEL FUER BASSIGN/BBUILD
0020 REM
0030 DISP "EINGABE VON A UND B";
0040 INPUT A,B
0050 BBUILD A$,A,B
0060 PRINT "A=";A;",B=";B;TAB(40);"BBUILD: A$=";A$
0070 BASSIGN A$,X,Y
0080 PRINT "A$=";A$;TAB(40);"BASSIGN: X=";X;",Y=";Y
0090 PRINT
0100 GOTO 30
0110 END

END OF LISTING
```

```
A= 5 ,B= 5                          BBUILD: A$=■■■■■■■■■■■■■■■■
A$=■■■■■■■■■■■■■■■■                 BASSIGN: X= 5 ,Y= 5

A= 1.2345 ,B=-1.2345                BBUILD: A$=■■■■■■■■■■■■■■■■
A$=■■■■■■■■■■■■■■■■                 BASSIGN: X= 1.2345 ,Y=-1.2345

A= 100000 ,B= .00001                BBUILD: A$=■■■■■■■■■■■■■■■■
A$=■■■■■■■■■■■■■■■■                 BASSIGN: X= 100000 ,Y= .00001

A= 5.555 ,B= 6.666                  BBUILD: A$=■■■■■■■■■■■■■■■■
A$=■■■■■■■■■■■■■■■■                 BASSIGN: X= 5.555 ,Y= 6.666
```

BBUILD

ANWEISUNG BBUILD (Binary Build)

Funktion : Diese Anweisung überträgt in internem Format das Ergebnis von einer Liste mit arithmetischen- oder Stringausdrücken an eine Stringvariable.

Format : BBUILD Stringvar., {num. Ausdr. / String Ausdr.} [, {num. Ausdr. / String Ausdr.}] .....

Wirkung : Die angegebenen Ausdrücke werden berechnet.

Ihre Werte werden der String-Variablen in Binärdarstellung zugewiesen. Strings sind in internem Format dargestellt.

Bemerkung : Eine Zahl belegt in einfacher Genauigkeit 4 Byte (1 Wort), in doppelter hingegen 8 Byte. Jeder String mit N Zeichen benötigt INT ((N - 1) / 4 + 2)* 4 Byte.

Beispiel :

```
FILE     BBUILD

0010 REM BEISPIEL FUER BASSIGN/BBUILD
0020 REM
0030 DISP "EINGABE VON A UND B";
0040 INPUT A,B
0050 BBUILD A$,A,B
0060 PRINT "A=";A;",B=";B;TAB(40);"BBUILD: A$=";A$
0070 BASSIGN A$,X,Y
0080 PRINT "A$=";A$;TAB(40);"BASSIGN: X=";X;",Y=";Y
0090 PRINT
0100 GOTO 30
0110 END

END OF LISTING
```

```
A=-1 ,B=-3.25                          BBUILD: A$=■■■■■■■■■■■■■■■■■■■■■■■■
A$=■■■■■■■■■■■■■■■■■■■■■■■■           BASSIGN: X=-1 ,Y=-3.25

A=-.1 ,B= 10                           BBUILD: A$=■■■■■■■■■■■■■■■■■■■■■■■■
A$=■■■■■■■■■■■■■■■■■■■■■■■■           BASSIGN: X=-.1 ,Y= 10

A= 123456.79 ,B= 987654.32             BBUILD: A$=■■■■■■■■■■■■■■■■■■■■■■■■
A$=■■■■■■■■■■■■■■■■■■■■■■■■           BASSIGN: X= 123456.79 ,Y= 987654.32

A= 11111 ,B=-11111                     BBUILD: A$=■■■■■■■■■■■■■■■■■■■■■■■■
A$=■■■■■■■■■■■■■■■■■■■■■■■■           BASSIGN: X= 11111 ,Y=-11111
```

BEEP

Anweisung BEEP

Funktion : Die Anweisung bewirkt ein akustisches Signal.

Format : BEEP

Wirkung : Für die Dauer von ca. 0,2 Sekunden wird ein akustisches Signal gegeben.

Beispiel :

```
:
:
0050 DISP "ZAHL ZWISCHEN 1 UND 10 EINGEBEN";
0060 INPUT I
0070 IF (I>=1) AND (I<=10) THEN 100
0080 BEEP
0090 GOTO 50
:
:
:
```

ANWEISUNG BPAD (Binary Pad)

Funktion: Füllt eine Stringvariable mit binären Füllzeichen bis zur deklarierten Länge auf

Format: BPAD Stringvariable

Bemerkung: Das binäre Füllzeichen entspricht der Binärdarstellung von 255: 11111111

Für die Ausführung der Anweisung muß das System mit der Option STR initialisiert sein.

Beispiel:

```
FILE     BPAD

0010 DCL 30A$
0020 DISP "BITTE STRING EINGEBEN:";
0030 RKB A$
0040 PRINT "A$:";A$
0050 BPAD A$
0060 PRINT "A$:";A$
0070 GOTO 20
0080 END

END OF LISTING


A$:BEISPIEL PBAD
A$:BEISPIEL PBAD##########################
```

BUILD

ANWEISUNG BUILD

Funktion: Dieser Befehl überträgt, im Standard-Format, das Ergebnis einer Liste mit arithmetischen Ausdrücken und/ oder Stringausdrücken an eine Stringvariable.

Format:

BUILD Stringvariable , { Num. Ausdr. / Stringausdr. } [, { Num. Ausdr. / Stringausdr. } .....] [Delimiter d]

Wirkung: Es werden die Ausdrücke berechnet und die Ergebnisse in Zeichen-Format an die Stringvariable übertragen. Die Elemente werden durch das Zeichen getrennt, das in der ISO-Tabelle dem angegebenen Delimiter entspricht.

Bemerkung: Der Wert des arithmetischen Ausdruckes wird im Standardformat (siehe Abschnitt 8.4.) an die Stringvariable übergeben.
Ist das Resultat eines Ausdruckes eine Zeichenkette, so werden die einzelnen Zeichen ohne Veränderung an die Stringvariable übertragen.

Beispiel:

```
FILE

0010 REM *PROGRAMMBEISPIEL 'BUILD'*
0020 REM
0030 DCL 30A$
0040 DISP "A UND B EINGEBEN";
0050 INPUT A,B
0060 LET C=A↑B
0070 BUILD A$,A,"HOCH",B,"=",C;32
0080 PRINT A$
0090 GOTO 40
0100 END

END OF LISTING
```

```
 2  HOCH  3  =  8
 2.5  HOCH  4.55  =  64.658711
 10  HOCH  9  =  1.0000000E+09
```

BUILD USING

ANWEISUNG BUILD USING

Funktion: Dieser Befehl überträgt, im spezifizierten Format, das Ergebnis von einer Liste mit arithmetischen und/oder Stringausdrücken an eine Stringvariable.

Format:

BUILD USING { Zeilennr. / Stringvar. } , Stringvariable, { num. Ausdr. / Stringausdr. } [, { num. Ausdr. / Stringausdr. }] .....

Zeilennummer ist die Nummer derjenigen Programmzeile, die das Format einer Zeichenkette in der "Stringvariablen" festlegt. Anstelle der Zeilennummer kann der Name einer Stringvariablen stehen, die das Format definiert.

Wirkung: Es werden die Ausdrücke berechnet und die Resultate an die "Stringvariable" übertragen. Das Format ist jenes, das in der Programmzeile mit derjenigen Nummer steht, die in der Anweisung angegeben ist.

Steht in einer Anweisung anstelle der "Zeilennummer" der Name einer Stringvariablen, so werden die Ergebnisse in dem Format übertragen, das darin angegeben ist.

Bemerkung: Die Zeichen, die für die Formatspezifikation von Zeichenstrings verwendet werden dürfen, sind in der Beschreibung "Formatspezifikation" angeführt.

Beispiele:

Beispiel 1:
Formatangabe für BUILD USING in einer Image- Zeile

```
0010 REM *PROGRAMMBEISPIEL 'BUILD USING'*
0020 REM
0030 DCL 40A$
0040 DISP "A UND B EINGEBEN";
0050 INPUT A,B
0060 LET C=A↑B
0070 BUILD USING 100,A$,A,B,C
0080 PRINT A$
0090 GOTO 40
0100 :#####.### HOCH #####.### = #########.###
0110 END

END OF LISTING
```

```
    2.000 HOCH      3.000 =          8.000
    2.500 HOCH      4.550 =         64.659
   10.000 HOCH      9.000 = ************
```

Beispiel 2:
Aufbereitung des Formates in einer Stringvariablen.

```
FILE

0010 REM *PROGRAMMBEISPIEL 'BUILD USING'*
0020 REM
0030 DCL 40(A$,B$)
0040 DISP "A UND B EINGEBEN";
0050 INPUT A,B
0060 LET C=A↑B
0070 LET B$="#####.### HOCH #####.### = #########.###"
0080 BUILD USING B$,A$,A,B,C
0090 PRINT A$
0100 GOTO 40
0120 END

END OF LISTING
```

```
    2.000 HOCH      3.000 =          8.000
    2.500 HOCH      3.550 =         25.863
   10.000 HOCH      9.000 = ************
```

CHAIN

ANWEISUNG CHAIN

Funktion : Dieser Befehl unterbricht die Abarbeitung eines Programmes und beginnt mit der Ausführung eines anderen, ohne daß ein manueller Eingriff nötig ist.

Format : CHAIN { "Filename" / Stringvariable }

"Filename" ist der Name eines Programmes, das auf einer Diskette gespeichert ist.

Die Stringvariable enthält den Namen eines Programmes, das auf einer Diskette gespeichert ist.

Wirkung : Es wird die Abarbeitung des laufenden Programmes beendet. Alle in diesem Programm verwendeten externen Files werden geschlossen.

Bemerkung : Alle in dem durch CHAIN aufgerufenen Programm verwendeten externen Files müssen mit der FILES-Anweisung geöffnet werden, auch wenn in dem aufrufenden Programm gleiche Files verwendet werden. Da über die Files, deren Filedesignator 1, 2, 3 oder 4 ist, Informationen im Hauptspeicher zurückbehalten werden, wird die Ausführung von CHAIN schneller, wenn die von beiden Programmen verwendeten externen Files in der FILES-Anweisung an den Positionen 1-4 aufgeführt sind.

Beispiel:

```
FILE     HAUPT

0010 FILES SFILE
0020 PRINT "ANFANG HAUPTPROGRAMM (HAUPT)"
0030 SCRATCH :1
0040 WRITE :1,"HAUPT"
0050 CHAIN "UP1"
0060 END

END OF LISTING


FILE     UP1

0010 FILES SFILE
0020 PRINT "ANFANG UNTERPROGRAMM 1 (UP1)"
0030 DISP "WELCHES PROGRAMM SOLL RECHNEN";
0040 INPUT A$
0050 APPEND :1
0060 WRITE :1,"UP1"
0070 CHAIN A$
0080 END

END OF LISTING


FILE     UP2

0010 FILES SFILE
0020 PRINT "ANFANG UNTERPROGRAMM 2 (UP2)"
0030 PRINT "FOLGENDE PROGRAMME HABEN GERECHNET:"
0040 READ :1,A$ EOF 70
0050 PRINT A$
0060 GOTO 40
0070 PRINT "UP2"
0080 END

END OF LISTING


RUN HAUPT
ANFANG HAUPTPROGRAMM (HAUPT)
ANFANG UNTERPROGRAMM 1 (UP1)
WELCHES PROGRAMM SOLL RECHNEN?
UP2
ANFANG UNTERPROGRAMM 2 (UP2)
FOLGENDE PROGRAMME HABEN GERECHNET:
HAUPT
UP1
UP2
```

CONVERT

ANWEISUNG CONVERT

Funktion: Dieser Befehl wandelt eine Zeichenkette in die entsprechenden numerischen Codes der ISO-Tabelle um oder umgekehrt.

Format:
1) CONVERT Stringausdruck TO num. Vektor LENGTH num. Variable
2) CONVERT num. Vektor TO Stringvariable LENGTH arithmet. Ausdruck

Wirkung:

1. Format:
Es wird der Stringausdruck berechnet. Jedes Zeichen des sich dabei ergebenden Strings erhält den entsprechenden numerischen Code aus der ISO-Tabelle (siehe Anhang) und wird der Reihe nach den Elementen des Vektors aus der Anweisung zugewiesen. Die "Längen"-Variable enthält die Anzahl der umgewandelten Zeichen.

2. Format:
Es wird der arithmetische Ausdruck berechnet und das Ergebnis auf die nächste ganze Zahl n gerundet.
Die Werte der ersten n Elemente des Vektors aus der Anweisung werden ebenfalls auf die nächste ganze Zahl gerundet und dann in die entsprechenden Zeichen der ISO-Tabelle umgewandelt. Der Zeichenstring, der daraus entsteht, wird der entsprechenden Stringvariablen zugewiesen.

Beispiel :

```
FILE

0010 REM *PROGRAMMBEISPIEL FUER DIE ANWEISUNG 'CONVERT'*
0020 REM
0030 DCL 32A$
0040 DIM A(32)
0050 PRINT
0060 PRINT
0070 DISP "STRING EINGEBEN";
0080 RKB A$
0090 CONVERT A$ TO A LENGTH L
0100 PRINT "A$ = '";A$;"'"
0110 PRINT "LAENGE=";L
0120 PRINT "ISO-CODE:";
0130 FOR I=1 TO L STEP 1
0140 PRINT A(I);
0150 NEXT I
0160 PRINT
0170 PRINT
0180 DISP "LAENGE DES VEKTORS";
0190 INPUT N
0200 PRINT "LAENGE=";N
0210 PRINT "ISO-CODE:";
0220 FOR I=1 TO N STEP 1
0230 DISP "ELEMENT";I;
0240 INPUT A(I)
0250 PRINT A(I);
0260 NEXT I
0270 PRINT
0280 CONVERT A TO A$ LENGTH N
0290 PRINT "A$ = '";A$;"'"
0300 GOTO 50
0310 END

END OF LISTING
```

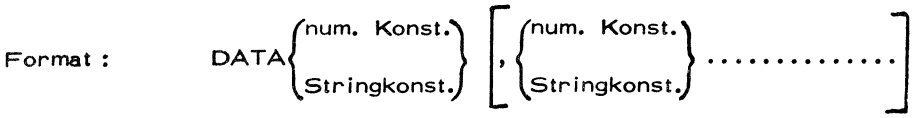
```
A$ = 'BEISPIEL *CONVERT*'
LAENGE= 18
ISO-CODE: 66  69  73  83  80  73  69  76  32  42  67  79  78  86  69  82  84  42


LAENGE= 8
ISO-CODE: 73  83  79  45  67  79  68  69
A$ = 'ISO-CODE'


A$ = 'OLIVETTI'
LAENGE= 8
ISO-CODE: 79  76  73  86  69  84  84  73

LAENGE= 5
ISO-CODE: 66  65  83  73  67
A$ = 'BASIC'
```

DATA

ANWEISUNG DATA

Funktion : Dieser Befehl erzeugt ein internes File von Daten, die dann den Variablen aus den READ- und MATREAD-Befehlen zugewiesen werden.

Format :



Wirkung : Zu Beginn der Ausführung des Programmes wird im Hauptspeicher eine Tabelle erzeugt, die alle Konstanten aller DATA-Anweisungen des Programmes enthält. Ein Pointer zeigt auf das erste Element der Tabelle. Die Konstanten der Tabelle werden den Variablen aus den READ- und MATREAD-Anweisungen zugewiesen, wobei der Pointer nach jeder Zuweisung auf das jeweils nächste Element der Tabelle zeigt. (Mit dem Befehl RESTORE kann der Pointer wieder auf das erste Element zurückgesetzt werden.)

Bemerkung : Stringkonstanten, die ein Komma (,) oder Leerzeichen am Anfang oder Ende enthalten, müssen in Anführungszeichen stehen.

Die DATA-Anweisungen können an jeder Stelle im Programm stehen. Jede String-Konstante muß einer String-Variablen zugewiesen werden. Jeder numerischen Variablen eines READ- oder MATREAD-Befehles muß eine bestimmte Zahl aus der Tabelle entsprechen.

Beispiel :

```
0010 PRINT
0020 PRINT "IHR TYP:"
0030 PRINT
0040 RESTORE
0050 FOR I=1 TO 5 STEP 1
0060 READ A$
0070 DISP A$;
0080 RKB B$
0090 PRINT A$;":",B$
0100 NEXT I
0110 GOTO 10
0120 DATA "ALTER","GROESSE","GEWICHT","HAARFARBE","BES. KENNZ."
0130 END
```

Beispiel :

```
IHR TYP:

ALTER:          25
GROESSE:        170
GEWICHT:        56
HAARFARBE:      blond
BES. KENNZ.:    Brillentraegerin

IHR TYP:

ALTER:          <60
GROESSE:        150
GEWICHT:        60-70
HAARFARBE:      grau
BES. KENNZ.:    kein
```

ANWEISUNG DCL (Declare)

Funktion: Mit dieser Anweisung wird festgelegt, welche numerischen Variablen mit einfacher Genauigkeit verarbeitet werden sollen und welche Längen für Strings max. vorgesehen sind.

Format:

DCL { { S(Variablenliste num.) / n(Variablenliste alphan.) } [, { S(Variablenliste num.) / n(Variablenliste alphan.) } .....] / SINGLE }

n ist eine ganze Zahl zwischen 1 und 1023; nach einem Feldnamen muß ein leeres Klammernpaar folgen. Beziehen sich S und n auf eine einzige numerische Variable oder Stringvariable, so können die Klammern wegfallen.

Wirkung: Die Werte der numerischen Variablen zwischen den Klammern hinter dem S werden im Speicher in einfacher Genauigkeit dargestellt, ebenso auch die Elemente von Feldern.
Wird bei einer DCL-Anweisung der Parameter SINGLE angegeben, so werden alle Werte der numerischen Variablen in einfacher Genauigkeit gerechnet. Dem Inhalt der Stringvariablen, auf die sich die Zahl n bezieht, werden im Hauptspeicher n Bytes zugewiesen, ebenso werden für jedes Feldelement eines Stringfeldes n Bytes im Hauptspeicher reserviert.

Bemerkung: Bezieht man sich in einem Programm mehrmals durch verschiedene DCL auf dieselbe Variable, so hat die Anweisung mit der höchsten Zeilennummer Gültigkeit. Bei mehreren Deklarationen innerhalb einer Anweisung ist die jeweils letzte gültig.

Beispiele:

```
30 DCL SINGLE

90 DCL 30 A$

20 DCL 30 (A$ , B$ , C$ (), D$ ), S (I, J, K)

10 DCL S (I, I1, I2), 20 (A$ , E3$ ), 10 (B$ ,D$ )
```

DEF

ANWEISUNG DEF

Funktion:

DEF ist eine nicht ausführbare Anweisung, die innerhalb einer Zeile eine numerische Funktion oder eine Stringfunktion definiert (Definition einer einzeiligen Funktionsprozedur).

Format:

DEF { FNα (Parameterliste) = Num. Ausdruck
    { FNα$ (Parameterliste) = Stringausdruck

α = A, .. Z, Parameterliste: Einfache num. oder alphanum. Variable

Für α muß ein Großbuchstabe des Alphabetes stehen.
FNα ist der Name einer numerischen Funktion, die als Ergebnis einen numerischen Wert liefert.
FNα$ ist der Name einer Stringfunktion, als Ergebnis wird ein String geliefert. Parameter können numerische Variable oder Stringvariable sein.
Die in einer DEF-Anweisung innerhalb der Klammer stehenden Parameter sind Schein-Variable, die keinen Bezug zu gleichnamigen Variablen außerhalb der Funktion haben. Beim Funktionsaufruf sind diese Parameter durch entsprechende Variable zu ersetzen, wobei der Scheinvariablen der aktuelle Wert der an entsprechender Stelle im Aufruf stehenden Variablen zugewiesen wird. Die Scheinvariablen nennt man formale Parameter.

Auf der rechten Seite der Anweisung können außer den Parameternamen auch noch andere Variable, sogenannte globale Variable, stehen, die aber schon vor dem Funktionsaufruf einen Wert besitzen müssen.

Wirkung:

Die Funktion FNα oder FNα$ wird durch den Ausdruck auf der rechten Seite definiert. Die Funktion wird in einem Programm so oft ausgeführt, wie ihr Name FNα oder FNα$ zusammen mit den aktuellen Parametern vorkommt.

Bemerkung:

Eine Funktion kann an jeder Stelle im Programm stehen, darf jedoch nur ein einziges Mal definiert werden und in dieser Form nur eine Zeile umfassen.
Auch direkte rekursive Aufruffolgen (Bsp. 10 DEF FNA = FNA) oder
indirekte rekursive Aufruffolgen (Bsp. 10 DEFFNA = X+FNB
50 DEF FNB = FNA + Y)
sind nicht erlaubt.
Es können maximal 15 Parameter verwendet werden. Formale Parameter haben keinerlei Beziehung zu Variablen mit den gleichen Namen.
Tritt in der Abarbeitung eines Programmes die Anweisung DEF auf, so wird die Abarbeitung in der folgenden Programmzeile fortgesetzt.
Die aktuellen Parameter müssen in Typ und Anzahl mit den formalen übereinstimmen. In einem Programm dürfen maximal 26 Funktionen definiert werden.
Für die Ausführung der Funktion FNα$ muß das System mit der Option STR initialisiert sein.

Beispiel 1: Einzeilige numerische Funktionen

```
FILE


0010 DEF FNB(X)=PI/180*X
0020 DEF FNC(X)=COS(FNB(X))
0030 DEF FNS(X)=SIN(FNB(X))
0040 DEF FNT(X)=TAN(FNB(X))
0050 DEF FNA(X)=180*ATN(X)/PI
0060 PRINT "  X         SIN         COS         TAN         ATN"
0070 PRINT
0080 DISP "VON GRAD?,BIS GRAD?,SCHRITTWEITE";
0090 INPUT A,B,C
0100 FOR I=A TO B STEP C
0110 PRINT USING 150,I,FNS(I),FNC(I),FNT(I),FNA(I)
0120 NEXT I
0130 PRINT
0140 GOTO 80
0150 :###  ##.######  ##.######  ##.######  ####.####
0160 END

END OF LISTING
```

X	SIN	COS	TAN	ATN
1	0.017452	0.999848	0.017455	45.0000
2	0.034899	0.999391	0.034921	63.4349
3	0.052336	0.998630	0.052408	71.5651
4	0.069756	0.997564	0.069927	75.9638
5	0.087156	0.996195	0.087489	78.6901
40	0.642788	0.766044	0.839100	88.5679
45	0.707107	0.707107	1.000000	88.7270
50	0.766044	0.642788	1.191754	88.8542

Beispiel 2: Einzeilige alphanumerische Funktion

```
FILE


0010 DEF FNA$(A$,A,B)=EXT$(A$,A,B)
0020 DISP "STRING",
0030 RKB X$
0040 DISP "VON ... BIS";
0050 INPUT X,Y
0060 PRINT X$;TAB(25);"EXT$(";X$;",";X;",";Y;")=";FNA$(X$,X,Y)
0070 GOTO 20
0080 END

END OF LISTING


P6060 BASIC              EXT$(P6060 BASIC, 7 , 11 )=BASIC
OLIVETTI                 EXT$(OLIVETTI, 1 , 5 )=OLIVE
OLIVETTI P6060           EXT$(OLIVETTI P6060, 1 , 14 )=OLIVETTI P6060
```

DEF/FNEND

ANWEISUNG DEF/FNEND (Define/ Funktion End)

Funktion:

Es ist eine nicht ausführbare Anweisung, die sich auch über mehrere Zeilen erstrecken darf und eine numerische Funktion oder eine Stringfunktion definiert (Definition von Funktionsunterprogrammen).

Format:

```
DEF { FN[x]  [(Parameterliste)]   [Liste d. lokalen Variablen] }
    { FN[x]$ [(Parameterliste)]   [Liste d. lokalen Variablen] }
      .
      .
      .
    { FN*  = Num. Ausdruck   }   mindestens eine Zuweisung
    { FN*$ = Stringausdruck  }
      .
      .
      .
    FNEND
```

FN[x] ist der Name einer numerischen Funktion; dem Funktionsnamen wird der errechnete Wert zugewiesen.
FN[x]$ ist der Name einer Stringfunktion; dem Funktionsnamen wird als Ergebnis ein Zeichenstring zugewiesen.

Parameter können numerische Variable oder Stringvariable sein.
Lokale Variable sind nur im Funktionsunterprogramm verwendete Variable, die sowohl numerisch, als auch alphanumerisch sein können.

FN* oder FN*$ sind Pseudovariable, denen vor dem Rücksprung aus der Funktion FN[x] oder FN[x]$ der jeweilige Funktionswert zugewiesen wird.

In arithmetischen Ausdrücken oder Stringausdrücken können außer den Parametern und lokalen Variablen auch noch sogenannte globale Variable auftreten, die aber schon vor dem Funktionsaufruf einen Wert besitzen müssen.

Zwischen den Programmzeilen DEF und FNEND können außer den Anweisungen für die Berechnung der Werte der Pseudovariablen (FN* und FN*$) beliebige BASIC-Anweisungen stehen.

Wirkung:

Die Funktionen FNα und FN$\alpha\$$ werden durch alle Anweisungen zwischen den Programmzeilen FNα oder FN$\alpha\$$ und der Zeile mit FNEND definiert.
Die Funktion wird in einem Programm so oft ausgeführt, wie ihr Name FNα oder FN$\alpha\$$ zusammen mit den aktuellen Parametern, die in runden Klammern eingeschlossen sind, vorkommt.

Bemerkung:

Eine Funktion, die sich über mehrere Zeilen erstreckt, kann an jeder Stelle des Programmes definiert werden; direkte oder indirekte rekursive Funktionsaufrufe sind nicht zulässig.
Durch die Anweisungen GO TO, ON ... GO TO, GO SUB, ON ... GO SUB, IF ... THEN darf nicht aus der Funktion hinausgesprungen werden. Auch eine STOP-Anweisung im Inneren einer Funktion ist verboten.

Mit den Anweisungen GO TO, ON...GO TO, GO SUB, ON...GO SUB, IF...THEN darf nicht von außen in eine Funktion gesprungen werden.

Es ist mit der Anweisung DCL möglich auch innerhalb einer Funktion die Länge der Pseudo-Stringvariablen FN*$\$$ festzulegen.

Lokale Parameter arbeiten mit doppelter Genauigkeit.

Der Wert einer globalen Größe darf innerhalb einer Funktion verändert werden, wenn die Funktion eine Zuweisung an die globale Variable enthält.
Im Inneren einer Funktion sind Wertzuweisungen an Parameter gestattet, der Wert des entsprechenden Argumentes bleibt jedoch unverändert.
Die aktuellen Parameter müssen im Typ und in der Anzahl den formalen entsprechen. Tritt in der Abarbeitung eines Programmes die Anweisung DEF auf, so wird die Abarbeitung in der Programmzeile nach dem FNEND fortgesetzt.

Parameternamen haben keinerlei Beziehung zu Variablen mit den gleichen Namen, die im Programm vorkommen.

Lokalen Größen muß innerhalb der Funktion ein Wert zugewiesen werden. Namen von lokalen Größen haben keinerlei Beziehung zu etwaigen gleichlautenden Namen von Variablen im Programm.
Insgesamt dürfen maximal 15 Parameter und lokale Größen verwendet werden.
In einem Programm dürfen maximal 26 Funktionen vorkommen.
Für die Ausführung der Funktion FN$\alpha\$$ muß das System mit der Option STR initialisiert sein.

Beispiel 1 : Mehrzeilige numerische Funktion.

```
FILE


0010 DEF FNA(I,J)L
0020 LET L=I-INT(I/J)*J
0030 IF L=0 THEN 70
0040 LET I=J
0050 LET J=L
0060 GOTO 20
0070 LET FN*=J
0080 FNEND
0090 DISP "EINGABE A,B";
0100 INPUT A,B
0110 PRINT "DER GROESSTE GEMEINSAME TEILER VON";A;"UND";B;"IST";FNA(A,B)
0120 GOTO 90
0130 END

END OF LISTING


DER GROESSTE GEMEINSAME TEILER VON 85 UND 63 IST 1
DER GROESSTE GEMEINSAME TEILER VON 125 UND 35 IST 5
DER GROESSTE GEMEINSAME TEILER VON 95478 UND 24 IST 6
```

Beispiel 2 : Mehrzeilige alphanumerische Funktion.

```
FILE


0010 DEF FNA$(X$)I,A$
0020 LET A$=""
0030 FOR I=LEN(X$) TO 1 STEP -1
0040 LET A$=A$+EXT$(X$,I,I)
0050 NEXT I
0060 LET FN*$=A$
0070 FNEND
0080 DISP "EINGABE STRING";
0090 RKB A$
0100 PRINT USING 110,A$,FNA$(A$)
0110 : 'RRRRRRRRRRRRRR I 'LLLLLLLLLLLLLL
0120 GOTO 80
0130 END

END OF LISTING


   EIN I NIE
 NEGER I REGEN
   MIT I TIM
GAZELLE I ELLEZAG
  ZAGT I TGAZ
    IM I MI
 REGEN I NEGER
   NIE I EIN
```

DELAY

ANWEISUNG DELAY

Funktion:

Diese Anweisung bewirkt eine Pause vor der Ausführung des nächsten Programmschrittes.

Format:

DELAY n

n .. Zehntelsekunden

n ist eine positive ganze Zahl mit maximal 4 Stellen.

Wirkung:

Der folgende Programmschritt wird nach n Zehntelsekunden ausgeführt.

Bemerkung:

Mit der Taste CONTINUE kann die Wirkung von DELAY aufgehoben werden.

Beispiele:

```
          .
          .
          .
0100 DISP "EINGABE DER MATRIX A"
0110 DELAY 20
0120 DISP "ANZAHL ZEILEN,ANZAHL SPALTEN";
0130 INPUT N,M
          .
          .
          .
```

DEPAD

ANWEISUNG DEPAD

Funktion: Die Füllzeichen am Ende einer Stringvariablen werden entfernt.

Format:

DEPAD Stringvariable, n

n ist eine ganze Zahl zwischen Ø und 255

Wirkung:

Aus Stringvariablen werden die Zeichen entfernt, die in der ISO-Tabelle der Zahl n entsprechen. Die Zeichen werden rechts beginnend entfernt, solange, bis das erste Zeichen auftritt, das nicht den ISO-Code n hat.

Bemerkung:

Für die Ausführung der Anweisung muß das System mit der Option STR initialisiert sein.

Beispiel:

```
FILE

0010 REM *PROGRAMMBEISPIEL FUER 'PAD' UND 'DEPAD'*
0020 REM
0030 DCL 32A$
0040 PRINT
0050 PRINT
0060 DISP "STRING EINGEBEN";
0070 RKB A$
0080 PRINT "EINGABE: '";A$;"'"
0090 PAD A$,42
0100 PRINT "A$ NACH 'PAD':   '";A$;"'"
0110 DEPAD A$,42
0120 PRINT "A$ NACH 'DEPAD': '";A$;"'"
0130 GOTO 40
0140 END

END OF LISTING



EINGABE: 'BEISPIEL PAD-DEPAD'
A$ NACH 'PAD':   'BEISPIEL PAD-DEPAD**************'
A$ NACH 'DEPAD': 'BEISPIEL PAD-DEPAD'


EINGABE: '*** OLIVETTI ***'
A$ NACH 'PAD':   '*** OLIVETTI ******************'
A$ NACH 'DEPAD': '*** OLIVETTI '


EINGABE: '*** P6060 *** BASIC *** '
A$ NACH 'PAD':   '*** P6060 *** BASIC *** ********'
A$ NACH 'DEPAD': '*** P6060 *** BASIC *** '
```

DIM

ANWEISUNG DIM (Dimension)

Funktion : Die Anweisung legt für ein oder mehrere Felder die Dimension fest.

Format : DIM Feldname (Zeilen [, Spalten]) [, Feldname (Zeilen [, Spalten])]...

Wirkung : Folgt einem Feldnamen nur eine Zahl r zwischen Klammern, so handelt es sich um einen Vektor mit r Elementen. Die Feldindices müssen kleiner gleich r sein; folgt einem Feldnamen ein Zahlenpaar zwischen Klammern (r, c), so handelt es sich um eine Matrix mit r Zeilen und c Spalten. Die Indices dürfen die jeweiligen Feldgrenzen nicht überschreiten ($\leq$ r und $\leq$ c). Die Indices einer Feldvereinbarung müssen größer ø (Null) sein.

Bemerkung : Wird ein eindimensionales Feld nicht durch DIM deklariert, so werden ihm vom System 10 Elemente zugewiesen.

Wird ein 2-dimensionales Feld nicht durch DIM deklariert, so werden ihm vom System 10 Zeilen und 10 Spalten bei num. Feldern und 5 Zeilen und 5 Spalten bei alphanumerischen Feldern zugewiesen.

Der vom Compiler zugelassene höchste Wert für eine Dimension beträgt 65536, bei zwei Dimensionen muß das Produkt kleiner als 65536 sein. Die tatsächlich höchstzulässigen Werte für die Dimensionen hängen von der Größe des Hauptspeichers und seiner Belegung durch das Programm ab.

Die Anweisung DIM kann an jeder Stelle des Programmes stehen.

Es können auch mehrere DIM-Anweisungen in einem Programm vorkommen.

Ein eindimensionales Feld darf nicht den gleichen Namen wie ein zweidimensionales haben.

Jedes Feldelement wird anfangs mit dem Wert "nicht definiert" belegt.

Beispiel :

```
FILE

0010 REM *BEISPIEL 'DIM'*
0020 REM
0030 DCL SA()
0040 DIM A(2,3)
0050 FOR I=1 TO 2 STEP 1
0060 FOR J=1 TO 3 STEP 1
0070 LET A(I,J)=100*RND
0080 NEXT J
0090 NEXT I
0100 DISP "WELCHES ELEMENT";
0110 INPUT I,J
0120 PRINT "A(";I;",";J;")=";A(I,J)
0130 GOTO 100
0140 END

END OF LISTING


A( 1 , 1 )= 63.8293
A( 2 , 2 )= 21.5078
A( 1 , 3 )= 97.356
A( 1.5 , 2.1 )= 21.5078
```

DISP

ANWEISUNG DISP (Display)

Funktion : Die Anweisung dient zur Ausgabe von num. und/oder alphanum. Daten im Display.

Format :

DISP [{ num. Ausdr. / Stringausdr. / TAB (num. Ausdr.) } [{ ; / , } { num. Ausdr. / Stringausdr. / TAB (num. Ausdr.) }] .... [; / ,]]

"," und ";" sind Trennzeichen mit bestimmter Bedeutung für die Ausgabe. (siehe Abschnitt 8.4).

Wirkung : Die Ergebnisse der Ausdrücke werden im Standardformat dargestellt und im Display sichtbar gemacht.

Die Position der Ausdrücke in einer Zeile hängt sowohl ab von der erforderlichen Länge der Darstellung als auch von den verwendeten Trennzeichen (Komma oder Strichpunkt) und der Funktion TAB.

Stellenkontrolle der Zeichen im Display

Die Ergebnisse eines Ausdruckes werden durch den Befehl DISP in einem Register, dem Puffer des Display (siehe Abb. 1), in Form einer Folge von Zeichen dargestellt und im Display angezeigt.

Ergibt die Darstellung der Liste der Ausdrücke im Standardformat mehr als 32 Zeichen, so können die überzähligen Zeichen nicht angezeigt werden.

Bei genügend langem DELAY ist es aber möglich, mit Hilfe der Tasten →
bzw. auch ← und ihren Kombinationen mit SHIFT und REPEAT den gesamten Inhalt des Display-Puffers (die ersten 80 Zeichen der Ausgabe) zu lesen. Bei gedrückter PRINT-ALL-Taste wird in jedem Fall die gesamte Zeile gedruckt.

Bemerkung : Die Anweisung DISP ohne Angabe von Parametern löscht den Inhalt des Puffers und setzt den Pointer an die erste Stelle. Ebenso werden alle Zeichen im Display gelöscht.

Ist der Puffer des Display voll, so erscheinen im Display die ersten 32 Zeichen. Wird ein weiteres Zeichen erzeugt, so werden alle Zeichen vom Display gelöscht und das generierte Zeichen kommt an die erste Stelle im Puffer und erscheint als erstes Zeichen im Display.

Das Standardformat ist in Abschnitt 8.4 am Ende dieses Kapitels genau beschrieben.

Beispiel : FILE Die PRINT-ALL-Taste ist aktiviert.

```
0010 DISP "WERTE VON X UND Y";
0020 INPUT X,Y
0030 PRINT X,Y,X↑Y
0040 GOTO 10
0050 END

END OF LISTING


WERTE VON X UND Y?
3,5
 3                 5                 243
WERTE VON X UND Y?
2,32
 2                 32                4.2949673E+09
WERTE VON X UND Y?
```

DISP USING

ANWEISUNG DISP USING

Funktion : Mit dieser Anweisung können Zahlen und Strings auf dem Display sichtbar gemacht werden, die Formatspezifikation ist vom Benutzer frei wählbar.

Format : DISP USING { Zeilennr. / Stringvar. } , { num. Ausdr. / Stringausdr. } [, { num. Ausdr. / Stringausdr. }] ...

"Zeilennummer" bezieht sich auf die Programmzeile mit der Formatspezifikation (siehe Anweisung IMAGE).

"Stringvariable" enthält eine Formatspezifikation.

Wirkung : Die Ergebnisse der Ausdrücke werden der Reihe nach in dem Format auf dem Display ausgegeben, das durch die Formatanweisung spezifiziert wurde oder durch die Stringvariable festgelegt wird.

Jede Größe, die auf dem Display dargestellt wird, wird von links nach rechts, gemäß dem jeweiligen Abbildungszeichen des Formatfeldes, ausgegeben.

Bemerkung : Jede Ausgabe auf dem Display durch die Anweisung DISP USING erfolgt ab der ersten Stelle, auch wenn eine vorhergehende DISP-Anweisung mit "," oder ";" endete. Es dürfen nicht mehr Größen ausgegeben werden, als durch die Formatspezifikation angegeben wird.

Sind es weniger, so haben die restlichen Formatelemente keine Wirkung.

Die Ausgabeelemente der DISP USING-Anweisung müssen dem spezifizierten Format entsprechen.

Beispiel 1 :

```
0010 REM *BEISPIEL 'DISP USING'*
0020 REM
0030 DCL 30A$
0040 LET A$="I=####.###   J=####.###"
0060 DISP "I,J";
0070 INPUT I,J
0080 DISP USING A$,I,J
0090 DELAY 20
0100 GOTO 60
0120 END

END OF LISTING
```

Die PRINT-ALL-Taste ist aktiviert.

```
RUN
I,J?
1,2
I=    1.000    J=    2.000
I,J?
1.5,2.365
I=    1.500    J=    2.365
I,J?
3.1415,99654.871
I=    3.142    J=********
I,J?
```

Beispiel 2 :

```
LIST
FILE

0010 REM *BEISPIEL 'DISP USING'*
0020 REM
0030 DISP "Ihr Name";
0040 RKB A$
0050 DISP USING 80,A$
0060 DELAY 50
0070 GOTO 30
0080 : ****** 'CCCCCCCCCCCCCC ******
0090 END

END OF LISTING
```

Die PRINT-ALL-Taste ist aktiviert.

```
RUN
Ihr Name?
Micky Mouse
 ******   Micky Mouse    ******
Ihr Name?
Dracula
 ******     Dracula      ******
```

END

ANWEISUNG END

Funktion : Die Anweisung gibt das physische Ende eines Programmes an.

Format : END

Wirkung : Die Ausführung eines Programmes wird beendet. Die Werte von Variablen sind nicht mehr definiert und externe Files werden geschlossen.

Bemerkung : Die END-Anweisung muß in jedem Programm genau einmal vorkommen. Das Programm verbleibt nach der END-Anweisung weiterhin im Arbeitsspeicher.

Die END-Anweisung hat die höchste Nummer im Programm.

Bei der Programmerstellung mit automatischer Zeilennumerierung ist nach der Eingabe von END die Numerierung mit SHIFT und CLEAR zu unterbrechen.

Beispiel :

```
0500 REM ***     DIE LETZTE ANWEISUNG    ***
0510 REM *** IN EINEM BASIC - PROGRAMM ***
0520 REM ***           IST IMMER :       ***
0530 REM
0540 REM                 "END"
0550 REM
0560 END
```

FILES

ANWEISUNG FILES

Funktion:

Diese Anweisung legt die Namen und die Anzahl der Files auf den Floppy Disks fest, auf die ein Programm zugreift .

Format:

FILES { Filename / * } [; { Filename / * }] . . . .

Wirkung:

Alle Files, die in der Anweisung mit ihrem Namen angegeben sind, werden geöffnet. Ein sequentielles File wird hierbei in den Lesemodus versetzt.

Jedem Filenamen wird eine Zahl (Designator) zugewiesen, die der Reihenfolge in der Liste entspricht: der erste Filename bekommt die Zahl 1, dem zweiten wird die Zahl 2 zugewiesen usw.
Wird statt eines Filenamens ein * gesetzt, so wird der Platz und Filedesignator reserviert für ein File, das in einer nachfolgenden FILE: Anweisung geöffnet wird.

Bemerkung:

Bei jeder Operation, die sich auf ein File bezieht, muß als 1. Operand der Filedesignator angegeben werden.
Die Anweisung FILES darf in einem Programm nur einmal vorkommen.
Jedes File kann mit einer weiteren FILE: Anweisung durch ein anderes File ersetzt werden (siehe Anweisung FILE:).
Die Anweisung FILES muß jeder FILE: Anweisung vorausgehen; die FILES Anweisung muß auch jeder Anweisung vom Typ: SETW:, READ:, WRITE:, RESTORE:, SCRATCH:, APPEND:, MAT READ:, MAT WRITE: und WHERE: vorausgehen.
Ein Filename muß mit einem Buchstaben bzw. mit dem Bibliothekskennzeichen (+ oder *) beginnen.
Folgende Zeichen sind nicht zulässig: Schrägstrich (/), Komma (,), Strichpunkt (;), Leerstelle (␣), Doppelpunkt (:).

Beispiel:

```
FILE     FILES

0010 REM *** BEISPIEL FUER 'FILES' UND 'FILE' ***
0020 REM
0030 REM *** OEFFNEN VON 5 FILES ***
0040 REM
0050 FILES SDAT;TDAT;UDAT;VDAT;WDAT
0060 REM
0070 REM *** OEFFNEN FUER SCHREIBEN ***
0080 REM
0090 FOR I=1 TO 5 STEP 1
0100 SCRATCH :I
0110 READ A$(I)
0120 REM
0130 REM *** BESCHREIBEN DER FILES ***
0140 REM
0150 WRITE :I,"DAS IST FILE ",A$(I)
0160 REM
0170 REM *** SCHLIESSEN DER FILES ***
0180 REM
0190 FILE : I,*
0200 NEXT I
0210 DATA SDAT,TDAT,UDAT,VDAT,WDAT
0220 DISP "WELCHES FILE";
0230 INPUT I
0240 REM
0250 REM *** OEFFNEN DES ANGEGEBENEN FILES ***
0260 REM
0270 FILE : 1,A$(I)
0280 REM
0290 REM *** OEFFNEN FUER LESEN ***
0300 REM
0310 RESTORE :1
0320 REM
0330 REM *** LESEN VOM FILE ***
0340 REM
0350 READ :1,A$,B$
0360 PRINT "FILE";I;": ";A$;B$
0370 GOTO 220
0380 END

END OF LISTING
```

```
FILE 3 : DAS IST FILE UDAT
FILE 5 : DAS IST FILE WDAT
FILE 1 : DAS IST FILE SDAT
FILE 4 : DAS IST FILE VDAT
FILE 2 : DAS IST FILE TDAT
```

FILE:

ANWEISUNG FILE:

Funktion:

Diese Anweisung ermöglicht den Zugriff auf ein File, dessen Name nicht in einer FILES-Anweisung spezifiziert wurde und das Schließen von Files vor dem Programmende.

Format:

FILE: arithm. Ausdruck, { Stringausdruck / * }

Der arithmetische Ausdruck wird berechnet und das Ergebnis wird gerundet. Die so erhaltene ganze Zahl n ist der Filedesignator des Filenamens in der Anweisung.

Wirkung:

Der Ergebnisstring des Stringausdrucks muß ein Filename sein. Das so durch seinen Namen angegebene File ersetzt das File, das bisher den Filedesignator n hatte.

Dieses File wird geschlossen und an seiner Stelle wird das File mit dem angegebenen Filenamen unter dem gleichen Filedesignator geöffnet.

Wird statt des Filenamens ein * angegeben, so wird das entsprechende File mit dem Filedesignator n geschlossen, ohne daß ein anderes File geöffnet wird.

Bemerkung:

Der Filename in der Anweisung muß verschieden sein von allen Files, die geöffnet sind.

Der Filedesignator n muß größer als ∅ sein und darf höchstens gleich der Anzahl der Files in der FILES-Anweisung sein.

FKEY#

ANWEISUNG FKEY# (Funktion Key)

Funktion: Mit Hilfe dieser Anweisung kann einer Funktionstaste ein Inhalt zugewiesen werden.

Format: FKEY# n, Stringkonstante [:]

n ist ganze Zahl zwischen 1 und 16

Wirkung: Einer Taste, die durch den Wert des Identifikators n bestimmt ist, wird der Inhalt der Stringkonstanten zugewiesen.

Bemerkung: So oft die Funktionstaste gedrückt wird, wird die der Taste zugewiesene Zeichenfolge in den Tastaturpuffer übertragen. Folgt der Stringkonstanten ein Doppelpunkt, so wird der Inhalt des Tastaturpuffers unmittelbar (d.h. ohne EOL) an das System übermittelt. Den 16 Funktionstasten können insgesamt maximal 238 Zeichen zugewiesen werden.

Der Inhalt der Funktionstaste bleibt erhalten, bis er durch eine neue Anweisung FKEY mit gleichem n überschrieben wird oder durch den Befehl LDKEYS oder eine Neuinitialisierung des Systems wieder durch den Standardinhalt ersetzt wird. Wird STKEYS eingegeben, wird der so definierte Inhalt zum Standardinhalt.

Die Übermittlung des Inhaltes von Funktionstasten an Variable eines Programmes kann nur mit den Anweisungen INPUT oder RKB erfolgen.

Die Funktionstastenbelegung kann nicht nur über Programm, sondern auch im CALCULATOR-MODE und im DEBUGGING-MODE erfolgen.

Beispiel :

```
FILE     FKEY


0010 DCL 80A$
0020 FKEY #1,DIES
0030 FKEY #2,IST
0040 FKEY #3,EIN
0050 FKEY #4,TEST
0060 FKEY #5,BEISPIEL
0070 FKEY #6,1.11111:
0080 FKEY #7,2.22222:
0090 FKEY #8,ENDE:
0100 DISP "EINGABE STRING (F8=ENDE)";
0110 RKB A$
0120 PRINT "STRING: ";A$
0130 IF A$="ENDE" THEN 190
0140 DISP "EINGABE NUMERISCHER WERT";
0150 INPUT A
0160 PRINT "NUMERISCHER WERT";A
0170 PRINT
0180 GOTO 100
0190 FKEY #1,RUN FKEY:
0200 PRINT
0210 PRINT
0220 PRINT "*** DRUECKEN SIE DIE TASTE F1 ZUM ERNEUTEN START DES PROGRMMES ***"
0230 PRINT
0240 PRINT
0250 PRINT
0260 END

END OF LISTING


STRING: DIES IST EIN TEST BEISPIEL
NUMERISCHER WERT 2.22222

STRING: EIN BEISPIEL IST DIES
NUMERISCHER WERT 1.11111

STRING: 1.11111
NUMERISCHER WERT 1.11111

STRING: ENDE


*** DRUECKEN SIE DIE TASTE F1 ZUM ERNEUTEN START DES PROGRMMES ***


STRING: TEST BEISPIEL
NUMERISCHER WERT 2.22222

STRING: ENDE


*** DRUECKEN SIE DIE TASTE F1 ZUM ERNEUTEN START DES PROGRMMES ***
```

FNEND

ANWEISUNG FNEND (Function End)

Funktion:

Kennzeichnet das Ende einer mehrzeiligen Funktionsdefinition

Format:

FNEND

Wirkung:

siehe DEF/FNEND

Bemerkung:

Jede mehrzeilige Funktionsdefinition muß mit der Anweisung FNEND enden.

FOR

ANWEISUNG FOR

Funktion:

Kennzeichnet den Beginn einer Schleife

Format:

FOR Laufvariable = Num. Ausdruck TO num. Ausdruck [STEP num. Ausdruck.]

.

.

.

beliebige BASIC-Instruktionen

.

.

NEXT Laufvariable

Laufvariable ist eine einfache numerische Variable; die arithmetischen Ausdrücke in der Laufanweisung geben den Anfangswert der Laufvariablen, den Endwert und die Schrittweite an, um die sich die Laufvariable nach jedem Durchlauf der Schleife erhöht. Nach Beendigung der Schleife durch NEXT ist der Wert der Laufvariablen = letzter erreichter Wert + Schrittweite.

Wirkung:

Die Folge von Anweisungen zwischen FOR und NEXT wird solange ausgeführt, bis der Wert der Laufvariablen den angegebenen Endwert übersteigt. Ist der Anfangswert der Laufvariablen größer (kleiner, wenn eine negative Schrittweite angegeben wird) als der Endwert, wird die Schleife nicht ausgeführt, der Wert der Laufvariablen bleibt unverändert und das Programm wird mit dem 1. Befehl nach NEXT fortgesetzt. Ist der Anfangswert kleiner (größer bei negativer Schrittweite) als der Endwert, so wird die Schleife durchlaufen, wobei bei jeder NEXT-Anweisung der Wert der Laufvariablen um die Schrittweite erhöht wird. Ist der neue Wert der Laufvariablen kleiner oder gleich (bei negativer Schrittweite größer oder gleich) als der Endwert, wird die Schleife von neuem durchlaufen und zwar solange, wie der Wert der Laufvariablen den Endwert nicht überschreitet (bzw. unterschreitet bei negativer Schrittweite).
Nach Beendigung der Schleife fährt das Programm mit dem auf NEXT folgenden Programmschritt fort.

Bemerkung:

Fehlt die Angabe der Schrittweite, so wird diese implizit als 1 angenommen. Zwei oder mehrere Schleifen können geschachtelt werden, sie dürfen sich jedoch nicht überschneiden.

richtig:	falsch:
FOR A = 1 TO 10	FOR A = 1 TO 10
FOR B = 1 TO 5	FOR B = 1 TO 5
NEXT B	NEXT A
NEXT A	NEXT B

Ist die Schrittweite Null, so hat man eine Endlos-Schleife, wenn der Wert der Laufvariablen nicht in der Schleife verändert wird.

Eine Schleife muß immer mit FOR beginnen.

Mit den Anweisungen:

GO TO

ON ... GO TO

IF ... THEN

(wobei die jeweilige Zeilennummer größer ist als die der Anweisung NEXT) kann aus einer Schleife gesprungen werden.

Bei einem Sprung aus der Schleife behält die Laufvariable ihren letzten Wert bei, es kann jedoch nicht zurück in die Schleife gesprungen werden.

Folgende Elemente können auch noch Bestandteil einer Schleife sein:

GO SUB

ON ... GO SUB

Jeder Anweisung FOR muß ein NEXT entsprechen. Sind mehrere Schleifen geschachtelt, so müssen sie verschiedene Laufvariable haben.

Der Wert einer Laufvariablen darf innerhalb der Schleife verändert werden, nicht jedoch der Endwert und die Schrittweite.

Die Anweisung NEXT bewirkt eine Fehlermeldung, wenn nicht zuerst die Anweisung FOR ausgeführt wurde.

Es können bis zu 15 Schleifen geschachtelt werden.

Beispiel :

