B A S I C - S Y S T E M F Ü R D E N L G P - 3 0

INHALTSVERZEICHNIS

1. ALLGEMEINES

Mit dem BASIC - Grundsystem , das auf einem Lochstreifen ge - speichert ist , können die meisten anfallenden Probleme gelöst werden . Lediglich die speziellen Fünktionsunterprogramme (Wurzel , Logarithmus etc.) , das Ausstanzprogramm und die Zusatzfunktionen (siehe 4.5) sind nicht im Grundsystem ent - halten und müssen zusätzlich eingelesen werden .
Das Grundsystem wird wie das Verkehrsprogramm "10.4" mit einem "Bootsstrap"-Hilfsprogramm eingelesen und ist dann sofort einsatz - fähig .

2. ARBEITEN MIT DEM BASIC - SYSTEM

2.1 Der Systemmodus

Im Systemmodus kann der Benützer Systembefehle eingeben , die eine bestimmte Funktion auslösen . Der Systemmodus wird folgendermaßen aufgerufen :

1) Taste "Einzeloperation" drücken 2) Taste "Zähler löschen" dr.
3) Taste "Normal" drücken 4) Taste "Start" drücken

Der Computer meldet sich darauf mit "READY".

Systembefehle :

2.1.1 Der "Clear" - Befehl

Durch den "Clear"-Befehl wird der Speicherplatz , der für die BASIC - Programme reserviert ist , gelöscht und der Compiler auf ein neues Programm vorbereitet . "Clear" ist immer vor der Eingabe eines neuen Programmes und nach dem Neueinlesen des Systems zu geben .

2.1.2 Der "Manual" - Befehl

Der "Manual"-Befehl muß gegeben werden , wenn ein BASIC - Programm oder einzelne BASIC - Anweisungen von Hand eingegeben werden sollen .

Vorgehen : "Manual" tippen , "Start" drücken : Der Computer meldet sich mit "6-bit" Taste "6 Bit Eingabe" drücken und Wagenrücklauftaste an der Schreibmaschine betätigen: Der Computer meldet sich mit "+" und zeigt damit an , daß er sich im Programmodus befindet .

2.1.3 Der "Normal" - Befehl

Der "Normal"Befehl muß gegeben werden , wenn ein BASIC-Programm vom Lochstreifen eingelesen werden soll . Das Vorgehen ist das Selbe wie bei "Manual" . Nach der Meldung "6-bit" muß der Auswahlschalter am Fotoleser auf die Stellung "Abtaster" gebracht werden .

2.1.4 Der "run" - Befehl

Der "Run"-Befehl bewirkt , daß das eingegebene und compilierte BASIC - Programm durchlaufen wird . Der Programmbefehl "END" bewirkt die Rückkehr in den Systemmodus . Der Programmdurchlauf darf nur auf folgende Weise unterbrochen werden : Taste "Sprung" am Computer drücken und warten bis der Computer "ready" druckt . Dann Taste wieder lösen .Wurde der Programmaublauf auf andere Weise unterbrochen , so ist nachträglich folgendes Verfahren anzuwenden , um das System vor Schaden zu bewahren :

1) Systemmodus aufrufen 2) Taste "Sprung" drücken 3) System - befehl "Run" geben 4) Nach der Meldung "Ready" "Sprung"-Taste lösen .

2.1.5 Der "Get" - Befehl

Mit dem "Get"-Befehl werden die Funktionsunterprogramme , das Ausstanzprogramm und die anderen Zusätze eingelesen
Vorgehen : "Get" tippen , Auswahlschalter am Fotoleser ggf. umschalten , "Start" drücken .
Das Einlesen eines Lochstreifens sollte nicht unterbrochen werden .

2.1.6 Der "Save" - Befehl

Der "Save"-Befehl bewirkt , daß das gespeicherte BASIC - Programm mit allen Unterprogrammen hexadezimal ausgestanzt wird . Die hexadezimalen Streifen können dann später mit dem "Get" Befehl schnell wieder eingelesen werden und weiterverwertet werden . Das "Save" - Ausstanzprogramm ist nicht im Grundsystem enthalten sondern muß zusätzlich eingelesen werden . Dabei wird ein Teil des Systems zerstört . Zur Erleichterung der Wiedererneuerung des Systems ist der Teil , der zerstört wird auf dem gleichen Streifen mitgelocht .
Vorgehen 1) "Save"-Programm mit "Get" einlesen (Das Einlesen wird nach dem Lesen des halben Streifens beendet) 2) System - befehl "Save" geben und Auswahlschalter am Motorlocher betätigen 3)"Start" drücken (Programm wird ausgestanzt) 4) Nach Beendigung des Ausstanzens "Start" drücken und Auswahlschalter auf Stellung "Schreibmaschine" bringen 5) "Get" tippen und den Rest des "Save"-Programmes einlesen (System ist wieder betriebsbereit)

