K C – P A S C A L

(Version 4.4)

+++ AM87 +++

Technische Universitaet Karl-Marx-Stadt
Sektion Verarbeitungstechnik
WB Verarbeitungsmaschinen
Vertiefungseinrichtung Medizientechnik

Karl-Marx-Stadt, Juli 1987

Bearbeiter:

Dipl.-Ing. Albrecht Mugler

Inhaltsverzeichnis

1. Einleitung

1.1. Zum Programm

KC-Pascal ist eine schnelle, leicht bedienbare und leistungsfaehige PASCAL-Version fuer die Kleincomputer KC85/1, KC85/2 und KC85/3, sowie fuer alle CP/M-kompatiblen Systeme auf Computern wie A5120, A5130 und PC1715. Sie koennen jedes auf einem Kleincomputertyp einlesen und compilieren. Dadurch sind Sie in der Lage, lauffaehige Maschinenprogramme fuer alle Kleincomputertypen zu erzeugen (s. T-Kommando.
Die Version fuer CP/M-kompatible Computer soll Ihnen gestatten, auf komfortableren Systemen KC-PASCAL-Programme zu erstellen, zu testen, zu drucken und zu speichern. Dabei werden alle kassettenorientierten Funktionen der Kleincomputer mittels Diskettenspeicher realisiert.

Dem vorliegenden Compiler liegt die Version HP4S (HISOFT) zugrunde. Abweichungen von der im "PASCAL User Manual and Report" (Jensen/Wirth) gegebenen Beschreibung sind:

- FILEs sind nicht implementiert, aber Variablen koennen auf Band gespeichert werden.
- Ein RECORD-Typ darf keinen VARIANT-Teil enthalten.
- PROCEDUREs und FUNKRTIONs sind nicht als Parameter zugelassen.

Gegenueber den Vorgaengerversionen PASCAL V4.2 und PASCAL V4.3 wurden neue Funktionen eingebaut (Compiler Option P, Editor-Kommandos V, X und Z). Mit dem Kommando '0' ist es jetzt moeglich, TURBO-PASCAL-Quellen im KC weiterzuverarbeiten. Darueberhinaus koennen Sie jetzt mit 32 Kbyte oder 48 Kbyte Arbeitsspeicher entwickelte und mittels Kommando T uebersetzte Programme auch auf Systemen mit nur 16 Kbyte abarbeiten (ab Version 4.3; es sei denn, das .COM-File ist laenger).
Alle System- und Fehlermeldungen erscheinen in deutscher Sprache.
Wesentliche Verbesserungen sind durch die neuen vordefinierten Funktionen und Prozeduren zur Manipulation von INTEGER-Zahlen sowie fuer die graphische Arbeit und durch die vordefinierte Konstane Pi gegeben (PLOT, CLRPLOT, SETC, GETC, GOTOXY, KEYPRESSED, PI, SWAP, HI, LO usw.)

Weiterhin wurde der Editor verbessert und der Zufallszahlengenerator durch die leistungsfaehigere Version ersetzt.

1.2. Was gehoert zu KC-PASCAL

PASENTRY.COM	KC-PASCAL Kaltstart fuer CP/M-kompatible Systeme (Diskette)
PASREC.COM	KC-PASCAL Warmstart fuer CP/M-kompatible Systeme (Diskette)
PASCAL41.COM	KC-PASCAL fuer KC85/1 (Kassette)
PASCAL42.COM	KC-PASCAL fuer KC85/2 und KC85/3 (Kassette) (wie oben, aber mit SETC, GETC, PLOT, CLRPLOT)
PASCAL43.COM	KC-PASCAL fuer KC85/3 (Kassette) (wie oben, aber zusaetzlich LINEPLOT, CIRCLE)

Weitere Versionen fuer andere Computer auf Z80/U880-Basis sind in Vorbereitung (Z1013, AC1). Darueberhinaus ist eine Version verfuegbar, die ueber eine spezielle Sprungtabelle an Computer mit unterschiedlichester Hardwarekonfiguration anpassbar ist.

1.3. Die Struktur

KC-PASCAL besteht aus verschiedenen Modulen und Tabellenbereichen. Die Reihenfolge der Programmteile und Befeiche ist nachfolgend dargestellt.