```
FILE

0010 REM *** PROGRAMMBEISPIEL FOR / NEXT ***
0020 REM
0030 DISP "1.SCHLEIFE:ANFANGSWERT,ENDWERT,SCHRITTWEITE";
0040 INPUT A1,A2,A3
0050 DISP "2.SCHLEIFE:ANFANGSWERT,ENDWERT,SCHRITTWEITE";
0060 INPUT B1,B2,B3
0070 PRINT ,"ANFANGSWERT","ENDWERT","SCHRITTWEITE"
0080 PRINT
0090 PRINT "I-SCHLEIFE",A1,A2,A3
0100 PRINT "J-SCHLEIFE",B1,B2,B3
0110 PRINT
0120 FOR I=A1 TO A2 STEP A3
0130 PRINT "I-SCHLEIFE: I=";I,"J-SCHLEIFE: ";
0140 FOR J=B1 TO B2 STEP B3
0150 PRINT "J=";J;
0160 NEXT J
0170 PRINT
0180 NEXT I
0190 PRINT
0200 PRINT "I=";I;TAB(15);"(LETZTER WERT(";I-A3;") + SCHRITTWEITE(";A3;"))"
0210 PRINT "J=";J;TAB(15);"(LETZTER WERT(";J-B3;") + SCHRITTWEITE(";B3;"))"
0220 PRINT
0230 PRINT
0240 GOTO 30
0250 END

END OF LISTING
```

```
                ANFANGSWERT     ENDWERT         SCHRITTWEITE

I-SCHLEIFE        1               3               1
J-SCHLEIFE        1               1.5             .2

I-SCHLEIFE: I= 1                J-SCHLEIFE: J= 1 J= 1.2 J= 1.4
I-SCHLEIFE: I= 2                J-SCHLEIFE: J= 1 J= 1.2 J= 1.4
I-SCHLEIFE: I= 3                J-SCHLEIFE: J= 1 J= 1.2 J= 1.4

I= 4          (LETZTER WERT( 3 ) + SCHRITTWEITE( 1 ))
J= 1.6        (LETZTER WERT( 1.4 ) + SCHRITTWEITE( .2 ))
```

```
                ANFANGSWERT     ENDWERT         SCHRITTWEITE

I-SCHLEIFE       -1.2            -1.9            -.3
J-SCHLEIFE        1               3               2

I-SCHLEIFE: I=-1.2              J-SCHLEIFE: J= 1 J= 3
I-SCHLEIFE: I=-1.5              J-SCHLEIFE: J= 1 J= 3
I-SCHLEIFE: I=-1.8              J-SCHLEIFE: J= 1 J= 3

I=-2.1        (LETZTER WERT(-1.8 ) + SCHRITTWEITE(-.3 ))
J= 5          (LETZTER WERT( 3 ) + SCHRITTWEITE( 2 ))
```

GO SUB

ANWEISUNG GO SUB

Funktion : Übergibt die Kontrolle der Abarbeitung des Programmes an eine bestimmte Anweisung, bei der ein Unterprogramm beginnt.

Format : GO SUB Zeilennummer

Wirkung : Das Programm fährt bei der Anweisung fort, die durch die Zeilennummer festgelegt wird und die die erste Anweisung eines Unterprogrammes ist. Die letzte Anweisung des Unterprogrammes ist RETURN, mit der in die Zeile nach GO SUB zurückgesprungen wird (in die Zeile mit der nächst höheren Zeilennummer).

Bemerkung : In einem Unterprogramm können auch mehrere RETURN vorkommen. Der Aufruf eines Unterprogrammes kann in einer mehrzeiligen Funktion vorkommen. Die Anweisung GO SUB ist dann ein Element der mehrzeiligen Funktion. Ein Unterprogramm kann selbst GO SUB Anweisungen enthalten. Nach dem RETURN wird jedesmal in die Zeile nach dem letzten GO SUB gesprungen (die Zeile, mit der nächst höheren Zeilennummer). Die Anzahl der möglichen Schachtelungen von Unterprogrammen hängt ab vom Platz, der bei der Ausführung des Programmes im Arbeitsspeicher zur Speicherung der Rücksprungadressen frei bleibt. Der rekursive Aufruf von Unterprogrammen ist möglich.

Beispiel : FILE

```
0010 REM *** PROGRAMMBEISPIEL FUER REKURSIVEN UNTERPROGRAMMAUFRUF ***
0020 REM
0030 LET A=0
0040 PRINT "DAS UNTERPROGRAMM RUFT SICH SELBST AUF:"
0050 PRINT
0060 GOSUB 100
0070 PRINT
0080 PRINT "ENDE DER UNTERPROGRAMMAUFRUFE"
0090 GOTO 170
0100 LET A=A+1
0110 IF A>5 THEN 160
0120 PRINT "Ausfuehrung";A;"des Unterprogrammes"
0130 GOSUB 100
0140 LET A=A-1
0150 PRINT "Ruecksprung von RETURN zu GOSUB";A
0160 RETURN
0170 END

END OF LISTING
```

```
DAS UNTERPROGRAMM RUFT SICH SELBST AUF:

Ausfuehrung 1 des Unterprogrammes
Ausfuehrung 2 des Unterprogrammes
Ausfuehrung 3 des Unterprogrammes
Ausfuehrung 4 des Unterprogrammes
Ausfuehrung 5 des Unterprogrammes
Ruecksprung von RETURN zu GOSUB 5
Ruecksprung von RETURN zu GOSUB 4
Ruecksprung von RETURN zu GOSUB 3
Ruecksprung von RETURN zu GOSUB 2
Ruecksprung von RETURN zu GOSUB 1

ENDE DER UNTERPROGRAMMAUFRUFE
```

GO TO

ANWEISUNG GO TO

Funktion: GO TO bewirkt einen Sprung zu einer bestimmten Zeile des Programmes.

Format: GO TO Zeilennummer

Wirkung: Die Abarbeitung des Programmes setzt in der Zeile fort, in die mit GO TO gesprungen wurde.

Bemerkung: Sprünge in mehrzeilige Funktionen, FOR/NEXT-Schleifen und Subroutinen sind nicht erlaubt, ebenso sind aber auch Sprünge aus Subroutinen heraus unzulässig, wenn danach kein RETURN erreicht wird.

Beim GO TO innerhalb einer Schleife muß das Sprungziel immer zwischen FOR und NEXT liegen, falls die Schleife nicht vorzeitig beendet werden soll.

Beispiele:

```
FILE


0010 REM *** PROGRAMMBEISPIEL 'GOTO' ***
0020 REM
0030 REM *** BEISPIEL FUER EINE ENDLOSSCHLEIFE ***
0040 REM
0050 PRINT "DIES IST EINE ENDLOSSCHLEIFE ................................."
0060 PRINT "DAS PROGRAMM KANN NUR MIT DER TASTE 'BREAK' ABGEBROCHEN WERDEN"
0070 PRINT
0080 GOTO 50
0090 END

END OF LISTING
```

```
DIES IST EINE ENDLOSSCHLEIFE .................................
DAS PROGRAMM KANN NUR MIT DER TASTE 'BREAK' ABGEBROCHEN WERDEN

DIES IST EINE ENDLOSSCHLEIFE .................................
DAS PROGRAMM KANN NUR MIT DER TASTE 'BREAK' ABGEBROCHEN WERDEN

DIES IST EINE ENDLOSSCHLEIFE .................................
DAS PROGRAMM KANN NUR MIT DER TASTE 'BREAK' ABGEBROCHEN WERDEN

DIES IST EINE ENDLOSSCHLEIFE .................................
DAS PROGRAMM KANN NUR MIT DER TASTE 'BREAK' ABGEBROCHEN WERDEN

DIES IST EINE ENDLOSSCHLEIFE .................................
DAS PROGRAMM KANN NUR MIT DER TASTE 'BREAK' ABGEBROCHEN WERDEN

DIES IST EINE ENDLOSSCHLEIFE .................................
DAS PROGRAMM KANN NUR MIT DER TASTE 'BREAK' ABGEBROCHEN WERDEN

DIES IST EINE ENDLOSSCHLEIFE .................................
DAS PROGRAMM KANN NUR MIT DER TASTE 'BREAK' ABGEBROCHEN WERDEN

DIES IST EINE ENDLOSSCHLEIFE .................................
DAS PROGRAMM KANN NUR MIT DER TASTE 'BREAK' ABGEBROCHEN WERDEN
```

I F . . . THEN

ANWEISUNG IF....THEN

Funktion : Diese Anweisung ermöglicht bedingte Verzweigungen in einem Programm.

Format :

$$\text{IF}\left\{\begin{array}{l}\text{Vergleich}\\ \text{(Vergleich 1)}\left\{\begin{array}{l}\text{AND}\\ \text{OR}\end{array}\right\}\text{(Vergleich 2)}\end{array}\right\}\text{THEN Zeilennr.}$$

$$\text{mit : Vergleich} = \left\{\begin{array}{l}\text{num. Ausdr.}\\ \text{Stringausdr.}\end{array}\right\} \text{ Vergl. Operator } \left\{\begin{array}{l}\text{num. Ausdr.}\\ \text{Stringausdr.}\end{array}\right\}$$

Wirkung : Es werden die Ausdrücke berechnet, die Ergebnisse miteinander verglichen und ihr Wahrheitswert gebildet. Sind zwei Vergleiche mit den Boole'schen Operatoren AND oder OR verknüpft, so wird der Wahrheitswert der Verknüpfung gebildet. Liefert der gesamte Ausdruck den Wahrheitswert "wahr", so wird zu der angegebenen Zeilennummer verzweigt.

Liefert der Vergleich oder die Verknüpfung von zwei Vergleichen den Wahrheitswert "falsch", wird das Programm mit der nächsten Zeile fortgesetzt.

Bemerkung : Befindet sich die Anweisung IF...THEN in einer mehrzeiligen Funktion, so muß das Sprungziel der Verzweigung innerhalb der Funktion liegen. Dasselbe gilt auch, wenn die Anweisung in einer Schleife vorkommt und die Schleife nicht aufgrund der Bedingung vorzeitig verlassen werden soll. Beide Vergleichsausdrücke müssen entweder arithmetische Ausdrücke oder Stringausdrücke sein.

Beispiel :

```
FILE

0010 REM *PROGRAMMBEISPIEL 'IF-THEN'*
0020 REM
0030 LET I=1/3*3
0040 IF I=1 THEN 70
0050 PRINT USING 80,I
0060 GOTO 90
0070 PRINT "Da haben Sie Glueck gehabt"
0080 :I=##.###############
0090 END

END OF LISTING


RUN
I= 0.9999999999999
```

IMAGE statement

FORMATSPEZIFIKATIONEN

Funktion:

Die Formatspezifikation bestimmt das Format, in dem die Ausgabegrößen (Zahlen und Strings) für den Drucker und für das Display im jeweiligen Puffer abgebildet werden (siehe PRINT USING, DISP USING und BUILD USING).

Format:

: { Text / Image_Field } [{ Text / Image_Field }] .....

":" ist das Kennzeichen für die Formatspezifikation.

Wirkung:

Beziehen sich die Operanden der Anweisung PRINT USING, MAT PRINT USING, DISP USING oder BUILD USING auf eine Zeile mit einer Formatspezifikation, so werden die Ausgabegrößen für den Drucker und für das Display dem Format entsprechend im jeweiligen Puffer generiert. Die Ausgabe der Werte erfolgt gemäß der Beschreibung im Abschnitt "Ausgabe von Werten bei PRINT USING, DISP USING, BUILD USING".

FORMATFELDER (IMAGE FIELD)

Die Formatfelder einer Formatspezifikation müssen numerisch für numerische Werte, bzw. String-Formatfelder für Stringausdrücke sein.

Die Formatfelder für Zahlenausgabe können sein:

Felder für ganzzahlige Größen

Felder für Dezimalzahlen

Felder für Zahlen in Exponentialdarstellung

Felder mit dem Zeichen ⊄ als Symbol

Formatfeld für ganzzahlige Größen: #[#] ... [#]

Es besteht aus einer Folge von Symbolen, die für Ziffern stehen.

Größe: Minimum 2 Zeichen, Maximum 20

Formatfeld für Dezimalzahlen: #.## ... [#]

Es besteht aus einer Folge von Symbolen, die für Dezimalzahlen stehen.

Größe: Minimal 3 Zeichen, maximal 26 Zeichen inkl. Dezimalpunkt

Formatfeld für Exponentialdarstellung:

Für die Mantisse gelten die Regeln für Dezimalzahlen, zusätzlich ist für das Exponentenfeld ↑↑↑↑ einzugeben.

Größe: Minimum 7 Zeichen, Maximum 30 Zeichen (3 # -Zeichen, ↑↑↑↑)

Bemerkung:

Das Symbol ↑↑↑↑ ist immer das letzte Zeichen des Formatfeldes.

Formatfeld mit ⊄ als Symbol :

⊄ [⊄] ... [⊄] { Formatfeld für Ganzzahlgrößen / Formatfeld für Dezimalgrößen }

Es besteht aus einer Folge von ⊄-Zeichen.

Größe: Minimum 12 Zeichen, Maximum 20 Zeichen

Formatfelder für Stringausdrücke:

'L L ... L

'R R ... R

'C C ... C

bestehen aus einem Hochkomma und eventuell einem oder mehreren Buchstaben L, R oder C.

Größe: Als Minimum 1 Zeichen, maximal 74 Zeichen

Beispiele:

```
LIST
FILE


0010 REM *BEISPIELE FUER FORMATFELDER*
0020 REM
0030 PRINT "Zur Kontrolle der Druckpositionen:"
0040 PRINT
0050 PRINT "123456789!123456789!123456789!123456789!123456789!123456789!"
0060 :########################## ERSTER WERT  ## ZWEITER WERT
0070 :      #####
0080 :##.##########################          ##.#      ###.####
0090 :##################.###
0100 PRINT USING 60,-5,6
0110 PRINT USING 70,12345
0120 PRINT USING 70,0001,0002,0003
0130 PRINT USING 70,123.4,123.75
0140 PRINT USING 90,1.1,12E10
0150 PRINT USING 80,12,0.1,12,0.1236
0160 LET A$="###."
0170 PRINT USING A$,-25.8
0180 :#####.########################††††
0190 PRINT USING 180,-12,12345678901E77,-123,1234567890
0200 :$$$$$$$$$$$$$      $######.####      $$$$.###††††
0210 PRINT USING 200,1500,25.45,25.15E15
0220 PRINT "123456789!123456789!123456789!123456789!123456789!123456789!"
0230 END

END OF LISTING


RUN
Zur Kontrolle der Druckpositionen:

123456789!123456789!123456789!123456789!123456789!123456789!
                      -5 ERSTER WERT   6 ZWEITER WERT
     *****
         1
         2
         3
       123
       123
                1.100
     120000000000.000
**************************           0.1       12.0000
**************************
-26.
-1200.00000000000000000000E-02
 1234.56789010000000000000E+84
-1230.00000000000000000000E-01
 1234.56789000000000000000E+06
        $1500      $    25.4500      $251.500E+14
123456789!123456789!123456789!123456789!123456789!123456789!
```

Beispiel 2:

```
LIST
FILE


0010 REM *BEISPIELE FUER FORMATFELDER*
0020 REM
0030 PRINT "Zur Kontrolle der Druckposition:"
0040 PRINT
0050 PRINT "123456789!123456789!123456789!123456789!123456789!1234567891!"
0060 FOR I=1 TO 3 STEP 1
0070 PRINT
0080 :'LLLLLLLLLLLL
0090 :'RRRRRRRRRRRR
0100 :'CCCCCCCCCCCC
0110 :'LLLLLLLLLLLL
0120 :'LLLLLLL  ††††    $$$$   ####      !"#$%&'()_=<>+ABCDEFGabcdefghi
0130 PRINT I
0140 NEXT I
0150 :'LLLLLLLL   ::::   'CCCCCCCC    ::::    'RRRRRRRR
0160 PRINT USING 80,"OLIVETTI","P6060"
0170 PRINT USING 90,"Olivetti","P6060"
0180 PRINT USING 100,"OLIVETTI","P6060"
0190 PRINT USING 120,"Maerz",125,350
0200 PRINT USING 150,"HUBERT","MARIA","JOSEFA","SYLVIA"
0210 PRINT USING 150,"ENDE"
0220 PRINT "123456789!123456789!123456789!123456789!123456789!1234567891!"
0230 END

END OF LISTING


RUN

Zur Kontrolle der Druckposition:

123456789!123456789!123456789!123456789!123456789!1234567891

 1

 2

 3
OLIVETTI
P6060
      Olivetti
         P6060
   OLIVETTI
    P6060
Maerz     ††††    $125    350       !"#$%&'()_=<>+ABCDEFGabcdefghi
HUBERT       ::::      MARIA       ::::        JOSEFA
SYLVIA       ::::                  ::::
ENDE         ::::                  ::::
123456789!123456789!123456789!123456789!123456789!1234567891
```

AUSGABE VON WERTEN BEI DEN ANWEISUNGEN PRINT USING, MAT PRINT USING, DISP USING, BUILD USING

Die Ausgabe der Daten erfolgt nach den folgenden Regeln :

I. Bei numerischen Größen entspricht in einem Formatfeld für ganzzahlige Werte jeder Ziffer das Symbol #.- Für die Ausgabe des Vorzeichens ist über die maximale Stellenzahl hinaus ein weiteres Symbol vorzusehen. Die Zahl wird rechtsbündig in das Formatfeld übertragen; ist die Zahl nicht ganzzahlig, so wird der gebrochene Anteil abgeschnitten; ist die Zahl positiv, so ist das letzte Zeichen vor der ersten Ziffer eine Leerstelle; ist die Zahl negativ, so steht ein Minus vor der ersten Ziffer dieser Zahl; im Falle eines Formatüberlaufes werden an Stelle der auszugebenden Ziffern * (Sternchen) ausgegeben.

II. Bei numerischen Größen entspricht in einem Formatfeld für Dezimalzahlen jeder Ziffer das Symbol #.

Sind für den gebrochenen Anteil der Zahl zu wenig Symbole vorgesehen, so wird gerundet und der letzte Teil rechts abgeschnitten;
ist die Zahl positiv, so ist das letzte Zeichen vor der ersten Ziffer eine Leerstelle;
ist die Zahl negativ, so steht ein Minus vor der ersten Ziffer dieser Zahl;
sind für den ganzzahligen Anteil der Zahl zu wenig Symbole # vorgesehen, wird für jedes Zeichen des Formatfeldes ein * ausgegeben.

III. Bei numerischen Größen entspricht in einem Formatfeld für Zahlen in Exponentialdarstellung jeder Ziffer ein Symbol #, analog zu II. Anstelle der Zeichen ↑↑↑↑ wird der Buchstabe E, ein Minus- oder Leerzeichen und 2 darauffolgende Ziffern gesetzt. Diese Ziffern bilden den Exponenten zur Basis 10.

IV. Der numerische Wert wird auf ein Formatfeld mit einem $-Zeichen als Symbol abgebildet, wobei nur ein einziges $-Zeichen links im Formatfeld stehen muß.

Anmerkung : In allen Formatfeldern für die Ausgabe numerischer Werte muß eine Stelle für das Vorzeichen enthalten sein. Bei der Ausgabe einer positiven Zahl wird die erste Stelle des Formatfeldes (# oder $) durch ein Leerzeichen ersetzt, bei der Ausgabe einer negativen Zahl wird die erste Stelle des Formatfeldes für die Ausgabe des "-" Zeichens verwendet.

Zahlen, deren Absolutbetrag kleiner 1 ist, werden mit 0 vor dem Dezimalpunkt dargestellt. Ein Formatfeld für Dezimalzahlen muß daher vor dem Dezimalpunkt zwei #-Zeichen haben.

INPUT

ANWEISUNG INPUT

Funktion : Diese Anweisung erlaubt eine Zuweisung von Werten an Variable während der Ausführung eines Programmes.

Die Werte werden über die Tastatur eingegeben.

Format : INPUT {num. Var. / Stringvar.} [, {num. Var. / Stringvar.} .....]

Wirkung : Das Programm wird angehalten und auf dem Display erscheint ein Fragezeichen. Der Operator kann nun die Werte, getrennt durch Komma, eingeben. Sie werden der Reihe nach den Variablen der INPUT-Anweisung zugewiesen. Nachdem man allen Variablen einen Wert zugewiesen hat, kann durch die Taste EOL der Ablauf des Programmes fortgesetzt werden.

Bemerkung : Zwei Fragezeichen (??) auf dem Display geben an, daß noch nicht allen Variablen Werte zugewiesen wurden. Das System wartet auf die Eingabe der restlichen Daten. Die Eingabewerte müssen mit dem Typ der entsprechenden Variablen übereinstimmen. Man darf nicht mehr Werte eingeben, als Variable vorhanden sind. Ein String muß in Anführungszeichen (") stehen, falls er folgende Zeichen enthält.

(,) Komma

Leerzeichen am Anfang

Meldungen : ?

Das Fragezeichen gibt an, daß auf eine Eingabezeile gewartet wird. Das Fragezeichen kann auch Bestandteil einer vorangehenden Display-Anzeige sein, falls sie durch das Trennzeichen ; abgeschlossen wurde.

??

Zwei Fragezeichen geben an, daß noch nicht allen Variablen in der Variablenliste der Anweisung INPUT Werte zugewiesen wurden. Die Eingabe ist fortzusetzen, bis allen Variablen Werte zugewiesen wurden.

"TOO MUCH INPUT-EXCESS IGNORED"

Diese Meldung gibt an, daß mehr Werte eingegeben wurden, als Elemente in der Variablenliste stehen.

"INCORRECT FORMAT-RETYPE LINE"

Wird versucht, an numerische Variable alphanumerische Daten zu übergeben, so erscheint diese Meldung. Zahlen, die an alphanumerische Variable übergeben werden, werden als String interpretiert und ohne Fehlermeldung übernommen.

Beispiel 1 :

```
LIST
FILE


0010 DISP "ZAHL,STRING,ZAHL";
0020 INPUT I,S$,J
0030 PRINT S$,I,J
0040 GOTO 10
0050 END

END OF LISTING


RUN

ZAHL,STRING,ZAHL?
-1.2563,STRING,.25E-3
STRING           -1.2563           .00025
ZAHL,STRING,ZAHL?
STRING,3,STRING
INCORRECT FORMAT-RETYPE LINE
3,STRING,-3
STRING            3                -3
ZAHL,STRING,ZAHL?
1,2,3
2                  1                3
ZAHL,STRING,ZAHL?
1,"XXXXXXX",3,5,4
TOO MUCH INPUT-EXCESS IGNORED
XXXXXXX            1                3
ZAHL,STRING,ZAHL?
```

Beispiel 2 :

```
0010 LET K=I=9
0020 INPUT I,B(I)
0030 PRINT "Altes I =";K;"  , Neues I =";I;"  , B(";K;") =";B(K)
0040 END

END OF LISTING


RUN
?
2,18.6
Altes I = 9    , Neues I = 2     B( 9 ) = 18.6
```

LET

ANWEISUNG LET

Funktion : Die Anweisung erlaubt es, Werte von Ausdrücken an eine oder mehrere Variable eines Programmes zuzuweisen.

Format :

$$[\text{LET}]\left\{\begin{matrix}\text{n.V.}\ [=\text{n.V.}]\ [=\ldots]=\left\{\begin{matrix}\text{n.V.}\\ \text{n.A.}\end{matrix}\right\}\\ \text{S.V.}\ [=\text{S.V.}]\ [=\ldots]=\left\{\begin{matrix}\text{S.V.}\\ \text{S.A.}\end{matrix}\right\}\end{matrix}\right\}$$

n.V. = num. Variable

S.V. = Stringvariable

Wirkung : Der Ausdruck rechts vom letzten Gleichheitszeichen wird ausgeführt und das Ergebnis den Variablen auf der linken Seite zugewiesen.

Bemerkung : Das Schlüsselwort LET muß nicht eingegeben werden.

Die Anzahl der Variablen links vom Gleichheitszeichen wird beschränkt durch die maximale Anzahl von Zeichen pro Zeile (80).

Die aktuelle Länge der Stringvariablen in einer Zuweisung ist gleich der Anzahl der Zeichen, die sich als Resultat des Stringausdruckes ergeben.

Ist die vereinbarte Länge der Stringvariablen kleiner, so erfolgt eine Fehlermeldung. Nach Drücken der Taste CONTINUE wird der Ergebnisstring entsprechend der Länge der Variablen abgeschnitten.

Beispiel 1 :

```
LIST
FILE


0010 REM *BEISPIEL FUER 'LET'*
0020A=B=C=D=E=F=G=H=I=J=K=L=M=N=O=P=Q=R=S=T=U=V=W=X=Y=Z=A1=B0=C3=D3=E4=G8=U4=4
0030 PRINT A,B,C,D,L
0040 LET A=0
0050 LET A=A+1
0060 PRINT A
0070 IF A<=4 THEN 50
0080 PRINT "A1=";A1
0090 LET A1=PI*C3+SQR(P*Q+R-5)/LGT(W↑Z)
0100 PRINT "A1=";A1
0110 END

END OF LISTING


RUN

 4              4              4              4              4
 1
 2
 3
 4
 5
A1= 4
A1= 14.174592
```

Beispiel 2 :

```
FILE
0010 PRINT
0020 PRINT
0030 LET A$="Die Kapitel"
0040 LET B$=" 4 "
0050 LET C$=" 5 "
0060 LET D$=" 6 "
0070 LET E$=" 7 bis 9 "
0080 DCL 80(F$,G$,H$)
0090 LET F$="beschreiben das System."
0100 LET G$="beschreiben die Sprache."
0110 LET H$=" "+A$+B$+","+C$+"und"+D$+F$
0120 PRINT "H$=";H$
0130 LET H$=A$+E$+G$
0140 PRINT "H$= ";H$
0150 LET S$=""
0160 FOR I=1 TO 9 STEP 1
0170 LET S$=S$+CHR$(I)
0180 NEXT I
0190 PRINT "S$=";S$
0200 END

END OF LISTING
RUN

H$= Die Kapitel 4 , 5 und 6 beschreiben das System.
H$= Die Kapitel 7 bis 9 beschreiben die Sprache.
S$=┌┴┘☒▯♂o↖→
```

NEXT

ANWEISUNG NEXT

Funktion:

NEXT gibt das Ende einer Schleife an.

Format:

NEXT Laufvariable

Wirkung:

Siehe Anweisung FOR/NEXT

ON ... GOSUB

ANWEISUNG ON .. GO SUB

Funktion:

Diese Anweisung bewirkt den Sprung in ein Unterprogramm, wobei das Sprungziel vom Wert eines Ausdruckes abhängt.

Format:

ON arithmetischer Ausdruck GO SUB Zeilennummer [, Zeilennummer] ...

"Zeilennummer" ist die Nummer der Programmzeile, die das Sprungziel enthält.

Wirkung:

Der arithmetische Ausdruck wird berechnet und das Ergebnis wird gerundet.

Diese ganze Zahl gibt an, in welcher der Zeilennummern rechts von GO SUB das Sprungziel liegt.
So wird das Ergebnis 3.85 eines arithmetischen Ausdruckes auf die ganze Zahl 4 gerundet und es erfolgt ein Sprung zu der Zeile, die als 4. Zeilennummer in der Anweisung angegeben ist. Jede Zeilennummer in der Anweisung gibt die erste Zeile des jeweiligen Unterprogrammes an. Die letzte Zeile eines Unterprogrammes enthält die Anweisung RETURN, die den Rücksprung in die Zeile nach ON .. GO SUB bewirkt.

Bemerkung:

In jedem Unterprogramm können auch mehrere RETURN-Anweisungen vorkommen.
Der Aufruf von Unterprogrammen kann auch in mehrzeiligen Funktionen enthalten sein; in diesem Falle ist dann auch die Anweisung GO SUB Bestandteil dieser Funktion.
Ergibt der arithmetische Ausdruck nach der Rundung eine Zahl kleiner als 1 oder größer als die Anzahl der Zeilennummern in der Verteilerfunktion ON .. GO SUB, so wird das Programm mit der nächsten Anweisung nach ON ... GO SUB fortgesetzt.

Beispiel:

```
FILE

0010 REM *BEISPIEL FUER 'ON GOSUB'*
0020 REM
0030 DISP "Waehlen Sie ein Unterprogramm";
0040 INPUT I
0050 PRINT
0060 PRINT "I=";I
0070 PRINT
0080 ON I GOSUB 140,160,180,200
0090 IF I<0.5 THEN 120
0100 IF I<4.5 THEN 30
0110 IF I=9 THEN 220
0120 PRINT "KEIN UNTERPROGRAMM UNTER DIESER NUMMER"
0130 GOTO 30
0140 PRINT "DAS IST DAS ERSTE UNTERPROGRAMM"
0150 RETURN
0160 PRINT "DAS IST DAS ZWEITE UNTERPROGRAMM"
0170 RETURN
0180 PRINT "DAS IST DAS DRITTE UNTERPROGRAMM"
0190 RETURN
0200 PRINT "DAS IST DAS VIERTE UNTERPROGRAMM"
0210 RETURN
0220 END

END OF LISTING


3

I= 3

DAS IST DAS DRITTE UNTERPROGRAMM
Waehlen Sie ein Unterprogramm?
2.1

I= 2.1

DAS IST DAS ZWEITE UNTERPROGRAMM
Waehlen Sie ein Unterprogramm?
-12

I=-12

KEIN UNTERPROGRAMM UNTER DIESER NUMMER
Waehlen Sie ein Unterprogramm?
.499

I= .499

KEIN UNTERPROGRAMM UNTER DIESER NUMMER
Waehlen Sie ein Unterprogramm?
.511

I= .511

DAS IST DAS ERSTE UNTERPROGRAMM
Waehlen Sie ein Unterprogramm?
9

I= 9
```

ON ... GOTO

ANWEISUNG ON ... GO TO

Funktion : Diese Anweisung bewirkt den Sprung in eine bestimmte Programmzeile in Abhängigkeit vom Wert eines arithmetischen Ausdruckes.

Format : ON arithmetischer Ausdruck GO TO Zeilennummer [, Zeilennummer] ...

"Zeilennummer" ist die Nummer der Programmzeile, die das Sprungziel enthält.

Wirkung : Der arithmetische Ausdruck wird berechnet und das Ergebnis auf die nächste ganze Zahl gerundet; diese ganze Zahl gibt an, in welcher der Zeilennummern rechts von GO TO das Sprungziel liegt.

So wird die Zahl 2.75 auf 3 gerundet und zu der Zeile gesprungen, die als 3. Zeilennummer angegeben ist.

Bemerkung : Die Anweisung ON ... GO TO kann einen Sprung in eine mehrzeilige Funktion bewirken, wenn die Anweisung selbst bereits Bestandteil einer solchen Funktion ist.

Ergibt der arithmetische Ausdruck eine gerundete Zahl kleiner als 1 oder größer als die Anzahl der Zeilennummern der Verteilerfunktion ON ... GO TO, so wird diese Anweisung ignoriert und das Programm fährt mit der nächsten Anweisung fort.

Beispiel :

```
LIST
FILE


0010 REM *BEISPIEL FUER 'ON GOTO'*
0020 REM
0030 FOR I=-1 TO 5 STEP 1
0040 ON I GOTO 70,90,110,130
0050 PRINT "KEIN GUELTIGES I"
0060 GOTO 140
0070 PRINT "I=EINS"
0080 GOTO 140
0090 PRINT "I=ZWEI"
0100 GOTO 140
0110 PRINT "I=DREI"
0120 GOTO 140
0130 PRINT "I=VIER"
0140 NEXT I
0150 END

END OF LISTING

RUN

KEIN GUELTIGES I
KEIN GUELTIGES I
I=EINS
I=ZWEI
I=DREI
I=VIER
KEIN GUELTIGES I
```

PAD

ANWEISUNG PAD

Funktion:

Eine Stringvariable wird mit Füllzeichen aufgefüllt.

Format:

PAD Stringvariable , n

n ist eine ganze Zahl zwischen 0 und 255 (ISO-Zeichen).

Wirkung:

Der Teil zwischen aktueller Länge und Speicherlänge einer Stringvariablen wird mit dem Zeichen, das der Zahl n in der ISO-Tabelle entspricht, aufgefüllt.

Bemerkung :

Für die Ausführung der Anweisung muß das System mit der Option STR initialisiert sein.

Beispiel :

Siehe Anweisung DEPAD.

ANWEISUNG PRINT

Funktion:

Mit der PRINT-Anweisung kann man Zahlen und Strings im Standardformat über den Drucker ausgeben.

Format:

PRINT [{ Stringausdruck / num. Ausdruck / TAB (num. Ausdruck) } [{ , / ; } { Stringausdruck / num. Ausdruck / TAB (num. Ausdr.) }] .... [{ , / ; }]]

", " und ";" sind Trennzeichen der Ausgabeelemente.

Wirkung:

Die Ergebnisse der Ausdrücke werden im Standardformat am Ende des Kapitels in Abschnitt 8.4. beschrieben und von links nach rechts der Reihe nach gedruckt.
Die Position der Zeichen in der Druckzeile kann mit den Anweisungselementen

TAB (arithmetischer Ausdruck)

", " Komma

";" Strichpunkt

beliebig festgelegt werden, wie es in Abschnittt 8.4. beschrieben wird.
PRINT ohne Angabe von Ausgabeelementen bewirkt die Ausgabe einer Leerzeile.

Stellenkontrolle der Zeichen in einer Druckzeile

Der PRINT-Befehl erzeugt in einem Register, dem Puffer des Druckers, eine Folge von Zeichen, die dem ausgegebenen String entspricht (siehe Abb. 1).

Ein zweites Register, POINTER genannt, weist auf die erste freie Stelle im Puffer. Sobald der Puffer voll ist, wird sein Inhalt in einer Druckzeile ausgegeben (siehe Abb. 1) und der Pointer auf die 1. Stelle im Puffer zurückgesetzt.

Mit den Elementen ",", ";" und TAB (arithmetischer Ausdruck) kann man das Ausgabeformat einer Druckzeile festlegen.

Bemerkung:

Wird das Komma "," als Trennzeichen verwendet, wird der Puffer des Druckers in 5 Druckzonen zu je 16 Stellen aufgeteilt.
Die Funktion TAB bewirkt die Ausgabe des nächsten Elementes ab einer bestimmten Position.

Beispiele:

```
FILE

0010 PRINT "Es folgen 10 Leerzeilen"
0020 FOR I=1 TO 10 STEP 1
0030 PRINT
0040 NEXT I
0050 PRINT "Das waren 10 Leerzeilen"
0060 END

END OF LISTING


RUN

Es folgen 10 Leerzeilen









Das waren 10 Leerzeilen
```

Beispiel 2:

```
FILE


0010 PRINT "OLIVETTI"
0020 PRINT ,,,,,
0030 PRINT "P6060"
0040 PRINT 1,2,3,4,5
0050 PRINT 1;2;3;4;5
0060PRINT1.123,1.123456789E56,-12.123456789,0.0000009,"DIESER STRING IST ZU LANG
    "
0070 END

END OF LISTING
```

```
OLIVETTI

P6060
 1             2              3              4              5
 1  2  3  4  5
 1.123         1.1234568E+56  -12.123457     .0000009
DIESER STRING IST ZU LANG
```

Beispiel 3:

```
FILE


0010 DCL 256A$
0020 LET A$=""
0030 FOR I=0 TO 255 STEP 1
0040 LET A$=A$+CHR$(I)
0050 NEXT I
0060 PRINT A$
0070 END

END OF LISTING
```

```
**** FORMALLY CORRECT PROGRAM ****
■┌┴┘≥θ≠⌂↖→≡↓↑€θθθθθθ≠π⊣⊠↑♀θθθθ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]↑_`abcdefghijklmnopqrstuvwxyz{|}~%$
```

PRINT USING

ANWEISUNG PRINT USING

Funktion : Diese Anweisung bewirkt den Ausdruck von Daten in einem definierten Format.

Format : PRINT USING {Zeilennummer / Stringvariable} , {num. Ausdr. / Stringausdr.} [, {num. Ausdr. / Stringausdr.}] ...

"Zeilennummer" gibt die Zeile mit der Formatspezifikation an.
"Stringvariable" ist der Name der Stringvariablen, die die Formatspezifikation enthält.

Wirkung : Die Werte der Ausdrücke werden in dem Format, das in der Formatspezifikation aufbereitet ist oder in der Stringvariablen angegeben wird ausgedruckt. Jeder Wert korrespondiert mit einem Formatfeld, von links beginnend.

Bemerkung : Jede Anweisung PRINT USING bewirkt, daß die Werte in einer neuen Zeile ausgedruckt werden, auch wenn die vorhergehende Anweisung PRINT mit "," oder ";" endet. Sind mehr Werte auszudrucken, als im Formatstring vorgesehen sind, so werden die restlichen Werte in den nächsten Zeilen im gleichen Format ausgegeben. Sind hingegen weniger Ausgabeelemente als Formatelemente, so werden an Stelle der restlichen Formatelemente Leerzeichen gedruckt. Die Ausgabeelemente und die Formatfelder müssen in ihrem Typ übereinstimmen.

Beispiel :

```
FILE     USING1


0010 REM *** PROGRAMMBEISPIEL 'PRINT USING' ***
0020 REM *** NUMERISCHE FELDER ***
0030 REM
0040 DISP "NUMERISCHER WERT";
0050 INPUT A
0060 PRINT USING 110,A,A,A
0070 PRINT USING 120,A
0080 PRINT USING 130,A,A
0090 PRINT
0100 GOTO 40
0110 :MIT / OHNE NACHKOMMASTELLEN: ##### , #####.# UND #####.####
0120 :EXPONENTIALDARSTELLUNG:                ##.####^^^^
0130 :DOLLAR FELDER:                       $$$$$.# UND $####.####
0140 END

END OF LISTING



RUN
MIT / OHNE NACHKOMMASTELLEN:    123 ,    123.0 UND   123.0000
EXPONENTIALDARSTELLUNG:                    1.23E+02
DOLLAR FELDER:                          $123.0 UND $ 123.0000

MIT / OHNE NACHKOMMASTELLEN:    123 ,    123.5 UND   123.4560
EXPONENTIALDARSTELLUNG:                    1.23E+02
DOLLAR FELDER:                          $123.5 UND $ 123.4560

MIT / OHNE NACHKOMMASTELLEN:   -123 ,   -123.5 UND  -123.4560
EXPONENTIALDARSTELLUNG:                   -1.23E+02
DOLLAR FELDER:                         $-123.5 UND $-123.4560

MIT / OHNE NACHKOMMASTELLEN:      0 ,      0.5 UND     0.5000
EXPONENTIALDARSTELLUNG:                    5.00E-01
DOLLAR FELDER:                           $0.5 UND $   0.5000

MIT / OHNE NACHKOMMASTELLEN:     -0 ,     -0.5 UND    -0.5000
EXPONENTIALDARSTELLUNG:                   -5.00E-01
DOLLAR FELDER:                          $-0.5 UND $  -0.5000
```

Beispiel :

```
FILE     USING2

0010 REM *** PROGRAMMBEISPIEL 'PRINT USING' ***
0020 REM *** ALPHANUMERISCHE FELDER ***
0030 REM
0040 DCL 80 A$
0050 DISP "STRING";
0060 RKB A$
0070 PRINT "STRING: ";A$
0080 PRINT
0090 PRINT USING 150,A$
0100 PRINT USING 160,A$
0110 PRINT USING 170,A$
0120 PRINT
0130 PRINT
0140 GOTO 50
0150 :RECHTSBUENDIGE AUSGABE: 'RRRRRRRRRRRRRRRRRR
0160 :LINKSBUENDIGE AUSGABE:  'LLLLLLLLLLLLLLLLLL
0170 :ZENTRIERTE AUSGABE:     'CCCCCCCCCCCCCCCCCC
0180 END

END OF LISTING


RUN
STRING: A

RECHTSBUENDIGE AUSGABE:                      A
LINKSBUENDIGE AUSGABE:  A
ZENTRIERTE AUSGABE:              A


STRING: OLIVETTI P6060

RECHTSBUENDIGE AUSGABE:       OLIVETTI P6060
LINKSBUENDIGE AUSGABE:  OLIVETTI P6060
ZENTRIERTE AUSGABE:        OLIVETTI P6060


STRING: COMPUTERSYSTEM OLIVETTI P6060

RECHTSBUENDIGE AUSGABE: COMPUTERSYSTEM OLIVE
LINKSBUENDIGE AUSGABE:  COMPUTERSYSTEM OLIVE
ZENTRIERTE AUSGABE:     COMPUTERSYSTEM OLIVE
```

RANDOMIZE

ANWEISUNG RANDOMIZE

Funktion : Beim Aufruf der Anweisung RANDOMIZE in Verbindung mit der Funktion RND wird eine von der Standardfolge verschiedene Folge von Zufallszahlen erzeugt.

Format : RAN [DOMIZE]

Bemerkung : Nach dem Einschalten der Maschine wird bei der Erzeugung der Zufallszahlen jeweils mit der gleichen Basiszahl begonnen.

Mit Hilfe von RANDOMIZE wird bei jedem Programmlauf eine andere Folge von Zufallszahlen erzeugt.

Beispiel 1 :

```
LIST
FILE


0010 REM *ERZEUGUNG VON ZUFALLSZAHLEN*
0020 REM *MIT UND OHNE RANDOMIZE*
0030 REM
0040 FOR I=1 TO 20 STEP 1
0050 PRINT RND,
0060 NEXT I
0070 END

END OF LISTING


RUN

 .63829321       .41839390       .97356004       .88649157       .21507853
 4.7279296E-02   .98707879       .95457924       .55713896       .81827105
 4.7693357E-02   .68675523       .17611750       3.5848356E-02   .50296996
 .62662024       .88834013       .85254312       .73719855       .70058045
RUN
 .63829321       .41839390       .97356004       .88649157       .21507853
 4.7279296E-02   .98707879       .95457924       .55713896       .81827105
 4.7693357E-02   .68675523       .17611750       3.5848356E-02   .50296996
 .62662024       .88834013       .85254312       .73719855       .70058045
RUN
 .63829321       .41839390       .97356004       .88649157       .21507853
 4.7279296E-02   .98707879       .95457924       .55713896       .81827105
 4.7693357E-02   .68675523       .17611750       3.5848356E-02   .50296996
 .62662024       .88834013       .85254312       .73719855       .70058045
```

Beispiel 2 :

```
LIST
FILE


0010 REM *ERZEUGUNG VON ZUFALLSZAHLEN*
0020 REM *MIT UND OHNE RANDOMIZE*
0030 REM
0035 RANDOMIZE
0040 FOR I=1 TO 20 STEP 1
0050 PRINT RND,
0060 NEXT I
0070 END

END OF LISTING


RUN

 .46928183        .93340150        .73981330        .93170842        .66550775
 4.9468396E-02    .41167724        .16474361        4.9257438E-02    .45449761
 .29001342        .73709782        1.0500667E-02    .92415374        .38637937
 .26806888        .15637405        7.7169896E-03    .18172449        3.9252707E-03
RUN
 .30027045        .84385646        .14483391        .59239684        .91177673
 .51377738        .21853162        .35957285        .80906657        .55912861
 .96844639        .33748557        .87592708        .94394928        .14757729
 .35964649        .44423446        .63348580        .61190501        .13459401
RUN
 .13125906        .14975876        .45703308        .24691844        .52625347
 .70871824        .81387396        .89417735        .15268155        .50867605
 .49596029        .44052650        .92023354        .46784937        .41861974
 .59421705        .61918537        .90043514        .44113692        .74452268
```

READ

ANWEISUNG READ

Funktion:

Mit der READ-Anweisung können Werte aus dem internen File an Variable zugewiesen werden.

Die Werte werden in einer internen Tabelle mit Hilfe der Anweisung DATA generiert (siehe Anweisung DATA).

Format:

READ { num.Var. / Stringvar. } [, { num.Var. / Stringvar. }] ...

Wirkung:

Es werden die Werte der internen Tabelle der Reihe nach den Variablen in der READ-Anweisung zugewiesen, wobei mit dem Lesen an der Stelle in der Tabelle begonnen wird, auf die der Pointer zeigt (siehe Anweisung DATA).

Der Pointer kann mit der Anweisung RESTORE auf das 1. Element in der Tabelle zurückgesetzt werden.

Bemerkung:

Den Indices von Feldern werden die Werte erst beim Aufruf innerhalb der READ-Anweisung zugewiesen; es kann also eine Variable der READ-Anweisung als Index eines Feldelementes der gleichen READ-Anweisung verwendet werden. Die Werte müssen in ihrem Typ mit den Variablen übereinstimmen (Zahlen oder Stringvariable).

READ darf nur dann verwendet werden, wenn im Programm mindestens eine DATA-Anweisung vorgekommen ist. Es dürfen nur soviele Variable in einer READ-Anweisung stehen, als noch Werte in der internen Tabelle frei sind.

Beispiele:

```
NEW
AUTO#
10 DATA1,2.54
20 READI,A(I)
30 PRINT"I=";I,"A(I)=";A(I)
40 END
RUN

I= 1             A(I)= 2.54
```

Beispiel 2:

```
LIST
FILE


0010 DATA 1,2,3,4,5,6
0020 FOR I=1 TO 10 STEP 1
0030 READ X
0040 PRINT X
0050 NEXT I
0060 END

END OF LISTING


RUN

 1
 2
 3
 4
 5
 6
ERROR 88  IN LINE 30
```

READ:

ANWEISUNG READ:

Funktion : Mit Hilfe dieser Anweisung kann man Variablen des Programmes Daten aus einem externen File zuweisen.

Format : READ: Filedesignator, {num. Var. / Stringvar.} [, {num. Var. / Stringvar.}] ... [EOF Zeilennr.]

"Filedesignator" : ist ein arithmetischer Ausdruck.

Wirkung : Der Wert des arithmetischen Ausdruckes wird berechnet und auf die nächste ganze Zahl n gerundet. Diese Zahl n bezeichnet das File in der FILES-Anweisung, dessen Elemente gelesen werden sollen. Der Lesevorgang beginnt bei dem Element, auf das der Pointer des Files zeigt. Nach dem Lesen weist der Pointer auf das nächste Element des Datenfiles. Wird versucht, über das Ende des Files hinaus zu lesen, erfolgt eine Fehlermeldung. Ist jedoch der Parameter EOF Zeilennr. angegeben, wird das Programm beim Erreichen des File-Endes, ohne daß eine Fehlermeldung erfolgt, bei der Zeile "Zeilennr." fortgesetzt.

Bemerkung : In einem Random-File kann vor der Anweisung READ: noch die Anweisung SETW: stehen (siehe Anweisung SETW:), um den Pointer an die Stelle des Files zu setzen, ab der die File-Elemente gelesen werden sollen. Fehlt die Anweisung SETW: vor der ersten READ:-Anweisung, so beginnt das Lesen beim ersten Wort des Files.

Textfiles können mit READ: wie sequentielle Datenfiles gelesen werden. Jede Zeile des Textes entspricht einem String, in dem auch die (vierstellige) Zeilennummer enthalten ist. Der Filedesignator muß größer als Null sein und kleiner oder gleich der Anzahl der Filenamen in der FILES:-Anweisung. In einer READ:-Anweisung kann eine Variable als Index eines darauffolgenden Feldelementes derselben READ:-Anweisung verwendet werden. Die vom File mit den Variablen der READ:-Anweisung übereinstimmen.

Beisplele:

```
FILE     READ1


0010 FILES SFILE
0020 DCL 4A$
0030 RESTORE :1
0040 FOR I=1 TO 100 STEP 1
0050 READ :1,A,A$
0060 PRINT A$;A,
0070 NEXT I
0100 END

END OF LISTING


RUN
FILE 1          FILE 2          FILE 3          FILE 4          FILE 5
FILE 6          FILE 7          FILE 8          FILE 9          FILE 10
FILE 11         FILE 12         FILE 13         FILE 14         FILE 15
FILE 16         FILE 17         FILE 18         FILE 19         FILE 20
FILE 21         FILE 22         FILE 23         FILE 24         FILE 25
FILE 26         FILE 27         FILE 28         FILE 29         FILE 30
FILE 31         FILE 32         FILE 33         FILE 34         FILE 35
FILE 36         FILE 37         FILE 38         FILE 39         FILE 40
ERROR 84  IN LINE 50
```

```
FILE     READ2


0010 FILES SFILE
0020 DCL 4A$
0030 RESTORE :1
0040 FOR I=1 TO 100 STEP 1
0050 READ :1,A,A$ EOF 90
0060 PRINT A$;A,
0070 NEXT I
0080 GOTO 100
0090 PRINT "FILE-ENDE NACH";I-1;"WERTEPAAREN ERREICHT"
0100 END

END OF LISTING


RUN
FILE 1          FILE 2          FILE 3          FILE 4          FILE 5
FILE 6          FILE 7          FILE 8          FILE 9          FILE 10
FILE 11         FILE 12         FILE 13         FILE 14         FILE 15
FILE 16         FILE 17         FILE 18         FILE 19         FILE 20
FILE 21         FILE 22         FILE 23         FILE 24         FILE 25
FILE 26         FILE 27         FILE 28         FILE 29         FILE 30
FILE 31         FILE 32         FILE 33         FILE 34         FILE 35
FILE 36         FILE 37         FILE 38         FILE 39         FILE 40
FILE-ENDE NACH 40 WERTEPAAREN ERREICHT
```

REMARK

ANWEISUNG REMARK

Funktion : Mit REMARK können in ein Programm erläuternde Texte eingefügt werden.

Format : REM [ARK] Kommentar

"Kommentar" ist ein beliebiger Text.

Wirkung : REMARK ist eine nicht ausführbare Anweisung; der Kommentar erscheint im Ausdruck (Listing), ist jedoch für die Programmausführung bedeutungslos.

Beispiel :

```
FILE      REM

0010 REM *** DIES IST EIN BEISPIEL FUER DIE ANWEISUNG ***
0020 REM
0030 REM              *** REM [ARK] ***
0040 REM
0050 END