2.2 Der Programmodus

Im Programmodus kann der Benützer BASIC - Anweisungen eingeben . Bei Eingabe von Hand (Manual) führt der Computer nach jeder vollständigen Anweisung einen Wagenrücklauf durch und druckt "+" (Bereitschaft , eine neue Anweisung zu empfangen) . Bei der Eingabe eines Lochstreifens (Normal) wird das ganze Programm ohne Unterbrechung eingelesen .
Verlassen des Programmodus : Von Hand : Einmal zusätzlich "Start" drücken . Lochstreifen : Dem letzten Befehl einen zusätzlichen Stoppcode anhängen . Der Computer meldet sich darauf mit "6-bit away" und hält an . Darauf ist die Taste "6 Bit Eingabe" zu lösen und der Auswahlschalter am Fotoleser ggf. umzuschalten.

3. PROGRAMMIEREN MIT BASIC

3.1 Allgemeines

3.1.1 Abweichungen von anderen BASIC - Systemen

Das BASIC für den LGP-30 unterscheidet sich in einigen Punkten von anderen BASIC - Versionen. Einige Befehle wurden neu geschaffen, andere wurden erweitert oder eingeschränkt bzw. weggelassen. Deshalb sollte man bei der Umwandlung von BASIC - Programmen anderer Computer in das LGP-30-BASIC vorsichtig vorgehen. Neben diesen größeren Unterschieden müssen einige formale Regeln beachtet werden, die durch den Aufbau des LGP-30 bedingt sind.

1) Nach jedem Wort muß auf dem Lochstreifen ein "'" (Stoppcode) gelocht werden bzw. bei Eingabe von Hand ein "Start" gegeben werden.
   z.b: 10 print a,b,c * 10'print'a'b'c''
2) Die Leertaste hat bei gedrückter "6 BIT Eingabe"-Taste nicht die Funktion eines Blanks, sondern stellt eine Zahl dar. Sie darf deshalb im Programm nicht eingefügt werden, um etwa das Programm übersichtlicher zu gestalten. Zu diesem Zwecke muss der Tabulatorsprung benützt werden.
3) Klammern: Es werden grundsätzlich die eckigen Klammern "[ ]" und nicht die runden "( )" verwendet.

3.1.2 Befehlsformate

Die meisten BASIC-Befehle bestehen aus vier Teilen

z.B:	10'	print'	['a'b'c'd'e'']'	'
	(1)	(2)	(3)	(4)

(1): Die Zeilennummer oder Adresse einer Anweisung. Durch die Zeilennummern werden die einzelnen Anweisungen geordnet. Dadurch wird die richtige Reihenfolge der Anweisungen hergestellt.

(2): Der Anweisungsstamm: Er legt die Art des Befehles fest.

(3): Daten, Adressen etc.: Sie sind die Variablen der Anweisung und legen ihre Funktionsweise fest (nicht bei END und STOP)

(4): Das Schlusszeichen: Es zeigt dem Compiler, dass die Anweisung zu Ende ist und dass die nächste folgt.

3.2 DATENTYPEN

Es gibt in BASIC zwei Datentypen.

a. Numerische Daten

Numerische Daten können im Programm in drei verschiedenen Formaten erklärt werden:

- Ganzzahlen (alle positiven und negative ganze Zahlen)
  Format: z.B: 12' 12345' -77' (bis zu 5 Zeichen+VZ)
- Zahlen in halblogarithmischer Form (alle rationalen Zahlen)
  Beispiele: 1.'e'3' ≙ 1000 1.234'e'-78 ≙ $1{,}234 \cdot 10^{-78}$
  -3.141'59265'e'0' ≙ -3,14159265
  Regeln: Jede Zahl muß einen Dezimalpunkt besitzen, die erste Zelle darf 5 Zeichen + Vorzeichen haben, die Zweite muß aus 5 Zeichen bestehen, Für die Mantisse gilt: $10 > |m| \geq 1$, Für den Exponenten gelten die Regeln der Ganzzahlen.
- Konstanten
  Vier Konstanten können jederzeit aufgerufen werden.
  Format: /x,'≙ 3,14.. /e,'≙ 2,71.. /2,'≙1,41 /3,'≙ 1,7.
  Beispiel: 10' let' h'='a'/└2'*'/3,'' Höhe eines gleichs. 3-Ecks