Laufzeitmodul (Runtimes)
Puffer fuer mit T uebersetzte Programme
Editor
Compiler
Quelltext
mittles C uebersetzte Programme
Variablenstack

Der Compiler benoetigt einschieszlich Editor und Laufzeitmodul etwa 20 Kbyte Speicherplatz.
Das Laufzeitmodul umfaszt einschlieszlich der Systemanpassung und einiger Pufferbereiche ca. 5 Kbyte.

1.4. Laden von KC-PASCAL

Bei Verwendung eines KC85/1 ist der Einsatz eines RAM-Erweiterungsmodules 690 003.5 von 16 Kbyte erforderlich. Ein weiteres RAM-Modul (insgesamt 32 Kbyte Zusatz-RAM) wird automatisch erkannt und kann zur Vergroeszerung des Arbeitsspeichers eingesetzt werden. Der Compiler wird ab Adresse 0300H in den Arbeitsspeicher geladen.
Beim KC85/2 oder KC85/3 ist eine Speichererweiterung M022 (16 Kbyte) im Modulschacht 8 erforderlich. Ein weiteres Modul M022 kann in den Modulschacht C gesteckt werden, womit sich insgesamt 48 Kbyte Speicher nutzen lassen (etwa 1000 PASCAL-Anweisungszeilen). Das zusaetzliche RAM-Modul musz vor Verwendung aktiviert werden. Dazu wird im Betriebssystem folgendes Kommando eingegeben:

```
SWITCH C 83
```

Der Compiler wird von Kassette ab Adresse 0200H in den Arbeitsspeicher des KC85/2 oder KC85/3 geladen.

Die CP/M-kompatible Version wird wie ueblich durch Eingabe des Namens aufgerufen (PASENTRY, PASREC).

1.5. Start von KC-PASCAL

Der Start des KC-PASCAL erfolgt ueber das Kommando PASENTRY. War der Compiler bereits gestartet und soll vorhandener Quelltext nicht geloescht werden (z.B. nach zeitweiliger Rueckkehr ins Betriebssystem), kann man mit dem Kommando PASREC (auf Betriebssystemebene) in KC-PASCAL zurueckkehren.

1.6. Besondere Steuertasten

Folgende Tasten besitzen eine besondere oder abweichende Bedeutung in KC-PASCAL:

KC85/1	KC85/2/3	CP/M	Wirkung
ENTER	ENTER	ENTER	– Zeile abschlieszen
<——	<——	CTL-H	– loescht letztes eingegebenes Zeichen
^	^	CTL-I	- zur naechsten TAB-Position

CL LN	Sh. DEL	CTL-Y	- loescht gesamte Zeile
CLS	Sh. HOME	CTL-L	- loescht den Bildschirm
STOP	BREAK	CTL-C	- unterbricht das Listen oder Bricht das Programm ab
CTL-P	HOME	CTL-P	- Ausgabe auf Drucker Ein/Aus
INS	Sh. Space	[	- eckige Klammer auf
DEL	¯ |	]	- eckige Klammer zu

Alle Anweisungen muessen **in Groszsschreibung** angegeben werden. Beachten Sie bitte, dasz beim KC85/2 Kleinbuchstaben zwar eingegeben werden koennen, aber als Groszbuchstaben angezeigt werden. Der Compiler unterscheidet Namen streng nach Grosz- und Kleinschreibung. Darueberhinaus werden die Zeichen '[' und ']' bei den Typen KC85/2 und KC85/3 nicht standardgemaesz dargestellt.
Ein Ladeprogramm fuer im KC-PASCAL-Format auf Band aufgenommene Daten und Programme ist im Laufzeitmodul des KC-PASCAL enthalten.

1.7. Die Druckerschnittstelle

KC85/1
Der Drucker ist vor Aufruf zu aktivieren (s. Handbuch KC85/1). Durch Betaetigen von CTL-P wird der Drucker parallel zur Bildschirmausgabe geschalten.

KC85/2, KC85/3
Ein Drucker zur Ausgabe von Programmlistings oder anderen Daten kann ueber einen USER-Kanal des KC85/2 oder KC85/3 erfolgen. Dafuer ist ein Modul M003 erforderlich. Das Treiberprogramm entnehmen Sie bitte dem Heft "Kleincomputer KC85 – Beschreibung zu M003 V24".
Dazu sollte man das Modul in Schacht "C" stecken (M022 in Schacht "8"). Das Einschalten des linken Kanales erfolgt ueber das Kommando (auf Betriebssystemebene):

```
V24 C 1 2 <ENTER>
```

Ist das Listing eines PASCAL-Programmes auf den Drucker ausgegeben werden, wird die Taste "HOME" betaetigt. Nochmaliges Betaetigen schaltet den Drucker wieder ab. Die Ausgabe auf den Bildschirm bleibt erhalten.
Beim Compilieren wird der Drucker immer deaktiviert (s. auch Compiler-Option P).

