ABS

WIRKUNG : Die Funktion ABS liefert den Absolutbetrag eines numerischen Ausdruckes.

BEISPIEL : Schalten Sie bitte mit Hilfe der MODE Taste den PC-1600 in den PRO-Modus und geben Sie folgendes kleine BASIC-Programm ein. Vermeiden Sie dabei aber die Eingabe der Doppelpunkte, die direkt hinter den Zeilennummern angegeben sind. Diese werden vom Computer nämlich automatisch eingefüet, sobald Sie die ENTER Taste betätigen und damit dem PC-1600 zu verstehen geben, daß er die Programmzeile in seinen Speicher aufnehmen soll. Der Doppelpunkt ist also eine Rückmeldung bzw. eine Bestätigung dafür, daß die Programmierung der Zeile erfolgt iat. Alle anderen im Programm vorkommenden Doppelpunkte sind jedoch wie gezeigt einzutippen.

```
10:PRINT "ZAHL :";
20:FOR X=-2 TO 2:PRINT X;" ";:NEXT X
30:PRINT
40:PRINT "BETRAG:";
50:FOR X=-2 TO 2:PRINT ABS(X);" ";:NEXT X
60:PRINT
70:END
```

Schalten Sie nun den Computer in den RUN-Modus und starten Sie das Programm mit dem Befehl RUN. (Vergessen Sie nicht die Betätigung der Taste ENTER.) Auf dem Display erscheint dann:

```
RUN
ZAHL :-2 -1 0 1 2
BETRAG: 2 1 0 1 2
>
```

ACS

WIRKUNG : Der Befehl ACS(X) liefert einen zum Argument X gehörenden Wert der Arcus-Cosinus-Funktion.

HINWEISE : Da die Arcus-Cosinus-Funktion die Umkehrung der Cosinus-Funktion ist, stellt der gelieferte Wert folglich einen Winkel dar. In Abhängigkeit von dem derzeit gültigen Winkelmodus (DEGREE, GRAD, oder RADIAN) ist das Ergebnis somit entweder in Altgrad, Neugrad oder im Bogenmaß zu werten. Das zulässige Funktionsargument X ist auf den Wertebereich -1 s X s 1 beschränkt. Der gelieferte Funktionswert liegt stets in folgenden Hauptwertebereichen:

```
DEG-Modus
RAD-Modus
GRAD-Modus

0•.... 180•
O .... π
0 .... 200 gon
```

BEISPIEL :

```
1O:DEGREE
20:PRINT "arccos(0.5)=";ACS(.5);" Grad"
30:PRINT "arcoos(O) =";ACS(O);" Grad"
40:END

>
RUN
arccos(O.5) = 60 Grad
arccos(O)   = 90 Grad
>
```

ADIN ON / OFF / STOP

WIRKUNG : Der Befehl ADIN erlaubt oder verhindert die Annahme von analogen Interrupt-Anforderungen.

HINWEISE : Eine analoge Interrupt-Anforderung liegt vor, wenn das am Analog-Eingang anstehende Signal innerhalb eines vereinbarten Pegelbereiches liegt (siehe hierzu: ON ADIN GOSUB).

ADIN IN erlaubt die Annahme eines analogen Interrupts. Mit der Anweisung ON ADIN GOSUB kann im Interrupt-Fall dann in eine Interrupt-Behandlungsroutine

verzweigt werden.

ADIN OFF verhindert die Annahme von analogen Interrupts.

ADIN STOP schaltet ebenfalls die Annahme analoger Interrupts aus, registriert jedoch die jeweils letzte Anforderung in einem Speicher. Sobald mit der Anweisung ADIN ON dann die Annahme von analogen Interrupts freigegeben wird, erfolgt die unmittelbare Verzweigung in die Interrupt-Routine. STOP ist der standardmäßig vom Sytem angenommene ADIN-Parameter.

AIN

Siehe auch : ADIN ON/OFF/STOP, ON ADIN GOSUB

WIRKUNG : AIN liefert einen Wert, der dem Spannungspegel am analogen Eingang entspricht.

HINWEIS : Der in der speziellen Variablen AIN enthaltene Integer-Wert kann zwischen 0 und 255 liegen.

Dieser Werte-Bereich deckt dabei alle Spannungen von 0 bis 2.495 V ab. Höhere Spannungen liefern den Integer-Wert 255.

BEISPIEL :
```
PRINT AIN
43
>
```

ALARM$

WIRKUNG : Mit ALARM$ läßt sich eine Alarmzeit und eine Alarmmeldung vereinbaren.

HINWEISE : **ALARM$="MM/DD/HH/mm"**

.... setzt die Alarmzeit, wobei die aufgeführten Platzhalter für die tatsächlichen Werte folgende Bedeutung haben:

Platzhalter	Bedeutung	zulässige Werte
MM	Monat (month)	01....12
DD	Tag (day)	01....31
HH	Stunde (hour)	00....23
mm	Minute (minute)	00....59

Anstelle aktueller Werte können auch "wildcards" in Form von Fragezeichen für den Monat und den Tag verwendet werden. So setzt beispielsweise die Angabe "??/??/13/30" die Alarmzeit auf 13:30 des heutigen Tages.

Wird die mit dieser Anweisung vereinbarte Uhrzeit erreicht, gibt der Computer für die Dauer einer Sekunde wiederholte Piepstöne von sich.

Zusätzlich kann mit der Anweisung ALARM$ eine <Meldung> vereinbart werden, die bei erreichter Alarmzeit zusammen mit dieser Zeitangabe auf dem dem Display angezeigt wird. Diese <Meldung> darf aus maximal 26 Zeichen bestehen.

Die <Meldung> überschreibt die Funktionstastenbelegung der Ebene II.

ALARM$=""

....löscht sowohl die gesetzte Alarmzeit als auch eine eventuell vereinbarte Alarmmeldung.

Das Befehlswort ALARM$ kann darüberhinaus als String-Variable verwendet werden, um z.B. die Vereinbarungen von Alarmzeit und Alarmmeldung sichtbar zu machen.

BEISPIEL : Angenommen heute sei der 12. Oktober und Sie hätten am nächsten Tag um 14:30h einen wichtigen Termin. Mit der Anweisung

```
ALARM$="10/13/13/45;TERMIN UM 14H30"
```

können Sie sich dann vom PC-1600 rechtzeitig an diesen erinnern lassen. Diese Vorsichtsmaßnahme nützt Ihnen allerdings nur, wenn der Computer zum Alarmzeitpunkt eingeschaltet ist. Im ausgeschalteten Zustand ertönt nur ein Piepssignal, es erscheint aber nicht die Meldung auf dem Display, da hierzu der Computer eingeschaltet sein müßte. Falls sich das Gerät automatisch zu einer bestimmten Zeit einschalten soll, müssen Sie den WAKE$-Befehl verwenden.

AREAD

Siehe auch : RUN

WIRKUNG : Mit AREAD kann ein in der Anzeige befindlicher String oder numerischer Wert mit dem Start des Programmes in eine bereitgestellte Variable eingelesen werden.

HINWEISE : Diese Art der Datenübergabe an ein Programm geht nur unter folgenden Voraussetzungen:

a) Das Befehlswort AREAD muß in der ersten Programm-Zeile stehen.

b) Es hat ihm unmittelbar eine Markierung vorauszugehen. Diese darf nur aus einem der nachstehend genannten Buchstaben bestehen:

A,S,D,F,G,H,J,K,L,Z,X,C,V,B,N,M

c) Das Programm muß über die DEF-Taste in Verbindung mit der gewählten Marke in Betrieb genommen werden.

d) Der in der Anzeige befindliche Wert und die mit AREAD bereitgestellte Variable müssen typenmäßig übereinstimmen.

Weist die Anzeige einen numerischen Wert auf, so wird dieser mit bis zu 10 Mantissen- und bis zu 2 Exponenten-Ziffern eingelesen, wobei sowohl die Mantisse als auch der Exponent vorzeichenbehaftet sein darf.

Bei Strings hängt die Anzahl der eingelesenen Zeichen von der über den DIM-Befehl definierten Stringlänge der Variablen ab. Eine nicht DIMensionierte Variable kann standardmäßig bis zu 16 Zeichen aufnehmen.

Steht bei Ausführung des AREAD-Befehles nur das Bereitschaftszeichen > in der Anzeige, so wird einer numerischen Variable der Wert 0 und einer Stringvariable ein Nullstring (ASCII-Code &00) zugewiesen.

BEISPIEL :

```
10:"F":AREAD K$
20:PRINT "** ";K$
```

Start bei leerem Display:

```
CLS       Display löschen
DEF F     Programm starten

Anzeige:      **
              >
```

Start bei beschriebenem Display:

```
HALLO         Text eingeben
DEF F         Programm starten

Anzeige:      ** HALLO
              >
```

ARUN

WIRKUNG : Ein im Arbeitsspeicher befindliches und mit dem Befehl ARUN versehenes Programm wird beim Einschalten des Computers automatisch gestartet.

HINWEISE : Dazu müssen allerdings folgende Bedingungen erfüllt sein:

a) ARUN muß sich in der ersten Programmzeile befinden, und zwar unmittelbar hinter der Zeilennummer. Es darf diesem Befehl also weder ein anderer Befehl noch eine Marke vorausgehen.

b) Der Computer muß im RUN-Modus ausgeschaltet worden sein, damit er sich beim Einschalten ebenfalls in diesem Modus befindet, da nur in dieser Betriebsart Programme ablauffähig sind.

Werden bei ausgeschaltetem Computer irgendwelche Optionen (z.B. Drucker, RAM-Module, Cassetten-Recorder, Diskettenlaufwerk usw.) angeschlossen oder gewechselt, so kann es bei der Einschaltung des Computers unter Umständen zur Ausgabe eines ERROR-Codes kommen. In diesem Falle unterbleibt natürlich ein automatischer Programm-Start und das Programm muß über RUN manuell in Betrieb genommen werden.

Ein mit ARUN gestartetes Programm löscht keine Variablen. Sollte eine derartige Löschung jedoch erwünscht sein, so ist im Programm der Befehl CLEAR an geeigneter Stelle einzufügen.

BEISPIEL :

```
10:ARUN:CLS
20:PRINT"GUTEN TAG !"
30:PRINT"ES IST JETZT ";TIME$;" UHR."
40:PRINT"ES SIND ";MEM;" BYTES FREI."
50:END
```

ASC

Siehe auch : CHR$

WIRKUNG : Die Funktion ASC(X$) liefert den ASCII-Code des Argumentes X$, welches ein String aus einem oder mehreren Zeichen sein kann.

HINWEISE : Wird als Funktionsargument ein String genommen, der aus mehr als einem Zeichen besteht, so wird der ASCII-Code seines ersten Zeichens geliefert.

Der Zusammenhang zwischen dem gelieferten Code und dem zugehörigen Zeichen ist aus dem Anhang C ersichtlich.

BEISPIEL :

```
10:WAIT 0:CLS
20:PRINT "BITTE EIN ZEICHEN ODER"
30:INPUT "EINEN STRING EINGEBEN:",S$
40:WAIT 100
50:PRINT "DER ASCII-CODE LAUTET:";ASC(S$)
60:END

RUN
BITTE EIN ZEICHEN ODER
EINEN STRING EINGEBEN:
SHARP
DER ASCII-CODE LAUTET: 83
>
RUN
BITTE EIN ZEICHEN ODER
EINEN STRING EINCEBEN:
*
DER ASCII-CODE LAUTET: 42
```

Siehe auch : ACS, ATN, SIN

WIRKUNG : Der Befehl ACS(X) liefert einen zum Argument X gehörenden Wert der Arcus-Sinus-Funktion.

HINWEISE : Da die Arcus-Sinus-Funktion die Umkehrung der Sinus-Funktion ist, stellt der gelieferte Wert folglich einen Winkel dar. In Abhängigkeit von dem derzeit gültigen Winkelmodus (DEGREE, GRAD, oder RADIAN) ist das Ergebnis somit entweder in Altgrad, Neugrad oder im Bogenmaß zu werten. Das zulässige Funktionsargument X ist auf den Wertebereich -1 _ X _ 1 beschränkt. Der gelieferte Funktionswert liegt stets im Hauptwertebereich. Hierbei gilt:

DEG-Modus : -90°.... 90°
RAD-Modus : -π/2 .... π/2
GRAD-Modus: (-100 .... 100) gon

BEISPIEL :

```
10:DEGREE
20:GOSUB 100
30:FOR DX=-10 TO 10
40:X=DX/10
50:F=ASN(X):Z=Z+1
60:IF Z=3 THEN GOSUB 100
70:PAUSE "   ";STR$(X),F
80:NEXT DX
90:END
100:CLS:PRINT"ARGUMENT","ARCUS-SINUS"
110:Z=0:RETURN
```

ATN

Siehe auch: ACS, ASN, TAN

WIRKUNG : Der Befehl ATN(X) liefert einen zum Argument X gehörenden Wert der Arcus-Tangens-Funktion.

HINWEISE : Da die Arcus-Tangens-Funktion die Umkehrung der Tangens-Funktion ist, stellt der gelieferte Wert folglich einen Winkel dar. In Abhängigkeit von dem derzeit gültigen Winkelmodus (DEGREE, GRAD, oder RADIAN) ist das Ergebnis somit entweder in Altgrad, Neugrad oder im Bogenmaß zu verstehen. Das Funktionsargument X unterliegt keiner wertmäßigen Beschränkung. Der Funktionswert wird je nach Winkelmodus innerhalb folgender Hauptwertebereiche geliefert:

DEG-Modus : -90°.... 90°
RAD-Modus : -π/2 .... π/2
GRAD-Modus: (-100 .... 100) gon

BEISPIEL :

```
10:DEGREE
20:GOSUB 100
30:FOR DX=0 TO 100
40:X=DX*.1
50:F=ATN(X):Z=Z+1
60:IF Z=3 THEN GOSUB 100
70:PAUSE " ";STR$(X),F
80:NEXT DX
90:END
100:CLS:PRINT "ARGUMENT","ARCUS-TANGENS"
110:Z=0:RETURN
```

AUTO

Siehe auch : RENUM

WIRKUNG : Mit dem Kommando AUTO kann zur Erleichterung des Programmierens im PRO-Modus eine automatische Zeilennumerierung vorgenommen werden.

HINWEISE : Nach Aktivierung von AUTO erscheint die erste generierte Zeilennummer in der Anzeige mit einem nachgestellten Cursor, Nun kann der gewünschte Zeileninhalt eingegeben werden. Schließt man die Eingabe dann durch Betätigung der ENTER Taste ab, so wird in der folgenden Zeile die nächste Zeilennummer generiert und so fort.

Ergibt sich bei der Generation der Zeilennummern die Nummer einer bereits existierenden Zeile, so wird diese Zeile angezeigt,

Die erzeugten Zeilennummern hängen davon ab, ob AUTO mit oder ohne Parameter versehen wird und welche Werte für diese gewählt werden:

AUTO

Wird das Kommando AUTO ohne Parameter angegeben, so beginnt die Numerierung mit der Zeile 1O und setzt sich im Zehnerabstand, also mit den Zeilen 20, 30 usw. fort.

AUTO <Zeilennummer>

Ist dem Befehlswort AUTO eine einzelne Integerzahl beigefügt, gilt diese als die erste Zeilennummer. Alle weiteren Zeilennummern folgen im Zehnerabstand.

AUTO <Zeilennummer>,<Zeilenabstand>

Mit einem weiteren Parameter kann die Schrittweite der Zeilennumerierung bestimmt werden. Die beiden Parameter sind durch ein Komma voneinander zu trennen.

Die automatische Zeilennumerierung läßt sich mit Betätigung der BREAK- oder CL Taste aufheben. Es kann aber auch die ENTER Taste zu diesem Zweck verwendet werden, wenn man diese gleich nach dem Erscheinen einer neuen Zeilennummer betätigt, ohne zuvor etwas in die Zeile hineinzuschreiben.

BEISPIELE:

Anweisung	Generierte Zeilennummern
AUTO	10,20,30,40,.....
AUTO 100	100,110,120,.....
AUTO 400,20	400,420,440,.....

BEEP

Siehe auch : BEEP ON/OFF

WIRKUNG : BEEP erzeugt eine Anzahl von Tönen spezifizierte Tonhöhe und Dauer.

<Anzahl> bestimmt, wie oft der standardmäßige oder aber näher spezifizierte Ton vom Computer erzeugt werden soll. Die zulässigen Werte sind: 0 ... 65535.

<Tonhöhe> legt die Frequenz des Tones fest und darf durch einen Integer-Wert von 0 bis 255 vertreten sein. Es lassen sich Frequenzen von 230 Hz bis 7 kHz erzeugen, wobei Frequenz und Wert des Parameters in einem umgekehrt "proportionalen" Verhältnis stehen. Je höher der Wert, desto niedriger die Frequenz:

0 bedeutet ca. 7 kHz
255 bedeutet ca. 230 Hz

Fehlt dieser Parameter, wird als Standard eine Frequenz von 4 kHz geliefert.

<Dauer> bestimmt die Dauer eines Tones. Sie ist abhängig von der Frequenz, also dem Wert von <Tonhöhe>. Je tiefer die Frequenz, umso länger die Dauer. Der Wert darf im Bereich 0...65279 liegen und wird bei fehlendem Parameter zu 160 angenommen.

BEEP ON/OFF

Siehe auch : BEEP

WIRKUNG : BEEP ON/OFF schaltet die Wirksamkeit des BEEP befehles ein bzw. aus.

HINWEISE : Die Kommandos BEEP ON/OFF haben ebenfalls einen Einfluß auf die Wiedergabe des Kontroll-Tones, der beim Einladen eines Programmes von einer Cassette geliefert wird.

BEEP ON erlaubt das Wirksamwerden einer BEEPAnweisung. Beim Laden von Programmen bzw. Daten von Cassette werden diese über den eingebauten Lautsprecher hörbar gemacht, um so den Ladevorgang kontrollieren zu können.

BEEP OFF schaltet den Lautsprecher ab. Es wird dann weder eine BEEP-Anweisung wirksam noch ein Lade-Kontrollton hörbar.

BEISPIEL :

```
10:BEEP 4
20:BEEP OFF
30:BEEP 4
40:BEEP ON
50:END
```

Zeile 10 erzeugt vier gleichfrequente Töne
Zeile 20 schaltet den Lautsprecher ab
Zeile 30 erzeugt wegen Zeile 20 keinen Ton
Zeile 40 schaltet den Lautsprecher wieder ein
Zeile 50 beendet das Programm

BLOAD

Siehe auch : BSAVE, CLOAD, NEW, SET

WIRKUNG : BLOAD lädt ein Maschinensprache-Programm von einer Diskette oder einem RAM-Disk-Modul.

HINWEISE : Der Parameter **<Dateibezeichner>** bestimmt die Option von der das Programm zu holen und unter welchem Namen und Gruppenkennung (extension) es dort abgelegt ist. Der <Dateibezeichner> hat das Format:

<Datenquelle:Dateiname.Extension>

Folgende Datenquellen können adressiert werden:

S1: oder S2: RAM-Modul des Faches S1 oder S2
X: oder Y: Diskette
COM1: RS-232C-Schnittstelle
COM2: Optoelektronische Schnittstelle

<Bank> spezifiziert die Speicherbank, in die das Maschinensprache-Programm geladen werden soll. Es sind die Banknummern 0 bis 7 möglich.

<Adresse> bestimmt die Adresse innerhalb der gültigen Speicherbank, ab der die Ablage des Programmes erfolgen soll. Der zulässige Wertebereich lautet: 0 ... 65535.

Werden die beiden optionalen Parameter <Bank> und <Adresse> nicht angegeben, so legt sich das Programm in genau die Speicherbank ab derjenigen Anfangsadresse ab, wo es sich vor der Sicherung zuvor im Arbeitsspeicher befunden hat.

Wurde das Programm mit dem BSAVE-Kommando unter Angabe einer Auto-Startadresse gesichert, so wird es genau ab dieser Adresse in den Speicher geladen und nach Beendigung des Ladevorganges automatisch ausgeführt.

Ist das auf Diskette befindliche Programm mit der I-Option des SET-Befehles geschützt, bleibt diese Option unbeachtet.

Schauen Sie bitte in den Anhang D, um weitere Informationen über die Speicherbänke und Adreß-bereiche zu erhalten.

BEISPIEL : BLOAD "X:RXOUT"

BREAK ON/OFF

Siehe auch : CONT

WIRKUNG : Erlaubt oder verbietet das Wirksamwerden der Betätigung der BREAK-Taste.

HINWEISE : BREAK OFF schaltet die Funktion der BREAK-Taste ab. Ein laufendes Programm kann dann durch Betätigung dieser Taste nicht unterbrochen werden.

BREAK ON schaltet die Funktion der BREAK-Taste ein. Eine Betätigung der BREAK-Taste bricht dann ein laufendes Programm ab und gibt dabei folgende Meldung aus:

BREAK IN <Zeilennr.>

Mit dem CONT-Kommando kann der Ablauf des Programmes fortgesetzt werden.

Eine gute Idee für den Einsatz der Anweisungen BREAK ON und BREAK OFF ist, diese unmittelbar vor und hinter einem Programmteil einzubinden, das auf keinen Fall vom Anwender unterbrochen werden darf, weil z.B. gerade wichtige Daten übertragen oder eine Grafik gedruckt wird.

Falls durch die Anwendung dieser Anweisungen ein Programm nicht aus einer Endlosschleife befreit werden kann, bleibt nur der Weg, die RESET-Taste zu bedienen. Die Anweisung BREAK OFF sollte also nur dann eingesetzt werden, wenn ein Programm einen fehlerfreien Ablauf gewährleistet und das Auftreten solcher Endlosschleifen ausgeschlossen werden kann.

BSAVE

Siehe auch : BLOAD, CSAVE M

WIRKUNG : Sichert ein Maschinensprache-Programm auf einer Diskette oder einem RAM-Disk-Modul.

HINWEISE : Der Parameter **<Dateibezeichner>** bestimmt, auf welchem Medium das Programm zu sichern und unter welchem Namen und welcher Extension es dort abzulegen ist. Er hat das Format:

<Datenquelle:Dateiname.Extension>

Im Gegensatz zum SAVE-Kommando muß bei diesem <Dateibezeichner> die Extension unbedingt angegeben werden.

Folgende Datenquellen sind adressierbar:

S1: oder S2: RAM-Modul des Faches S1 oder S2
X: oder Y: Diskette
COM1: RS-232C-Interface
COM2: Optoelektronisches Interface

<Bank> bestimmt die Speicherbank, auf der das Maschinensprache-Programm vorzufinden ist. Es sind die Banknummern 0 bis 7 möglich.

<Start> bestimmt die Adresse innerhalb der gültigen Speicherbank, ab der das Programm beginnt. Für Adresse sind die Werte 0 ... 65535 zulässig.

<Ende> bestimmt die höherwertige Adresse innerhalb der gültigen Speicherbank, mit der das Programm endet.

<Auto> bestimmt, ab welcher Adresse das Programm automatisch gestartet werden soll, wenn man es mit BLOAD wieder an seinen Platz zurücklädt. Fehlt dieser optionale Parameter, so wird er auf den Standard-Wert &FFFF gesetzt und damit ein Auto-Start verhindert.

Über die Aufteilung des Speichers in Bänke und deren Adreßbereiche können Sie sich anhand des Anhanges D informieren.

BEISPIEL : BSAVE "S1:SORT",#1,&8000,&8AFF

Diese Anweisung sichert ein auf Speicherbank 1 befindliches Maschinen-Programm von der Adresse &8000 beginnend bis einschließlich der Adresse &8AFF auf dem im Modulfach S1 befindlichen RAM-Modul unter dem Namen SORT.

CALL

Siehe auch : NEW, POKE, XPOKE

WIRKUNG : Mit CALL kann von einem BASIC-Programm oder der RUN-Ebene aus, ein Maschinensprache-Programm gestartet und anschließend in die aufrufende Ebene zurückgekehrt werden. Dabei ist die Übergabe eines einzelnen Variablen-Wertes möglich.

HINWEISE : **<Bank>** bestimmt die Speicherbank aus dem Bereich 0...7, in der das Maschinenspracheprogramm gespeichert ist. Wird dieser Parameter nicht angegeben, gilt Speicherbank 0.

<Adresse> nennt die Anfangsadresse innerhalb der gültigen Speicherbank, bei der das Programm beginnt. Es sind die Adressen von 0...65535 (&0...&FFFF) zulässig. Die Adreßangabe muß erfolgen und darf nicht weggelassen werden.

<Varibale> bestimmt die Variable, deren Wert an das Maschinensprache-Programm übergeben wird, und in die nach Beendigung des Programmes der aktualisierte Wert abzugeben ist. In diese Variable wird jedoch nur dann ein Wert zurückgegeben, wenn bei Rückkehr auf die BASIC Ebene das Carry-Flag gesetzt ist. Bei Variablen numerischen Typs wird der Wert an das Registerpaar DE abgegeben. Da er vorzeichenbehaftet sein kann, muß er folglich im Bereich von -32768 bis 32767 liegen. Bei Stringvariablen nimmt dieses Registerpaar die Adresse auf, ab der der String im Speicher abgelegt ist. Register B führt den Wert der Stringlänge.

BEISPIEL : s.POKE

CHAIN

Siehe auch : CSAVE, MERGE

WIRKUNG : CHAIN lädt von einem BASIC-Programm aus ein auf Cassette befindliches anderes BASIC-Programm und startet es.

HINWEISE : **CHAIN**

lädt das erste auf Cassette auffindbare Programm in den Speicher und beginnt dessen Abarbeitung mit der ersten Zeile.

CHAIN <Zeilen-Nr.>

verhält sich wie CHAIN ohne Parameter, startet das Programm jedoch mit der angegebenen Zeile.

CHAIN <Dateiname>

sucht auf der Cassette nach dem Programm, das den spezifizierten Namen trägt, lädt dieses und startet es mit seiner ersten Zeile.

CHAIN <Dateiname>,<Zeilen-Nr.>

lädt das Programm angegebenen Namens und nimmt es ab der spezifizierten Zeile in Betrieb.

Die Anwendung des CHAIN-Befehles erlaubt, lange Programme, die nicht in den Speicher passen, als einzelne Teilprogramme zerlegt, Stück für Stück nacheinander automatisch zu laden und ablaufen zu lassen.

Sollte eines der zu ladenden Programme mittels PASS-Befehl geschützt sein, so generiert CHAIN die Ausgabe eines ERROR-Codes und stoppt damit den Programmablauf.

BEISPIEL :

```
>
RUN
```

PROG1

```
10:REM PROGRAMM 1
20:
30:
:
:
400: CHAIN "PROG2"
```

PROG2

```
10:REM PROGRAMM 2
20:
30:
:
:
200:END
```

Mit dem RUN-Befehl wird das gerade im Speicher befindliche Programm PROG1 gestartet. Erreicht dieses Programm seine letzte Zeile, so sucht der Computer wegen des CHAIN-Befehles auf der Cassette nach dem Programm PROG2 und lädt es bei Vorhandensein in den Speicher, wobei das Programm PROG1 dabei überschrieben wird. Ist der Ladevorgang abgeschlossen, wird das neu in den Speicher geholte Programm mit seiner ersten Zeile gestartet, da der CHAIN-Befehl ohne Parameter <Zeilen-Nr.> in PROG1 angegeben ist.

CHR$

Siehe auch : ASC

WIRKUNG : Die Funktion CHR$ liefert das Zeichen, dessen zugehöriger ASCII-Code als Funktionsargument angegeben ist.

HINWEISE : Das betreffende Argument kann entweder eine Konstante, eine Variable oder ein Ausdruck sein. Der numerische Wert dieser Größen muß aber in jedem Falle vom Typ Integer sein.

Mit dieser Funktion können Steuerzeichen, die nicht über die Tastatur zugänglich sind, an die verschiedensten Peripheriegeräte wie Drucker usw. oder aber an die seriellen Schnittstellen weitergeleitet werden.

In welcher Zuordnung die Zeichen zu den ASCII-Codes stehen, können Sie aus Anhang C ersehen.

BEISPIEL :

```
10:FOR X= 33 TO 126
20:PAUSE CHR$(X);
30 NEXT X
40:END
```

Dieses Programm zeigt alle darstellbaren Zeichen des standardmäßigen ASCII-Bereiches (Codes: &21 bis &7E) auf dem Display an. Das erste Zeichen ist hierbei das Ausrufezeichen !, das letzte die sogenannte Tilde ~.

CLEAR

Siehe auch : DIM, ERASE, TITLE

WIRKUNG : CLEAR löscht sämtliche im Speicher befindlichen Variablen. Dies gilt auch für die reservierten, sprich Standard-Variablen.

HINWEISE : Die numerischen Standardvariablen A bis Z bzw. @(1) bis @(26) werden dabei mit dem Wert 0 belegt und den Stringvariablen A$ bis Z$ bzw. @$(1) bis @$(26) ein Nullstring (ASCII-Code 0) zugewiesen.

Man kann das CLEAR-Kommando auch innerhalb eines Programmes verwenden. In jedem Falle läßt sich mit ihm Speicherplatz wiedergewinnen, der durch die Erzeugung und Belegung von Variablen für das eigentliche Programm verlorengegangen ist. Zum Beispiel mögen im ersten Teil eines Programmes soviele Variablen verwendet worden sein, daß kein freier Speicherplatz mehr übrig bleibt. Braucht man diese Variablen im zweiten Programmteil nicht mehr, so kann mit CLEAR wieder Platz für neue Variablen geschaffen werden.

BEISPIEL :

```
5:WAIT 30              'Setzt Wartezeit für PRINT
10:DIM C(5)            'Dimensioniert Array C(N)
20:FOR N= 1 TO 5       'Diese Zeilen lesen die
30:READ A:LET C(N)=A   'DATA-Werte ein und
40:PRINT C(N)          'zeigen sie an.
50:NEXT N
60:DATA  10,20,30,40,50'Stellt  die  Daten  bereit
70:CLEAR                'Löscht alle Variablen
80:PRINT A              'Beweise Löschung
90:END
```

Prüfen Sie nach Ablauf des Programmes, ob auch das Array gelöscht worden ist, indem Sie eine Zuweisung versuchen, z.B: C(2)=99. Existiert das Array nicht, erscheint die Meldung: ERROR 6.

CLOAD

Siehe auch : CLOAD?, CSAVE, MERGE

WIRKUNG : CLOAD lädt BASIC-Programme oder Belegungen für die Funktionstasten von Cassette, die mit CSAVE gesichert worden sind.

HINWEISE : Vor dem Ladevorgang löscht CLOAD grundsätzlich das im Speicher befindliche Programm.

Ohne Parameterangabe wird die nächste auf der Cassette auffindbare Datei geladen. Mit Angabe des Parameters <Dateiname> wird zuvor nach der betreffenden Datei auf der Cassette gesucht.

Programme können sowohl im PRO- als auch im RUN-Modus geladen werden. Tastaturbelegungen sind im RESERVE-Modus zu laden. Es ist darauf zu achten, daß der Modus zu der zu ladenden Datei paßt. Ein im RESERVE-Modus geladenes Programm zerstört die derzeitige Tastaturbelegung. Umgekehrt zerstört eine im RUN- oder PRO-Modus geladene Tastaturbelegung das gerade im Speicher stehende Programm.

Ist der CLOAD-Befehl mit der Option R versehen, wird ein geladenes BASIC-Programm automatisch gestartet. Liegt kein BASIC-Programm dabei vor, wird ein ERROR-Code angezeigt.

Eine Fehlermeldung erscheint auch dann, wenn das Programm mit einem Kennwort durch Anwendung des PASS-Befehles geschützt sein sollte.

BEISPIELE : CLOAD
lädt das nächste auffindbare Programm.

CLOAD "P1"
sucht auf der Cassette das Programm P1 und lädt es bei Vorhandensein.

CLOAD?

Siehe auch : CLOAD, CSAVE, MERGE

WIRKUNG : Dient dem Vergleich zwischen einer im Speicher und einer auf Cassette befindlichen Datei auf deren Übereinstimmung.

HINWEIS : Diese Verifikation kann sich dabei auf ein im PROoder RUN-Modus benutztes Programm beziehen oder auf eine im RESERVE-Modus genutzte Belegung der Funktionstasten.