END OF LISTING
```

RESTORE

ANWEISUNG RESTORE

Funktion:

Mit dieser Anweisung wird der Pointer des internen Files auf das erste Element des Files (d.h. auf das erste Element der DATA-Anweisung mit der niedrigsten Zeilennummer) gesetzt.

Format:

RESTORE

Wirkung:

Der Pointer des internen Files wird auf die erste Stelle gesetzt.

Bemerkung:

Von zwei aufeinanderfolgenden RESTORE-Anweisungen ohne I/O Operationen dazwischen, hat die zweite keinerlei Wirkung. Eine RESTORE-Anweisung ohne DATA ist wirkungslos.

Beispiel

```
FILE     REST

0010 FOR I=1 TO 5 STEP 1
0020 READ A
0030 PRINT A,
0040 NEXT I
0050 FOR K=1 TO 5 STEP 1
0060 READ B
0070 PRINT B,
0080 NEXT K
0090 PRINT
0100 RESTORE
0110 FOR I=1 TO 5 STEP 1
0120 READ A
0130 PRINT A,
0140 NEXT I
0150 DATA 10,20,30,40,50,60,70,80,90,100
0160 END

END OF LISTING


RUN
 10            20            30            40            50
 60            70            80            90            100

 10            20            30            40            50
```

RESTORE:

ANWEISUNG RESTORE:

Funktion : Mit dieser Anweisung wird der Pointer eines externen Text- oder sequentiellen Datenfiles auf das erste Element im File positioniert und das File für Leseoperationen geöffnet.

Format : RESTORE: Filedesignator

"Filedesignator" ist ein arithmetischer Ausdruck.

Wirkung : Der Wert des arithmetischen Ausdrucks wird berechnet und auf die nächste ganze Zahl n gerundet.

Der Pointer des Files, das durch den Filedesignator n bestimmt ist, wird auf die 1. Stelle gesetzt.

Bemerkung : Ein sequentielles Daten- oder Textfile ist nach der Anweisung RESTORE: im Lese-Modus. Bei einem Random-File bewirkt diese Anweisung, daß der Pointer auf das 1. Wort gesetzt wird. Anschließend kann sowohl gelesen als auch geschrieben werden.

Beispiel : FILE RESTOR

```
0010 FILES SFILE
0020 RESTORE :1
0030 FOR I=1 TO 10 STEP 1
0040 READ :1,A
0050 PRINT A,
0060 NEXT I
0070 PRINT
0080 RESTORE :1
0090 FOR K=1 TO 5 STEP 1
0100 READ :1,B
0110 PRINT B,
0120 NEXT K
0130 END

END OF LISTING
```

```
RUN
 10            20            30            40            50
 60            70            80            90            100

 10            20            30            40            50
```

RETURN

ANWEISUNG RETURN

Funktion:

RETURN bildet das logische und physische Ende eines Unterprogrammes und bewirkt einen Rücksprung in die Programmzeile, die unmittelbar auf GO SUB oder ON....GO SUB folgt.

Format:

RETURN

Wirkung:

Siehe Anweisungen GO SUB und ON..GO SUB

RKB

ANWEISUNG RKB (Read Keyboard)

Funktion:

Diese Anweisung ermöglicht die Eingabe beliebiger Zeichen (einschließlich aller Sonderzeichen, wie z.B. Komma (,) und Anführungszeichen (")) über die Tastatur und ihre Zuweisung an eine Stringvariable.

Format:

RKB Stringvariable

"Stringvariable" ist eine einfache oder indizierte Variable.

Wirkung:

Die Ausführung des Programmes wird unterbrochen und im Display erscheint ein Fragezeichen "?". Die eingegebene Zeichenfolge wird der Stringvariablen zugewiesen.

Bemerkung:

RKB ist die einzige Anweisung mit der man auch das Anführungszeichen (") einer Stringvariable zuweisen kann.

Beispiel:

```
FILE     RKB

0010 DISP "INPUT STRING";
0020 INPUT A$
0030 DISP "RKB STRING";
0040 RKB B$
0050 PRINT "INPUT :";"'"+A$+"'"
0060 PRINT "RKB   :";"'"+B$+"'"
0070 PRINT
0080 GOTO 10
0090 END

END OF LISTING


RUN
INPUT STRING?
A,B
TOO MUCH INPUT-EXCESS IGNORED
RKB STRING?
A,B
INPUT :'A'
RKB   :'A,B'

INPUT STRING?
"A,B"
RKB STRING?
"A,B"
INPUT :'A,B'
RKB   :'"A,B"'

INPUT STRING?
A
RKB STRING?
A
INPUT :'A
RKB   :'     A
```

SCRATCH:

ANWEISUNG SCRATCH:

Funktion : Die Anweisung ermöglicht das Schreiben auf ein externes, sequentielles Datenfile ab dem ersten Element.

Format : SCRATCH: Filedesignator

"Filedesignator" ist ein arithmetischer Ausdruck

Wirkung : Der Wert des arithmetischen Ausdruckes wird berechnet und auf die nächste ganze Zahl n gerundet.

Der Pointer des Files mit dem Filedesignator n wird an den Anfang des Files gesetzt. Mit WRITE: können nun die Daten sequentiell auf das externe File geschrieben werden.

Bemerkung : Der Filedesignator muß eine Zahl größer als Null und kleiner oder gleich der Anzahl der Filenamen in der FILES-Anweisung sein.

Die SCRATCH:-Anweisung darf nur auf sequentielle Datenfiles angewendet werden.

Beispiel :

```
FILE      SCRAT

0010 FILES SFILE
0020 SCRATCH :1
0030 FOR I=1 TO 10 STEP 1
0040 WRITE :1,I
0050 NEXT I
0060 SCRATCH :1
0070 FOR K=10 TO 100 STEP 10
0080 WRITE :1,K
0090 NEXT K
0100 PRINT "ENDE DES SCHREIBENS"
0110 END

END OF LISTING


RUN
ENDE DES SCHREIBENS

EXEC FLP,SFILE

 10              20              30              40              50
 60              70              80              90              100
```

SETW:

ANWEISUNG SETW: (Set Word)

Funktion : Mit dieser Anweisung kann der Pointer eines Random Files auf ein beliebiges Wort im File gesetzt werden.

Format : SETW: Filedesignator TO Wortdesignator

"Filedesignator" ist ein arithmetischer Ausdruck
"Wortdesignator" ist ein arithmetischer Ausdruck

Wirkung : Die Ergebnisse der arithmetischen Ausdrücke werden auf die nächste ganze Zahl gerundet : n ist der Filedesignator und m der Wortdesignator.

Der Pointer des Files mit dem Filedesignator n wird auf das m-te Wort (4 Byte-Worte) im File positioniert.

Bemerkung : Das File mit dem Designator n muß den Typ Random haben. Nach der Anweisung SETW: können Daten auf das File geschrieben (WRITE:) oder Daten vom File gelesen (READ:) werden.

Der Filedesignator n muß eine ganze Zahl größer als Null und kleiner oder gleich der Anzahl der Filenamen in der FILES-Anweisung sein.

Der Wortdesignator m muß ganzzahlig, größer als Null und (von 4 Byte-Worten ausgehend) kleiner oder gleich der Anzahl der maximal zu speichernden Worte sein (siehe Kapitel 5.4.2 Erstellen von Datenfiles).

Beispiel :

FILE SETW

```
0010 FILES RFILE
0020 DCL SINGLE
0030 SETW :1 TO 1
0040 FOR I=1 TO 100 STEP 1
0050 LET X=INT(100*RND+1)
0060 WRITE :1,X
0070 NEXT I
0080 DISP "POINTER AUF WORT";
0090 INPUT A
0100 IF A<1 THEN 80
0110 IF A>100 THEN 80
0120 SETW :1 TO A
0130 READ :1,B
0140 PRINT "AUF WORT";A;"STEHT DIE ZAHL";B
0150 GOTO 80
0160 END

END OF LISTING
```

```
RUN
AUF WORT 1 STEHT DIE ZAHL 64
AUF WORT 100 STEHT DIE ZAHL 53
AUF WORT 87.5 STEHT DIE ZAHL 100
AUF WORT 88 STEHT DIE ZAHL 100
AUF WORT 88.2 STEHT DIE ZAHL 100
AUF WORT 97 STEHT DIE ZAHL 9
AUF WORT 3.99999 STEHT DIE ZAHL 89
AUF WORT 4 STEHT DIE ZAHL 89
AUF WORT 3.49999 STEHT DIE ZAHL 98
AUF WORT 3 STEHT DIE ZAHL 98
```

STOP

ANWEISUNG STOP

Funktion : Mit dieser Anweisung kann die Ausführung des Programmes unterbrochen werden.

Format : STOP

Wirkung : Die Ausführung des Programmes wird unterbrochen. Auf dem Display erscheint die Meldung "STOP IN LINE Zeilennummer" wobei "Zeilennummer" die Nummer der Zeile mit der STOP-Anweisung angibt.

Das System geht in den DEBUGGING-Mode über (siehe Kapitel 10).

Bemerkung : STOP darf nicht in einer mehrzeiligen Funktion vorkommen. Die Ausführung des Programmes kann mit CONTINUE oder START Zeilennummer fortgesetzt werden, oder durch Drücken der Taste STEP Anweisung für Anweisung abgearbeitet werden.

Die Meldung "STOP IN LINE Zeilennummer" kann unterdrückt werden, wenn zuvor eine DISP-Anweisung steht, die mit Strichpunkt (;) abgeschlossen wurde.

Beispiel :

```
FILE     STOP

0010 FOR I=1 TO 10 STEP 1
0020 LET A(I)=INT(10*RND+1)
0030 NEXT I
0040 STOP
0050 DISP "ABFRAGE DER WERTE";
0060 STOP
0070 END

END OF LISTING


RUN
STOP       IN LINE 40
A(5)
 3
A(3)
 10
ABFRAGE DER WERTE
A(6)
 1
A(10)
 9
```

TRACE OFF

ANWEISUNG TRACE OFF

Funktion:

Mit Hilfe dieser Anweisung kann man die Wirkung von TRACE ON aufheben.

Format:

TRACE OFF

Wirkung:

Es werden die Nummern der ausgeführten Programmzeilen nicht mehr gedruckt (die Gültigkeit von TRACE ON wird aufgehoben).

Bemerkung:

Fehlt TRACE OFF, werden die Zeilennummern solange gedruckt, bis die END-Anweisung erreicht wird. Die Anweisung TRACE OFF hebt auch die Wirkung der Konsoltaste TRACE auf.

Beispiel:

```
FILE     OFF

0010 TRACE ON
0020 DCL SINGLE
0030 DIM A(25)
0040 DEF FNA(X)
0050 LET FN*=X*X
0060 FNEND
0070 FOR I=1 TO 3 STEP 1
0080 PRINT FNA(I),
0090 NEXT I
0100 PRINT
0110 TRACE OFF
0120 GOSUB 140
0130 GOTO 190
0140 FOR K=10 TO 30 STEP 10
0150 PRINT FNA(K),
0160 NEXT K
0170 PRINT
0180 RETURN
0190 END

END OF LISTING


RUN
#70
#80
#90
#80
#90
#80
#90
#100
 1              4              9
#110
 100            400            900
```

TRACE ON

ANWEISUNG TRACE ON

Funktion : Ausdruck der Zeilennummern während eines Programmlaufs.

Format : TRACE ON

Wirkung : Die Zeilennummer der jeweils ausgeführten Anweisung wird ausgedruckt. Ausnahme : nicht ausführbare Anweisungen. (REMARK, DCL, DIM, DEF FN etc.) und die Anweisung TRACE ON.

Der Ausdruck der Zeilennummern erfolgt zwischen den programmgesteuerten Ausgaben.

Bemerkung : Die Anweisung TRACE ON hat dieselbe Wirkung wie die entsprechende Konsoltaste. Die Wirkung wird aufgehoben durch die Anweisung TRACE OFF oder durch Drücken der erleuchteten Konsoltaste TRACE.

FILE ON

Beispiel :

```
0010 TRACE ON
0020 DCL SINGLE
0030 DIM A(25)
0040 DEF FNA(X)
0050 LET FN*=X*X
0060 FNEND
0070 FOR I=1 TO 3 STEP 1
0080 PRINT FNA(I),
0090 NEXT I
0100 PRINT
0110 GOSUB 130
0120 GOTO 180
0130 FOR K=10 TO 30 STEP 10
0140 PRINT FNA(K),
0150 NEXT K
0160 PRINT
0170 RETURN
0180 END

END OF LISTING


RUN
#70
#80
#90
#80
#90
#80
#90
#100
 1                4                9
#110
#130
#140
#150
#140
#150
#140
#150
#160
 100              400              900
#170
#120
#180
```

WHERE:

ANWEISUNG WHERE:

Funktion : In einem externen Datenfile können die Positionen des Pointers, der Typ des adressierten Datenelementes und gegebenenfalls die Länge des adressierten Strings abgefragt werden.

Format : WHERE: Filedesignator, n_1 , n_2 , n_3

"Filedesignator" ist ein arithmetischer Ausdruck.

n_1, n_2, n_3 sind numerische Variable.

Wirkung :

- Der Wert des arithmetischen Ausdrucks wird berechnet und auf die nächste ganze Zahl gerundet. Diese Zahl bezeichnet das externe Datenfile, dessen Pointerposition abgefragt werden soll.
- Der Variablen n_1 wird die Position des Pointers zugewiesen.
- Der Variablen n_2 wird ein Wert zugewiesen, der Aufschluß über den Datentyp des adressierten Datenelementes gibt.

  n_2 kann folgende Werte annehmen :

Wert von n_2	Bedeutung
ø	Der Pointer zeigt auf ein nicht identifizierbares Wort.
1	Der Pointer weist entweder : . auf eine einfach genaue numerische Variable oder . auf das zweite Wort einer doppelt genauen numerischen Variablen oder . auf ein Wort innerhalb eines Strings (d. h. nicht auf das Kontrollwort).
2	Der Pointer zeigt auf das erste Wort einer doppelt genauen numerischen Variablen.
3	Der Pointer weist auf das Kontrollwort eines Strings.
4	Die End - of - file - Marke ist erreicht.
5	Der Pointer weist auf eine nicht initialisierte einfach genaue numerische Variable.

Wert von n_2	Bedeutung
6	Der Pointer zeigt auf das erste Wort einer nicht initialisierten doppelt genauen numerischen Variablen.
7	Der Pointer zeigt auf das Kontrollwort eines nicht initialisierten Strings.

- Wenn der Pointer auf den Anfang eines alphanumerischen Datenelementes weist ($n_2 = 3$), wird der Variablen n_3 die Anzahl der Zeichen des Strings zugewiesen, andernfalls ist $n_3 = \emptyset$.

Bemerkung : Die Variablen n_2 und n_3 können nur dann spezifiziert werden, wenn sich das externe Datenfile im Lesemodus befindet.

Beispiel : Ausdruck eines externen Datenfiles (anstelle des Dienstprogramms FLP).

```
FILE     FLP

0100 REM *** AUSDRUCK EINES FILES MIT HILFE DES STATEMENTS 'WHERE' ***
0110 FILES FILE1
0120 DCL S(X),1023X$
0130 LET P=1
0140 PRINT "Wort - Nr.","Typ/Laenge","Inhalt"
0150 PRINT
0160 PRINT
0170 SETW :1 TO P
0180 REM ********************************************************************
0190 WHERE : 1,A,B,C
0200 REM     A=WORT-NR.     B=DATENTYP     [C=LAENGE DES STRINGS]
0210 REM ********************************************************************
0220 IF B>0 THEN 270
0230 REM     B=0 → NICHT ANFANG EINES DATENELEMENTES ************************
0240 PRINT ,," Pointer innerhalb eines Datenelementes"
0250 LET P=P+1
0260 GOTO 190
0270 IF B>1 THEN 330
0280 REM     B=1 → EINFACH GENAUE NUMERISCHE VARIABLE ************************
0290 READ :1,X
0300 PRINT A," S",X
0310 LET P=P+1
0320 GOTO 190
0330 IF B>2 THEN 390
0340 REM     B=2 → DOPPELT GENAUE NUMERISCHE VARIABLE ************************
0350 READ :1,Y
0360 PRINT A," D",Y
0370 LET P=P+2
0380 GOTO 190
0390 IF B>3 THEN 450
0400 REM     B=3 → STRING DER LAENGE C ***************************************
0410 READ :1,X$
0420 PRINT A,C," ";X$
0430 LET P=P+INT((C-1)/4+2)
0440 GOTO 190
0450 IF B>4 THEN 500
0460 REM     B=4 → FILE - ENDE ERREICHT **************************************
0470 PRINT "FILE - ENDE ERREICHT"
0480 GOTO 530
0490 REM     B>4 → WORT NICHT BESCHRIEBEN ************************************
0500 PRINT A,," nicht beschrieben"
0510 LET P=P+1
0520 GOTO 170
0530 END

END OF LISTING
```

Wort - Nr.	Typ/Laenge	Inhalt
1	S	64
2	S	42
3	S	98
4	D	45
6	D	11
8	D	3
10	D	50
12		nicht beschrieben
13		nicht beschrieben
14		nicht beschrieben
15	8	OLIVETTI
18	5	P6060
21	12	*** TEST ***
25	S	10
26	D	6.5713896
28	S	9
29	D	1.4769336
31	4	ENDE

FILE - ENDE ERREICHT

WRITE:

ANWEISUNG WRITE:

Funktion : Mit WRITE: werden Daten auf ein externes File geschrieben.

Format : WRITE: Filedesignator, {num. Ausdr. / Stringausdr.} [, {num. Ausdr. / Stringausdr.}]... [EOF Zeilennr.]

"Filedesignator" ist ein arithmetischer Ausdruck

Wirkung : Der Wert des arithmetischen Ausdrucks wird berechnet und auf die nächste ganze Zahl n gerundet. Diese Zahl n bezeichnet das File in der FILES-Anweisung, in das geschrieben werden soll. Der Schreibvorgang beginnt bei dem Element, auf das der Pointer des Files zeigt. Nach dem Schreiben weist der Pointer auf das nächste Element des Datenfiles. Wird versucht, über das Ende eines Files hinaus zu schreiben, erfolgt eine Fehlermeldung. Ist jedoch der Parameter EOF Zeilennummer angegeben, wird das Programm beim Erreichen des File-Endes ohne daß eine Fehlermeldung erfolgt bei der Zeile "Zeilennr." fortgesetzt.

Bemerkung : Ist bei einem Random-File WRITE: die erste Anweisung (ohne eine vorherige SETW:-Anweisung), oder folgt WRITE: auf eine FILE:- oder RESTORE:-Anweisung, so wird mit dem Schreiben beim ersten Wort des Files begonnen.

Ein numerischer Ausdruck wird immer in doppelter Genauigkeit geschrieben, auch wenn für die Variablen einfache Genauigkeit vereinbart wurde :

```
ØØ1Ø DCL SINGLE
Ø1ØØ WRITE: 1, 2 * X
```

Der Wert 2 * X wird doppelt genau in das File geschrieben.

```
ØØ1Ø DCL SINGLE
Ø ØØ Y = 2*X
Ø11Ø WRITE: 1, Y
```

Der Wert Y = 2 * X wird einfach genau in das File geschrieben.

Beispiel :

```
FILE     WRITE1


0010 FILES SFILE
0020 DCL 4A$
0030 LET A$="FILE"
0040 SCRATCH :1
0050 FOR I=1 TO 100 STEP 1
0060 WRITE :1,I,A$
0070 NEXT I
0080 END

END OF LISTING



RUN
ERROR 84  IN LINE 60


FILE     WRITE2


0010 FILES SFILE
0020 DCL 4A$
0030 LET A$="FILE"
0040 SCRATCH :1
0050 FOR I=1 TO 100 STEP 1
0060 WRITE :1,I,A$ EOF 90
0070 NEXT I
0080 GOTO 100
0090 PRINT "FILE-ENDE NACH";I-1;"WERTEPAAREN ERREICHT"
0100 END

END OF LISTING



RUN
FILE-ENDE NACH 40 WERTEPAAREN ERREICHT
```

8.2 DIE ANWEISUNGEN DER OPTION MAT (Matrix)

Die im folgenden angegebenen Anweisungen können nur ausgeführt werden, wenn das System mit der Option MAT initialisiert wurde (außer MAT INPUT).

Die Anweisungen erlauben alle Matrix-Operationen mit numerischen Matrizen (zweidimensionalen numerischen Feldern) und Ein-/Ausgabe-Operationen für numerische und alphanumerische Matrizen (zweidimensionale alphanumerische Felder).

Man unterscheidet zwischen den deklarierten und den aktuellen Dimensionen einer Matrix. Die deklarierten Dimensionen werden in einer DIM-Anweisung festgelegt oder, falls diese fehlt, mit den Dimensionen 10 x 10 (10 Zeilen und 10 Spalten) für numerische Matrizen und 5 x 5 für alphanumerische Matrizen angenommen. Die aktuellen Dimensionen einer Matrix können bei bestimmten Operationen verändert werden, sie sind immer kleiner oder gleich den deklarierten Dimensionen.

MAT ... =

ANWEISUNG MAT...= (Matrix Zuweisung)

Funktion : Es werden die Elemente einer Matrix den Elementen einer anderen Matrix zugewiesen.

Format : MAT Matrix = Matrix

Wirkung : Die Werte aller Elemente der Matrix auf der rechten Seite des Gleichheitszeichens werden den korrespondierenden Elementen der Matrix auf der linken Seite zugewiesen.

Bemerkung : Die deklarierten Dimensionen der Matrix auf der linken Seite des Gleichheitszeichens müssen größer oder gleich den aktuellen Dimensionen der rechten Matrix sein.

Beispiel :

```
FILE      MAT1

0010 DIM A(8,8)
0020 FOR I=1 TO 8 STEP 1
0030 FOR K=1 TO 8 STEP 1
0040 LET A(I,K)=10*I+K
0050 NEXT K
0060 NEXT I
0070 PRINT "MATRIX A :"
0080 MAT PRINT A;
0090 PRINT
0100 MAT B=A
0110 PRINT "MATRIX B :"
0120 MAT PRINT B;
0130 PRINT
0140 END

END OF LISTING


RUN
MATRIX A :
 11  12  13  14  15  16  17  18
 21  22  23  24  25  26  27  28
 31  32  33  34  35  36  37  38
 41  42  43  44  45  46  47  48
 51  52  53  54  55  56  57  58
 61  62  63  64  65  66  67  68
 71  72  73  74  75  76  77  78
 81  82  83  84  85  86  87  88

MATRIX B :
 11  12  13  14  15  16  17  18
 21  22  23  24  25  26  27  28
 31  32  33  34  35  36  37  38
 41  42  43  44  45  46  47  48
 51  52  53  54  55  56  57  58
 61  62  63  64  65  66  67  68
 71  72  73  74  75  76  77  78
 81  82  83  84  85  86  87  88
```

MAT ... ±

ANWEISUNG MAT ... ± (Matrix Addition/Subtraktion)

Funktion:

Diese Anweisung ermöglicht die Addition bzw. die Subtraktion der Elemente zweier Matrizen; die Ergebnisse können einer dritten Matrix zugewiesen werden.

Format:

MAT Matrix = Matrix$\left\{\begin{matrix}+\\-\end{matrix}\right\}$Matrix

Wirkung:

Es werden die Werte der korrespondierenden Elemente der beiden Matrizen rechts vom Gleichheitszeichen addiert bzw. subtrahiert und die Ergebnisse dieser Operation der Matrix links vom Gleichheitszeichen zugewiesen. Diese Matrix auf der linken Seite nimmt die aktuellen Dimensionen der beiden rechten Matrizen an.

Bemerkung:

Die Matrizen auf der rechten Seite des Gleichheitszeichens müssen die selben aktuellen Dimensionen haben. Die deklarierten Dimensionen der Matrix auf der linken Seite des Gleichheitszeichens müssen größer oder gleich den aktuellen Dimensionen der beiden rechten Matrizen sein.

Die Ergebnismatrix kann auch als Operand auf der rechten Seite an der Stelle eines oder auch beider Operanden stehen.

Beispiel:

```
FILE      MAT2

0010 DIM A(3,4),B(3,4),C(3,4)
0020 DISP "MATRIX A";
0030 MAT INPUT A
0040 DISP "MATRIX B";
0050 MAT INPUT B
0060 PRINT "MATRIX A :"
0070 MAT PRINT A;
0080 PRINT
0090 PRINT "MATRIX B :"
0100 MAT PRINT B;
0110 PRINT
0120 MAT C=A+B
0130 PRINT "MATRIX C = MATRIX A + MATRIX B :"
0140 MAT PRINT C;
0150 PRINT
0160 MAT C=C-A
0170 PRINT "MATRIX C = MATRIX C - MATRIX A :"
0180 MAT PRINT C;
0190 PRINT
0200 END

END OF LISTING
```

```
RUN
MATRIX A :
 1  2  2  1
 2  3  3  2
 3  4  4  3

MATRIX B :
 2  1  1  2
 3  2  2  3
 4  3  3  4

MATRIX C = MATRIX A + MATRIX B :
 3  3  3  3
 5  5  5  5
 7  7  7  7

MATRIX C = MATRIX C - MATRIX A :
 2  1  1  2
 3  2  2  3
 4  3  3  4
```

MAT ... *
(scalar multip)

ANWEISUNG MAT ... Skalar * (Skalare Multiplikation)

Funktion:

Die Werte aller Elemente einer Matrix werden mit dem Ergebnis eines arithmetischen Ausdruckes multipliziert und die Ergebnisse den korrespondierenden Elementen der Matrix auf der linken Seite des Gleichheitszeichens zugewiesen.
Die Matrix auf der linken Seite nimmt die aktuellen Dimensionen der rechten Matrix an.

Format:

MAT Matrix = (num. Ausdruck) * Matrix

Bemerkung:

Die deklarierten Dimensionen der Matrix links vom Gleichheitszeichen müssen größer oder gleich den aktuellen Dimensionen der Matrix rechts vom Gleichheitszeichen sein. Die angegebene Matrix auf der linken Seite des Gleichheitszeichens kann auch als Operand auf der rechten Seite auftreten.

Beispiel:

```
FILE      MAT3

0010 DIM A(5,3),B(5,3)
0020 DISP "MATRIX A";
0030 MAT INPUT A
0040 DISP "FAKTOR X";
0050 INPUT X
0060 MAT B=(X)*A
0070 PRINT "MATRIX A :"
0080 MAT PRINT A;
0090 PRINT
0100 PRINT "MATRIX B = (";X;") * MATRIX A :"
0110 MAT PRINT B;
0120 PRINT
0130 END

END OF LISTING


RUN
MATRIX A :
 10   20   30
 20   30   10
 30   10   20
 10   20   30
 20   30   10

MATRIX B = ( 3 ) * MATRIX A :
 30   60   90
 60   90   30
 90   30   60
 30   60   90
 60   90   30
```

MAT ... *

ANWEISUNG MAT...* (Matrix Multiplikation)

Funktion : Mit dieser Anweisung kann eine Matrizenmultiplikation durchgeführt werden (Zeilen x Spalten); das Ergebnis wird einer dritten Matrix zugewiesen.

Format : MAT Matrix = Matrix * Matrix

Wirkung : Die beiden Matrizen auf der rechten Seite des Gleichheitszeichens werden nach den Regeln der Matrizenmultiplikation multipliziert. Voraussetzung ist, daß die Anzahl der Spalten der linken Matrix gleich der Anzahl der Zeilen der rechten Matrix ist.

Wird eine Matrix A mit den aktuellen Dimensionen (p, m) mit einer Matrix B mit den Dimensionen (m, n) multipliziert, so erhält man eine Ergebnismatrix C mit den Dimensionen (p, n) für die gilt :

$$i = 1, 2, \ldots, p$$

$$j = 1, 2, \ldots, n$$

$$c_{i,j} = \sum_{k=1}^{m} a_{i,k} * b_{k,j}$$

Bemerkung : Die Ergebnismatrix darf nicht als Operand auf der rechten Seite des Gleichheitszeichens stehen.

Die deklarierten Dimensionen der Matrix auf der linken Seite des Gleichheitszeichens müssen größer oder gleich der Anzahl der Zeilen der ersten Matrix, auf der rechten Seite und der Anzahl der Spalten der zweiten Matrix sein.

Beispiel :

```
FILE     MAT4

0010 DIM A(2,3),B(3,2),C(2,2),D(3,3)
0020 DISP "MATRIX A";
0030 MAT INPUT A
0040 DISP "MATRIX B";
0050 MAT INPUT B
0060 MAT C=A*B
0070 MAT D=B*A
0080 PRINT "MATRIX A :"
0090 MAT PRINT A;
0100 PRINT
0110 PRINT "MATRIX B :"
0120 MAT PRINT B;
0130 PRINT
0140 PRINT "MATRIX C = MATRIX A * MATRIX B :"
0150 MAT PRINT C;
0160 PRINT
0170 PRINT "MATRIX D = MATRIX B * MATRIX A :"
0180 MAT PRINT D;
0190 PRINT
0200 END

END OF LISTING



RUN
MATRIX A :
 2   4   6
 6   4   2

MATRIX B :
 6   4
 2   2
 4   6

MATRIX C = MATRIX A * MATRIX B :
 44   52
 52   44

MATRIX D = MATRIX B * MATRIX A :
 36   40   44
 16   16   16
 44   40   36
```

MAT ... CON

ANWEISUNG MAT... CON (einer Matrix)

Funktion : Jedem Element einer Matrix wird der Wert 1 zugewiesen.

Format : MAT Matrix = CON (num. Ausdr., num. Ausdr.)

Wirkung : Jedem Element der numerischen Matrix links vom Gleichheitszeichen wird der Wert 1 zugewiesen.

Falls Parameter angegeben sind, werden die beiden num. Ausdrücke berechnet und auf die nächste ganze Zahl gerundet; diese beiden Zahlen bestimmen die aktuellen Dimensionen der Matrix.

Bemerkung : Die deklarierten Dimensionen der Matrix links vom Gleichheitszeichen müssen größer oder gleich den aktuellen Dimensionen, die sich aus den num. Ausdrücken ergeben, sein.

Beispiel :

```
FILE      MAT5

0010 DIM A(8,8)
0020 MAT A=CON
0030 PRINT "MATRIX A (DIMENSIONEN 8 x 8) :"
0040 MAT PRINT A;
0050 PRINT
0060 MAT A=CON(5,3)
0070 PRINT "MATRIX A (DIMENSIONEN 5 x 3) :"
0080 MAT PRINT A;
0090 PRINT
0100 END

END OF LISTING


RUN
MATRIX A (DIMENSIONEN 8 x 8) :
 1  1  1  1  1  1  1  1
 1  1  1  1  1  1  1  1
 1  1  1  1  1  1  1  1
 1  1  1  1  1  1  1  1
 1  1  1  1  1  1  1  1
 1  1  1  1  1  1  1  1
 1  1  1  1  1  1  1  1
 1  1  1  1  1  1  1  1

MATRIX A (DIMENSIONEN 5 x 3) :
 1  1  1
 1  1  1
 1  1  1
 1  1  1
 1  1  1
```

MAT ... IDN

ANWEISUNG MAT ... IDN (Einheitsmatrix)

Funktion:

Eine quadratische Matrix erhält die Werte der Einheitsmatrix.

Format:

MAT Matrix = IDN [(num. Ausdr., num. Ausdr.)]

Wirkung:

Den Elementen der Hauptdiagonale wird der Wert 1 zugewiesen, alle anderen Elemente der quadratischen Matrix werden mit Null belegt. Falls als Parameter angegeben, werden die num. Ausdrücke berechnet und die Ergebnisse gerundet; diese ganzen Zahlen legen die aktuellen Dimensionen der quadratischen Matrix fest.

Bemerkung:

Die Ergebnisse der beiden num. Ausdrücke müssen gleich und größer als Null sein. Die deklarierten Dimensionen der quadratischen Matrix müssen größer oder gleich den ganzen Zahlen, die sich aus den arithmetischen Ausdrücken ergeben, sein.

Beispiel:

```
FILE      MAT6


0010 DIM A(7,7)
0020 MAT A=IDN
0030 PRINT "MATRIX A (DIMENSIONEN 7 x 7) :"
0040 MAT PRINT A;
0050 PRINT
0060 MAT A=IDN(5,5)
0070 PRINT "MATRIX A (DIMENSIONEN 5 x 5) :"
0080 MAT PRINT A;
0090 PRINT
0100 END

END OF LISTING


RUN
MATRIX A (DIMENSIONEN 7 x 7) :
 1  0  0  0  0  0  0
 0  1  0  0  0  0  0
 0  0  1  0  0  0  0
 0  0  0  1  0  0  0
 0  0  0  0  1  0  0
 0  0  0  0  0  1  0
 0  0  0  0  0  0  1

MATRIX A (DIMENSIONEN 5 x 5) :
 1  0  0  0  0
 0  1  0  0  0
 0  0  1  0  0
 0  0  0  1  0
 0  0  0  0  1
```

MAT INPUT

ANWEISUNG MAT INPUT (Matrix Input)

Funktion : Diese Anweisung erlaubt eine Zuweisung von Werten an die Elemente einer Matrix über die Tastatur.

Format : MAT INPUT Feld [(num. Ausdr., num. Ausdr.)]

Wirkung : Die Programmausführung wird unterbrochen und auf dem Display erscheint ein Fragezeichen (?). Die Elemente werden nun zeilenweise durch Komma getrennt eingegeben.

Nach Ende der Eingabe aller Werte über die Tastatur muß die Taste EOL gedrückt werden. Werden nicht alle Elemente der Matrix mit Werten belegt, so erscheinen auf dem Display zwei Fragezeichen und das System wartet auf die restliche Eingabe. Sind alle Elemente belegt, wird mit der Ausführung des Programmes fortgesetzt.

Werden die Parameter angegeben, so werden die num. Ausdrücke berechnet und die Ergebnisse gerundet; die ganzen Zahlen bestimmen die aktuellen Dimensionen der Matrix und legen somit die Anzahl der einzugebenden Werte fest.

Bemerkung : Die Eingabewerte müssen entsprechend dem Typ der Matrix numerisch oder alphanumerisch sein.

Die Anzahl der Eingabewerte muß gleich sein der Anzahl der Matrixelemente oder gleich dem Produkt der ganzen Zahlen sein, die sich aus den num. Ausdrücken ergeben.

Strings müssen in Anführungszeichen stehen, wenn sie ein Komma oder Leerzeichen am Anfang oder Ende enthalten.

Beispiel

```
FILE     MAT7

0010 DIM A(5,5),B(15,12),A$(5,5),B$(8,9)
0020 DISP "MATRIX A";
0030 MAT INPUT A
0040 DISP "MATRIX B";
0050 MAT INPUT B(5,3)
0060 DISP "MATRIX A$";
0070 MAT INPUT A$
0080 DISP "MATRIX B$";
0090 MAT INPUT B$(2,5)
0100 PRINT "MATRIX A (DIMENSIONEN 5 x 5) :"
0110 MAT PRINT A;
0120 PRINT
0130 PRINT "MATRIX B (DIMENSIONEN 5 x 3) :"
0140 MAT PRINT B;
0150 PRINT
0160 PRINT "MATRIX A$ (DIMENSIONEN 5 x 5) :"
0170 MAT PRINT A$;
0180 PRINT
0190 PRINT "MATRIX B$ (DIMENSIONEN 2 x 5) :"
0200 MAT PRINT B$,
0210 PRINT
0220 END

END OF LISTING


RUN
MATRIX A (DIMENSIONEN 5 x 5) :
 0  0  1  0  0
 0  1  0  1  0
 1  0  1  0  1
 0  1  0  1  0
 0  0  1  0  0

MATRIX B (DIMENSIONEN 5 x 3) :
 11  12  13
 21  22  23
 31  32  33
 41  42  43
 51  52  53

MATRIX A$ (DIMENSIONEN 5 x 5) :
--■--
-■-■-
■-■-■
-■-■-
--■--

MATRIX B$ (DIMENSIONEN 2 x 5) :
HEUTE          IST            DAS            WETTER         SCHOEN
HERR           LEHRER         WIR            WOLLEN         SPAZIERENGEHN
```

ANWEISUNG MAT ... INV (Inverse Matrix)

Funktion:

Von einer Matrix wird die inverse Matrix gebildet.

Format:

MAT Matrix = INV (Matrix)

Wirkung:

Es wird die Inverse der Matrix rechts vom Gleichheitszeichen gebildet; die Werte der inversen Matrix werden in der entsprechenden Reihenfolge den Elementen der Matrix links vom Gleichheitszeichen zugewiesen. Ist M eine quadratische Matrix und N die dazugehörige inverse Matrix so gilt:

$$M*N = N*M = I$$

I = die Einheitsmatrix

Nicht zu jeder Matrix existiert eine inverse Matrix. Ist die Determinate einer Matrix gleich Null, so gibt es keine dazugehörige Inverse. Die Matrix links vom Gleichheitszeichen hat die gleichen Dimensionen wie die rechte Matrix.

Bemerkung:

Die deklarierten Dimensionen der Matrix links vom Gleichheitszeichen müssen gleich oder größer als jene der Matrix rechts vom Gleichheitszeichen sein. Die Berechnung der inversen Matrix liefert auch den Wert der Determinate. Er wird mit der Standardfunktion DET abgerufen.

Beispiel

```
FILE     MAT8

0010 DIM A(3,3)
0020 DISP "MATRIX A";
0030 MAT INPUT A
0040 MAT B=INV(A)
0050 PRINT "MATRIX A :"
0060 MAT PRINT USING 140,A
0070 PRINT
0080 PRINT "MATRIX B = INV(MATRIX A) :"
0090 MAT PRINT USING 140,B
0100 PRINT
0110 MAT A=INV(B)
0120 PRINT "MATRIX A = INV(MATRIX B) :"
0130 MAT PRINT USING 140,A
0140 :###.###   ###.###   ###.###
0150 PRINT
0160 PRINT
0170 GOTO 20
0180 END

END OF LISTING
```

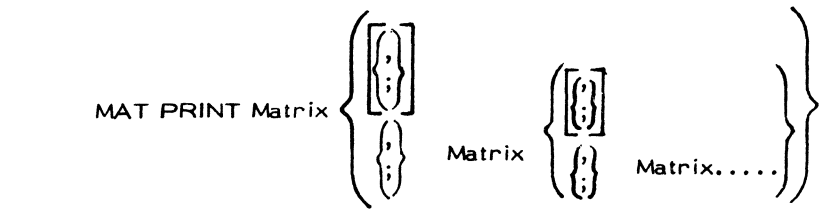
```
RUN
MATRIX A :
  1.000      2.000      3.000
  2.000      3.000      1.000
  3.000      1.000      2.000

MATRIX B = INV(MATRIX A) :
 -0.278      0.056      0.389
  0.056      0.389     -0.278
  0.389     -0.278      0.056

MATRIX A = INV(MATRIX B) :
  1.000      2.000      3.000
  2.000      3.000      1.000
  3.000      1.000      2.000


MATRIX A :
  1.000      0.000      0.000
  0.000      1.000      0.000
  0.000      0.000      1.000

MATRIX B = INV(MATRIX A) :
  1.000      0.000      0.000
  0.000      1.000      0.000
  0.000      0.000      1.000

MATRIX A = INV(MATRIX B) :
  1.000      0.000      0.000
  0.000      1.000      0.000
  0.000      0.000      1.000
```

MAT PRINT

ANWEISUNG MAT PRINT (Matrix Print)

Funktion : Es werden die Werte der Elemente einer oder mehrerer Matrizen über den Drucker ausgegeben.

Format :



"," und ";" bilden die Trennzeichen für die Ausgabe der Elemente der davorstehenden Matrix.

Wirkung : Die Werte der Matrixelemente werden zeilenweise im Standardformat gedruckt. Die Position der Elemente in der Druckzeile wird mit den Zeichen "," und ";" bestimmt.

Positionskontrolle in der Druckzeile

Regeln : Das erste Element jeder Zeile einer Matrix wird immer in der ersten Position einer neuen Druckzeile ausgegeben.

Enthält eine Anweisung MAT PRINT mehr als eine Matrix, so müssen die Feldnamen durch "," oder ";" getrennt werden.

Folgt einer Matrix ein Komma (,), so werden die Elemente jeder Zeile der Matrix beginnend am Anfang einer der 5 Druckspalten, in die eine Druckzeile unterteilt ist, gedruckt.

Folgt hingegen der Matrix ein Strichpunkt (;), so werden die Elemente jeder Zeile der Matrix unmittelbar an den bestehenden Ausdruck anschließend, ausgegeben.

Folgt der letzten Matrix einer MAT PRINT-Anweisung kein Trennungszeichen, so wird das Komma (,) als Trennzeichen angenommen.

Die Matrizen werden entsprechend ihren aktuellen Dimensionen gedruckt.

Beispiel

```
FILE     MAT9


0010 DIM A(5,4),A$(2,3)
0020 FOR I=1 TO 5 STEP 1
0030 FOR K=1 TO 4 STEP 1
0040 LET A(I,K)=10*I+K
0050 NEXT K
0060 NEXT I
0070 DISP "MATRIX A$";
0080 MAT INPUT A$
0090 PRINT "MATRIX A (TRENNZEICHEN KOMMA ',') :"
0100 MAT PRINT A,
0110 PRINT
0120 PRINT "MATRIX A (TRENNZEICHEN STRICHPUNKT ';') :"
0130 MAT PRINT A;
0140 PRINT
0150 PRINT
0160 PRINT "MATRIX A$ (TRENNZEICHEN KOMMA ',') :"
0170 MAT PRINT A$,
0180 PRINT
0190 PRINT "MATRIX A$ (TRENNZEICHEN STRICHPUNKT ';') :"
0200 MAT PRINT A$;
0210 PRINT
0220 END

END OF LISTING


RUN
MATRIX A (TRENNZEICHEN KOMMA ',') :
 11             12             13             14
 21             22             23             24
 31             32             33             34
 41             42             43             44
 51             52             53             54

MATRIX A (TRENNZEICHEN STRICHPUNKT ';') :
 11  12  13  14
 21  22  23  24
 31  32  33  34
 41  42  43  44
 51  52  53  54


MATRIX A$ (TRENNZEICHEN KOMMA ',') :
PETER          SABINE         HANS
GERTRUD        KLAUS          RENATE

MATRIX A$ (TRENNZEICHEN STRICHPUNKT ';') :
PETERSABINEHANS
GERTRUDKLAUSRENATE
```

MAT PRINT USING

ANWEISUNG MAT PRINT USING (Matrix Print Using)

Funktion : Es werden die Elemente eines oder mehrerer Felder in einem vom Benutzer spezifizierten Format gedruckt.

Format : MAT PRINT USING {Zeilennr. / Stringvar} , Feld [, Feld]...

"Zeilennummer" gibt die Zeile mit der gewählten Formatspezifikation an.
"Stringvariable" enthält eine Formatspezifikation.

Wirkung : Die Werte der Elemente jeder Zeile eines Feldes werden in dem Format ausgegeben, das durch die Formatspezifikation oder durch den Inhalt der Stringvariablen festgelegt ist. Der Ausdruck erfolgt von links nach rechts in der entsprechenden Reihenfolge, wobei jedem Element jeder Zeile des Feldes ein Formatelement entspricht.

Bemerkung : Mit jeder Zeile des Feldes wird eine neue Druckzeile begonnen. Enthält ein Feld mehr Elemente in einer Zeile, als im Format vorgesehen ist, so werden die restlichen Werte in der nächsten Zeile im selben Format ausgegeben.

Sind dagegen weniger Zeilenelemente vorhanden als im Format vorgesehen, so werden die restlichen Formatfelder mit Blanks ausgefüllt.

Die Werte der Feldelemente müssen im Typ den Formatspezifikationen entsprechen.

Beispiel

```
FILE     MAT10


0010 DIM A(5,4),A$(2,3)
0020 DCL 80(X$,Y$)
0030 FOR I=1 TO 5 STEP 1
0040 FOR K=1 TO 4 STEP 1
0050 LET A(I,K)=10*I+K
0060 NEXT K
0070 NEXT I
0080 DISP "MATRIX A$";
0090 MAT INPUT A$
0100 DISP "FORMATSTRING FUER MATRIX A";
0110 INPUT X$
0120 DISP "FORMATSTRING FUER MATRIX A$";
0130 INPUT Y$
0140 PRINT "MATRIX A (FORMAT:"
0150 PRINT X$
0160 MAT PRINT USING X$,A
0170 PRINT
0180 PRINT "MATRIX A$ (FORMAT:"
0190 PRINT Y$
0200 MAT PRINT USING Y$,A$
0210 PRINT
0220 GOTO 100
0230 END

END OF LISTING



RUN
MATRIX A (FORMAT:
    ###.##     ###.##     ###.##     ###.##
     11.00      12.00      13.00      14.00
     21.00      22.00      23.00      24.00
     31.00      32.00      33.00      34.00
     41.00      42.00      43.00      44.00
     51.00      52.00      53.00      54.00

MATRIX A$ (FORMAT:
    'LLLLLLLLL     'LLLLLLLLL     'LLLLLLLLL
    PETER          SABINE         HANS
    GERTRUD        KLAUS          RENATE

MATRIX A (FORMAT:
###.###     ###.##     ###.#     ###
 11.000      12.00      13.0      14
 21.000      22.00      23.0      24
 31.000      32.00      33.0      34
 41.000      42.00      43.0      44
 51.000      52.00      53.0      54

MATRIX A$ (FORMAT:
'LLLLLLLLL     'CCCCCCCCC     'RRRRRRRRR
PETER            SABINE            HANS
GERTRUD          KLAUS           RENATE
```

MAT READ

ANWEISUNG MAT READ (Matrix Read)

Funktion : Den Elementen eines Feldes werden Werte aus dem (mit DATA generierten) internen Datenfile zugewiesen. Das Feld kann sowohl numerisch als auch alphanumerisch sein.

Format : MAT READ Feld [(num. Ausdr., num. Ausdr.)] [, Feld[(num. Ausdr., num. Ausdr.)]]...

Wirkung : Zu Beginn der Ausführung des Programmes oder durch die Anweisung RESTORE wird der Pointer an die erste Stelle des internen Files gesetzt.

Durch die Ausführung der Anweisung READ oder MAT READ werden die Werte aus dem internen File der Reihe nach, beginnend ab der Position, auf die der Pointer zeigt, den Variablen oder Feldelementen, die in der READ-Anweisung angegeben sind, zugewiesen.

Bei MAT READ (ohne Parameter) bestimmen die deklarierten Dimensionen die Größe des Feldes.

Sind Parameter angegeben, so werden die num. Ausdrücke berechnet und gegebenenfalls auf die nächste ganze Zahl gerundet. Diese Zahlen sind die aktuellen Dimensionen des Feldes.

Den Feldelementen werden die Werte aus dem File zeilenweise zugewiesen.

Nach jeder Wertzuweisung zeigt der Pointer auf das nächste File-Element.

Bemerkung : Jedem Element der Matrix muß ein Wert in der internen Tabelle entsprechen. Die Werte des internen Files müssen dem Typ nach dem Feld entsprechen (numerisch oder alphanumerisch).

Strings in einer DATA-Anweisung müssen in Anführungszeichen stehen, wenn sie ein Komma (,) oder Leerzeichen am Anfang oder Ende enthalten.

Beispiel

```
FILE     MAT11

0010 DIM A(3,2),B(2,3),A$(2,5)
0020 MAT READ A
0030 RESTORE
0040 MAT READ B,A$
0050 PRINT "MATRIX A :"
0060 MAT PRINT A;
0070 PRINT
0080 PRINT "MATRIX B :"
0090 MAT PRINT B;
0100 PRINT
0110 PRINT "MATRIX A$ :"
0120 MAT PRINT A$;
0130 PRINT
0140 PRINT
0150 DATA 10,20,30,40,50,60
0160 DATA "HANS ","UND ","LIESE ","GEHEN ",SPAZIEREN.,"PAUL ","UND ","LOTTE "
0170 DATA SCHWIM,MEN IM SEE.
0180 END

END OF LISTING


RUN
MATRIX A :
 10  20
 30  40
 50  60

MATRIX B :
 10  20  30
 40  50  60

MATRIX A$ :
HANS UND LIESE GEHEN SPAZIEREN.
PAUL UND LOTTE SCHWIMMEN IM SEE.
```

MAT READ:

ANWEISUNG MAT READ:

Funktion : Einem oder mehreren Feldern werden Werte aus externen Datenfiles zugewiesen.

Format : MAT READ: Filedesignator, Feld[(n.A., n.A.)][,Feld[(n.A., n.A.)]]...[EOF Zeilennr.]

"Filedesignator" ist ein arithmetischer Ausdruck.

n.A. bedeutet numerischer Ausdruck.

Wirkung : Es gelten die gleichen Regeln wie für die Anweisung READ: . Die aus dem Datenfile gelesenen Daten werden den Elementen der angegebenen Matrix zeilenweise zugewiesen. Sind nach dem Namen der Matrix die aktuellen Dimensionen angegeben, so wird die Matrix "redimensioniert", d.h. im weiteren Ablauf des Programmes gelten diese aktuellen Dimensionen.

Bemerkung : Das Produkt der Dimensionen der Matrix darf die Anzahl der im Datenfile vorhandenen Daten nicht überschreiten.

Beispiel :

```
FILE     MAT12


0010 FILES SFILE
0020 RESTORE :1
0030 DIM A(6,6),A$(7,4),B(4,9),B$(4,7)
0040 MAT READ :1,A,A$ EOF 220
0050 RESTORE :1
0060 MAT READ :1,B,B$ EOF 220
0070 PRINT "MATRIX A (DIMENSIONEN 6 x 6) :"
0080 MAT PRINT A;
0090 PRINT
0100 PRINT "MATRIX B (DIMENSIONEN 4 x 9) :"
0110 MAT PRINT B;
0120 PRINT
0130 PRINT "MATRIX A$ (DIMENSIONEN 7 x 4) :"
0140 MAT PRINT USING 150,A$
0150 :' ' ' '
0160 PRINT
0170 PRINT "MATRIX B$ (DIMENSIONEN 4 x 7) :"
0180 MAT PRINT USING 190,B$
0190 :' ' ' ' ' ' ' '
0200 PRINT
0210 GOTO 280
0220 PRINT "FILE-ENDE ERREICHT"
0230 REM
0240 REM *** SCHREIBEN DER HIER GELESENEN MATRIZEN SIEHE ***
0250 REM
0260 REM         *** PROGRAMMBEISPIEL 'MAT WRITE:' ***
0270 REM
0280 END

END OF LISTING


RUN MAT12
MATRIX A (DIMENSIONEN 6 x 6) :
 11  12  13  14  21  22
 23  24  31  32  33  34
 41  42  43  44  51  52
 53  54  61  62  63  64
 71  72  73  74  81  82
 83  84  91  92  93  94

MATRIX B (DIMENSIONEN 4 x 9) :
 11  12  13  14  21  22  23  24  31
 32  33  34  41  42  43  44  51  52
 53  54  61  62  63  64  71  72  73
 74  81  82  83  84  91  92  93  94