Numerische Daten werden intern alle in ein einheitliches Format umgewandelt und können daher miteinander verknüpft werden.

b. Alphanumerische Daten

Alphanumerische Daten bestehen aus maximal 5 Zeichen:
Format: ['ABCDE']'

VARIABLEN

Jeder der oben beschriebenen Datentypen kann im Programmdurchlauf einer Variablen zugeordnet werden, wobei es keinen Unterschied zwischen alphanumerischen und numerischen Variablen gibt.

a. Normale Variablen

Der Variablenname einer normalen Variable besteht aus 4 Zeichen, wobei die Zeichen 1(=l), [ ] ' - / nicht zugelassen sind. Außerdem muß jede Variable einen Buchstaben enthalten.
Vor dem Programmdurchlauf werden normale Variablen nullgesetzt.

b. Indizierte Variablen

Eine indizierte Variable repräsentiert eine Zahl einer Tabelle oder Matrix (siehe 3.7) und darf aus 3 Zeichen bestehen. Die Indizes können normale Variablen oder Konstanten sein (nicht indizierte Variablen) und stehen in eckigen Klammern.
z.B: a['i']' a['i'j']' xyz['3't']' r32['w5u'4prc']'
Wichtig! Zwischen Variablennamen und "[" darf kein "'" stehen.

3.3 DATENEINGABE UND DATENAUSGABE

3.3.1 Dateneingabe

Mit dem Befehl für die Dateneingabe können während des Programmablaufs Daten eingegeben werden.

3.3.1.1 Eingabe von numerischen Daten

Numerische Daten werden mit dem INPUT-Befehl eingegeben.
Allgemeines Befehlsformat: input'a'b'c'...x'y'z''
Ein INPUT-Befehl darf maximal 31 Variablen enthalten.

Der INPUT-Befehl läßt zwei Eingabeformate zu.

a. Ganzzahlen: Format: vXXXXXXX (v=Vorzeichen)
   z.B: -3 54 -9999999
   Bei positiven Zahlen muss eine Leertaste als Vorzeichen getippt werden!
b. Zahlen in halblogarithmischer Form: Wenn man Zahlen in diesem Format eingeben will, muß man vor jeder Eingabe auf die Taste "Start" drücken ohne etwas einzugeben bzw. auf dem Lochstreifen einen Stoppcode vor die Zahl setzen.
   Format: vXXXXXXX (e)vYY Das YY ist der Exponent (siehe 3.2).
   Es müssen alle 7 Ziffern der Mantisse geschrieben werden!
   z.B: 3141592 e -3 entspricht (0,003141592)
   2500000 e 4 entspricht 25000

Nach jeder Eingabe führt der Rechner einen Wagenrücklauf durch.

3.3.1.2 Eingabe von alphanumerischen Daten

Alphanumerische Daten werden mit dem Befehl INPUT ALPHA eingegeben.
Allgemeines Befehlsformat: input'alpha'a'b'c'..Y'z''
Ein alphanumerisches Wort darf aus maximal 5 Zeichen bestehen.

3.3.2 Datenausgabe

Mit dem Befehl für die Datenausgabe können während des Programmablaufes Daten ausgegeben werden.

3.3.2.1 Ausdruck von Texten

Mit dem PRINT-Befehl können Texte ausgedruckt werden.
Allgemeines Befehlsformat: print'['x'y'z'...a'b'c'']'
druckt den Text: xyz...abc aus.
Wichtig! Vor der "[" muß immer ein zusätzliches "'" stehen.

Ein PRINT-Befehl kann beliebig viele Zeichen enthalten . Die Kombination WR-Tab-Tab-"'" wird nicht gespeichert . So können Zeichen auf mehreren Zeilen zu einem PRINT-Befehl zusammen - gefaßt werden . (siehe Beispielprogramm im 5.)
Für 7 Schreibmaschinenfunktionen müssen Sonderzeichen verwendet werden :

cr : Wagenrücklauf tab: Tabulatorsprung color : Farbumschaltung
gr : Großschreibung kl : Kleinschreibung ap : Hochkomma
back : Rücktaste

Beispiel: print' 'cr'cr'gr's'kl'o'n'd'e'r'z'eIi'c'h'e'n'color'
'ap'back'.' '' '' druckt : Sonderzeichen!