Wird CLOAD? ohne Parameterangabe angewendet, muß der Anwender für die richtige Positionierung des Cassetten-Bandes sorgen, weil der Vergleich mit der unmittelbar als nächstes auf der Cassette auffindbaren Datei stattfindet.

Mit Angabe des Parameters **<Dateiname>** stellt man sicher, daß zum Vergleich nur die genannte Datei von der Cassette herangezogen wird.

Der Vergleich erfolgt blockweise. Sofern sich eine Abweichung zwischen zwei Blöcken ergibt, wird der Ablauf gestoppt und ein ERROR-Code angezeigt. Fällt der Vergleich positiv aus, d.h. stimmen beide Dateien überein, erscheint auf dem Display wieder das Bereitschaftszeichen >.

BEISPIEL :

```
>
CLOAD "PROG01"          Programm laden
Cassette zurückspulen
>
CLOAD?"PROG01"          Programm überprüfen
```

CLOAD M

Siehe auch : BLOAD, CALL, CSAVE M, NEW

WIRKUNG : Lädt Maschinensprache-Programme von der Cassette in den Arbeitsspeicher.

HINWEISE : Maschinensprache-Programme unterscheiden sich zu BASIC-Programmen in ihrem Aufzeichnungsformat. Deshalb können sie nicht mit dem Befehl CLOAD geladen werden. Auch werden sie in einem anderen Speicherbereich abgelegt als BASIC-Programme.

<Bank> bestimmt die Speicherbank (0...7), in die das Maschinensprache-Programm zu laden ist.

<Adresse> bestimmt dabei, ab welcher Adresse innerhalb der gültigen Speicherbank das Programm abgelegt werden soll.

Diese beiden Parameter können auch weggelassen werden. Wird jedoch einer von beiden angegeben, muß auch der jeweils andere spezifiziert sein. Fehlt dieses Parameterpaar, so wird das Programm in jene Speicherbank ab jener Adresse geladen, wie es sich vor seiner Aufzeichnung im Speicher befunden hat.

Wurde das Programm bei seiner Aufzeichnung mit einer Autostart-Adresse versehen, wird es an eben dieser Adresse nach Abschluß des Ladevorganges automatisch gestartet.

BEISPIEL : CLOAD M "MAC1"

Diese Anweisung lädt von der Cassette das mit MAC1 bezeichnete Maschinensprache-Programm an die Adresse und in die Speicherbank, wo es sich bei der Aufzeichnung zuvor befunden hat.

CLOSE

Siehe auch : END, OPEN

WIRKUNG : CLOSE schließt alle spezifizierten Dateien.

HINWEISE : Mit CLOSE wird die Möglichkeit des Zugriffs auf Dateien beendet, d.h. diese geschlossen.

Ohne Parameterangabe schließt CLOSE alle offenen Dateien. Mit Angabe der Parameter <Datei-Nr.> werden nur die Daten geschlossen, die unter der jeweils gleichen Nummer zuvor mit OPEN zu einem bestimmten Zweck geöffnet worden sind.

Eine einmal geöffnete Datei muß, bevor sie für einen anderen Zugriffszweck (Eingabe, Ausgabe, Datenanhang) geöffnet werden kann, zuvor mit CLOSE geschlossen werden.

Versucht man, eine bereits geöffnete Datei zu öffnen, wird ein ERROR-Code ausgegeben.

Alle geöffneten Dateien werden automatisch bei Ausführung der Befehle END, NEW, RUN und LOAD und bei Ausschaltung des Computers geschlossen. Eine Dateischließung erfolgt auch dann, wenn man ein Programm editiert.

BEISPIEL :

```
5:MAXFILES=2
10:OPEN "X:PAYMENT" FOR INPUT AS #1
20:OPEN "CAS:UPDATE" FOR INPUT AS #2
:
:
400:CLOSE #1,#2
```

CLS

WIRKUNG : CLS löscht das Display.

HINWEISE : Das Kommando CLS löscht den Inhalt sämtlicher Display-Zeilen und setzt den Cursor an die linke obere Display-Ecke. Diese "Home-Position" trägt den im MODE 0 Koordinatenpunkt (0,0).

COLOR

WIRKUNG : COLOR selektiert den gewünschten Farbstift des Druckers CE-1600P (bzw. CE-150).

HINWEISE : Der Stift wird über den Parameter <Farb-Code> bestimmt, wobei folgende Farben möglich sind:

<Farb-Code>	Farbe
0	schwarz
1	blau
2	grün
3	rot

Mit Einschaltung des Computers wird automatisch der schwarze Stift ausgewählt.

COM$

Siehe auch : SETCOM

WIRKUNG : COM$ liefert einen String, der die zuletzt über SETCOM vereinbarten Kommunikations-Parameter des spezifizierten Ports enthält.

HINWEISE : Die in diesem String enthaltenen Parameter sind in genau der Reihenfolge angeordnet, wie sie mit SETCOM festzulegen sind. Sie lautet:

, <WL>, <PR>, <ST>, <XO>, <SI>

COM$ "COM1:" liefert die Einstellungen der RS-232C-Schnittstelle,
COM$ "COM2:" dagegen die des SIO-Interface (optoelektronischer Port).
COM$ "COM:" meint dasjenige Interface, das zuletzt über SETDEV selektiert worden ist.

BEISPIEL :

```
10:SETCOM "COM1:",300,,,2
20:PRINT COM$ "COM1:"

RUN
300,8,N,2,X,S
>
```

COMn ON/OFF/STOP

Siehe auch : ON COMn GOSUB, SETCOM

WIRKUNG : Diese Anweisungen erlauben oder verhindern die Annahme von Interrupt-Anforderungen, die über eines der beiden Kommunikations-Ports an den Computer gestellt werden.

HINWEISE : COMn meint, stellvertretend für COM1 und COM2 geschrieben, jeweils folgendes Interface:

COMl = RS-232C-Interface

COM2 = SIO-Interface

COMn ON erlaubt die Annahme eines ankommenden Interrupts, der über die Anweisung ON COMn GOSUB mit der entsprechend dafür vorgesehen Unterroutine bearbeitet werden kann.

COMn OFF verbietet die Annahme des Interrupts. Die Anweisung ON COMn GOSUB wird dann nicht beachtet.

COMn STOP verbietet ebenfalls die Annahme dieser Interrupts, merkt sich den jeweils zuletzt angeforderten in einem Zwischenspeicher. Bei der nächsten Ausführung von COMn ON wird dieser Interrupt dann unverzüglich abgearbeitet. COMn STOP ist die vom System standardmäßig angenommene Einstellung.

CONT

Siehe auch : RESUME, RUN, STOP, WAIT

WIRKUNG : Das Kommando CONT setzt abgebrochene oder unterbrochene Programmabläufe fort.

HINWEISE : Eine solche Fortsetzung ist nur bei folgenden Abbruchoder Unterbrechungsursa¢hen möglich:

- Abbruch durch STOP-Anweisung
- Abbruch durch Betätigung der Taste BREAK
- Unterbrechung durch PRINT-Anweisung

In diesen Fällen wird CONT jedoch ignoriert:

- Programm abgearbeitet oder durch END beendet
- Programm im PRO-Modus geändert
- Programm mit ERROR-Code abgebrochen

Anstelle des CONT-Kommandos läßt sich auch eine GOTO-Anweisung mit spezifizierter Zeilennummer verwenden. Noch einfacher geht es aber, wenn man schlicht und ergreifend die Taste ⇓ betätigt.

BEISPIEL :

```
10:PRINT "PROGRAMM STOPPT HIER"
20:STOP
30:PRINT "PROGRAMM FORTGESETZT"
40:PRINT "PROGRAMM BEENDET
50:END

RUN
PROGRAMM STOPPT HIER
BREAK IN 20
>
CONT
PROGRAMM FORTGESETZT
PROGRAMM BEENDET
>
```

COPY

Siehe auch : SET

WIRKUNG : COPY kopiert eine Datei.

HINWEISE : Es sind drei verschiedene Kopiervorgänge möglich

a) Kopie von einem zum anderen Medium unter Beibehaltung der Dateibezeichnung.

b) Kopie von einem zum anderen Medium unter geänderter Dateibezeichnung.

c) Kopie auf demselben Medium unter einer anderen Dateibezeichnung.

Die in Anführungszeichen zu setzenden Parameter **<Dateibezeichner 1>** und **<Dateibezeichner 2>** sind in folgende Angaben zu unterteilen:

<Dateibezeichner> = <Dn:FILENAMEn.EXTn>

Hierbei soll nach der Ersetzung von n gelten:

D1:	Datenquelle (Quellmedium)
D2:	Datenziel (Zielmedium)
FILENAME1	Name der Quelldatei
FILENAME2	Name der Zieldatei
EXT1	Extension der Quelldatei
EXT2	Extension der Zieldatei

Bei Anwendung von COPY muß jeder Dateibezeichner mit einer Extension versehen sein. Diese Angabe darf keinesfalls fehlen.

Die Mediumangabe D2 des zweiten Dateibezeichners ist optiona1 und kann also weggelassen werden. Fehlt sie, sind Ziel- und Quellmedium identisch. Es kann dann nur eine Kopie gemäß Fall c) vorgenommen werden, bei dem sich die Namen der Quell und Zieldatei voneinander unterscheiden müssen.

In einer COPY-Anweisung sind keine “wildcards” zulässig. Bezeichnungen können also nicht durch einen Stern (*) oder ein Fragezeichen (?) a1s mehrdeutige Benennungen ausgelegt werden.

Eine Datei kann nicht a1s eine Zieldatei kopiert werden, wenn bereits eine Datei gleichen Namens existiert.

Mit COPY sind keine Dateien von einem seriellen Interface auf Diskette, RAM-Disk oder Cassette kopierbar.

Der Befehl COPY kann auch dazu benutzt werden, um eine Datei von einer Diskette auf eine andere Diskette zu kopieren. A1s Mediumbezeichnung sind dann die logischen Namen X: und Y: zu verwenden.

BEISPIEL : COPY "S1:RICH" TO "S2:RICH"

Diese Anweisung kopiert die Datei RICH, die sich im Modul des Modulfaches S1 befindet, auf jenes RAM-Modul in Modulfach S2.

COS

Siehe auch : ACS, SIN, TAN

WIRKUNG : Die Funktion COS liefert den Cosinus-Wert eines angegebenen Winkelargumentes.

HINWEISE : Der durch einen numerischen Ausdruck vertretene Winkel kann entweder in Altgrad, Bogenmaß oder Neugrad vorliegen. Damit der Computer den dazugehörigen Funktionswert liefern kann, muß er im passenden Winkelmodus betrieben werden. Hierbei gilt:

Winkelmaß	Winkelmodus
Altgrad	DEGREE
Bogenmaß	RADIAN
Neugrad	GRAD

BEISPIEL :

```
10:DEGREE
20:G$=CHR$ (&F8)
30:PRINT "cos(60";G$;") = ";COS(60)
40:PRINT "cos(90";G$;") = ";COS(90)
50:END

>RUN
cos(60°) = 0.5
cos(90°) = 0
>
```

CSAVE

Siehe auch : CLOAD, CLOAD?, LLIST, MERGE

WIRKUNG : CSAVE sichert eine Tastaturbelegung, ein BASICProgramm oder einen Teil davon auf Cassette.

HINWEISE : Bei der Sicherung eines Programmes oder eines Programmteiles muß der RUN- oder der PRO-Modus eingestellt sein. Zur Sicherung einer Tastaturbelegung ist der RESERVE-Modus erforderlich.

CSAVE ohne Parameterangabe sichert die Daten ohne Mitaufzeichnung einer Benennung.

CSAVE "<Dateiname>" sichert die Daten dagegen unter Mitaufzeichnung des angegebenen Namens.

Bei Verwendung der Option A erfolgt die Datenaufzeichnung im ASCII-Format, im anderen Falle im komprimierteren Binär-Format.

Die erste **<Zeilen-Nr.>** nennt die Zeilennummer, mit der die Programmaufzeichnung beginnen soll. Die zweite <Zeilen-Nr.> legt die Programmzeile fest, die a1s letzte mit aufzuzeichnen ist.

Ist das aufzuzeiohnende Programm über PASS mit einem Kennwort (password) geschützt, kann keine Aufzeichnung mittels CSAVE erfolgen bevor dieses Kennwort nicht gelöscht worden ist.
Mit CSAVE gesicherte Dateien können nur mit dem Befehl CLOAD geladen werden.

BEISPIEL : CSAVE "PROG1";200,330

Diese Anweisung zeichnet von dem im Speicher befindlichen BASIC-Programm die Zeilen 200 bis einschließlich 330 unter der Bezeichnung PROG1 auf Cassette im binaren Format auf.

CSAVEM

Siehe auch : CLOAD M, CALL, BSAVE

WIRKUNG : CSAVE M sichert Maschinensprache-Programme auf Cassette.

HINWEISE : **<Dateiname>**

bestimmt, unter welchem Namen das Programm auf Cassette aufgezeichnet werden soll und unter welchem Namen man es dort wieder auffinden und in den Speicher zurückladen kann.

<Bank>

selektiert dabei jene Speicherbank von 0 bis 7, auf der das Programm abgelegt ist.

<Start>

bestimmt jene Adresse, ab der mit der Sicherung des Programmes begonnen und dieses zu höheren Adressen hin fortgesetzt werden soll, bis die Endadresse des Programmes erreicht ist.

<Ende>

spezifiziert die Adresse, in der das letzte Byte des Maschinensprache-Programmes vorzufinden ist, d.h. bis zu einschließlich welchem Byte die Aufzeichnung des Programmes vorzunehmen ist.

<Auto>

ist optional und bestimmt, mit welcher Adresse das Maschinensprache-Programm nach seiner Rückladung mit CLOAD M automatisch zu starten ist. Fehlt diese Angabe, wird sie standardmäßig zu &FFFF angenommen, was einer Ausschaltung der Autostart-Funktion gleichkommt.

CSAVEM

BEISPIEL : CSAVE M "MAC1";#4,&8000,&8AFF

.....zeichnet ein Maschinensprache-Programm, das sich in der Speicherbank 4 befindet und sich im Adreßbereich &8000 bis einschließlich &8AFF er streckt, auf einer Cassette unter dem Namen MAC1 auf. Wird es zu einem späteren Zeitpunkt in den Speicher zurückgeholt, legt es der Computer an genau der ursprünglichen Stelle ab (sofern in der Anweisung CLOAD M nichts anderes vereinbart wird). Da keine Autostart-Adresse <Auto> bei der Aufzeichnung angegeben worden ist, findet kein automatischer Start des Programmes statt.

CSIZE

Siehe auch : PCONSOLE

WIRKUNG : CSIZE bestimmt, in welcher Größe die Zeichen mit dem Drucker CE-1600P bzw. CE-150 abbildbar sind.

HINWEISE : Die **<Größe>** wird durch einen ganzzahligen Wert im Bereich von 1..9 wie folgt festgelegt:

<Größe>	Zeichen/Zeile	Höhe (mm)	Breite (mm)
1	160	1.2	0.8
2	80	2.4	1.6
3	53	3.6	2.4
4	40	4.8	3.2
5	32	6.0	4.0
6	26	7.2	4.8
7	22	8.4	5.6
8	20	9.6	6.4
9	17	10.8	7.2

Die hier gezeigten Zeichengrößen gelten bei dem Wert <Zeilenlänge>=0, mit dem PCONSOLE die Länge der Zeilen begrenzungslos schaltet.

BEISPIEL :

```
10:FOR I=1TO 4
20:CSIZE I
30:LPRINT "SHARP PC-1600"
40:LPRINT
50:NEXTI
60:CSIZE 2
70:END
```

CURSOR

Siehe auch : PRINT

WIRKUNG : CURSOR setzt den Cursor des Displays an die gewünschte Zeichenposition.

<Spalte> bestimmt dabei die gewünschte Spalte (X-Position) im Bereich von 0 bis 25.

<Spalte>=0 : linker Displayrand
<Spalte>=1 : 2. Spalte von links
<Spalte>=2 : 3. Spalte von links
.
.
<Spalte>=25 : rechter Displayrand

<Zeile> bestimmt dagegen die gewünschte Zeile (Y-Position) im Bereich von 0 bis 3.

<Zeile>=0 : oberste Zeile
<Zeile>=1 : zweite Zeile
<Zeile>=2 : dritte Zeile
<Zeile>=3 : unterste Zeile

Diese Parameter wird nur im MODE 0 vom PC-1600 angenommen.

BEISPIEL :

```
10:WAIT 50
20:FOR N=1 TO 6
30:READ A$
40:CURSOR 12,1:PRINT A$
50:CURSOR 12,1:PRINT " "
60:NEXT N
70:WAIT 0
80:END
90:DATA "H","A","L","L","O","!"
```

DATA

Siehe auch : READ, RESTORE

WIRKUNG : DATA dient zur Auflistung von numerischen oder String-Konstanten, die mit der READ-Anweisung gelesen werden können.

HINWEISE : Eine DATA-Anweisung kann gleichzeitig numerische und String-Konstante in gemischter Aufzählung enthalten. Die Konstanten müssen jeweils durch ein Komma voneinander getrennt sein.

Mit jeder neuen READ-Anweisung läßt sich immer eine Konstante nach der anderen aus dieser Liste ablesen. Damit das Zusammenspiel zwischen READ und DATA funktioniert, muß die gerade zu lesende Konstante vom selben Typ sein wie die in der READ-Anweisung angegebene Variable.

Welches Element der Liste gerade lesbar ist, wird durch einen internen Zeiger bestimmt, der automatisch nach jedem Lesevorgang entsprechend um eine Position weiter gesetzt wird.

Sind alle in der DATA-Liste enthaltenen Elemente gelesen, kann keine weitere READ-Anweisung ausgeführt werden, bevor nicht eine Rücksetzung des internen Zeigers mit Hilfe des RESTORE-Befehles erfolgt.

DATA gehört wie REM zu den sogenannten nichtausführbaren Anweisungen, was bedeuten soll, daß bei einem Sprung auf eine solche Anweisung der Computer nach der nächsten Anweisung sucht, die nicht mit dem Befehlswort DATA beginnt und dort den weiteren Programmablauf fortsetzt.

DATA-Anweisungen können an beliebiger Stelle im BASIC-Programm vorkommen. Sie brauchen nicht so plaziert zu werden, daß sie vor dem READ-Befehl, der ihre Inhalte liest, stehen. Der Computer sucht sich nämlich die nächste auffindbare und noch nicht abgelesene DATA-Anweisung selbst.

BEISPIELE :

```
10:DATA "MICHAEL",23
20:FOR J=1 TO 5
30:READ A$,B
40:PAUSE A$,B
50:NEXT J
60:END
70:DATA "SUSANNE",-24,"NICOLE",38,"   PETER",57

>
RUN
MICHAEL     23
SUSANNE   - 24
NICOLE      38
   PETER    57
ERROR 4 IN 30
```

Dieses Programmbeispiel wird mit Ausgabe eines ERROR-Codes beendet, da nach 4 Durchläufen der durch die Zeilen 20 bis 50 gebildeten Programmschleife die nächste READ-Anweisung keine lesbaren Daten mehr vorfindet. Die Programmschleife möchte zwar gerne fünf Datenpaare A$,B lesen, es stehen in den DATA-Anweisungen des Programmes jedoch nur vier solche Datenpaare bereit.

```
10:FOR I=lTO 5
20:READ N
30:PAUSE N
40:NEXT I
50:END
60:DATA 10,2*I,I+N,4,ACS(I/10)
```

DATE$

Siehe auch : TIME$, ALARM$

WIRKUNG : DATE$ ist eine Systemvariable, die das Datum der eingebauten Echtzeit-Uhr enthält.

HINWEISE : Sie kann dazu verwendet werden, das Datum zu lesen oder aber auch zu setzen.

Stellen des Datums

Um das Datum zu setzen, ist dieser Variablen ein String folgendes Formates zuzuweisen: "MM/DD"

MM bedeutet dabei die zweistellige Monatsangabe im Bereich 01..12.

DD bedeutet dabei die zweistellige Tagesangabe im Bereich 01..31.

Abfrage des Datums

Zur Abfrage des aktuellen Datums ist DATE$ als Variable ohne Wertzuweisung zu verwenden. Sie liefert einen String, dessen Format dem obig gezeigten entspricht.

Der Tag wird um den Wert 1 erhöht, wenn ein Übergang der in der Variablen TIME$ geführten Uhrzeit von 23:59:59 auf 00:00:00 stattfindet.

BEISPIEL : DATE$= "12/25"

Diese Anweisung setzt das Datum der Echtzeit-Uhr auf den 25. Dezember.

DEG

Siehe auch : DMS

WIRKUNG : Wandelt einen Winkel, der in Altgrad gemessen wird, von der sexagesimalen Form, also der Darstellung in Stunden, Minuten und Sekunden, in seine dezimale Entsprechung um.

HINWEISE : Der zu wandelnde Winkel muß dabei im Format hh.mmssrr vorliegen, wobei die Ziffern:

hh die Stunden,
mm die Minuten,
ss die Sekunden und
rr den dezimalen Sekundenrest bestimmen.

Folgende Werte sind dabei einzuhalten:

hh : O bis ..
mm : 00 bis 59
ss : 00 bis 59
rr : 00 bis 99

Das Ergebnis wird mit bis zu zehn signifikanten Ziffern angezeigt.

BEISPIEL :

```
10:X=DEG 50.3000
20:PRINT X
30.END

>
RUN
                    50.5
 >
```

DEGREE

Siehe auch : RADIAN, GRAD

WIRKUNG : Versetzt den Computer in den Winkelmodus DEGREE. In diesem Modus werden alle Winkelangaben als in Altgrad gegeben angesehen und auch in diesem Maß ausgegeben.

HINWEISE : Zur Kennzeichnung dieser Betriebsart wird in der Status-Zeile das Symbol DEG angezeigt.

Die Argumente der Funktionen SIN, COS und TAN werden dann als in Altgrad vorliegend angesehen und die Werte der Funktionen ASN, ACS und ATN in dezimalen Altgraden ausgegeben.

BEISPIEL :

```
10:DEGREE
20:PAUSE "WINKELANGABEN IN ALTGRAD"
30:PRINT ASN(0.5),ASN(1)
40:PRINT ACS(0.5),ASN(1)
50:PRINT ATN(0.5),ATN(1)
60:END
```

Lassen Sie dieses Programm zum Vergleich auch in den beiden anderen Winkel-Modi (GRAD und RADIAN) laufen.

DELETE

Siehe auch : NEW

WIRKUNG : DELETE löscht die spezifizierten Zeilen eines BASIC-Programmes.

HINWEISE :

DELETE <Zeilen-Nr.>
löscht genau diese spezifizierte Zeile, sofern sie im Programm vorhanden ist.

DELETE <Zeilen-Nr.>,
löscht, mit der genannten Zeile beginnend, alle weiteren Programm-Zeilen bis zum Programm-Ende.

DELETE <Zeilen-Nr.>,<Zeilen-Nr.>
löscht alle Zeilen eines Programmes, beginnend mit der er~ten und endend mit der zweiten angegebenen Zeile. Die zweite Zeilennummer muß dabei höherwertiger a1s die erstgenannte Nummer sein.

DELETE ,<Zeilen-Nr.>
löscht alle Zeilen eines Programmes, beginnend mit der ersten Programmzeile bis einschließlich der spezifizierten Zeile.

Um ein Programm komplett zu löschen, sollte das Kommando NEW verwendet werden.

BEISPIELE :

```
DELETE 150
DELETE 200,
DELETE 50,150
DELETE ,35
```

DIM

Siehe auch : CLEAR, ERASE

WIRKUNG : DIM dient der Reservierung von Speicherplatz für die Aufnahme von Array-Variablen des numerischen oder des String-Typs. Daneben bestimmt DIM auch die von einem Array erfaßbare Elemente-Anzahl. Bei String-Arrays läßt sich zusätzlich die Länge der aufzunehmenden Strings definieren.

HINWEISE : Mit Ausnahme der Standard-Variablen A bis Z und A$ bis Z$, die gleichbedeutend mit den beiden eindimensionalen Standard-Arrays @(1) bis @(26) und @$(1) bis @$(26) sind, müssen alle anderen Array-Variablen mit DIM dimensioniert werden, um für diese ausreichend Platz im Speicher bereit-zustellen.

Solange diese Dimensionierung fehlt, können die nicht zu den Standard-Variablen zählenden Arrays nicht benutzt werden.

<Größe> bestimmt bei eindimensionalen Arrays

die Anzahl der Elemente, die in einem solchen Array, das auch a1s Liste betrachtet werden kann, aufnehmbar sind. Die zulässigen Werte für <Größe> liegen im Bereich von 0...255, wobei sich die Elemente-Anzahl aus <Größe> + 1 ergibt.

<Zeile> bestimmt bei zweidimensionalen Arrays,

die man auch als Tabellen auffassen kann, die Nummer der höchsten vorkommenden Zeile einer solchen Tabelle. Die zulässigen Werte für <Zeile> liegen ebenfalls im Bereich: 0...255.

<Spalte> bestimmt bei zweidimensionalen Arrays die höchste vorkommende Spalte einer solchen Tabelle. Der zulässige Wertebereich ist auch hier: 0...255.

<Länge> bestimmt bei den String-Arrays wie lang die aufzunehmenden Strings sein dürfen. Weisen die Strings jedoch mehr Zeichen auf als mit <Länge> vorgegeben, werden sie auf das entsprechende Maß reduziert und alle überzähligen Zeichen gekappt. Fehlt die Angabe des Parameters <Länge>, können die Strings standardmäßig bis zu 16 Zeichen enthalten. Die maximale Stringlänge beträgt 80 Zeichen.

Die Gesamtheit der Tabellenelemente ergibt sich egen des Numerierungsbeginnes bei Null aus dem Zusammenhang:

Elementanzahl = (<Zeile> + 1) * (<Spalte> + 1)

Die Dimensionierung DIM A(2,3) deklariert somit ein Array mit 3 Zeilen und 4 Spalten. Es enthält folgende Tabellenelemente:

Spalte:	1	2	3	4
1. Zeile	A(0,0)	A(0,1)	A(0,2)	A(0,3)
2. Zeile	A(1,0)	A(1,1)	A(1,2)	A(1,3)
3. Zeile	A(2,0)	A(2,1)	A(2,2)	A(2,3)

DIM

Nachdem ein Array DIMensioniert worden ist, kann man es nicht umdimensionieren, solange nicht ein Reset des Computers oder aber einer der Befehle CLEAR, NEW, RUN oder ERASE ausgeführt wird. Ein laufendes Programm bricht mit der Ausgabe eines ERROR-Codes ab, wenn es entweder auf ein nicht mit DIM deklariertes Array trifft oder eine DIMAnweisung vorfindet, die sich auf ein bereits dimensioniertes Array bezieht. Indizes, die die mit <Größe>, <Spalte> oder <Zeile> vereinbarten

Maximalwerte überschreiten, führen ebenfalls zu einem Programmabbruch. Negative Indizes sind nicht erlaubt !

BEISPIELE :

```
10:DIM C(13)
20:DIM F$(10)
30:DIM H(4,6)
40:DIM G$(7,5)*25
```

DMS

WIRKUNG : DMS wandelt einen in Altgrad vorliegenden Winkel von der dezimalen Darstellungsart in das Format "Stunden,Minuten,Sekunden" um, das man auch als sexagesimale Notation bezeichnet.

HINWEISE : Das Wandlungsergebnis wird im Format hh.mmssrr ausgegeben, wobei folgendes gilt:

hh	Stunden	00.....
mm	Minuten	00...59
ss	Sekunden	00...59
rr	Dezimaler Sekundenrest	00...99

BEISPIEL :

```
10:X=DMS 50.5
20:PRINT X
30:END

>
RUN
                    -50.3
>
```

DSKF

WIRKUNG : DSKF liefert den Betrag des noch zur Verfügung stehenden Speicherplatzes auf dem spezifizierten Medium.

HINWEISE : Folgende Spezifikationen können für das <Medium> gewählt werden:

S1: RAM-Modul im Modulfach S1
S2: RAM-Modul im Modulfach S2
X:,Y: Diskette im Diskettenlaufwerk

Der gelieferte Betrag gibt die Anzahl der noch freien Speicherplätze in Bytes an.

BEISPIEL :

```
>
DSKF "S1:"
```

Diese Anweisung liefert die Anzahl der freien Bytes des im Modulfach S1 befindlichen Modules.

END

Siehe auch : STOP

WIRKUNG : END beendet ein laufendes Programm und schließt alle offenen Dateien und Schnittstellen.

HINWEIS : Es ist nicht zwingend erforderlich, ein Programm mit einer END-Anweisung in der letzten Programmzeile abzuschließen. Bei speziellen Strukturen, die sich bei der Verwendung von Unter-Routinen ergeben, oder in Fällen, wo mehrere voneinander unabhängige Programme in den Speicher zu laden sind, stellt die Einfügung von END-Anweisungen sicher, daß der Computer nicht aus Versehen in unerlaubte Programmteile verzweigt.

Fehlt die END-Anweisung, endet das Programm mit Aüsführung der letzten Programmzeile.

BEISPIEL :

```
10: GOSUB 50
20:PRINT "NACH WIEDERKEHR ENDET DAS"
30:PRINT "HAUPTPROGRAMM MIT ZEILE 40"
40:END
50:PRINT "HIER IST DAS UNTERPROGRAMM"
60:RETURN

>
RUN
HIER IST DAS UNTERPROGRAMM
NACH WIEDERKEHR ENDET DAS
HAUPTPROGRAMM MIT ZEILE 40
>
```

EOF

WIRKUNG : EOF liefert einen Wert, aus dem ersichtlich ist, ob beim Lesen einer sequentiellen Datei deren Ende erreicht worden ist.

HINWEISE : Der Parameter **<Dateinummer>** sorgt für die Anwahl der richtigen Datei und muß mit der Nummer übereinstimmen, unter der die Datei geöffnet wurde.

Die möglichen gelieferten Werte sind 0 oder -1. Sie haben folgende Bedeutung:

0 Dateiende noch nicht erreicht
-1 Dateiende erreicht

BEISPIEL : s. Erklärung des OPEN-Befehles.

ERASE

Siehe auch : CLEAR

WIRKUNG : ERASE löscht einfache Variablen und Arrays.

HINWEIS : Hierbei sind nur solche numerischen Variablen und String-Variablen löschbar, die nicht zu den Standard-Variablen A bis Z bzw. @(1) bis @(26) und A$ bis Z$ bzw. @$(1) bis @$(26) gehören.

Mit der ERASE-Anweisung lassen sich zwar gezielt einfache Variablen und String-Variablen löschen, jedoch nicht individuelle Elemente eines Arrays. Arrays sind nur komplett löschbar und müssen mit zwei nachfolgenden, aber leeren Klammern kenntlich gemacht sein.

Die zu löschenden Variablen können in Form einer Parameterliste an das Befehlswort ERASE angefügt werden, wobei diese durch Kommas voneinander zu trennen sind.

BEISPIEL : 1O:ERASE AB,Z$()

ERL

Siehe auch : ERN, ON ERROR GOTO, RESUME

WIRKUNG : ERL liefert die Nummer derjenigen Zeile, in der während des Programmablaufes ein Fehler festgestellt wsrden ist.

HINWEISE : Die Systemvariable ERL ist dazu gedacht, einen Fehler lokalisieren zu können, wenn durch diesen mittels ON ERROR GOTO Anweisung in eine Fehlerbehandlungs-Routine verzweigt wird. In ERL liegt nur dann eine Zeilennummer vor, wenn der Fehler während eines Programmablaufes aufgetreten ist.

BEISPIEL : s. Erklärung von ERN.

ERN

Siehe auch : ERL, ON ERROR GOTO, RESUME

WIRKUNG : ERN liefert den ERROR-Code eines Fehlers, der während eines laufenden Programmes festgestellt worden ist.

HINWEISE : Die Systemvariable ERN wird üblicherweise dazu verwendet, innerhalb einer FehlerbehandlungsRoutine die Art des aufgetretenen Fehlers feststellen, um geeignete Gegenmaßnahmen ergreifen zu können. Zur genauen Lokalisation des Fehlers wird sie deshalb auch oft gemeinsam mit ERL für die weitere Entscheidungsfindung benutzt.