MATRIX A$ (DIMENSIONEN 7 x 4) :
( A B C
D E F G
H I J K
L M N O
P Q R S
T U V W
X Y Z )

MATRIX B$ (DIMENSIONEN 4 x 7) :
( A B C D E F
G H I J K L M
N O P Q R S T
U V W X Y Z )
```

ANWEISUNG MAT ... TRN (Transponierte Matrix)

Funktion:

Diese Anweisung bewirkt die Berechnung der Transponierten einer Matrix.

Format:

MAT Matrix = TRN (Matrix)

Wirkung:

Es werden die Zeilen und Spalten der Matrix rechts vom Gleichheitszeichen vertauscht und die so entstandene transponierte Matrix wird der Matrix auf der linken Seite des Gleichheitszeichens zugewiesen.
Die Werte der Spalte x der Matrix rechts vom Gleichheitszeichen stimmen mit den Werten der Zeile x der linken Matrix überein; ebenso korrespondieren die Werte der Zeile y der rechten Matrix mit den Werten der Spalte y der linken Matrix.

Bemerkung:

Die deklarierten Dimensionen der Matrix links vom Gleichheitszeichen müssen größer oder gleich den aktuellen Dimensionen der rechten Matrix sein. Die beiden Matrizen dürfen nicht den gleichen Namen haben.

Beispiel

```
FILE     MAT13


0010 DIM A(3,4),B(4,3)
0020 FOR I=1 TO 3 STEP 1
0030 FOR K=1 TO 4 STEP 1
0040 LET A(I,K)=10*I+K
0050 NEXT K
0060 NEXT I
0070 MAT B=TRN(A)
0080 PRINT "MATRIX A :"
0090 MAT PRINT A;
0100 PRINT
0110 PRINT "MATRIX B = TRN(MATRIX A) :"
0120 MAT PRINT B;
0130 PRINT
0140 END

END OF LISTING


RUN
MATRIX A :
 11   12   13   14
 21   22   23   24
 31   32   33   34

MATRIX B = TRN(MATRIX A) :
 11   21   31
 12   22   32
 13   23   33
 14   24   34
```

MAT WRITE:

ANWEISUNG MAT WRITE: (Matrix Write)

Funktion : Schreibt die Elemente der angeführten Matrizen auf das durch den Filedesignator bestimmte externe Datenfile.

Format : MAT WRITE: Filedesignator, Feld [, Feld]... [EOF Zeilennr.]

"Filedesignator" ist ein arithmetischer Ausdruck.

Wirkung : Es gelten die gleichen Regeln wie für die Anweisung WRITE:.

Die Elemente der angegebenen Matrizen werden zeilenweise in das Datenfile geschrieben.

Beispiel

```
FILE     MAT14


0010 FILES SFILE
0020 SCRATCH :1
0030 DIM A(9,4),A$(4,7)
0040 FOR I=1 TO 9 STEP 1
0050 FOR K=1 TO 4 STEP 1
0060 LET A(I,K)=10*I+K
0070 NEXT K
0080 NEXT I
0090 DISP "MATRIX A$";
0100 MAT INPUT A$
0110 PRINT "MATRIX A (DIMENSIONEN 9 x 4) :"
0120 MAT PRINT A;
0130 PRINT
0140 PRINT "MATRIX A$ (DIMENSIONEN 4 x 7) :"
0150 MAT PRINT USING 160,A$
0160 :' ' ' ' ' ' ' '
0170 PRINT
0180 MAT WRITE :1,A,A$ EOF 200
0190 GOTO 260
0200 PRINT "FILE ZU KLEIN"
0210 REM
0220 REM *** LESEN DER HIER GESCHRIEBENEN MATRIZEN SIEHE ***
0230 REM
0240 REM         *** PROGRAMMBEISPIEL 'MAT READ:' ***
0250 REM
0260 END

END OF LISTING


RUN
MATRIX A (DIMENSIONEN 9 x 4) :
 11  12  13  14
 21  22  23  24
 31  32  33  34
 41  42  43  44
 51  52  53  54
 61  62  63  64
 71  72  73  74
 81  82  83  84
 91  92  93  94

MATRIX A$ (DIMENSIONEN 4 x 7) :
( A B C D E F
G H I J K L M
N O P Q R S T
U V W X Y Z )

EXEC FLP,SFILE

 11              12              13              14              21
 22              23              24              31              32
 33              34              41              42              43
 44              51              52              53              54
 61              62              63              64              71
 72              73              74              81              82
 83              84              91              92              93
 94              (               A               B               C
D                E               F               G               H
I                J               K               L               M
N                O               P               Q               R
S                T               U               V               W
X                Y               Z               )
```

MAT ... ZER

ANWEISUNG MAT... ZER (Null Matrix)

Funktion : Mit dieser Anweisung kann man alle Elemente einer Matrix auf Null setzen.

Format : MAT Matrix = ZER [(num. Ausdruck, num. Ausdruck)]

Wirkung : Jedem Element der angegebenen Matrix wird der Wert Null zugewiesen. Bei Angabe der Parameter werden die num. Ausdrücke berechnet und die Ergebnisse auf die nächsten ganzen Zahlen gerundet (m, n). Diese bilden die aktuellen Dimensionen der Matrix.

Bemerkung : Die deklarierten Dimensionen der Matrix müssen größer oder gleich den durch m und n angegebenen, aktuellen Dimensionen sein.

Beispiel :

```
FILE     MAT15

0010 DIM A(5,3)
0020 MAT A=ZER
0030 PRINT "MATRIX A (DIMENSIONEN 5 x 3) :"
0040 MAT PRINT A;
0050 PRINT
0060 MAT A=ZER(4,2)
0070 PRINT "MATRIX A (DIMENSIONEN 4 x 2) :"
0080 MAT PRINT A;
0090 PRINT
0100 END

END OF LISTING


RUN
MATRIX A (DIMENSIONEN 5 x 3) :
 0  0  0
 0  0  0
 0  0  0
 0  0  0
 0  0  0

MATRIX A (DIMENSIONEN 4 x 2) :
 0  0
 0  0
 0  0
 0  0
```

8.3 DIE STANDARDFUNKTIONEN

Die Standardfunktionen liefern als Ergebnis einen numerischen oder alphanumerischen Wert. Ist das Ergebnis der Funktion numerisch, so wird von einer numerischen Funktion gesprochen, entsprechend wird von alphanumerischen Funktionen gesprochen, wenn das Ergebnis der Funktion alphanumerisch ist.

Numerische Funktionen können auch alphanumerische Parameter (z.B. LEN) und alphanumerische Funktionen können auch numerische Parameter (z.B. CHR$) haben.

Der Aufruf numerischer Standardfunktionen ist innerhalb numerischer Ausdrücke (bzw. anstelle von numerischen Ausdrücken) möglich, analog dazu ist der Aufruf alphanumerischer Funktionen innerhalb von Stringausdrücken möglich.

Das allgemeine Format für Funktionen lautet:

FKT (Arg 1, Arg 2 ...)

FKT: Ist der Name der Funktion, der bei numerischen Funktionen eine Folge von 3 Buchstaben ist, die die Bedeutung der Funktion beschreibt. Alphanumerische Funktionen haben Namen, denen als 4. Zeichen das Dollarzeichen ($) angeführt ist.

Arg: Die Argumente (Parameter) der Funktion können numerische und/oder alphanumerische Ausdrücke sein.

8.3.1 Die trigonometrischen Funktionen

Die trigonometrischen Funktion

SIN (num. Ausdr.)
COS (num. Ausdr.)
TAN (num. Ausdr.)
COT (num. Ausdr.)
ASN (num. Ausdr.)
ACS (num. Ausdr.)
ATN (num. Ausdr.)

liefern die Werte der entsprechenden Sinus-, Cosinus-, Tangens-, Cotangens-, Arcussinus-, Arcuscosinus- und Arcustangensfunktion, wobei das Argument im Bogenmaß vorliegen muß, bzw. bei der Arcussinus-, Arcuscosinus- und Arcustangensfunktion das Ergebnis ein Winkel im Bogenmaß ist.

Für die Umwandlung eines Arguments von Altgrad in Bogenmaß, bzw. eines Arguments von Bogenmaß in Altgrad, dienen die Funktionen

RAD (num. Ausdr.)
DEG (num. Ausdr.)

Werden die Funktionen im CALCULATOR-MODE oder DEBUGGING-MODE verwendet, so ist mit der Vorwahl

SDEG
SGRAD
SRAD

die Angabe der Winkel sowohl in Altgrad (SDEG) oder Neugrad (SGRAD) als auch im Bogenmaß (SRAD) möglich. Werden keine Angaben gemacht, sind Winkel im Bogenmaß einzugeben.

```
FILE     WINKEL

0010 REM *** PROGRAMMBEISPIEL FUER DIE WINKELFUNKTIONEN ***
0020 REM
0030 DEF FNA(X)=PI/180*X
0040 DISP "VON GRAD, BIS GRAD, SCHRITTWEITE";
0050 INPUT A,E,S
0060 IF S=0 THEN 180
0070 PRINT "VON";A;"GRAD BIS";E;"GRAD, SCHRITTWEITE";S;"GRAD"
0080 PRINT
0090 PRINT " GRAD         BOGEN       SINUS      COSINUS      TANGENS     COTANGENS"
0100 PRINT
0110 FOR I=FNA(A) TO FNA(E) STEP FNA(S)
0120 PRINT USING 140,DEG(I),I,SIN(I),COS(I),TAN(I),COT(I)
0130 NEXT I
0140 :###.####   ###.####    ##.####   ##.####   ####.####  ####.####
0150 PRINT
0160 PRINT
0170 GOTO 40
0180 END

END OF LISTING
```

```
RUN
VON 10 GRAD BIS 45 GRAD, SCHRITTWEITE 5 GRAD

 GRAD         BOGEN       SINUS      COSINUS      TANGENS     COTANGENS

 10.0000      0.1745      0.1736     0.9848       0.1763      5.6713
 15.0000      0.2618      0.2588     0.9659       0.2679      3.7321
 20.0000      0.3491      0.3420     0.9397       0.3640      2.7475
 25.0000      0.4363      0.4226     0.9063       0.4663      2.1445
 30.0000      0.5236      0.5000     0.8660       0.5774      1.7321
 35.0000      0.6109      0.5736     0.8192       0.7002      1.4281
 40.0000      0.6981      0.6428     0.7660       0.8391      1.1918
 45.0000      0.7854      0.7071     0.7071       1.0000      1.0000
```

```
FILE     ABINSG

0010 REM *** PROGRAMMBEISPIEL FUER DIE FUNKTIONEN 'ABS','INT' UND 'SGN' ***
0020 REM
0030 DISP "EINGABE EINER ZAHL";
0040 INPUT A
0050 PRINT USING 60,A,ABS(A),A,INT(A),A,SGN(A)
0060 : ABS(###.###)=###.###   INT(###.###)=###.###   SGN(###.###)=##
0070 PRINT
0080 GOTO 30
0090 END

END OF LISTING
```

```
RUN
 ABS(  2.000)=  2.000   INT(  2.000)=  2.000   SGN(  2.000)= 1

 ABS(  2.500)=  2.500   INT(  2.500)=  2.000   SGN(  2.500)= 1

 ABS( -2.500)=  2.500   INT( -2.500)= -3.000   SGN( -2.500)=-1

 ABS(  0.000)=  0.000   INT(  0.000)=  0.000   SGN(  0.000)= 0
```

8.3.2 Mathematische Standardfunktionen

EXP	(X)	Exponentialfunktion (e^x, e: Eulersche Zahl)
HSN	(X)	Sinus hyperbolicus von X
HCS	(X)	Cosinus hyperbolicus von X
HTN	(X)	Tangens hyperbolicus von X
LOG	(X)	Natürlicher Logarithmus (x)
LGT	(X)	Zehnerlogarithmus (Briggscher Logarithmus) ($\log_{10} x$)
SQR	(X)	Quadratwurzel von X ($\sqrt{x}$)
ABS	(X)	Absolutbetrag von X (|x|)
INT	(X)	Nächste ganze Zahl, die kleiner oder gleich dem Wert des Arguments ist
SGN	(X)	Vorzeichen von X Die Funktion liefert bei positivem X als Ergebnis + 1, bei negativem X - 1 und 0, falls X gleich 0 ist.
X :		Arithmetischer Ausdruck

8.3.3 Numerische Funktionen ohne Argument

PI — Die Funktion PI liefert als Ergebnis die Zahl = 3,141592.... in doppelter Genauigkeit.

RND — Die Funktion RND liefert eine im Intervall (0,1) liegende Zufallszahl. Bei wiederholtem Aufruf von RND wird eine Standardfolge von Zufallszahlen geliefert. Wurde vor dem Aufruf von RND die Anweisung RANDOMIZE ausgeführt, liefert die Funktion RND eine von der Standardfolge verschiedene Folge von Zufallszahlen.

Beispiel zu RND (Random) :

```
FILE     RND

0010 REM *** PROGRAMMBEISPIEL FUER DIE FUNKTION 'RND' ***
0020 REM
0030 RANDOMIZE
0040 FOR I=0 TO 3 STEP 1
0050 FOR K=1 TO 5 STEP 1
0060 PRINT RND*10^I,
0070 NEXT K
0080 NEXT I
0090 PRINT
0100 FOR I=1 TO 10 STEP 1
0110 PRINT INT(1000*RND+1),
0120 NEXT I
0130 END

END OF LISTING


RUN
 .30027045       .84385646       .14483391       .59239684       .91177673
 5.1377738       2.1853162       3.5957285       8.0906657       5.5912861
 96.844639       33.748557       87.592708       94.394928       14.757729
 359.64649       444.23446       633.48580       611.90501       134.59401

 455             252             524             932             466
 77              813             409             972             860
```

DET

DET — Die Funktion DET liefert als Ergebnis den Wert der Determinante der zuletzt invertierten Matrix. Bei Aufruf der Funktion muß das System mit der Option MAT initialisiert sein.

Beispiel zu DET (Determinant) :

```
FILE      DET


0010 REM *** PROGRAMMBEISPIEL FUER DIE FUNKTION 'DET' :***
0020 REM
0030 DIM A(5,5),B(5,5)
0040 DISP "MATRIX A";
0050 MAT INPUT A
0060 MAT B=INV(A)
0070 PRINT "MATRIX A"
0080 MAT PRINT USING 90,A
0090 : ####.### ####.### ####.### ####.### ####.###
0100 PRINT
0110 PRINT "INVERSE MATRIX"
0120 MAT PRINT USING 90,B
0130 PRINT
0140 PRINT
0150 PRINT "DETERMINATE DER MATRIX A:";DET
0160 PRINT
0170 PRINT
0180 PRINT
0190 GOTO 40
0200 END

END OF LISTING


RUN
MATRIX A
     1.000     5.000     3.000     6.000     4.000
     8.000    -3.000     2.000     9.000    -1.000
     0.000     3.000     0.000     6.000    -7.000
     9.000     8.000     2.000    -1.000     0.000
     7.000     1.000     4.000     3.000    -2.000

INVERSE MATRIX
    -0.053     0.077    -0.030     0.078    -0.038
     0.048    -0.039     0.048     0.069    -0.051
     0.074    -0.155    -0.033    -0.134     0.340
     0.054     0.069     0.040    -0.015    -0.068
     0.067     0.042    -0.088     0.017    -0.080

DETERMINATE DER MATRIX A: 28409.000
```

8.3.4 Spezielle numerische Funktionen

Es stehen folgende spezielle numerische Funktionen zur Verfügung :

IOC — Die Funktion IOC fragt den Zustand einer peripheren Einheit über ein Arbeitsregister ab.

LEN — Die Funktion liefert als Ergebnis die aktuelle Länge (Anzahl der Zeichen) des als Argument angeführten Stringausdruckes.

SCN — Die Funktion SCN ermöglicht das Aufsuchen der Position eines Teilstrings in einem String.

TAB — Die Funktion TAB ermöglicht bei der Ausgabe von Werten im Standardformat das Drucken ab einer bestimmten Druckposition.

IOC

Funktion IOC (Input/Output Control)

Funktion : Abfrage des Zustandes einer peripheren Einheit über ein Arbeitsregister.

Format : IOC (X)

X ist ein numerischer Ausdruck.

Wirkung : Der numerische Ausdruck wird berechnet und gegebenenfalls auf die nächste ganze Zahl gerundet. Durch die Funktion IOC (X) können die im Arbeitsregister enthaltenen Informationen über den Zustand einer peripheren Einheit abgefragt und im Programm verarbeitet werden.

Die durch IOC (X) gelieferte Information wird in Kapitel 12, Seite 12.10 Tabelle 12.1 beschrieben.

Bemerkungen :

1.) Die Funktion IOC kann wie jede andere numerische Funktion verwendet werden.

2.) Bevor der Zustand einer peripheren Einheit durch IOC abgefragt werden kann, muß der Inhalt des zugehörigen Zustandsregisters dieser Peripherie durch TEST bzw. WAIT in das Arbeitsregister übertragen worden sein.

3.) Die Interpretation der IOC-Werte für X = 3, 4, 5 ist abhängig von dem verwendeten I/O-Kanal.

Beispiel :

```
LIST
FILE

0010 REM *** IOC-FUNKTION ***
0020 BUFFER #9,132
0030 SEND #9,"O L I V E T T IE" AND GO
0040 WAIT #9
0050 IF IOC(6)=1 THEN 70
0060 GOTO 30
0070 DISP "BITTE SCHNELLDRUCKER EINSCHALTEN";
0080 STOP
0090 GOTO 30
0100 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
BITTE SCHNELLDRUCKER EINSCHALTEN
```

LEN

Funktion LEN (Length)

Funktion : Die Funktion LEN liefert als Ergebnis die aktuelle Länge (Anzahl der Zeichen) des als Argument angeführten Stringausdruckes.

Format : LEN (Stringausdruck)

Beispiel :

```
FILE     LEN


0010 REM *** PROGRAMMBEISPIEL FUER DIE FUNKTION 'LEN' ***
0020 REM
0030 DCL 80A$
0040 DISP "STRING (MAX. 40 ZEICHEN)";
0050 RKB A$
0060 IF LEN(A$)>40 THEN 40
0070 PRINT "Der String '";A$;"' enthaelt";LEN(A$);"Zeichen"
0080 PRINT
0090 GOTO 40
0100 END

END OF LISTING


RUN
Der String 'OLIVETTI P6060' enthaelt 14 Zeichen

Der String '' enthaelt 0 Zeichen

Der String 'A+B=C' enthaelt 5 Zeichen
```

SCN

Funktion SCN (Scan)

Funktion : Die Funktion SCN ermöglicht das Aufsuchen der Position eines Teilstrings in einem String.

Format : SCN (Stringausdruck$_1$, Stringausdruck$_2$, num. Ausdruck$_1$, num. Ausdruck$_2$)

Wirkung : Zunächst werden die Ergebnisstrings der beiden Stringausdrücke gebildet ("String$_1$", "String$_2$") und die Ergebnisse p_1, p_2 der beiden numerischen Ausdrücke errechnet und gerundet.

"String$_2$" ist ein Teilstring von "String$_1$".

p_1 gibt an, das wievielte Auftreten von "String$_2$" in "String$_1$" ermittelt werden soll.

p_2 gibt die Stelle in "String$_1$" an, von der ab mit dem Suchen begonnen werden soll.

Als Ergebnis wird die Stelle des ersten Zeichens von "String$_1$" geliefert, an der "String$_2$" ab der p_2-ten Stelle das p_1-te mal in String $_1$ auftritt.

Bemerkung : Kommt "String$_2$" im angegebenen Teil von "String$_1$" nicht oder weniger als p_1-mal vor, so liefert die Funktion SCN als Ergebnis den Wert 0.

Für die Ausführung der Funktion SCN muß das System mit der Option STR initialisiert sein.

Beispiel :

```
FILE     SCN


0010 REM *** PROGRAMMBEISPIEL FUER DIE FUNKTION 'SCN' ***
0020 REM
0030 DCL 80(A$,B$)
0040 DISP "STRING (MAX. 40 ZEICHEN)";
0050 RKB A$
0060 IF LEN(A$)>40 THEN 40
0070 DISP "ZU SUCHENDER SUBSTRING (MAX. 40 ZEICHEN)";
0080 RKB B$
0090 IF LEN(B$)>40 THEN 70
0100 DISP "DAS WIEVIELTE MAL, AB STELLE";
0110 INPUT A,B
0120 PRINT "SCN (";A$;",";B$,",";A;",";B;")=";SCN(A$,B$,A,B)
0130 PRINT
0140 GOTO 40
0150 END

END OF LISTING


RUN
SCN (OLIVETTI P6060,OLIVE, 1 , 1 )= 1

SCN (OLIVETTI P6060,60, 2 , 1 )= 13

SCN (OLIVETTI P6060,60, 1 , 2 )= 11

SCN (OLIVETTI P6060,OLIVE, 2 , 1 )= 0
```

TAB

Funktion TAB (Tabulation)

Funktion : Die Funktion TAB ermöglicht bei der Ausgabe von Werten im Standardformat das Drucken ab einer bestimmten Druckposition.

Format : TAB (X)

X ist ein numerischer Ausdruck.

Beispiel :

```
LIST
FILE


0010 LET A=17.6
0020 LET B=4152.698
0030 LET C=462.1896
0040 PRINT TAB(10);"O U T P U T"
0050 PRINT
0060 PRINT A,B,C;"                        (PRINT ohne TAB)"
0070 PRINT
0080 PRINT
0090 PRINT A;TAB(39);B;TAB(49);C;"     (PRINT mit TAB)"
0100 END

END OF LISTING


RUN

         O U T P U T

 17.6            4152.698        462.1896                    (PRINT ohne TAB)

 17.6                                    4152.698  462.1896    (PRINT mit TAB)
```

Beispiel :

```
FILE     TAB

0010 REM *** PROGRAMMBEISPIEL FUER DIE FUNKTION 'TAB' ***
0020 REM
0030 DISP "TABULIEREN AN STELLE";
0040 INPUT X
0050 IF X=0 THEN 120
0060 IF X<78 THEN 100
0070 IF X>80 THEN 90
0080 GOTO 30
0090 LET X=(X/80-INT(X/80))*80
0100 PRINT TAB(X);X;
0110 GOTO 30
0120 END

END OF LISTING


RUN
          10        20        30        40        50        60        70
     5         15        25        35        45        55        65        75
 1         11        21        31        41        51        61        71
```

8.3.5 Alphanumerische Funktionen

Für die Ausführung von alphanumerischen Funktionen muß das System mit der Option STR initialisiert sein.

An alphanumerischen Funktionen stehen zur Verfügung :

BLN$	Zwei Strings werden durch eine Operation bitweise verknüpft
CHR$	Ein num. Ausdruck wird in ein ISO-Zeichen umgewandelt.
EXT$	Einem Stringausdruck wird ein Teilstring entnommen
REP$	In einem String wird ein Teilstring durch einen anderen String ersetzt

BLN $

Funktion BLN$ (Boolean)

Funktion : Die Zeichen zweier Strings werden nach den Gesetzen der Boole'schen Algebra Bit für Bit verknüpft.

Format : BLN$ (num.Ausdr., Stringausdr.1, Stringausdr. 2)

Stringausdruck : alphanum. Ausdruck.

Wirkung : Der num. Ausdruck wird berechnet und gegebenenfalls auf die nächste ganze Zahl P gerundet.

P gibt an,welche Boole'sche Operation der Verknüpfung zugrunde gelegt wird.

Die Ergenisse der Stringausdrücke werden gebildet und ergeben S1 (String 1) und S2 (String 2). Haben S1 und S2 verschiedene Länge, so wird der kürzere String mit Byte-Inhalt "NUL"-Zeichen bis zur Länge des längeren Strings ergänzt. Die einander entsprechenden Bits werden mit der durch P angegebenen Operation verknüpft und es wird das entsprechende Bit des Ergebnisstrings S_r gebildet.

Der Ergebnisstring S_r hat die Länge Ø, wenn P kleiner als Ø oder größer als 15 ist, ansonsten hat er die Länge des größeren der beiden zu verknüpfenden Strings.

Argument P	Ergebnisstring
0	String, der aus Bytes mit dem ISO-Code ø besteht
1	S1 AND S2
2	S1 AND (NOT S2)
3	S1
4	(NOT S1) AND S2
5	S2
6	S1 ORE S2
7	S1 OR S2
8	S1 NOR S2
9	NOT (S1 ORE S2)
10	NOT S2
11	S1 OR (NOT S2)
12	NOT S1
13	(NOT S1) OR S2
14	S1 NAND S2
15	String, der aus Bytes mit dem ISO-Code 255 besteht.

Verknüpfungstabelle

		P															
S1	S2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
			AND		S1		S2	ORE	OR	NOR	NO-RE	$\overline{S2}$		$\overline{S1}$		NAND	

Beispiel :

```
0010 REM *** BEISPIEL FUER BLM$ ***
0020 REM
0030 REM Die Boole'sche OPeration AND wird zur Umwandlung von
0040 REM Kleinbuchstaben in Grossbuchstaben verwendet.
0050 REM
0060 DCL 40(A$,B$)
0070 LET B$=CHR$(32)
0080 PAD B$,32
0090 DISP "STRING (OHNE SONDERZEICHEN)";
0100 INPUT A$
0110 PRINT A$
0120 LET B$=EXT$(B$,1,LEN(A$))
0130 PRINT BLM$(6,A$,B$)
0140 PRINT
0150 GOTO 70
0160 END

END OF LISTING


ABCDEFGHIJKabcdefghijk
abcdefghijkABCDEFGHIJK
```

CHR$

FUNKTION CHR$ (Charakter)

Funktion: Die Funktion liefert als Ergebnis das Zeichen aus der ISO-Code-Tabelle, dessen Wert dem Argument entspricht.

Format des Aufrufes:

CHR$ (num. Ausdruck)

Wirkung: Der Wert des numerischen Ausdruckes wird berechnet und gegebenenfalls gerundet. Er muß einen Wert n zwischen 0 und 255 ergeben.

Bemerkung: Ist n kleiner als 128, so wird das entsprechende Zeichen der ISO-Code-Tabelle als Ergebnis geliefert.
Ist n = 128 wird das Zeichen Φ , ist n > 128, wird das Zeichen III als Ergebnis geliefert.
Liegt der Wert des numerischen Ausdruckes nicht zwischen 0 und 255, so wird als Ergebnis der Nullstring geliefert und es wird ERROR 2 gemeldet.

Beispiel

```
FILE     CHR

0010 REM *** PROGRAMMBEISPIEL FUER DIE FUNKTION 'CHR$' ***
0020 REM
0030 DCL 80A$
0040 LET A$=""
0050 PRINT "NR. DES ZEICHENS:";
0060 DISP "NR. DES ZEICHENS";
0070 INPUT I
0080 IF I<0 THEN 160
0090 IF I>99 THEN 130
0100 PRINT " ";
0110 IF I>9 THEN 130
0120 PRINT " ";
0130 PRINT I;
0140 LET A$=A$+"   "+CHR$(I)+" "
0150 GOTO 60
0160 PRINT
0170 PRINT "ISO-ZEICHEN:     ";A$
0180 PRINT
0190 PRINT
0200 GOTO 40
0210 END

END OF LISTING


RUN
NR. DES ZEICHENS:  79   76   73   86   69   84   84   73
ISO-ZEICHEN:        O    L    I    V    E    T    T    I


NR. DES ZEICHENS:  80   54   48   54   48   32   66   65   83   73   67
ISO-ZEICHEN:        P    6    0    6    0         B    A    S    I    C


NR. DES ZEICHENS:   0    1    2    3    4    5    6    7    8    9
ISO-ZEICHEN:        ■    ┌    ┴    ┘    ≥    ▒    ≠    ▫    ↖    →


NR. DES ZEICHENS:  10   11   12   13   14   15   16   17   18   19
ISO-ZEICHEN:        ≡    ↓    ↕    ∈    0    0    B    0    0    0

          .
          .
          .
          .
          .
          .
          .
          .

NR. DES ZEICHENS: 120  121  122  123  124  125  126  127  128  129
ISO-ZEICHEN:        x    y    z    {    |    }    ¯    ▮    ↕    ■
```

EXT$

DIE FUNKTION EXT$ (Extract)

Funktion: Einem Stringausdruck wird ein Teilstring entnommen.

Format des Aufrufes:

EXT$ (Stringausdruck, num. Ausdruck$_1$, num. Ausdruck$_2$)

Wirkung: Der Ergebnisstring "String" des Stringausdruckes wird gebildet und die beiden numerischen Ausdrücke werden berechnet und gegebenenfalls gerundet. Die Ergebnisse seien p_1 und p_2.
Als Ergebnis des Funktionsaufrufes wird der Teilstring vom p_1-ten Zeichen bis zum p_2-ten Zeichen des Ergebnisstrings "String" geliefert.

Bemerkung: Ist $p_1 = p_2$, so wird als Ergebnis ein Zeichen geliefert.
p_1 muß größer als Ø und kleiner oder gleich p_2 sein.
p_2 muß kleiner oder gleich der aktuellen Länge des Ergebnisstrings sein.

Beispiel:

```
FILE     EXT

0010 REM *** PROGRAMMBEISPIEL FUER DIE FUNKTION 'EXT$' ***
0020 REM
0030 DCL 80A$
0040 DISP "STRING";
0050 RKB A$
0060 DISP "AB STELLE, BIS STELLE";
0070 INPUT A,B
0080 PRINT "EXT$ (";A$;",";A;",";B;")='";EXT$(A$,A,B);"'"
0090 PRINT
0100 GOTO 40
0110 END

END OF LISTING


RUN
EXT$ (OLIVETTI P6060, 1 , 5 )='OLIVE'

EXT$ (OLIVETTI P6060, 10 , 14 )='P6060'

EXT$ (OLIVETTI, 8 , 1 )=''

EXT$ (OLIVETTI, 1 , 20 )=''
ERROR 2   IN LINE 80
```

REP$

DIE FUNKTION REP$ (Replace)

Funktion: In einem Stringausdruck wird ein Teilstring durch einen anderen String ersetzt.

Format des Aufrufes:

REP$ (Stringausdruck$_1$, Stringausdruck$_2$, Stringausdruck$_3$, num. Ausdruck$_1$, num. Ausdruck$_2$)

Wirkung: Die Stringausdrücke werden ausgewertet und liefern die Ergebnisstrings "String$_1$", "String$_2$" und "String$_3$". Die beiden numerischen Ausdrücke werden berechnet und gegebenenfalls gerundet. Sie liefern die Ergebnisse p_1 und p_2.

"String$_1$" ist der String, der durch die Funktion modifiziert wird

"String$_2$" ist der Teilstring von "String$_1$", der ersetzt wird

"String$_3$" ist der String, durch den "String$_2$" in "String$_1$" ersetzt wird

p_1 gibt an, wie oft "String$_2$" durch "String$_3$" ersetzt wird (p_1-mal)

p_2 gibt die Position des Zeichens in "String$_1$" an, bei dem mit der Suche nach "String$_2$" begonnen werden soll.

Bemerkung: Ist $p_1 = 0$, so hat die Funktion keine Wirkung

Ist p_1 kleiner 0, wird "String$_2$" bei jedem Auftreten durch "String$_3$" ersetzt.

Ist $p_1 > 0$ und "String$_2$" ein Nullstring, so wird "String$_3$" vordem Zeichen p_2 in "String$_1$" p_1-mal eingefügt.

Ist $p_1 < 0$ und "String$_2$" ein Nullstring, so wird ein (behebbarer) Fehler gemeldet (ERROR 2).

Beispiel

```
FILE     REP

0010 REM *** PROGRAMMBEISPIEL FUER DIE FUNKTION 'REP$' ***
0020 REM
0030 DCL 80(A$,B$,C$)
0040 DISP "STRING";
0050 RKB A$
0060 DISP "ZU ERSETZENDER STRING";
0070 RKB B$
0080 DISP "ERSETZENDER STRING";
0090 RKB C$
0100 DISP "WIE OFT, AB STELLE";
0110 INPUT A,B
0120 PRINT "REP$ (";A$;",";B$;",";C$;",";A;",";B;")='";REP$(A$,B$,C$,A,B);"'"
0130 PRINT
0140 GOTO 40
0150 END

END OF LISTING


RUN
REP$ (TEXT TEXT TEXT,X,S, 3 , 1 )='TEST TEST TEST'

REP$ (TEXT TEXT TEXT,X,S,-1 , 1 )='TEST TEST TEST'

REP$ (TEXT TEXT TEXT,TEXT,, 1 , 3 )='TEXT  TEXT'

REP$ (TEXT TEXT TEXT,,TEXT , 1 , 6 )='TEXT TEXT TEXT TEXT'
```

8.4 DAS STANDARDFORMAT

Bei den Anweisungen des Typs

ASSIGN
BUILD
DISP
PRINT
MAT PRINT

werden die Ergebnisse von numerischen oder alphanumerischen Ausdrücken im Standardformat dargestelit bzw. übergeben. Darüber hinaus erfolgt beim Rechnen im DEBUGGING-MODE und bei der Stellung ST des Dezimalstellenrades beim Rechnen im CALCULATOR-MODE die Ausgabe der Werte im Standardformat.

Die Zahlendarstellung und die Darstellung von Strings erfolgt in all diesen Fällen im Standardformat; bei den Anweisungen DISP und PRINT bestehen durch die Stellenbeschränkung und die Möglichkeit der Verwendung von verschiedenen Trennzeichen zusätzliche Möglichkeiten, die getrennt behandelt werden.

8.4.1 Die Zahlendarstellung

a) Darstellung ganzer Zahlen

Ganze Zahlen werden linksbündig entsprechend ihrer Stellenzahl ausgegeben. Das erste Zeichen ist bei positiven Zahlen das Leerzeichen, bei negativen Zahlen das Vorzeichen "-". Am Ende der Zahl steht immer ein Leerzeichen. Ist die Zahl intern in doppelter Genauigkeit dargestellt, so erfolgt die Ausgabe im Standardformat mit maximal 8 Ziffern, ist sie in einfacher Genauigkeit dargestellt, so erfolgt die Ausgabe im Standardformat mit maximal 6 Ziffern. Ganze Zahlen, die in doppelter Genauigkeit mehr als 8 Stellen, in einfacher Genauigkeit mehr als 6 Stellen haben, werden im Gleitkommaformat dargestellt. Dabei erfolgt bei Zahlen in doppelter Genauigkeit eine Rundung auf 8 Stellen, bei Zahlen in einfacher Genauigkeit erfolgt keine Rundung, da auch intern nicht mehr als 6 Stellen vorhanden sind.

b) Dezimalzahlen in Festkommaformat

ba) Darstellung von Dezimalzahlen in doppelter Genauigkeit

Die Darstellung von Dezimalzahlen erfolgt mit maximal 8 Ziffern. Zusätzlich werden eine Stelle vor der Zahl für das Vorzeichen ("-" bei negativen Zahlen, Leerzeichen bei positiven Zahlen), eine Stelle für den Dezimalpunkt und eine Leerstelle am Ende der Zahl benötigt.

Führende Nullen werden unterdrückt, Nullen am Ende der Zahl werden im Dezimalteil nur dann dargestellt, wenn sie Ergebnis einer Rundung sind.

Die Anzahl der Nachkommastellen in der Darstellung richtet sich nach der Anzahl der Vorkommastellen, wobei die Summe aus Vorkomma- und Nachkommastellen max. 8 ist. Ist die Summe aus signifikanten Vorkomma- und Nachkommastellen kleiner als 8, so erfolgt die Darstellung mit entsprechend weniger Stellen. Hat die Zahl genau 8 Vorkommastellen, so wird zwar der Dezimalpunkt, jedoch keine Nachkommastelle dargestellt. Hat die darzustellende Zahl einen Absolutbetrag kleiner als 1, so ist die erste Stelle nach dem Vorzeichen der Dezimalpunkt.

bb) Darstellung von Dezimalzahlen in einfacher Genauigkeit

Für die Darstellung von Dezimalzahlen in einfacher Genauigkeit gelten sinngemäß die gleichen Regeln wie für die Darstellung von Zahlen in doppelter Genauigkeit. Die maximale Anzahl von Ziffern ist jedoch auf 6 begrenzt. Die letzten Stellen werden nicht gerundet, da auch in der internen Darstellung nicht mehr als 6 Ziffern vorhanden sind.

bc) Zahlendarstellung in Gleitkommaformat

Zahlen haben in Gleitkommadarstellung folgendes Format :

- Vorzeichen Mantisse ("-" oder Leerstelle)
- Mantisse
  - Eine Vorkommastelle (ungleich Null)
  - Dezimalpunkt
  - Sieben Nachkommastellen (bei der doppelt genauen Darstellung wird die letzte Ziffer gegebenenfalls gerundet, bei einfach genauer Darstellung sind nur 5 Stellen signifikant).
- E (Kennzeichen für den Exponenten zur Basis 10)
- Vorzeichen der Exponenten ("+" oder "-")
- Exponent (2 Stellen), führende Nullen werden ausgedruckt
- Leerzeichen nach der Zahl

Insgesamt erfordert die Gleitkommadarstellung daher 15 Stellen.

Der Übergang zur Gleitkommadarstellung erfolgt dann, wenn mehr signifikante Stellen vorhanden sind, als in Festkommadarstellung ausgegeben werden können.

bd) Beispiele zur Zahlendarstellung

GLEITKOMMADARSTELLUNG	DOPPELT GENAU	EINFACH GENAU
1.000000000000E+00	1	1
1.000000000000E+01	10	10
1.000000000000E+02	100	100
1.000000000000E+03	1000	1000
1.000000000000E+04	10000	10000
1.000000000000E+05	100000	100000
1.000000000000E+06	1000000	1.0000000E+06
1.000000000000E+09	1.0000000E+09	1.0000000E+09
1.000000000000E-01	.1	.1
1.000000000000E-02	.01	.01
1.000000000000E-03	.001	.001
1.000000000000E-04	.0001	.0001
1.000000000000E-07	.0000001	.0000001
1.000000000000E-10	1.0000000E-10	1.0000000E-10
1.100000000000E+00	1.1	1.1
1.123450000000E+00	1.12345	1.12345
1.123456700000E+00	1.1234567	1.12345
1.123456789000E+00	1.1234568	1.12345
-1.000000000000E+00	-1	-1
-1.123456789000E+00	-1.1234568	-1.12345
1.234567890000E+02	123.45679	123.456
-9.999999910000E+00	-9.9999999	-9.99999
-9.999999910000E+07	-99999999.	-9.9999900E+07

8.4.2 Darstellung von Strings

Die Strings werden zeichenweise linksbündig dargestellt. Die Anzahl der ausgegebenen Zeichen im Beispiel entspricht der aktuellen Länge des Strings.

Beispiel :

```
 FILE

 0010 DCL 80A$
 0020 DISP "STRING";
 0030 RKB A$
 0040 PRINT A$
 0050 GOTO 20
 0060 END

 END OF LISTING


 Strings werden mit ihrer aktuellen Laenge gedruckt.

 [illegible]

 Dieser String wird mit '*' bis zur maximalen Laenge aufgefuellt:**********
```

8.4.3 Stellenkontrolle bei den Anweisungen DISP und PRINT

Für das Display und den Drucker steht je ein Puffer mit 80 Zeichen zur Verfügung, in dem die Aufbereitung der Ausgabezeile erfolgt. Für jeden Puffer existiert ein Pointer, der auf die Stelle im Puffer zeigt, an der mit der Darstellung des nächsten Ausgabeelementes begonnen werden kann.

Die Stellung des Pointers ist einerseits abhängig von den bisher dargestellten Ausgabeelementen, andererseits kann seine Stellung durch die Wahl der Trennzeichen ", " oder ";" oder durch Verwendung der Funktion TAB verändert werden.

Ist die vollständige Darstellung im Puffer nicht möglich, so wird der bisherige Inhalt des Puffers ausgegeben, sein Inhalt gelöscht und mit der weiteren Darstellung der Ausgabeelemente wieder beim ersten Zeichen im Puffer begonnen. Die Darstellung von Strings mit einer Länge von mehr als 80 Zeichen in einer Zeile ist nicht möglich.

8.4.4 Das Trennzeichen Komma ","

Wird in der Liste der Ausgabeelemente das Trennzeichen Komma verwendet, so wird der Puffer in fünf Zonen zu je 16 Zeichen unterteilt. Die Zonen beginnen demnach bei den Positionen 1, 17, 33, 49 und 65. Ist das Trennzeichen, das dem zuletzt dargestellten Ausgabeelement folgt, das Komma, so wird der Wert des Pointers auf das nächsthöhere ganzzahlige Vielfache von 16 + 1 erhöht. Ist dieser Wert größer oder gleich 80, wird der Inhalt des Puffers ausgegeben und erneut mit der Aufbereitung des Pufferinhalts ab der ersten Stelle begonnen.

8.4.5 Das Trennzeichen Strichpunkt ";"

Das Trennzeichen ";" bewirkt keine Änderung der Stellung des Pointers. Es werden daher durch ";" getrennte Ausgabeelemente unmittelbar aneinander anschließend ausgegeben. Der Inhalt des Puffers wird ausgegeben, sobald durch ein weiteres Ausgabeelement 80 Zeichen erreicht oder überschritten werden. Dieses Ausgabeelement kommt dann wieder als erstes Element in den Puffer.

8.4.6 Die Funktion TAB (num. Ausdruck)

Die Funktion TAB erlaubt es, eine beliebige Position im Puffer direkt anzulaufen. Ist diese Position bereits mit Daten belegt, so wird der Inhalt des Puffers ausgegeben und der Pointer wird auf die Stelle im Puffer gesetzt, die durch die Funktion TAB bestimmt ist.

Ist der Wert n, den das gerundete Ergebnis des arithmetischen Ausdrucks liefert, kleiner als 1, so erfolgt eine Fehlermeldung. Ist der Wert n > 80, so wird n = (n/8 - INT (n/8Ø)) * 8Ø gesetzt. Um nicht durch das Trennzeichen eine weitere Tabulation zu bewirken, ist es günstig, nach Aufruf der Funktion TAB das Trennzeichen Strichpunkt ";" zu setzen.

8.4.7 Trennzeichen am Ende der Anweisung DISP oder PRINT

Wird hinter das letzte Element einer Ausgabeliste ein Trennzeichen (";" oder ", ") gesetzt, so wird der Inhalt des Puffers nur dann ausgegeben, wenn 80 Zeichen erreicht wurden. Ausgabeelemente nachfolgender PRINT- (bzw. DISP-) Anweisungen werden an den bestehenden Inhalt des Puffers angefügt. Dadurch ist es möglich, die Elemente mehrerer PRINT- (oder DISP-) Anweisungen in einer Zeile auszugeben.

Fehlt am Ende der Ausgabeliste einer PRINT- (bzw. DISP-) Anweisung das Trennzeichen, so wird der Inhalt des Puffers ausgegeben und der Pointer an die erste Stelle gesetzt. Folgt einer PRINT - (bzw. DISP-) Anweisung, die durch ein Trennzeichen abgeschlossen wurde, eine PRINT- (oder DISP-) Anweisung ohne Ausgabeelemente, so wird der bestehende Inhalt des Puffers ausgegeben und der Pointer auf die erste Stelle zurückgesetzt.

Hat der Druckpuffer keinen Inhalt, bewirkt eine PRINT-Anweisung ohne Ausgabe-Element eine Leerzeile, ist der Puffer des Displays leer, so wird durch eine DISP-Anweisung ohne Ausgabe-Element die Anzeige im Display gelöscht.

8.4.8 Besonderheiten der Anweisung DISP

Im Display können bei der Ausgabe 32 Zeichen angezeigt werden. Da der Puffer des Display jedoch eine Kapazität von 80 Zeichen hat, ist es möglich, daß nicht alle im Puffer generierten Zeichen auch im Display sichtbar sind.
Mit Hilfe der Tasten [→] und [←] und deren Kombinationen mit SHIFT und REPEAT (siehe Kapitel 1) ist es jedoch möglich, die weiteren Zeichen im Display sichtbar zu machen. Dazu ist jedoch notwendig, daß die Anzeige genügend lang im Display sichtbar bleibt, was gegebenenfalls durch eine DELAY - Anweisung mit genügend großem n erreicht werden kann.

8.4.9 Beispiele

```
FILE     TRENN

0010 REM *** WIRKUNG DER TRENNZEICHEN ',' UND ';' ***
0020 REM
0030 DISP "STRING 1,STRING 2,ZAHL 1,ZAHL 2";
0040 INPUT A$,B$,A,B
0050 PRINT "KOMMA ";A$,B$,A,B
0060 PRINT "STRICHPUNKT:";A$;B$;A;B
0070 PRINT
0080 PRINT "ES WURDEN DIE STRINGS ";
0090 PRINT "A$='";A$;"' UND B$='";B$,"' EINGEGEBEN"
0100 PRINT "DIE ZAHLEN WAREN",
0110 PRINT "A=";A,"UND","B=";B
0120 PRINT
0130 GOTO 30
0140 END

END OF LISTING
```

```
RUN
KOMMA            OLIVETTI        P6060             1.2345679       123456.79
STRICHPUNKT:OLIVETTIP6060 1.2345679  123456.79

ES WURDEN DIE STRINGS A$='OLIVETTI' UND B$='P6060' EINGEGEBEN
DIE ZAHLEN WAREN                 A= 1.2345679    UND             B= 123456.79

KOMMA            TEST            BEISPIEL          1               2
STRICHPUNKT:TESTBEISPIEL 1  2

ES WURDEN DIE STRINGS A$='TEST' UND B$='BEISPIEL' EINGEGEBEN
DIE ZAHLEN WAREN                 A= 1            UND             B= 2
```

```
FILE     ZAHL

0010 REM *** ZAHLENDARSTELLUNG ***
0020 REM
0030 FOR I=1 TO 10 STEP 1
0040 PRINT I,10/I,10↑I,INT(100*RND+1),SQR(I)
0060 NEXT I
0070 END

END OF LISTING
```

```
RUN
 1            10            10            14            1
 2            5             100           15            1.4142136
 3            3.3333333     1000          46            1.7320508
 4            2.5           10000         25            2
 5            2             100000        53            2.2360680
 6            1.6666667     1000000       71            2.4494897
 7            1.4285714     10000000      82            2.6457513
 8            1.25          1.0000000E+08 90            2.8284271
 9            1.1111111     1.0000000E+09 16            3
 10           1             1.0000000E+10 51            3.1622777
```

```
FILE     PUFFER

0010 REM *** AUSDRUCK DES DRUCKPUFFERS BEI ERREICHEN VON 80 ZEICHEN ***
0020 REM
0030 TRACE ON
0040 FOR I=1 TO 7 STEP 1
0050 PRINT "***";I;"***",INT(100*RND+1),
0060 NEXT I
0070 PRINT
0080 TRACE OFF
0090 END

END OF LISTING


RUN
#40
#50
#60
#50
#60
#50
*** 1 ***         14              *** 2 ***         15              *** 3 ***
#60
#50
#60
#50
 46              *** 4 ***         25              *** 5 ***         53
#60
#50
#60
#50
#60
#70
*** 6 ***         71              *** 7 ***         82
#80
```

```
FILE     STRING


0010 REM *** AUSDRUCK EINES STRINGS MIT BELIEBIGER ANZAHL VON ZEICHEN ***
0020 REM
0030 DCL 1023A$,80B$
0040 DISP "TEILSTRING";
0050 RKB B$
0060 DISP "ZU DRUCKENDE LAENGE";
0070 INPUT A
0080 PRINT "TEILSTRING = '";B$;"'"
0090 PRINT "ZU DRUCKENDE LAENGE =";A,"ZEICHEN"
0100 PRINT "LAENGE DES TEILSTRINGS =";LEN(B$);"ZEICHEN"
0110 PRINT
0120 LET A$=""
0130 FOR I=1 TO A/LEN(B$) STEP 1
0140 LET A$=A$+B$
0150 NEXT I
0160 IF LEN(A$)=A THEN 210
0170 IF LEN(A$)+LEN(B$)=A THEN 200
0180 LET A$=A$+EXT$(B$,1,A-LEN(A$))
0190 GOTO 210
0200 LET A$=EXT$(A$+B$,1,A)
0210 PRINT A$
0220 PRINT
0230 PRINT
0240 GOTO 40
0250 END

END OF LISTING



RUN
TEILSTRING = '*-TEST*-'
ZU DRUCKENDE LAENGE = 250 ZEICHEN
LAENGE DES TEILSTRINGS = 8 ZEICHEN

*-TEST*-*-TEST*-*-TEST*-*-TEST*-*-TEST*-*-TEST*-*-TEST*-*-TEST*-*-TEST*-*-TEST*-
*-TEST*-*-TEST*-*-TEST*-*-TEST*-*-TEST*-*-TEST*-*-TEST*-*-TEST*-*-TEST*-*-TEST*-
*-TEST*-*-TEST*-*-TEST*-*-TEST*-*-TEST*-*-TEST*-*-TEST*-*-TEST*-*-TEST*-*-TEST*-
*-TEST*-*-


TEILSTRING = '*-*OLIVETTI P6060-*-'
ZU DRUCKENDE LAENGE = 1023 ZEICHEN
LAENGE DES TEILSTRINGS = 20 ZEICHEN

*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-
*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-
*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-
*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-
*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-
*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-
*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-
*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-
*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-
*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-
*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-
*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-
*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-*-*OLIVETTI P6060-*-*-*
```

9. EINGABE UND EDITING EINES PROGRAMM- ODER TEXT-FILES

9. EINGABE UND EDITING EINES PROGRAMM - ODER TEXT - FILES

Dieses Kapitel vermittelt die notwendigen Kenntnisse, um ein Programm- oder Text-File erzeugen oder ausgeben zu können.

9.1 STRUKTUR EINES PROGRAMMES

Ein BASIC-Programm besteht aus einer Folge von Zeilen (Anweisungen genannt). Jede Programmzeile muß eine Zeilennummer und mindestens ein BASIC-Schlüsselwort enthalten.

Mögliche Zusammensetzung einer BASIC-Programmzeile :

- Eine Zeilennummer
- Ein oder mehrere BASIC-Schlüsselwörter
- Einen oder mehrere Operatoren
- Einen oder mehrere Operanden

Die Zeilennummer : identifiziert jede Programmzeile. Sie setzt sich aus 4 Ziffern zusammen, wobei Nummern, die kleiner als 1000 sind mit führenden Nullen ausgegeben werden.

Das BASIC-Schlüsselwort : gibt die Funktion der Anweisung an.

Die Operatoren : sind Symbole, die Operationen und Relationen zwischen den Operanden kennzeichnen.

Die Operanden : können Konstante, Variable oder Ausdrücke sein.

Die letzte Zeile eines BASIC-Programmes muß die Anweisung END enthalten.

Beispiel eines BASIC - Programmes:

```
FILE     BASIC

0010 REM *** Beispiel fuer ein BASIC - Programm ***
0020 REM
0030 DCL 80A$
0040 DISP "ANFANGSWERT, ENDWERT, SCHRITTWEITE";
0050 INPUT A,B,C
0060 DISP "AUSZUDRUCKENDER STRING";
0070 RKB A$
0080 FOR I=A TO B STEP C
0090 PRINT A$;I;
0100 NEXT I
0110 END