CP/M
Der Drucker wird wie ueblich mit CTL-P aktiviert.

KC85/1, KC85/2, KC85/3, CP/M
Soll der Drucker innerhalb eines Programmes ein- oder ausgeschalten werden, geschieht das mit dem Steuerzeichen 16 (10H) wie folgt:

```
    .....
100 WRITE(CHR(16));
110 WRITELN('Jetzt ist der Drucker eingeschaltet!');
120 WRITE(CHR(16));
130 WRITELN('...und jetzt wieder aus!');
    .....
```

1.8. Compilieren und Starten

Der Compiler erzeugt nach dem Aufruf ein Listing der folgenden Form:

```
aaaa nnnn Quelltext-Zeile
```

aaaa ist die Adresse, an der der aus dieser Zeile generierte
Code beginnt
nnnn ist die Zeilen-Nr. (fuehrende Nullen werden unterdrueckt)

Das Listen kann durch BREAK unterbrochen werden, jede andere Taste setzt das Listen fort.
Wird waehrend des Compilierens ein Fehler festgestellt, erscheint die Meldung '*FEHLER*', gefolgt von einem Pfeil, der unter dem Fehler erzeugter Zeichen steht, und einer Fehlernummer (siehe Fehlerliste). Das Listen wird gestoppt, 'E' ermoeglicht das Editieren der angezeigten Zeile 'P' das Editieren der vorhergehenden Zeile (falls sie existiert), jede andere Taste setzt den Compilerlauf fort.
Treten waherend des Copmilierens sehr viele Fehler auf, sollten Sie unbedingt 'E' betaetigen und mit ENTER in den Editor zurueckkehren. Sonst kann es vorkommen, dasz der Compiler nicht wieder in den Editor "zurueckfindet". Erwarten Sie bitte nicht, dasz der Compiler alle Ihre Fehler exakt behandelt, sondern ueberpruefen Sie gegebenenfalls vorher Ihre Anweisungen!
Wird waehrend dem Compilieren eine Taste betaetigt, folgt eine Unterbrechung. Mit BREAK (CTL-C) erfolgt der Abbruch, jede andere Taste setzt den Compilerlauf fort.
Falls das Programm fehlerhaft endet (z. B. ohne 'END.'), erscheint die Meldung 'Kein Text mehr!', die Steuerung wird an den Editor uebergeben.
Wenn die Compilierung erfolgreich abgeschlossen wird, das Programm aber Fehler enthaelt, wird die Zahl der ermittelten Fehler angezeigt und der Objekt-Code geloescht. Nach fehlerfreier Compilierung erscheint die Frage 'Lauf?'. Soll das Programm sofort gestartet werden, musz mit 'J' geantwortet werden, sonst wird die Steuerung an den Editor uebergeben.
Die beim Lauf des Objekt-Codes moeglichen Runtime-Fehler sind der entsprechenden Liste zu entnehmen.
Der Programmlauf kann mittels einer beliebigen Taste unterbrochen werden. Anschlieszendes BREAK bricht den Lauf ab, jede andere Taste setzt ihn fort.

2. **Syntax**

Soweit nicht ausdruecklich anders angegeben, entspricht die Implementierung der im "PASCAL User Manual and Report" (Jensen/ Wirth Second Edition) vorgegebenen.
Nur die ersten 10 Zeichen sind signifikant. Kleinbuchstaben werden nicht in Groszbuchstaben umgewandelt, so dasz z. B. die Namen HALLO, HAllo und hallo verschieden sind.
Reservierte Worte und vordefinierte Namen duerfen nur in Groszbuchstaben eingegeben werden.