BEISPIEL :

```
10:ON ERROR GOTO 70
20:FOR N=lTO 20
30:READ A
40:PRINT A
50:NEXT N
60:END
70:PRINT "HABE EINEN FEHLER IN ZEILE";ERL ;" BEME
   RKT:"
80:IF ERL =30AND ERN =4THEN PRINT "KEINE DATA-ZEI
   LE GEFUNDEN!"
90:END
RUN
HABE EINEN FEHLER IN ZEILE
30 BEMERKT:
KEINE DATA-ZEILE GEFUNDEN!
```

EXP

WIRKUNG : Die Funktion EXP(X) liefert zu einem Argument X die zur Basis der Zahl e gebildete Potenz.

HINWEISE : Das Argument X kann eine numerische Konstante oder Variable oder aber ein numerischer Ausdruck sein. Dabei muß X im Wertebereich: -227.9559242 bis +230.2585092 liegen. Werte außerhalb dieses Bereiches liefern den Funktionswert 0.

Die transzendente Basiszahl e wird intern mit 2.7181828 aneenähert.

BEISPIEL :

```
PRINT EXP(1O)
              220026.46579
>
```

FILES

Siehe auch : LFILES , SET

WIRKUNG : FILES liefert eine Liste der auf einer Diskette oder RAM-Disk befindlichen Dateien, also ein Inhaltsverzeichnis dieser Speichermedien.

HINWEISE : Das Inhaltsverzeichnis zeigt jede Datei unter Nennung folgender Einzelinformationen an:

- -Dateiname
- -Extension (z.B.: .BAS für BASIC-Programme)
- -eventuellem Schreibschutzstatus
- -Datum der Aufzeichnung
- -Uhrzeit der Aufzeichnung

Die gewünschte Liste wird durch den anzugebenden Parameter <Dateibezeichner> bestimmt. Dieser ist in obigem Diagramm aufgeschlüsselt dargestellt. Er besteht aus den Anteilen:

<Datenquelle>, <Dateiname> und <Extension>.

Der Parameter **<Datenquelle>:** meint eines der nachstehenden Speichermedien:

X: Diskette
Y: andere Diskette
S1: RAM-Disk in Modulfach S1
S2: RAM-Disk in Modulfach S2

Der **<Dateiname>** bestimmt, welche Datei(en) in der Auflistung erscheinen sollen. Dieser Name kann aus bis zu acht Zeichen bestehen. Ersetzt man ihn durch einen Stern, werden alle Dateien, die dieselbe <Extension> aufweisen, angezeigt. Er ist damit beliebig mehrdeutig angegeben. Mit Hilfe von Fragezeichen läßt sich der <Dateiname> auch teilweise mehrdeutig machen.

Da das Display nur eine begrenzte Anzahl Dateien anzeigen kann, lassen sich mit der Taste ↑ die weiteren Dateiangaben Stück für Stück auflisten. Durch Betätigung jeder beliebigen Taste (ausgenommen: [SHIFT] ,[CTRL], [DEF] und [SML] läßt sich die Auflistung abbrechen.

Die **<Extension>** bestimmt, welche Art Datei(en) anzuzeigen sind. So bewirkt beispielsweise die Angabe .BAS die Anzeige von BASIC-Dateien. Die <Extension> kann ebenfalls durch einen Stern oder aber mit Fragezeichen mehrdeutig ausgelegt werden.

Mehrdeutige Datei spezifikationen

Durch Anwendung der Joker-Zeichen (wildcards) * und ? lassen sich wie folgt mehrdeutige Dateibezeichnungen angeben und damit mehrere Dateien ansprechen:

Dateispezifikation	Beispiele	passender Dateien
TEST?	TEST .BAS	TEST1 .PPP
T??T	TEST .BAS	TEXT .BAS
S?MPLE.BIN	SIMPLE .BIN	SAMPLE .BIN
A?????	ABCDEF .	APPLES .XXX
?A*	RATES .FIN	RANDOMLY.GEN
*.XYZ	SHARP .XYZ	PC1600 .XYZ
NEW.A?C	NEW .ABC	NEW .A5C

Eine Mehrdeutigkeit kann auch dadurch erreicht werden, indem man den <Dateinamen> oder aber die <Extension> oder aber beide wegläßt. Allerdings kann bei vorhandener <Extension> der <Dateiname> nicht entfallen, er muß dann durch einen Stern repräsentiert werden.

Grundlegende Parameter - Formate

"<Datenquelle>:"
"<Datenquelle>:<Dateiname>"
"<Datenquelle>:*.<Extension>"
"<Datenquelle>:*.*"
"<Datenquelle>:<Dateiname>.<Extension>"

"<Datenquelle>:<Dateiname>." ist ein besonderes Format und sucht nach Dateien des angegebenen Namens, die keine <Extension> aufweisen.

FOR .. NEXT

WIRKUNG : Mit FOR und NEXT lassen sich Programmschleifen bilden und damit Anweisungen mehrfach in vorbestimmter Weise (determiniert) ausführen.

HINWEISE : Der erste Parameter ist eine <num. Variable> und bestimmt, welche Variable als Schleifenzähler (Laufvariable) dienen soll.

Der zweite Parameter weist dieser Laufvariablen einen Anfangswert zu, der durch jeden beliebigen <num. Ausdruck> gebildet sein kann.

Der dritte Parameter gibt den Endwert der Laufvariablen an. Ist er über- oder unterschritten, wird die Programmschleife verlassen und mit der Anweisung nach dem NEXT-Befehl fortgefahren. Dieser Parameter kann ebenfalls ein beliebiger <num. Ausdruck> sein. Er muß im Werte-Bereich -32768...32767 1iegen.

Der vierte Parameter ist optional und gibt die Schrittweite, also den Betrag an, der nach jedem Schleifendurchlauf zur Laufvariable hinzugezählt wird (unter Berücksichtigung des Vorzeichens). Die Schrittweite kann nur ganzzahlig sein. Werte mit Nachkommstellen werden auf Integer-Werte gerundet. Der Wert Null ist nicht als Schrittweite erlaubt und hat die Anzeige des ERROR-Codes 19 zur Folge. Wegen der soeben erwähnten Abrundung sind Werte, die dem Betrage nach kleiner als 1 sind, unzulässig, da sie gerundet den Wert Null ergeben. Der zulässige Werte-Bereich ist damit -32768...-1 und 1...32767. Fehlt die Angabe des STEP-Wertes, gilt für ihn standardmäßig 1.

Trifft der Computer auf einen NEXT-Befehl, sucht er sich diejenige zuvor stehende FOR-Anweisung, die mit derselben Laufvariable behaftet ist, wie sie in der NEXT-Anweisung spezifiziert ist. Dann addiert er die Schrittweite zum derzeitigen Wert der Laufvariablen und überprüft, ob ihr Endwert "überschritten" wird. Bei positiver Schrittweite überprüft er, ob der neue Variablenwert größer als der Endwert ist und bei negativen, ob dieser kleiner als der Endwert ist. Trifft dieses zu, wird die Programmschleife verlassen. Im anderen Falle werden die in der Schleife aufgeführten Anweisüngen erneut ausgeführt: diesmal mit dem neuen Wert der Laufvariablen.

Damit der Computer die richtige FOR-Anweisung finden kann, muß in der NEXT-Anweisung die zugehörige Laufvariable als Parameter angegeben werden.

```
10:FOR I=1 TO 20
50:NEXT I
230:FOR K=2 TO 17 STEP 2
290:NEXT K
```

FOR-NEXT-Schleifen lassen sich auch ineinander verschachteln:

```
10:FOR M=1 TO 10
20:FOR N=5 TO 20 STEP 5
80:NEXT N
90:NEXT M
```

FOR-NEXT-Anweisungen treten immer paarweise auf. Zu jeder FOR-Anweisung muR in logischer Folge eine passende NEXT-Anweisung vorzufinden sein. Ist diese Bedingung nicht gegeben, weil ein Teil des Paares fehlt oder aber durch eine verkehrte Schachtelung vorliegt, stoppt der Computer das Programm und gibt einen ERROR-Code aus.

GCURSOR

WIRKUNG : GCURSOR positioniert den Grafik-Cursor auf dem gewünschten Display-Punkt.

HINWEISE : Im Grafik-Modus läßt sich das Display als Matrix mit 156 x 32 einzeln adressierbaren Punkten ansprechen. Der Grafik-Cursor kann hierbei nicht nur auf eine innerhalb dieser Matrix liegende Koordinate positioniert werden, sondern auch außerhalb davon. Nach seiner Positionierung kann mit der GPRINT-Anweisung mit der Ausgabe von Grafik-Mustern begonnen werden.

Soll der Cursor im sichtbaren Bereich liegen, so sind folgende Koordinaten einzuhalten:

<X-Koordinate> : 0...155
<Y-Koordinate> : 0...31

Beide Koordinaten können jedoch auch Werte im Bereich von -32768 bis 32767 annehmen.

Wird die Angabe der Y-Koordinate weg8elassen, gilt die derzeit gültige Y-Position des Grafik Cursors.

Im Anzeigemodus MODE 1 (PC-1500 Modus) ist der Y-Parameter bedeutungslos und sollte daher in dieser Betriebsart nicht angegeben werden.

BEISPIEL :

```
:
300:X=5
310:Y=20
320:GOSUB 600
:
590:END
600:GCURSOR
610:GPRINT 255,255,255,255
620:RETURN
```

GLCURSOR

Siehe auch : LCURSOR

WIRKUNG : GLCURSOR bewegt im Grafik-Modus den Schreibstift des Druckers an die spezifizierte Position.

HINWEISE : Im Unterschied zum Kommando LCURSOR wird hier der Stift nicht an eine bestimmte Zeichenspalte gesetzt, sondern an die vorgegebene Position, die durch ein Koordinatenpaar (X,Y) definiert ist. Dieser Punkt kann somit auch ein Punkt innerhalb einer Zeichenspalte sein.

Von einem definierten Punkt (X,Y) bestimmmt:

X den horizontalen und
Y den vertikalen Abstand des Punktes zum derzeit

gültigen Koordinatenursprung (Origo).

Die Parameter, <X-Koordinate> und <Y-Koordinate> dürfen wertemäßig im Bereich von -2048 bis 2047 angegeben werden.

Während der Fahrt zur angegebenen Position wird der Stift angehoben.

BEISPIEL : Siehe LLINE

GOSUB ... RETURN

Siehe auch : GOTO, ON..GOSUB, ON..GOTO

WIRKUNG : GOSUB ruft das durch eine <Zeilennummer> oder eine <Marke> spezifizierte Unterprogramm auf.

HINWEISE : Ein Unterprogramm (subroutine) ist eine Gruppe von aufeinanderfolgenden Programmzeilen, die im Ablauf des Gesamtprogrammes mehrfach benötigt werden. Mit Hilfe von Unterprogrammen läßt sich also gewaltig Speicherplatz sparen, wenn man die wiederkehrenden Anweisungen nicht entsprechend oft in das Programm einfügt, sondern sie einmal zusammengefaßt im Speicher ablegt und sie stellvertretend durch GOSUB-Anweisungen aufruft.

Jedes Unterprogramm muß mit dem Befehl RETURN abgeschlossen sein, damit eine Rückkehr in das aufrufende Hauptprogramm möglich ist und der normale Programmablauf hinter der Anweisung GOSUB fortgesetzt werden kann.

Ein Unterprogramm läßt sich beliebig oft von diversen GOSUB-Anweisungen aufrufen, wobei es selbst wiederum andere Unterprogramme aufrufen kann und so fort. Somit entstehen verschachtelte Unterprogramme, von denen der PC-1600 eine Tiefe von bis zu 15 Ebenen verarbeiten kann. Ist diese Verschachtelungstiefe größer, wird das laufende Programm abgebrochen und ein ERROR-Code auf dem Display angezeigt. Das fehlerhafte Programm muß dann so umstrukturiert werden, daß eine solche Schachtelungstiefe vermieden wird.

Unterprogramme können sich auch selbst aufrufen, wenn man das Programm entsprechend geschickt auslegt und dafür sorgt, daß das Unterprogramm verlassen werden kann und die zulässige Tiefe der Verschachtelung nicht überschritten wird.

BEISPIEL :

```
10:WAIT 100
15:PRINT "HAUPTPROGRAMM GESTARTET"
20: GOSUB 90
25:PRINT "ZURUECK IM HAUPTPROGRAMM"
30: GOSUB 60
35:PRINT "ZURUECK IM HAUPTPROGRAMM"
40:WAIT 0
45:PRINT "PROGRAMM BEENDET"
50:END
55:REM $$$$ UNTERPROGRAMME $$$$$
60:REM --- UNTERPROCRAMM 1 --
65:PRINT "UNTERPROGRAMM 1 GESTARTET"
70: GOSUB 90
75:PRINT "ZURUECK IM UNTERPROGRAMM 1"
80:RETURN
85:REM --- UNTERPROGRAMM 2 --
90:PRINT"UNTERPROGRAMM 2 GESTARTET"
95: RETURN

>RUN
HAUPTPROGRAMM GESTARTET
UNTERPROGRAMM 2 GESTARTET
ZURUECK IM HAUPTPROGRAMM
UNTERPROGRAMM 1 GESTARTET
UNTERPROGRAMM 2 GESTARTET
ZURUECK IM UNTERPROGRAMM 1
ZURUECK IM HAUPTPROGRAMM
PROGRAMM BEENDET
>
```

GOTO

Siehe auch : GOSUB..RETURN, ON..GOTO, CONT

WIRKUNG : GOTO verzweigt den weiteren Programmablauf zu der durch eine <Zeilennummer> oder einer <Marke> spezifizierten Zeile.

HINWEISE : Die GOTO-Anweisung führt einen nichtbedingten Sprung aus, d.h., der Sprung zur spezifizierten Zeile wird unweigerlich ausgeführt und hängt von keiner Bedingung ab (es sei denn man wählt die Anweisung IF..THEN GOTO).

Im Gegensatz zu GOSUB erfolgt kein Rücksprung an die nach GOTO folgende Anweisung.

Wird als Sprungziel eine Zeile bestimmt, die die nichtausführbaren Befehle DATA oder REM enthält, wird der Programmablauf mit der nächsten Zeile (bzw. ausführbaren Anweisung) fortgesetzt. Bei nicht vorhandenem Sprungziel erfolgt ein Abbruch des Programmes mit Anzeige eines ERROR-Codes.

Im RUN-Modus läßt sich mit der GOTO-Anweisung auch ein Programm ab einer speziellen Zeile starten. Im Gegensatz zum Kommando RUN werden hierbei keine Variablen gelöscht.

Mit der Anweisung GOTO kann auch ein Programm, welches mit der BREAK-Taste unterbrochen worden ist, fortgesetzt werden. (Siehe auch: CONT).

Eine GOTO-Anweisung ohne Parameterangabe startet das Programm mit der ersten vorhandenen Zeile.

BEISPIEL :

```
10:INPUT A$
20:IF A$="J" THEN 40
30:PRINT "NEIN" : GOTO 50
40:PRINT "JA"
50:END
```

GPRINT

Siehe auch : GCURSOR

WIRKUNG : GPRINT zeichnet Grafik-Muster auf dem Display.

ANMERKUNGEN: Mit GPRINT können jeweils acht direkt übereinanderliegende Display-Punkte durch geeignete Bit-Muster beeinflußt und somit Grafiken auf dem Display abgebildet werden. Jede vertikale Punktreihe hat dabei die Höhe eines im Text-Modus dargestellten Zeichens. Wie sich der Einfluß der Bit-Muster, die je aus 8 Bits bestehen und damit einem Byte entsprechen, auswirkt, hängt von den optionalen Parametern SET, OR und XOR ab.

SET überschreibt die bisher auf dem Display befindliche Anzeige und bildet damit das Bit-Muster originalgetreu ab.

OR verknüpft das definierte Bitmuster mit den momentanen Punktzuständen unter Anwendung der logischen Oder-Funktion (OR function). Alle dunklen Punkte bleiben unverändert. Die hellen Punkte bleiben jedoch nur dann hell, wenn die ihnen zugeordneten Bits des Bitmusters gelöscht sind. Im anderen Falle werden die betreffenden Punkte gesetzt, also dunkelgesteuert.

XOR verknüpft das definierte Bitmuster und das auf dem Display befindliche Punktmuster mit der Exklusiv-Oder-Funktion (XOR function).

Fehlt die Parameterangabe SET, OR oder XOR, so wird standardmäßig die SET-Funktion angenommen.

Die **<Bit-Muster>** können in dezimaler oder aber hexadezimaler Aufzählung angegeben werden, wobei sie jeweils durch ein Semikolon voneinander zu trennen sind. Hexadezimale Werte sind dabei wie üblich durch ein vorangestelltes Kaufmannsund & zu kennzeichnen, Desweiteren lassen sich diese Muster auch mit einem <Hex-String> vereinbaren, wobei jedes Byte zwischen 00 und FF stets mit zwei Ziffern ohne Verwendung des & darzustellen ist. Der Verbund der so aufgeführten Bytes ist dann in Anführungsstriche einzuschließen. Wird innerhalb eines solchen Strings eine Ziffer vergessen und damit seine Zeichenanzahl ungerade, so interpretiert der Computer alle Zweiergruppen an Ziffern als gültiges Byte und läßt die letzte verbleibende Ziffer unbeachtet.

Nachstehende Anweisungen sind in ihrer Wirkung identisch:

```
GPRINT 16;40;18;253;18;40;16         (dezimal)
GPRINT  &10;&28;&12;&FD;&12;&28;&10  (hexadezimal)
GPRINT "102812FD122810"              (Hex-String)
```

GPRINT ohne jegliche Parameterangabe setzt den Grafik-Cursor um eine Punktzeile nach unten ohne dabei den auf dem Display befindlichen Inhalt zu beeinflussen. Schließt eine GPRINTAnweisung mit einem Semikolon ab, so bleibt der Grafik-Cursor an der letzten Position stehen. Die nächste GPRINT-Anweisung setzt sich dann an dieser Stelle fort. Wählt man ein Komma als Trennzeichen, so wird zwischen die so getrennten Punktmuster eine Lücke von einer Punktbreite gesetzt.

Beispiel des Zusammenhanges zwischen Punkt und Bit - Muster, gezeigt für die Option SET:

Punkte	Bit	Zustand	
x	0	0	
x	1	0	76543210 <-Bit
x	2	1	::::::::
	3	0	1010110 Binärwert
x	4	1	
	5	0	&D5 Hex-Wert
x	6	1	
x	7	1	214 Dezimalwert

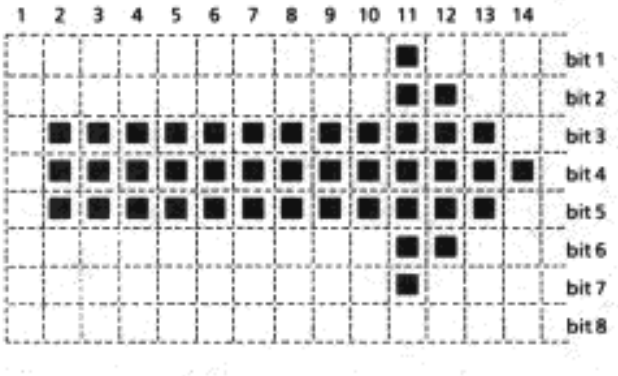


Abbildung A

Das in obiger Abbildung gezeigte Grafik-Symbol kann auf dem Display durch die Ausgabe von 14 Bit-Mustern erzeugt werden. Dabei gilt:

```
Spalte 1:  00000000 = &00
Spalte 2:  00011100 = &1C
:
:
Spalte 10: 00011100 = &1C
Spalte 11: 01111111 = &7F
Spalte 12: 00111110 = &3E
Spalte 13: 00011100 = &1C
Spalte 14: 00001000 = &08
```

Mit folgender Anweisung könnte diese Grafik also zur Anzeige gebracht werden:

```
GPRINT &0;&1C;&1C;&1C;&1C;&1C,&1C;
&lC;&lC;&lC;&7F;&3E;&1C;&8
```

BEISPIEL :

```
10:CLS
20:FOR I=1 TO 77
30:GPRINT 255;214;
40:NEXT I
50:END
```

GRAD

Siehe auch : DEGREE, RADIAN

WIRKUNG : Versetzt den Computer in den Winkelmodus GRAD. In diesem Modus werden alle Winkelangaben als in Neugrad gegeben angesehen und auch in diesem Maß ausgegeben.

HINWEISE : Zur Kennzeichnung dieser Betriebsart wird in der Status-Zeile das Symbol GRAD angezeigt.

Alle Argumente der Funktionen SIN, COS und TAN werden als in Neugrad vorliegend angesehen und die Werte der Umkehrfunktionen ASN, ACS und ATN im Winkelmaß Neugrad geliefert.

BEISPIEL :

```
10:GRAD
20:PAUSE "WINKEL IN NEUGRAD"
30:PRINT ASN(0.5),ASN(1.0)
40:PRINT ACS(0.5),ACS(1.0)
50:PRINT ATN(0.5),ATN(1.0)
60:END

>
RUN
WINKEL IN NEUGRAD
33.33333333         100.
66.66666667         0.
29.516723535        0.
>
```

Probieren Sie dieses Programm auch in den beiden anderen Winkel-Modi (DEGREE und RADIAN) aus.

GRAPH

Siehe auch : TEXT

WIRKUNG : Setzt den Drucker (CE-1600P bzw.CE-150) in den Grafik-Modus.

HINWEIS : Mit Ausführung von GRAPH werden die Parameter <Limit 1> und <Limit 2> des Befehles PAPER mit ihren standardmäßigen Werten belegt.

HEX$

Siehe auch : VAL

WIRKUNG : Die Funktion HEX$ liefert einen String, der sich aus den Hex-Ziffern des als Hexadezimalwert aufgefaßten Argumentes zusammensetzt.

HINWEISE : Das Argument muß ganzzahlig sein und dezimal betrachtet im Bereich 0....65535 liegen. Der gelieferte String ergibt sich damit im Bereich von &0....&FFFF.

HEX$(64) liefert den String: "&40"

Will man eine hexadezimale Konstante in einen String umwandeln, muß ihr ein sogenanntes Kaufmannsund & vorangestellt werden.

HEX$(&E0) liefert den String: "&E0"

BEISPIEL :

```
10:PRINT "WANDLUNG: DEZIMAL ZU HEX"
20:INPUT " DEZIMAL-ZAHL = ";X
30:IF X>65535 THEN 100
40:IF X<0 THEN 110
50:PRINT "HEXADEZIMAL-WERT = ";HEX$(X):PRINT
60:INPUT "NOCH EINE ZAHL (J/N) ";A$
70:IF A$="J" THEN 20
80:IF A$="N" THEN END
90:GOTO 60
100:PRINT "FEHLER: MAXIMUM=65535 !":GOTO 20
110:PRINT "FEHLER: MINIMUM=0 !":GOTO 20
120:END
```

IF .. THEN .. ELSE

WIRKUNG : IF...THEN...ELSE entscheidet über den weiteren Programmverlauf, je nachdem, ob eine Bedingung erfüllt ist oder nicht.

HINWEISE : Die Entscheidung hängt dabei von einer zwischen den Befehlswörtern IF...THEN zu überprüfenden **<Bedingung>** ab. Ist diese erfüllt, wird mit der hinter dem Befehlswort THEN angegebenen Zeile, die durch eine **<Zeilen-Nr.>** oder eine **<Marke>** spezifiziert ist, oder den dortigen stehenden **<Anweisungen>** fortgefahren. Im anderen Falle wird die nächste Zeile ausgeführt.

Enhält die Anweisung IF...THEN den Zusatz ELSE, so wird bei nicht erfüllter Bedingung das Programm nicht mit der nächsten Zeile fortgesetzt, sondern mit der hinter ELSE spezifizierten Zeile oder den dortigen Anweisungen.

Folgt dem Befehlswort ELSE keine Zeilenspezifikation, werden alle Anweisungen (auch die durch einen Doppelpunkt voneinander getrennten) ausgeführt, solange diese sich in derselben Programmzeile befinden.

Fehlt der Zusatz ELSE, so gilt das eben gesagte für das Befehlswort THEN.

IF .. THEN .. ELSE

Die Anweisungen IF..THEN..ELSE können innerhalb einer Programmzeile von maximal 80 Zeichen Länge auch ineinander verschachtelt werden.

Die zwischen den Befehlswörtern IF...THEN abzufragende **<Bedingung>** wird durch einen **logischen Ausdruck** gebildet, der sich, so komplex er auch immer sein mag, stets aus folgenden Grundformen aufbauen läßt:

X=Y	Vergleich auf Gleichheit
X<>Y	Vergleich auf Ungleichheit
X<Y	Vergleich, ob X kleiner als Y
X>Y	Vergleich, ob X größer als Y
X<=Y	Test, ob X kleiner oder gleich Y.
X>=Y	Test, ob X größer oder gleich Y.
X AND Y	Logische UND-Verknüpfung
X OR Y	Logische ODER-Verknüpfung
NOT X	Logische Verneinung

Beispiele logischer Ausdrücke:

X=1

Bedingung ist erfüllt, wenn X den Wert 1 hat.

(P=2 AND Q=4) OR P=1

Die Bedingung ist erfüllt, wenn entweder P den Wert 1 besitzt (unabhängig von Q) oder aber P=2 und Q=4 gilt.

BEISPIEL :

```
10:INPUT "SOLL ICH PIEPSEN ",A$
20:IF A$="N" THEN 60
30:IF A$="J" THEN BEEP 3:GOTO 10
40:PRINT "NUR J ODER N EINGEBEN !"
50:GOTO 10
60:PRINT "SCHADE !"
70:END
```

INIT

Siehe auch : SETCOM, SETDEV, TITLE

WIRKUNG : INIT bewirkt folgendes:

1. Initialisiert die Module in den Modulfächern S1, S2 und spezifiziert ihren Gebrauch.

2. Initialisiert und formatiert Disketten.

3. Setzt die Größe des Empfangspuffers für die beiden seriellen Schnittstellen.

HINWEISE : **Modul-Initialisierung**

Das mit den Parametern S1: oder S2: ausgewählte

Modul kann mit einem der folgenden Parameter für den gewünschten Einsatz initialisiert werden. (Dieses Befehlsformat ist nur in MODE O aktiv !)

"F" Formatiert das RAM-Modul als RAM-Disk, so daß auf diesem Dateien in gleicher Weise gespeichert und verwaltet werden können wie auf einer echten Diskette.

"M" Initialisiert das spezifizierte Modul a1s Erweiterung des Arbeitsspeichers, damit sich auch umfangreichere Programme in den Computer laden lassen.

"P" Initialisiert ein Modul als Programm speicher, so daß sich auf diesem ein Programm auf unbestimmte Zeit speichern und jederzeit schnell zurückholen läßt. Wegen der im Modul befindlichen Puffer batterie bleibt das Programm auch dann erhalten, wenn sich das Modul außerhalb des Computers befindet. Nach Einsetzung des Modules in den Computer ist dann das Programm sofort wieder nutzbar.

Der INIT-Befehl ist nicht anwendbar auf Module, die bereits Programme oder Dateien enthalten und mit dem Schreibschutzschalter geschützt sind (Schalterstellung ON).

Disketten-Initialisierung

INIT "X:" initialisiert und formatiert eine im Laufwerk befindliche Diskette. Neue Disketten müssen vor ihrem ersten Gebrauch formatiert werden, da sie sonst nicht benutzbar sind. Wird der Inhalt einer bereits gebrauchten Diskette nicht mehr benötigt, kann ihr gesamter Inhalt mit INIT gelöscht werden.

Empfangspuffer-Einstellung

Die Größe des Empfangspuffers für die Aufnahme der Daten, die über eine der beiden seriellen Schnittstellen (RS-232C, SIO) geliefert werden, wird mit der Anweisung:

INIT "COMn:" (Puffergröße> bestimmt.

Der Puffer kann maximal 16383 Bytes groß sein und minima1 zu 40 Bytes gewählt werden, wobei der Parameter **<Puffergröße>** jedoch im Bereich 80....16383 liegen muß. Die Puffergröße von 40 Bytes wird durch den Wert 0 eingestellt. Diese Größe wird nach jedem Einschalten des Computers automatisch als Standardwert eingestellt.

Mit dem Paramter **"COMn"** kann folgende Auswahl der Schnittstellen getroffen werden:

COM1: RS-232C-Interface
COM2: SIO-Interface
COM: über SETDEV selektiertes Interface

Werden Dateien zwischen zwei Computern ausgetauscht, die direkt miteinander verbunden sind (ohne Modem, Akustikkoppler usw.), erlaubt ein großer Empfangspuffer einen schnellen Austausch. Werden keine Kommunikationsfunktionen benötigt, lädt andererseits ein minimaler Pufferspeicher (40 Bytes) einen umso größeren Arbeitsspeicher zu. In der Praxis muß also zwischen den beiden Extremen ein guter Kompromiß gefunden werden. Hierbei hat sich eine typische Puffergröße von 255 Bytes als ratsam ergeben.

Steht für die Einrichtung der spezifizierten Puffergröße nicht ausreichend Speicherplatz zur Verfügung oder befindet sich eine Datei für die APPEND-Option noch im geöffneten Zustand, wird ein ERROR-Code ausgegeben.

INKEY$

Siehe auch : INPUT

WIRKUNG : INKEY$ überprüft während der Programmausführung den Tastaturpuffer, ob zwischenzeitlich Zeichen über die Tastatur eingegeben worden sind, und liest ein einzelnes Zeichen in die spezifizierte Stringvariable ein.

HINWEISE : **INKEY$** und **INKEY$(0)** haben dieselbe Bedeutung und lesen jeweils das zuletzt über die Tastatur eingegebene Zeichen aus dem Tastaturpuffer und weisen es der spezifizierten Stringvariable zu.

Wurde zwischenzeitlich kein Zeichen über die Tastatur eingegegeben, liefert INKEY$ einen Nullstring (ASCII-Code 0).

Enthält der Tastaturpuffer mehrere Eingaben, kann mit **INKEY$(1)** die älteste Eingabe gelesen werden.

Der Unterschied zwischen den Befehlen INKEY$ und INPUT besteht darin, daß INPUT auf die Eingabe von Zeichen wartet und dieses dann auf dem Display anzeigt, während INKEY$ nicht wartet und auch keine Anzeige tätigt.

INKEY$

Mit der INKEYS-Anweisung lassen sich nur die Zeichen nachstehender Tabelle lesen:

INKEY$ Character Table

Low \ High	0	1	2	3	4	5	6
0			SPACE	0		P	
1	SHIFT	F1		1	A	Q	
2	SML	F2		2	B	R	
3	CTRL	F3		3	C	S	
4	KBD/CLICK	F4		4	D	T	
5	BS	F5		5	E	U	
6		F6		6	F	V	
7				7	G	W	
8	◀	CL	(	8	H	X	
9	◆	RCL	)	9	I	Y	
A	↓		*		J	Z	
B	↑	DEF	+		K		
C	▶				L		
D	ENTER		−	=	M		
E	ON		.		N		
F	OFF	MODE	/		O		

Abbildung F : Von INKEY$ akzeptierte Zeichen

BEISPIEL :

```
:
:
300:A$=INKEY$
310:IF A$="" THEN 300
320:IF A$="*" THEN 500
330:GOTO 300
:
500PRINT "HALLO"
```

INP

Siehe auch : OUT

WIRKUNG : INP liefert ein Datenbyte vom spezifizierten Port des Z80-kompatiblen Mikroprozessors.

HINWEISE : Der Parameter <Port> bestimmt den Eingabe-Port, von dem ein Byte zu holen ist. Die Spezifikation des Ports geschieht durch eine Adresse, also mit mit einem 16-Bit-Wert im Bereich 0....65535 bzw. &0...&FFFF.

BEISPIEL :
```
:
300:A=INP(20)
```

INPUT

WIRKUNG : INPUT erlaubt während eines laufenden Programmes Variablen Werte zuweisen zu können.

HINWEISE : Mit Ausführung einer INPUT-Anweisung wird der Programmablauf gestoppt und eine <Meldung> auf dem Display angezeigt, sofern eine solche in der Anweisung spezifiziert worden ist. Fehlt diese, erscheint an ihrer Stelle ein Fragezeichen.