3.3.2.2 Ausdruck numerischer Daten
Numerische Daten werden ebenfalls mit dem PRINT-Befehl ausgegeben .
Allgemeines Befehlsformat : print'a'b'c'...x'y'z''
druckt den Zahlenwert der Variablen a,b,c...x,y,z aus .
Das Format dercauszudruckenden Zahl kann frei gewählt werden (siehe SET-Befehl) . Ohne Formatangabe wird automatisch das Standardformat gewählt (6Stellen) .
Format der auszudruckenden Zahl z :

.XXXXXX	$0.1 \leq z < 1$
X.XXXXX	$1 \leq z < 10$
XX.XXXX	$10 \leq z < 100$
XXX.XXX	$100 \leq z < 1000$
XXXX.XX	$1000 \leq z < 10000$
XXXXX.X	$10000 \leq z < 100000$
XXXXXX.	$100000 \leq z < 1000000$
X.XXXXX e YY	$1000000 \leq z < 1$ e 100

Das Format wird je nach der Größe der Zahl automatisch so ge - wählt , daß eine möglichst große Anzahl von Nachkommastellen gedruckt wird

3.3.2.3 Alphanumerische Daten
Diese können nicht ausgedruckt werden , da bei der Eingabe 2 Bits jedes Zeichens verlorengehen .

3.3.2.4 Ausgabe über den Lochstreifenstanzer
Vor dem Programmdurchlauf ist die Taste KH 32 zu drücken und der Auswahlschalter am Locher umzuschalten .

3.3.2.5 Kombination von 3.3.2.1 und 3.3.2.2
Die Kombination ist möglich . z.B: print'a'b'['cr'']'x'y''

3.4 PROGRAMMVERZWEIGUNGEN ("SPRÜNGE")

3.4.1 Unbedingte Programmverzweigungen

Allgemeines Befehlsformat: goto'XXX''

Funktion: Als nächste Anweisung wird die Anweisung in Zeile XXX ausgeführt . (Sprung nach XXX)

XXX darf höchstens 310 sein .

3.4.2 Bedingte Programmverzweigungen

Funktion: Der Computer testet , ob eine bestimmte Bedingung erfüllt ist oder nicht . Ist sie erfüllt , so springt er zur angegebenen Zeile , ist sie nicht erfüllt , so fährt er im Programmablauf fort .

Allgemeines Befehlsformat: if'a'R'b'goto'XXX''

R ist eine Relation . Es stehen 6 Relationen zur Verfügung:

"=" Gleichheit
"not" Ungleichheit
"max" größer (>)
"min" kleiner(<)
"max=" größergleich (≥)
"min=" kleinergleich (≤)

z.B:

```
10'    if'    p'max'q'goto'40''
20'    if'    u'not'v'goto'50''
```

3.4.3 Indirekte Programmverzweigungen

Allgemeines Befehlsformat: goto'var'u'' bzw. if'a'R'b'goto'var'u'

Funktion: Als Sprungziel wird der Inhalt der Variablen u genommen .

3.5 VARIABLENZUWEISUNGEN

Eine Variablenzuweisung weist einer Variablen einen numerischen oder alphanumerischen Wert zu .

3.5.1 Die LET-Anweisung

Eine LET-Anweisung weist einer Variablen entweder den Wert einer anderen Variablen zu (a) oder das Verknüpfungsprodukt mehrerer anderer Variablen (b) .

(a): Allgemeines Befehlsformat: let'a'='b''
Funktion: Die Variable b wird der Variablen a zugewiesen.

(b): Bei Anweisungen dieser Art wird der linken Variable das Verknüpfungsprodukt der rechten Operanden zugewiesen .
Beispiel: let'a'='b'+'c'-'d'*'e'/'f'**'g''

(" " : Multiplikation " " : Division " " : Potenzierung)
Jeder Operator hat einen Vorrang . Die Operationen werden der Reihe nach , beim höchsten Vorrang beginnend , ausgeführt .