END OF LISTING
```

9.2 STRUKTUR EINES TEXT - FILES

Ein Text - File besteht aus einer Folge von Zeilen. Jede Zeile besteht aus einer Zeilennummer, gefolgt von Zeichen aus dem Zeichenvorrat des P6060 (siehe Anhang). Jede Zeile enthält maximal 80 Zeichen (einschließlich der Zeilennummer).

Beispiel eines Text-Files:

```
TEXT
AUTO#
10 Beispiel fuer ein Textfile
20
30 Ein beliebiger Text wird mit Zeilennummern eingegeben.
40 In der gleichen Art wie bei Programmfiles
50 koennen Zeilen geaendert, geloescht oder hinzugefuegt werden.
60
70 Wird bei der Ausgabe als letzter Parameter des Befehles LIST
80 ein '%' eingegeben, so erfolgt der Ausdruck
90 des Textes ohne Zeilennummern.
100
110 Die Speicherung eines Textes auf einer Diskette erfolgt wie
120 bei Programmen mit dem Befehl SAVE.
130
140 Ebenso wird ein auf Diskette gespeichertes Textfile mit dem Befehl
150 OLD in den Arbeitsspeicher geladen.
160
```

Text-Files werden - um nur einige Beispiele zu nennen - für Sprachanalysen, Aufbereitung von Dokumenten und zur Erstellung von kommerziellen und administrativen Bibliotheken verwendet.

Text-Files können außerdem in der Programmierung verwendet werden :

1) Als sequentielle Datenfiles, die vom Programm gelesen werden können. Die Zeilen dieses Text-Files können nur als Strings (von max. 80 Zeichen) eingelesen werden, die auch die 4-stellige Zeilennummer enthalten.

2) Zur Erstellung von Programmen, Unterprogrammen oder zur Definition von Funktionen, die als "Quellenprogramme" auf Disketten gespeichert werden und später in eine ausführbare Form übersetzt werden. (Siehe die Anweisungen COMPILE und LINK.)

9.3 EINGABE EINES PROGRAMMES

(siehe auch Kapitel 7.12 : Struktur eines BASIC-Programmes).

9.3.1 Vorbereitung des Hauptspeichers

Befindet sich im Hauptspeicher bereits ein Programm oder Text-File und soll ein neues Programm über die Tastatur eingegeben werden, so sind die Tasten NEW und EOL zu drücken.

9.3.2 Zeilennumerierung

Die Eingabe eines Programmes erfolgt Zeile für Zeile.

Die Zeilennummern (1-9999) sind sinnvollerweise in aufsteigender Reihenfolge einzugeben, jedoch kann mit jeder beliebigen Nummer begonnen werden und in beliebigen Abständen numeriert werden.

Lücken in der Zeilennumerierung erleichtern das spätere Einfügen von Zeilen.

Mit der Anweisung AUTO [, Zeilennr.] [,Abstand]

können Zeilennummern automatisch vorgegeben werden.

Standardwerte : 10, 10

9.3.3 Programmeingabe und Syntaxkontrolle

Jede Zeile wird durch Drücken der Taste (EOL) abgeschlossen und sofort vom System analysiert. Ist die Zeile syntaktisch korrekt (siehe Kap. 8), so wird sie sofort übersetzt und in den Arbeitsspeicher übernommen.

Bei fehlerhafter Syntax wird die Zeile vom System nicht übernommen und auf dem Display erscheint eine entsprechende Fehlermeldung (siehe Anhang).

Die fehlerhafte Zeile blieb im Tastaturpuffer gespeichert und kann mit der Taste RECALL zur Korrektur ins Display zurückgeholt werden. Dabei zeigt der Pointer auf eine der möglichen Fehlerstellen. Nach der Korrektur kann die Zeile mit der Taste (EOL) in den Arbeitsspeicher übernommen werden.

9.4 EINGABE EINES TEXTFILES

Ein Textfile kann nur eingegeben werden, wenn vorher die Anweisung TEXT (EOL) erfolgte. Auch ein Textfile muß zeilenweise eingegeben werden, wobei die Zeilennumerierung wie unter 9.3.2 beschrieben erfolgt. Jede Zeile wird durch Drücken der Taste (EOL) dem Arbeitsspeicher übergeben. Dabei erfolgt keine syntaktische Kontrolle und Fehlermeldungen sind nicht möglich, da in einer Textzeile alle ISO-Zeichen gemäß Anhang verwendet werden können.

9.5 SPEICHERN VON PROGRAMMEN ODER TEXTFILES AUF DISKETTEN

Wird der P6060 ausgeschaltet, so geht der Inhalt des Arbeitsspeichers verloren. Deshalb ist es wichtig, Programme und Textfiles nach der Eingabe möglichst rasch auf einer Diskette zu speichern. Sie werden dazu mit einem Namen versehen. Eine neue Diskette muß für die Aufnahme von Bibliotheken gemäß Kapitel 5 vorbereitet sein.

Die Anweisung SAVE name (genaues Format siehe Kap. 6.3 Systembefehle) führt die gewünschte Speicherung auf der Diskette durch. Der Inhalt des Arbeitsspeichers bleibt unverändert.

Beispiel :	SAVE U, TEST 1	Speicherung des Inhaltes des Arbeitsspeichers unter dem Namen TEST 1 auf die Anwenderdiskette

9.6 KORREKTUR EINES PROGRAMMES (EDITING)

9.6.1 Hilfstasten und Befehle für das Editing

Die Funktion der für das Editing verwendbaren Tasten und Befehle wurde in Kapitel 1 bzw. Kapitel 6 beschrieben.

9.6.2 Editing am Beispiel eines Programmes

Um die Technik des Editing besser zu verstehen, betrachten wir das folgende Programm zur Lösung von quadratischen Gleichungen.

Im Beispiel sind die wesentlichsten Punkte am linken Rand mit Nummern gekennzeichnet:

```
1.    NEW
2.    AUTO#
3.    10 DIZP "GIB DIE KOEFFIZIENTEN EIN";
      ERROR  102
      10 DISP "GIB DIE KOEFFIZIENTEN EIN";
      20 INPUT A,B,C
      30 DISP
      40 A1=0
      50 A2=0
4.    AUTO# 30
      30 PRINT
      40 PRINT
      50 PRINT "DIE GLEICHUNG:";A;"X↑2 +";B;"X +";C;"= 0 HAT ";
      60 IF A=0 THEN 280
      70 R=-B/(2*A)
      80 D=R*R-C/A
      90 X1=X2=R
      100 IF D>0 THEN 210
      110 IF D<0 THEN 160
      120 PRINT "GLEICHE LOESUNGEN"
      130 PRINT
      140 PRINT "X1=X2=";X1
      150 GOTO 390
      160 D=SQR(-D)
      170 PRINT "KOMPLEXE LOESUNGEN"
      180 PRINT
      190 PRINT "X1 - i*I1=";X1;"-i";D;" X2 + i*I2=";X2;"+ i";D
      200 GOTO 390
      210 D=SQR(D)
      220 X1=X1-D
      230 X2=X2+D
      240 PRINT "ZWEI LOESUNGEN"
      250 PRINT
      260 PRINT "X1=";X1;" X2=";X2
      270 GOTO 390
      280 IF B=0 THEN 340
      290 X1=-C/B
      300 PRINT "EINE LOESUNG"
      310 PRINT
      320 PRINT "X1=";X1
      330 GOTO 390
      340 IF C=0 THEN 380
      350 PRINT "KEINE LOESUNG"
      360 PRINT
      370 GOTO 390
      380 PRINT "EINE UNBESTIMMTE LOESUNG"
      390 DISP "EINE ANDERE GLEICHUNG";
      400 RKB A$
      410 IF PRINT$<>"NEIN" THEN 10
      ERROR  113
5.    410 IF A$<>"NEIN" THEN 10
      420 END
```

1 Man gibt die Anweisung NEW ein, da sich im Hauptspeicher bereits ein Programm befindet.

2 Man gibt die Anweisung AUTO # ein, die eine automatische Zeilennumerierung mit einem Abstand von 10 beginnend bei 10 bewirkt. Auf dem Display wird die 10 angezeigt.

3 Das System meldet im Display einen syntaktischen Fehler (siehe Anhang). Das Schlüsselwort DISP wurde nicht korrekt eingegeben. Drückt man die Taste RECALL , so erscheint im Display:

10 DI$_o$ZP "GIB DIE KOEFFIZIENTEN EIN"

Drückt man die Taste [→] so erscheint im Display

10 DIZ$_o$P "GIB DIE KOEFFIZIENTEN EIN"

Drückt man CHAR DELETE , so erscheint im Display

10 DI$_o$P "GIB DIE KOEFFIZIENTEN EIN"

Nach drücken von S erscheint im Display

10 DIS$_o$P "GIB DIE KOEFFIZIENTEN EIN"

Mit EOL wird die Korrektur der Zeile beendet.

4 Die Zeile wurde angenommen und in den Arbeitsspeicher übertragen. Jetzt kann mit der Eingabe der nächsten Anweisung fortgefahren werden. In der Zeile 30 hat der Operator DISP und in Zeile 40 und 50 A1=0 bzw. A2=0 eingetastet. Diese beiden Zeilen sollen überschrieben werden. Man drückt CLEAR RECALL mit SHIFT und die Zeile im Tastatur-Puffer wird gelöscht und der Pointer auf die erste Position gesetzt. Die automatische Zeilennumerierung wurde unterbrochen.
Man drückt AUTO# 30 und EOL und am Display wird die Nummer 30 angezeigt; die automatische Zeilennumerierung ist wieder hergestellt. Nun können die neuen Zeilen eingegeben werden und die bisherigen Zeilen 30 bis 50 werden überschrieben.

5 Die Anweisung END wurde eingegeben.

Man drückt CLEAR / RECALL mit SHIFT : die automatische Zeilennumerierung ist unterbrochen. Das System ist im Command-Mode: Man kann nun jeden beliebigen Befehl eingeben. Das Programm liegt in ausführbarer Form vor und befindet sich im Arbeitsspeicher; es kann nun ausgeführt oder geändert werden.

6 Es soll nachträglich noch die Anweisung für eine Zeilenschaltung eingefügt werden. Die Eingabe lautet:

```
385 PRINT EOL
```

Diese Zeile wird im Arbeitsspeicher zwischen der Zeile 380 und der Zeile 390 eingefügt.
Drückt man RES (Resequence), so werden die Zeilen des Programmes nochmals durchnumeriert mit einem Abstand von 10 beginnend mit 10. Wie man in folgender Listung ersieht, werden alle Sprungziele ebenfalls automatisch neu numeriert.

Listen des Programmes: Tasten LIST und EOL drücken

6.

```
385 PRINT
RES
```

```
LIST
FILE

0010 DISP "GIB DIE KOEFFIZIENTEN EIN";
0020 INPUT A,B,C
0030 PRINT
0040 PRINT
0050 PRINT "DIE GLEICHUNG:";A;"X↑2 +";B;"X +";C;"= 0 HAT ";
0060 IF A=0 THEN 280
0070 LET R=-B/(2*A)
0080 LET D=R*R-C/A
0090 LET X1=X2=R
0100 IF D>0 THEN 210
0110 IF D<0 THEN 160
0120 PRINT "GLEICHE LOESUNGEN"
0130 PRINT
0140 PRINT "X1=X2=";X1
0150 GOTO 400
0160 LET D=SQR(-D)
0170 PRINT "KOMPLEXE LOESUNGEN"
0180 PRINT
0190 PRINT "X1 - i*I1=";X1;"-i";D;" X2 + i*I2=";X2;"+ i";D
0200 GOTO 400
0210 LET D=SQR(D)
0220 LET X1=X1-D
0230 LET X2=X2+D
0240 PRINT "ZWEI LOESUNGEN"
0250 PRINT
0260 PRINT "X1=";X1;" X2=";X2
0270 GOTO 400
0280 IF B=0 THEN 340
0290 LET X1=-C/B
0300 PRINT "EINE LOESUNG"
0310 PRINT
0320 PRINT "X1=";X1
0330 GOTO 400
0340 IF C=0 THEN 380
0350 PRINT "KEINE LOESUNG"
0360 PRINT
0370 GOTO 400
0380 PRINT "EINE UNBESTIMMTE LOESUNG"
0390 PRINT
0400 DISP "EINE ANDERE GLEICHUNG";
0410 RKB A$
0420 IF A$<>"NEIN" THEN 10
0430 END

END OF LISTING
```

Ausführung des Programmes:

```
GIB DIE KOEFFIZIENTEN EIN?
0,3,9


DIE GLEICHUNG  0 X↑2 + 3 X + 9 = 0 HAT EINE LOESUNG

X1=-3
EINE ANDERE GLEICHUNG?
JA
GIB DIE KOEFFIZIENTEN EIN?
9,3,0


DIE GLEICHUNG  9 X↑2 + 3 X + 0 = 0 HAT ZWEI LOESUNGEN

X1=-.33333333  X2=-1.0000000E-13
EINE ANDERE GLEICHUNG?
JA
GIB DIE KOEFFIZIENTEN EIN?
0,0,0


DIE GLEICHUNG  0 X↑2 + 0 X + 0 = 0 HAT EINE UNBESTIMMTE LOESUNG

EINE ANDERE GLEICHUNG?
JA
GIB DIE KOEFFIZIENTEN EIN?
1,2,3


DIE GLEICHUNG  1 X↑2 + 2 X + 3 = 0 HAT KOMPLEXE LOESUNGEN

X1 = -1*I1=-1 -ı 1.4142136  X2 = -1*I2=-1 + ı 1.4142136
EINE ANDERE GLEICHUNG?
NEIN
```

Bemerkung: Man beachte, daß auch bei Zuweisungen ohne das Schlüsselwort LET dieses im Listing erscheint.

9.6.3 Editing im Display

Das Editing gilt auch für die Programmzeilen im Display unter Zuhilfenahme der Tasten [↓] und [↑] der Anweisung FETCH.

Erhält eine Zeile im Display durch das Editing mehr als 76 Zeichen, so wird das Ausgabeformat durch Unterdrückung nicht signifikanter Leerstellen beim Listen verändert, da die Zeilennummer immer mit führenden Nullen auf 4 Stellen aufgefüllt wird.

Beispiel zum Editing im Display:

Nach FETCH 120 EOL
erscheint im Display: $_{o}$120 DISP

Diese Anweisung soll in PRINT abgeändert werden:

Man drückt SHIFT und [→]
Im Display erscheint:

120 DISP$_{o}$ d. h. der Pointer wurde ans Anweisungsende verschoben

Drückt man viermal CHAR. DELETE, so wird DISP gelöscht.

Drückt man SHIFT $\overset{\text{PRINT}}{\text{A}}$, im Display erscheint: 120 PRINT

Nach der Eingabe von EOL wird im Programm die vorige Zeile durch die eben eingegebene ersetzt.

Eine kürzere Korrekturmöglichkeit besteht in diesem Falle auch darin, daß einfach neu die Zeile 120 PRINT eingegeben wird. Durch Drücken von EOL wird die alte Zeile 120 im Arbeitsspeicher durch die neue überschrieben.

Nachträgliches Löschen einer Zeile im Programm:

SHIFT DEL.LINE 280 EOL

Zeile 280 im Arbeitsspeicher ist gelöscht.

10. DEBUGGING - MODE

10. DEBUGGING - MODE

Der DEBUGGING - MODE dient zum Testen und Prüfen von Programmen und Programmteilen. Treten während der Ausführung eines Programmes nicht behebbare Fehler auf, so ermöglicht der Übergang in den DEBUGGING-MODE die Abfrage von Variablenwerten sowie Berechnungen zur Ermittlung der Fehlerursache. Der DEBUGGING-MODE kann in diesem Falle nur durch Drücken der Taste BREAK und somit durch Übergang in den COMMAND-MODE verlassen werden. In allen anderen Fällen sind folgende Operationen verfügbar :

- Korrektur behebbarer Fehler
- Abfrage von Variablenwerten
- Zuweisung neuer Werte an Variable
- Zeilenweise Abarbeitung eines Programmes im DEBUGGING-MODE
- Neubelegung der Funktionstasten
- Fortsetzung des Programmlaufes in einer beliebigen Zeile
- Anweisung zum STOP des Programmlaufes in einer beliebigen Zeile

Daneben besteht die Möglichkeit,mit - vom Benutzer definierten - ein- oder mehrzeiligen Funktionen genauso wie mit den Standardfunktionen zu arbeiten.

10.1 ERREICHEN UND VERLASSEN DES DEBUGGING - MODE

Befindet sich das System in den Betriebsarten COMMAND-MODE oder CALCULATOR-MODE, so erreicht man den DEBUGGING-MODE durch Ausführung des Befehles PREPARE.

Befindet sich das System im RUNNING-MODE, wird der DEBUGGING-MODE erreicht durch

- Ausführung eines (programmierten) STOP-Statements.
- Drücken der Konsol-Taste STEP.
- Auffinden eines behebbaren Fehlers durch das System während eines Programmlaufes.
- Ausführung eines zuvor eingegebenen STOP - Zeilennummer - Befehles.
- Auftreten eines nicht behebbaren Fehlers.

Erwartet das System eine Tastatureingabe, erreicht man den DEBUGGING-MODE durch Drücken der STEP-Taste.

Der DEBUGGING-MODE wird verlassen durch :

- Drücken der Konsoltaste BREAK. In diesem Fall wird der COMMAND-MODE erreicht.
- Drücken der Konsoltaste CONTINUE. In diesem Fall wird die Ausführung des Programmes fortgesetzt.
- Eingabe des Befehles START Zeilennummer. In diesem Fall wird die Ausführung des Programmes bei der angegebenen Zeilennummer fortgesetzt. Es ist vom Anwender sicherzustellen, daß eine Ausführung des Programmes ab der angegebenen Zeilennummer sinnvoll ist.
- Durch Drücken der Taste STEP. Es wird genau ein Statement ausgeführt und dann wieder in den DEBUGGING-MODE zurückgekehrt. Hierdurch wird eine Abarbeitung des Programmes im Einzelschrittverfahren ermöglicht.
- Durch Drücken der Taste CONTINUE. Die Ausführung des Programmes wird fortgesetzt. Wurde während der Ausführung eines Programmes eine manuelle Eingabe durch STEP unterbrochen, so besteht die Möglichkeit, Kontrollrechnungen durchzuführen. Nach Drücken von CONTINUE muß die vom System erwartete Eingabe getätigt werden.
- Durch Drücken der Taste BREAK nach Auftreten eines nicht behebbaren Fehlers, wie zu Beginn des Kapitels schon beschrieben.

10.2 OPERATIONEN IM DEBUGGING - MODE

10.2.1 Behebbare Fehler

Unter behebbaren Fehlern verstehen wir solche, die während des Programmlaufes korrigiert werden können und nicht zum Abbruch eines Programmes führen.

Beispiele solcher Fehler sind u. a.

- eine Variable soll verarbeitet werden, ohne daß ihr zuvor ein Wert zugewiesen wurde,
- es soll die Quadratwurzel aus einer negativen Zahl gezogen werden,
- eine singuläre Matrix soll invertiert werden (d. h. die Determinate ist Null),
- eine angesprochene IPSO-Einheit ist nicht eingeschaltet
- der für die Erzeugung eines Bildes beim Plotten vereinbarte Puffer kann keine weiteren Punkte aufnehmen.

Der Benutzer hat die Möglichkeit, die Fehler durch explizite Wertzuweisungen oder Änderungen zu beheben oder nur die Taste CONTINUE zu drücken. In diesem Fall werden vom System Werte zugewiesen.

Die behebbaren Fehler sind in der Fehlerliste gesondert angeführt (Fehler 1 bis Fehler 16); zusätzlich sind dort die Werte angegeben, die vom System als Standardwerte zugewiesen werden.

Beispiel:

```
NEW                          Eingabe eines neuen Programmes
10 X = Y ↑ 2
20 PRINT
30 PRINT X
40 END
RUN
ERROR 1 IN LINE 1o           Im Display erscheint die Meldung,
                             daß ein behebbarer Fehler aufge-
                             treten ist, da die Variable Y nicht
                             definiert ist.
```

Möglichkeiten:

1) CONTINUE

```
   Ø                         Es wird der Variablen Y der Wert
                             Ø zugewiesen und Ø↑2 = Ø ausgedruckt.
```

2) Wertzuweisung durch den Benutzer

```
   Y = 2                     Direkte Wertzuweisung an Y , START 10
   START 10                  bewirkt die Fortsetzung des Programmlaufes,
   4                         es wird der Wert 2↑2 = 4 ausgegeben.
```

10.2.2 Abfrage von Variablenwerten

Werte von Variablen können dadurch abgefragt werden, daß der Name der Variablen eingegeben wird und anschliessend die Taste EOL gedrückt wird.

Beispiel:

```
NEW                     Eingabe eines neuen Programmes

10 A = 3
20 B = A↑2
30 C = B↑2
40 STOP
50 END

RUN                     Start des Programmlaufes

STOP IN LINE 40         Durch das STOP-Statement wird das
                        System in die Betriebsart des DEBUGGING-
                        MODES versetzt.

A    EOL                Es wird der Wert von A abgefragt

3                       Es wird der Wert ausgegeben

C    EOL                Es wird der Wert der Variablen C abge-
81                      fragt und ausgegeben
```

Genau wie im CALCULATOR-MODE erfolgt die Ausgabe der Werte über Display, ein zusätzlicher Ausdruck erfolgt, wenn die Taste PRINT ALL aktiviert ist.

Auch das Ausgabeformat, d.h. die Anzahl der Dezimalstellen etc. ist dasselbe wie im CALCULATOR-MODE und wird durch die Stellung des Dezimalstellenrades bestimmt. Eine genaue Beschreibung ist im Kapitel 11 enthalten.

Neben den Werten der numerischen Variablen können auch die Werte alphanumerischer Variablen abgefragt werden.

Beispiel:

```
NEW                          Eingabe eines neuen Programmes

10 A$ = "TEST"
20 B$ = "AUSGABE"
30 C$ = A$ + B$
40 STOP
50 END

RUN                          In Zeile 40 steht eine STOP - An-
STOP IN LINE 40              weisung, das System geht in die
                             Betriebsart DEBUGGING über.

A $    EOL                   Die Variablenwerte werden abge-
TEST                         fragt und ausgegeben

C $    EOL
TESTAUSGABE
```

Zu beachten ist jedoch folgendes: Ein Programm wird zeilenweise abgearbeitet, daher können nur Variable abgefragt werden, die zu diesem Zeitpunkt bereits einen definierten Wert haben, da sonst eine Fehlermeldung erfolgt. Ein Beispiel soll diesen Sachverhalt erläutern:

```
NEW                          Eingabe eines neuen Programmes

10 STOP
20 A = 5
30 END

RUN

STOP IN LINE 10              Ausführung des STOP - Statements,
A      EOL                   Übergang in den DEBUGGING - MODE
ø
ERROR 1                      Es erscheint kurz die Zahl ø auf dem
                             Display und anschließend die Fehlermeldung
                             ERROR 1, da die Variable A zur Zeit der
                             Ausführung des STOP - Statements noch
                             keine Werte zugewiesen bekommen hat. Be-
                             vor die Wertzuweisung erfolgen kann oder
                             die Taste CONTINUE gedrückt werden
                             kann, muß zusätzlich die Fehlermeldung
                             ERROR 1 gelöscht werden. Dies geschieht
                             durch Drücken der Tasten SHIFT und CLEAR.
```

Nach Abfrage der Variablenwerte wird der Programmlauf fortgesetzt durch Drücken der Taste CONTINUE u. s. w.

10.2.3 Wertzuweisung an Variable

Die Wertzuweisung an Variable geschieht durch folgende Anweisung:

Variablenname = num. Ausdruck EOL

Wertzuweisungen können an numerische und alphanumerische Variable erfolgen. Diese Variablen können globale, im Programm verwendete Variable sein, oder die lokalen Variablen Φ, Φ0 - Φ2 des CALCULATOR-MODE sein. Werte die im DEBUGGING an globale Variable zugewiesen werden, bleiben nach Wiederaufnahme des Programmlaufes durch CONTINUE etc. erhalten. Jede im Programm bereits definierte Variable sowie alle Funktionen können im numerischen Ausdruck verwendet werden.

Beispiel:

```
NEW                    Eingabe eines neuen Programmes

10 A = 3
20 B$ = "TEST"
30 PRINT A; B$
40 STOP
50 PRINT A; B$
60 END

RUN

3 TEST
STOP IN LINE 40
A = 100 EOL
B$ = "PROGRAMM" EOL
CONTINUE
100 PROGRAMM
```

10.3 RECHNEN IM DEBUGGING

Der DEBUGGING-MODE umfaßt alle Möglichkeiten, die auch im CALCULATOR-MODE zur Verfügung stehen.

Auch das Eingabeformat einer Anweisung ist das gleiche wie dort (siehe Kapitel 11).

Insbesondere gilt :

- es kann in Neugrad, Altgrad oder Bogenmaß gerechnet werden
- die Funktionstasten können im DEBUGGING belegt werden
- die Variablen Φ, Φ∅, Φ1 und Φ2 stehen zur Verfügung

Die entsprechenden Anweisungen sind die gleichen wie im CALCULATOR-MODE, es ist aber zu beachten, daß bei Fortsetzung des Programmlaufs nur noch im Bogenmaß gerechnet wird !

Zusätzlich zu den Möglichkeiten für das manuelle Rechnen sind folgende Operationen möglich :

- Verwendung aller im BASIC-Programm angeführten Variablen und nicht nur der Variablen Φ, Φ∅, Φ1 und Φ2. Beide Arten können gemischt verwendet werden
- Verwendung von Variablenwerten, die im Programm bestimmt wurden
- Verwendung von Funktionen, die im Programm durch DEFFN oder DEFFN / FNEND definiert sind

Beispiel 1 : (Bestimmung des Maximums zweier Zahlen)

```
NEW                          Eingabe eines neuen Programmes

10 DEFFNA (X, Y)
20 IF X = Y THEN 50
30 FN♣ = Y
40 GOTO 60
50 FN♣ = X
60 FNEND
```

PRE		Es wird die Preexecution durchgeführt und das System befindet sich im DEBUGGING-MODE.
FNA (50, 20) EOL 50		Es wird die Funktion FNA als Standardfunktion verwendet.
100	EOL	Der lokalen Variablen Φ wird der Wert 100,
Φ ø = 200	EOL	der lokalen Variablen Φ ø wird der Wert 200 zugewiesen.
FNA (Φ, Φ ø) EOL 200		Es wird der Wert der Funktion FNA abgefragt.

10.4 START, STOP ANWEISUNGEN IM DEBUGGING

Die Anweisung STOP ln EOL

"ln" Zeilennummer

bewirkt, daß der Programmlauf unmittelbar vor Ausführung des Statements mit der Zeilennummer "ln" unterbrochen wird. Durch diese Anweisung wird der DEBUGGING-MODE aber nicht verlassen, dazu ist wieder die Taste CONTINUE oder STEP zu drücken oder eine Anweisung START ln einzugeben.

Die DEBUGGING-Anweisung STOP In hat die gleiche Wirkung wie das BASIC-Statement

```
In STOP
```

Durch die Kombination einer START- mit einer STOP-Anweisung ist es möglich, einzelne Programmteile isoliert auszuführen.

```
STOP In 1          EOL
START In 2         EOL
```

Es wird das Programmstück zwischen den Zeilennummern In 2 (inclusive) und In 1 (exclusive) ausgeführt.

11. CALCULATOR - MODE

11. CALCULATOR - MODE

11.1 EINLEITUNG

In der Betriebsart CALCULATOR-MODE kann das System P6060 als Tischrechner verwendet werden, ohne daß Programme und Textfiles, die sich im Arbeitsspeicher befinden, dadurch berührt werden. Im einzelnen lassen sich numerische Ausdrücke berechnen, die aus numerischen Konstanten, Standardfunktionen und/oder den gespeicherten Werten zuvor durchgeführter Rechenoperationen bestehen.
An Standardfunktionen stehen dabei die Funktionen zur Verfügung, die auch in einem Basic-Programm verwendet werden können.
Für die trigonometrischen Funktionen kann zusätzlich festgelegt werden, ob mit dem Bogenmaß, mit Altgrad oder mit Neugrad gerechnet werden soll. Das Ergebnis einer Operation läßt sich abspeichern, entweder als einzelner Wert oder summiert zu den gespeicherten Werten vorausgegangener Operationen.

Das Ergebnis einer Operation wird im Display angezeigt, es kann aber auch gedruckt werden. Dazu ist zuvor die Taste PRINT ALL zu drücken. Das Format, in dem die Werte ausgegeben werden, wird bestimmt durch das Dezimaistellenrad auf der Konsole.
Ebenso wie durch ein Basic-Programm lassen sich die Funktionstasten auch im CALCULATOR-MODE belegen, die Standardbelegung kann danach wieder durch den Befehl LDKEYS erreicht werden.

11.2 ERREICHEN DES CALCULATOR MODES

Der CALCULATOR-MODE wird erreicht durch Drücken der Taste CALC MODE auf der Konsole, der CALCULATOR-MODE wird verlassen durch nochmaliges Drücken dieser Taste oder aber durch Eingabe eines Systembefehls.

Im einzelnen sind folgende Eingaben zulässig :

- Drücken der Taste CALC / MODE ; der CALCULATOR-MODE wird verlassen
- Eingabe eines System-Befehles : der CALCULATOR-MODE wird verlassen und der Befehl wird durchgeführt
- Eingabe einer CALCULATOR-MODE-Anweisung :
  - Anweisung, daß bei der Berechnung trigonometrischer Funktionen das Argument als Bogenmaß, Alt- oder Neugrad interpretiert werden soll
  - Anweisung zur Belegung der Funktionstasten
  - Anweisung zur Berechnung eines numerischen Ausdrucks

Nicht akzeptiert werden :

- Basic-Statements
- Zeilennummern vor einer der oben angeführten Anweisungen
- Variablennamen, die von Φ, Φ0, Φ1 oder Φ2 verschieden sind

Alle Anweisungen im CALCULATOR-MODE werden durch Drücken von EOL bzw. der Taste SUM abgeschlossen (außer Drücken der Taste CALC-MODE).

11.3 FESTLEGEN DES ARGUMENT-TYPS TRIGONOMETRISCHER FUNKTIONEN

Die Festlegung, ob im Bogenmaß, in Altgrad oder in Neugrad gearbeitet wird, geschieht durch eine der folgenden Anweisungen :

- SRAD	es wird im Bogenmaß gerechnet
- SDEG	es wird in Altgrad gerechnet
- SGRAD	es wird in Neugrad gerechnet

Nach Einschalten des CALCULATOR-MODES wird "SRAD" angenommen, d.h. die Argumente trigonometrischer Funktionen werden als Bogenmaß interprtiert.

11.4 ANWEISUNGEN ZUR BELEGUNG DER FUNKTIONSTASTEN

Das Format der Anweisung ist das folgende :

FKEY#n, String-Konstante

- "n" : positive ganze Zahl, 1≤n≤16
- "String-Konstante" : beliebige Folge von ISO-Zeichen

Wie in der Basic-Anweisung "FKEY" kann auch hier ein EOL durch einen Doppelpunkt am Ende des Strings codiert werden, so daß das Drücken der EOL - Taste entfällt, wenn die Funktionstaste gedrückt ist.

Bemerkung : Die FKEY-Anweisung darf keine Zeilennummer enthalten. Die ursprüngliche Standardbelegung wird durch Eingabe des Befehls LDKEYS erreicht. Dabei wird der CALCULATOR-MODE verlassen und der COMMAND-MODE erreicht.

Beispiel : Die Standardbelegung der Funktionstasten sei :

```
F1 : STANDARD
F2 : BELEGUNG

CALC-MODE
FKEY#1, CALCULATOR-MODE

F1  F2
CALCULATOR-MODE-BELEGUNG        Die modifizierte Belegung wird ange-
                                zeigt.

FKEY#1, 1*1:                    Nach Drücken der Funktionstaste wird
F1                              der String 1*1 als num. Ausdruck
                                interpretiert, ausgerechnet das Er-
1.0000000000000                 gebnis angezeigt.
```

Nach der Anweisung "SDEG" wird in Altgrad gerechnet, dabei sind die Winkel in dezimaler Form, z. B. 257 Grad (und nicht in der Form Grad, Minute, Sekunde) einzugeben. Nach der Anweisung "SGRAD" wird in Neugrad gerechnet, auch hier müssen Winkel in dezimaler Form eingegeben werden. Die Verwendung von Alt- oder Neugrad im CALCULATOR-MODE hat keinen Einfluß auf die Interpretation der Argumente trigonometrischer Funktionen in einem Basic-Programm, dort wird nur im Bogenmaß gerechnet.

Für den Zusammenhang von Altgrad, Neugrad und Bogenmaß gilt :

100 Neugrad = 90 Altgrad = /2
= 3, 1415.....

Beispiel :

```
CALC MODE

COS (PI)
-1. 0000000000000

SDEG
COS (180)
-1. 0000000000000

SGRAD
COS (200)
-1. 0000000000000

SRAD
COS (PI)
-1. 0000000000000
```

- num. - exp.	SUM : Der Wert des numerischen Ausdrucks wird berechnet und zum aktuellen Wert der Variablen Φ addiert.
- RESULT n =	num. exp. EOL Der Wert des numerischen Ausdrucks wird berechnet und der Variablen Φ n zugewiesen.
- RESULT n =	num. exp. SUM Der Wert des Ausdrucks wird berechnet und der Variablen Φ n zugewiesen. Zusätzlich wird der Wert zum aktuellen Wert der Variablen Φ addiert.

Wird für "n" kein Wert angegeben:

RESULT = num. exp. SUM ,

wird der Wert des Ausdrucks zum Wert von Φ addiert.

Die Variablen werden durch eine der folgenden Anweisungen neu initialisiert, d. h. es wird ihnen der Wert Ø zugewiesen:

- Ø	EOL :	es wird Φ = Ø gesetzt
- 0	RESULT n = Ø EOL :	es wird die Variable Φ n = Ø gesetzt
- CALC MODE	CALC MODE	durch Aus- und Einschalten des CALCULATOR-MODE werden alle lokalen Variablen auf Ø gesetzt.

Numerische Ausdrücke im CALCULATOR-MODE können enthalten :

- numerische Konstante
- lokale Variable
- Standardfunktion (SIN, ...)
- Operationszeichen (+, *, -, /, ↑)

Die Reihenfolge und Priorität in der Berechnung ist die gleiche wie in einem BASIC - Programm, sie können durch Setzen von Klammern beeinflußt werden.

LDKEYS
F1 F2
STANDARDBELEGUNG

Die Standardbelegung wird wieder hergestellt und angezeigt. Der CALCULATOR-MODE wird verlassen.

Bemerkung:

Die Summe der Länge der Strings, mit denen die Funktionstasten belegt sind, darf 238 nicht überschreiten. Das gleiche gilt für die Länge der Strings der Standard-Belegung.

11.5 LOKALE VARIABLE UND NUMERISCHE AUSDRÜCKE

Eine Anweisung zur Berechnung numerischer Ausdrücke hat folgende Form:

num. Ausdruck
Φ [n]= num. Ausdruck

- "Φ" : Das Symbol ist der Taste RESULT zugeordnet
- "num. Ausdruck" : numerischer Ausdruck
- "n" : eine der Zahlen 0, 1 oder 2

Φ, Φ Ø, Φ 1, Φ 2 sind die Namen lokaler Variabler.
Diesen Variablen können die Resultate der im CALCULATOR-MODE durchgeführten Rechnungen als Wert zugewiesen werden.

Eine Anweisung im CALCULATOR-MODE kann durch die Taste EOL oder SUM abgeschlossen werden, damit ergeben sich 4 mögliche Formate :

- num. - expr. EOL

der Wert des Ausdrucks "num. -expr." wird berechnet und der lokalen Variablen Φ zugewiesen.

11.5.1 Zahlendarstellung

Wie in einem Basic-Programm werden numerische Werte auch im CALCULATOR-MODE intern im Gleitkomma-Format (halblogarithmische Darstellung) dargestellt. Die Rechnung und Darstellung erfolgt in doppelter Genauigkeit. Jeder Zahl Z wird eine Mantisse m und ein Exponent n zugeordnet:

$$Z = m \cdot 10^n$$

"m" : 13-stellige Mantisse der Form

X. YYYYYYYYYYYY

$1 \leq X \leq 9$, $0 \leq Y \leq 9$

"n" : ganze Zahl, $-99 \leq n \leq 99$

Eine Zahl kann als ganze Zahl oder als Dezimalzahl eingegeben werden; sie darf bis zu 13 Ziffern (ohne Dezimalpunkt und Vorzeichen) enthalten:

248 -5 1234.567890123

Eine weitere Form der Eingabe ist das Gleitkomma-Format :

nEa

"n" : Dezimalzahl mit maximal 13 Ziffern

"a" : Ganze Zahl zwischen -98 und +99

5.698E2 -678.3456E4 2E -27

Da alle eingegebenen Zahlen durch das System in die halblogarithmische Darstellung überführt werden, ist die gewählte Eingabeform für die Berechnung unerheblich.

11.5.2 Standardfunktionen

Die im CALCULATOR-MODE verwendbaren Anweisungen sind die gleichen wie in der Programmierung mit BASIC . Eine Liste der Funktionen ist in Kapitel 8 angegeben.

11.6 VERARBEITUNG DER ANWEISUNGEN

Wird die Eingabe einer Anweisung durch EOL oder SUM abgeschlossen, wird die Anweisung syntaktisch überprüft. Ein eventueller Syntax-Fehler wird angezeigt, die Tastatur ist mit Ausnahme der Tasten RECALL und CLEAR gesperrt.

Durch Drücken der Taste RECALL wird die fehlerhafte Anweisung im Display sichtbar, die Stellung des Printers gibt eine der möglichen Fehlerpositionen an. Die Fehler können mit den Editing-Routinen korrigiert werden. Die korrigierte Anweisung kann dann wieder durch EOL oder SUM eingegeben werden. Wird CLEAR gedrückt, wird die fehlerhafte Anweisung gelöscht. Zur Ausführung gelangen nur syntaktische korrekte Anweisungen.

11.7 AUSGABE DER ERGEBNISSE

Im Display sind sichtbar

- alle Eingaben
- alle Fehlermeldungen
- die Ergebnisse

Wird RESULT gedrückt, erscheint im Display das Zeichen "Φ" als Symbol der lokalen Variablen Φ.

Die Werte der lokalen Variablen können durch folgende Anweisungen abgefragt werden:

- RESULT EOL : es wird der aktuelle Wert der Variablen Φ ausgegeben, der Wert bleibt unverändert.
- RESULT n EOL : es wird der aktuelle Wert der Variablen Φ n ausgegeben, der Wert von Φ wird durch den Wert von Φ n überschrieben, der Wert von Φ n bleibt unverändert.
- RESULT n = RESULT n EOL : es wird der Wert von Φ n ausgegeben, der Wert aller Variablen (incl. Φ) bleibt unverändert.

Das Format, in dem Zahlenwerte ausgegeben werden, wird bestimmt durch die Stellung des Dezimalstellenrades auf der Konsole:

- Position ø : Der Wert wird gerundet und ganzzahlig ausgegeben
- Position n : Der Wert wird (gerundet) als Dezimalzahl mit n Stellen nach dem Dezimalpunkt ausgegeben.
- Position Flt: Der Wert wird in Gleitkomma-(halblogarithmischer) Darstellung ausgegeben
- Position St : Der Wert wird im Standardformat ausgegeben (siehe Kapitel 8)

Hat der auszugebende Zahlenwert mehr als 16 Stellen (incl. der Stellen für das Vorzeichen und den Dezimalpunkt), erfolgt die Ausgabe im Standardformat unabhängig von der aktuellen Stellung des Dezimalstellenrades.

12. PROGRAMMIERUNG PERIPHERER EINHEITEN

12. PROGRAMMIERUNG PERIPHERER EINHEITEN

12.1 ALLGEMEINES

Dieses Kapitel beschreibt die logischen Komponenten (Anwenderprogramm, Input/Output-Kanal und periphere Einheit), die den Datenaustausch zwischen der CPU der P6060 und der externen Einheit ermöglichen. Ohne auf spezielle periphere Einheiten einzugehen, werden zunächst die grundlegenden Funktionen und Operationen erläutert.

Dies vermittelt dem Leser die Arbeitsweise, das Konzept für die Programmierung der Peripherie und erleichtert das Verständnis der BASIC-Anweisungen.

12.1.1 Logische Komponenten für den Datentransfer

Die logischen Komponenten, die an einem Datenaustausch beteiligt sind, sind :

- das Anwenderprogramm im Arbeitsspeicher
- der Input/Output-Kanal (I/O-Kanal)
- die periphere Einheit

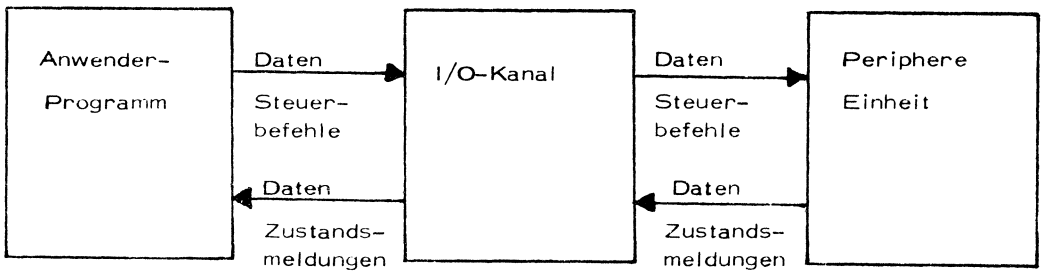


Bild 12.1

Jede der aufgeführten Komponenten führt bestimmte Funktionen aus, von denen nur die grundlegenden beschrieben werden sollen.

- Das Anwenderprogramm teilt dem System durch BASIC-Anweisungen mit, welche I/O-Operationen durchgeführt werden sollen, in welcher Folge und auf welche Art und Weise.
- Der I/O-Kanal führt die im Programm festgelegten I/O-Operationen durch.
- Die periphere Einheit empfängt die durch den I/O-Kanal übermittelten Daten oder sendet an die Zentraleinheit (CPU) die vom Programm verlangten Daten. Ferner sendet sie der CPU Zustands- und Rückmeldungen.

In den folgenden Abschnitten werden die einzelnen logischen Komponenten beschrieben. Begonnen wird mit der Beschreibung der peripheren Einheit, dann des Anwenderprogramms und schließlich des I/O-Kanals. Diese Reihenfolge der Darstellung läßt die genauen Zusammenhänge, das Zusammenspiel P6060 und Peripherie erkennen.

12.1.2 Periphere Einheit

Eine periphere Einheit besteht im allgemeinen aus drei Teilen :

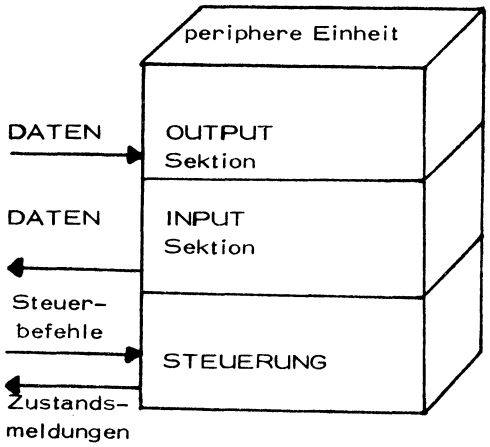


Bild 12.2

Dabei beziehen sich die Begriffe 'OUTPUT' und 'INPUT' auf die CPU des P6060:

'OUTPUT' bedeutet die Übermittlung von Daten von der CPU an die periphere Einheit, 'INPUT' bedeutet, daß Daten von der peripheren Einheit an die CPU übermittelt werden.

Eine periphere Einheit kann auch Steuer- oder Prüfbefehle erhalten (z. B. Anweisungen zur Prüfung oder zur Positionierung eines Datenträgers). Dies erfolgt durch den Teil 'STEUERUNG'. Dieser sendet auch Meldungen an die CPU über den Zustand der peripheren Einheit, ferner Meldungen darüber, ob die gewünschte Operation fehlerfrei durchgeführt worden ist.

Die in einer BASIC-Anweisung angesprochene Peripherie wird durch den Operanden 'per.-Einh.' festgelegt; 'per.-Einh.' ist die Adresse der Peripherie.

Die Adresse ist eine ganze Zahl, sie ist die Dezimaldarstellung einer 8-stelligen Binärzahl.

Durch die einzelnen Bits wird der Kanal, die periphere Einheit und die Art der Operation (INPUT oder OUTPUT) definiert. Es gilt :

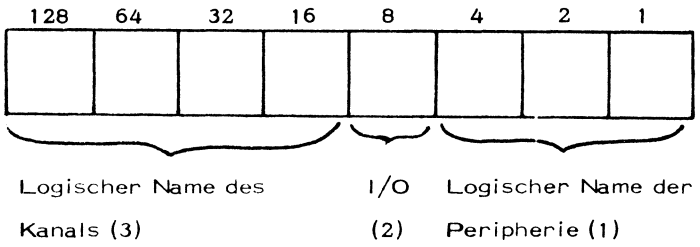


Bild 12.3

(1) Logischer Name der Peripherie :

Dieser wird durch den Wert der letzten 3 Bits festgelegt, es können also bis zu 8 logische Namen festgelegt werden. Hat eine periphere Einheit 2 Datenträger (CTU 1000/1010), so hat jeder Datenträger einen eigenen logischen Namen d. h., wird als eigene periphere Einheit betrachtet.

(2) I/O :

Der Wert des 4.-ten Bits legt fest, ob an die Einheit gesendet wird (OUTPUT : Wert = 1) oder ob von der Einheit Daten emfpangen werden sollen (INPUT : Wert = Ø). Beispiel : CTU 1000 mit 2 Stationen hat je zwei INPUT und zwei OUTPUT Sektionen, damit insgesamt 4 logische Namen.

(3) Logischer Name des Kanals :

Da 16 Bitkombinationen möglich sind, können 16 verschiedene I/O-Kanäle definiert werden.

Damit ergibt sich, daß bis zu 256 Adressen von peripheren Einheiten festgelegt werden können, aufgeteilt auf 16 Kanäle, die Kanalnamen gehen von Ø bis 15.

Für jeden Kanal lassen sich bis zu 16 Sektionen definieren :
8 für INPUT und 8 für OUTPUT.

Die Einheiten, die an dem Kanal Ø angeschlossen sind, bekommen als Adressen die Zahlen Ø bis 15, die des Kanals 1 die Zahlen 16 bis 31, die des Kanals 15 die Zahlen 240 bis 255.

Schließlich sei bemerkt, daß einige periphere Einheiten über einen eigenen Puffer verfügen, in dem die von der CPU gesendeten Daten gespeichert werden, bevor sie auf den Datenträger geschrieben werden, bzw. in den die vom Datenträger gelesenen Daten gespeichert werden, bevor sie an die CPU geschickt werden. Dadurch wird erreicht, daß der I/O-Kanal selbst nur sehr kurze Zeit belegt ist :

Die Dauer der eigentlichen Datenübertragung ist unabhängig von der Zeit, die die periphere Einheit zum Aufzeichnen oder Lesen der Daten vom Datenträger benötigt. Damit wird eine rationelle, optimale Nutzung des I/O-Kanals ermöglicht.

12.1.3 I/O-Kanal

Der P6060 kann bis zu 16 I/O-Kanäle für den Datenaustausch steuern. Ein I/O-Kanal besteht aus mehreren Hardware-Komponenten (Teile des Arbeitsspeichers, logische Schaltkreise der CPU und Interface), diese werden von Soft- und Firmware-Komponenten gesteuert.

Jeder Kanal besteht aus :

- Puffer
- 16 Zustandsregistern
- Kanalsteuerung

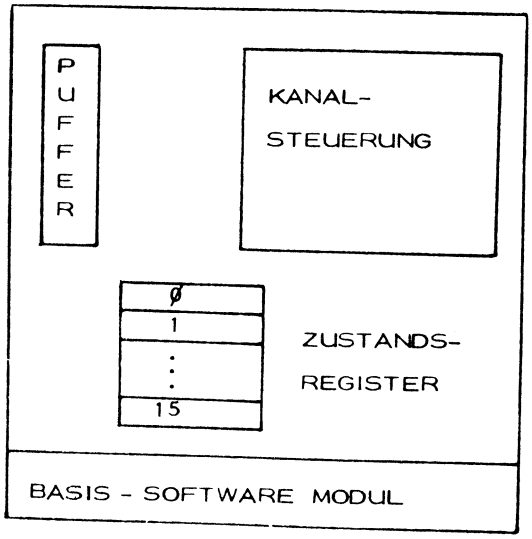


Bild 12.4

12.1.3.1 Puffer

Der Puffer ist Teil des Arbeitsspeichers, seine Größe wird durch die BASIC-Anweisung BUFFER festgelegt. Er wird dem I/O-Kanal zum Datenaustausch mit peripheren Einheiten zugewiesen.

12.1.3.2 Zustandsregister

Jedes Zustandsregister besteht aus 2 Bytes (= 16 Bits) des Arbeitsspeichers. In diesem Register legt die KANALSTEUERUNG Informationen über den Zustand des Kanals und der peripheren Einheit, über das Ergebnis einer Übertragung und über die Interpretation eines empfangenen Signals ab. Da an einen Kanal je 8 INPUT und OUTPUT Sektionen angeschlossen werden können, hat jeder Kanal 16 solcher Register.

12.1.3.3 Kanalsteuerung

Die Kanalsteuerung besteht aus Firmware und Software-Komponenten, die aus Modulen des Betriebssystems aufgebaut sind und die die gewünschten Operationen durchführen.

Jeder Kanal ist eine selbständige INPUT/OUTPUT-Einheit in dem Sinn, daß die Operationen eines Kanals unabhängig von denen eines anderen Kanals durchgeführt werden.

Da jeder Kanal über einen eigenen Puffer verfügt, können I/O-Operationen eines Kanals gleichzeitig mit I/O-Operationen eines anderen Kanals durchgeführt werden, ferner gleichzeitig mit I/O-Operationen, für eine integrierte Peripherie (Floppy, Drucker, Display), und gleichzeitig mit den übrigen Operationen der CPU. Diese Arbeitsweise bezeichnen wir als "überlappend".

Ein überlappendes Arbeiten wird erreicht, wenn die Option 'AND GO' in den BASIC-Anweisungen angegeben wird.

Zu beachten ist, daß jeder Kanal zu einem bestimmten Zeitpunkt aber nur eine periphere Einheit bedienen kann, obwohl bis zu 8 INPUT und 8 OUTPUT-Sektionen anschließbar sind.

12.2 ANWENDERPROGRAMM

12.2.1 Die BASIC-Anweisungen

Die folgenden BASIC-Anweisungen stehen für die Datenübertragung von bzw. zu einer peripheren Einheit zur Verfügung :

BUFFER : Reserviert einen Teil des Arbeitsspeichers als Puffer für den Datenaustausch mit einer Peripherie

CMD : Schickt Steuer- oder Prüfbefehle an eine Peripherie

RECEIVE : Fordert Daten von der Peripherie an, deren Adresse in der Anweisung angegeben ist

SEND : Schickt eine Zeichenfolge an die Peripherie, deren Adresse in der Anweisung angegeben ist

TEST : Der Inhalt des Zustandsregisters der Peripherie, deren Adresse in der Anweisung angegeben ist, wird in das 'ARBEITSREGISTER' übertragen, ohne daß das Ende einer laufenden I/O-Operation abgewartet wird.

WAIT : Es wird gewartet, bis der I/O-Kanal, zu dem die Peripherie, deren Adresse in der Anweisung angegeben ist, die laufende I/O-Operation abgeschlossen hat. Dann wird der Inhalt des Zustandsregisters dieser Peripherie in das Arbeitsregister geladen.

Funktion IOC (num.-Ausdr.) : Diese Funktion ermöglicht es, den Inhalt des Arbeitsregisters per Programm abzufragen.

Die Anweisung BUFFER ist eine nicht ausführbare Anweisung und kann daher an einer beliebigen Stelle im Programm stehen, dem entsprechenden I/O-Kanal wird ein Puffer der gewünschten Größe eingerichtet.

Sind für einen Kanal mehrere BUFFER-Anweisungen im Programm aufgeführt, wird der Puffer auf die maximale Größe dimensioniert :

Beispiel :