Während dieser Pause im Programmablauf können Daten über die Tastatur eingegeben oder über die seriellen Schnittstellen empfangen werden. Im letzten Fall müssen diese über SETDEV selektiert worden sein. Die entgegengenommenen Daten werden den a1s Parameter aufgelisteten <Variablen> der Reihe nach zugeordnet. Die Variablen sind in der Liste durch Kommas, die entsprechenden Tastatureingaben durch Betätigung der ENTER-Taste voneinander zu trennen.

Wird keine <Meldung> vereinbart, so erscheint in der Anzeige nur ein Fragezeichen, um auf eine notwendige Dateneingabe aufmerksam zu machen. Ist dagegen eine <Meldung> vereinbart, erfolgt vor Anzeige des Fragezeichens eben die Ausgabe dieser Meldung. Die Ausgabe des Fragezeichens läßt sich unterdrücken, wenn man dem Parameter <Meldung> ein Semikolon nachstellt.

In allen eben beschriebenen Fällen wird der Cursor hinter dem Fragezeichen bzw. der Meldung positioniert. Folgt der <Meldung> jedoch ein Komma, wird der Cursor auf den Anfang der Zeile gesetzt, die sich oberhalb der Zeile befindet, in der die Meldung angezeigt wird.

Um Daten von der RS-232C-Schnittstelle über den INPUT-Befehl einzulesen, ist das Format:

INPUT <Varible>, <Variable> ...

zu wählen. Hierbei kann weder eine <Meldung> vereinbart werden, noch erfolgt die Anzeige eines Fragezeichens.

BEISPIEL :

```
10:PRINT "BERECHNUNG: QUADER-VOLUMEN"
20:PRINT "Werte in Metern eingeben !"
30:INPUT "L=";L,"B=";B,"H=";H
40:V=L*B*H
50:PRINT "QUADER-VOLUMEN=";V;"m";CHR$(&FD)
60:END

>
RUN
BERECHNUNG: QUADER-VOLUMEN
Werte in Metern eingeben !
L=100
B=2*4
H=SQR(144)
QUADER-VOLUMEN= 9600m2
>
```

Siehe auch : DIM, INPUT, OPEN, PRINT#, SET

WIRKUNG : INPUT # liest Datensätze aus einer sequentiellen Datei, die sich auf einer Cassette, Diskette oder RAM-Disk befinden.

Disketten-Dateien und RAM-Disk-Dateien:

<Datei-Nr.> ist die Nummer, die der Datei bei ihrer Öffnung durch den OPEN-Befehl zugewiesen worden ist. Ein Versuch, eine ungeöffnete Datei zu lesen, endet mit der Ausgabe eines ERRORCodes.

Cassetten-Dateien:

<Dateiname> bestimmt, welche Datei auf Cassette zu suchen und dann zu lesen ist. Fehlt dieser Parameter, wird die nächste auf Cassette befindliche Datei genommen.

Cassetten-, Disketten- und RAM-Disk-Dateien:

<Variable> bestimmt die Namen der Variablen, in die die Daten der Datensätze einzulesen sind. Die Aufzählung der Variablen kann sowohl aus einfachen Variablen, StandardVariablen oder Arrays bestehen. Reihenfolge und Typ der gelieferten Daten müssen zu den bereitgestellten Variablen passen. Stringvariable sind dabei in ausreichender Länge zu dimensionieren. Arrays müssen in der Variablen-Liste mit dem Pseudo-Index (*) versehen sein, z.B.: A($).

Ist die Anzahl der gelesenen Daten geringer als die der aufgelisteten Variablen, hält das Programm an und wartet auf die restlichen Daten. Dieser Wartezustand kann durch Betätigung der

BREAK Taste beendet werden. Ist die Anzahl der gelieferten Daten dagegegen größer als die der bereitgestellten Variablen, so werden alle überzähligen Daten ignoriert.

BEISPIEL :

```
lO:MAXFILES=1
20:0PEN "X:=ADRESSE" FOR INPUT AS #1
30:PRINT "NEW YORK":PRINT
40:PRINT "NAME","TELEFON-NR."
50:IF EOF(1) THEN 100
60:INPUT #1,N$,S$,T$
70:IF S$="NEW YORK" THEN 90
80:GOT0 50
90:PRINT N$,T$ : GOTO 50
lOO:CLOSE #1
110:END
```

INSTAT

Siehe auch : OUTSTAT

WIRKUNG : INSTAT liefert einen numerischen Wert, der der Einstellung der RS-232C-Steuersignale angibt.

HINWEISE : Die Einstellung der Steuersignale wird durch einen 8-Bit-Wert, also einem Byte beschrieben. Die Zuordnung und Bedeutung ist wie folgt:

Bit	Signal	Bit=0	Bit=1
0	DTR	high	low
1	RTS	high	low
2	CTS	high	low
3	CD	high	low
4	DSR	high	low
5	CI	high	low

Die Bits 6 und 7 sind ständig 0.

BEISPIEL :
```
INSTAT "COM1:"
 63
>
```

Der in diesem Beispiel gelieferte Wert kann in seiner Bedeutung nach der Umwandlung in den HexWert (&3F) und dessen Umwandlung in die binäre Entsprechung wie folgt analysiert werden:


```
        63      dezimal
=      &3F      hexadezimal

=   00111111    binär
```

Daraus folgt, das alle sechs Steuersignale im Zustand "low" befindlich sind.

INSTR

WIRKUNG : INSTR sucht in einem String nach dem erstmaligen Auftauchen eines Teilstringes und liefert dessen Position a1s Integer-Wert.

HINWEISE : Wird der Teilstring nicht gefunden oder ist der abzusuchende String ein Nullstring, so wird als Ergebnis der Wert 0 geliefert.

Der Parameter **<Position>** bestimmt, ab welcher Position mit der Suche zu beginnen ist. Fehlt diese Angabe, startet die Suche mit Position 1, also dem ersten Stringzeichen.

BEISPIEL :

```
10:INPUT "GIB EIN WORT EIN: ",W$
20:N=INSTR(W$,"A");
30:IF N=0 THEN 80
40:PRINT "DER BUCHSTABE A TAUCHT"
50:PRINT "ZUM ERSTEN MALE AN DER"
60:PRINT "POSITON ";N;" AUF."
70:GOTO 10
80:PRINT "IN DEM WORT KOMMT DER"
90:PRINT "BUCHSTABE A NICHT VOR !"
100:GOTO 10
```

INT

WIRKUNG : Die Funktion INT(X) schneidet vom Argument X alle Nachkommastellen ab und liefert lediglich dessen ganzzahligen Anteil (INTeger-Wert).

HINWEISE : Der Funktionswert ergibt sich aus der Abrundung des Argumentes auf den nächstniedrigeren ganzzahligen Wert.

Die Rundung ist völlig unabhängig davon, ob ein positiver oder negativer Wert als Argumentes angegeben ist. So werden positive Werte betragsmäßig kleiner, negative hingegen größer.

BEISPIEL :

```
10:PRINT INT(3.3)
20:PRINT INT(-3.3)
30:PRINT INT(.2)

>RUN
 3.
-4.
 0.
```

KBUFF$

Siehe auch : INKEY$

WIRKUNG : KBUFF$ schreibt einen String von Zeichen in den Tastaturpuffer.

HINWEISE : Das KBUFF$-Kommando kann dazu verwendet werden, um den Computer im "batch mode,, zu betreiben. Alle System-Kommandos können in der Reihenfolge ihrer gewünschten Abarbeitung in einer Datei abgelegt und später in den <Befehlsstring> eingelesen werden. KBUFF$ schreibt diesen String dann in den Tastaturpuffer, was die Wirkung hat, als ob die im String enthaltenen Kommandos über die Tastatur eingegeben worden seien.

Somit lassen sich automatisch Kommandos oder spezielle Aufgaben, wie z.B. Speicherlöschungen, das Laden spezieller Programme usw. ausführen.

Der Parameter **<Befehlsstring>** darf bis zu 32 Zeichen aufweisen. Dieser String überschreibt den gesamten bisher im Tastaturpuffer stehenden Inhalt.

BEISPIELE :

```
:
:
200:KBUFF$= "45"
210:INPUT A
220:PRINT "A = ";A
:
:
```

Zeile 200 schreibt die Zahl 45 in den Tastatur puffer.

Zeile 210 wartet mit dem INPUT-Befehl auf die Eingabe eines numerischen Wertes. Da er bereits im Tastaturpuffer steht, nimmt der INPUT-Befehl diesen Wert, so als sei er gerade über die Tastatur eingetippt worden. Da mit dem KBUFF$-Kommando aber kein "carriage return" an den Puffer übergeben wurde, wartet die INPUTAnweisung darauf, daß man die ENTER Taste bebetätigt.

Zeile 220 gibt zur Kontrolle den so eingelesenen Wert nochmals aus.

Um auch ein "carriage return" (Code &OD) oder andere nichtdarstellbare Zeichen in den Tastaturpuffer schreiben zu können, ist die Funktion CHR$ zu verwenden.

```
10:PAUSE "ENDLOSER SELBSTAUFRUF"
20:KBUFF$="RUN"+CHR$(&D)
```

KEY ON/OFF/STOP

Siehe auch : ON KEY GOSUB

WIRKUNG Mit KEY ON/OFF/STOP kann die Verzweigung in eine Interrupt-Routine, die aufgrund der Betätigung der Funktionstasten gewünscht wird, verhindert oder erlaubt werden.

HINWEISE : Mit diesen Anweisungen kann also die Wirkung von ON KEY GOSUB beeinflußt werden.

KEY <Tasten-Nr.> 0FF
hat zur Folge, daß in der Anweisung ON KEY GOSUB die genannte Funktionstaste unbeachtet bleibt.

KEY <Tasten-Nr.> 0N
bewirkt, daß die spezifizierte Taste von der Anweisung ON KEY GOSUB beachtet wird.

KEY <Tasten-Nr.> STOP
verhindert, wie bei der OFF-Option, ebenfalls das Wirksamwerden der angegebenen Funktionstaste im Zusammenhang mit der Anweisung ON KEY GOSUB. Sobald im Programm jedoch die Ausführung der Anweisung KEY <Tasten-Nr.> 0N erfolgt, wird der bis dahin verweigerte Interrupt angenommen und ein Sprung gemäß der Verzweigungsvereinbarung durchgeführt. STOP ist die standardmäßig angenommene Option.

Eine Funktionstaste, die der Anweisung ON KEY GOSUB zugeordnet worden ist, kann nicht mehr als normale Funktionstaste dienen und vordefinierte Strings liefern.

BEISPIEL : siehe ON KEY GOSUB

KEYSTAT

Siehe auch : INKEY$

WIRKUNG : KEYSTAT bestimmt, woher die Eingabe von Befehlen standardmäßig erwartet wird, und ob ein Klickton zu hören ist, wenn man die Tasten betätigt. Ebenso bestimmt KEYSTAT, ob die Tasten eine Wiederholfunktion haben oder nicht.

HINWEISE : **<Parameter** 1) bestimmt die Wiederholfunktion:

0	Wiederholfunktion AUS
1	Wiederholfunktion AN

<Parameter 2> bestimmt die Klickfunktion:

0	Klickfunktion AUS
1	Klickfunktion AN

Nach Einschaltung des Computers gilt standardmäßig die Einstellung:

KEYSTAT 0,0,0.

BEISPIEL :
```
>
KEYSTAT,1,1
```

KILL

Siehe auch : SAVE, SET

WIRKUNG : KILL löscht Dateien, die sich auf Diskette oder einer RAM-Disk befinden.

Der **<Dateibezeichner>** bestimmt welche Datei von welchem Speichermedium zu löschen ist. In diesem Bezeichner muß die Extension angegeben sein. Bei BASIC-Dateien ist also immer die Extension .BAS an den Dateinamen anzuhängen.

Die Benutzung von mehrdeutigen Namen unter Verwendung sogenannter "wildcards" (* oder ?) ist nicht möglich.

Ist die zu löschende Datei mit dem Attribut "P" (siehe SET-Befehl) geschützt, oder befindet sich der Schreibschutzschalter des RAM-Modules in der Stellung ON, wird ein ERROR-Code ausgegeben. Eine geöffnete Datei ist nicht löschbar und muß deshalb zuvor CLOSE geschlossen werden.

BEISPIEL :
```
>
KILL "S1:DATEI1"
```

Diese Anweisung löscht die Datei DATEI1 von der RAM-Disk des Modalfaches S1.

LCURSOR

siehe auch : GLCURSOR,PCONSOLE,TAB

WIRXUNG : LCURSOR setzt den Druckstift an eine bestimmbare Text-Spalte, vorausgesetzt, der Drucker befindet sich im Text-Modus,

HINWEISE : Das LCURSOR-Kommando kann in seiner Wirkung mit dem LCURSOR-Kommando verglichen werden, jedoch bewegt es den Druckstift nicht an jede beliebige X-Position, sondern nur an eine solche, bei der mit der Darstellung eines Zeichens im Text-Modus begonnen werden kann.

Der Parameter <Spalte> bestimmt also, an welche Zeichen-Position bzw. Text-Spalte der Druckstift gesetzt wird. Der Wert für diesen Parameter darf zwischen 0 und einem Maximalwert liegen, der um den Betrag 1 geringer ist als die über PCONSOLE spezifizierte maximale Zeilenlänge.

BEISPIEL :

```
10:LCURSOR 40
20:LPRINT "MITTE"
30:LPRINT "LINKS"
40:END
```

Dieses Programm läuft nur im Text-Modus.

LEFT$

Siehe auch : MID$, RIGHT$

WIRXUNG : Die Funktion LEFT$ liefert den linksbündigen Teil eines vorgegebenen Strings, wobei bestimmt werden kann, wieviele Zeichen von rechts aus dem String abgelesen werden.

HINWEISE : Die (Anzahl> der Zeichen des Teilstrings muß im Bereich von 0 bis 80 liegen. Gibt man die Anzahl als gebrochene Zahl an, wird sie zur nächsten ganzen Zahl hin gerundet. Ist die Anzahl größer als die Zeichenanzahl des vorgegebenen Strings, wird der gesamte vorgegebene String geliefert.

BEISPIEL :

```
10:X$="SHARP PC-1600"
20:FOR N=1 TO 14
30:TS=LEFT$(X$,N)
40:PAUSE T$
50:NEXT N

>
RUN
S
SH
SHA
SHAR
SHARP
SHARP
SHARP P
SHARP PC
SHARP PC-
SHARP PC-1
SHARP PC-16
SHARP PC-160
SHARP PC-1600
SHARP PC-1600
```

LEN

WIRKUNG : LEN ermittelt die Länge eines Strings, d.h. die Anzahl der in ihm enthaltenen Zeichen.

ANMERKUNG : Diese Anzahl berücksichtigt auch solche Zeichen, die einem Leerzeichen (space) entsprechen oder aber zu den nicht darstellbaren Zeichen, wie den Steuercodes (control codes) gehören. Ein solches Steuerzeichen könnte z.B. ein "carriege return" (Sysmbol: <CR>, Code: &OD) sein.

BEISPIELE : Beispiel 1

```
10:INPUT "GIB EIN WORT EIN :",W$
20:N=LEN(W$)
30:PRINT "DAS WORT HAT";N;" BUCHSTABEN"
40:END
```

Beachten Sie was passiert, wenn die Wörter aus mehr als 16 Zeichen bestehen.

Beispiel 2

```
10:A$="EINS";B$="ZWEI";C$="DREI"
20:S$=A$+CHR$(13)+B$+CHR$7+C$
30:PAUSE S$
40:PRINT "ANZAHL DER ZEICHEN = ";LEN S$
50:END

>
RUN
EINS ZWEI DREI
ANZAHL DER ZEICHEN = 14
>
```

LET

WIRKUNG : LET weist Variablen Werte zu.

HINWEISE : Numerischen Variablen lassen sich nur numerische Werte zuweisen und Stringvariablen nur Strings.

Das Befehlswort LET ist optional und kann somit auch weggelassen werden. Damit sind die beiden folgenden Zuweisungen identisch:

```
LET A=5 oder einfach: A=5
```

Sind Wertzuweisungen hinter den Befehlswörtern THEN und ELSE erwünscht, so muß das Befehlswort LET verwendet werden, wenn eine solche Zuweisung direkt hinter dem Befehlswort erfolgt:

```
IF A=B THEN LET P=1 ELSE LET P=0
```

Erfolgt diese Zuweisung nicht direkt hinter dem Befehlswort THEN oder ELSE, weil sie an zweiter oder noch weiterer Stelle innerhalb einer Mehrfachanweisung steht, so kann LET entfallen.

```
IF A=B THEN BEEP 5: P=1 ELSE BEEP 1:P=0
```

BEISPIEL :

```
lO:LET XS="1. STRING"
20:Y$="2. STRING"
30:PRINT X$: PRINT Y$
40:LET A=6
50:B=A+1
60:LET R$=LEFTS(X$,A),S$=LEFT$(Y$,B)
70:PRINT R$;S$
80:END
```

Siehe auch : CSIZE

WIRKUNG : LF bewegt das im Drucker eingespannte Papier um die gewünschte Zeilenanzahl vorwärts oder rückwärts.

HINWEISE : Ohne Parameter bewegt das Kommando LF das Papier im Text-Modus um eine Zeile vorwärts.

<Anzahl> bestimmt, wenn dieser Parameter einen positiven Wert aufweist, um wieviele Zeilen das Papier vorwärtsbewegt wird. Ist dieser Wert negativ, bestimmt er entsprechend, um wieviele Zeilen das Papier rückwärts zu bewegen ist. Der Parameter <Anzahl> kann ein beliebiger numerischer Ausdruck sein. Von Werten, die nicht ganzzahlig sind, werden die Nachkommastellen abgeschnitten und nur der ganzzahlige Anteil berücksichtigt.

Der bei einer Vorwärts- oder Rückwärtsbewegung eingehaltene Zeilenabstand hängt von der mit CSIZE vorgenommenen Einstellung der Zeichengröße ab.

BEISPIEL :

```
10: LPRINT " ZEILE"
20:LF 2
30:LPRINT "4. ZEILE"
40:LF-4
50:LPRINT "1."
60:LF(7)
70:END
```

LFILES

Siehe auch : FILES, SETDEV

WIRKUNG : LFILES gibt ein Inhaltsverzeichnis der näher zu spezifizierenden Dateien eines Speichermediums über einen Drucker oder ein serielles Interface aus .

HINWEISE : Die Parameter von LFILES entsprechen denen vom Befehl FILES. Der Unterschied zu FILES besteht also nur darin, daß die Informationen nicht zum Display, sondern zum Drucker oder zu einer der beiden seriellen Ports geleitet werden.

Wohin die Daten geleitet werden, hängt von der über SETDEV spezifizierten Einstellung ab. Wurde SETDEV nicht benutzt, erfolgt standardmäßig die Ausgabe über den Drucker.

BEISPIELE :

```
>
LFILES "X:"
```

Diese Anweisung listet alle Dateien der mit X: angesprochenen Diskette auf.

```
>
LFILES "S2:*.BIN"
```

Diese Anweisung listet nur jene Dateien der mit S2: selektierten RAM-Disk(ette) auf, die mit der Extension BIN behaftet sind.

LINE

Siehe auch : LLINE

WIRKUNG : LINE zeichnet eine Gerade (Linie) zwischen zwei Display-Punkten.

Die gezeichnete Linie erstreckt sich dabei vom 1. **<Punkt>** mit den Koordinaten (X1,Yl) bis zum 2. **<Punkt>** mit den Koordinaten (X2,Y2) des als Punktmatrix aufgefaßten Displays.

Der Ursprung (0,0) des zugrundegelegten Koordinatensystemes befindet sich dabei in der linken oberen Display-Ecke.

Wird die Angabe des 1. Punktes weggelassen, so wird hierfür die momentane Position des GrafikCursors angenommen.

<Attribut> bestimmt, welchen Einfluß die Linie auf die Display-Punkte hat, die von dieser berührt werden:

S	Alle im **<Bitmuster>** gesetzten Bits steuern die berührten Display-Punkte dunkel, alle gelöschten Bits steuern diese hell.
R	Alle im **<Bitmuster>** gesetzten Bits steuern die berührten Display-Punkte hell, alle gelöschten Bits steuerndiese dunkel. Damit wirkt R also in entgegengesetzter Weise zu S.
X	Invertiert alle von der verbindungs linie berührten Display-Punkte.

Fehlt die Angabe des Attributes, wird für dieses standardmäßig die Option S angenommen.

<Bitmuster> ist ein 16-Bit-Wert, liegt also im Bereich von 0 bis 65535 und bestimmt das Aussehen der Linie.

Beispiele: 65535 (&FFFF) liefert eine durohgehende Linie:

Binär-Wert von &FFFF = 1111111111111111
Die Linienstruktur ist: xxxxxxxxxxxxxxxx

43690 (&AAAA) liefert eine gepunktete Linie:

Binär-Wert von &AAAA = 1010101010101010
Die Linienstruktur ist: x x x x x x x x

26214 (&6666) liefert eine gestrichelte Linie:

Binär-Wert von &6666 = 0110011001100110
Die Linienstruktur ist: xx xx xx xx

Fehlt der Parameter <Bitmuster>,wird eine durchgehende Linie gezeichnet.

B Zeichnet ein Rechteck, wobei die an gegebenen Punkte als diagonal gegen überliegende Eckpunkte des Rechtecks angesehen werden.

BF Zeichnet ebenfalls ein Rechteck, wie bei der Option B, füllt es jedoch mi dem <Bitmuster> aus.

Die X- und Y-Koordinaten eines Punktes können im Bereich -32768....32767 angegeben werden, obwohl nur die Werte 0...155 für X und 0...31 für Y im sichtbaren Bereich des Displays liegen.

BEISPIEL :

```
10:CLS
20:FOR N=10 TO 100 STEP 30
30:M=N+20
40:LINE (N,10)-(M,20),,,BF
50:NEXT N
60:END
```

LIST

Siehe auch : LLIST

WIRKUNG : LIST zeigt die Zeilen eines BASIC-Programmes auf dem Display an.

HINWEISE : Ohne Angabe einer <Zeilen-Nr.> beginnt LIST die zu erstellende Auflistung mit der ersten im Programm enthaltenen Zeile. Sie wird in der oberen Display-Zeile dargestellt. Soweit es der Platz zuläßt, zeigt das Display auch die nachfolgenden Programmzeilen an. Der Cursor wird unsichtbar hinter der ersten Zeilennummer positioniert und durch einen blinkenden Doppelpunkt kenntlich gemacht.

Alle weiteren Zeilen können dadurch zur Anzeige gebracht werden, indem man mit der Taste m den Cursor nach unten bewegt und den Display-Inhalt nach oben hin wegrollen (scrollen) 1äßt.

Wird LIST der Parameter <Zeilen-Nr.> beigefügt, so beginnt die Auflistung eben mit dieser Zeile. Existiert im Programm keine Zeile dieser Nummer, so wird die Auflistung mit der Zeile begonnen, die die nächsthöhere Nummer aufweist. Liegt die <Zeilen-Nr.> über der höchsten im Programm vorkommenden Zeilennummer, so wird ein ERROR-Code angezeigt. Anstelle der <Zeilen-Nr.> ist auch eine existierende <Marke> angebbar.

Ist das Programm mit der "P"-Option des Befehles SET geschützt, wird bei einem LIST-Versuch ebenfalls ein ERROR-Code generiert.

Ein Programm, daz über PASS mit einem Kennwort geschützt ist, läßt sich nicht auflisten, da der Zugang zum PRO-Modus in diesem Falle verwehrt ist, der LIST-Befehl aber nur in diesem Modus akzeptiert wird.

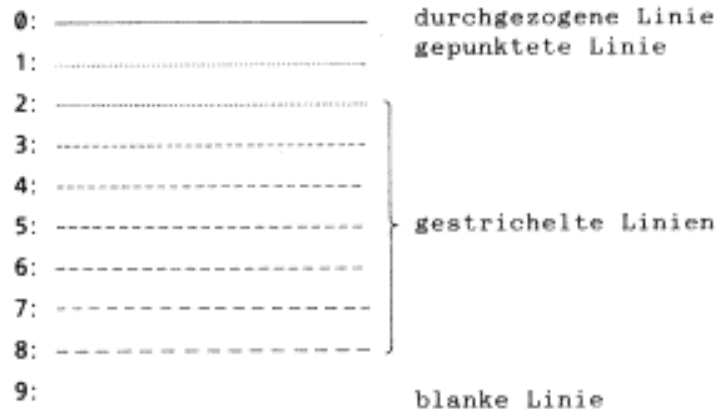
LLINE

Siehe auch : COLOR, RLINE, SORGN

WIRKUNG : LLINE zeichnet eine oder eine Reihe von Linien zwischen den jeweils definierten Punkten des im Drucker eingespannten Papieres.

HINWEISE : Im Grafik-Modus zeichnet LLINE eine vom <Punkt) (X1,Y1) (oder wenn diese Angabe fehlt: von der momentanen Stiftposition) ausgehende Gerade zum <Punkt> (X2,Y2). Die X- und Y-Koordinaten müssen jeweils im Bereich -2048...2047 liegen. Sie sind absolut, d.h., sie beziehen sich auf den mittels SORGN festgelegten Koordinatenursprung (0,0). In der LLINE-Anweisung können bis zu fünf zusammenhängende Linien vereinbart werden.

<Typ> bestimmt die aus 9 verschiedenen Musterngewünschte Linienstruktur:



Fehlt der Parameter <Typ>, bleibt die zuletzt vorgenommene Einstellung erhalten.

Wird LLINE unmittelbar nach einem LPRINT-Befehl im Grafik-Modus ausgeführt, ist in diesem Falle der Parameter <Typ> unbedingt mit anzugeben, da hier kein Standardwert angenommen wird. Dieses betrifft auch übernommene PC-1500-Programme, die LINE-Anweisungen enthalten.

<Farbe> bestimmt den zu benutzenden Farbstift. Es stehen vier Farbstifte zur Auswahl:

0	schwarz
1	blau
2	grün
3	rot

Bei fehlender Angabe des Parameters <Farbe> wird der momentan selektierte Stift genommen.

Die Angabe der Option B sorgt für das Zeichnen eines Rechteckes, das die beiden Punkte (X1,Y1) und (X2,Y2) als Diagonale hat.

Die Anweisung LLINE (X1,Y1)-(X2,Y2),,,B ergibt also folgendes Bild:

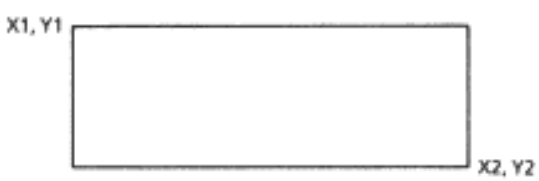


Abbildung 42

BEISPIEL :

```
10:GRAPH
20:GLCURSOR (40,40)
30:SORGN
40:LLINE -(100,0)-(0,100)-(0,0)
50:TEXT
60:END
```

LLIST/LLIST*

Siehe auch : LIST

WIRKUNG : LLIST listet die im Speicher befindlichen Zeilen eines BASIC-Programmes mit Hilfe eines Druckers auf Papier auf oder sendet diese über eines der beiden seriellen Ports aus.

LLIST wird zwar in ähnlicher Weise verwendet wie der Befehl LIST, ist jedoch flexibler in seinen möglichen Parameterangaben.

Ausgabe zum Drucker:

LLIST
sendet das komplette Programm, also alle Zeilen des Programmes zum Drucker.

LLIST <Zeilen-Nr.>
bringt nur die gewÜnschte Zeile zu Papier.

LLIST <Zeilen-Nr.>,<Zeilen-Nr.>
beginnt die Auflistung mit der Zeile der zuerst angegebenen Zeilennummer und beendet diese mit der Zeile der als zweites angegebenen Zeilennummer.

LLIST <Zeilen-Nr.>,
beginnt die Auflistung mit der spezifizierten Zeile und setzt diese bis hin zum Programmende fort.

LLIST ,<Zeilen-Nr.>
beginnt die Auflistung mit der ersten Programm zeile und beendet sie mit Ausdruck der angegebe nen Zeile.

Wird LLIST im Zusammenhang mit dem Drucker verwendet, so berücksichtigt dieser die Einstellung der mit PCONSOLE bestimmten Zeilenlänge. Ist sie größer oder gleich 18 Zeichen gesetzt, werden alle Zeilen, die über diese Länge hinausgehen, in der nächsten Druckzeile fortgesetzt. Ist sie jedoch zu 16 oder 17 bestimmt worden, erzeugen alle längeren Zeilen die Anzeige von ERROR 76. In diesem Falle unterbleibt der Ausdruck der betreffenden Programmzeile.

Ist mit CSIZE die <Größe> der Zeichen mit dem Wert 1 festgelegt, so erfolgt der Ausdruck in dieser Zeichengröße. Bei allen anderen Werten werden die Zeichen in der Größe abgebildet, als sei CSIZE 2 eingestellt worden.

LLIST und LLIST' schalten automatisch in den Text-Modus um.

Vor dem Ausdruck jeder Zeile wird der mit dem Befehl PCONSOLE bestimmte <Heftrand> beachtet.

Ausgabe über einen seriellen Port:

Ist mit dem Befehl SETDEV und der Option PO ein serieller Port für eine Datenausgabe geöffnet worden, sendet LLIST alle Programmzeilen nun zu dem betreffenden Port anstatt zum Drucker.

Jede Zeile enthält die mit PCONSOLE festgelegte Zeichenanzahl und wird mit dem von PCONSOLE bestimmten "end of line code" beendet.

Ausgabe über Drucker und Interface:

Ist das Programm mit einem Kennwort mittels PASS geschützt, werden LLIST und LLIST* ignoriert.

LLIST* listet nur diejenigen Programmzeilen auf, die mit einem Apostroph (‘) beginnen. Alle Kommentare, die mit dem Befehlswort REM gekennzeichnet sind oder nicht am Zeilenanfang stehen, werden von LLIST* nicht berücksichtigt.

LLIST/LLIST*

Mit LLIST* lassen sich alle wichtigen in einem Programm befindlichen Kommentare herausfiltern. Endet eine mit Apostroph beginnende Kommentarzeile mit einem Semikolon, so wird der nächste Kommentar in derselben Druckzeile angefügt.

Benutzt man LLIST* im Zusammenhang mit einem seriellen Port, so we'rden alle Programmzeilen, die ab 100 aufwärts numeriert sind, so in ihrem Inhalt abgeschnitten, daß sie in den Puffer passen. Um unerwünschte Beschneidungen zu vermeiden, sollten deshalb nur Zeilennummern bis einschließlich 99 verwendet werden.

BEISPIEL :

```
10:REM TEST-PROGRAMM
20:'1. KOMMENTAR ;
30:PRINT "HALLO, ";
40:'2. KOMENTAR
50:PRINT "HIER "; ,INGNORIERTER KOMMENTAR
60:PRINT "IST DER PC-1600 !"
70:REM ENDE DES PROGRAMMES
80:END

>
LLIST 20,60

20:,1. KOMMENTAR ;
30:PRINT "HALLO, ";
40:,2. KOMENTAR
50:PRINT "HIER "; 'INGNORIERTER KOMMENTAR
60:PRINT "IST DER PC-1600 !"

>
LLIST*

1. Komentar  2. Komentar
```

LN

WIRKUNG : Die Funktion LN(X) liefert den natürlichen Logarithmus des Argumentes X.

Die Basis dieser Lograrithmus-Funktion ist die Zahl e. Die Funktion LN ist die Umkehrung der Funktion EXP.

A1s Argument ist jeder beliebige numerische Ausdruck erlaubt, sofern sein Resultat innerhalb des zulässigen Wertebereiches liegt.

Das Argument muß größer oder gleich 1E-99 sein. Werte, die darunter liegen, bewirken die Anzeige des ERROR-Codes 39.

BEISPIEL :

```
10:CLS: INPUT "Argument = ";X
20:PRINT "Der Logarithmus zur Basis"
30:PRINT "e lautet: ";LN(X)
40:INPUT "Weitere Berechnung (J/N)";A$
50:IF A$="J" THEN 10
60:END
```

LOAD/LOAD*

Siehe auch : CHAIN, LLIST*, MERGE, BEM, RUN, SAVE

WIRKUNG : LOAD lädt eine Datei in den internen Speicher, die sich auf einer Cassette, Diskette oder RAMDisk befindet oder aber über einen der beiden seriellen Ports geliefert wird.