Vorränge :	"+"	und "-"	Vorrang 1
	"*"	und "/"	Vorrang 2
	"**"		Vorrang 3

Durch Setzen von Klammern können eigene Prioritäten gesetzt werden . Es gelten die üblichen Rechenregeln .
z.B: let'a'='['b'+'c']'*'['d'+'e']''
Es dürfen maximal 7 Klammern ineinander verschachtelt werden .
Die Zahl der sich öffnenden Klammern muß gleich der Zahl der sich schließenden Klammern sein .
Neben den 5 Grundrechnungsarten stehen noch eine Reihe von zusätzlichen Funktionen zur Verfügung :

"-" Dreht das Vorzeichen einer Zahl um
"abs" Berechnet den absoluten Wert einer Zahl (Betragsfunktion)
"rad" Rechnet einen Winkel ins Bogenmaß um
"deg" Rechnet einen Winkel ins Gradmaß um
"int" Rundet eine Zahl zur nächsten ganzen Zahl auf
"sgn" Ersetzt eine positive Zahl durch "1" , eine negative durch "-1"
Die Null wird nicht verändert .
"sqr" Berechnet die Quadratwurzel einer positiven Zahl .
"sin" Berechnet den Sinus einer Zahl (Bogenmaß)
"cos" Cosinus
"ln" Berechnet den natürlichen Logarithmus einer positiven Zahl
"log" Berechnet den dekadischen Logarithmus
"exp" Potenziert e mit einer Zahl
"atn" Berechnet den Arcustangens einer Zahl (Ergebnis im Bogenmaß)

Beispiel: let'a'='-'['b']'+'sin'['c'+'exp'['e'**'f']'*'e']''

Im BASIC - Grundsystem sind einige dieser Funktionen nicht enthalten sondern müssen zusätzlich eingelesen werden . Es gibt 4 Zusatzlochstreifen (BASIC-Ergänzungsstreifen BES) , die mit "hex" eingelesen werden . Die BES müssen vor dem Einlesen des Programmes eingelesen werden ! Das Einlesen der BES darf nicht unterbrochen werden!

BES 1 :	sqr
BES 2 :	sin,cos
BES 3 :	ln,log,exp,**
BES 4 :	atn

Die BES beschränken den Datenspeicher für BASIC - Programme (4.5)

3.5.2 Übertragung von Datenblöcken

Ein Datenblock wird durch den Befehl DATA definiert
Allgemeines Befehlsformat : data'a'b'c'..x'y'z''
Der Datenblock kann aus Variablen , Zahlen und alphanumerischen Daten (maximal 31) bestehen . In einem Programm dürfen maximal 6 DATA-Anweisungen stehen (Erweiterung siehe 4.5)

Die Daten der Datenblöcke können durch READ-Anweisungen beliebigen Variablen zugeordnet werden .
Allgemeines Befehlsformat : read'a'b'c'..x'y'z''

```
Beispiel :      100'   data'   10'20'['nein']'kon1'a['24']''
                101'   data'   1''
                110    read'   a'b'c''
                120    read'   x'y'z['1'u']''
```

Resultat: a=10 b=20 c="nein" x=kon1 y=a(24) z(1,u)=1

Mit dem Befehl RESTORE kann erreicht werden , daß der nächste READ-Befehl Daten vom ersten Datenblock liest .
Allgemeines Befehlsformat : restore''

3.6 UNTERPROGRAMMTECHNIK

Unterprogramme (Prozeduren) werden mit dem Befehl GOSUB aufgerufen .
Allgemeines Befehlsformat : gosub'XXX''
XXX ist eine Zeilennummer (kleiner 310) und bedeutet den Anfang des Unterprogramms .
Funktion : a) Es erfolgt ein Sprung nach XXX

Ein Unterprogramm wird mit dem Befehl RETURN beendet .
Allgemeines Befehlsformat : return''
Funktion : b) Es erfolgt ein Rücksprung ins Hauptprogramm zu der , dem Gosub-Befehl folgenden Anweisung .

```
Beispiel :      10'    gosub'   100''      Sprung nach 100
                20'    gosub'   100''          -"-
                ........
                100'   print'   ['a'b'c'']''
                110'   return''              Sprung nach 1)20 2)30
```

Das Programm bewirkt den Ausdruck von "abcabc" .