```
0010 BUFFER #0,100
0100 BUFFER #15,200
0150 BUFFER #12,10
  :
  :
  :
RUN
```

Es wird für den Kanal Ø ein Puffer der Größe 200 Bytes reserviert (die Peripherie-Adressen Ø, 12, 15 beziehen sich auf Kanal Ø).

Man beachte, daß der Puffer so groß gewählt werden muß, daß er den Record mit der größten Länge aufnehmen kann, der von einer an den Kanal angeschlossenen Peripherie aufgezeichnet oder gelesen werden soll.

Die Instruktion CMD erlaubt es, Steuerbefehle zu senden, die typisch für die entsprechende Einheit sind (es gibt nicht für alle peripheren Einheiten solche Steuerbefehle). Daneben lassen sich durch CMD Prüfbefehle an die Peripherie schicken, durch die die periphere Einheit veranlaßt wird, an die CPU Statusmeldungen zu senden.

Die Anweisung RECEIVE ermöglicht es der CPU, Daten auf zwei verschiedene Arten zu empfangen, je nachdem ob die Option "AND GO" in der Anweisung enthalten ist oder nicht. Wenn die Option "AND GO" nicht vorhanden ist, werden die Daten in den Puffer geladen und der im Statement von dort aus aufgeführten Stringvariablen zugewiesen. Ist "AND GO" angegeben, werden die Daten in den Puffer geladen, aber die Zuweisung an die Stringvariable erfolgt erst, nachdem eine weitere Anweisung auszuführen ist, die sich auf denselben Kanal bezieht.

Die Instruktion SEND erlaubt die Ausgabe von Daten auf zwei Arten, je nachdem ob "AND GO" vorhanden ist oder nicht.

Fehlt "AND GO", wird das Ergebnis des Stringausdruckes in den Puffer der CPU gegeben und von dort an die periphere Einheit gesendet; ist "AND GO" angeführt, wird der String in den Puffer geladen und das Programm fortgesetzt, parallel dazu werden die Daten an die Peripherie gesendet, die Übertragung ist mit Sicherheit beendet, sobald eine weitere Instruktion ausgeführt wird, die sich auf denselben Kanal bezieht.

Die Anweisungen TEST, WAIT und die Funktion IOC ermöglichen es, die im Zustandsregister der Peripherie, deren Adresse in der TEST- oder WAIT-Anweisung angegeben ist, abgelegte Information im Programm zu verarbeiten.

12.2.2 Das Arbeitsregister

Neben den jeweils 16 Zustandsregistern, die für jeden einzelnen I/O-Kanal eingerichtet werden, wird im Arbeitsspeicher ein weiteres Register bereitgestellt, das als ARBEITSREGISTER bezeichnet wird.

Durch die Anweisung TEST bzw. WAIT wird der Inhalt des Zustandsregisters, das zu der in der Anweisung aufgeführten peripheren Einheit gehört, in das Arbeitsregister übertragen und kann dann im Anwenderprogramm mittels der IOC-Funktion ausgewertet werden.

Da es nur ein Arbeitsregister für alle I/O-Kanäle gibt, ist zu beachten:

Bei jeder Ausführung einer WAIT oder TEST-Anweisung wird der Inhalt des Arbeitsregisters durch den Inhalt des Zustandsregisters der in der Anweisung angesprochenen Peripherie überschrieben, damit ist der alte Inhalt verloren.

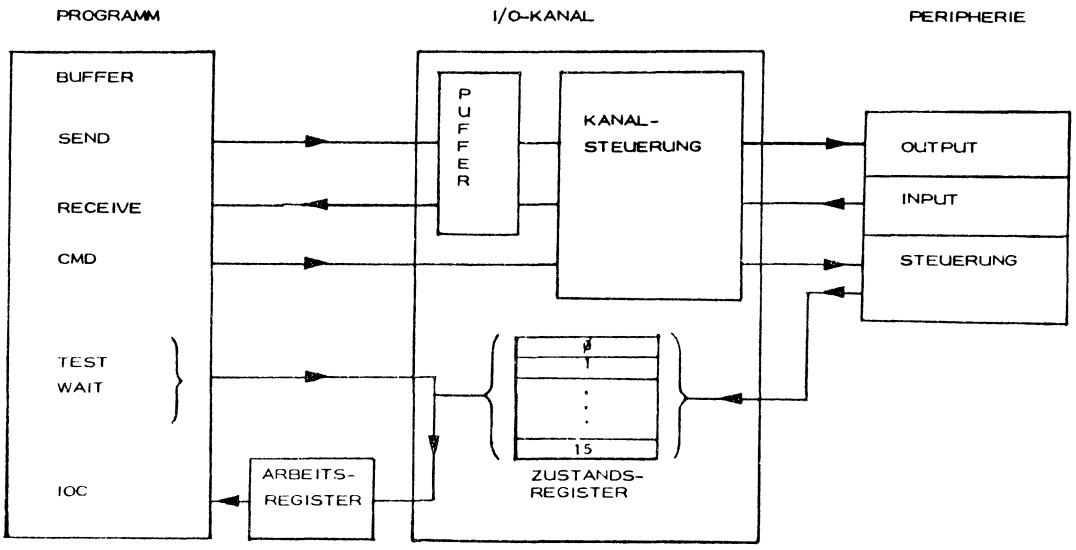


Bild 12.6 : Mittels TEST/WAIT und der IOC-Funktion wird die Fehlerbehandlung vom Programm gesteuert (Fehlermeldungen unterbleiben)

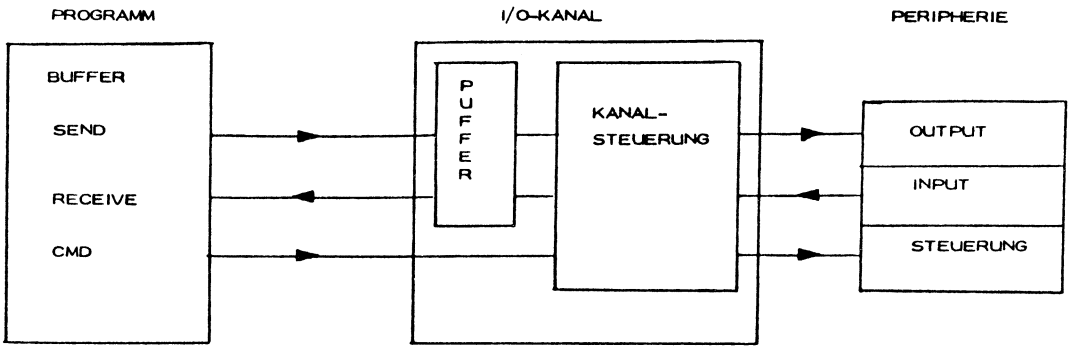


Bild 12.7 : Fehlermeldungen werden vom System angezeigt.

Tabelle 12.1

Gerundeter Wert des Argumentes num.-Ausdr.	Wenn IOC (num.-Ausdr.) = 1 ist, so gilt :	Wenn IOC (num.-Ausdr.) = ø ist, so gilt :
1	Die periphere Einheit steht vorübergehend nicht zur Verfügung	Die periphere Einheit steht zur Verfügung
2	Fehler während der Datenübertragung	Fehlerfreie Übertragung
3-5	Hängt vom Typ des I/O-Kanals ab	Hängt vom Typ des I/O-Kanals ab
6	Die periphere Einheit ist außer Betrieb. Vorausgehende Informationen sind nicht signifikant	Die periphere Einheit ist in Betrieb
7	Der I/O-Kanal ist belegt	Der I/O-Kanal ist frei
8	Zwischen CPU und peripherer Einheit findet ein Datentransfer statt	Es findet kein Datentransfer statt

Bem.: Ist der Wert von num.-exp. < 1 oder > 8, liefert IOC (.) einen Wert zwischen ø und 127, dessen Bedeutung vom Typ des I/O-Kanals abhängt.

12.3 I/O-KANAL

Nachdem der Aufbau und die Arbeitsweise einer peripheren Einheit, des I/O-Kanals und des Anwenderprogramms im Arbeitsspeicher beschrieben ist, werden nun die verschiedenen Arten beschrieben, in denen eine Datenübertragung zwischen CPU und Peripherie gesteuert werden kann. (Siehe Abb. 12.6.)

Jede der ausführbaren Anweisungen, die bis jetzt beschrieben sind, bewirkt, daß entsprechende Module des Betriebssystems geladen werden, die die gewünschte Operation durchführen.

Wie die Anweisungen ausgeführt werden, hängt davon ab, wie der I/O-Kanal gesteuert wird.

Drei Arten des Datenaustausches stehen zur Verfügung :

- Der I/O erfolgt nicht überlappt, eventuelle Fehler werden vom System angezeigt.
- Der I/O erfolgt überlappt, eventuelle Fehler werden vom System angezeigt.
- Der I/O erfolgt überlappt, die Fehlerbehandlung wird durch das Programm gesteuert.

Selbstverständlich können in einem Programm alle Arten des Datenaustausches nebeneinander verwendet werden.

12.3.1 I/O nicht überlappt, Fehlerbehandlung durch das System

Für diese Art des I/O gelten die Anweisungen :

- RECEIVE # per.-Einh., String-Var.
- SEND # per.-Einh., String-Ausdr.
- CMD # per.-Einh., Steuer-Bef.

RECEIVE #n, Stringvariable

Die Peripherie mit der Adresse n wird veranlaßt, Daten an die CPU zu senden, diese Daten werden der Stringvariablen zugewiesen. Eventuelle Übertragungsfehler während der Operation werden vom System angezeigt, die Programmausführung wird unterbrochen. Die Adresse n ist ein numerischer Ausdruck.

SEND# n, String-Ausdruck

Die CPU sendet das Ergebnis des Stringausdruckes an die Peripherie n. Eventuelle Fehler während der Operation werden vom System angezeigt, die Programmausführung wird unterbrochen. Die Adresse n ist ein numerischer Ausdruck.

CMD# n, C_1[, C_2....]

Die CPU sendet die Steuerbefehle C1, C2,..... an die Peripherie. Eventuelle Fehler während der Operation werden angezeigt, die Programmausführung wird unterbrochen. Die Adresse n ist ein numerischer Ausdruck.

Folgende Fehler werden vom System angezeigt :

- Übertragungsfehler :

  im Display wird
  ERROR 15 IN LINE ...
  angezeigt.

- Die angesprochene Einheit ist nicht angeschlossen :

  im Display wird
  ERROR 14 IN LINE ...
  angezeigt.

- Die angesprochene Einheit ist vorübergehend nicht verfügbar :

  das System wartet, bis die Einheit wieder verfügbar ist.

Es gilt also :

- Sobald die Anweisung RECEIVE ausgeführt ist, hat die Stringvariable den gelesenen String als Inhalt zugewiesen bekommen und kann verarbeitet werden.
- Das System wartet, bis die Datenübertragung abgeschlossen ist und führt erst dann die nächste Anweisung aus.
- Eventuell auftretende Fehler werden während der Ausführung angezeigt, der Programmablauf wird unterbrochen.

12.3.2 I/O überlappt und Fehlerbehandlung durch das System

Für diese Art des Datenaustausches stehen folgende ausführbare Basic-Anweisungen zur Verfügung :

- RECEIVE #per.-Einh., String-Var. AND GO
- SEND #per.-Einh., String-Ausdr. AND GO
- CMD #per.-Einh., Steuer-Bef. AND GO

RECEIVE #n, Stringvariable AND GO

Die von der Peripherie gesendeten Daten werden in den Puffer des I/O-Kanals geschrieben.

Die Zuweisung an die String-Variable erfolgt erst, wenn eine weitere Anweisung ausgeführt wird, die sich auf denselben I/O-Kanal bezieht.

Das heißt, das System veranlaßt den Empfang der Daten, wartet aber nicht ab, bis diese vollständig in den Puffer geladen sind, es wird sofort die nächste Anweisung im Programm ausgeführt.

Treten Fehler bei der Übertragung auf, werden diese vom System erst dann angezeigt, wenn eine weitere Anweisung dieselbe Peripherie anspricht.

SEND #n, String-Ausdruck, AND GO

Das System lädt den Puffer des I/O-Kanals mit dem Ergebnis des Stringausdruckes. Die Übertragung wird veranlaßt, es wird sofort die nächste Programmanweisung ausgeführt.

Die Übertragung ist sicher abgeschlossen, wenn die nächste Anweisung ausgeführt wird, die sich auf denselben Kanal bezieht.

Eventuell auftretende Übertragungsfehler werden erst angezeigt, sobald die nächste Anweisung ausgeführt werden soll, die sich auf dieselbe Peripherie bezieht.

CMD #n, C1[, C2....] AND GO

Das Programm sendet die Steuerbefehle "C1, C2,..." an die Peripherie. Eventuelle Übertragungsfehler werden erst angezeigt, sobald eine weitere Anweisung ausgeführt wird, die sich auf dieselbe Peripherie bezieht.

Es gilt also:

- Durch RECEIVE AND GO angeforderte Daten stehen erst dann im Programm zur Verfügung, wenn eine weitere I/O-Operation, die sich auf denselben Kanal bezieht, ausführt ist (das entsprechende gilt für SEND AND GO).
- Übertragungsfehler, die bei der Ausführung einer I/O-Operation auftreten, werden erst angezeigt, sobald eine weitere I/O-Operation ausgeführt wird, die sich auf dieselbe Peripherie bezieht.

Zur Erläuterung:

Fehlerbedingungen werden in Zustandsregistern, die für jede Peripherie gesondert eingerichtet werden, gespeichert. Das Zustandsregister einer Peripherie wird erst wieder geprüft, wenn das System auf diese Peripherie zugreift.

12.3.3 I/O überlappend, Fehlerbehandlung durch das Programm

Folgende(ausführbare) Basic-Anweisungen stehen für diese Art der Datenübertragung zur Verfügung :

- RECEIVE #per.-Einh., String-Var. AND GO
- SEND #per.-Einh., String-Ausdr. AND GO
- CMD #per.-Einh., Steuer-Befehl AND GO
- WAIT #per.-Einh.
- TEST #per.-Einh.
- Die Funktion IOC (X)

Die ersten drei Anweisungen arbeiten wie vorher beschrieben (AND GO bedeutet überlappende Arbeitsweise). Die Unterschiede liegen in der Behandlung eventuell auftretender Fehler.

Durch die Anweisung TEST n bzw. WAIT n, wird der Inhalt des Zustandsregisters der Peripherie mit der Adresse n in das Arbeitsregister übertragen, gleichzeitig werden die Bits im Zustandsregister, die eine Fehlermeldung verursachen könnten, = ø gesetzt. Der Zustand der Peripherie kann nun durch die Funktion IOC abgefragt werden und es kann in Abhängigkeit vom IOC-Wert im Programm verzweigt werden. In diesem Falle erfolgt keine Fehlermeldung durch das System, weder wenn die Übertragung fehlerhaft war, noch wenn eine Peripherie nicht angeschlossen ist. Der Benutzer ist selbst für die Fehlerbehandlung im Programm verantwortlich.

Wichtig ist, daß die Anweisung TEST bzw. WAIT so im Programm plaziert wird, daß sie ausgeführt wird, bevor eine andere I/O-Anweisung ausgeführt wird, die sich auf dieselbe Peripherie bezieht. Sonst werden die eventuell aufgetretenen Fehler vom System angezeigt.

Unterschiede zwischen TEST und WAIT

WAIT : Es wird gewartet, bis die laufende Datenübertragung beendet ist, anschliessend wird der Inhalt des Zustandsregisters in das Arbeitsregister übertragen.

TEST : Die Übertragung der Information aus dem Zustandsregister erfolgt sofort, ohne daß das Ende einer laufenden I/O-Operation abgewartet wird.

Durch Verwendung der IOC-Funktion kann geprüft werden, ob die Übertragung beendet ist oder nicht,und zwar durch IOC (8); ist IOC (8) = 1, ist die Übertragung noch nicht beendet, die CPU kann andere Operationen ausführen, während die Übertragung läuft. Durch eine erneute TEST-Anweisung in Verbindung mit der Abfrage von IOC (8) kann dann wieder geprüft werden, ob die Übertragung schon beendet ist usw.

Diese Möglichkeit ist besonders nützlich, wenn in einem Programm Datenübertragung über mehrere I/O-Kanäle erfolgt und man eine rationelle Ausnutzung der CPU haben will.

12.4 BASIC - ANWEISUNGEN

Im folgenden Kapitel sind die BASIC-Anweisungen für die Programmierung von peripheren Einheiten in alphabetischer Reihenfolge geordnet und ausführlich beschrieben.

Anweisung BUFFER#

Funktion : Für einen I/O-Kanal wird im Arbeitsspeicher ein Puffer (temporärer Speicher) für den Datenaustausch mit einer peripheren Einheit reserviert.

Format : BUFFER# per.-Einh., Puffergröße.
"per.-Einh." ganze Zahl zwischen Ø und 255 (einschließlich)
"Puffergröße" : ganze Zahl zwischen 1 und 32767 (byte).

Wirkung : Im Arbeitsspeicher wird für den I/O-Kanal, an den die periphere Einheit mit der Adresse "per.-Einh." angeschlossen ist, ein Puffer für den Datenaustausch eingerichtet. Die Größe in Byte wird durch den Operanden "Puffergröße" festgelegt.

Bemerkungen:

1.) Die Anweisung BUFFER ist eine nichtausführbare Anweisung. Die Anweisung kann an einer beliebigen Stelle des Programmes stehen (analog einer DIM- oder DCL-Anweisung).

2.) Enthält ein Programm mehrere BUFFER-Anweisungen die sich auf denselben Kanal beziehen, wird der Puffer mit der maximalen Größe festgelegt, die in den Anweisungen angegeben ist.

3.) Soll eine RECEIVE-Anweisung ausgeführt werden, darf die deklarierte Länge der String-Variablen, der der Pufferinhalt zugewiesen wird, nicht größer sein als die Größe des Puffers, der dem entsprechenden I/O-Kanal zugewiesen wurde.

Beispiel : Im folgenden Beispiel wird dem Kanal Ø ein Puffer von 256 Bytes, dem Kanal 1 ein Puffer von 64 Bytes zugewiesen. (Die peripheren Einheiten mit den Adressen Ø bis 15 sind an den Kanal Ø, die Einheiten mit den Adressen 16 bis 31 sind an den Kanal 1 angeschlossen.)

```
10 BUFFER #5,200
20 BUFFER #12,128
 •
 •
120 BUFFER #1,256
 •
 •
200 BUFFER #30,50
210 BUFFER #16,64
```

CMD

Anweisung CMD # (Comand)

Funktion : Sendet an eine periphere Einheit ein oder mehrere Steuerbefehle.

Format : CMD#per.-Einh., Steuer-Bef.[,Steuer-Bef.] ... [AND GO]

"per.-Einh." : Adresse der Peripherie; der (gerundete) Wert muß eine Zahl zwischen Ø und 255 (einschließlich) sein.

"Steuer-Bef." : Ganze Zahl, die für einen Steuerbefehl oder Prüfbefehl für die periphere Einheit steht.

"AND GO" : Legt fest, daß der Steuerbefehl (wenn mehrere Steuerbefehle gesendet werden, der letzte in der Reihe) überlappend mit anderen Operationen der CPU ausgeführt wird.

Wirkung :

1.) F ehlt "AND GO", werden die durch die Operanden "Steuer-Bef." spezifizierten Steuerbefehle in der Reihenfolge gesendet und ausgeführt, wie sie in der Anweisung angeführt sind. Die periphere Einheit ist durch die Adresse "per.-Einh." bestimmt.

In das Zustandsregister der Peripherie "per.-Einh." werden die Informationen übertragen, die den "Zustand" der Peripherie angeben.

Treten während der Übertragung der Steuerbefehle Fehler auf, werden diese vom System angezeigt.

2.) Ist "AND GO" angegeben, wird die Übertragung der Steuerbefehle initialisiert. Das System wartet das Ende der Übertragung und die Rückmeldung der Peripherie nicht ab, sondern die Programmausführung wird sofort mit der nächsten Anweisung fortgesetzt.

Eventuelle Fehlermeldungen werden erst angezeigt, wenn dieselbe Peripherie erneut angesprochen wird.

RECEIVE

Anweisung RECEIVE#

Funktion : Die Zentraleinheit fordert die periphere Einheit auf, Daten zu senden.

Format : RECEIVE# per.-Einh., String-Var. [AND GO]

"per.-Einh." : Numerischer Ausdruck, Adresse der Peripherie, zulässig :
0-7; 16-23; 32-39,.....
..... 240-247,

allgemein :
(n · 16) bis (n · 16 + 7), n = 0, 1, ...
...., 15

"String-Var.": String-Variable

"AND GO" : Legt fest, daß die Operation "RECEIVE" überlappt mit anderen Operationen der Zentraleinheit ausgeführt wird.

Wirkung :

1.) Fehlt "AND GO", wird von der peripheren Einheit mit der Adresse "per.-Einh." ein Datensatz (String) in den Puffer des I/O-Kanals, an den sie angeschlossen ist, übertragen : der Inhalt des Puffers wird dann der String-Variablen "String-Var." zugewiesen. Eventuell auftretende Fehler bei der Übertragung werden vom System angezeigt.

2.) Ist "AND GO" aufgeführt, wird die Übertragung eines Strings in den I/O-Puffer initialisiert. Das Ende der Übertragung wird nicht abgewartet.

Der im Puffer stehende String wird erst dann der String-Variablen "String-Var." zugewiesen, wenn die nächste I/O-Operation ausgeführt wird, die sich auf denselben I/O-Kanal bezieht. Eventuell auftretende Fehler werden erst dann vom System angezeigt, wenn eine I/O-Operation ausgeführt wird, die sich auf dieselbe periphere Einheit bezieht.

Bemerkungen:

1.) Ist "per.-Einh." keine ganze Zahl, wird der gerundete Wert genommen.

2.) Besitzt die Peripherie "per.-Einh." einen eigenen Puffer, muß vor dem RECEIVE eine Anweisung CMD ausgeführt werden, die die Informationen vom Datenträger in der Puffer der peripheren Einheit schreibt.

3.) Die deklarierte Länge der Stringvariablen "String-Var." darf nicht größer sein (in Byte) als der Puffer des I/O-Kanals, an den die periphere Einheit "per.-Einh." angeschlossen ist.

4.) Enthält der empfangene Datensatz weniger Zeichen, als die deklarierte Länge der Stringvariablen "Strng-Var." angibt, so ist IOC (5) = 1, wenn eine TEST oder WAIT-Anweisung ausgeführt wird, die sich auf denselben Kanal bezieht.

Beispiel :

1.) Der Empfang der Strings A$ und B$ erfolgt nicht überlappt, die Strings können sofort verarbeitet werden :

```
100 RECEIVE #4,A$
110 PRINT A$
 .
 .
 .
200 RECEIVE #1, B$
 .
 .
 .
300 PRINT B$
```

2.) Der Empfang von A$ und B$ erfolgt überlappt :

Der String A$ steht erst nach Ausführung der Anweisung 110 zur Verfügung, entsprechend kann B$ erst nach Ausführung der Anweisung 200 verarbeitet werden :

```
100 RECEIVE #4,A$ AND GO
110 RECEIVE #4,B$ AND GO
 .
 .
 .
200 RECEIVE #4,A$ AND GO
 .
 .
```

SEND

Anweisung SEND#

Funktion : Es wird ein String aus dem Arbeitsspeicher an eine periphere Einheit gesendet.

Format : SEND# per.-Einh., String-Ausdr. [AND GO]

"per.-Einh." : Numerischer Ausdruck, Adresse der Peripherie, zulässig :
8-15; 24-31; 40-47;...., 248-255

allgemein :
(n . 16 + 8) bis ((n + 1) . 16-1)
n = Ø, 1, 2,..., 15

"String-Ausdr.": Stringausdruck.

"AND GO" : Legt fest, daß die Übertragung überlappt mit anderen Operationen der Zentraleinheit ausgeführt wird.

Wirkung :

1.) Fehlt "AND GO", wird der Stringausdruck "String-Ausdr." in den Puffer des entsprechenden I/O-Kanals geladen und an die periphere Einheit mit dem Namen "per.-Einh." übertragen. Eventuell auftretende Fehler bei der Datenübertragung werden vom System angezeigt.

2.) Ist "AND GO" in der Anweisung angeführt, wird die Übertragung des Stringausdruckes in den I/O-Puffer initialisiert und die Programmausführung mit der nächsten Anweisung fortgesetzt.

Eine weitere I/O-Operation, die sich auf denselben I/O-Kanal bezieht, wird erst dann ausgeführt, wenn die Peripherie wieder frei ist.

Eventuelle Fehler, die bei der Übertragung auftreten, werden vom System angezeigt, sobald eine I/O-Operation mit derselben peripheren Einheit "per.-Einh." auszuführen ist.

Bemerkungen: 1.) Ist "per.-Einh." keine ganze Zahl, so wird der gerundete Wert angenommen.

2.) Wenn die periphere Einheit "per.-Einh." einen Puffer enthält, werden die von der Zentraleinheit gesendeten Daten zunächst dort gespeichert. Damit die Daten aus dem Puffer auf den Datenträger gebracht werden, muß ein Steuerbefehl gesendet werden.

3.) Die Anzahl der Zeichen des Strings (aktuelle Länge), auch das Ergebnis des Stringausdruckes, darf nicht größer sein als die Kapazität des Puffers des I/O-Kanals, an den die periphere Einheit "per.-Einh." angeschlossen ist.

Beispiel : 1.) Im folgenden Beispiel werden die Strings, die von der Peripherie 4 empfangen werden, an die Peripherie mit der Adresse 31 geschickt. Sendung und Empfang erfolgen überlappt.

```
0100 LET A$="P6060"
0110 FOR I=1 TO N STEP 1
0120 RECEIVE #4,A$ AND GO
0130 SEND #31,A$ AND GO
0140 NEXT I
```

2.) Weil die RECEIVE-Anweisung überlappt erfolgt, hat die Stringvariable A$ in der Anweisung 110 noch den Wert, der ihr in der Anweisung 90 zugewiesen wurde. Erst nach dem SEND-Befehl hat die Variable A$ den neuen Wert :

```
0080 BUFFER #4,80
0090 A$="OLIVETTI"
0100 RECEIVE #4,A$ AND GO
0110 SEND #15,A$
0120 PRINT A$
0130 END


RUN

P6060
```

TEST

Anweisung TEST #

Funktion : Der Inhalt des Zustandsregisters der peripheren Einheit wird in das Arbeitsregister übertragen, ohne daß das Ende einer eventuell laufenden I/O-Operation mit dieser Peripherie abgewartet wird.

Format : TEST # per.-Einh.

"per.-Einh.": Numerischer Ausdruck, Adresse der Peripherie

Wirkung : Die Information im Zustandsregister der peripheren Einheit "per.-Einh." wird in das Arbeitsregister übertragen.

Im Zustandsregister der Peripherie "per.-Einh." werden die beiden Bits = ø gesetzt, die die Information "außer Betrieb" und "Fehler bei Datenübertragung" enthalten. Dadurch wird verhindert, daß das System diese Fehler meldet.

Bemerkungen: 1.) Wenn "per.-Einh." keinen ganzzahligen Wert hat, wird auf die nächste ganze Zahl gerundet.

2.) Die Anweisung TEST muß ausgeführt werden, bevor eine I/O-Operation mit derselben peripheren Einheit ausgeführt wird, sonst gilt :

- Es erfolgen Fehlermeldungen.
- Wenn die periphere Einheit den Kanal belegt, wartet das System bis er frei ist.

3.) Der Inhalt des Arbeitsregisters kann durch die Funktion IOC (num.-Ausdr.) abgefragt werden.

WAIT

Anweisung WAIT #

Funktion : Es wird gewartet, bis eine laufende I/O-Operation mit der angesprochenen Peripherie beendet ist, der Inhalt des zur peripheren Einheit gehörenden Zustandsregisters wird vom I/O-Kanal in das Arbeitsregister übertragen.

Format : WAIT # per.-Einh.

"per.-Einh." : Numerischer Ausdruck, Adresse der Peripherie .

Wirkung : Die Information, die im Zustandsregister der peripheren Einheit "per.-Einh." steht, wird in das Arbeitsregister übertragen.

Im Zustandsregister der Peripherie "per.-Einh." werden die beiden Bits = ø gesetzt, die die Information "außer Betrieb" und "Fehler bei Datenübertragung" repräsentieren. Dadurch wird verhindert, daß das System solche Fehlermeldungen ausgibt.

Bemerkungen:

1.) Ist "per.-Einh." keine ganze Zahl, wird auf die nächste ganze Zahl gerundet.

2.) Die Anweisung WAIT muß ausgeführt werden, bevor eine andere I/O-Instruktion mit derselben peripheren Einheit ausgeführt wird, sonst erfolgt gegebenenfalls eine Fehlermeldung durch das System.

3.) Das Arbeitsregister kann durch die Funktion IOC (num.-Ausdr.) abgefragt werden.

12.5 Periphere IPSO-Einheiten

Im folgenden werden nur die wichtigsten, nachstehend aufgeführten, peripheren Einheiten beschrieben. Für eine vollständige Übersicht siehe das Handbuch "P 6060 - Programmierung peripherer Einheiten".

- PN 20 Lochstreifenstanzer
- LN 20 Lochstreifenleser

- CTU 1010 Magnetbandkassetteneinheit (eine Station)
- CTU 1050 Zweite Magnetbandstation (zum Anschluß an CTU 1010)
- CTU 1000 Magnetbandstation

- PR 1220 Schnelldrucker 100 Z/sec.
- PR 1230 Schnelldrucker 175 Z/sec.
- PR 1240 Schnelldrucker 300 Z/sec.

- EDITOR 4ST Ein- und Ausgabe-Schreibmaschine

Um eine der oben genannten Einheiten an den P 6060 anschließen zu können, benötigt man die Steuereinheit IPSO 6600 (IPSO-Interface). An jede Interfacekarte können bis zu 4 periphere Input- und Output-Einheiten angeschlossen werden. Da in die Basiseinheit zwei Interface-Karten gesteckt werden können, sind also bis zu 8 Input- und Output-Einheiten (gleichzeitig) anschließbar.

Die IPSO-Steuereinheit besteht aus einem Steuerprogramm innerhalb des Betriebssystems, logischen Steuerkreisen und einem Teil des Arbeitsspeichers, der als Puffer verwendet wird (die Dimension des Puffers wird durch die Anweisung BUFFER im Anwenderprogramm festgelegt). Die genannten Teile zusammen werden als IPSO-Kanal bezeichnet und ermöglichen den Datenaustausch zwischen der Zentraleinheit und einer IPSO-Peripherie.

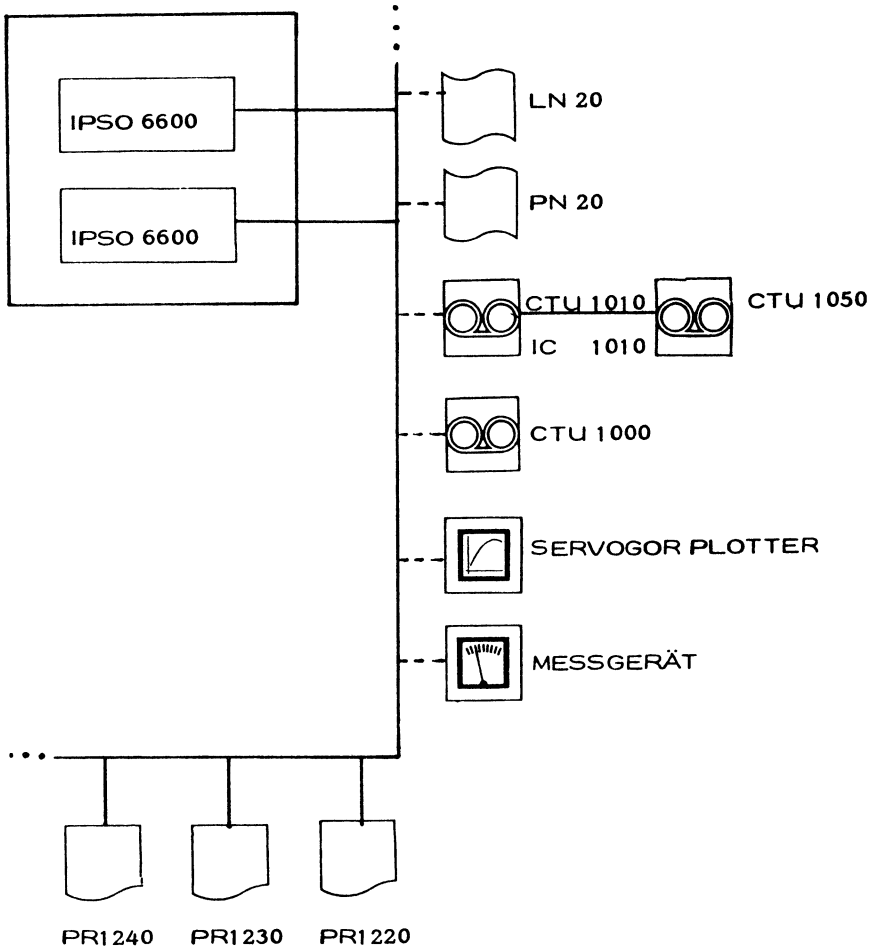

IPSO 6600
IPSO 6600
LN 20
PN 20
CTU 1010
CTU 1050
IC 1010
CTU 1000
SERVOGOR PLOTTER
MESSGERÄT
PR1240
PR1230
PR1220

Bild 12.7

12.5.1 BASIC-Anweisungen für den Kanal IPSO

Der Dialog zwischen der Zentraleinheit und einer IPSO-Peripherie erfolgt genau so, wie im Abschnitt 12.1 beschrieben.

Die beiden IPSO-Kanäle haben die logischen Namen Ø und 1, damit stehen folgende Namen für IPSO-Peripherie zur Verfügung :

Kanal Ø		Kanal 1	
INPUT	OUTPUT	INPUT	OUTPUT
Ø - 7	8 - 15	16 - 23	24 - 31

Im Abschnitt 5 werden die Standardadressen angegeben.

Folgende BASIC-Anweisungen können für den Datenaustausch verwendet werden :

BUFFER # per.-Einh. Puffergröße
CMD # per.-Einh., Steuer-Befehl,.... (AND GO)
RECEIVE # per.-Einh., String-Var. (AND GO)
SEND # per.-Einh., String-Ausdr. (AND GO)
TEST # per.-Einh.
WAIT # per.-Einh.

Für "per.-Einh." ist die Adresse der Peripherie einzusetzen. Für SEND # sind die Namen 8-15 bzw. 24-31 zulässig, für RECEIVE # die Namen Ø-7 bzw. 16-23, in den Anweisungen BUFFER, TEST und WAIT können die Namen Ø-15 bzw. 16-31 verwendet werden, je nachdem, ob die periphere Einheit an den IPSO-Kanal Ø oder 1 angeschlossen ist. Für die Anweisung CMD findet man die Adressen und Steuerbefehle in Abschnitt 5.

Die IPSO-BASIC-Anweisungen erlauben nur die Übertragung von Strings, besonders wichtig sind daher die Befehle mit String-Operationen; dazu gehören :

ASSIGN :	Der Wert eines Stringausdruckes wird einer oder (nach Aufspaltung) mehreren numerischen oder alphanumerischen Variablen zugewiesen.
BASSIGN :	Der Wert eines Stringausdruckes im internen Format wird einer oder mehreren numerischen oder alphanumerischen Variablen zugewiesen.
BBUILD :	Die Werte einer oder mehrerer Variablen werden (in internem Format) einer Stringvariablen zugewiesen.
BPAD :	Füllt einen String mit binären Füllzeichen bis zur deklarierten Länge auf.
BUILD :	Wert einer oder mehrerer Variablen werden (im Standardformat) einer Stringvariablen zugewiesen.
BUILD USING :	Der Wert einer oder mehrerer Variablen wird im definierten Format einer Stringvariablen zugewiesen.
CONVERT :	Ein String wird in eine Folge der entsprechenden ISO-Codes umgewandelt und umgekehrt.
DEPAD :	Die Füllzeichen am Ende einer Stringvariablen werden gelöscht.
PAD :	Füllt einen String mit dem angegebenen Füllzeichen bis zur deklarierten Länge auf.

Für die IPSO-Kanäle ist die Tabelle der IOC-Funktion wie folgt zu ergänzen :

IOC (3) = 1	Aufforderung an die Peripherie, betriebsbereit zu sein.
IOC (4) = 1	Die durch den letzten Steuer- oder Prüfbefehl gegebene Anweisung ist korrekt beendet.
IOC (5) = 1	Die Länge des von der peripheren Einheit gesendeten Datensatzes ist kleiner als die deklarierte Länge der Stringvariablen im RECEIVE-Statement.

Der Wert des Arguments num. -Ausdr. der IOC-Funktion wird auf die nächste ganze Zahl gerundet.

Liegt der Wert des Arguments nicht zwischen 1 und 8, ist das Resultat der IOC-Funktion gleich Ø und liefert keine Information über den Zustand der peripheren Einheit.

Bemerkung : Wird ein Drucker durch den Systembefehl CONFIGURE angeschlossen, so darf eine PRINT-Anweisung erst dann ausgeführt werden, wenn laufende I/O-Operationen mit peripheren Einheiten, die an denselben IPSO-Kanal angeschlossen sind, beendet sind. (Beispiel : eine PRINT-Anweisung darf nicht zwischen einer RECEIVE#n,... AND GO - und einer WAIT#n-Anweisung stehen.)

12.5.2 Steuerung IPSO-peripherer Einheiten

12.5.2.1 Lochstreifenstanzer PN20

Standard-Adresse : 11

Bei Installationen kann die Adresse geändert werden in :

Kanal 0 8 - 15

Kanal 1 24 - 31

Die Abfrage auf "Ende des Lochstreifens" erfolgt über IOC (4).

12.5.2.2 Lochstreifenleser LN20

Standard-Adresse : 3

Bei Installationen kann die Adresse geändert werden in :

Kanal 0 0 - 7

Kanal 1 16 - 23

12.5.2.3 Magnetbandkassetteneinheit CTU 1010 bzw. CTU 1000, CTU 1050 bzw. CTD

Standard-Adressen :

	INPUT	OUTPUT
CTU 1010 (Station I)	4	12
CTU 1050 (Station II)	6	14

Bei Installationen können die Adressen geändert werden in :

	INPUT	OUTPUT
Kanal Ø	0 - 7	8 - 15
Kanal 1	16 - 23	24 - 31

Anmerkung :

Ist die Input-Adresse X, dann muß für den Output die Adresse Y = X + 8 verwendet werden.

Steuerbefehle

Für alle Steuerbefehle ist die Input-Adresse der Magnetbandeinheit zu programmieren.

Name	Funktion	Befehls-Code
	Keine Bedeutung	0
BR	BACK RECORD, 1 Block zurückspulen	1
ER	ERASE, 20 mm auf Band löschen	2
RW	REWIND, Zurückspulen des Bandes	3
BOT	BEGIN OF TAPE, Vorlauf zur Marke BOT (gleiche Funktion wie LBR)	4
WCB	WRITE CONTROL BLOCK, Satzmarke HS1 schreiben	5
SCB	SEARCH CONTROL BLOCK, Vorlauf auf nächstes HS1	6
IN	INITIALISIEREN, LOAD BEFORE WRITE (LBR) Löschen Band bis BOT	7
CP	CLEAR POINTER, Pointer auf Null setzen	8
LP	LOCK POINTER, Ende des Records festlegen	9
RB	READ BLOCK, Lesen eines Records in CTU-Puffer	10
WR	WRITE BLOCK, Record (CTU-Puffer) auf Band schreiben	11
PP	PUT POINTER, Tabulation des Pointers im CTU-Puffer	12
IS (*)	Vorbereiten - Überschreiben eines Records	13
CC (*)	Kassettenwechsel	14
EE (*)	Satzmarke HS2 schreiben, 400 mm löschen des Magnetbandes	15
(*)	gilt nicht für CTU 1000	

Prüfbefehle

Name	Funktion	Command-Code
LD	LEADER, prüfen ob Leader (nicht bespielbarer Teil des Bandes) erreicht ist.	ø
ERR	ERROR, prüft auf Übertragungsfehler	1
COB	CONTROL BLOCK HS1, prüft, ob HS1 gelesen ist	2
END	END, prüft auf Bandende (HS2)	3
BET	BEGIN/END OF TAPE, Prüfung auf Bandanfang oder Bandende(BOT, EOT)	4
SID	SIDE B, prüft ob Seite B eingelegt ist	5
REC	RECORD, prüft ob die Kassette gegen Aufzeichnung gesperrt ist	6
IMP	Prüft ob einer der Prüfkreise auf 1 steht	7

Bemerkung : 1.) Als Adresse ist der Name der OUTPUT-Selektion im entsprechenden CMD-Befehl anzugeben.

2.) Mit IOC (4) wird das Ergebnis der Prüfung abgefragt.

Beispiel : Im folgenden Beispiel werden 4 Datensätze über die Tastatur eingegeben und auf die Kassette geschrieben; anschließend werden sie von der Kassette gelesen und ausgedruckt.

```
FILE      TAPE

0010 REM   **** CTU-1010 ****
0020 DCL 256(A$,B$)
0030 BUFFER #4,256
0040 CMD #4,4
0050 FOR I=1 TO 4 STEP 1
0060 RKB A$
0070 CMD #4,8
0080 SEND #12,A$
0090 CMD #4,9,11
0100 NEXT I
0110 CMD #4,4
0120 FOR I=1 TO 4 STEP 1
0130 CMD #4,10
0140 RECEIVE #4,B$
0150 PRINT B$
0160 NEXT I
0170 END

END OF LISTING
```

12.5.2.4 Schreibmaschine EDITOR 4ST

Standard-Adresse : 8

Zusätzlich zum Quellenprogramm, welches auf dem integrierten Drucker arbeitet ist nach jeder Print-Anweisung mit Hilfe eines Command-Befehles (CMD#8, 2) eine Zeilenschaltung durchzuführen.

Steuerbefehle

Befehls-Code	Funktion
1	Tabulation
2	Wagenrücklauf mit Zeilenschaltung
3	Partieller Wagenrücklauf mit Zeilenschaltung
4	Setzen der Tabulatorstops
5	Löschen der Tabulatorstops
6	Wagenrücklauf ohne Zeilenschaltung
7	Wie 6
8	Vorwahl ROT
9	Wie 8 mit Tabulation
10	Wie 8 mit Wagenrücklauf
11	Wie 8 mit partiellem Wagenrücklauf
12	Tastaturfreigabe
13	Wie 12
14	Wie 12
15	Wie 12

12.5.2.5 Schnelldrucker PR1220, PR1230, PR1240

Standardadresse : 9

Bei Installation können die Adressen geändert werden in :

Kanal 0 8 - 15

Kanal 1 24 - 31

Steuerung :

ISO-Zeichen	Funktion	Dezimal-Code
BEL	Drucker in Status lokal bringen	7
HT	Zeichen in doppelter Größe angeben	9
LF	Druckt Inhalt des Puffers, anschließend Zeilenschaltung	10
VT	Druckt Inhalt des Puffers, vertikale Tabulation	11
FF	Druckt Inhalt des Puffers, Formular in Grundstellung bringen	12
CR	Druckt Inhalt des Puffers, Schreibkopfrücklauf (keine Zeilenschaltung)	13
DC1	Druckt Inhalt des Puffers, auf dem zweiten Sprocket wird eine Zeilenschaltung durchgeführt	17
DC2	Druckt Inhalt des Puffers, vertikale Tabulation auf dem zweiten Sprocket	18
DC3	Druckt Inhalt des Puffers, Formular auf dem zweiten Sprocket wird in Grundstellung gebracht	19

Die Steuerzeichen für den Schnelldrucker werden durch Drücken der Taste CONTROL zusammen mit folgenden Tasten erzeugt :

Taste	Darstellung	Zeichen
G		BEL
I		HT
J		LF
K		VT
L		FF
M		CR
Q		DC1
R		DC2
S		DC3

Bemerkung : Das Papierende kann auf zwei Arten abgefragt werden :

- durch Verwendung des Steuerbefehls 10 (CMD per Einheit, 10 AND GO). Wird das Papierende erreicht, so ist IOC (4) = 1.
- ist der Programmstecker des Druckers entsprechend vorbereitet, so ist bei Papierende IOC (1) = 1.

In beiden Fällen geht der Drucker in den Zustand "lokal".

Beispiele :

```
FILE     PAPIER

0010 BUFFER #9,132
0020 FOR I=1 TO 200 STEP 1
0030 SEND #9,"E" AND GO
0040 WAIT #9
0050 IF IOC(1)=1 THEN 70
0060 GOTO 90
0070 DISP "PAPIERENDE ERREICHT";
0080 STOP
0090 NEXT I
0100 END

END OF LISTING


RUN
PAPIERENDE ERREICHT
```

```
FILE     PAPER

0010 BUFFER #9,132
0020 FOR I=1 TO 132 STEP 1
0030 CMD #9,10 AND GO
0040 WAIT #9
0050 IF IOC(4)=1 THEN 80
0060 SEND #9,"EE" AND GO
0070 GOTO 100
0080 DISP "PAPIERENDE ERREICHT"
0090 STOP
0100 NEXT I
0110 END

END OF LISTING
```

Die Steuerung der Printer erfolgt über ISO-Zeichen im auszugebenden String. Dafür gibt es 2 Möglichkeiten :

a) Das Steuerzeichen ist im String enthalten (SEND# 9, "P6060≡ ")

b) Das Steuerzeichen wird über CHR$( ) zum auszugebenden String addiert.
(SEND# 9, "P6060" + CHR$(10))

Beispiel : Im folgenden Beispiel soll der String "OLIVETTI P-6060" auf dem Schnelldrucker ausgegeben werden.

Durch "AND GO" in der Anweisung 50 und durch die Anweisung 60 kann nun die IOC-Funktion verwendet werden um den Zustand des Druckers per Programm abzufragen. IOC (6) ist 1, wenn der Drucker nicht eingeschaltet ist, in diesem Fall wird eine entsprechende Display-Anweisung ausgegeben.