HINWEISE : Im **<Dateibezeichner>** kann die **<Datenquelle>** aus folgenden Medien auswählen:

S1: RAM-Disk des Modulfaches S1
S2: RAM-Disk des Modulfaches S2
X:, Y: Diskette
COM1: RS-232C-Interface
COM2: SIO-Interface
CAS: Cassette

Ist dem Ladebefehl LOAD die Option R beigefügt, wird die Datei geladen und das in ihr enthaltene BASIC-Programm automatisch gestartet. Besteht die Datei nicht aus einem Programm, erfolgt die Anzeige eines ERROR-Codes.

Alle offenen Dateien, die nicht zur mit LOAD geladenen gehören, werden durch LOAD geschlossen, sofern nicht die R-Option angegegben ist.

LOAD* setzt an den Beginn jeder geladenen Zeile eine Zeilennummer mit nachfolgendem Apostroph. Dies hat den Effekt, daß im Speicher ein BASICProgramm entsteht, das sich ausschließlich aus Kommentarzeilen zusammensetzt. Die Numerierung der Zeilen beginnt mit 10 und wird im Zehnerabstand fortgesetzt.

Durch Anwendung von LOAD$ ist es möglich, ASCII-Dateien in den Arbeitsspeicher zu laden. Dieses ist der einzige Weg, mit dem PC-1600 auf dieser Ebene Textdateien zu verarbeiten. Eine solche Textdatei kann mit LLIST* ausgedruckt werden.

Eine Datei mit folgenden Text wird mittels LOAD* so im Speicher abgelegt:

```
10 ‘ LOAD* ERLAUBT ES, AUF
20 ‘ BASIC-EBENE TEXTE ZU
30 ‘ LADEN UND AUF DEM
40 ‘ DISPLAY SICHTBAR ZU
50 ‘ MACHEN. DIES REICHT,
60 ‘ UM EINFACHE NOTIZEN
70 ‘ ODER ADRESSEN AUFNEH-
80 ‘ MEN ZU KOENNEN.
```

BEISPIEL :

```
>
LOAD "BIOCALC,”,R
```

Diese Anweisung lädt von der momentan gültigen Datenquelle das Programm BIOCALC und startet es anschließend automatisch.

LOC

WIRKUNG : LOC gibt die Anzahl der bisher gelesenen oder geschriebenen Datensätze der mit <Dateinummer> spezifizierten Datei an.

HINWEISE : LOC ist nur im Zusammenhang mit Dateien verwendbar, die sich auf einer Diskette oder einer RAMDisk befinden.

Mit LOC kann der Programmablauf in Abhängigkeit von der Anzahl der gelesenen oder geschriebenen Datensätze gesteuert werden, solange die Datei geöffnet ist.

BEISPIEL :

```
10:OPEN "X:DATEI1" FOR INPUT AS #1
20:IF EOF(1) THEN 50
30:INPUT #1,N
40:GOTO 20
50:M=LOC(1)
60:PRINT "DIE DATEI BESTEHT AUS"
70:PRINT M;" DATENSÄTZEN"
80:CLOSE #1
90:END
```

Zeile 10 öffnete die Datei DATEI1 zum Lesen. Zeile 20 überprüft, ob das Dateiende erreicht ist und verzweigt gegebenenfalls zu Zeile 50.

Zeile 30 liest einen Datensatz, der hier nur aus einem numerischen Wert besteht. Zeile 40 bereitet einen weiteren Lesevorgang vor.

Zeile 50 übergibt an die Variable M die Anzahl der insgesamt gelesenen Datensätze. Zeile 70 zeigt diese Anzahl an und

Zeile 80 schließt die gelesene Datei. Zeile 90 beendet das Programm.

LOCK/UNLOCK

WIRKUNG : LOCK schaltet die Funktionsfähigkeit der Taste <MODE> ab. UNLOCK schaltet sie wieder ein.

HINWEISE : Mit LOCK kann verhindert werden, daß durch die versehentliche Berührung der <MODE>-Taste in einen verkehrten Betriebsmodus geschaltet wird. Trotz Betätigung dieser Taste behält der Computer den derzeit aktiven Modus bei.

Mit UNLOCK kann die <MODE> Taste wieder funktionsfähig gemacht und der Computer in einen anderen Modus geschaltet werden.

LOF

WIRKUNG : LOF gibt die Anzahl der Bytes an, aus denen eine Datei besteht.

HINWEISE : LOF ermittelt die Crößе von Dateien, die sich entweder auf einer Diskette oder einer RAM-Disk befinden.

Dazu muß die Datei über den OPEN-Befehl geöffnet und von diesem mit einer <Dateinummer> versehen worden sein.

Über diese **<Dateinummer>** ist die Datei, deren Größe ermittelt werden soll, anzusprechen.

BEISPIEL :

```
10:OPEN "X:DATEI1" FOR INPUT AS #1
20:N=LOF(1)
30:PRINT "DATEI1 BESTEHT AUS ";N
40:PRINT "BYTES."
50:CLOSE #1
60:END
```

LOG

Siehe auch : LN

WIRKUNG : Die Funktion LOG liefert den dekadischen Logarithmus des angegebenen Argumentes.

HINWEISE : Um einen Logarithmus zu einer anderen Basis als 10 zu erhalten, z.B. zur beliebigen Basis B, ist die folgende Formel zu verwenden:

```
LOG(X)/LOG(B)
```

Die Umkehrung des Zehnerlogarithmus kann mit der Hilfe des Potenzoperators ^ gebildet werden, wenn man als Potenzbasis die Zahl 10 wählt.

BEISPIEL :

```
>
LOG (2)

             3.010299957E-01
>
```

LPRINT

Siehe auch : PCONSOLE, PRINT, PZONE, TAB

WIRKUNG : LPRINT gibt Daten über einen Drucker oder über eine serielle Schnittstelle aus.

Die Anweisungen LPRINT und LPRINT USING werden dabei in gleicher Weise verwendet wie PRINT und PRINT USING.

Ist über SETDEV ein serieller Port selektiert, gibt LPRINT die auszugebenden Daten nicht an den Drucker, sondern an dieses Interface weiter.

Wenn im folgenden einmal nur von dem Drucker die Rede ist, denken Sie bitte daran, daß die Datenausgabe auch für die Ports gelten kann, obwohl wenn es nicht ausdrücklich hervorgehoben ist.

LPRINT ohne Parameter sorgt für den Vorschub des Papieres um eine Zeile.

LPRINT mit Parameterangabe sendet die Werte der aufgelisteten Ausdrücke nacheinander aus. Diese Ausdrücke können sowohl numerisch sein oder auch einem String entsprechen. Wird ein Semikolon zur Trennung der Ausdrücke verwendet, so werden ihre Werte unmittelbar hintereinander ausgegeben. Ist a1s Trennzeichen ein Komma gesetzt, so wird der Wert des nachfolgenden Ausdruckes in die nächste durch PZONE bestimmte Position gedruckt.

Endet die Liste der Ausdrücke mit einem Semikolon, wird die folgende LPRINT-Anweisung an der nächsten mit PZONE vereinbarten Position fortgesetzt. Schließt kein Semikolon die besagte Liste ab, wird ein Zeilenvorschub ausgegeben.

LPRINT USING verhält sich in Analogie zu PRINT USING, so daß Sie hierzu auf die dort gemachten Beschreibungen zurückgreifen können.

Die TAB-Option spezifiziert, in welcher Spalte der Druck der nächsten Ausgabe erfolgen soll. Überschreitet die Spaltenangabe den Wert der mit PCONSOLE festgelegten Zeilenlänge, erfolgt die Anzeige eines ERROR-Codes.

Bei Verwendung von LPRINT im Grafik-Modus wird kein Zeilenende-Code (EOL-Code) wie <CR> (&OD) oder <LF> (&OA) am Zeilenende ausgegeben.

MAXFILES

Siehe auch : CLOSE, OPEN

WIRKUNG : MAXFILES bestimmt die Anzahl der Dateien, die gleichzeitig geöffnet sein dürfen.

HINWEISE : Der maximale Wert für **<Dateianzahl>** beträgt 15.

Alle Dateien müssen geschlossen sein, bevor das Kommando MAXFILES ausgeführt werden kann. Nach Einschaltung des Computers ist diese Anzahl automatisch zu Null gesetzt. Somit muß MAXFILES nach Einschaltung des Gerätes mindestens einmal aktiviert werden, damit überhaupt ein Zugriff auf Dateien erfolgen kann.

Für jede zu verwaltende Datei werden durch den Befehl MAXFILES 313 Bytes reserviert. Ist nicht genügend Speicherplatz frei, um diese Bereiche für die Verwaltung anlegen zu können, wird ein ERROR-Code angezeigt.

BEISPIEL :
```
>
MAXFILES=3
```

MEM

Siehe auch : STATUS

WIRKUNG : MEM liefert die Anzahl der noch freien Bytes des Arbeitsspeichers.

HINWEISE : Die gelieferte Anzahl schließt dabei den Bereich der Variablen ein. Damit ist MEM gleichbedeutend mit der Anweisung STATUS 0.

MERGE

Siehe auch : CLOAD

WIRKUNG : MERGE lädt ein weiteres BASIC-Programm von der Cassette in den Arbeitsspeicher ohne das bereits vorhandene Programm zu löschen.

HINWEISE : Wird MERGE ohne Parameterangabe verwendet, lädt der PC-1600 das nächste Programm, das er auf der Cassette vorfindet. Mit Parameterangabe wird das mit **<Dateiname>** bestimmte Programm von Cassette in den Speicher geholt.

Das mit MERGE geladene Programm wird an das Ende des bereits im Speicher befindlichen Programmes angefügt. Es darf sogar Zeilennummern aufweisen, die auch in dem schon im Speicher stehenden Programm vorkommen. Dieses ist möglich, weil nach Ausführung von MERGE grundsätzlich nur auf das zuletzt geladene Programm mit RUN und LIST zugegriffen werden kann. Deshalb sollte man alle Programme mit einer Markierung versehen, um mit GOTO <Marke> und LIST <Marke> das gewünschte Programm ansprechen zu können.

Bei Verwendung von MERGE sollten auch die DATA Anweisungen mit einer Markierung versehen sein, damit die Read-Anweisung innerhalb des richtigen Programmes zuHreift. Um dieses sicherzustellen, muß vor dem ersten READ ein RESTORE-Befehl ausgeführt werden, der sich auf diese Markierung bezieht. Beispiel:

```
10:RESTORE "A"
20:READ X,Y,Z
30:PRINT X;Y;Z
40:"A":DATA 3,4,5
```

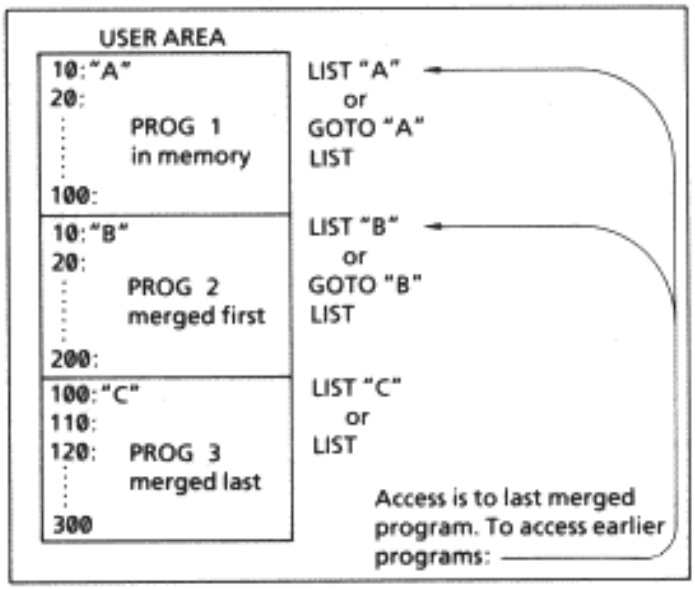


Abbildung E

MID$

Siehe auch : LEFT$, RIGHT$

WIRKUNG : MID$ liefert einen Teilstring, der mitten aus einem vorgegebenen String abzulesen ist.

HINWEISE :

<Position> bestimmt, bei welcher Position des Vorgabe-Strings die Zeichenablesung beginnen soll. Die <Position> kann im Bereich 1...80 liegen. Werte, die sich außerhalb davon befinden, haben die Anzeige eines ERROR-Codes zur Folge. Ist die <Position> größer a1s die Anzahl der im String enthaltenen Zeichen, so wird ein Nullstring geliefert.

<Anzahl> legt fest, wieviele Zeichen von dem Vorgabe-String zu kopieren sind. Sie kann im Bereich 0...80 liegen. Werte mit Nachkommastellen werden auf die nächste ganze Zahl abgerundet.

BEISPIEL :

```
10:Z$="ABCDEFG"
20:LET Y$=MID$(Z$,3,4)
30:PRINT Y$

>
RUN
CDEF
>
```

MOD

Siehe auch : INT

WIRKUNG : MOD ist ein Operator und liefert von den mit ihm verknüpften Werten den Rest einer ganzzahligen Division.

HINWEIS : Die mit MOD verknüpften Werte sollten ganzzahlig sein. Sie dürfen jedoch durch jede numerische Konstante, numerische Variable oder allgemein durch jeden numerischen Ausdruck vertreten sein.

Ist der Wert einer solchen Konstante, Variable oder Ausdrucks nicht ganzzahlig, so wird er vor Ausführung der Division auf einen ganzzahligen Betrag abgerundet.

BEISPIEL :

```
10:INPUT "DIVIDEND = ";N
20:INPUT "DIVISOR = ";M
30:R=N MOD M
40:PRINT "REST VON N/M IST: ";R
50:GOTO 10
```

Benutzen Sie die BREAK-Taste, um dieses Programm zu verlassen.

MODE

WIRKUNG : MODE selektiert den Anzeige-Modus.

HINWEISE : Das MODE-Kommando kann nur im direkten Betriebsmodus ausgeführt werden, nicht jedoch innerhalb eines Programmes.

MODE 0

Mit MODE oder MODE 0 wird der PC-1600 in einen Anzeige-Modus geschaltet, der alle vier Zeilen des Displays ausnutzt. PRINT-Anweisungen zeigen die Daten in aufeinanderfolgenden Display-Zeilen an. Ist die unterste Display-Zeile beschrieben, wird im PRO-Modus oder bei Benutzung der INPUTAnweisung der bisherige Display-Inhalt um eine Zeile nach oben geschoben (scrolling), wenn der Platz für eine weitere Datenanzeige nicht ausreicht. Anzeigen, die nicht in eine Zeile von 26 Zeichen passen, setzen sich in der nächsten Display-Zeile fort. In diesem Modus gilt der PC-1600-Zeichensatz (s. Anhang C).

In MODE 0 kann der PC-1600 mit dem speziell auf ihn zugeschnittenen Drucker CE-1600P oder auch anderer PC-1600-Peripherie verwendet werden.

In diesem Modus können keine Programm, die auf dem PC-1500 erstellt worden sind, abgearbeitet werden. Für diese muß MODE 1 eingestellt sein.

MODE 1

MODE 1 selektiert den PC-1500-kompatiblen Modus. Hierbei wird bei allen Anzeigen nur die unterste Display-Zeile verwendet. Der Zeichensatz wir dem des PC-1500 angepaßt.

Die Länge einer angezeigten Zeile ist auf 26 Zeichen begrenzt. PRINT-Anweisungen, die Daten aufweisen, die nicht mit Z6 Zeichen dargestellt werden können, werden auf die ersten 26 Zeichen gekürzt. Aufeinanderfolgende PRINT-Anweisungen löschen den vorhergehenden Inhalt der untersten Display-Zeile.

Ein "Scrollen" des Display-Inhaltes findet nicht statt, mit Ausnahme bei der Erstellung oder dem Editieren von Programmen sowie der Dateneingabe über den INPUT-Befehl.

Im PRO-Modus werden alle vier Zeilen benutzt.

Der Computer kann in diesem Modus zusammen mit den Peripheriegeräten des PC-1500 verwendet werden.

NAME

Siehe auch : COPY, FILES

WIRKUNG : NAME benennt eine auf Diskette oder auf RAM-Disk befindliche Datei um.

Der **<1.Dateibezeichner>** bestimmt, welche Datei umzubenennen ist. Er muß aus den drei Angaben Speichermedium (hier: Datenquelle), Dateiname und Extension bestehen.

Der **<2.Dateibezeichner>** bestimmt, unter welchem Namen die Datei fortan auf dem bisherigen Medium geführt werden soll und unter welcher Extension. Im <2. Dateibezeichner> muß also die Angabe des Speichermediums mit der vom <1. Dateibezeichner> identisch sein. Sowohl Extension als auch Dateiname können unverändert übernommen werden, wenn mindestens eine Änderung bei einem dieser beiden Bezeichnungen vorkommt. Bleibt die Extension erhalten, kann ihre Angabe im <2. Dateibezeichner> ebenso weggelassen werden wie die Medium-Angabe.

Eine Umbenennung der Datei erfolgt nicht, wenn:

- die Diskette oder RAM-Disk mit einem Schreibschutz versehen ist,

- die Datei mit der P-Option des SET-Kommandos schreibgeschützt ist, oder

- die Datei offen ist.

BEISPIEL :
```
>
NAME "X:ALTEDATEI.XXX" AS "NEUEDATEI.BAS"
```

Siehe auch : DELETE, STATUS, TITLE

WIRKUNG : NEW löscht Programme oder Tastaturbelegungen und reserviert Speicherplatz für MaschinenspracheProgramme.

HINWEISE : Im RESERVE-Modus löscht NEW alle Belegungen der Funktionstasten, und zwar aller drei Ebenen.

Gibt man ein Programm in den Speicher ein, ohne das bisher dort befindliche Programm zu lö~chen, bleiben alle Zeilen mit den Zeilennummern, die in dem neuen Programm nicht vorkommen, weiterhin erhalten. Es entsteht somit ein Programmgemisch, das zu den verschiedensten Ablauffehlern des neuen Programmmes führen kann. Aus diesem Grunde sollte vor Eingabe eines neuen Programmes der Speicher erst einmal vom alten Programm befreit werden. Hierzu dient das Kommando NEW.

PC-1600-Modus (MODE 0)

Der vom Computer adressierbare interne Speicher hat eine Kapazität von 12 KByte = 12288 Bytes. Hiervon stehen dem Anwender maximal 12090 Bytes für eigene Programmzwecke zur Verfügung. Dieser Bereich wird Anwenderbereich genannt und teilt sich in vier Unterbereiche auf:

- Maschinensprache-Bereich
- BASIC-Programm-Bereich
- Freier Bereich
- Variablen-Bereich

Somit ist klar: Je mehr Platz zur Aufnahme von Maschinensprache-Programmen reserviert wird, umso weniger steht einem dann für die Speicherung von BASIC-Programmen zur Verfügung.

NEW
löscht sämtliche BASIC- und MaschinenspracheProgramme, die über TITLE spezifiziert worden sind, beläßt jedoch den Bereich zur Aufahme von Maschinensprache-Programmen in seiner bisherigen Größe bestehen.

NEW O
1öscht sämtliche BASIC- und MaschinenspracheProgramme, die über TITLE spezifiziert worden sind, und setzt die oberste Bereichsadresse für die Maschinensprache-Programme gleich mit der niedrigsten Adresse (197 = &OOC5). Damit steht anschließend kein Platz mehr zur Aufnahme von Maschinensprache-Programmen zur Verfügung,

NEW "Sn:"
löscht alle Programme, die sich in dem durch Sn: bestimmten Speicherbereich befinden:

SO: internes RAM des PC-1600
S1: RAM des im Fach S1 eingesetzten Modules
S2: RAM des im Fach S2 eingesetzten Modules

NEW "Sn:",<Adresse>
löscht entsprechend wie Anweisung NEW "Sn:" und bestimmt mit <Adresse> die oberste Adresse des Maschinensprache-Bereichs, die maximal auf 65535 = &FFFF lauten kann. Die unterste Adresse lautet 197 = &OOC5.

PC-1500-Modus (MODE 1)

NEW
löscht alle Programme des Anwenderbereiches und beläßt den reservierten Maschinensprache-Bereich bei der zuvor gesetzten Größe.

NEW 0
löscht alle Programme des Anwenderbereiches und setzt die oberste Adresse des MaschinenspracheBereichs gleich mit dessen niedrigster Adresse (197 = &00C5). Damit steht kein Speicherplatz für Maschinensprache-Programme zur Verfügung.

NEW <Adresse>
löscht alle Programme des Anwenderbereiches und bestimmt mit <Adresse) die oberste Adresse des Maschinensprache-Bereiches. Die unterste Adresse lautet 197 = &00C5. So ergibt sich bei NEW 1000 beispielsweise folgende Aufteilung:

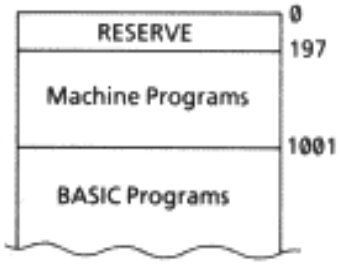


Abbildung F:

BEISPIEL :
```
LIST
10:REM DIESES IST EIN KLEINES
20:REM DREIZEILIGES PROGRAMM.
30: END
>
NEW
LIST
>
```

ON ADIN GOSUB

Siehe auch : ADIN ON/OFF/STOP, AIN , RETI

WIRKUNG : ON ADIN GOSUB verzweigt den Programmablauf in eine Interrupt-Routine, wenn am Analog-Eingang ein vereinbarter Spannungspegel anliegt.

HINWEISE : Die Parameter <Pegel 1> und <Pegel 2> bestimmen den Pegelbereich der analogen Eingangsspannung. Die erlaubten Werte dieser Parameter liegen im Bereich von 0 bis 255. (Abschnitt 6.6 beschreibt den Zusammenhang zwischen diesen Werten und den Spannungspegeln am Analogeingang.)

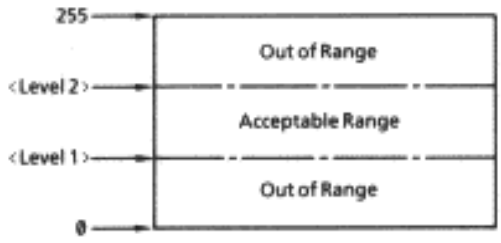


Abbildung 44 : Akzeptierter Pegel-Bereich

Die Wirkung von ON ADIN GOSUB läßt sich mit der Anweisung ADIN ON/OFF/STOP beeinflussen. Wird keine solche Anweisung nach einem ON ADIN GOSUB angegeben, gilt standardmäßig ADIN STOP.

Die Interrupt-Routine muß mit der RETI-Anweisung abgeschlossen sein.

BEISPIEL :
```
20:0N ADIN(100,165) GOSUB 500
:
490:END
500:REM INTERRUPT-ROUTINE
:
600:RETI
```

ON COMn GOSUB

Siehe auch : COMn ON/OFF/STOP, RETI

WIRKUNG : ON COMn GOSUB verzweigt den Programmablauf in eine Interrupt-Routine, wenn über ein Interface eine Interrupt-Anforderung gestellt wird.

HINWEISE : Der Parameter COMn: ist durch COM1: oder COM2: zu konkretisieren und meint mit:

COM1: das RS-232C-Interface
COM2: das SIO-Interface.

Die Wirkung von ON COMn GOSUB läßt sich mit der Anweisung COMn ON/OFF/STOP beeinflussen. Wird keine solche Anweisung gegeben, gilt standardmäßig COMn STOP.

Die Interrupt-Routine muß mit der RETI-Anweisung abgeschlossen sein.

BEISPIEL :

```
20:ON COM1 GOSUB 500
:
490:END
500:REM INTERRUPT-ROUTINE
510:REM FUER RS-232C
:
600:RETI
```

ON ERROR GOTO

Siehe auch : ERL, ERN, RESUME

WIRKUNG : ON ERROR GOTO bewirkt, daß beim Auftreten eines Fehlers die Anzeige des zugehörigen ERROR-Codes verhindert und stattdessen in die spezifizierte Fehlerbehandlungs-Routine gesprungen wird.

HINWEISE : Eine Fehlerbehandlungs-Routine kann vom Anwender des Computers geschrieben werden, um mit dieser festzustellen, in welcher Zeile des Programmes ein Fehler aufgetreten und welcher Art dieser ist. Dann kann innerhalb dieser Routine darüber entschieden werden, welche Maßnahmen weiter zu ergreifen sind: ob z.B. durch eine Korrektur der Fehler behoben werden kann, der Benutzer des Programmes zu einer erneuten Dateneingabe aufgefordert wird und dergleichen mehr.

Soll die Fehlerbehandlungs-Routine das Programm weder durch eine STOP- noch eine END-Anweisung beenden, so muß die Rückkehr in das eigentliche Programm über den Befehl RESUME erfolgen.

Tritt in der Fehlerbehandlungs-Routine selbst ein Fehler auf, wird das Programm abgebrochen und ein ERROR-Code angezeigt.

Es gibt keine Begrenzung, wie oft die Anweisung ON ERROR GOTO innerhalb eines Programmes anwendbar ist. Tritt ein Fehler auf, gilt immer die zuletzt ausgeführte ON ERROR GOTO Anweisung.

ON ERROR GOTO

Mit ON ERROR GOTO 0 kann auf die normale Fehlerbehandlung des BASICs zurückgeschaltet werden. Damit wird beim Auftreten eines Fehlers wieder ein entsprechender ERROR-Code angezeigt und der Ablauf des fehlerhaften Programmes abgebrochen.

Die Wirkung einer ON ERROR COTO Anweisung wird ebenfalls durch die Befehle RUN und END sowie der Betätigung der SHIFT + CL Tasten aufgehoben.

Der Start eines Programmes mit GOTO oder DEF hat keine solche aufhebende Wirkung.

BEISPIEL :

```
5:ON ERROR GOTO 100
10:SAVE "X:DEMO"
20:PRINT "OK"
30:END
:
100:IF ERN =160THEN PRINT
    "KEINE DISKETTE IM LAU
    FWERK"
110:PRINT "BITTE DISKETTE
    EINSETZEN"
120:PRINT "UND TASTE J DRÜ
    CKEN : ";
130:INPUT ""; A$
140:IF A$="J"THEN RESUME
150:STOP
```

ON .. GOSUB/GOTO

Siehe auch : GOSUB..RETURN, GOTO

WIRKUNG : ON...GOSUB und ON...GOTO dienen der bedingten Programmverzweigung in Abhängigkeit des IntegerWertes eines numerischen Ausdruckes.

HINWEISE : Die als Parameter aufgelisteten **<Zeilennummern>** oder **<Marken>** bestimmen jene Programmzeilen, an die nach folgenden Regeln gesprungen wird:

Weist das Verzweigungskriterium **<num. Ausdruck>** den Wert 1 auf, so findet ein Sprung zur ersten Zeile, die in der Parameter-Liste definiert ist, statt. Ergibt der numerische Ausdruck den Wert 2, so wird zur zweiten spezifizierten Zeile gesprungen und so fort. Ist der <num. Ausdruck> mit Nachkommastellen behaftet, wird er auf die nächstniedrigere ganze Zahl (Integer-Wert) abgerundet.

Ist der Wert größer a1s die Anzahl der in der Liste enthaltenen Elemente, wird das Programm mit der nächsten Programmzeile fortgeführt, was auch für jene Fälle gilt, in denen der Wert kleiner als 1 lautet.

BEISPIEL :

```
10:INPUT "NUMMER (1-3) = ";N
20:ON N GOSUB 100,200,300
:
90:END
100:REM ERSTES UNTERPROGRAMM
190:END
200:REM ZWEITES UNTERPROGRAMM
290:END
300:REM DRITTES UNTERPROGRAMM
380:END
```

ON KEY GOSUB

Siehe auch : KEY ON/OFF/STOP, RETI

WIRKUNG : ON KEY GOSUB verzweigt in eine von maximal sechs Interrupt-Routinen, wenn dazu eine Aufforderung über eine der sechs Funktionstasten erfolgt.

HINWEISE : Die Wahl der Funktionstaste bestimmt, welche der sechs Interrupt-Routinen abzuarbeiten ist. Wird die erste Funktionstaste (F1) gedrückt, erfolgt ein Sprung zur ersten Boutine, die in der Liste der Parameter durch eine **<Zeilen-Nr.>** oder eine **<Marke>** spezifiziert ist. Bei der zweiten Taste (F2) wird zu der an zweiter Stelle genannten Interrupt-Routine verzweigt usw. Enthält die Parameterliste mehr als sechs Elemente, werden die überzähligen Elemente ignoriert. Sind in der Aufzählung weniger als sechs Interrupt-Routinen deklariert, bleiben die entsprechenden Tasten wirkungslos.

Die sechs Funktionstasten lösen nur dann einen Interrupt aus, wenn ein Programm läuft und eine KEY ON Anweisung ausgeführt worden ist. Ohne sie gilt standardmäßig KEY STOP, was die Annahme des Interrupts verhindert, aber zwischenspeichert.

Jede Interrupt-Routine muß mit der Anweisung RETI abgeschlossen sein, damit eine RÜckkehr in den normalen Programmablauf möglich ist.

RUN und END löschen sämtliche Zuordnungen von Interrupt Routinen und Funktionstasten, so daß diese Tasten wieder normal belegt sind.

BEISPIEL :

```
10:ON KEY GOSUB 100,200
20:KEY (1)ON :KEY (2)0N
30:PAUSE "$";:GOTO 30
100:FOR I=1TO 6: PAUSE "1"; :NEXT I:RETI
200:FOR J=1TO 6: PAUSE "2"; :NEXT J:RETI
```

ON PHONE GOSUB

WIRKUNG : Mit ON PHONE GOSUB wird der Programmablauf in eine Interrupt-Routine verzweigt, wenn über das RS-232C-Interface eine entsprechende Anforderung an Pin 9 (CI-Signal) anliegt.

HINWEISE : Die Interrupt-Routine muß mit der Anweisung RETI abgeschlossen sein.

Die Wirkung von ON PHONE GOSUB läßt sich mit der Anweisung PHONE ON/OFF/STOP beeinflussen. Ohne eine solche Anweisung wird standardmäßig so verfahren, a1s sei PHONE STOP angegeben worden.

BEISPIEL :

```
10:ON PHONE GOSUB 500
20:PHONE ON
:
190:PHONE OFF
:
250:'PHONE INTERRUPT
:
290:RETI
```

ON TIME$ GOSUB

Siehe auch : RETI, TIME$, TIME$ ON/OFF/STOP

WIRKUNG : ON TIME$ GOSUB verzweigt den Programmablauf in eine Interrupt-Routine, wenn ein vereinbarter Zeitpunkt erreicht ist.

HINWEISE : Die Spezifikation des gewünschten Zeitpunktes erfolgt im selben Format wie bei der Stellung der Uhr über TIME$.

Ist die betreffende Zeit erreicht, wird die Interrupt-Routine, die hinter dem Befehlswort GOSUB durch eine <Zeilen-Nr.> oder eine <Marke> bestimmt ist, aufgerufen.

Sie muß, damit eine Rückkehr in das normale Programm möglich ist, mit der RETI-Anweisung abgeschlossen sein.

Die Anweisung ON TIME$ GOSUB läßt sich mittels TIME$ ON/OFF/STOP in ihrer Wirkung beeinflussen. Standardmäßig wird die Einstellung TIME$ STOP angenommen.

BEISPIEL : 10:ON TIME$="05/12/15/30/" GOSUB 500

OPEN

WIRKUNG : OPEN öffnet eine Datei und erlaubt somit, auf diese zum Zwecke des Lesens, Schreibens oder Anhängens von Daten zugreifen zu können.

HINWEISE : Je nach gewünschter Art des Zugriffes ist dem OPEN-Befehl ein entsprechendes Attribut (INPUT, OUTPUT oder APPEND) beizufügen und die Datei in diesem Modus zu öffnen. Auf die Datei kann dann nur zu genau diesem Zwecke zugegriffen werden. Bevor eine Datei zu einem anderen Zweck geöffnet werden kann, muß sie zuvor mit CLOSE geschlossen werden.

INPUT erlaubt eine Dateiöffnung, um von der Datei Datensätze sequentiell mittels INPUT# lesen zu können.