Mit der Kombination von DATA-READ mit GOSUB-RETURN können Prozeduren mit Parameterübergabe realisiert werden .
Bei einer solchen Prozedur wird die Prozedur auf eine Reihe von aktuellen Parametern angewandt , die beim Prozeduraufruf einer gleichen Anzahl von prozedurinternen Variablen zugeordnet werden :

Beispiel :

```
10'    data'    2'3'4''
20'    gosub'   100''
30'    data'    7'8'9''
40     gosub'   100''
....
100'   read'    x'y'z''
110'   let'     a$='x'+'y'+'z''
120'   print'   a''
130'   return''
```

Die Prozedur (100 bis 130) wird einmal auf die Parameter "2,3,4" , das zweite Mal auf die Parameter "7,8,9" angewandt . Es werden die Zahlen "9" und "24" ausgedruckt .

3.7 TABELLEN UND MATRIZEN

Tabellen und Matrizen haben die Form:

$a_1 , a_2 , a_3 \ \ldots\ a_n$

$$\begin{matrix} a_{11} , a_{12} , a_{13} \ \ldots\ a_{1n} \\ a_{21} , a_{22} , a_{23} \ \ldots\ a_{2n} \\ \ldots \\ a_{m1} , a_{m2} , a_{m3} \ \ldots\ a_{mn} \end{matrix}$$

Tabellen und Matrizen müssen definiert werden bevor mit ihnen gearbeitet werden kann .
Das geschieht durch den DIM-Befehl .
Allgemeines Befehlsformat :

```
1-fach    dim'a['n']''
2-fach    dim'a['n'm']''
```

n=Zeilenzahl m=Spaltenzahl

1) n und m dürfen nur Zahlen sein
2) Die Feldvariable darf aus maximal 3 Zeichen bestehen
3) Zwischen Feldvariable und "[" darf kein "'" stehen
4) Mit einer DIM-Anweisung kann nur eine Tabelle oder Matrix definiert werden .

Regeln für indizierte Variablen siehe 3.2

3.8 PROGRAMMIERUNG VON SCHLEIFEN

Die Schleifenanweisungen erlauben es, einen Programmabschnitt mit verschiedenen Werten einer Variablen zu durchlaufen. Der Wert der Variablen wird bei jedem Durchlauf gemäß den Angaben in der FOR-Anweisung geändert.

Allgemeines Befehlsformat: for'x'='a'to'b'step'c''

Funktion: Die Variable x erhält den Anfangswert a. Bei jedem Durchlauf wird der Wert von x um c erhöht bis der Wert von b erreicht ist. Fehlt die Schrittweite (step c) im FOR-Befehl, so wird automatisch c=1 gesetzt.

Die Schleife wird durch den Befehl NEXT begrenzt.

Allgemeines Befehlsformat: next'at'XXX''

XXX ist die Zeilennummer des dazugehörigen FOR-Befehls.

Beispiel:

```
10'  for'x'='0'to'10''
20'  let'y'='x'*'x''
30'  print'y''
40'  next'at'10''
```

Druckt die Quadratzahlen bis 10 aus.

Wichtig:

1) Eine Schleife kann verlassen werden, bevor a=b ist.
2) Die Adresse XXX des NEXT-Befehls muß einen FOR-Befehl enthalten.
3) Eine Schleife darf mehrere NEXT-Befehle enthalten.
4) Soll der Anfangswert der Schleifenvariablen ein anderer sein, als im FOR-Befehl angegeben, so kann der FOR - Befehl beim ersten Mal übersprungen werden.
5) FOR_NEXT-Schleifen dürfen verschachtelt aber nicht verschränkt werden.
6) Erfolgt während eines Schleifendurchlaufs ein Sprung zum FOR-Befehl, ohne daß NEXT durchlaufen wurde, so wird beim Anfangswert angefangen.

3.9 SONSTIGE BEFEHLE

3.9.1 Die END-Anweisung

Allgemeines Befehlsformat: end''

Der END-Befehl ist das logische Ende des Programmes. Wird dieserv Befehl durchlaufen, so erfolgt ein Rücksprung in den System - modus. In einem Programm können mehrere END-Befehle stehen.

3.9.2 Die LINE-Anweisung

Allgemeines Befehlsformat: line'x''

Der Befehl veranlaßt die Ausführung von X (kleiner 10) Wagenrück - läufen.

3.9.3 Die SET-Anweisung

Allgemeines Befehlsformat: set'x''

Nach diesem Befehl werden alle Zahlen vom PRINT-Befehl mit x (maximal 9) Ziffern ausgedruckt bis der nächste SET-Befehl durch - laufen wird. Ohne SET-Befehl wird mit 6 Ziffern gedruckt.