```
LIST
FILE      PRINT

0010 REM  **** SCHNELLDRUCKER ****
0020 BUFFER #9,132
0030 DCL 132A$
0040 A$="OLIVETTI P-6060"
0050 SEND #9,A$+CHR$(10) AND GO
0060 WAIT #9
0070 IF IOC(6)=0 THEN 110
0080 DISP "DRUCKER EINSCHALTEN          ->";TAB(52);"CONTINUE DRUECKEN";
0090 STOP
0100 GOTO 50
0110 END

END OF LISTING
```

12.6 Der I/O-Kanal TASTATUR

Wie in den vorangehenden Kapiteln beschrieben ist, kann die Tastatur des P 6060 während eines Programmlaufs zur Dateneingabe verwendet werden (INPUT- oder RKB-Anweisung), aber auch zur Abfrage von Variablenwerten und zur Wertzuweisung an Variable (DEBUGGING - MODE).

Daneben kann die Tastatur von einem BASIC-Programm wie eine externe periphere Einheit durch die Statements BUFFER, SEND, RECEIVE etc. angesprochen werden. Dadurch wird zweierlei möglich:

- die Dateneingabe während eines Programmlaufs kann überlappt erfolgen: die Programmausführung wird nicht notwendiger Weise wie bei einer INPUT- oder RKB-Anweisung unterbrochen.
- Das BASIC-Programm kann Strings in das Display senden, diese Strings können mittels der Tastatur modifiziert und anschließend - wie eine normale Eingabe - an das Programm zurückgegeben werden.

12.6.1 BASIC-Anweisungen für den Kanal TASTATUR

Folgende BASIC-Anweisungen können für den Kanal TASTATUR verwendet werden :

BUFFER # per.-Einh., Puffergröße
CMD # per.-Einh., Steuer-Bef. [,Steuer-Befehl] [AND GO]
RECEIVE # per.-Einh., String-Var. [AND GO]
SEND # per.-Einh., String-Ausdr. [AND GO]
TEST # per.-Einh.
WAIT # per.-Einh.

Ferner kann die Funktion IOC (num.-Ausdr.) verwendet werden.

Bei der Ausführung einer RECEIVE-Anweisung gibt das System die Tastatur für die Dateneingabe frei : die eingetasteten Zeichen werden im Tastaturpuffer gespeichert und im Display angezeigt. Sobald die Taste END OF LINE gedrückt wird, wird der eingegebene String dem Tastaturpuffer entnommen, in den Puffer des Kanals TASTATUR übertragen und von dort aus der Stringvariablen zugewiesen, die im RECEIVE-Statement aufgeführt ist.

Bei der Ausführung einer SEND-Anweisung wird der in der Anweisung aufgeführte String in den durch die BUFFER-Anweisung definierten Puffer des Kanals TASTATUR geladen und von dort in den Tastaturpuffer übertragen. Der String kann (durch eine CMD-Anweisung) im Display sichtbar gemacht werden.

Mittels der Tastatur kann der String modifiziert und über eine INPUT, RKB oder RECEIVE-Anweisung an das Programm zurückgegeben werden.

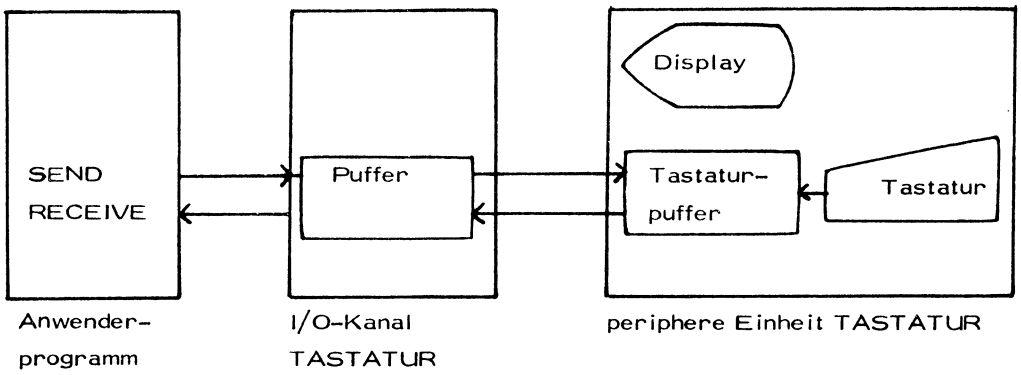


Bild 12.7 Die Tastatur des P 6060 als periphere Einheit

Der I/O-Kanal TASTATUR hat den logischen Namen 2, damit stehen folgende Werte für die Adresse "per.-Einh." zur Verfügung.

INPUT	OUTPUT
32-39	40-47

Von den angegebenen Adressen kann jeweils eine beliebige gewählt werden, aber diese sollte dann konsequent beibehalten werden : Wird z. B. in einer RECEIVE- AND GO-Anweisung die Adresse 32 verwendet, muß in einer nachfolgenden TEST- oder WAIT-Anweisung dieselbe Adresse verwendet werden, denn für jede der Adressen von 32-47 wird ein eigenes Zustandsregister eingerichtet. Statusmeldungen werden nur in dem Register abgelegt, das zu der in der Anweisung angegebenen Adresse gehört. Steht also in der TEST-Anweisung eine andere Adresse als 32, so wird ein anderes Zustandsregister getestet, die erhaltene Information ist irrelevant.

Für die einzelnen BASIC-Anweisungen gilt zunächst das in 12.4 gesagte. Daneben gibt es bei der Verwendung der Tastatur als periphere Einheit einige Besonderheiten, diese werden im folgenden zusammengestellt.

BUFFER : Als Adresse "per.-Einh." kann eine beliebige Zahl zwischen 32 und 47 gewählt werden. Die Puffergröße sollte 80 nicht überschreiten, da auch der Tastaturpuffer nur 80 Zeichen aufnehmen kann.

CMD : Folgende Steuerbefehle sind gültig :

Steuer-Befehl	Wirkung
ø	ohne Wirkung
1	Inhalt des Tastatur-Puffers wird im Display angezeigt.
16	Inaktiviert die Tastaur als perphere Einheit.
2-15 17-31	Diese Code werden nicht benützt.

D ie zugehörige Adresse "per. -Einh. " liegt zwischen 40 und 47.

RECEIVE : Fehlt "AND GO", wird die Programmausführung wie bei einer INPUT- oder RKB-Anweisung unterbrochen, das System wartet darauf, daß der Benutzer einen String eingibt. Die Konsollampe ON-LINE leuchtet auf. Sobald der Benutzer die Taste END OF LINE drückt, erlischt die Konsollampe ON-LINE, der im Tastaturpuffer enthaltene String wird in den Puffer des I/O-Kanals übertragen und von dort aus der in der Anweisung aufgeführten String-Variablen zugewiesen und der Puffer des I/O-Kanals wird geleert. Die Programmausführung wird fortgesetzt.

Ist "AND GO" angeführt, wird die Tastatur für die Eingabe aktiviert, die Konsollampe ON-LINE leuchtet auf um dem Benutzer anzuzeigen, daß er nun Zeichen eingeben kann. Die Programmausführung wird nicht unterbrochen: es wird sofort das nächste Statement ausgeführt. Wird die Taste END OF LINE gedrückt, erlischt die Konsollampe ON-LINE und der String wird in den Puffer des I/O-Kanals übertragen. Er wird nicht sofort der in der Anweisung stehenden Stringvariablen zugewiesen, sondern erst, wenn eine weitere I/O-Anweisung auszuführen ist, die sich auf den Kanal TASTATUR bezieht, z. B. wenn eine Anweisung CMD mit dem Steuerbefehl ø folgt.

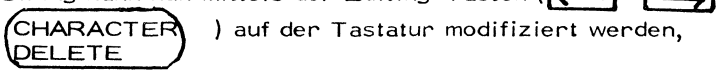
Für die Adresse "per.-Einh." sind Werte zwischen 32 und 39 zulässig.

Als String kann auch der Leerstring eingegeben werden, d. h. es kann die Taste END OF LINE gedrückt werden, ohne daß zwar ein Zeichen eingegeben wurde.

Werden mehr Zeichen über die Tastatur eingegeben als der Parameter "Puffer-Größe" in der Anweisung BUFFER zuläßt, wird der Leerstring in den Puffer des I/O-Kanals geschrieben, d. h. die eingegebenen Zeichen gehen verloren.

SEND :

Der Wert des String-Ausdrucks "String-Ausdr." (im folgenden kurz "String" genannt) wird in den durch die Anweisung BUFFER definierten Puffer geschrieben. Von dort wird er in den Tastaturpuffer übertragen. Ist der String vom Leerstring verschieden, werden dabei Zeichen, die sich im Tastaturpuffer befinden, überschrieben. Der String wird nicht im Display angezeigt. Um dies zu erreichen, muß auf die SEND-Anweisung eine CMD-Anweisung folgen, die den Steuerbefehl 1 gibt.

Dadurch wird der String im Display angezeigt, der Pointer befindet sich hinter dem letzten Zeichen des Strings. Der String kann nun mittels der Editing-Tasten (

) auf der Tastatur modifiziert werden, ferner können weitere Zeichen ein- oder angefügt werden bis z u einer Gesamtlänge von 80 Zeichen.

Um den so erhaltenen String wieder einer Stringvariablen zuzuweisen, muß eine RECEIVE, INPUT, MAT INPUT oder RKB-Anweisung folgen.

Hat der Wert des Stringausdrucks in der SEND-Anweisung eine Länge größer als 80 Zeichen, werden nur die ersten 80 Zeichen in den Tastaturpuffer geladen.

Für die Adresse "per.-Einh." sind Werte zwischen 40 und 47 zulässig.

Ist der Wert des Stringausdrucks "String-Ausdr." der Leerstring, bleibt der Inhalt des Tastaturpuffers erhalten.

TEST : Der Inhalt des Zustandsregisters wird in das Arbeitsregister übertragen, ohne das Ende einer laufenden I/O-Operation abzuwarten.

WAIT : Das Ende einer laufenden I/O-Operation wird abgewartet und der Inhalt des Zustandsregisters in das Arbeitsregister überführt.

Für die IOC-Funktion sind nur die Argumente 7 und 8 relevant; dabei gilt :

IOC (8) = 1
IOC (7) = 1

} Bei Ausführung einer RECEIVE AND GO-Anweisung ist die Eingabe nocht nicht durch END OF LINE abgeschlossen.

Sonst gilt IOC (8) = IOC (7) = ø.

12.6.2 Zusammentreffen von RECEIVE-Anweisungen mit Eingabeanforderungen oder Systemmeldungen.

Da bei der Ausführung einer RECEIVE-Anweisung ohne AND GO die Programmausführung unterbrochen wird, tritt keine Konfliktsituation im Zusammenhang mit Fehlermeldungen durch das System oder mit INPUT bzw. RKB-Anweisungen auf.

Da aber die Ausführung einer RECEIVE AND GO-Anweisung überlappt erfolgt, d.h. die Programmausführung fortgesetzt wird, kann der Fall eintreten, daß INPUT- bzw. RKB-Anweisungen auszuführen sind oder daß das System in den DEBUGGING-MODE geht, bevor die Eingabe durch END OF LINE abgeschlossen ist. Für diese Fälle gilt folgende Regelung :

- Übergang in den DEBUGGING MODE :

  Die Konsoltasten RUNNING und ON-LINE leuchten mit konstanter Helligkeit, anstelle der Taste CONTINUE leuchtet die Taste STEP. Der Inhalt des Tastaturpuffers bleibt unverändert. Sollen Variablenwerte abgefragt oder Wertanweisungen an Variable durchgeführt werden, muß der Tastaturpuffer durch Drücken der Tasten [SHIFT] und [CLEAR] geleert werden. Damit werden aber auch die bis dahin eingegebenen Zeichen gelöscht.

  Wird der DEBUGGING MODE verlassen und die Programmausführung fortgesetzt (durch Drücken der Konsoltaste CONTINUE), beginnt die Konsoltaste RUNNING wieder zu blinken und die Eingabe kann fortgesetzt werden. Wurde der Tastaturpuffer gelöscht, so muß der String vollständig neu eingegeben werden.

- INPUT- bzw. RKB-Anweisung

Die Konsollampen RUNNING und ON LINE leuchten mit konstanter Helligkeit, ferner leuchtet die Taste CONTINUE.

Die INPUT-, MAT INPUT- und RKB-Anweisung haben Vorrang gegenüber der RECEIVE-Anweisung, d. h. es muß gegebenenfalls der Tastaturpuffer gelöscht werden und es müssen die Werte für die Variablen in der INPUT-, MAT INPUT bzw. RKB-Anweisung eingegeben werden.

Wenn diese Eingaben durchgeführt sind, beginnt die RUNNING-Lampe wieder zu blinken und es sind die Zeichen für die in der RECEIVE-Anweisung aufgeführten Stringvariablen erneut einzugeben.

In der Tabelle sind die Statusanzeigen zusammenfassend dargestellt :

Konsollampe				Status
RUNNING	ON LINE	CONTINUE	STEP	
konstant	aus	an	aus	INPUT, MAT INPUT oder RKB ohne RECEIVE.
konstant	aus	aus	an	DEBUGGING MODE ohne RECEIVE.
blinkt	an	an	aus	RECEIVE mit AND GO
konstant	an	an	aus	RECEIVE ohne AND GO oder INPUT, MAT INPUT, RKB und RECEIVE mit AND GO.
konstant	an	aus	an	DEBUGGING MODE und RECEIVE mit AND GO

Tabelle 12.8 Statusanzeigen

12.6.3 Beispiele

Im ersten Programmbeispiel wird eine einfache Routine zur überlappten Eingabe von Datensätzen vorgestellt. Der Unterschied gegenüber einer Eingabe über eine INPUT- oder RKB-Anweisung besteht darin, daß das System kein "?" als Aufforderung zur Eingabe gibt und damit "kontinuierlich" erfolgen kann.

```
FILE      RECEIV

0010  **** BEISPIEL FUER DIE VERWENDUNG DER TASTATUR ALS PERIPHERER EINHEIT ****
0020 BUFFER #32,80
0030 DCL 80A$
0040 FILES FILE
0050 SCRATCH :1
0060 RECEIVE #32,A$ AND GO
0070 FOR I=1 TO 10 STEP 1
0080 RECEIVE #32,A$ AND GO
0090 WRITE  1,A$
0100 NEXT I
0110 END

END OF LISTING
```

Im zweiten Beispiel wird eine Routine zur Korrektur gespeicherter Datensätze beschrieben. Die zu ändernden Datensätze brauchen dazu nicht neu eingegeben werden, sondern Korrekturen können "mittels des Pointers" im Display durchgeführt werden.

```
LIST
FILE      SEND

0010  * BEISPIEL FUER KORREKTUR VON DATEN, DIE IN EINEM FILE GESPEICHERT SIND *
0020 FILES FILE
0030 BUFFER #32,80
0040 DCL 32(A$,B$)
0050 PRINT "ALTER NAME:","NEUER NAME:"
0060 PRINT
0070 FOR I=1 TO 10 STEP 1
0080 SETW :1 TO (I-1)*9+1
0090 READ :1,A$
0100 SEND #40,A$
0110 CMD #40,1
0120 PRINT A$;
0130 RECEIVE #32,B$
0140 SETW :1 TO (I-1)*9+1
0150 WRITE :1,B$
0160 PRINT TAB(32),B$
0170 NEXT I
0180 END

END OF LISTING
```

```
RUN SEND
ALTER NAME:                      NEUER NAME:

BILLY THE RHIPPER                BILLY THE KITT
JACK THE ROTHOUT                 JACK MIT ROTHOUT
AUGUST DER RUEPPEL               AUGUST DER KRUEPPEL
DON CAMILLO OLIVETTI             DON CAMILLO OLIVETTI
CARL ZEISS                       KARL ZEISS
PEPPONE SIEMENS                  PEPPONE SIEMENS
KARL XAVER VALENTIN              KARL  VALENTIN
BIEBER HAUS                      BIEBER HAUS
FISCHERS FRITZLEIN               FISCHERS FRITZLEIN
HOLY NIGHT                       HOLLY AT NIGHT
```

13. PLOTANWEISUNGEN

13. PLOTANWEISUNGEN

13.1 EINLEITUNG

Ein Plotter ist eine periphere Einheit zur Erstellung von Zeichnungen und Skizzen, allgemein eine Einheit, die automatisch numerische Daten in die Zeichenebene überträgt und dadurch ein Bild erzeugt. Als zu verarbeitende Daten werden vom Plotter rechtwinklige Koordinaten gefordert. Ein Bild wird immer aus einer großen Anzahl von Punkten, numerisch repräsentiert durch Koordinaten, bestimmt.

Da numerische Informationen allein nicht übersichtlich und oft unvollständig sind, fordert der Anwender den Anschluß eines Plotters. Besondere Bedeutung wird der graphischen Information dort bemessen, wo Strukturen und Abhängigkeiten untersucht und erklärt werden müssen. In der Mathematik und Statistik, den reinen numerischen Wissenschaften, erlaubt erst die graphische Aussage vielen Anwendern, die Zahlen zu interpretieren und zu verstehen. In der numerischen Steuerung für Werkzeugmaschinen wird vor der eigentlichen Fertigung des Werkstückes der Arbeitsprozeß mit Hilfe des Plotters simuliert, um Fehlproduktionen und Werkzeugschäden zu vermeiden. Sicher lassen sich noch viele Anwendungen aufführen. Für alle diese Lösungen gilt, daß sehr viele Zahlen automatisch in eine Zeichnung, die für das menschliche Auge angenehmste Form, umgesetzt werden.

Der integrierte Printer des P6060 kann als Plotter verwendet werden.
An den P6060 können zudem verschiedene periphere Plotter angeschlossen werden. Im Rahmen der System-Option PLOT werden integrierter Printer und peripherer Plotter durch spezielle BASIC-Anweisungen unterstützt.

13.2 DEFINITION DES BILDES

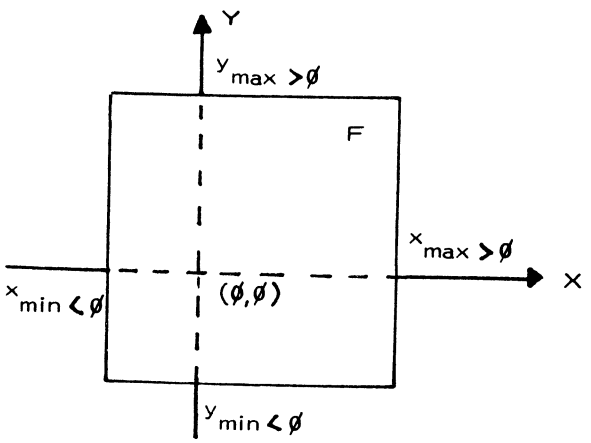
Ein Bild wird durch ein BASIC-Programm mit speziellen Anweisungen der Option PLOT konstruiert. Diese Anweisungen zeichnen Punkte und Linien und schreiben Buchstaben und Sonderzeichen. Andere Anweisungen legen das Koordinatensystem fest oder bestimmen, ob über einen externen Plotter oder den integrierten Printer ausgegeben werden soll. Das Bild schließlich ist die Gesamtheit aller Punkte, Linien und Buchstaben. Jeder Punkt für sich wird digital, durch numerische Informationen in Form von "X"- und "Y"-Koordinaten, entwickelt und auf der Zeichenfläche markiert.

Jeder Punkt dieser Zeichenfläche wird über seine Koordinaten angesprochen. Die Beschränkung liegt hier in der Größe der Ausgabeeinheit. Sicher ist immer nur ein Teil des allgemeinen Koordinatensystems abzubilden; es kommt also darauf an, den Teil herauszunehmen, der die wesentlichen Informationen des Bildes enthält. In vielen Fällen muß dieser Teil maßstäblich veränderlich sein. Diese Grundfunktionen sollen nicht Punkt für Punkt dem Anwender überlassen sein, sondern in der unterstützenden Systemsoftware enthalten sein.

Es kann sowohl die Größe der Zeichenfläche als auch die Lage des Koordinatensystems frei gewählt werden. Der Ursprung kann innerhalb oder außerhalb der Zeichenfläche liegen.

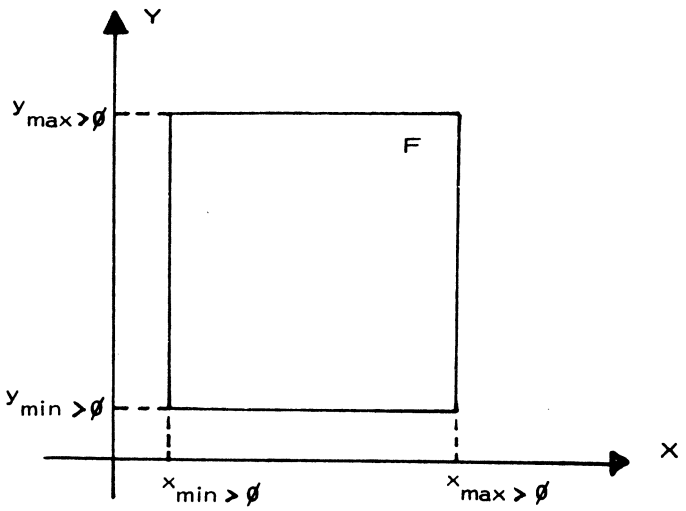
a) Ursprung innerhalb der Zeichenfläche F :

x_{min} und $y_{min} < \emptyset$, x_{max} und $y_{max} > \emptyset$:



b) Ursprung außerhalb der Zeichenfläche F :

z. B. x_{min}, x_{max}, y_{min} und $y_{max} > \emptyset$:



Die Festlegung des Koordinatensystems der Zeichenfläche erfolgt über die Anweisung SCALE, in der das Maximum und Minimum der Abszisse und Ordinate angegeben wird. Dies wird am Anfang definiert und bestimmt damit die Maßeinheit und die Lage des Nullpunktes des Koordinatensystems.

Da die Extremwerte für Abszisse (X-Richtung) und Ordinate (Y-Richtung) angegeben werden, können sich unterschiedliche Maßstäbe für die X-Achse und Y-Achse bei gleichbleibender Zeichenfläche ergeben.

13.3 ZEICHNEN MIT DEM INTEGRIERTEN PRINTER

13.3.1 Der integrierte Printer als Digitalplotter

Der Minicomputer P6060 kann den integrierten Printer wie einen Digitalplotter benutzen. Das Bild, das über den integrierten Printer ausgegeben wird, setzt sich aus einzelnen Punkten zusammen. Die Zeichenfläche selbst kann als eine große rechtwinklige Punktmatrix mit konstantem Punktabstand in Zeile und Spalte angesehen werden. Die Grundeinheit ist also ein Punkt.

Die Dimension der Zeichenfläche (Breite und Höhe des Feldes) ist definierbar, die Maße werden in Zoll angegeben. Die Relation zwischen Zollmaß und Punkt-Einheit ist :

1 Zoll $\hat{=}$ 70 Punkte

Über den integrierten Printer werden immer 7 zusammengehörende Zeilen der Matrix gleichzeitig ausgegeben.

13.3.2 Darstellung von Punkten, Geraden, Buchstaben und Sonderzeichen

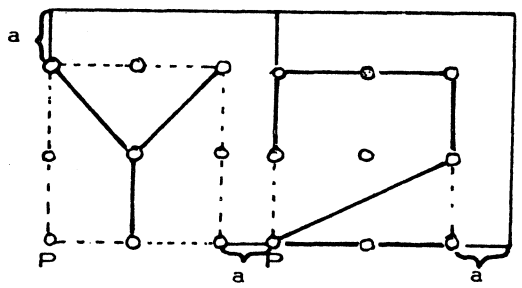
Ein BASIC-Programm konstruiert aus Daten die Punkte, Geraden und Strings des Bildes. Ob Geraden oder alphanum.-Zeichen, alle Elemente der Zeichnung werden auf die Grundeinheit durch ihre Koordinaten, Abszisse und Ordinate, eindeutig fixiert. Die Koordinaten selbst sind einfach genaue Werte.

Im allgemeinen werden die Koordinaten eines Punktes berechnet und als Punkt in die Zeichenfläche gesetzt. Dabei ergeben sich Abweichungen zwischen den echten Koordinaten und den dargestellten Punkten aufgrund des Auflösungsvermögens (1/70 Zoll als kleinste darstellbare Einheit).

Geraden sind nichts anderes als eine Folge dicht beeinander liegender Punkte. Der Weg von einem Ausgangspunkt zum Endpunkt wird analytisch durch die Lösung von Gleichungen bestimmt. Die Folge der zwischen Ausgangspunkt und Endpunkt liegenden Punkte wird berechnet und Punkt für Punkt in die Punktmatrix übertragen.

Das System P6060 übernimmt diese Aufgabe, ohne daß der Programmersteller Punkt für Punkt konstruieren muß. Auch Buchstaben, numerische Zeichen und Sonderzeichen werden vom P6060 automatisch generiert.

Der Zeichengenerator erzeugt mittels eines 9-Punkt-Rasters und den Verbindungslinien das gewünschte Symbol. Das Baugesetz eines Zeichens benötigt mindestens 2 Punkte. Der Programmierer definiert nur die Größe der Symbole (Anweisung CSIZE). Unabhängig von der Dimension wird der Abstand zu anderen Zeichen aufgebaut. Zum Beispiel wird der Buchstabe Y und die Zahl 2 folgendermaßen konstruiert.



a = Abstand zum nächsten Zeichen

Der Punkt P zeigt die Position an, die als Ausgangspunkt für ein Symbol oder einen ganzen String gilt.

Auch die Schreibrichtung des Strings ist frei wählbar. Diese Drehung wird durch einen Winkel zur Abszisse angegeben.

Da Leerräume in Breite und Höhe als Bestandteil eines Zeichens gelten, wird auf natürliche Weise das Blatt in Linien aufgeteilt. Zu beachten ist, daß die Größe der Zeichen im Zeichengenerator in Zoll angegeben wird und nicht mit der Maßeinheit für das Koordinatensystem gekoppelt ist.

13.3.3 Konstruktion und Ausgabe des Bildes

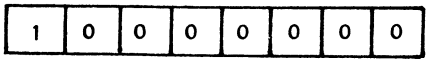
Das Plotten über den integrierten Printer kann nicht allein als natürliche Folge von Plotanweisungen gesehen werden. Der Drucker ist keine Ausgabeeinheit mit einem Schreibstift, der beliebig über das Papier geführt werden kann.

Benutzt man den integrierten Printer als Plotter, muß das Bild zunächst digital entwickelt und gespeichert werden. Erst wenn dieser Vorgang beendet ist, kann die Ausgabe zeilenweise erfolgen.

Für die Aufzeichnung des Bildes ist es notwendig, einen Puffer im Arbeitsspeicher aufzubauen und ein Datenfile auf einer Diskette zu eröffnen (CREATE). Die Größe des Puffers und der Name des Files werden in der ersten Plotanweisung definiert. Diese Anweisung heißt INIMAGE.

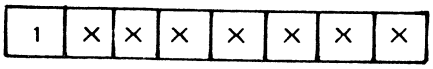
Im Puffer wird byteweise gespeichert. Die vier folgenden Formate sind möglich :

1)



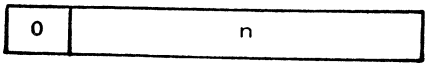
Definiert erstes und letztes Byte des Puffers.

2)



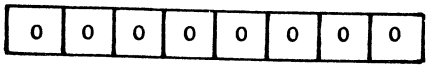
Definiert eine Spalte (Kolonne) mit 7 Punkten, eine Markierung bedeutet den Wert 1 für das Bit.

3)



Heißt, n-Spalten mit 7 Punkten sind nicht markiert.

4)



Dieses Byte wird nicht verwendet.

Im Puffer wird noch ein Arbeitsspeicher von 256 Byte eingerichtet.

Nacheinander wird der Puffer mit allen Informationen gefüllt, die über Anweisungen generiert werden. Die Zeichenfläche ist aufgeteilt in Zeilen mit je 7 Punkten, von denen jeder für sich alleine markiert sein kann. Diese Zeilen sind sequentiell zusammengestellt und ergeben somit die Gesamtheit aller Punkte, das Bild.

Angenommen, ein Bild der Größe 2 x 8 Zoll (1 Zoll $\hat{=}$ 2.5399 cm) wird aufgebaut und ein Puffer von 3 K-Byte im Arbeitsspeicher eingerichtet. Nach den Anweisungen INIMAGE und FRAME wird mit 89 Byte des Formats 3 die gesamte Fläche beschrieben. Berücksichtigt man ferner, daß für Systemzwecke 258 Bytes benötigt werden, bleiben 2725 Bytes frei (Format 4).

Die folgende Tabelle soll zeigen, wie in Abhängigkeit der Zeichenfläche der Puffer von 3 K-Byte aufgeteilt wird.

Zeichenfläche (Zoll)	max. Platzbedarf *	Puffer (3K) 3072 Byte				
		Arbeitsspeicher	Anfangsbyte	Byte Format 3	Byte frei	Endbyte
8 x 2	11.200	256	1	89	2725	1
8 x 8	44.800	256	1	353	2461	1
8 x 20	112.000	256	1	882	1932	1
8 x 40	224.000	256	1	1764	1050	1

* wenn jeder Punkt der Zeichenfläche markiert wird.

Mit jeder Markierung vermindert sich der noch freie Platz im Puffer. Ist der Puffer vollständig gefüllt, dann wird dieses erste Teilbild in das Datenfile auf der Diskette geschrieben und der Puffer im Arbeitsspeicher neu aufbereitet, um neue Punkte zu speichern. Auf diese Weise wird das Bild aus Teilbildern zusammengesetzt, die im Puffer in Byteform entwickelt werden.

Die Anweisung DRAW schreibt ein letztesmal den Inhalt des Puffers ins Datenfile. Dann wird dieser Platz benötigt um das Bild aus den Teilbildern zusammenzusetzen. Als Ergebnis sieht man das Bild zeilenweise entstehen. Das Bild ist gespeichert und kann als Ausgangssituation für weitere Verarbeitungen benutzt werden.

In manchen Programmen kann es genügen, allein im Puffer des Arbeitsspeichers zu arbeiten. In diesem Fall kann das Bild nicht in Teilbildern entwickelt werden und der Puffer muß groß genug bemessen sein. Nach der Anweisung DRAW ist aber dann das Bild nicht mehr verfügbar.

Beim Plotten über den integrierten Printer werden die folgenden Arbeitsschritte ausgeführt :

- Definition des Datenfiles und des Puffers im Arbeitsspeicher durch die Anweisungen INIMAGE oder LDIMAGE
- Definition der Zeichenfläche, der Bildgröße, mit der Anweisung FRAME, die nur in Verbindung mit INIMAGE gilt
- Folge von Plotanweisungen, mit der Aufgabe, das gewünschte Bild zu konstruieren
- Ausgabe der Zeichnung mit der Anweisung DRAW

Sekundär sind sicher die folgenden Probleme :

Optimale Dimensionierung des Puffers und des Files, minimale Speicherkapazität auf der Diskette und maximale Geschwindigkeit für das Programm.

Berücksichtigt man, was über den Puffer und die Speicherform gesagt wurde, dann ist vorteilhaft, das Bild möglichst horizontal zu entwickeln. Ist der Puffer sehr groß, dann sind wenige Speichervorgänge für das Bild notwendig. Ein zu kleiner Puffer bedeutet viele Speicherungen und mehr Zeitaufwand. Eine allgemeine Regel zu definieren, ist nicht möglich. Sicher ist, daß Größe des Puffers und des Files von der Größe des Bildes und der Anzahl der markierten Punkte abhängt.

Richtwerte lassen sich folgendermaßen finden :

- 1 Zoll^2 benötigt 700 Bytes, wenn alle Punkte markiert sind.
- Der maximale Speicherbedarf für ein Bild, in dem alle Punkte markiert sind, wird bestimmt nach der Formel :

  Fläche x 700 Bytes + 384 Bytes für Parameter

  Breite und Höhe des Bildes in Zoll gemessen. Ein Puffer, der nach dieser Regel dimensioniert ist, kann das gesamte Bild aufnehmen.
- Bildet man das Verhältnis "Anzahl der markierten Punkte" zu "Anzahl der nicht markierten Punkte", kann als Größe des Files die Verhältniszahl, multipliziert mit maximalem Speicherbedarf überschlagsmäßig angesetzt werden.

13.4 ARBEITEN MIT PERIPHEREM PLOTTER

Die Anweisungen in der Option PLOT sollen nicht nur das Plotten über den integrierten Printer ermöglichen, sondern auch das Arbeiten mit peripheren Plottern unterstützen. In diesem Falle werden die einzelnen Plot-Anweisungen als Aufruf einer BASIC-Funktion, die Bestandteil des Programmes sein muß und auf die Art und die Funktion des Plotters abgestimmt ist, angesehen.

Das BASIC-Programm muß dann zusätzlich zu den BASIC-Anweisungen für den integrierten Printer die mehrzeilige Funktion FNP enthalten.

Diese Funktion ist als Textfile auf einer Diskette gespeichert und kann mit dem Befehl UNK an das bestehende Programm angehängt werden.

Die Funktion FNP enthält :

a) 6 Argumente mit einer Stringvariablen an der 5. Stelle

b) die Anweisung FN* = 0

c) die Anweisung EXTERNAL PLOTTER

d) die einzelnen PLOT-Anweisungen, die direkt in Kommandos für den entsprechenden externen Plotter übersetzt werden.

Form der Funktion FNP :

```
DEF FNP (A, B, C, D, E$, F)  lokale Parameter
  •
  •
EXTERNAL PLOTTER
  •
  •
  •
FN   = Ø
  •
  •
  •
  •
FNEND
```

Die Funktion FNP darf nicht aufgerufen werden, sondern die Plotanweisungen werden direkt in Funktionsaufrufe übersetzt und die Parameter in die Argumente übernommen. Die ersten 4 Argumente (A, B, C, D) der Funktion nehmen die 4 möglichen numerischen Parameter der Anweisungen auf. Die Stringvariable in der Anweisung CPLOT wird in die Stringvariable E$ geschrieben. Der 6. Operand (F) enthält als numerische Information den Typ der Plotanweisung, die im Programm aufgerufen wird. Auf der Basis dieser Werte wird die Funktion FNP entwickelt.

An das System P6060 können verschiedene Plotter angeschlossen werden. Notwendig ist für jeden einzelnen Plotter, daß einmal die Funktion FNP die Plotanweisungen in Steuerbefehle des Plotters umsetzt. Die Arbeitsweise mit einem peripheren Plotter unterscheidet sich wesentlich von der mit dem integrierten Printer. Es entfällt die Entwicklung des Bildes im Puffer und das Speichern in ein Datenfile. Deshalb werden die Anweisungen INIMAGE, LDIMAGE, FRAME und DRAW hinfällig und falls vorhanden überlesen.

Die folgende Tabelle zeigt die Zuordnung zwischen Plotanweisung und dem numerischen Wert des 6. Arguments der Funktion FNP :

BASIC-Anweisung	Typ der Anweisung (Argument F)
SCALE	10
CSIZE	11
OFFSET	12
CTAB	13
XAXIS	14
YAXIS	15
PLOT	16
IPLOT	17
CPLOT	18
IDOT	19
DOT	20
MOVE	21

13.5 BASIC - PLOTANWEISUNGEN

Kurzbeschreibung

Mit den BASIC-Anweisungen der Option PLOT werden Bilder generiert. Sie gelten für den integrierten Printer und alle angeschlossenen externen Plotter.

CPLOT	Schreiben eines Strings
CSIZE	Definition der Größe und Schreibrichtung für einen String
CTAB	Tabulation für die Ausgabe von Strings (in Schriftrichtung senkrecht dazu)
DOT	Zeichnen eines Punktes
DRAW	Ausgabe des Bildes über den integrierten Printer
EXTERNAL PLOTTER	Definition zur Ausgabe auf externen Plotter
FRAME	Festlegung der Größe der Zeichenfläche (nur bei integr. Printer)
IDOT	Zeichnen eines Punktes mit Inkrementen dx und dy
INIMAGE	Definiert Puffer und Datenfile für das Bild (nur integr. Printer)
IPLOT	Zeichnen einer Geraden zu dem Punkt, der durch Inkremente dx und dy erreicht wird
LDIMAGE	Laden eines gespeicherten Bildes (nur bei integr. Printer)
MOVE	Positionieren auf einen Punkt
OFFSET	Definiert einen neuen Ursprung für das Kartesische Koordinatensystem
PLOT	Zeichnen einer Geraden zu dem durch Koordinaten definierten Punkt
SCALE	Festlegen von Lage und Maßstab des Koordinatensystem
XAXIS	Zeichnen einer Parallele zur X-Achse
YAXIS	Zeichnen einer Parallele zur Y-Achse

CPLOT

Anweisung CPLOT (Characterplot)

Funktion : Ausgabe einer Liste von Strings mittels Plot-Zeichengenerator

Format : CPLOT [String-Ausdr.] [; [String-Ausdr.]] ... [;]

mit : "String-Ausdr." für alphanumerische Ausdrücke, Beschränkung auf maximal 8 Stringausdrücke.

Wirkung : Stringausdrücke werden an die Stelle geschrieben, die vorher positioniert wurde. Die Größe der Zeichen und die Schreibrichtung richtet sich nach der Anweisung CSIZE.

Wird als letztes Zeichen in der Anweisung das Trennzeichen ";" geschrieben, wird auf den Ausgangspunkt für das direkt nachfolgende Zeichen positioniert. Wird ";" nicht als letztes Zeichen angegeben, dann wird als neuer Ausgangspunkt der Stringanfang eine Zeile tiefer gesetzt. Dies ermöglicht eine einfache spaltenmäßige Stringausgabe.

Bemerkung : Der Zeichengenerator erzeugt folgende Zeichen :

- die Grossbuchstaben:

  A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z

- die Ziffern:

  0,1,2,3,4,5,6,7,8,9

- die Sonderzeichen:

  ! # $ % & ' / ( ) * + , - : ; . < = > ? @ [ ] ↑ _ \

Enthält ein String andere Zeichen, so werden diese durch das Symbol "!!!" dargestellt.

Es wird nur der Teil des Strings geschrieben, der in die definierte Zeichenebene hineinpaßt. Wird über den Rand hinausgeschrieben, wird als Hinweis nach der Ausgabe des Bildes das Maximum oder Minimum in X und (oder) Y ausgegeben.

Fehlermeldungen: Die Anweisung INIMAGE oder LDIMAGE wurde nicht gegeben. (ERROR 241)

Die Operation konnte nicht ausgeführt werden :

der Inhalt des Puffers kann nicht auf die Diskette geschrieben werden oder der Puffer ist für das Bild nicht groß genug. Der Fehler ist behebbar, die Punkte, die die Meldung hervorrufen, werden nicht gezeichnet. (ERROR 250)

CSIZE

Anweisung CSIZE (Charactersize)

Funktion : Definition der Zeichengröße von Strings und der Schreibrichtung, bezüglich der X-Achse.

Format : CSIZE b, h, w

mit : numerischen Werten für die Parameter b (Breite), h (Höhe) und dem Winkel w im Bogenmaß.
b, h ≥ 0,03

Wirkung : Die numerischen Werte für die Breite und Höhe stehen für das Punktraster in dem die Zeichen generiert werden. Die Dimension für Breite und Höhe ist das Maß Zoll. Der Winkel, der im Bogenmaß angegeben wird, hat als Drehrichtung die mathematisch positive Definition (entgegen dem Uhrzeigersinn).

Bemerkung : Die Anweisung CSIZE gilt so lange für alle nachfolgenden CPLOT-Anweisungen, bis eine erneute CSIZE-Anweisung erfolgt.

Wenn die Anweisung CSIZE fehlt, wird für Breite und Höhe 1/7 Zoll genommen und der Winkel $\emptyset$ gesetzt.

Von den festgelegten Werten für Breite und Höhe werden 6/10 für das Zeichen selbst und 4/10 für den Zwischenraum zum nächsten Zeichen nach rechts bzw. oben genommen.

Fehlermeldungen: Der Wert für Breite oder Höhe ist kleiner 0,03 (ERROR 251).

Die Anweisungen INIMAGE oder LDIMAGE wurden nicht gegeben (ERROR 241).

CTAB

Anweisung CTAB (Charactertab)

Funktion : Tabulation auf einen Punkt, dessen Koordinaten durch eine Verschiebung um n Zeichen und m Zeilen gegeben sind (bezüglich der letzten ausgeführten CPLOT-Anweisung)

Format : CTAB n, m

mit : den Parametern n und m als numerische "Ausdrücke"

Wirkung : Die nächste String-Ausgabe mittels CPLOT erfolgt um n Zeichenbreiten und m Zeilen (gemäß CSIZE) versetzt. Die Anweisung CTAB bezieht sich immer auf die letzte CPLOT-Anweisung.

Bemerkungen : Die Anweisung CTAB ist vergleichbar mit der Anweisung MOVE, die eine Koordinatenbewegung zu einem durch Koordinaten definierten Punkt ausführt und als Vorbereitung für PLOT, DOT, ... verwendet wird. Der Vorteil der Anweisung CTAB besteht darin, daß man nicht auf Koordinaten umrechnen und auch die Schreibrichtung für den String nicht berücksichtigen muß.

Das Positionieren sollte so ausgeführt werden, daß innerhalb der Zeichenfläche geblieben wird. Reicht der Rahmen für das Gesamtbild nicht aus, so wird am Ende das Maximum oder Minimum der Koordinatenwerte als Hinweis ausgegeben.

Fehlermeldungen: Die Anweisungen INIMAGE oder LDIMAGE wurden nicht gegeben (ERROR 241).

DOT

Anweisung **DOT**

Funktion : Markieren eines Punktes, der durch Koordinaten definiert ist

Format : DOT X, Y

mit : X und Y sind numerische Ausdrücke, die die Abszisse und die Ordinate des Punktes angeben.

Wirkung : Die numerischen Werte X und Y bestimmen Abszisse und Ordinate des Punktes, der markiert werden soll. Es erfolgt die Koordinatenbewegung zu dem angegebenen Punkt, ohne daß eine Linie gezogen wird.

Bemerkungen : Liegt der Punkt außerhalb der Bildfläche, wird er nicht markiert. In diesem Falle wird der Wert der Abszisse und (oder) Ordinate als Außenpunkt registriert. Als Hinweis wird nach der Ausgabe des Bildes, das Maximum und (oder) Minimum der Außenpunkte ausgegeben.

Fehlermeldungen: Die Anweisung INIMAGE oder LDIMAGE wurde nicht gegeben. (ERROR 241)

Die Operation konnte nicht ausgeführt werden :

der Inhalt des Puffers kann nicht auf die Diskette geschrieben werden oder der Puffer ist für das Bild nicht groß genug. Der Fehler ist behebbar, die Punkte, die die Meldung hervorrufen, werden nicht gezeichnet. (ERROR 250)

DRAW

Anweisung DRAW

Funktion : Ausgabeanweisung für das Bild über den integrierten Printer

Format : DRAW [d]

mit : d als numerischen Ausdruck ($0.03 \leq d < 8$) für die Verschiebung des Koordinatenursprungs in X-Richtung

Wirkung : Mit dieser Anweisung wird das erzeugte Bild ausgegeben. Es wird aus den Teilbildern des Datenfiles, festgelegt durch die Anweisung INIMAGE oder LDIMAGE, zusammengesetzt.

Wird der Parameter d (in Zoll) angegeben, so wird das im Datenfile gespeicherte Bild bei der Ausgabe entsprechend nach rechts verschoben.

Bemerkungen : Die Anweisung DRAW veranlaßt zunächst die Aufzeichnung des Puffers im Arbeitsspeicher auf das genannte Datenfile auf der Diskette. Der Inhalt des Datenfiles bleibt so lange erhalten, bis mit den Anweisungen INIMAGE oder LDIMAGE neue Plotoperationen mit dem gleichen File ausgeführt werden.

Nach dem Schreiben aus dem Puffer steht im Zentralspeicher, wie sich das Bild aus gespeicherten Teilbildern in dem File zusammensetzt. Das Bild kann nicht in beliebig viele Teilbilder gesplittert werden. Es sind maximal n = (Puffergröße - 256) /128 (alle Angaben in Byte) Teilbilder möglich, wobei jedes Teilbild durch eine Abspeicherung des Pufferinhalts auf die Diskette erzeugt wird.

Ist in den Anweisungen INIMAGE oder LDIMAGE als Parameter OFF aufgeführt, dann werden mit DRAW nur die Extremwerte, Maximum und Minimum der Abszissen und Ordinaten des Bildes, ausgegeben. Diese Werte, gemessen in Zoll, zeigen an, welcher Bereich für das Bild vorzusehen ist, und können dann in der Anweisung SCALE für die Festlegung der Zeichenfläche angegeben werden.

Die Ausführung der Anweisung DRAW kann durch Drücken der Tasten STEP oder BREAK unterbrochen werden. STEP führt in den Debugging-Mode, die Ausführung kann wiederholt werden. BREAK bedeutet vorzeitiges Programmende. In jedem Fall werden jedoch die Extremwerte ausgedruckt. Steht im Programm die Anweisung EXTERNAL PLOTTER, wird die Anweisung DRAW überlesen.

Fehlermeldungen: Die Anweisung INIMAGE oder LDIMAGE wurde nicht gegeben. (ERROR 241)

Der Wert für die Verschiebung bei der Ausführung der DRAW-Anweisung überschreitet zulässige Grenzen (ERROR 239).

Der integrierte Printer fehlt (ERROR 240).

Anweisung EXTERNAL PLOTTER

Funktion : Vorwahl für einen externen Plotter

Format : EXTERNAL PLOTTER

Wirkung : Mit dieser Anweisung wird angegeben, daß das Bild nicht in einem Datenfile erzeugt wird, sondern direkt über den angeschlossenen externen Plotter gezeichnet wird. Die eigentliche Plottergrundsoftware, die bei verschiedenen Plottern unterschiedlich ist, muß in der Funktion FNP zur Verfügung stehen. Diese Funktion enthält auf BASIC-Ebene alle spezifischen Anweisungen für den externen Plotter.

Bemerkungen : Die Anweisung EXTERNAL PLOTTER ist eine nicht ausführbare Anweisung, kann also an jeder Stelle des Programmes stehen (z. B. in FNP).

Die Anweisungen INIMAGE, LDIMAGE, FRAME und DRAW gelten nur für den integrierten Printer. Steht im gleichen Programm die Anweisung EXTERNAL PLOTTER, dann werden diese Anweisungen überlesen.

Fehlermeldungen: Im Programm fehlt die Funktion FNP (ERROR 238).

FRAME

Anweisung FRAME

Funktion : Festlegung der Größe der Zeichenfläche

Format : FRAME b, h

mit : b (Breite) und h (Höhe) als numerische Ausdrücke

0.03 <= b <= 8 (ca. 20 cm)

0.03 <= h <= 936 (ca. 23.4 m)

Wirkung : Die numerischen Werte für "Breite" und "Höhe" definieren die Größe der Zeichenfläche, in der das Bild generiert wird. Die Dimension der Parameter ist Zoll.

Bemerkungen : Die obere Grenze der "Breite" wird durch die Breite des Thermopapiers bestimmt und entspricht ca. DIN A4 mit Hochformat. Die Höhe ist nur theoretisch limitiert (23 Meter!) und hängt, genau wie die Gesamtfläche des Bildes, von der Größe des Puffers im Arbeitsspeicher und derjenigen des externen Datenfiles ab.

Im Programm muß die Anweisung FRAME nach INIMAGE, aber vor allen anderen Plotanweisungen stehen. Wird die Anweisung FRAME nicht aufgeführt, dann wird für das Bild eine Größe von 8 x 8 Zoll angenommen.

Steht im Programm die Anweisung EXTERNAL PLOTTER, so wird die Anweisung FRAME überlesen.

Fehlermeldungen: Der Wert für die Breite liegt nicht innerhalb der vorgegebenen Grenzen (ERROR 245).

Der Wert für die Höhe liegt nicht innerhalb der vorgegebenen Grenzen (ERROR 246).

Die Anweisung FRAME steht vor INIMAGE (ERROR 242).

Der Puffer im Arbeitsspeicher ist zu klein, um das Bild aufzunehmen (ERROR 248).

Die Anweisung INIMAGE wurde nicht gegeben (ERROR 241).

IDOT

Anweisung IDOT (Incrementdot)

Funktion : Zeichnen eines Punktes durch Angabe von Inkrementen dx und dy

Format : IDOT dx, dy

mit : dx und dy als numerische Ausdrücke

Wirkung : Die numerischen Werte für dx und dy heißen Inkremente und definieren einen Punkt, der bezüglich des zuletzt angesprochenen Punktes X/Y die Koordinaten dx/dy hat (Koordinaten bezüglich des Ursprungs : X + dx/ Y + dy).

Bemerkungen : Wenn IDOT auf keinen Bezugspunkt zurückgreifen kann, wird der Ursprung des Koordinatensystems als Ausgangspunkt genommen.

Wenn der zu zeichnende Punkt außerhalb des Rahmens fällt, werden das Maximum und (oder) Minimum von Abszisse und Ordinate gespeichert, damit ein Hinweis auf zu kleine Zeichenfläche gegeben werden kann.

Fehlermeldungen: Die Anweisung INIMAGE oder LDIMAGE wurde nicht gegeben (ERROR 241).

Die Operation konnte nicht ausgeführt werden :

Der Inhalt des Puffers kann nicht auf die Diskette geschrieben werden oder der Puffer ist für das Bild zu klein. Der Fehler ist behebbar, die Punkte, die die Meldung hervorrufen, werden nicht gezeichnet (ERROR 250).

INIMAGE

Anweisung INIMAGE

Funktion : Definition des Datenfiles, das das Bild speichern soll, Aufbauen des Puffers im Arbeitsspeicher und Laden der Parameter, die für die Plotoperationen notwendig sind

Format : INIMAGE [OFF / Filename] [, n]

mit : "Filename" = Name des sequentiellen Datenfiles. "n" = Größe des Puffers (2<= n <= 48 Kbyte).

Bemerkungen : Erster Parameter :

- OFF : Das gesamte Plot-Programm wird ausgeführt. Anstelle der Markierung von Punkten und der Speicherung des Bildes wird über das gesamte Bild das Maximum und Minimum der Abszissen und Ordinaten bestimmt.

- Filename : Das File mit dem Namen "Filename" ist für die Speicherung des Bildes vorgesehen. Das sequentielle File darf nicht gesichert sein und kann der Package-, Common- oder User-Bibliothek angehören. Mit der Anweisung INIMAGE wird das File zur Speicherung des Bildes vorbereitet und ein eventuell bereits vorhandenes Bild gelöscht. Damit ist das File kein gewöhnliches Datenfile.

- Wenn weder OFF noch ein Filename als Parameter eingesetzt ist, wird als Filename SYSPLO gesetzt. Existiert das genannte File SYSPLO nicht, dann wird das Bild im Puffer des Arbeitsspeichers erzeugt. Jede neue INIMAGE oder LDIMAGE-Anweisung löscht dann das gespeicherte Bild im Arbeitsspeicher.

Zweiter Parameter :

- Im Arbeitsspeicher wird ein Puffer von n-Kbyte eingerichtet, der alle Informationen über die gezeichneten Punkte aufnehmen soll. Nach dem Aufbau des Puffers sind keine Punkte darin gespeichert. Ist der Puffer durch fortlaufende Plotanweisungen gefüllt, wird der Pufferinhalt auf das genannte File als Teilbild gespeichert und danach neu aufgebaut. Auf diese Weise wird das Bild aus Teilbildern zusammengesetzt.

- Ist "Filename" aufgeführt, aber fehlt "n", dann wird n = 3 gesetzt.

- Ist "OFF" aufgeführt, ist "n" nicht bedeutend und wird überlesen, wenn es aufgeführt ist.

- Die Anweisung INIMAGE initialisiert die Standard-Parameter für die im Programmfolgenden Plot-Anweisungen in den Arbeitsspeicher.

  Diese sind :

  - Die Dimension des Bildes : 8 x 8 Zoll

  - Die Maßeinheiten für die Achsen, Minimum und Maximum von Abszissen und Ordinaten für die Zeichenfläche.
    1 Maßeinheit = 1 Punkt = 1/70 Zoll
    Abszisse : Minimum = - 279.5; Maximum = 279.5
    Ordinate : Minimum = - 279.5; Maximum = 279.5

    Dies bedeutet, daß der Ursprung des Koordinatensystems in der Mitte der Zeichenfläche liegt.

  - Die Dimension der Buchstaben im Zeichengenerator.
    Breite des Punkterasters für 1 Zeichen = 1/7 Zoll (10 Punkte)
    Höhe des Punkterasters für 1 Zeichen = 1/7 Zoll (10 Punkte)
    Winkel = Ø (Bogenmaß).

  Die Werte für die Plot-Parameter können durch die Anweisungen FRAME, SCALE und CSIZE modifiziert und damit dem Problem angepaßt werden.

Bemerkungen : Die Anweisung INIMAGE (oder alternativ LDIMAGE) muß als erste Plotanweisung geschrieben sein. Alle Plotanweisungen greifen in der Ausführung auf Parameter zurück, die durch INIMAGE definiert werden. Die Ausführung eines neuen INIMAGE kann sich nicht nur auf den Inhalt des Puffers sondern auch des Datenfiles auswirken und bedeutet immer die Vorbereitungen für ein neues Bild und damit das Löschen des gespeicherten Bildes.

Nach der Ausführung der INIMAGE-Anweisung sind die Daten des Files nicht mehr verfügbar. Nach dem Speichern eines Bildes ist das File geschützt und kann nur als Plotfile verwendet werden. Als Datenfile ist es nur nach PURGE und erneutem CREATE einzusetzen.

Wenn EXTERNAL PLOTTER im Programm aufgeführt ist, wird INIMAGE bedeutungslos und daher überlesen.

Fehlermeldungen: Das File "Filename" existiert nicht (ERROR 236).

Das File "Filename" ist nicht sequentiell (ERROR 207).

Das File "Filename" ist gesichert (ERROR 199).

Das File ist kleiner dimensioniert als der Puffer (ERROR 249).

Nach der Preexecution ist der noch freie Speicher kleiner als 1280 Bytes (ERROR 244).

Nach der Preexecution ist die Dimension des Files und der noch freie Speicherraum nicht ausreichend (ERROR 237).

Overflow des Zentralspeichers (ERROR 181).

IPLOT

Anweisung IPLOT (Incrementplot)

Funktion : Zeichnen einer Verbindungslinie zu dem Punkt, der durch Inkremente dx und dy erreicht wird

Format : IPLOT dx, dy

mit : "dx" und "dy" als numerische "Ausdrücke"

Wirkung : Die numerischen Werte "dx" und "dy" bezeichnen nicht Koordinaten sondern Inkremente, d.h. die Koordinaten dx/dy bezüglich der Koordinaten X/Y des zuletzt angesprochenen Punktes (Koordinaten bezüglich des Ursprungs : X + dx/ Y + dy). Bis zu diesem so definierten Punkt wird linear interpoliert und eine Gerade gezeichnet.

Bemerkungen : Wenn noch kein Ausgangspunkt beim ersten Aufruf vorliegt, wird der Ursprung des Koordinatensystems als Ausgangspunkt angenommen.

Liegen diese linearen Elemente ganz oder teilweise außerhalb der Zeichenfläche, dann wird der Teil gezeichnet, der ins Bild paßt. Für außerhalb liegende Punkte wird das Maximum und Minimum festgestellt und als Hinweis nach Ausgabe des Bildes geschrieben.

Fehlermeldungen: Die Anweisung INIMAGE oder LDIMAGE wurde nicht gegeben (ERROR 241).

Die Operation konnte nicht ausgeführt werden :

Der Inhalt des Puffers kann nicht auf die Diskette geschrieben werden oder der Puffer ist für das Bild nicht groß genug. Der Fehler ist behebbar, die Punkte, die die Meldung hervorrufen, werden nicht gezeichnet (ERROR 250).

LDIMAGE

Anweisung LDIMAGE (Loadimage)

Funktion : Definition eines Plotfiles mit Übernahme des letzten gespeicherten Bildes mit den Parametern aus diesem File und Aufbau des Arbeitsspeichers für Plot-Anweisungen.

Format : LDIMAGE Filename

mit : "Filename" als Namen des Plotfiles.

Wirkung : Das in dem File "Filename" gespeicherte Plot-Bild wird geladen, kann anschließend ergänzt werden und wird dann zurückgespeichert. Das File muß als sequentielles Datenfile in einer der Bibliotheken kreiert sein und ist nach dem ersten Speichervorgang nur noch für Plotoperationen offen.

In den Arbeitsspeicher werden die Parameter für die Plotanweisungen geladen. Die Parameter sind die gleichen, wie sie bei INIMAGE beschrieben sind. Es gelten die Werte, die zum Zeitpunkt des Abspeicherns für das Bild definiert waren. Der Arbeitsspeicher enthält auch Informationen, wie den neuen Koordinatenursprung und den zuletzt markierten Punkt. Wurde ein Puffer im Arbeitsspeicher durch INIMAGE dimensioniert, gilt dieser auch für LDIMAGE.

Bemerkungen : Die Anweisung LDIMAGE (oder alternativ INIMAGE) muß die erste ausgeführte Plotoperation im Programm sein. Alle Plotanweisungen benötigen Informationen, die durch LDIMAGE bereitgestellt sind. Nach der Ausführung der LDIMAGE-Anweisung kann das geladene Bild ergänzt werden. Es können neue Punkte dazu gezeichnet werden, aber bereits vorhandene Punkte können nicht gelöscht werden.

Wenn in ein File ein Bild gespeichert wurde, ist es geschützt und kann nicht wie ein gewöhnliches sequentielles Datenfile benutzt werden.

Steht im Programm die Anweisung EXTERNAL PLOTTER, wird LDIMAGE bedeutungslos und überlesen.

Fehlermeldungen: Das File "Filename" existiert nicht (ERROR 187).

Das File "Filename" ist nicht sequentiell oder nicht zum Speichern eines Bildes vorbereitet (ERROR 252).

Overflow im Zentralspeicher (ERROR 184).

MOVE

Anweisung MOVE

Funktion : Positionieren auf einen durch Koordinaten definierten Punkt

Format : MOVE X, Y

mit : "X" und "Y" als numerische "Ausdrücke", die Abszisse und Ordinate eines Punktes bezeichnen

Wirkung : Die numerischen Werte für "X" und "Y" stehen für die Abszisse und Ordinate eines Punktes. Der Punkt wird nicht markiert. Er wird als Ausgangspunkt für nachfolgende Plotoperationen gespeichert.

Bemerkungen : Ein Positionieren kann aus der Zeichenfläche herausführen. In diesem Falle werden das Maximum oder Minimum oder Abszissen und Ordinaten der außenliegenden Punkte als Hinweis nach dem Plot ausgegeben.

Fehlermeldungen: Die Anweisung INIMAGE oder LDIMAGE wurde nicht gegeben (ERROR 241).

OFFSET

Anweisung OFFSET

Funktion : Verschiebt den Koordinatenursprung im Koordinatensystem.

Format : OFFSET X, Y

mit : "X" und "Y" numerische "Ausdrücke"

Wirkung : Die numerischen Werte "X" und "Y" sind die neuen Koordinaten für eine Verschiebung des Koordinatenursprungs und legen damit ein neues rechtwinkliges Koordinatensystem fest.

Die Koordinaten aller Punkte nach der OFFSET-Anweisung beziehen sich auf dieses neue Koordinatensystem. Der Ursprung des Koordinatensystems darf auch außerhalb der Zeichenfläche liegen.

Bemerkungen : Der neue Ursprung, der durch OFFSET definiert wird, bezieht sich immer auf den Koordinatenursprung, der aus der letzten SCALE-Anweisung resultiert, auch wenn bereits zuvor eine andere OFFSET-Anweisung ausgeführt wurde.

Der Koordinatenursprung kann durch die Anweisung OFFSET oder ein neues SCALE geändert werden.

Fehlermeldungen: Die Anweisung INIMAGE oder LDIMAGE wurde nicht gegeben (ERROR 241).

PLOT

Anweisung PLOT

Funktion : Zeichnen einer Geraden zu dem durch Koordinaten definierten Punkt.

Format : PLOT X, Y

mit : "X" und "Y" als numerische "Ausdrücke".

Wirkung : Die numerischen Werte "X" und "Y" stehen für die Abszisse und Ordinate eines Punktes. Es wird eine Gerade gezeichnet, die von dem zuletzt gespeicherten Punkt zu dem in den Parametern definierten Punkt führt. Danach gilt diese Koordinate als Ausgangspunkt für die nächste Plotoperation.

Bemerkungen : Ist noch kein Ausgangspunkt vorhanden, wird als Ausgangswert der Ursprung des Koordinatensystems verwendet. Liegen diese linearen Elemente ganz oder teilweise außerhalb der Zeichenfläche, dann wird nur der Teil gezeichnet, der ins Bild paßt. Für außerhalb liegende Punkte wird das Maximum und (oder) Minimum festgestellt und als Hinweis nach der Ausgabe des Bildes geschrieben.

Fehlermeldungen: Die Anweisung INIMAGE oder LDIMAGE wurde nicht gegeben (ERROR 241).

Die Operation konnte nicht ausgeführt werden :

Der Inhalt des Puffers kann nicht auf die Diskette geschrieben werden oder der Puffer ist für das Bild nicht groß genug. Der Fehler ist behebbar, die Punkte, die die Meldung hervorrufen, werden nicht gezeichnet (ERROR 250).

SCALE

Anweisung SCALE

Funktion : Festlegen der Einheiten des Koordinatensystems durch Minimum und Maximum der beiden Achsen; implizit werden dadurch Maßstabsfaktoren und der Koordinatenursprung definiert.

Format : SCALE X-min, X-max, Y-min, Y-max

mit : "X-min", "X-max", "Y-min", "Y-max" numerische Ausdrücke und den Bedingungen X-min $<$ X-max und Y-min $<$ Y-max

Wirkung : Die numerischen Werte für "X-min", "X-max", "Y-min" und "Y-max" definieren die Zeichenfläche als X- und Y-Intervall. Die Breite der Zeichenfläche in Zoll, in der Anweisung FRAME festgelegt oder als Standardparameter angenommen, zeigt das Intervall (X-min - X-max) an. In diesem Fall gilt als Maßeinheit für die Abszisse :

(Breite/ (X-max - X-min)) Zoll

Analog gilt für die Ordinate die Maßeinheit :

(Höhe/ (Y-max - Y-min)) Zoll

Implizit bestimmen die Werte X-min, X-max, Y-min und Y-max die Lage des Nullpunktes (Koordinatenursprung).

Bemerkungen : Der Koordinatenursprung darf auch außerhalb der Zeichenfläche liegen.

Die Maßeinheit für beide Achsen kann unterschiedlich sein und durch eine neue SCALE-Anweisung geändert werden.

Fehlt das SCALE, wird als Maßeinheit 1/70 Zoll für beide Achsen genommen. Als Koordinatenursprung wird die Bildmitte mit den Parametern

X-min = - (Breite/2) 70
X-max = (Breite/2) 70
Y-min = - (Höhe /2) 70
Y-max = (Höhe /2) 70

gesehen.

Fehlermeldungen: X-min > = X-max oder Y-min > = Y-max (ERROR 247).

Die Anweisung INIMAGE oder LDIMAGE wurde nicht gegeben (ERROR 241).

XAXIS

Anweisung XAXIS

Funktion : Zeichnen einer Parallelen zur X-Achse.

Format : XAXIS Y [, t] [, X1 , X2]

mit : Y , t , X1 und X2 als numerische "Ausdrücke" und t $\neq$ 0

Wirkung : Es wird eine Parallele zur X-Achse mit der Ordinate Y gezeichnet. Mit X1 als Startpunkt und X2 als Endpunkt wird das Intervall für die X-Achse definiert. Fehlt diese Information, dann wird die Parallele zur Achse über die gesamte Zeichenfläche gezeichnet. Ist t angegeben, wird die Achse vom Startpunkt beginnend mit Markierungen im Abstand t versehen.

Bemerkungen : Ist X1 < X2 und t negativ oder auch X1 > X2 und t positiv oder $|t| > |X2 - X1|$ dann wird die Achse nicht markiert.

Wird die Achse ganz oder teilweise außerhalb der Zeichenfläche definiert, dann wird nur der Teil gezeichnet, der ins Bild paßt. Für außerhalb liegende Punkte wird das Maximum und (oder) Minimum festgestellt und als Hinweis nach der Ausgabe des Bildes geschrieben.

Fehlermeldungen: Die Anweisungen INIMAGE oder LDIMAGE wurde nicht gegeben (ERROR 241).

Die Operation konnte nicht ausgeführt werden :

Der Inhalt des Puffers kann nicht auf die Diskette geschrieben werden oder der Puffer ist für das Bild nicht groß genug. Der Fehler ist behebbar, die Punkte, die die Meldung hervorrufen, werden nicht gezeichnet (ERROR 250).

YAXIS

Anweisung YAXIS

Funktion : Zeichnen einer Parallelen zur Y-Achse.

Format : YAXIS X [, t] [, Y1 , Y2]

mit : X, t, Y1 und Y2 als numerischen Variablen oder Ausdrücken
und $t \neq 0$

Wirkung : Es wird eine Parallele zur Y-Achse mit der Abszisse X gezeichnet. Mit Y1 als Startpunkt und Y2 als Endpunkt wird das Intervall für die Y-Achse definiert. Fehlt diese Information, dann wird die Parallele zur Achse über die gesamte Zeichenfläche gezeichnet. Ist t angegeben, wird die Achse von Y1 beginnend mit Markierungen im Abstand t versehen.

Bemerkungen : Ist $Y1 < Y2$ und t negativ ($Y1 > Y2$ und t positiv) oder $|t| > Y2 - Y1$, dann wird die Achse nicht markiert.

Wird die Achse ganz oder teilweise außerhalb der Zeichenfläche definiert, dann wird nur der Teil gezeichnet, der ins Bild paßt. Für außerhalb liegende Punkte wird das Maximum und (oder) Minimum festgestellt und als Hinweis nach der Ausgabe des Bildes geschrieben.

Fehlermeldungen: Die Anweisungen INIMAGE oder LDIMAGE wurde nicht gegeben (ERROR 241).

Die Operation konnte nicht ausgeführt werden :

Der Inhalt des Puffers kann nicht auf die Diskette geschrieben werden oder der Puffer ist für das Bild nicht groß genug. Der Fehler ist behebbar, die Punkte, die die Meldung hervorrufen, werden nicht gezeichnet (ERROR 250).

13.6 BEISPIELE MIT OPTION PLOT

13.6.1 Aufbau eines Plot-Programmes

Integrierter Printer		Externer Plotter
	1. Vereinbarungsteil	
INIMAGE	LDIMAGE	-
FRAME	-	FRAME
SCALE	SCALE	SCALE
	2. Plot-Anweisungen	
OFFSET		OFFSET
MOVE		MOVE
XAXIS		XAXIS
YAXIS		YAXIS
PLOT/IPLOT		PLOT/IPLOT
DOT/IDOT		DOT/IDOT
	3. Generierung von Strings	
CSIZE		CSIZE
CPLOT		CPLOT
CTAB		CTAB
	4. Ausgabe des Plot-Bildes	
DRAW		(wird simultan ausgeführt)

13.6.2 Die Logarithmus-Funktion

Die Funktion LOG(X) ist zu zeichnen im Intervall 0.01 <= X <= 10. Die Berechnung der Funktion erfolgt in 2 Abschnitten : im Intervall 0.01 bis 1 mit der Schrittweite 0.Ø1, im Intervall 1 bis 10 mit der Schrittweite 0.5.

Zur Speicherung des Bildes wird das File DIS genommen, der Puffer im Arbeitsspeicher wird mit 4 K-Byte aufgebaut.

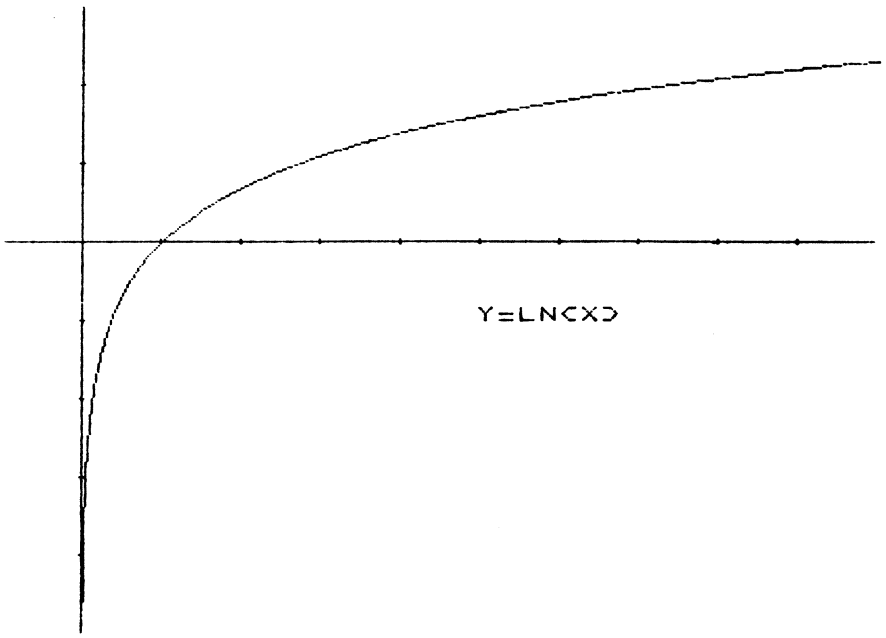
Das Koordinatensystem wird gezeichnet für :

X-min = - 1; X-max = 10
Y-min = - 5; Y-max = 3

Soll keine Verzerrrung durch das Verhältnis X : Y = 11 : 8 Einheiten auftreten, dann muß mit der Anweisung FRAME für einen einheitlichen Maßstab gesorgt werden. Bei einer Breite für das Bild von 6 Zoll ergibt sich für die Höhe 4.363636 Zoll nach dem Verhältnis

11 : 8 = 6 : X