OUTPUT gestattet eine Öffnung der Datei, um in diese sequentiell Datensätze mit PRINT# hineinschreiben zu können. In diesem Modus können keine Datensätze an die Datei angefügt werden.

APPEND öffnet die Datei in der Weise, daß an ihren bisherigen Inhalt mit Hilfe von PRINT# weitere Datensätze anfügbar sind.

Der Parameter **<Dateibezeichner>** bestimmt, auf welchem Medium die Datei zu finden ist und unter welchem Namen und welcher Extension.

Mit dem Parameter **<Datei-Nr.>** wird der Datei eine Nummer zugewiesen, unter der auf sie mit den weiteren Dateibefehlen zugegriffen werden kann. Diese Nummer darf zwischen 1 und dem über MAXFILES bestimmten Maximum liegen.

OPEN

Die Öffnung einer Datei bleibt erfolglos, wenn:

- der OUTPUT-Modus gewünscht ist, aber die Datei mit einem Schreibschutz versehen ist,

- ein serielles Interface über COM1: bzw. COM2: im APPEND-Modus spezifiziert wird.

BEISPIEL :

```
5:MAXFILES=1
10:OPEN "X:DATA"FOR OUTPUT AS #1
20:FOR J=lTO 5
30:PRINT #1,J
40:NEXT J
50:CLOSE #1
60:OPEN "X:DATA"FOR INPUT AS #1
70:IF EOF(1)THEN 110
80:INPUT #1,J
90:PRINT J
100:GOTO 70
110:REM DATEIENDE ERREICHT
120:CLOSE #1
130:END
```

Zeile 10 eröffnet eine neue Disketten-Datei,so daß in diese Daten hineingeschrieben werden können.
Zeile 30 schreibt die Werte 1 bis 5 direkt auf einanderfolgend in diese Datei hinein.
Zeile 50 schließt die Datei, damit sie gleich zu einem anderen Zweck geöffnet werden kann.
Zeile 60 öffnet die Datei zum Lesen der Daten.
Zeile 70 prüft, ob das Dateiende erreicht ist und verzweigt im zu baejaenden Falle zur Zeile 110.
Zeile 80 liest die Daten.
Zeile 90 zeigt die Daten an.

OUT

Siehe auch : INP

WIRKUNG : OUT gibt ein Byte über den gewünschten Port des Z80-kompatib1en Mikroprozessors aus.

HINWEISE : **<Port>** ist eine Adresse (16-Bit-Wert) im Werte-Bereich von 0...65535 (&0...&FFFF), die den gewünschten Port selektiert.

<Byte> gibt das an den Port zu liefernde Byte an. Werden mehrere Bytes a1s Parameter aufgelistet, so wird jedes folgende Byte an die nächste Port-Adresse abgegeben. Aufeinanderfolgende Bytes werden also an aufeinanderfolgende Ports übergeben.

BEISPIEL : OUT 80,187

Diese Anweisung sendet den Wert 187 = &BB an den Port 80 (=&50).

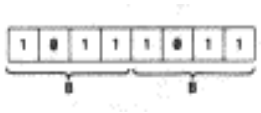


Abbildung G

OUTSTAT

Siehe auch : INSTAT

WIRKUNG : OUTSTAT bestimmt den Zustand der Steuersignale RTS und DTR des RS-232C-Interface.

HINWEISE : Die Zustände der Signale RTS (request to send) und DTR (data terminal ready) werden durch den Parameter <Code> wie folgt bestimmt:

<Code>	RTS	DTR
0	high	high
1	high	low
2	low	high
3	low	low

Fehlt dieser Parameter, bleiben beide Signale während der Ausführung eines Interface-Befehles "high". Davor oder danach werden sie auf "low" gesetzt. Ist der Empfangspuffer gefüllt, so geht RTS auf "low", um damit den Datentransfer zu stoppen und den Puffer entleeren zu können.

BEISPIEL :
```
>
OUTSTAT "COM1:",2
```

Diese Anweisung setzt das RTS-Signal des Ports auf "low" und das DTR-Signal auf "high".

PAPER

WIRKUNG : PAPER teilt dem Computer mit, welches Format das in den Drucker eingespannte Papier aufweist, und bestimmt, in welchem vertikalen Bereich gedruckt werden darf.

HINWEISE : **<Typ>** beschreibt das Papierformat wie folgt:

C = Einzelblattpapier und
R = Endlospapier (Papierrolle).

Die beiden anderen Parameter sind optional und legen den zulässigen Druckbereich in Y-Richtung in Schritten von 0.2mm fest. Fehlen diese Parameter, werden standardmäßige Werte angenommen.

Die mit PAPER definierten Grenzen beziehen sich auf die momentane Stiftposition und werden mit Ausführung dieses Befehles unmittelbar wirksam. Mit Betätigung der am Drucker befindlichen Taste , dem Einschalten des Druckers sowie der Ausführung der Befehle TEXT oder GRAPH sind die Begrenzungen auf die dann gerade vorliegende Stiftposition zu beziehen.

<Limit 1> bestimmt die obere Druckgrenze, also um wieviele Einheiten zu je 0.2mm das Papier rückwärts transportiert werden kann. Hier sind Einheiten zwischen 30 und 2047 erlaubt. Die Standardwerte lauten:

30 (= 6.0mm) für Einzelblatt- und
999 (=199.8mm) für Endlospapier.

<Limit 2> legt die untere Druckgrenze fest, d.h. um wieviele Einheiten zu je 0.2mm das Papier vorwärts bewegbar ist. Auch hier sind für den Parameter Werte von 30 bis 2047 zulässig.

A1s Standardwerte für <Limit 2> gelten folgende:

1354 (=272.8mm) für Einzelblatt- und
999 (=199.8mm) für Endlospapier.

Bei der Ausführung der Befehle GRAPH und TEXT werden die Einstellungen auf die Standardwerte von <Limit 1> und <Limit 2> zurückgesetzt. Im Text-Modus gilt dabei für (Limit 2) der Wert für Endlospapier.

Ein Versuch, im Grafik-Modus den Druckstift über die mit PAPER festgelegten Grenzen hinauszubewegen, spiegelt die Projektion des verhinderten Druckweges an diesen Bereichsgrenzen.

BEISPIEL :

```
>
PAPER C
```

Dieses Kommando bereitet den Drucker auf den Gebrauch von Einzelblatt-Papier im DIN-A4-Format vor. Die Druckgrenzen werden auf die Standardwerte eingestellt.

PASS

Siehe auch : CLOAD, CSAVE, SET

WIRKUNG : Das PASS-Kommando erlaubt es, ein Programm durch die Vergabe eines Kennwortes (password) gegen den unerlaubten Zugriff fremder Personen zu schützen. Ein solches **<Kennwort>** besteht aus bis zu acht beliebig kombinierten Zeichen, die wie eine Stringkonstante in Anführungsstriche einzuklammern sind. Das Anfahrüngszeichen " kann somit nicht innerhalb des Kennwortes verwendet werden.

Nachdem ein Kennwort gesetzt worden ist, kann der Computer nicht mehr in den PRO- oder den RESERVE-Modus versetzt werden. Die nachstehend genannten Befehle bleiben dann ebenso wirkungslos wie die Tasten ↑ und ↓ :

LIST, LLIST, CSAVE, SAVE, CLOAD, LOAD, NEW, TITLE, MERGE und CHAIN.

Damit kann das Programm weder aufgelistet, gespeichert noch verändert werden. Ebenso ist ein Überschreiben durch Laden eines anderen Programmes verhindert. Befinden sich mehrere Programme im Speicher, so gilt diese Schutzwirkung für alle diese Programme. Der einzige Weg, um den Schutz aufzuheben, ist, das PASSKommando mit dem betreffenden Kennwort nochmalig einzugeben.

Das PASS-Kommando ist nur dann anwendbar, wenn sich im Programmspeicher auch tatsächlich ein Programm befindet.

BEISPIEL : PASS "GEHEIM"

Dieses Kommando schützt alle im Speicher vorhandenen Programme durch das Kennwort "GEHEIM".

PAUSE

Siehe auch : PRINT, WAIT

WIRKUNG : PAUSE zeigt die als Parameter angegebenen Daten auf dem Display an und wartet mit der weiteren Programmausführung für eine definierte Zeit von 0.85 Sekunden.

Damit wirkt PAUSE wie eine PRINT-Anweisung, der ein entsprechendes WAIT-Kommando vorausgegangen ist. Die Anwendung von PAUSE erweist sich a1s besonders nützlich, wenn der Computer im PC-1500 kompatiblen MODE 0 betrieben wird. Hier würden sonst alle Ausgaben von PRINT-Anweisungen sofort aus der einen Display-Zeile herauswandern und damit unsichtbar bleiben.

In MODE 1 kann die Anzahl der als Parameter auf-geführten Ausdrücke maxima1 zwei betragen, wenn als Trennzeichen ein Komma verwendet wird.

Da PAUSE und PRINT sich nur in ihrer Anzeigedauer voneinander unterscheiden, verhalten sie "ich auch bezüglich der USING-Anweisung gleich. Näheres kann hierzu den Erklärungen von USING entnommen werden.

BEISPIEL :
```
10:PAUSE "LIES SCHNELL DIESEN SATZ !"
20:CLS
30:END
```

PCONSOLE

WIRKUNG : Mit PCONSOLE kann die Zeilenlänge und der Code für das Zeilenende sowohl für den Drucker als auch die seriellen Interfaces bestimmt werden.

HINWEISE : **Einstellung für den Drucker**

Lautet der erste Parameter "LPT1:", so bezieht sich die mit PCONSOLE vorzunehmende Einstellung auf den derzeitig aktivierten Drucker.

Die **<Zeilenlänge>** bestimmt, wieviele Zeiehen in einer Druckzeile enthalten sein können. Der Wert darf zwischen 16 und 255 liegen. Gibt man jedoch den Wert 0 an, wird damit die Zeilenlänge auf unendlich gesetzt.

Der **<EOL-Code>** oder auch Zeilenend-Code (end of line code) genannt, bestimmt, welche Zeichen zur Kennung des Zeilenendes zum Drucker gesendet werden, wenn die Befehle LLIST, LPRINT oder LFILES zur Ausführung kommen. Es gilt:

<EOL-Code>	Zeichen	Hex-Code(s)
0	<CR>	&OD
1	<LF>	&OA
2	<CR> + <LF>	&OD + &OA

Andere Angaben führen zur Anzeige eines ERRORCodes.

Der **<Heftrand>** legt fest, wieviele Leerzeichen der eigentlichen Ausgabe vorausgeschickt werden sollen, damit ein Rand zum Abheften des Papieres verbleibt. Für diesen Wert muß gelten:

<Heftrand> ≤ <Zeilenlänge> - 4

Die standardmäßig vorausgesetzten Einstellungen lauten:

- unendliche Zeilenlänge
- <EOL-Code> = 0
- (Heftrand> = 0

Falls die mit LPRINT ausgedruckten Zeichen den Stift weniger a1s 4 mm bewegen (z.B. CSIZE 1), kehrt dieser nicht zum linken Rand zurück, wenn ein Zeilenvorschub stattfindet.

Einstellungen für die seriellen Ports

Wird als erster Parameter "COMn:" angegeben, so beziehen sich die dann folgenden Parameter auf eine der beiden seriellen Schnittstellen gemäß:

COM1: RS-232C Interface
COM2: SIO-Interface
COM: über SETDEV selektiertes Interface

Die **<Zeilenlange>** bestimmt in diesem Falle, wieviele Zeichen über den Port ausgesendet werden können, bevor eine Trennung der einzelnen Zeilen mittels <EOL-Code> erfolgt. Auch hier gelten die schon für den Drucker genannten Parameterwerte.

<EOL-Code> bestimmt was auf dem Port auszugeben ist, um das Zeilenende zu signalisieren. Hierbei sind dieselben Einstellungen wie für den Drucker möglich. Der Code bestimmt darüberhinaus, welche Zeichenfolge als einfacher "carriage return" zu interpretieren ist, wenn Daten mit einer INPUTAnweisung über den Port eingelesen werden.

Alle über PCONSOLE vorgenommenen Einstellungen bleiben solange wirksam, bis sie durch ein neues PCONSOLE-Kommando überschrieben werden.

Wird die Zeichengröße mit CZISE variiert, ändert sich die über PCONSOLE definierte Zeilenlänge in entsprechender Proportion. Die Einstellung des Heftrandes wird davon jedoch nicht berührt.

Beispiel : PCONSOLE "LPT1:",42,2,3

PEEK

Siehe auch : POKE, XPEEK, XPOKE

WIRKUNG : PEEK liefert den Inhalt einer spezifizierten Speicheradresse.

HINWEISE :

<Adresse> bestimmt die Adresse im Bereich von &0bis &FFFF (0 bis 65535) bezüglich der jeweils gültigen Speicherbank.

<Bank> beschreibt, auf welche Speicherbank sich die Angabe der Adresse bezieht. Es sind hier die Werte von O bis 7 angebbar. Fehlt der Parameter <Bank>, wird hierfür standarämaßig der Wert 0 angenommen. Näheres zur Aufteilung des Speichers in "memory banks" geht aus Anhang D hervor.

BEISPIEL :

```
>
PEEK (0,100)
```

Dieser Befehl liefert den Inhalt der Speicherzelle 100 (=&64) der Speicherbank O.

PHONE ON/OFF/STOP

Siehe auch : ON PHONE GOSUB

WIRKUNG : Erlaubt oder verbietet dem Computer die Annahme von Interrupt-Anforderungen, die von einem Modem über das RS-232C-Interface gestellt werden.

HINWEISE :

PHONE ON

Mit dem Befehl PHONE ON wir die Annahme solcher Interrupt-Anforderungen zugelassen. Tritt eine Anforderung über das RS-232C-Interface auf kann mit der Anweisung ON PHONE GOSUB in die entsprechende Interrupt-Routine verzweigt werden.

PHONE OFF

Über den Befehl PHONE OFF wird die Annahme der genannten Interrupt-Anforderungen verhindert.

PHONE STOP

Durch den Befehl PHONE STOP wird die Annahme der über das RS-232C-Interface gestellten InterruptAnforderungen ebenfalls verhindert, Der Unterschied zu PHON OFF besteht jedoch darin, daß die jeweils letzte Anforderung zwischengespeichert wird. Bei der nächsten Ausführung eines PHONE ON kommt sie dann unmittelbar zur Geltung. STOP ist die standardmäßig angenommene Einstellung nach Inbetriebnahme des Computers.

BEISPIEL :

```
10:ON PHONE GOSUB 250
20:PHONE ON
:
90:PHONE OFF
```

PITCH

Siehe auch : CSIZE

WIRKUNG : PITCH bestimmt den im Text-Modus vom Drucker einzuhaltenden Zeichen- und Zeilenabstand.

HINWEISE : Der Parameter <X-Abstand> legt den horizontalen Abstand zweier Zeichen fest und <Y-abstand> den vertikalen Abstand, d.h. den Zeilenabstand.

<X-Abstand> Abstand zweier Zeichen zueinander

Standard 6 * CSIZE <Größe> * 0.2mm
4-255 <X-Abstand> * 0.2mm

<Y-Abstand> Abstand zweier Zeilen zueinander

Standard 12 * CSIZE <Größe> * 0.2mm
4-255 <Y-Abstand> * 0.2mm

Fehlt die Angabe eines dieser Parameters, gilt für ihn die Standardeinstellung.

Bei Ausführung der Befehle TEXT, GRAPH, CSIZE, LLIST, TEST und PCONSOLE werden die standardmäßigen Werte für PITCH angenommen. PITCH ist auch im Grafik-Modus verwendbar. Es wird dann allerdings der Parameter <Y-Abstand> ignoriert.

BEISPIEL :

```
10:PITCH 30,40
20:LPRINT "ERSTE ZEILE"
30:LPRINT "ZWEITE ZEILE"
40:LPRINT "DRITTE ZEILE"
50:END
```

Probieren Sie dieses Programm auch mit anderen PITCH-Parameterwerten aus, um ihren Einfluß auf das Druckverhalten kennenzulernen.

POINT

Siehe auch : GPRINT, PRESET, PSET

WIRKUNG : POINT liefert eine Information über den Zustand des spezifizierten Display-Punktes.

HINWEISE : POINT (X,Y)

Im Format POINT (X,Y) bestimmen die Koordinaten X und Y den Display-Punkt. Ist der Punkt dunkel, also gesetzt, so liefert POINT (X,Y) den Wert 1, im anderen Falle den Wert 0. Die Koordinaten können in folgenden Bereichen liegen:

<X-Koordinate>	0 = Anfang der Display-Zeile
	155 = Ende der Display-Zeile
<Y-Koordinate>	0 = oberste Punktreihe
	31 = unterste Punktreihe

Obwohl die <X-Koordinate> und die <Y-Koordinate> in diesen Bereichen liegen sollten, um einen realen Display-Punkt zu adressieren, können sie dennoch jeweils im Bereich -32768...32767 angegeben werden. Das gelieferte Ergebnis lautet in diesen Fällen 0. Dieses gilt auch für das nachfolgend beschriebene Format.

POINT (X)

Im Format POINT (X) wird die momentane Display Zeile in fortlaufende Spalten mit der Breite eines Punktes und der Höhe von acht Punkten aufgeteilt aufgefaßt. X bestimmt die Spalte, von der der Zustand der acht Punkte, als Bitmuster gewertet, geliefert werden soll. Die Zählung beginnt bei Null.

POINT (25) liefert somit den Wert für das in der 26. Punktspalte stehende Bitmuster an. Lautet er beispielsweise 98 (binär: 01100010), so liegt damit das folgende Punktmuster vor:

```
: : 0  niederwertigstes Bit
:X: 1
: : O
: : O
: : O
:X: 1
:X: 1
: : 0  höchstwertigstes Bit
```

Unter den Beschreibungen zu GPRINT finden Sie weitere Details zur Bitmuster-Darstellung.

BEISPIEL :

```
10:CURSOR 0,0
20:P=POINT (3):Q=POINT (4)
30:P$=HEX$ P: Q$=HEX$ Q
40:CURSOR 0,1
50:PRINT P;Q
60:PRINT " "+P$+" "+Q$
```

Starten Sie dieses Programm mit dem Befehlswort RUN, so liefert es die Werte der Bitmuster, die die letzten beiden Punktspalten des Buchstabens R bilden. Es ergibt sich dann folgende Anzeige:

```
RUN
41 70
29 46
>
```

Die erste Zahlenreihe gibt die Werte dezimal, die darunterstehende Zeile hexadezimal an.

POKE

Siehe auch : PEEK, XPEEK, XPOKE

WIRKUNG : Mit POKE besteht ein direkter Zugriff auf die Speicherzellen des Computers. Es können hiermit Daten, die in Form von Bytes vorliegen, gezielt in die spezifizierten RAM-Adressen geschrieben werden.

HINWEISE : **<Adresse>** bestimmt, in welche Speicherzelle das (erste) angegebene Byte zu schreiben ist. Die im Bereich von 0..65535 bzw. &0…&FFFF liegende Adresse bezieht sich dabei auf die derzeit gültige oder die über <Bank> spezifizierte Speicherbank.

<Bank> ist eine ganze Zahl zwischen 0 und 7. Sie legt die gewünschte Speicherbank fest, auf die sich die Adresse beziehen soll. Fehlt dieser Parameter, so gilt diejenige Bank, in der sich der Datei-Kopf des gerade laufenden Programmes befindet. Über die Unterteilung des Speichers in Bänke gibt Anhang D Auskunft.

<Byte> bestimmt einen 8-Bit-Wert im Bereich von 0 bis 255 (bzw.: &0..&FF), der in die durch <Adresse> und <Bank> genau festgelegte Speicherzelle geschrieben werden soll.

Gibt man mehrere Bytes an, die dann mit Kommas voneinander getrennt sein müssen, so werden sie der Reihe nach in aufeinanderfolgende Adressen geschrieben. Der Parameter <Adresse> wirkt dabei als Anfangsadresse.

Reicht der zur Verfügung stehende Speicherplatz nicht aus, um alle aufgelisteten Bytes unterzubringen, so wird ein ERROR-Code angezeigt.

BEISPIEL : Das nachfolgende Beispiel, das nicht nur zur Anwendung des POKE-Befehles, sondern auch der Befehle CALL und PEEK dienen möge, zeigt, wie man ein Maschinensprache-Programm in Z80-Code eingibt, es überprüft und startet.

Als Programm soll eine kleine Routine dienen, die den Wert einer Variablen übernimmt, ihn inkrementiert und an die Variable zurückgibt.

Diese Routine lautet folgendermaßen:

Adresse	Code	Mnemonik
&FF00	&13	INC DE
&FF01	&37	SCF
&FF02	&C9	RET

Sie kann mit nachstehender Anweisung in den Speicher gePOKEd werden:

```
POKE &FF00,&13,&B7,&37,&C9
```

Überprüfen Sie anschließend durch Stichproben, ob diese Werte auch tatsächlich in den Speicher übernommen worden sind. Zum Beispiel:

```
>
HEX$ PEEK &FF02
C9
>
```

Ist alles in Ordnung, kann die Routine gestartet werden. A1s Übergabevariable soll in diesem Beispiel die Variable X dienen, die zuvor mit einem Testwert zu belegen ist:

```
>
CLEAR
X=77
CALL &FF00,X
```

Fragen Sie dann die Variable ab, ob wirklich ihr Wert um den Betrag 1 inkrementiert worden ist:

```
>
X
78
>
```

POWER

WIRKUNG : POWER schaltet die "auto power off"-Funktion ein bzw. aus oder ermöglicht die programmgesteuerte Ausschaltung des Computers.

HINWEISE :

POWER OFF schaltet den Computer unverzüglich aus, wenn dieses Kommando in einem laufenden Programm aktiviert wird.

Im RUN- oder PRO-Modus wird dieses Kommando auch bei Eingabe über die Tastatur wirksam.

POWER AOFF
POWER AOFF 0 Mit diesen beiden Kommandos kann die "auto power off"-Funktion des PC-1600 aktiviert werden. Diese sorgt dafür, daß sich der Computer nach ca. 10 Minuten selbsttätig abschaltet, sofern kein Programm läuft und über die Tastatur inner halb dieser Zeitspanne keine Ein gaben getätigt wurden.

POWER AOFF 1 schaltet diese "auto power off"Funktion ab.

Werden andere Werte als 0 oder 1 als Parameter dem Befehl POWER AOFF nachgestellt, so bleibt das Kommando wirkungslos.

Nach einem Total-Reset ist die "auto power off"Funktion aktiv. Ebenso läßt sich der Computer dann über ein Modem oder zu einer mit WAKE$ bestimmten Zeit automatisch einschalten.

PRESET

Siehe auch : PSET, LINE

WIRKUNG : PRESET löscht den Display-Punkt, der durch die angegebenen Koordinaten bestimmt ist.

HINWEISE : Die Koordinaten X und Y können im Bereich von -32768 bis 32767 liegen, ohne daß ein ERROR-Code generiert wird. Es sollten jedoch die folgenden Bereiche beachtet werden, wenn ein tatsächlich existierender Display-Punkt anzusprechen ist:

<X-Koordinate> : 0...155
<Y-Koordinate> : 0....31

Der gelöschte Punkt erscheint in der Anzeige hell.

BEISPIEL :

```
10:LINE (0,0)-(155,35),,,BF
20:FOR I=lTO 2000
30:X=RND(155): Y=RND(35)
40:PRESET(X,Y)
50:NEXT I
```

PRINT

Siehe auch : MODE, VSING, WAIT

ERKLÄRUNG : PRINT gibt Daten auf dem Display aus.

HINWEISE : Wie die Daten dabei auf dem Display positioniert werden, hängt von folgenden Faktoren ab:

- ob es sich um einzelne Daten handelt oder um eine Auflistung davon,

b) ob für die Auflistung als Trennungszeichen ein Komma (,) oder ein Semikolon (;) verwendet wird, und

c) ob sie numerischen Typs sind oder einen String darstellen.

Bevor Sie die nachfolgenden Erklärungen schrittweise durch einzelne Experimente nachvollziehen, sollten Sie wissen, daß das BASIC jede Zeile des Displays in zwei gleichgroße Zonen unterteilt. Jede Zone besteht damit aus 13 Spalten. Diese Zonen-Einteilung wird immer dann von der PRINT-anweisung berücksichtigt, wenn einzelne Daten vorliegen oder in der Auflistung ein Komma enthalten ist. Mit einem Semikolon voneinander getrennte Daten werden auf dem Display unmittelbar hintereinandergefügt angezeigt.

Wird die Zoneneinteilung berücksichtigt, wei1 in der PRINT-Anweisung nur ein Parameter beigefügt ist oder aber in der Auflistung ein Komma vorliegt, so erfolgt die Anzeige innerhalb einer solchen Zone :

bei numerischen Daten rechtsbündig,
bei String-Daten jedoch linksbündig.

Numerische Einzel-Daten werden in der rechten, einzelne String-Daten in der linken Zone dargestellt.

Beispiele zonenorientierter Ausgaben:

```
PRINT "A"
A

PRINT 2
2

PRINT "A",2
A          2

PRINT 2,"A"
2A

PRINT 1,2,3
1         2          3

PRINT "A","B",3
A         B          3
```

Im Anzeige - Modus MODE 1 kann die Liste nur zwei Daten enthalten, wenn das Komma zur Trennung verwendet wird.

In MODE 1

Anders a1s bei der LPRINT - Anweisung kann die Option TAB nicht in Verbindung mit der Anweisung PRINT stehen. Die Daten lassen sich aber durch die Anwendung der Option USING und der Anweisung CURSOR beliebig positionieren und formatiert darstellen.

Wird eine PRINT-Anweisung durch ein Semikolon abgeschlossen, so setzt sich die nächste Ausgabe in derselben Zeile fort. Schließt dagegen die PRINT-Anweisung mit einem Komma ab, erfolgt die nächste Ausgabe in der nächsten Zone. Beendet weder ein Semikolon noch ein Komma die PRINTAnweisung, so setzt sich die nächste Ausgabe in der folgenden Zeile fort:

```
1O:PRINT 2;
20:PRINT "A"
RUN
2A

10:PRINT 2,
20:PRINT "A"
RUN
             2A

1O:PRINT 2
20:PRINT "A"
RUN
                          2
A
```

Die a1s Parameter aufgeführten Daten brauchen nicht, wie es in den bisherigen Beispielen der Fall gewesen ist, aus Konstanten zu bestehen, sondern können beliebige numerische Ausdrücke oder Stringausdrücke sein. Beispiel:

```
1O:N$="SHARP "
20:PRINT N$+" PC";-(2+INT(1/EXP(-7,377)))

RUN
SHARP PC-1600
>
```

Eine PRINT-Anweisung ohne jegliche Parameterangabe bewirkt die Ausgabe einer Leerzeile, was praktisch mit einem Zeilenvorschub beim Drucker verglichen werden kann.

Ein PRINT-Befehl kann mit einer oder mehreren USING-Anweisungen versehen sein, so daß sich die Daten formatiert anzeigen lassen. Das jeweilige Format wird durch einen <Format-String>, der der USING-Anweisung als Parameter beizugeben ist, bestimmt.

Der <Format-String> besteht aus einer Reihe von speziellen Zeichen, die in Anführungsstriche eingeschlossen sein müssen.

Die Zeichen, die den <Format-String> bilden, sind:

das Nummernzeichen,
& das Kaufmannsund,
+ das Pluszeichen,
* der Multiplikationsstern,
^ der Potenzoperator,
, das Komma und
: der Dezimalpunkt.

Das Nummernzeichen und das Kaufmannsund sind sogenannte Platzhalter. Für jedes im <Formatstring enthaltene # kann eine Ziffer des numerischen Wertes angezeigt werden, für jedes & dagegen ein Zeichen eines Strings. Alle weiteren Formatsymbole dienen der näheren Beschreibung numerischer Formate.

Bei den numerischen Formaten lassen sich sowohl positive als auch negative Werte darstellen. Das Vorzeichen wird jedoch (von einer Ausnahme abgesehen) nur bei den negativen Werten angezeigt.

Zusammen mit dem Stringformat lassen sich insgesamt acht grundlegende Formate unterscheiden, die in den Erklärungen zu USING näher erläutert sind.

PRINT#

Siehe auch : INPUT#, OPEN, PRINT USING

WIRKUNG : PRINT# und PRINT# USING werden benutzt, um Daten sequentiell in eine Datei zu schreiben.

HINWEISE : Disketten- oder RAM-Disk-Dateien

<Datei-Nr.> ist die Nummer, unter der die Datei mittels OPEN-Befehl geöffnet wurde.Der Versuch, etwas in eine ungeöffnete Datei zu schreiben bewirkt die Ausgabe eines ERROR-Codes.

Sind die Daten durch Semikolons voneinander getrennt, so werden sie in der Datei ohne einen Zwischenraum aneinandergefügt. Wird ein Komma in der Auflistung der Daten verwendet, wird dagegen zwischen die Daten ein Leerzeichen geschrieben. Soll das Komma selbst mit in die Datei gelangen, ist es in Anführungsstriche zu setzen.

Die Benutzung von USING geschieht in gleicher Weise wie bei PRINT.

Cassetten-Dateien

Ist kein **<Dateiname>** spezifiziert, werden die Daten von der momentanen Bandposition an auf die Cassette aufgezeichnet. A1s Seperator zwischen den einzelnen Daten kann bei Cassetten-Dateien lediglich das Komma verwendet werden.

Die Aufzählung der Daten kann sowohl numerische als auch String-Variablen enthalten. In beiden Fällen ist auf die Verwendung der richtigen Trennungszeichen zu achten. Ansonsten kann es Probleme beim Einlesen mittels INPUT# geben. Mit Array-Variablen können keine individuellen Elemente angesprochen werden. Es muß das gesamte Array in der Form A(*) angegeben sein.

PSET

WIRKUNG : PSET setzt oder löscht einen Display-Punkt, der durch zwei Koordinaten bestimmt ist.

HINWEISE : Ein gesetzter Punkt erscheint dunkel, ein gelöschter dagegen hell.

Die Koordinaten X und Y können im Bereich von -32768 bis 32767 liegen, ohne daß ein ERROR-Code generiert wird. Es sollten jedoch die folgenden Bereiche beachtet werden, wenn ein tatsächlich existierender Display-Punkt anzusprechen ist:

<X-Koordinate> : 0...155
<Y-Roordinate> : 0....31

Benutzt man PSET ohne die Option <Code>, so wird der mit den Koordinaten bestimmte Display-Punkt gesetzt. Ist der <Code> vorhanden, invertiert PSET den gerade derzeitigen Zustand des Punktes. Ein heller Punkt wird dunkel, ein dunkler hell. Der <Code> muß aus dem Buchstaben X bestehen.

PZONE

Siehe auch : LPRINT

WIRKUNG : PZONE bestimmt, innerhalb welcher Zonen die Ausgaben, die zum Drucker oder einem seriellen Port gehen, eingebettet werden.

HINWEISE : In BASIC werden die über LPRINT auszugebenden Daten in Zonen mit fester Zeichenanzahl ausgegedruckt bzw. ausgesendet. Diese in Zonen eingeteilte Ausgabe gilt dann, wenn die Daten in einer Liste durch Komma voneinander getrennt vorkommen. PZONE bestimmt die <Länge> der Zonen.

Diese zonenorientierte Ausgabe gilt auch dann, wenn die Daten über das Interface ausgesendet werden. Den rechtsbündig in der Zone liegenden Daten wird somit eine entsprechende Anzahl an Leerstellen (spaces) vorausgeschickt.

Folgende Schnittstellen sind ansprechbar:

```
COM1: RS-232C-Interface
COM2: SIO-Interface
COM:  das derzeit über SETDEV
      selektierte   Interface
```

Folgende Werte sind für die <Länge> erlaubt:

Für COM1: 8...255 (Standard: 20)
Für COM2: 8....80 (Standard: 20)

BEISPIEL :

```
10:LET A=1,B=22,C=33
20:FOR I=1TO 30:LPRINT "-";:NEXT I
30:LPRINT
40:PZONE "LPT1:",10
50:LPRINT A,B,C
60:END
```

RADIAN

Siehe auch : DEGREE, GRAD

WIRKUNG : Versetzt den Computer in den Winkelmlodus RADIAN

HINWEISE : In diesem Modus werden alle Winkelangaben als im Bogenmaß gegeben angesehen und auch in diesem Maß ausgegeben.