3.9.4 Die STOP-Anweisung

Allgemeines Befehlsformat: stop''

Der Rechenvorgang wird unterbrochen wenn KH8 nicht gedrückt ist.

3.9.5 Die USE-Anweisung

Allgemeines Befehlsformat: use'XXX''

Dieser Befehl bewirkt, daß der Befehl in XXX abkopiert wird und in der Zeile des USE-Befehls gespeichert wird.
Wichtig! Wird nach dem Abkopieren der Befehl in XXX gelöscht, so bleibt er in der USE-Anweisung erhalten.

3.9.6 Hilfsbefehle

Diese Befehle werden nicht gespeichert und haben die Zeilennummer 0.

a) REM-Befehl Format: rem'abcdefghijklmnopqrstuvwxyz''
   Dient zum Anbringen von Merktexten auf dem Lochstreifen
b) WAIT-Befehl Format: wait''
   Unterbricht die Compilierphase. Weitercompilieren durch Drücken von "Start".

3.10 Separates Blatt

4. HINWEISE ZUM PROGRAMMIEREN

4.1 Korrekturen am Programm

Oft ist es nötig , an einem Programm nachträglich Korrekturen vorzunehmen .

a. Eine Anweisung soll gelöscht werden:
Dies ist mit Hilfe des "delete" Befehls möglich . Der "delete" Befehl wird im Programmodus gegeben . Darauf verlangt der Computer die Zeilennummern der zu löschenden Anweisungen . Nach jeder eingegebenen Zeilennummer druckt er eine Leertaste . Nach der letzten Zeilennummer ist ein zusätzlicher "Start" zu geben worauf die Rückkehr in den Programmodus erfolgt .

b. Eine neue Zeilennummer soll über die Alte geschrieben werden . Dazu kopiert man den Befehl mit der USE-Anweisung ab und löscht die alte Zeile mit "delete" .

c. Eine neue Anweisung soll über die Alte geschrieben werden . Das ist ohne Weiteres möglich , indem man einfach die neue Anweisung eintippt . Die alte wird automatisch gelöscht . Zu beachten ist , daß der Speicherplatz , den die alte Anweisung belegte , verlorengeht . Im Falle einer Speicherplatzknappheit ist es deshalb zweckmäßig , den Lochstreifen mit dem BASIC-Programm auszubessern und neu einzulesen .

4.2 Verbesserung von Zeit- und Speicherplatzbedarf

Eine Optimierung der Rechenzeit ist nur in beschränktem Maße möglich:

a. Verkleinerung der Zwischenräume zwischen den Anweisungen . Die Erkennung eines Zwischenraumes erfordert eine Rechenzeit von ca. 60 ms .Sind die Anweisungen in den beliebten Zehnerschritten nummeriert , so gehen bei jeder Anweisung 0,54 s verloren ! Es ist deshalb am günstigsten , die Endfassung eines Programmes in Einerschritten zu nummerieren .

b. Vermeiden von Doppeloperationen . Bei der Berechnung arithmetischer Ausdrücke sollte man Doppelberechnungen vermeiden , wie das Beispiel zeigt :

```
nicht :     10'   let'   a'='sin'['x'**'y'+'z']''
            20'   let'   b'='cos'['x'**'y'+'z']''
sondern:    10'   let'   c'='x'**'y'+'z''
            11'   let'   a'='sin'['c']''
            12'   let'   b'='cos'['c']''
```

Eine Verminderung des Speicherplatzbedarfs wird immer dann notwendig sein, wenn durch einen "error 13" angezeigt wird, daß das Programm zu lange ist! Man geht dann so vor:

1) Wurden Anweisungen nachträglich verändert oder gelöscht? Wenn ja: Lochstreifen verbessern und ganz neu einlesen.
2) Programmlänge durch geschicktes Zusammenfassen von Anweisungen in Unterprogrammen verkürzen. Identische Anweisungen durch USE-Befehle ersetzen (kein Speicherplatzverbrauch)
3) Überflüssige Texte weglassen.
4) Werden BES überflüssigerweise verwendet? Wenn ja: Weglassen!