```
FILE    *LN

0010 INIMAGEDIS,4
0020 FRAME 6,4.363636
0030 SCALE -1,10,-5,3
0040 XAXIS 0,1,-1,10
0050 YAXIS 0,1,-5,3
0060 MOVE 5,-1
0070 CPLOT "Y=LN(X)"
0080 MOVE .01,LOG(.01)
0090 FOR X=.01 TO 1 STEP .01
0100 PLOT X,LOG(X)
0110 NEXT X
0120 FOR X=1 TO 10 STEP .5
0130 PLOT X,LOG(X)
0140 NEXT X
0150 DRAW
0160 END

END OF LISTING
```



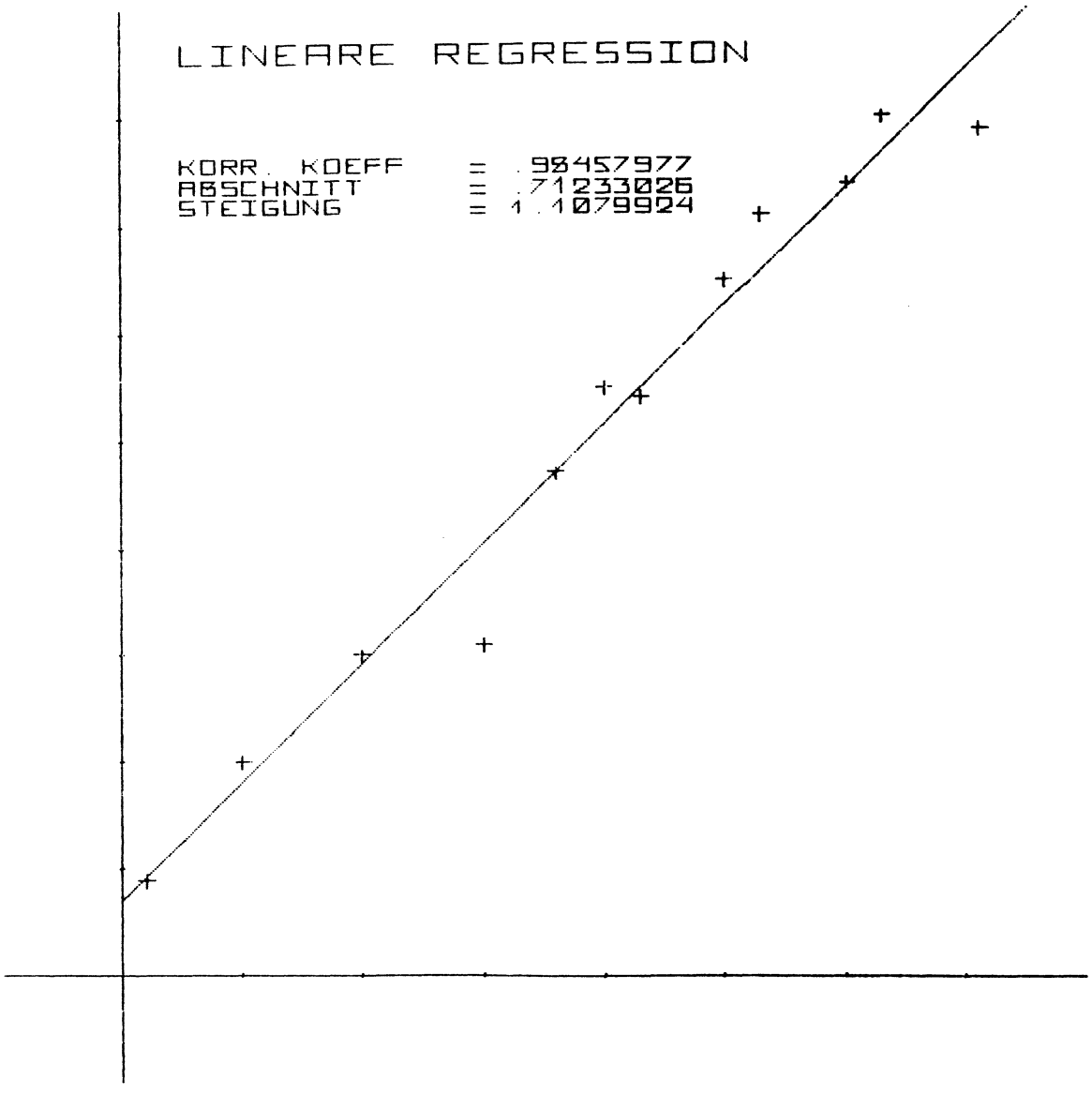
13.6.3 Lineare Regression

Eine Punktfolge (X, Y) wird auf die Zeichenfläche übertragen, die Punkte werden durch "+" in der Funktion FNC markiert. Die Regressionsgerade wird berechnet, gezeichnet und die Kennwerte für die Regression werden mit CPLOT ausgegeben.

```
   .
   .
   .
0900 INIMAGEDIS,4
0910 FRAME 8,8
0920 SCALE -1,10,-1,10
   .
   .
   .
0950 MOVE 0.5,9
0960 CSIZE 0.25,0.25,0
0970 CPLOT "LINEARE REGRESSION"
0980 CSIZE 1/6,1/6,0
0990 BUILD T$,R
1000 MOVE 0.5,8.5
1010 CPLOT "KORR. KOEFF. ="+T$
1020 BUILD T$,A1
1030 CPLOT "ABSCHNITT    ="+T$
   .
   .
   .
1060 PRINT TAB(20);"D A T E N P A A R E"
1070 PRINT
1080 PRINT
1090 PRINT
1100 PRINT TAB(13);"PAAR  "," X"," Y"
1110 PRINT
1120 SETW:1 TO 13
1130 FOR I=1 TO N2 STEP 1
1140 READ:1,X,Y
1150 LET K9=FNC(X,Y)
1160 NEXT I
1170 PLOT 7.1,A1+B1*7.1
1180 DRAW
   .
   .
   .
1500 DEF FNC(X,Y)
1510 MOVE X-0.1,Y
1520 PLOT X+0.1,Y
1530 MOVE X,Y+0.1
1540 PLOT X,Y-0.1
1550 LET FN*=1
1560 FNEND
   .
   .
   .
9900 END
```

D A T E N P A A R E :

PAAR	X	Y
1	1	2
2	2	3
3	3	3.1
4	4	5.5
5	5	6.5
6	6	7.4
7	5.3	7.1
8	.2	.9
9	7.1	7.9
10	6.29	8.01
11	3.6	4.7
12	4.31	5.4



ANHANG

Hexadezimal		0	1	2	3	4	5	6	7
	Dual	0000	0001	0010	0011	0100	0101	0110	0111
0	0000	0 NUL	16 DLE	32 SP	48 0	64 @	80 P	96 `	112 p
1	0001	1 SOH	17 DC1	33 !	49 1	65 A	81 Q	97 a	113 q
2	0010	2 STX	18 DC2	34 "	50 2	66 B	82 R	98 b	114 r
3	0011	3 ETX	19 DC3	35 #	51 3	67 C	83 S	99 c	115 s
4	0100	4 EOT	20 DC4	36 $	52 4	68 D	84 T	100 d	116 t
5	0101	5 ENQ	21 NAK	37 %	53 5	69 E	85 U	101 e	117 u
6	0110	6 ACK	22 SYN	38 &	54 6	70 F	86 V	102 f	118 v
7	0111	7 BEL	23 ETB	39 '	55 7	71 G	87 W	103 g	119 w
8	1000	8 BS	24 CAN	40 (	56 8	72 H	88 X	104 h	120 x
9	1001	9 HT	25 EM	41)	57 9	73 I	89 Y	105 i	121 y
A	1010	10 LF	26 SUB	42 *	58 :	74 J	90 Z	106 j	122 z
B	1011	11 VT	27 ESC	43 +	59 ;	75 K	91 [	107 k	123 {
C	1100	12 FF	28 FS	44 ,	60 <	76 L	92 /	108 l	124 |
D	1101	13 CR	29 CS	45 -	61 =	77 M	93]	109 m	125 }
E	1110	14 SO	30 RS	46 .	62 >	78 N	94 ^	110 n	126 ~
F	1111	15 SI	31 US	47 /	63 ?	79 O	95 _	111 o	127 DEL

LISTE DER DARSTELLBAREN ZEICHEN UND IHRE ENTSPRECHENDEN DEZIMALEN WERTE

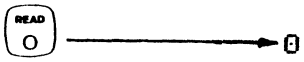
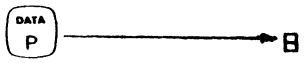
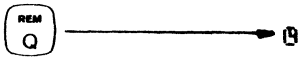
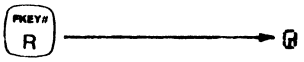
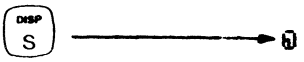
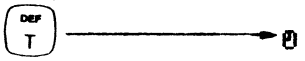
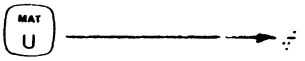
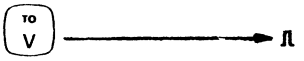
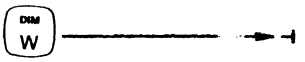
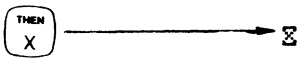
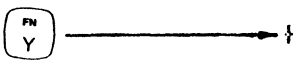
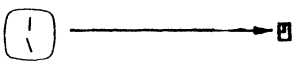
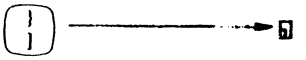
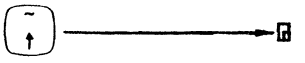
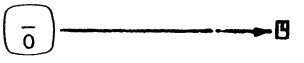
Dargestelltes Zeichen	Zugehörender dezimaler Wert
	0
	1
	2
	3
	4
	5
	6
	7
	8
	9
	10
	11
	12
	13
	14
	15
	16
	17
	18
	19
	20
	21
	22
	23
	24
	25
	26
	27
	28
	29
	30
	31
	32
!	33
"	34
#	35
$	36
%	37
&	38
'	39
(	40
)	41
*	42
+	43
,	44
-	45
.	46
/	47
0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57
:	58
;	59
<	60
=	61
>	62
?	63
@	64
A	65
B	66

Dargestelltes Zeichen	Zugehörender dezimaler Wert
C	67
D	68
E	69
F	70
G	71
H	72
I	73
J	74
K	75
L	76
M	77
N	78
O	79
P	80
Q	81
R	82
S	83
T	84
U	85
V	86
W	87
X	88
Y	89
Z	90
[	91
\	92
]	93
↑	94
←	95
`	96
a	97
b	98
c	99
d	100
e	101
f	102
g	103
h	104
i	105
j	106
k	107
l	108
m	109
n	110
o	111
p	112
q	113
r	114
s	115
t	116
u	117
v	118
w	119
x	120
y	121
z	122
{	123
|	124
}	125
~	126
◇	127
♠	128
♥	129
	.....
	.....
■	255

DARSTELLUNG DER ISO-ZEICHEN DER SPALTEN Ø UND 1 DER ISO-CODE-TABELLE

In der folgenden Tabelle ist dargestellt, welche Taste gemeinsam mit der Taste CONTROL zu drücken ist, um ein Zeichen der Spalten Ø und 1 der ISO-Tabelle darstellen zu können.

Tasten	Darstellung des Zeichens	entsprechendes ISO-Zeichen	Dezimalwert des ISO-Zeichens
CONTROL @	→	NUL	0
CONTROL PRINT A	→	TC1 (T01)	1
CONTROL STEP B	→	TC2 (T02)	2
CONTROL FOR C	→	TC3 (T03)	3
CONTROL USING D	→	TC4 (T04)	4
CONTROL DCL E	→	TC5 (T05)	5
CONTROL WRITE: F	→	TC6 (T06)	6
CONTROL ON G	→	BEL	7
CONTROL GOTO H	→	FEØ	8
CONTROL INPUT I	→	FE1	9
CONTROL GOSUB J	→	FE2	10
CONTROL RETURN K	→	FE3	11
CONTROL STOP L	→	FE4	12
CONTROL END M	→	FE5	13
CONTROL NEXT N	→	SO	14

Tasten	Darstellung des Zeichens	entsprechendes ISO-Zeichen	Dezimalwert des ISO-Zeichens
CONTROL			
READ O		SI	15
DATA P		DLE	16
REM Q		DC1	17
FKEY# R		DC2	18
DISP S		DC3	19
DEF T		DC4	20
MAT U		NAK	21
TO V		SYN	22
DIM W		ETB	23
THEN X		CAN	24
FN Y		EM	25
IF Z		SUB	26
{ [		ESC	27
| \		IS4	28
}]		IS3	29
~ ↑		IS2	30
_ 0		IS1	31

LISTE DER FEHLERCODES

Fehlermeldungen werden auf Grund eines numerischen Fehlercodes identifiziert. Bei Fehlermeldungen der Preexecution und bei den Ausführungsfehlern wird- wenn möglich - zusätzlich die Zeilennummer der Programmzeile, in der der Fehler auftrat, angegeben. (Beispiel: ERROR 6 IN LINE 155).

Im folgenden Abschnitt werden die möglichen Fehlercodes mit ihrer Beschreibung angegeben, wobei die Liste in die oben beschriebenen Gruppen unterteilt ist.

Fehler mit den Nummern 1 bis 16 beziehen sich auf behebbare Fehler bei der Ausführung eines BASIC-Programmes; Fehler mit Codes von 40 bis 55 können bei der Preexecutuion eines Programmes auftreten.

Die Fehlercodes von 65 bis 98 bezeichnen nicht behebbare Ausführungsfehler, Codes zwischen 100 und 128 können bei der fehlerhaften Eingabe von BASIC-Zeilen oder der Compilierung eines Textfiles auftreten.
Die Fehlercodes 151 und 152 zeigen an, daß im Zugriff auf eine Diskette ein Fehler auftrat.

Nicht behebbare Fehler in der Vebindung mit peripheren Einheiten werden durch die Codes 162 bis 171 gemeldet, und Fehler, die bei der Eingabe oder Ausführung von Systembefehlen auftreten, haben Nummern zwischen 181 und 214.

Die Codes 232 bis 235 zeigen Fehler in der Ausführung von Dienstprogrammen an, behebbare und nicht behebbare Fehler der Plot-Option werden in den Codes 236 bis 254 beschrieben.

Abschließend werden noch die Fehlercodes, die einen Systemabbruch anzeigen und die Meldungen für nicht betriebsbereite Einheiten, angeführt.

LISTE DER FEHLERCODES

1. Behebbare Fehler während der Ausführung von BASIC-Programmen

Fehlercode	Bedeutung
1	Eine numerische oder alphanumerische Variable wurde nicht initialisiert. Es wird für das auszuführende Statement der Wert 0 bzw. der Nullstring angenommen, die Variable gilt jedoch weiterhin als nicht initialisiert.
2	Fehlerhaftes Argument in einer Stringfunktion.
3	Overflow. Als Ergebnis der Operation wird die größte darstellbare Zahl mit dem richtigen Vorzeichen angenommen.
4	Underflow. Als Ergebnis der Operation wird Null angenommen.
6	Entweder ist das Argument einer Funktion außerhalb der zulässigen Grenzen , oder es soll die Wurzel einer negativen Zahl berechnet werden. Es wird dann die Wurzel des Absolutbetrages berechnet.
7	Bei einer Stringoperation entsteht ein String mit einer Länge von mehr als 1023 Zeichen. Der String wird nach der 1023. Stelle abgeschnitten.
8	Der String, der an eine Stringvariable zugewiesen werden soll, ist länger als die deklarierte Länge der Stringvariablen. Der String wird entsprechend der deklarierten Länge der Stringvariablen abgeschnitten.
9	Es soll der Logarithmus einer negativen Zahl berechnet werden. Es wird der Logarithmus des Absolutbetrages berechnet.
10	Es wird versucht, den Logarithmus von Null zu berechnen. Als Ergebnis wird -9.999999999999 E+99 gesetzt.
11	Eine negative Zahl hat einen nicht ganzzahligen Exponenten. Es wird mit dem Absolutbetrag der Basis gerechnet.

Fehlercode	Bedeutung
12	Der Wert Null hat einen negativen Exponenten. Als Ergebnis wird die größte darstellbare Zahl gesetzt.
13	Die zu invertierende Matrix ist singulär (d.h. die Determinante ist gleich Null). Die Ergebnismatrix enthält nicht definierte Werte, eine nachfolgende Verwendung der Funktion DET liefert jedoch ein korrektes Ergebnis.
14	Die adressierte Peripherie ist nicht angeschlossen oder eingeschaltet, oder es tritt während des Betriebes eine abnormale Kondition auf.
15	Übertragungsfehler während eines Datenaustausches
16	Beim letzten überlappten Input/Output für die gleiche Peripherie ist ein Übertragungsfehler aufgetreten

2. Fehler bei der Preexecution eines Programmes

40	Unerlaubter Sprung: a) Sprung zu einer nicht existierenden Zeilennummer b) Sprung aus einer mehrzeiligen Funktion oder von außen in den Vereinbarungsteil einer mehrzeiligen Funktion c) Sprung von außen in eine FOR/NEXT-Schleife
41	NEXT ohne zugehörendes FOR oder Überschneidung von FOR/NEXT-Schleifen.
42	Innerhalb der Definition einer mehrzeiligen Funktion wird eine andere mehrzeilige Funktion definiert.
43	Eine aufzurufende Funktion wurde nicht definiert.
44	Es sind mehr als 15 FOR/NEXT-Schleifen geschachtelt.

Fehlercode	Bedeutung
45	a) FN *, FN *$, FNEND treten außerhalb der Definition einer mehrzeiligen Funktion auf. b) In einer Definition einer mehrzeiligen Stringfunktion wird die Pseudovariable FN * oder in der Definition einer mehrzeiligen numerischen Funktion wird die Pseudovariable FN * $ verwendet.
46	Zwei oder mehrere ineinander geschachtelte FOR/NEXT-Schleifen haben dieselbe Laufvariable.
47	FOR ohne nachfolgendes NEXT.
48	In der Definition einer mehrzeiligen Funktion fehlt die FNEND-Anweisung.
49	Eine einfache indizierte und eine doppelt indizierte Variable haben denselben Namen.
50	END hat nicht die höchste Zeilennummer im Programm.
51	Das Programm enthält keine END-Anweisung.
52	Das auszuführende Programm enthält nicht compilierte Zeilen. Der Fehler tritt auf, falls die bei der Compilierung eines Textfiles gemeldeten Fehler nicht korrigiert wurden.
53	In der Definition einer mehrzeiligen Funktion erfolgt keine Wertzuweisung an die Pseudovariable FN * bzw. FN * $.
54	Für eine PRINTUSING-Anweisung (oder DISPUSING ...) fehlt die zugehörende Formatvereinbarung.
55	Innerhalb der Definition einer mehrzeiligen Funktion befindet sich eine STOP-Anweisung.

3. Nicht behebbare Fehler während der Ausführung eines BASIC-Programmes

Fehlercode	Bedeutung
65	Während der Programmausführung wird der zur Verfügung stehende Bereich des Anwenderspeichers überschritten (z.B. bei zu tiefer Schachtelung von Unterprogrammen).
66	Der Index einer indizierten Variablen hat einen nicht zulässigen Wert (d.h. kleiner als 1 oder größer als die maximale Dimension dieser Variablen).
67	Die Anweisung verlangt die Vereinbarung von Dimensionen einer Matrix, die größer sind als die maximalen Dimensionen.
68	Beim Aufruf der Programmausführung beginnend mit einer angeführten Zeilennummer (nach RUN Zeilennummer, oder START Zeilennummer) tritt ein NEXT ohne zugehörendes FOR auf.
69	Die Parameter im Aufruf einer Funktion stimmen nicht in ihrem Typ mit den Parametern der Definition dieser Funktion überein.
70	Es soll ein RETURN-Statement ohne vorangehendes GOSUB ausgeführt werden.
71	Die Summe der Längen der Strings, die den Funktionstasten zugewiesen werden sollen, ist größer als die maximal zulässige Länge (238 Byte).
72	Die Anzahl der Parameter in einem Funktionsaufruf stimmt nicht überein mit der Anzahl der Parameter in der Definition der Funktion.
73	Die aktuellen Dimensionen einer Matrix erlauben nicht die Ausführung der gewünschten Operation (z.B. nicht quadratische Matrix bei einer Inversion).
74	Innerhalb einer definierten Funktion wurde die maximal zulässige Anzahl von Aufrufen anderer definierter Funktionen (max. 255) überschritten.
75	Die zur Ausführung des Programmes benötigten Options wurden nicht in den Hauptspeicher geladen.

Fehlercodes	Bedeutung
76	Es wird versucht, ein File zu öffnen, das bei einer vorherigen Ausführung eines Programmes nicht geschlossen wurde. Das File muß mit dem Befehl VALIDATE geschlossen werden.
77	Es fehlt entweder das FILES-Statement oder der Filedesignator ist kleiner als 1 oder größer als die Anzahl der Files in der FILES-Anweisung.
78	Ein File ist nicht entsprechend dem Typ der auszuführenden Operation geöffnet.
80	Der Wert des Wortdesignators einer SETW: - Anweisung ist größer als es der Länge des entsprechenden Files entspricht.
81	Innerhalb eines Programmes wird versucht, ein bereits offenes File zu öffnen.
82	Der verfügbare Platz in einem externen File reicht nicht für die Ausführung der verlangten Operation.
83	Das angesprochene File ist nicht geöffnet.
84	Es wurde bei einer Lese- oder Schreiboperation das Fileende ohne EOF-Angabe erreicht.
85	Das Argument einer TAB-Funktion in einer PRINT oder DISP - Anweisung ist kleiner als 1.
86	Einer numerischen Variablen soll ein String zugewiesen werden.
87	Die deklarierte Länge der Stringvariablen in einer BBUILD-Anweisung reicht nicht für die Ausführung der Anweisung.
88	Zu wenig Daten im internen File bei der Ausführung einer READ-Anweisung oder im Stringausdruck bei der Ausführung von ASSIGN.

Fehlercode	Bedeutung
89	Die Formatfelder stimmen in ihrem Typ nicht mit den Daten der aufrufenden PRINT USING oder DISP USING Anweisung überein.
90	Ein Wert kleiner als 0 oder größer als 255 soll in ein ISO-Zeichen umgewandelt werden.
91	Der Ausdruck, der die Anzahl der in einer CONVERT-Anweisung umzuwandelnden Vectorelemente festlegt, ergibt einen negativen Wert.
92	Der in einer CHAIN-Anweisung aufgerufene Filename existiert nicht.
93	Die Daten eines externen Files stimmen in ihrem Typ nicht mit den Variablen der entsprechenden READ:-Anweisung überein. Derselbe Fehler tritt auch bei der Ausführung einer BASSIGN-Anweisung auf, falls der Typ der Daten und der entsprechenden Variablen nicht übereinstimmt.
96	Der Wortdesignator einer SETW:-Anweisung hat einen negativen Wert oder ist null.
97	Eine SCRATCH: - oder APPEND: - Anweisung bezieht sich auf ein Randomfile.
98	Die Anzahl der angegebenen Parameter in einer WHERE: - Anweisung verlangt, daß das sequentielle File für das Lesen geöffnet ist.

4. Fehlermeldungen des BASIC-Compilers

Fehlercode	Bedeutung
100	Eine Programmzeile besteht nur aus der Zeilennummer.
101	Zeilennummer mit syntaktischem Fehler.
102	Unzulässiges Schlüsselwort
103	Unzulässiger Parameter
104	Unzulässiger Ausdruck
105	Der Operator ist für den Typ der Operanden nicht zulässig.
106	In einem Aufruf einer Funktion stimmt die Anzahl oder Typ der Parameter nicht mit den verlangten Parametern überein.
109	Nicht interpretierbarer Syntaxfehler
110	Der Name der Funktion in der DEF-Anweisung kommt bereits in einer DEF-Anweisung mit einer anderen Zeilennummer vor.
111	Es wurden bereits zuviele verschiedene Zeilennummern als Sprungziele definiert. Es sind maximal 255 Sprungzeile zulässig, jeder Funktionsaufruf gilt als eine Verzweigung.
112	Die Anzahl der verwendeten Variablen übersteigt die maximal zulässige Anzahl(max. 123 numerische bzw. 255 Stringvariable).
113	Unzulässiges Zeichen (z.B. bei verschiedener Anzahl von öffnenden und schließenden Klammern in einem numerischen Ausdruck).
114	Rekursiver Aufruf in der Definition einer einzeiligen Funktion.

Fehlercode	Bedeutung
115	Unzulässiger Bezug auf eine Variable oder eine Funktion.
117	Speicherüberschreitung bei der Eingabe eines Programmes oder Textes.
118	Es wurde bereits eine FILES-Anweisung mit einer anderen Zeilennummer eingegeben.
119	Es wurde bereits die maximal zulässige Anzahl von Funktionen definiert.
120	Die Zeilennummer, auf die Bezug genommen wird, existiert nicht.
128	Unzureichender Platz für die Compilierung der eingegebenen Zeile.

5. Fehlermeldungen im Zusammenhang mit den Diskettenstationen

Fehlercode	Bedeutung
151	Fehler auf der Floppy-Disk-Einheit 1 (obere Station)
152	Fehler auf der Floppy-Disk-Einheit 2 (untere Station)

6. Nicht behebbare Fehler im Zusammenhang mit peripheren Einheiten

Fehlercode	Bedeutung
162	SEND für Input-Einheit oder RECEIVE für Output-Einheit.
163	Der mit SEND zu übertragende String ist größer als der Puffer des entsprechenden Kanals.
165	Das Statement bezieht sich auf einen IPSO-Kanal, der in der Konfiguration nicht vorhanden ist.
166	Für den angesprochenen Kanal wurde kein Puffer definiert.
167	Es wurde eine negative Peripherieadresse oder eine negative Command Nummer angegeben.
169	Eine Peripherieadresse oder eine Commandnummer sind größer als 255.
170	Die in einem RECEIVE-Statement angegebene Stringvariable ist länger als der Puffer des entsprechenden Kanals.
171	Der mit CONFIGURE angegebene externe Drucker ist nicht betriebsbereit.

7. Fehler bei der Eingabe oder Ausführung von Systembefehlen

Fehlercode	Bedeutung
172	Für die EVD-Option des Befehles CONFIGURE muß ein RS 232-C Interface vorhanden sein.
175	Im Befehl CONFIGURE wird mehr Speicherkapazität verlangt, als im System vorhanden ist.
176	Im Befehl CONFIGURE wird ein im System nicht vorhandener externer IPSO-Drucker verlangt.

Fehlercode	Bedeutung
181	Der zur Verfügung stehende Speicher reicht nicht für die Ausführung des Befehles.
182	Beim Befehl TRANSCODE wurde bei der Umwandlung eines Datenfiles in ein Textfile der Parameter # angegeben, es sind jedoch im Datenfile keine Zeilennummern enthalten.
183	Die angesprochene Bibliothek ist auf der Diskette nicht vorhanden, oder es wurde die zulässige maximale File-Anzahl für eine Bibliothek überschritten.
184	Die User Disk ist entweder nicht initialisiert, oder es wird versucht, auf eine User-Disk zuzugreifen, ohne daß eine eingelegt wurde.
185	Nicht initialisierte System-Disk.
186	In der Bibliothek existiert bereits ein File mit gleichem Namen.
187	Das File mit dem angegebenen Namen wird nicht gefunden.
188	Auf der Diskette ist zu wenig Platz für die Ausführung des Befehls.
189	Für ein File, das signifikante Information enthält, soll der reservierte Speicher verkleinert werden.
190	Der Befehl wird im gegenwärtigen Status des Systems nicht akzeptiert.
191	Es wurde kein Filename angegeben oder das File im Arbeitsspeicher hat keinen Namen.
192	Unzulässiges Zeichen im Befehl.
193	Es fehlen Parameter im Befehl.
194	Aufruf einer nicht existierenden Zeilennummer.
195	Im Debugging-Mode wird der Befehl "START Zeilennummer" nach einem Befehl "RUN Filename" nicht akzeptiert, da das auszuführende Programm ohne vorhergehende Preexecution auf der Diskette abgespeichert ist. Das gilt auch für den Befehl "PREPARE Filename".

Fehlercode	Bedeutung
196	Unzulässiger oder nicht signifikanter Parameter.
197	Der Befehl bezieht sich auf eine Zeilennummer, die innerhalb der Definition einer mehrzeiligen Funktion liegt.
198	Es wird mehr Speicherplatz verlangt, als auf der Diskette vorhanden ist.
199	Die verlangte Operation ist für ein geschütztes File nicht zulässig.
200	Unerlaubte Operationen, da die Bibliothek geschützt ist.
201	Der Befehl ist nicht ausführbar, da das System als Mono-Disk-System initialisiert ist.
202	In der Ausführung eines Befehls müßte ein File in mehr als 4 Teile aufgeteilt werden, was unzulässig ist.
203	In einem LIST- oder DELETE-Befehl ist die erste Zeilennummer größer als die zweite.
205	Unerlaubte Operation, da die Zeile geschützt ist.
206	Das File im Arbeitsspeicher ist kein Programm.
207	Die Operation ist für den angesprochenen Filetyp nicht ausführbar.
208	Die im OPTION-Befehl angeführte Option ist im System nicht vorhanden.
209	Es wurde eine Zeilennummer größer als 9999 erzeugt.
210	In einem LIST-Befehl wurde für den Ausdruck eines Programmes der Parameter X angegeben.
211	Leerer Arbeitsspeicher.
212	Die zu druckende(n) Zeile(n) existieren nicht.
213	Bei der Decompilierung einer Zeile tritt ein Line-Overflow auf.

Fehlercode	Bedeutung
214	Im Unterprogramm, das als mehrzeilige Funktion mit LINK angefügt werden soll, ist die erste Zeile kein DEF-Statement.

8. Fehler beim Aufruf oder der Ausführung von Dienstprogrammen

Fehlercode	Bedeutung
232	In einem Aufruf von LBCREATE ist die Anzahl der Sektoren , die für den Katalog reserviert werden sollen, größer als $14(n_1 + n_2 + n_3)$.
234	Es fehlt die Angabe des Namens des Dienstprogrammes.
235	Das aufgerufene Dienstprogramm existiert nicht.

9. Fehlermeldungen die sich auf PLOT-Operation beziehen.

Fehlercode	Bedeutung
236	behebbar: Das in INIMAGE genannte File existiert nicht, das Bild wird nur im Arbeitsspeicher generiert.
237	behebbar: Das genannte File ist nicht sequentiell oder zu klein. Nach der Preexecution bleibt nur eine Anwenderkapazität kleiner als 1280 bytes. Das Bild wird nur im Arbeitsspeicher generiert.

Fehlercode	Bedeutung
238	Für den externen Plotter fehlt die Funktion FNP (nicht behebbar).
239	In der Anweisung DRAW wird das Bild außerhalb des Zeichenbereiches verschoben (nicht behebbar).
240	nicht behebbar: Zur Ausführung von DRAW fehlt der Thermodrucker.
241	nicht behebbar:Es wurde weder INIMAGE noch LDIMAGE ausgeführt.
242	behebbar: FRAME folgt nicht unmittelbar auf INIMAGE.
243	behebbar: Als Abstand für die Markierung der Achsen wurde Ø angegeben. Der Operand wird ignoriert.
244	behebbar: Nach der Preexecution ist der verfügbare Speicherplatz für den Anwender kleiner als 1280 Bytes, die Ausführung kann dadurch langsamer werden.
245	nicht behebbar: In FRAME ist der Wert für die Breite nicht zulässig.
246	nicht behebbar: In FRAME ist der Wert für die Höhe nicht zulässig.
247	nicht behebbar: In SCALE ist X-Min =X-Max oder Y - Min = Y- Max.
248	nicht behebbar: Der in INIMIGAE vereinbarte Puffer ist für die in FRAME angegebene Größe des Bildes zu klein.
249	behebbar: Die für das File vereinbarte Größe ist kleiner als der Puffer im Arbeitsspeicher. Das Bild wird nur im Arbeitsspeicher erzeugt.

Fehlercode	Bedeutung
250	behebbar: 1.) Die Größe des Puffers oder des externen Files ist zu klein, sodaß keine weiteren Punkte gespeichert werden können. Alle folgenden Plot-Anweisungen mit Ausnahme von INIMAGE, LDIMAGE und DRAW werden ignoriert. 2.) In einem Programm, für das kein externes File zur Speicherung des Bildes existiert, soll ein DRAW-Statement zum zweiten Mal ausgeführt werden. Alle folgenden Plot-Anweisungen mit Ausnahme von INIMAGE und LDIMAGE werden ignoriert.
251	nicht behebbar: In der Anweisung CSIZE ergibt die Angabe von Breite oder Höhe einen negativen Wert.
252	nicht behebbar: Das File in der Anweisung LDIMAGE enthält kein Bild.
253	behebbar: Die angegebene Funktion enthält ein INIMAGE oder LDIMAGE-Statement.
254	behebbar: Bei der Ausführung der Zeichnung wurde die maximal mögliche Anzahl von Schreibvorgängen erreicht. Alle nachfolgenden Plotanweisungen mit Ausnahme von DRAW werden ignoriert.

10. Fehlermeldungen für einen nicht normalen Systemzustand

Fehlercode	Bedeutung
mA*	Falls m eine andere Zahl als 12 oder 16 ist, und der Fehler tritt auch bei anderen als den eingelegten Disketten auf, so liegt ein Hardwarefehler im Speicher vor.

Fehlercode	Bedeutung
12 A* 16 A*	Die eingelegte Systemdiskette ist beschädigt. Tritt der Fehler auch mit anderen Systemdisketten auf, liegt ein Hardwarefehler vor. Anmerkung: Fehlermeldungen, die nach dem numerischen Code A* haben, bewirken ein Löschen des Arbeitsspeichers. Nach Drücken der Taste CONTINUE wird das System neu geladen.
ABN FD*	Die obere Floppy-Disk-Station arbeitet nicht korrekt. Es ist zu überprüfen, ob der Einschub geschlossen und eine Diskette eingelegt wurde.
ABN FD**	Die untere Floppy-Disk-Station arbeitet nicht korrekt. Es ist zu überprüfen, ob der Einschub geschlossen und eine Diskette eingelegt wurde.
ABN PRT	Der integrierte Drucker oder der konfigurierte externe Drucker arbeitet nicht korrekt. Beim Thermodrucker ist zu überprüfen, ob sich der Schreibkopf in der richtigen Position befindet.
ABN FD - DCH OMITTED	Bei einer Diskettenstation wurde ohne vorherige Eingabe des DCHANGE-Befehles ein Einschub geöffnet. Der DCHANGE-Befehl muß bei Tausch der Disketten nachgeholt werden. Anmerkung: Statusmeldungen mit ABN können nach Beheben der Fehlerursache mit CLEAR gelöscht werden.

Alphabetisches Stichwortverzeichnis

Enthalten sind BASIC-Anweisungen, Systembefehle, Funktionen und Dienstprogramme. Im folgenden sprechen wir von

B = BASIC-Anweisungen und Funktionen

S = Systembefehl

D = Dienstprogramm

DEUTSCHE OLIVETTI GMBH · FRANKFURT-NIEDERRAD · LYONER STRASSE 34

Druck Nr. 04/513/P/11.78