Zur Kennzeichnung dieser Betriebsart wird in der Status-Zeile das Symbol RADIAN angezeigt.

Die Argumente der Funktionen SIN, COS und TAN werden dann als im Bogenmaß vorliegend angesehen und die Werte der Funktionen ASN, ACS und ATN in diesem Maß geliefert.

BEISPIEL :

```
10:RADIAN
20:PAUSE "WINKEL IM BOGENMASS"
30:PRINT ASN(0.5),ASN(1)
40:PRINT ACS(0.5),ACS(1)
50:PRINT ATN(0.5),ATN(1)
60:END

RUN
WINKEL IM BOGENMASS
5.235987E-01 1.570796327
1.047197551           0
0.463647609 7.853981E-01 >
```

Lassen Sie dieses Programm zum Vergleich auch in den beiden anderen Winkel-Modi (DEGREE und GRAD) laufen.

RANDOM

Siehe auch: RND

WIRKUNG : RANDOM bestimmt die Serie der Zufallszahlen , die über die Funktion RND geliefert werden.

HINWEISE : Benutzt man RANDOM zu Beginn eines Programmes,wird mit jedem Einschalten des Computer eine neue Reihe von Zufallszahlen erzeugt .

BEISPIEL :

```
10:RANDOM
20:FOR I=lTO 5
30:PRINT I; " . ZUFALLSZAHL = ";
40:PAUSE RND ( 10 )
50:NEXT I
60:END

RUN
1. ZUFALLSZAHL = 2
2. ZUFALLSZAHL = 1
3. ZUFALLSZAHL = 10
4. ZUFALLSZAHL = 8
5. ZUFALLSZAHL = 10
>
          OFF

          ON

>
RUN
 1. ZUFALLSZAHL = 2
 2. ZUFALLSZAHL = 1
 3 . ZUFALLSZAHL = 1 0
 4. ZUFALLSZAHL = 8
 5. ZllFALLSZAHL = 10
 >
```

RCVSTAT

Siehe auch : RCVSTAT

ERKLÄRUNG : RCVSTAT setzt Steuersignale für das RS-232CInterface und die Wartezeit (timeout) für beide seriellen Schnittstellen (SIO und RS-232C) für den Datenempfang.

HINWEISE : Die Auswahl des Interface erfolgt über den Parameter "COMn:". Für diesen gilt:

COM1: RS-232C-Interface
COM2: SIO-Interface
COM: Zuletzt über SETDEV selektiertes Interface

Empfängt der PC-1600 von einem peripheren Gerät Daten über eines der seriellen Interfaces, so muß er, damit dieses einwandfrei funktioniert, stets vom Peripherigerät über spezielle Steuersignale darüber informiert werden, ob dieses senden möchte bzw. der PC-1600 dazu bereit ist, Daten zu empfangen. Diesen Austausch von Steuersignalen nennt man auch "Handshaking".

Die Zustände der Steuersignale, die für einen weiteren Datentransfer erfüllt sein müssen, werden durch den Parameter **<Protokoll>** bestimmt. Dieser Parameter ist eine Dezimalzahl im Bereich 0 ... 255 und wird intern in eine 8-Bit-Zahl (also ein Byte) umgewandelt. Jedes Bit bestimmt dabei den Zustand eines der speziellen Steuersignale. Sind alle Bedingungen erfüllt, kann der Datentransfer fortgesetzt werden. Die Bits 7 und ~ werden intern immer zu Null angenommen, da sie für das "Handshaking" nicht benötigt werden.

Die Zuordnung zwischen den Bits und den Signalen sowie die Bedeutung der Bitzustände lautet wie folgt:

Bit (LSB)	Wert		Signal
Niederwertigstes Bit	O	O	DTR high
		1	DTR low
	1	O	RTS high
		1	RTS low
	2	0	CTS high
		1	CTS low
	3	0	CD high
		1	CD low
	4	0	DSR high
		1	DSR low
	5	0	CI high
		1	CI low
	6	0	ungenutzt
Höchstwertigstes Bit (MSB)	7	0	ungenutzt

Fehlt der Parameter (Protokoll>, wird für ihn standardmäßig der Wert 63 (00111111) angenommen, was den Datenempfang ohne Steuersignale erlaubt.

Unter "timeout" versteht man die Zeitdauer, die der Computer längstens auf eine Antwort des Peripheriegerätes wartet, bis er den Datentransfer abbricht. Mit dem Parameter <Zeit> kann sie mit den Werten 1 bis 255 als ein entsprechend Vielfaches von 0.5 Sekunden eingestellt werden. Der Wert O setzt die Wartedauer auf unendlich, was auch der standardmäßigen Annahme entspricht, wenn dieser Parameter weggelassen wird.

READ..DATA

Siehe auch : DATA, RESTORE

WIRRUNG : READ liest aus den im Programm befindlichen DATA-Zeilen die dort enthaltenen Werte ab und weist sie den gewünschten Variablen zu.

HINWEISE : Damit eine READ-Anweisung durchgeführt werden kann, muß innerhalb des Programmes mindestens eine DATA-Zeile vorhanden sein.

Jeder als Parameter aufgeführten Variablen wird dann der Reihe nach das nächste in einer DATA-Zeile auffindbare Element zugeordnet. Die in der Parameterauflistung enthaltenen Variablen müssen jeweils typenmäßig den in der DATA-Anweisung auftretenden Konstanten entsprechen.

Die einzulesenden Werte brauchen nicht allesamt in einer gemeinsamen DATA-Zeile stehen, sondern können auf beliebig viele DATA-Zeilen verteilt sein. Es gilt dabei immer, daß mit der nächsten READ-Variablen der nächste in einer DATA-Zeile auffindbare Wert gelesen wird.

Sind die in den DATA-Zeilen stehenden Werte alle gelesen, führt die nächste READ-Anweisung zur Ausgabe eines ERROR-Codes.

Beachten Sie die Beschreibungen von MERGE, wenn Sie READ- und DATA-Anweisungen in Programmen benutzen, die mit MERGE verbunden werden sollen.

BEISPIEL :

```
10:DIM B(10)
20:FOR I=1 TO 10 : READ B(I)
30:PRINT B(I) : NEXT I
40:DATA 10,20,30,40,50
50:DATA 60,70,80,90,100
```

REM

Siehe auch : LIST , LLIST

WIRKUNG : REM erlaubt die Einfügung von Kommentaren in den Programmtext.

HINWEISE : Diese Kommentare dienen zur Kennzeichnung von Programmteilen, so daß man ihre Funktionen noch zu einem späterem Zeitpunkt wieder erkennen und verstehen kann. Sie erscheinen ausschließlichim Programmlisting. Sie bleiben also bei der Programmausführung unsichtbar. Anstelle von REM kann auch der Apostroph , verwendet werden.
Die so gekennzeichneten Programmzeilen zählen zu den nicht ausführbaren Anweisungen. Werden diese durch GOTO oder GOSUB angesprungen, erfolgt die Programmausführung mit der nächsten Zeile , die nicht als Kommentarzeile ausgewiesen ist.
ommentare können an Anweisungen einer Zeile angefügt werden, wenn man vor dem Befehlswort REM den Doppelpunkt als Trennzeichen benutzt.

```
10:V=G*H/3: REM VOLUMEN DER PYRAMIDE
```

Einem Apostroph darf kein Doppelpunkt voraus-gehen , da dieser die Trennfunktion beinhaltet:

```
10:V=G*H/3 ,VOLUMEN DER PYRAMIDE
```

Nach der Einleitung eines Kommentares gilt der gesamte Rest der Zeile als Kommentar. Hinter einem Kommentar stehende Anweisungen , die zur selben Zeile gehören, werden ignoriert:

```
10:REM VOLUMEN DER PYRAMIDE: V=G*H/3
20:'VOLUMEN DER PYRAMIDE: V=G*H/3
```

RENUM

Siehe auch : DELETE, EDIT, LIST

WIRKUNG : RENUM nimmt eine Neunumerierung der Zeilen eines BASIC-Programmes vor.

HINWEISE : **<alte Zeilen-Nr.>** bestimmt, ab welcher Zeile mit der neuen Numerierung begonnen wird.

<neue Zeilen-Nr.> bestimmt, wie die erste neue Zeilennummer lautet.

<Zeilenabstand> bestimmt, in welchen Abstand die Nummern generiert werden sollen. Fehlt diese Angabe, erfolgt die Numerierung in Zehnerschritten.

Bei der Neunumerierung werden automatisch alle Zeilennummern in GOTO- und GOSUB-Anweisungen entsprechend korrigiert. Dies gilt auch für die Zeilennummern, die in anderen Verzweigungsanweisungen, wie z.B. IF..THEN..ELSE, enthalten sind. Taucht dabei aber eine nichtexistente Zeilennummer auf, wird die Meldung:

Undefined in <Zeilen-Nr.>

angezeigt und die Neunumerierung unterlassen. Die <Zeilen-Nr.> verweist dabei auf die Zeile, in der der Fehler entdeckt worden ist.

Die höchste generierbare Zeilennummer ist 65279.

RENUM

BEISPIEL :

```
10:INPUT "ANTWORT (J/N) =";A$
20:IF A$="J" THEN 10
30:IF A$="N" THEN 60
40:PRINT "NUR MIT J ODER N ANTWORTEN !"
50:GOTO 10
60:END


>RENUM 100,10,5


100:INPUT "ANTWORT (J/N) =";A$
105:IF A$="J" THEN 100
110:IF A$="N" THEN 125
115:PRINT "NUR MIT J ODER N ANTWORTEN !"
120:GOTO 10
125:END
```

RESTORE

Siehe auch : READ, DATA

WIRKUNG : RESTORE setzt den internen DATA-Zeiger zurück oder auf den Anfang der gewünschten DATA-Zeile. Damit lassen sich die in den DATA-Zeilen bereitgestellten Werte erneut lesen.

HINWEISE : Wird RESTORE ohne Parameter verwendet, zeigt der interne Zeiger anschließend auf den ersten Wert der zuerst im Programm auffindbaren DATA-Zeile.

Bei der Angabe einer **<Zeilen-Nr.>** oder einer **<Marke>** wird der Zeiger auf das erste Element der in dieser Zeile vorkommenden DATA-Anweisung gesetzt.

Nach Setzung des Zeigers lassen sich die Werte der momentan vom Zeiger bestimmten DATA-Liste mit jeder folgenden READ-Anweisung der Reihe nach ablesen und den durch READ bestimmten Variablen zuweisen. Sind alle Werte einer DATAAnweisung gelesen, sucht der Computer nach der nächsten im Programmspeicher befindlichen Zeile mit einer DATA-Anweisung und setzt den Zeiger auf deren Anfang und so fort.

Enthält die als Parameter angegebene Zeile keine DATA-Liste, wird der Zeiger auf den Anfang der nächsten auffindbaren DATA-Anweisung gesetzt.

RESTORE

BEISPIEL :

```
100:DIM A$(3*10)
110:GOSUB "OBST"
120:RESTORE
130:GOSUB "OBST"
140:RESTORE 310
150:GOSUB "OBST"
160:END
200:"OBST"
210:FOR N=1 TO 3
220:READ A$(I)
230:PAUSE A$
240:NEXT N
250:PAUSE
260: RETURN
300:DATA "PFLAUME","PFIRSICH","NEKTARINE"
310:DATA "APFEL","BIRNE","MANDARINE"


>
RUN
PFLAUME
PIRSICH
NEKTARINE

PFLAUME
PIRSICH
NEKTARINE

APFEL
BIRNE
MANDARINE
>
```

RESUME

Siehe auch : ON ERROR GOTO, ERL, ERN

WIRKUNG : RESUME dient zur Rückkehr von einer Fehlerbehandlungs-Routine in das Hauptprogramm.

Das RESUME-Kommando wirkt ähnlich wie das RETURN eines normalen Unterprogrammes. Es darf aber nur in solchen Routinen verwendet werden, die der Auswertung und Behandlung von Fehlern dienen und von ON ERROR GOTO aufgerufen werden.

RESUME <Zeilen-Nr.>

setzt das normale Programm nach Beendigung der Fehlerbehandlungsroutine mit der spezifizierten Zeile fort.

RESUME NEXT

setzt den normalen Programmablauf mit der Zeile fort, die unmittelbar auf diejenige folgt, die mit ihrem Fehler für die Aktivierung der Fehlerbehandlungsroutine sorgte.

RESUME

setzt den normalen Programmablauf mit der Zeile fort, die den Fehler verursachte.

RETI

Siehe auch : ON ADIN GOSUB, ON COMn GOSUB, ON KEY GOSUB, ON PHONE GOSUB, ON TIME$ GOSUB

WIRKUNG : RETI dient zur Rückkehr von einer InterruptRoutine in das Hauptprogramm.

HINWEISE : RETI wird in der gleichen Weise verwendet wie RETURN. Der Unterschied ist jedoch, daß RETI keine normalen Unterprogramme beendet, sondern solche, die der Behandlung eines Interrupts dienen. Solche Interrupt-Routinen werden durch die Anweisungen ON ADIN GOSUB, ON COMn G09UB, ON KEYn GOSUB, OH PHONE GOSUB und ON TIME$ GOSUB aufgerufen. Sollte während der Bearbeitung einer Interrupt-Routine ein weiterer Interrupt angefordert werden, wird dieser mit Ausführung des RETI-Befehles angenommen und die entsprechende Interrupt-Routine bearbeitet.

BEISPIEL :

```
10:0N PHONE GOSUB
:
:
100:PRINT "DA IST EIN ANRUF FÜR DICH!"
110:RETI
```

RIGHT$

Siehe auch : LEFT$, MID$

WIRXUNG : Die Funktion RIGHTS liefert den rechtsbündigen Teil eines vorgegebenen Strings, wobei bestimmt werden kann, wieviele Zeichen von rechts-aus dem String abgelesen werden.

HINWEISE : Die <Anzahl> der Zeichen des Teilstringes mup im Bereich von 0 bis 80 liegen. Gibt man die Anzahl als gebrochene Zahl an, wird sie zur nächsten ganzen Zahl hin gerundet. Ist die Anzahl größer als die Zeichenanzahl des vorgegebenen Strings, wird der gesamte vorgegebene String geliefert.

BEI5PIEL :

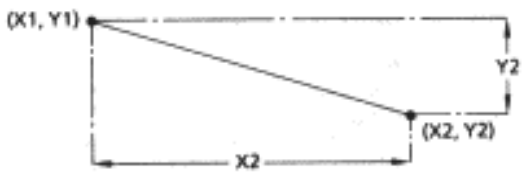
```
5:WAIT 32
10:X$="SHARP PC-1600"
20:FOR N=1TO 14
30:S$=RIGHT$(XS,N)
40:PRINT S$
50:NEXT N
60:WAIT
70:END

>RUN
0
00
600
1600
-1600
C-1600
PC-1600
PC-1600
P PC-1600
RP PC-1600
ARP PC-1600
HARP PC-1600
SHARP PC-1600
SHARP PC-1600
>
```

RLINE

Siehe auch : COLOR, LLINE

WIRKUNG : RLINE druckt eine oder bis zu vier aufeinanderfolgende Linien zwischen den jeweils definierten relativen Koordinaten-Punkten.

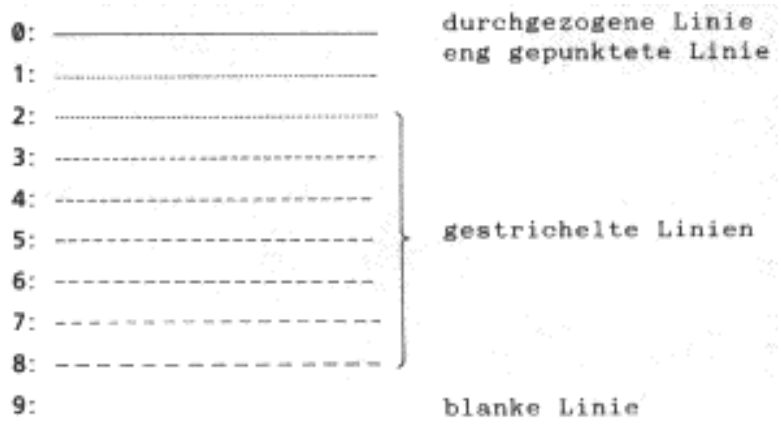
HINWEISE : Im Grafik-Modus druckt RLINE eine Linie von der Koordinate (X1,Y1) oder der momentanen Position des Stiftes bis zum Punkt (X2,Y2). Die X- und Y-Koordinaten müssen im Bereich von -2048 bis 2047 liegen. Anders a1s bei LLINE beziehen sie sich hier auf die letzte Stift-Position und nicht auf den mit SORGN vereinbarten Koordinatenursprung.



In einer RLINE-Anweisung können bis zu vier zu zeichnende Linien definiert werden.

Wird RLINE unmittelbar nach einem LPRINT-Befehl ausgeführt, muß der Parameter <Typ> unbedingt mit angegeben werden, da in diesem Falle für ihn kein Standardwert angenommen wird.

<Typ> bestimmt die aus 9 verschiedenen Mustern gewünschte Linienstruktur:



Der Typ 9 dient zum Test für die Stiftbewegung. Bei fehlender Angabe von <Typ> bleibt die letzte gewählte Einstellung gültig.

<Farbe> bestimmt die Wahl des Farbstiftes, von denen vier zur Auswahl stehen:

0	schwarz
1	blau
2	grün
3	rot

Bei fehlender Angabe der <Farbe> bleibt die zuletzt vorgenommene Einstellung erhalten.

Die Angabe der Option B sorgt für das Zeichnen eines Rechteckes, welches die Punkte (X1,Y1) und (X2,Y2) a1s Diagonale hat.

Die Anweisung RLINE (X1,Y1)-(X2,Y2),,,B ergibt also folgendes Bild:

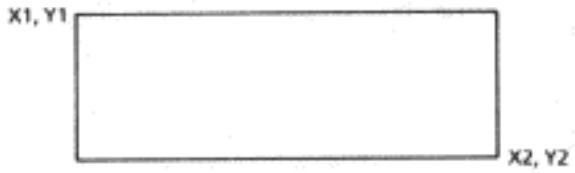


Abbildung 46

RMT ON/OFF

WIRKUNG : RMT ON/OFF gestattet oder verhindert die Fernbedienung des Cassetten-Recorder-Motors.

HINWEISE : RMT ON erlaubt die automatische Ein- bzw. Ausschaltung des Recorder-Motors bei Anwendung von BASIC-Befehlen, die sich auf Cassetten-Dateien beziehen.

RMT ON erlaubt die automatische Ein- bzw. Ausschaltung des Recorder-Motors bei der Anwendung von BASIC-Befehlen, die sich auf Cassetten-Dateien beziehen.

RMT OFF verbietet eine solche Fernbedienung des Recorder-Motors. Der Cassetten-Recorder läßt sich jedoch weiterhin manuell auf normale Weise bedienen.

Die automatische Steuerung des Motors kann nur dann erfolgen, wenn außer dem Befehl RMT ON auch der REMOTE-Schalter des Druckers aktiv ist, d.h. dieser in Stellung ON steht.

RND

Siehe auch : RANDOM

WIRKUNG : Die Funktion RND liefert eine Zufallszahl, deren maximale Größe vom angegebenen Argument abhängt.

HINWEISE : Mit Hilfe der RND-Funktion lassen sich scheinbar willkürlich verteilte Zahlen erzeugen. Schaltet man jedoch den Computer an und startet das mit der RND-Funktion versehene Programm erneut, so stellt man fest, daß genau dieselben "Zufallszahlen" geliefert werden wie zuvor. Sollen die gelieferten Werte einem "echten Zufallsgesetz" folgen, muß vor Aufruf von RND-Funktionen im Programm das Befehlswort RANDOM aufgeführt sein.

Ist das Argument der RND-Funktion kleiner als Null liefert jeder erneute Funktionsaufruf immer wieder dieselbe Zufallszahl.

Ist das Argument größer als Null, aber kleiner als 1, liegen die gelieferten Funktionswerte im Bereich von 0 bis 1, ohne Einschluß der 1. Alle Werte werden hierbei mit bis zu 10 signifikanten Stellen geliefert.

Ist das Argument jedoch größer a1s 1, liegen die Funktionswerte im Bereich von 1 bis zum Argument selbst. Hierbei werden allerdings nur IntegerWerte ausgegeben.

BEISPIEL :

```
10:FOR I=1 TO 3
20:FOR J=1 TO 10 : R=RND(9) : PRINT R; : NEXT J
30:PRINT : NEXT I
40:END

>RUN
6 4 2 5 6 8 2 7 6 8
5 5 7 7 1 2 6 5 3 6
3 1 5 7 3 4 5 7 4 2
```

ROTATE

WIRKUNG : ROTATE bestimmt die Orientierung der gedruckten Zeichen und die Druckrichtung.

HINWEISE : Der Parameter <Richtung> bestimmt sowohl die Druckrichtung als auch die Orientierungslage der Zeichen. Er muß ganzzahlig sein und im Bereich von 0 bis 3 liegen. Diese Werte haben folgende Bedeutung:

<Richtung>	0rientierung	Druckrichtung
0	normal lesbare Lage	nach rechts
1	auf rechter Seite liegend	nach unten
2	auf dem Kopf stehend	nach oben
3	auf linkernach Seite liegend	links

Entfällt der Parameter <Richtung>, bleiben alle bisherigen Einstellungen erhalten.

BEISPIEL :

```
5: GRAPH
10:GLCURSOR(400,0)
20:FOR D=0 TO 1
30:FOR P=0 TO 3
40: ROTATE P,D
50:LPRINT " ABC ";
60:NEXT P
70:NEXT D
80:END
```

RUN

Siehe auch : CONT, GOTO, LOAD, MERGE

WIRKUNG : RUN startet ein im Speicher befindliches BASICProgramm.

HINWEISE : Ohne Parameterangabe beginnt die Ausführung des Programmes mit dessen erster Zeile, also der, der kleinsten vorkommenden Zeilennummer.

Mit Angabe der **<Zeilennummer>** oder einer **<Marke>** wird das Programm ab der spezifizierten Stelle getsartet.

In jedem Falle löscht RUN Variablen und setzt den internen Zeiger für die Auswahl der in den DATA-Zeilen bereitgestellten Daten auf die erste mögliche Position.

BEISPIELE :

```
>
RUN

>
RUN 100

>
RUN "F"

>
RUN "MARKE"
```

RXD$

WIRKUNG : RXD$ liefert die Daten, sofern vorhanden, die über das momentan selektierte Interface an den Computer abgegeben werden.

HINWEISE : Das derzeit am Interface empfangene Daten-Byte wird von RXD$ als String geliefert.

Ist momentan kein Zeichen am selektierten Port (Interface) vorrätig, so erhält man über RXD$ zwei Leerzeichen (ieweils Code &20H).

Liegt ein Übertragungsfehler am Interface vor, liefert RXD$ die Code-Folge &3F,&20,&20, also ein Fragezeichen mit zwei nachfolgenden Spaces.

BEISPIEL :

```
10:FOR I=1 TO 4
20:A$=A$+"*"+RXD$
30:NEXT I
40:PRINT A$
50:END
```

Lassen Sie dieses Programm laufen, ohne daß an eine der seriellen Schnittstellen etwas angeschlossen ist. Somit können über diese folglich auch keine Daten ankommen. RXD$ muß daher bei jedem Aufruf zwei Leerstellen liefern. Damit ist das Ergebnis des Programmablaufes wie folgt:

```
>RUN
* * * *
>
```

SAVE / SAVE*

Siehe auch : LOAD, MERCE

WIRKUNG : SAVE sichert eine Datei auf einer Diskette oder RAM-Disk oder sendet diese über ein serielles Interface aus.

HINWEISE : SAVE sichert die Datei unter dem angegebenen Namen auf dem spezifizierten Medium oder sendet es über das gewählte Interface aus.

SAVE* kann nur die mit dem Befehl REM oder einem Apostroph gekennzeichneten Kommentarzeilen eines BASIC-Programmes sichern. Ansonsten verhält sich dieser Befehl wie SAVE. Auf diese Weise lassen sich Text-Dateien abspeichern, die später mit LOAD* wieder geladen werden können.

Gibt man die Option A als Parameter an, erfolgt die Dateisicherung im ASCII-Format. Fehlt diese Angabe, wird das kompaktere Binärformat gewählt.

Existiert bereits eine Datei unter gleichem Namen, so wird diese mit der neuen Datei übersehrieben, sofern sie nicht dureh die P-Option des SET-Kommandos gesehützt worden ist.

Bei RAM-Modulen ist ein Übersehreiben dann nieht möglieh, wenn sich der Schreibschutzschalter in Stellung ON befindet.

Läßt man in der Dateibezeichnung die Extension weg, wird sie automatisch zu .BAS angenommen.

BEISPIELE :
```
>
SAVE "X:CHESS"

>
SAVE* "S2:INFO"
```

SET

WIRKUNG : SET setzt oder entfernt den Schreibschutz von Disketten- oder RAM-Disk-Dateien.

HINWEISE : Das SET-Kommando dient dem Schutz von Dateien gegen unerlaubten Zugriff bzw. unbeabsichtigter Löschung.

Der Parameter **<Dateibezeichner>** spezifiziert das Speichermedium, den Dateinamen und die Extension der zu schützenden Datei.

"P" setzt den Schreibschutz.

Die betreffende Datei kann danach weder:

- mit dem NAME-Kommando umbenannt werden, noch
- mit dem KILL-Kommando gelöscht werden, noch
- im APPEND- oder OUTPUT-Modus geöffnet werden.

Es sind keine Daten in eine mit "P" geschützte Datei schreibbar.

" " beseitigt den Schreibschutz wieder.

SETCOM

Siehe auch : COM$, SETDEV

WIRKUNG : SETCOM definiert das Kommunikationsprotokoll der seriellen Schnittstellen.

HINWEISE : Folgende Interfaces können gewählt werden:

COM1: RS-232C-Interface
COM2: SIO-Interface
COM: mit SETDEV selektiertes Interface

Das Kommunikationsprotokoll wird durch sechs optionale Parameter, die durch Kommas voneinander getrennt sein müssen, bestimmt. Dabei gilt:

  Vbertragungsrate:	50 bis 38400 baud
<WL> Wortlänge:	5 bis 8 Bits
<PR> Parität:	E (even) = gerade
	O (odd) = ungerade
	N (none) = keine
<ST> Stopbit-Anzahl:	1 oder 2
<XO> XON/XOFF:	X = ja; N = nein
<SI> SHIFT IN/SHIFT OUT :	S = ja; N = nein

(Nur für 7-Bit-Übertragung)

Die standardmäßigen Einstellungen lauten:

RS-232C : 1200,8,N,1,X,S
SIO : 38400,7,E,2,X,S

Werden die Befehle SAVE, LOAD, BSAVE oder BLOAD benutzt, um binäre Dateien über einen seriellen Port zu übertragen, muß die Wortlänge auf 8 Bit und der Parameter <SI>=N gesetzt werden. Bei der Sicherung von ASCII-Dateien mittels SAVE oder dem Laden solcher Dateien über LOAD gelten keine Einschränkungen der Kommunikationsparameter. Dieses gilt auch für die Verwendung der Befehle PRINT# und INPUT#.

SETDEV

WIRKUNG : SETDEV bestimmt, welches serielle Interface für die Ein- oder Ausgabe von Daten bei Anwendung bestimmter BASIC-Befehle zu verwenden ist.

HINWEISE : Das Kommando SETDEV öffnet damit im Grunde ein Interface für den Datentransfer. Die Selektion übernehmen folgende Parameter:

COM1: öffnet das RS-232C-Interface
COM2: öffnet das SIO-Interface
COM: beläßt das zuletzt verwendete Interface

Mit zwei zusätzlichen Parametern, die optional sind und in beliebiger Reihenfolge angegeben werden können, lassen sich die Ein- und Ausgabe von Daten wie folgt leiten:

PO Leitet die über LPRINT, LLIST und LFILES gelieferten Daten an das spezifizierte Interface.

KI Bestimmt, daß der INPUT-Befehl die Daten vom spezifizierten Interface holt.

Wird der Befehl SETDEV ohne jegliche Angabe von Parametern verwendet, werden die standardmäßigen Einstellungen wieder hergestellt. Alle Datenausgaben werden damit zum Drucker geführt und die Eingaben von der Tastatur erwartet.

BEISPIELE :

```
>
SETDEV "COM1:",PO
>
SETDEV
```

SGN

WIRKUNG : Die Funktion SGN(X) liefert das Vorzeichen des Argumentes X.

HINWEISE : X kann dabei jeder beliebige numerische Ausdruck sein. Die dazu gelieferten Funktionswerte lauten -1,0 oder 1 und ergeben sich wie folgt:

Argument X	Funktionswert SGN(X)
X > 0	1
X = 0	0
X < 0	-1

BEISPIEL :

```
5:WAIT 100
10:FOR N=-3 TO 3
20:PRINT N,SGN(N)
30:NEXT N
40:END

>
RUN
                          -3    -1
                          -2    -1
                          —1    -1
                          0     0
                          1     1
                          2     1
                          3     1
 >
```

SIN

Siehe auch : ASN, COS, TAN

WIRKUNG : Diese Funktion liefert den Sinus-Wert eines Winkelargumentes,

HINWEISE : Der angegebene Winkel kann in Altgrad, Bogenmaß oder Neugrad vorliegen. Damit der richtige Wert der Sinusfunktion geliefert werden kann, muß der Computer in den richtigen Winkelmodus geschaltet sein. Hierbei gilt:

Winkelmaß	Winkelmodus
Altgrad	DEGREE
Bogenmaß	RADIAN
Neugrad	GRAD

BEISPIEL :

```
lO:DEGREE
20:G$=CHR$ (&F8)
3O:PRINT "sin(3O";G$;") = ";SIN(3O)
40:PRINT "sin(45";G$;") = ";SIN(45)
50:END

>
RUN
sin(30°) = 0.5
sin(45°) = 7.071067812E-O1
>
```

SNDBRK

WIRKUNG : SNDBRK sendet eine vereinbarte Anzahl von Unterbrechungscodes (break codes) über das gewählte Interface aus, um dem sendenden Peripheriegerät mitzuteilen, daß momentan keine Daten angenommen werden können.

HINWEISE : Das Interface wird wie folgt ausgewählt:

COM1: RS-232C-Interfac
COM2: SIO-Interface
COM: über SETDEV selektiertes Interface

Der Parameter **<Anzahl>** bestimmt, wie oft der Unterbrechungs-Code hintereinander ausgesendet werden soll. Der betreffende Wert kann zwischen 1 und 255 liegen.

BEISPIEL :

```
>
SNDBRK "COM1:",20
```

SNDSTAT

Siehe auch : RCVSTAT

WIRKUNG : SNDSTAT bestimmt das "Handshake"-Protokoll des RS-232C-Interface und die Wartezeit (timeout) beider serieller Schnittstellen.

HINWEISE : Die Auswahl der Schnittstelle erfolgt über den Parameter "COMn:", für den gilt:

COM1: RS-232C-Interface
COM2: SIO-Interface
COM: über SETDEV selektiertes Interface

Sendet der PC-1600 an ein peripheres Gerät Daten über eine der seriellen Schnittstellen aus, so muß er stets vom Peripherigerät über spezielle Steuersignale darüber informiert werden, ob er weiter senden darf oder nicht. Diese Prozedur nennt man "Handshaking".

Die Steuersignal-Zustände, die zur Fortsetzung des Datentransfers erfüllt sein müssen, werden mit dem Parameter **<Protokoll>** festgelegt, der durch eine Dezimalzahl im Bereich von 0 bis 255 gegeben sein kann. Dieser wird intern durch eine 8 Bit umfassende binäre Zahl dargestellt. Jedes Bit bestimmt dabei den geforderten Zustand eines speziellen Steuersignales. Sind alle Bedingungen erfüllt, kann der Datentransfer weitergeführt werden. Die Bits 7 und 8 sind für den Handshake nicht maßgebend und werden daher stets zu Null angenommen.