Wenn diese Maßnahmen nicht ausreichen, kann man den Speicherplatz mit dem Change-Befehl umorganisieren (siehe 4.5)

4.3 Fehlersuche

Wenn der Benützer bei der Programmierung Fehler macht, so wird dies in einigen Fällen durch eine "error"-Meldung angezeigt.

a. Errors in der Compilierphase

error 10	Der Schlußcode einer Anweisung fehlt
error 12	Die Zeilennummer einer Anweisung ist zu groß
error 13	Der Speicherplatz ist voll.
error 14	Die Zeilennummer einer Anweisung fehlt oder es wurde ein ungültiger Befehl gelesen.
error 16	Das Programm enthält zuviele Variablen.
error 18	Eine LET-Anweisung hat mehr als 32 Zeichen
error 19	Eine LET-Anweisung ist fehlerhaft (z.B: zwei aufeinanderfolgende Variablen)
error 20	Klammerfehler bei LET.

b. Errors beim Programmdurchlauf

error 51	Division durch Null.
error 52	Indizes einer indizierten Vaiable zu groß.
error 53	Argument der INT-Funktion zu groß
error 55	Argument von SQR negativ.
error 56	Argument von LN/LOG negativ oder Null
error 57	Argument von EXP oder ** zu groß. Überlauf!
error 58	Argument von LOG zu groß. (10^{64})
error 59	Argument von SIN/COS zu groß.

c. Errors im System

error 70	Ein BES fehlt.

error 99 Undefinierbarer Fehler

Zu jeder Fehlermeldung wird die Zeilennummer der fehlerhaften Zeile ausgedruckt . Sie kann bei error 14,70 und 99 falsch sein .

z.B: error 17 -00000002. = Error 17 in Zeile 2

4.4 Ausstanzen fertiger Programme

Vorgehen: Siehe Seite 3 unter "save"

Mit "save" ausgestanzte Programme können nach dem Wiedereinlesen mit "get" ohne irgendwelche Einschränkungen weiterverwendet werden .

4.5 "Change"-Unterprogramm - Änderungen für spezielle Zwecke

Das "Change"-Programm erlaubt es , den Speicherplatz für die BASIC - Programme optimal auszunützen .

Vorgehen :
1) "Change"-Programm mit "get" einlesen
2) "Change" tippen
3) Die verlangten Daten eintippen (Kommentar wird gedruckt)
4) Rückkehr in den Systemmodus - Das "Change"-Programm wird gelöscht (nur einmal verwendbar)
5) Programm einlesen

Das Change-Programm erlaubt es , die Speicherplatzverteilung individuell zu regeln und erzeugt eine optimale Verteilung für jedes Programm .

Bei Unklarheiten stehe ich Ihnen jederzeit zur Verfügung :

JOHN BANHART
73 ESSLINGEN-BERKHEIM
BRÜHLSTR. 32

5. BEISPIELPROGRAMM

```
 10'       line'     2''
  0'       rem'      Berechnung der Zahl e nach einer Taylorreihe''
 20'       print'    ['gr'b'kl'e'r'e'c'h'n'u'n'g' 'v'o'n' 'e'
                     ' 'n'a'c'h' 'e'i'n'e'r' 'gr't'kl'a'y'l'o'r'
                     ' 'r'ei'h'e'cr'cr'']''
 30'       data'     1'1''
 40'       read'     e'f''
 50'       print'    ['n'tab'n'!'tab'w'e'r't'e' 'f'u'e'r' 'e'']''
 60'       data'     0'10''
 70'       read'     a'b''
 80'       for'      n'='a'to'b''
 90'       let'      f'='f'*'n''
100'       let'      e'='e'+'1'/'f''
110'       print'    ['cr'n]'n'f'e''
120'       next'     at'80''
130'       print'    ['cr'gr'w'kl'o'l'l'e'n' 'gr's'kl'i'e' 'i'n'
                     ' 'e'i'n'e'm' 's'p'e'z'i'e'l'l'e'n' '
                     'gr'b'kl'e'r'e'i'c'h' 'gr'w'kl'e'r't'e' '
                     b'e'r'e'c'h'n'e'n'gr'?'kl'']''
140'       input'    antw''
150'       if'       antw'='['ja']'goto'170''
160'       end''
170'       data'     1'1''
180'       read'     e'f''
190'       input'    a'b''
200'       goto'     80''
```

```
ready  run
Berechnung der Zahl e nach einer Taylorreihe
n          n!        Werte für e
1.00000    1.00000   2.00000
2.00000    2.00000   2.50000
             .....
10.0000    3.62800 e 06   2.71828
Wollen Sie in einem speziellen Bereich  Werte berechnen?
Ja
 0
15
....
```