Die Zuordnung zwischen den Bits und den Signalen sowie die Bedeutung der Bitzustände lautet:

Bit		Wert	Signal
(LSB)			
Niederwertigstes Bit	0	0	DTR high
		1	DTR low
	1	0	RTS high
		1	RTS low
	2	0	CTS high
		1	CTS low
	3	0	CD high
		1	CD low
	4	0	DSR high
		1	DSR low
	5	0	CI high
		1	CI low
	6	0	ungenutzt
Höchstwertigstes Bit	7	0	ungenutzt
(MSB)			

Fehlt der Parameter <Protokoll>, wird für diesen standardmäßig der Wert 59 (00111011) angenommen. In diesem Falle ist das Signal CTS also high.

Unter "timeout" versteht man die Zeitdauer, die der Computer längstens auf eine vom Peripheriegerät stammende Antwort wartet, bevor er den Datentransfer abbricht. Mit dem Parameter <Zeit> kann diese als ein Vielfaches von 0.5 Sekunden eingestellt werden, wobei die Parameterwerte 1 bis 255 zulässig sind.

Der Wert 0 setzt die Wartedauer auf unendlich, was der standardmäßigen Einstellung entspricht, wenn dieser Parameter weggelassen wird.

BEISPIEL : SNDSTAT "COM1:",45

SORGN

Siehe auch : LLINE, GLCURSOR

WIRKUNG : SORGN bestimmt die momentane Stiftposition des Druckers als künftigen Koordinatenursprung.

HINWEISE : Diese Festlegung des Koordinatenpunktes (0,0) an die derzeitige Stiftposition gilt nur für den Grafik-Modus und bezieht sich damit nur auf alle folgenden grafischen Befehle.

Mit SORGN kann der standardmäßige Koordinatenursprung nach den individuellen Bedürfnissen verlagert werden.

Die standardmäßige X-Koordinate des Koordinatenursprungs wird duroh die am linken Papierrand liegende Schreibposition bestimmt, die Y-Koardinate durch die mit den Tasten ⇑ und ⇓ setzbare Papierposition.

STATUS

Siehe auch : MEM, NEW

WIRKUNG : STATUS ermittelt die Anzahl der freien Speicherplätze an und liefert weitere Informationen über andere Speicherbereiche.

HINWEISE : Der Parameter <Code> bestimmt denjenigen Bereich des Speichers, über den nähere Informationen gewünscht sind und welcher Art diese sein sollen. Hierbei gilt folgender Zusammenhang:

<Code>	Information
0	Anzahl der freien Speicherplätze des Benutzerbereiches gemessen in Bytes. (Freier Bereich + Bereich der Variablen)
1	Größe des momentarn geladenen Programmes in Bytes.
2	Niederwertigste Speicher-Adresse, ab der der freie Benutzerbereich beginnt.
3	Höchstwertigste Speicher-Adresse, mit der der freie Benutzerbereich endet.
4 : : 255	Letzte Zeilennummer des auszuführenden BASIC-Programmes.

256 Nummer der Speicherbank, in der die niederwertigste Adresse des freien Benutzerbereiches existiert.

257 Nummer der Speicherbank, in der die höchstwertigste Adresse des freien Benutzerbereiches existiert.

258 Betrag (in Bytes) des nichtzugewiesenen Speicherplatzes innerhalb des freien Benutzerbereiches.

259 Anzahl der freien Bytes des im Modulfach S1 steckenden Programm-Modules, die von dessen Programm nicht belegt sind.

260 Anzahl der freien Bytes des im Modulfach S2 steckenden Programm-Modules, die von dessen Programm nicht belegt sind.

Speicheraufteilung des internen RAM s:

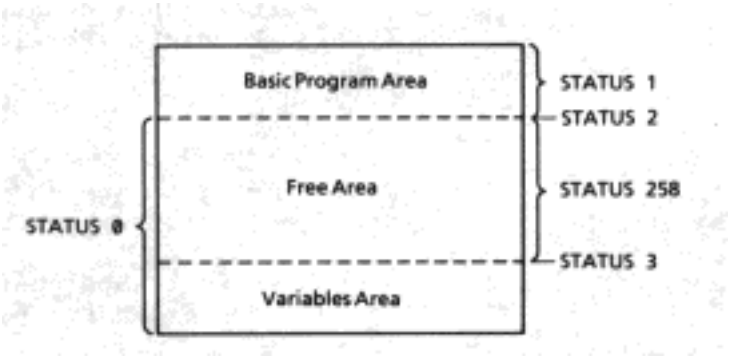


Abbildung 49

Speicheraufteilung des internen RAMs und der Erweiterungs-Module:

Das folgende Bild zeigt die Speicheraufteilung bei Verwendung von:

a) einem CE-1600M-Modul im Fach S1, wobei es für die Aufnahme von Programmen vorgesehen ist und der davon nicht berührte Speicherplatz zur Erweiterung des Arbeitsspeichers genutzt wird,

b) einem CE-1600M-Modul in Fach S2, welches ausschließlich zur Programmspeicherung initialisiert worden ist.

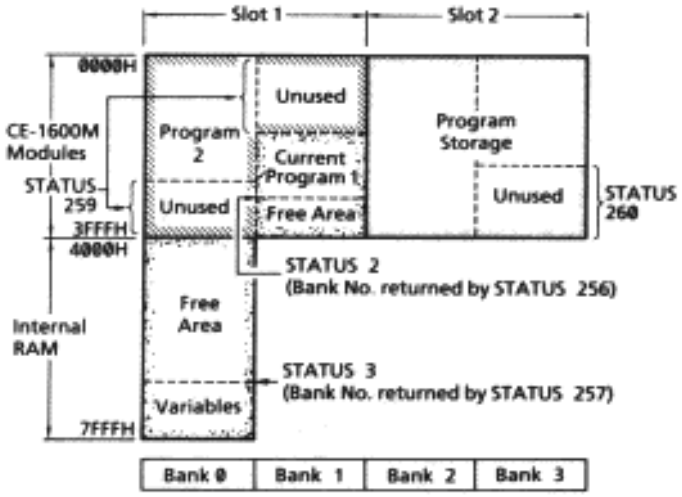


Abbildung 50 : Speicheraufteilung

STOP

Siehe auch : CONT, END

WIRKUNG : STOP dient der Unterbrechung eines Programmes, um so während der Testphase bei fehlerhaften Programmteilen anhalten und diese untersuchen zu können.

HINWEISE : Kommt ein STOP-Befehl zur Ausführung, wird das laufende Programm abgebrochen und eine Meldung angezeigt, die angibt, bei welcher Zeile dieser Abbruch erfolgte:

BREAK IN <Zeilennummer>

Nun kann durch Abfrage der Variablen und anderer Untersuchungen auf die Ursache der ermittelten Programmfehler geschlossen werden. Ebenso lassen sich nun den falsch belegten Variablen über die Tastatur die erwarteten Werte zuweisen und durch Start des Programmes mit der Anweisung CONT das weitere Verhalten getestet werden. Diese Fortsetzung ist aber nur möglich, wenn inzwischen keine Änderungen der Programmzeilen vorgenommen worden sind. -

Im Gegensatz zum Befehl END werden bei STOP keine geöffneten Dateien geschlossen.

BEISPIEL :

```
10:FOR N=1 TO 10
20:LET S=N*5
30:STOP
40:GRAPH
50:LINE (0,0)-(N,S)
60:NEXT N
```

STR$

Siehe auch : VAL

WIRKUNG : Die Funktion STR$ wandelt einen numerischen Wert in einen String um.

HINWEISE : Der gelieferte String setzt sich aus genau jenen Zeichen zusammen wie der numerische Wert auch. Jedoch hat der String keinen Wert, mit dem sich Berechnungen durchführen lassen.

Die Funktion STR$ kann man als die Umkehrung der VAL-Funktion ansehen.

Ist der numerische Wert negativ, enthält auch der String das betreffende Vorzeichen.

Ist der numerische Wert zu groß, um ihn mit zehn Ziffern darstellen zu können, erscheint er auch im String in der Fließkomma-Schreibweise.

BEISPIEL :

```
:
110:N=N*3
120:A$=STR$ (N)
130:B$=LEFT$ (A$,1)
140:M=VAL (B$)
:
```

SQR

WIRKUNG : Die Funktion SQR(X) liefert den positiven Wert der Quadratwurzel von X.

HINWEISE : Das Argument X darf ein beliebiger numerischer Ausdruck sein, dessen Wert jedoch im positiven Zahlen-Bereich liegen oder Null lauten muß. Negative Werte sind nicht erlaubt und führen daher zur Ausgabe eines ERROR-Codes.

Anstelle des funktionalen Operators SQR ist auch die Verwendung des Wurzelzeichens √ möglich. Dieses Zeichen ist in MODE 1 durch die offene eckige Klammer [ersetzt.

BEISPIELE :

```
>
SQR(5)
                    2.236067977
>

>
MODE 1
√2
                    1.414213562
>

>
MODE 0
[3
                    1.732050808
>
```

TAB

Siehe auch : LCURSOR, LPRINT

WIRKUNG : TAB bewegt im Text-Modus den Druckstift an die angegebene Druckspalte.

HINWEISE : Das TAB-Kommando verhält sich ähnlich wie das Kommando LCURSOR. Es bewegt den Druckstift an die angegebene Druckspalte, die im Bereich von 0 bis zum mit PCONSOLE festgelegten Maximalwert liegen kann.

Die Spaltenposition ist abhängig von der mit CSIZE eingestellten Zeichengröße.

TAB kann auch innerhalb einer LPRINT-Anweisung stehen. Im Gegensatz zu anderen BASIC-Versionen ist TAB aber nicht mit dem normalen PRINT-Befehl verwendbar.

BEISPIEL :

```
10:TEXT
20:LPRINT "LINKS";
30:TAB (38)
40:LPRINT "MITTE";
50:LPRINT TAB (70);"RECHTS"
60END
```

TAN

Siehe auch : ATN, COS, SIN

WIRKUNG : Die Funktion TAN liefert den Tangens eines als Winkel aufzufassenden nu=erischen Wertes.

HINWEISE : Der angegebene Winkel kann in Altgrad, Bogenmaß oder Neugrad vorliegen. Damit der richtige Wert der Tangensfunktion geliefert wird, muß sich der Computer im entsprechenden Winkelmodus befinden:

Winkelmaß	Winkelmodus
Altgrad	DEGREE
Bogenmaß	RADIAN
Neugrad	GRAD

Da die Tangensfunktion sogenannte Polstellenaufweist, an denen der Wert ins Unendliche geht, wird bei diesen Winkelargumenten (z.B. 90) ein ERROR-Code ausgegeben.

BEISPIEL :

```
10:DEGREE
20:PAUSE "WINKEL SIND IN ALTRGRAD !"
30:PAUSE "WINKEL: 0, TANGENS:";TAN(0)
40:PAUSE "WINKEL: 45, TANGENS:";TAN(45)
50:PAUSE "WINKEL: 90, TANGENS:";TAN(90)
```

```
>RUN
WINKEL SIND IN ALTGRAD !
WINKEL: 0, TANGENS: 0
WINKEL: 46, TANGENS: 1
WINKEL: 90, TANGENS:
ERROR 37 IN 70      (Taste CL drücken !)
```

TEST

WIRKUNG : TEST prüft, ob alle vier Druckstifte funktionsfunktionsfähig sind, indem mit jeder Farbe vier kleine Quadrate beschrieben werden.

HINWEISE : Die Quadrate werden nacheinander mit folgenden Farben gezeichnet:

schwarz, blau, rot und grün.

Anschließend wird der Drucker automatisch in den TEXT-Modus versetzt.

Um den Test zu unterbrechen, ist die BREAK-Taste zu betätigen.

TEXT

Siehe auch : GRAPH

WIRKUNG : TEXT versetzt den Drucker in den Text-Modus.

HINWEISE : Bei Aktivierung des Text-Modus wird die Größe der Zeichen auf den Wert 2 von CSIZE eingestellt und der Koordinatenursprung auf die dem linken Rand der momentanen Zeile am nächsten liegenden Position festgelegt.

TEXT setzt die mit den Parametern <Limit 1> und <Limit 2> gemachten Einstellungen des Befehles PAPER auf die Standardwerte zurück.

TIME

Siehe auch : TIME$

WIRKUNG : TIME$ setzt oder liefert Datum und Uhrzeit der im Computer eingebauten Echtzeit-Uhr.

HINWEISE : TIME ohne Parameterangabe liefert das momentane Datum und die momentane Uhrzeit im nachstehenden Format:

MMDDHH.mmss

Dabei bedeuten die hier gezeigten Platzhalter folgendes:

MM	Monat	01 bis 12
DD	Tag	01 bis 31
HH	Stunde	00 bis 23
mm	Minute	00 bis 59
ss	Sekunde	00 bis 59

Um Datum und Uhrzeit zu setzen, müssen diese im eben gezeigten Format der numerischen Variable TIME zugewiesen werden:

```
TIME = MMDDHH.mmss
```

Die Echtzeit-Uhr läuft auch dann weiter, wenn der Computer ausgeschaltet ist. Da das Datum ohne die Jahresangabe erfolgt, kann somit bei den Monatstagen der Einfloß eines Schaltjahres nicht berücksichtigt werden.

BEISPIEL : TIME=031220.3215

Diese Anweisung stellt die Echtzeit-Uhr auf 2O Uhr, 32 Minuten und 15 Sekunden des 12. März.

TIME$

Siehe auch : DATES$, TIME$ ON/OFF/STOP, ON TIME$ GOSUB

WIRKUNG : TIME$ ist eine Systemvariable, die die momentane Uhrzeit der im Computer eingebauten Echtzeit-Uhr enthält.

HINWEISE : TIME$ ohne Parameterangabe liefert einen String, der die Uhrzeit im folgenden Format angibt:

HH:MM:SS

Dabei bedeuten:

HH	die Stunden	von 00 bis 23
MM	die Minuten	von 00 bis 59
SS	die Sekunden	von 00 bis 59

Die Uhrzeit kann gestellt bzw. gesetzt werden, indem man der Systemvariablen TIMES einen String im eben beschriebenen Format zuweist:

TIME$="HH:MM:SS"

BEISPIEL :

```
>
TIME$="09:15:00"        Uhrzeit setzen
09:15:00
:                       Geraume Zeit warten
:
TIME$                   Zeit abfragen
09:15:06                Aktuelle Zeitangabe
>
```

TIME$ ON/OFF/STOP

Siehe auch : TIME$, ON TIME$ GOSUB

WIRKUNG : Diese Form von TIME$ erlaubt oder verbietet die Ausführung von ON TIME$ GOSUB Anweisungen.

HINWEISE : **TIME$ OFF**

verhindert, daß mit der Anweisung ON TIME$ GOSUB in ein Unterprogramm verzweigt werden kann.

TIME$ ON

erlaubt die Annahme der Anweisung ON TIME$ GOSUB und damit zeitabhängige Verzweigungen.

TIME$ STOP

verhindert ebenso wie TIME$ OFF die Ausführung von ON TIME$ GOSUB Anweisungen. Ist jedoch die Verzweigungsbedingung erfüllt, das heißt die vereinbarte Zeit erreicht, wird diese Tatsache in einem Zwischenspeicher vermerkt und bei der nächsten TIME$ ON Anweisung die Verzweigung vorgenommen. STOP ist die standardmäßig angenommene Option für TIME$.

TITLE

WIRKUNG : TITLE selektiert eines der ROM-Module oder den Arbeitsspeicher.

HINWEISE : In den auf der Rückseite des Computers befindlichen Modul-Fächern können insgesamt bis zu zwei RAM-Module installiert werden. Mit TITLE kann entschieden werden, welches Modul hiervon als Programmspeicher dienen soll oder ob hierfür wieder der interne Arbeitsspeicher des PC-1600 zu verwenden ist.

TITLE "S0:" selektiert den Arbeitsspeicher des PC-1600 als Programmspeicher. Dies ist auch die standardmäßige Annahme nach einem Total-Reset.

TITLE "S1:" selektiert das Modul des Faches S1

und

TITLE "S2:" das RAM-Modul des Faches S2 als Programmspeicher.

Ist das selektierte Modul als RAM-Disk initialisiert, wird ein ERROR-Code ausgegeben.

In einigen Situationen kann es nach Ausführung von TITLE vorkommen, daß während einer Programmunterbrechung die Tasten ^ und V ihre Wirkung zur Auflistung von Programmzeilen verlieren. Benutzen Sie in diesem Falle den LIST-Befehl oder lassen Sie das unterbrochene Programm einmal voll bis zum Ende durchlaufen, damit diese Tastenfunktion wieder hergestellt wird.

TRON / TROFF

Siehe auch : CONT, STOP

WIRKUNG : Mit TRON und TROFF kann der TRACE-Modus einbzw. ausgeschaltet werden.

HINWEISE : Ist dieser Modus über den Befehl TRON aktiviert, hält der Computer nach Ausführung einer jeden BASIC-Zeile für die Dauer einer halben Sekunde an und gibt die Nummer der betreffenden Zeile auf der rechten Seite des Displays aus.

Wird während einer solchen Pause, in der die Anzeige der Zeilennummer erfolgt, die Taste gedrückt, so geht der Computer in den Einzelschritt-Modus über. Die Taste muß nun jedesmal erneut betätigt werden, damit die folgende Programmzeile zur Ausführung kommt. Wird diese Taste dauernd gedrückt, arbeitet der Computer die Programmzeilen nacheinander ab, ohne die zugehörigen Zeilennummern anzuzeigen. Eine gerade abgearbeitete Zeile kann durch Betätigung der Tasten und sichtbar gemacht werden. Mit der Taste läßt sich die Ausführung des Programmes dann nach Ablauf einer Wartezeit von 0.5 Sekunden weiter fortsetzen.

Stoppt das Programm infolge eines PRINT- oder INPUT-Befehles, kann es durch die Betätigung der ENTER Taste weitergefahren werden.

Hält das Programm aufgrund eines STOP-Befehles an oder wurde es durch die BREAK-Taste abgebrochen, kann mit der CA Taste oder der Taste das Programm wieder in Betrieb genommen werden.

TRON / TROFF

TROFF schaltet den Trace-Modus aus.

TRON und TROFF können auch programmiert werden. Der Trace-Modus bleibt dann solange bestehen, bis im Programm der nächste TROFF-Befehl vorgefunden wird.

BEISPIEL :

```
10:TRON
20:FOR I=1 TO 3
30:PRINT "I=",I
40:NEXT I
50:TROFF
60:END

>
RUN

                    10
                    20
I=                  1
                    30
                    40
I=                  2
                    30
                    40
I=                  3
>
```

USING

Siehe auch : PAUSE, PRINT, PRINT#, LPRINT

WIRKUNG : USING erlaubt eine formatierte Datenausgabe.

HINWEISE : Das Format wird durch einen **<Format-String>** bestimmt, der sich aus folgenden Zeichen zusammensetzen kann:

- # Nummernzeichen
- & Kaufmannsund
- + Pluszeichen
- * Multiplikations-Operator
- ^ Potenz-Operator
- , Komma
- . Dezimalpunkt

Mit diesen Zeichen lassen sieh so vielgestaltige Format-Strings bilden, daß es unmöglich ist,alle zulässigen Strings in einem Syntax-Diagramm zu erfassen. Aus diesem Grunde soll es hier lediglich a1s Richtschnur dienen, wie man die Zeichen zu sinnvollen Strings kombiniert. Darüberhinaus gibt es eine Vielzahl weiterer Kombinationsmöglichkeiten, die dieselbe Wirkung haben, aber teilweise unanschaulich sind.

USING kann als parametrische Anweisung innerhalb einer PAUSE-, PRINT- ,PRINT#- oder auch LPRINT Anweisung verwendet werden.

Wird USING nicht als Parameter der genannten Befehle verwendet, sondern als alleinstehende Anweisung, gilt damit das damit eingestellte Format für alle folgenden PRINT- und LPRINT- und PAUSE-Anweisungen, bis es durch ein neues Format überschrieben wird. USING ohne Parameter, also vollkommen alleinstehend verwendet, hebt das Format auf.

USING

Folgende acht grundlegende Formate lassen sich unterscheiden:

(1) "###"	**Integer-Format:**
43	Bei drei Nummernzeichen lassen
-57	sich Integer-Werte mit bis zu
-2	zwei Stellen angezeigen. Ein Dezimalpunkt erscheint nicht.
(2) ###."	**Fixkomma-Integer-Format:**
98.	In diesem Format werden alle
-43.	Werte mit einem Dezimalpunkt
-9.	versehen.
(3) "###.##"	**Fixkomma-Format:**
64.29	Es werden grundsätzlich zwei
5.00	Stellen hinter dem Dezimalpunkt
-13.44	ausgegeben und vor diesem bis
-2.09	zwei Stellen dargestellt.
(4) "##.## "	**Fließkomma-Format:**
23.11E 05	In diesem Format wird der Wert
-4.33E-02	einschließlich dem Exponenten
7.00E 00	zeichen E und einem unter Umständen vorzeichenbehafteten Exponenten angezeigt. Exponent, Exponentsymbol und Exponentvorzeichen belegen in jedem Falle vier Zeichen-Plätze.
(5) "###,###."	**Integer-Tausender-Format**
34,567.	Es können fünf Ziffern einer
2.	Integer-Zahl angezeigt werden.
-230.	Die Tausenderstellen werden da-
2,345.	bei durch ein Komma abgetrennt.
(6) "+####"	**Vorzeichenbehaftetes Format**
-983	Es sind Integer-Werte mit bis
+211	zu drei Ziffern darstellbar. Jeder Wert wird mit seinem Vorzeichen versehen.

(7) "*####" **Gefülltes Integer-Format**
2345 Es sind bis zu vier Ziffern
**22 einer Integer-Zahl darstellbar.
*230 Führende Lücken werden,durch
-**66 einen Stern aufgefüllt.

(8) "&&&&&&" **String-Format**
ABCDEF Es kann ein String mit bis zu
GHI sechs Zeichen linksbündig ange
zeigt werden.

Die maximale Anzahl der Formatsymbole # oder *
beträgt in den Formaten (1), (2), (3), (4), (6)
und (7) 11 und im Format (5) 14.
Der PC-1600 gibt ohne Formatierungs-String die
numerischen Daten mit bis zu 1O signifik.anten
Stellen aus.

BEISPIELE :

```
lO:H$="ABCDEF":WAIT 64
20:USING "&&":PRINT H$
30:PRINT USING "&&&&";H$
40:PRINT H$: USING:PRINT H$

RUN
AB
ABC
ABC
ABCDEF
>
10:Z=1234
20: PAUSE USING "#####";Z
30:PAUSE USING "#####.";Z
40:PAUSE USINC "#####.##";Z
5O:PAUSE USING "+####.##^";Z
RUN
                          1234
                          1234.
                          1234.00
                          +1.23E 03
```

Siehe auch : HEX$, STR$

WIRKUNG : VAL WANDELT einen String in einen numerischen Wert um.

HINWEISE : Die VAL-Funktion kann als Umkehrung der beiden Funktionen STR$ und HEX$ angesehen werden. Sie konvertiert einen String, der aus numerischen Zeichen besteht, in einen numerischen Wert.

Ist das Funktionsargument ein dezimaler String, muß er aus den Zeichen 0 bis 9 zusammengesetzt sein. Er darf einen Dezimalpunkt und die Angabe eines Exponenten enthalten sowie ein Vorzeichen für die Mantisse und eines für den Exponenten aufweisen. In einem solchen Falle entspricht VAL genau der Umkehrung von STR$.

Ist das Funktionsargument dagegen ein hexadezimaler String, muß das erste Stringzeichen ein & sein und die nachfolgenden Zeichen aus solchen Symbolen bestehen, die zur Darstellung von Hex-Ziffern verwendet werden. In diesem Falle wirkt VAL als Umkehrung der HEX$-Funktion.

Enthält das Argument unzulässige Zeichen, wird der Funktionswert 0 geliefert.

BEISPIEL :

```
10:INPUT "FREQUENZ = ";A$
15:IF ASC(A$)<48 OR ASC(A$)>57 THEN 100
20:F=VAL(A$)
30:PRINT F
40:END
:
100:PRINT "NUR ZAHLENEINGABE ERLAUBT"
```

WAIT

Siehe auch : PRINT

WIRKUNG : WAIT bestimmt die Zeitdauer, die nach Ausführung einer PRINT-Anweisung abzuwarten ist, bevor mit der nächsten Anweisung fortgefahren wird.

HINWEISE : WAIT ohne Parameterangabe setzt die Wartezeit auf unendlich. Nach einer PRINT-Anweisung stoppt das Programm also vollständig. Wie man aber aufgrund des angezeigten Bereitschaftszeichens > erkennen kann, ist der Programmablauf aber nicht abgebrochen. Er pausiert lediglich und kann mit Betätigung der ENTER-Taste fortgesetzt werden.

Wird der Parameter **<Dauer>** angegeben, bestimmt dieser die Wartezeit als ein Vielfaches einer 1/64 Sekunde. Der Wert 64 ergibt also eine Pause von 1 Sekunde, der Wert 128 eine von 2 Sekunden usw. Der erlaubte Wertebereich erstreckt sich von 0 bis 65535.

Zusätzlich kann eine der Optionen P oder S angegeben werden. Dieses ist jedoch ausschließlich im MODE 0 (PC-1600-Modus) möglich und auch nur dann, wenn der Parameter <Dauer> vorhanden ist.

Die Option P entspricht der standardmäßigen Einstellung und kann, wenn nicht Option S gewünscht ist, auch weggelassen werden. Sie bedeutet, daß die Pause unmittelbar nach jeder PRINT-Anweisung stattfindet.

Die Option S bestimmt, daß die Pause nur dann erfolgt, wenn das Display in allen vier Zeilen mit Daten gefüllt ist und die Absicht hat zu "Scrollen", d.h. die Inhalte der Zeilen um je eine Zeile nach oben versetzen möchte.

Wird ein Programm mit RUN gestartet, hängen die standardmäßigen WAIT-Einstellungen vom gültigen Anzeigemodus ab:

MODE 0 (PC-1600-Modus): Keine Pause bei Ausführung einer PRINT-Anweisung.

MODE 1 (PC-1500-Modus): Programmausführung pausiert für eine unbestimmte Zeit und muß durch Betätigung der ENTER -Taste fortgesetzt werden.

BEISPIEL :

```
10:FOR I=1 TO 10
20:WAIT (64*I)
30:PRINT "*";
40:NEXT I
50:WAIT
60:END

>
RUN
**********
>
```

Jeder Stern erscheint 1 Sekunde später als der vorhergehende.

WAKE$

Siehe auch : KBUFF$

WIRKUNG : WAKE$ bestimmt die Uhr-Zeit, bei der sich der ausgeschaltete Computer selbsttätig einschaltet und welche Befehl er dann abarbeiten soll.

HINWEISE :

WAKE$(0)="<Zeit>:<Befehl>"

bestimmt, zu welcher Zeit der Computer sich einschalten und welchen Befehl er anschließend ausführen soll. In Verbindung mit dem KBUFF$-Befehl erlaubt dieses die automatische Abarbeitung von "Batch-Dateien" zu einer vorgegebenen Zeit. Der Parameter <Zeit> muß dabei im Format:

MM/DD/HH/mm (Monat/Tag/Stunde/Minute) vorliegen.

WAKE$(0)""

setzt WAKE(O)="<Zeit>:<Befehl>" außer Kraft.

WAKE$(1)="<Befehl>"

sorgt dafür, daß sich der Computer genau dann einschaltet und mit der Ausführung des BefehlsStrings beginnt, wenn am RS-232C-Interface das Signal CI (Pin 9) in den Zustand "high" geht. (Siehe auch Teil III, Kapitel 6)

WAKE$(1)""

schaltet WAKE$(1)="<Befehl>" wieder ab.

Die Länge des Befehls-Strings kann maximal 26 Zeichen betragen. Soll der Befehl nicht nur in der Anzeige erscheinen, sondern auch tatsachlich ausgeführt werden, muß an den WAKE$-String ein "carriage return" angehängt werden. Dieses kann mit der CHR$-Funktion geschehen (s. Beispiel).

Ist der Computer bereits eingeschaltet, wenn ein WAKE$-Befehl aktiviert wird, ertönt ein Alarmsignal.

BEISPIEL : Wenn Sie das folgende Programm eingeben:

```
1O:PRINT "Guten Morgen !"
20:PRINT "Schau doch einmal nach, ob"
30:PRINT "der Nikolaus schon da war."
40:END
```

und anschließend die Anweisung:

```
WAKE$(0)="12/06/07/30:RUN"+CHR$(&D)
```

erteilen und den Computer ausschalten, können Sie sich am Nikolaustag mit einer passenden Meldung wecken lassen.

XCALL

Siehe auch : NEW, POKE, XPOKE

WIRKUNG : XCALL ruft ein Maschinensprache-Programm auf.

HINWEISE : XCALL und CALL sind grundsätzlich zwei gleiche Befehle. Im Gegensatz zum CALL-Befehl dient der Befehl XCALL jedoch zum Start von zum PC-1500 kompatiblen Maschinensprache-Programmen.

Es gilt also:

XCALL ruft eine PC-1500-kompatible Routine auf. CALL ruft eine Z80-kompatible Routine auf.

Beachten Sie, daß der Befehl XCALL des PC-1600 dem Befehl CALL des PC-1500 entspricht.

Bei Aufruf der Maschinensprache-Routine kann ein einzelner Variablen-Wert übergeben werden. Nach Ablauf des Maschinensprache-Programmes wird dann der aktualisierte Wert an diese Variable zurückgegeben. Für die Wert-Übergabe stehen die beiden Prozessor-Register A und X bereit. Damit eine Wertübergabe erfolgen kann, muß die vereinbarte Variable natürlich auch wirklich existieren.

<Adresse> bestimmt die Adresse in der momentan gültigen Speicherbank, ab der das Programm zu starten ist.

<Variable> bestimmt die Variable, deren Wert an das Maschtnensprache-Programm Übergeben werden soll. Handelt es sich hierbei um eine numerische Variable, muß diese vom Typ Integer sein und einen Wert im Bereich von -32768 bis 32767 aufweisen. Ihr Wert wird dann an das X-Register abgegeben und nach Ablauf des Programmes der dann im Register stehende Wert der Variablen zugewiesen.

Bei einer String-Variablen wird an an Register X die Adresse übergeben, ab der der String abgelegt ist und in Register A die Länge des Strings eingetragen.

BEISPIEL :

```
:
:
400:XCALL 57405,X
410:PRINT ,,DER VOM MASCHINENSPRACHE-"
420:PRINT "PROGRAMM GELIEFERTE WERT"
430:PRINT "LAUTET : ";X
:
:
```

XPEEK

Siehe auch : PEEK, POKE, XPOKE

WIRKUNG : XPEEK liest ein Byte aus einer spezifizierten Speicherzelle.

HINWEISE : XPEEK und PEEK sind von ihrer Wirkungsweise her grundsätzlich identisch. XPEEK hält sich jedoch an das beim PC-1500 übliche Format des dortigen PEEK-Befehles. XPEEK des PC-1600 ist also nichts weiter als ein Äquivalent des PC-1500-Befehles PEEK.

<Adresse> bestimmt eine Adresse im Speicherbereich 0 (ME 0).

#<Adresse> bestimmt eine Adresse im Speicherbereich 1 (ME 1). Hierbei gilt die momentan selektierte Speicherbank.

Weitere Einzelheiten hierzu können Sie aus der in Anhang D gezeigten Aufteilung des Speichers erkennen.

XPOKE

Siehe auch : PEEK, POKE, XPEEK

WIRKUNG : XPOKE schreibt ein Byte in die spezifizierte Speicheradresse oder mehrere Bytes mit dieser Adresse beginnend in nacheinander folgende Speicherzellen.

HINWEISE : Die Befehle XPOKE und POKE sind in ihrer Wirkung grundsätzlich gleich. XPOKE stellt jedoch ein Äquivalent des POKE-Befehles dar, wie er vom PC-1500 verstanden wird.

<Adresse>	bestimmt die Adresse im Speicher bereich 0 (ME 0), in die das erste Byte der angegebenen Byte-Liste geschrieben werden soll.
#<Adresse>	bestimmt die Adresse im Speicher bereich 1 (ME 1), in die das erste Byte der angegebenen Byte-Liste zu schreiben ist.
<Byte>	bestimmt den jeweils gerade in die aktuelle Speicherzelle zu ladenden Wert zwischen 0 und 255.

Weitere Einzelheiten hierzu können Sie der im Anhang D beschriebenen Aufteilung des Speicherbereiches entnehmen.