2.1. **Typisierung**

PASCAL erfordert eine relative strenge, durch den Nutzer festzulegende Typisierung (Definition) der verschiedenen Datenelemente. Es gibt in verschiedenen PASCAL-Implementierungen zwei verschiedenen Arten der Typisierung:
Strukturaequivalenz und Namenaequivalenz.
KC-PASCAL benutzt die Namenaequivalenz fuer RECORDs und ARRAYs.
Beispiel: Zwei Variablen sind wie folgt definiert

```
VAR  A: ARRAY['A'..'C'] OF INTEGER
     B: ARRAY['A'..'C'] OF INTEGER
```

Man koennte annehmen, dasz man nun A:=B; schreiben koennte, aber dies wuerde zu '*FEHLER* 10' fuehren, da durch obige Definition zwei getrennte 'TYPErecords' erzeugt werden. D.h. der Programmierer hat entschieden, dasz A und B nicht demselben Datentyp angehoeren.
Nur ein Datenbtyp wird erzeugt durch:

```
VAR A,B : ARRAY['A'.. 'C'] OF INTEGER;
```

Jetzt kann zu A B zugewiesen werden und umgekehrt. Obwohl dieses Namenaequivalenzprinzip etwas komplizierter erscheint, werden durch die geforderte groeszere Gewissenhaftigkeit beim Programmieren die Fehler eingeschraenkt.

2.2. **Vorzeichenlose Zahl**

Ganze Zahlen (INTEGERs) haben in KC-PASCAL einen Absolutwert kleiner gleich 32767. Groeszere Zahlen werden als reelle Zahlen (REALs) betrachtet. Die Mantisse von REALs ist 23 Bit lang. Die Genauigkeit ist demzufolge etwa 7 signifikante Stellen. Beachten Sie bitte, dasz die Genauigkeit einer Berechnung abnimmt, wenn der Absolutwert des Ergbnisses viel kleiner als der der Argumente ist, z.B. liefert 2.00002-2 nicht genau 0.00002. Das ruehrt von der Ungenauigkeit in der Darstellung von Dezimalbruechen als Binaerbrueche her. Dieser Fall tritt nicht auf, wenn ganze Zahlen maesziger Groessz als REALs dargestellt werden, z.B. wird 200002-200000 = 2 exakt berechnet.
Die groeszte moegliche REAL-Zahl ist 3.4E38, die kleinste 5.9E-39. Bei der Beschreibung einer Zahl werden nur die Werte der ersten 7 Ziffern betrachtet, bei den weiteren zaehlt nur der Stellenwert. Fuehrende Nullen sollten vermieden werden, da diese als Ziffer gelten und somit die Genauigkeit beeintraechtigen (z.B. ist 0.000123456 ungenauer als
1.23456E-4).
Hexadezimalzahlen sind verfuegbar, um z.B. Speicher-Adressen darzustellen. Beachten Sie, dasz nach '#' mindestens eine Hex-Ziffer stehen musz (sonst FEHLER 51).

2.3. **Vorzeichenlose Konstante**

Beachten Sie, dasz Strings (Zeichenketten) nicht mehr als 255 Zeichen enthalten duerfen. String-Typen sind ARRAY[1..N] OF CHAR, wobei N eine ganze Zahl von 1 bis 255 ist. Zeichenstrings duerfen kein End-of-line-Zeichen /CHR (13)/ enthalten, sonst tritt 'Fehler 68' auf.
Der gesamte erweiterte ASCII-Zeichensatz mit 256 Elementen ist zulaessig. Um die Kompatibilitaet mit Standard-Pascal aufrechtzuerhalten, wird das Null-Zeichen nicht als ' ' dargestellt, sondern als CHR (0).

2.4. **Konstante**

Die nicht standardgemaesze CHR-Konstruktion ist hier zugelassen, so dasz Konstanten auch als Steuerzeichen benutzt werden koennen. In diesem Fall musz die Konstante in den runden Klammern vom INTEGER-Typ sein.

```
z.B.: CONST         bs= CHR(8);  BACKSPACE
                    cr= CHR(13); CARRIGE RETURN
```

2.5. Einfacher Typ

Skalare Aufzaehlungstypen (name,name,...)'duerfen nicht mehr als 256 Elemente enthalten.

2.6. Typ

Das reservierte Wort PACKED wird akzeptiert aber ignoriert, da Packen bei ARRAYs OF CHAR usw. schon stattfindet. Der einzige Fall, in dem Packen von Arrays vorteilhaft waere, ist der bei ARRAYs OF BOOLEAN, aber in diesem Fall wird normalerweise ein SET verwendet, wenn Packen erforderlich ist.

2.7. Felder und Mengen

Der Grundtyp eines SETs kann bis zu 256 Elemente enthalten. Das ermoeglicht SETs of CHAR zusammen mit SETs von jedem Nutzer-Aufzaehlungstyp zu deklarieren. Beachten Sie, dasz nur Untermengen von INTERGERs als Grundtypen verwendet werden koennen. Alle Teilmengen von INTEGERs werden als SETs of 0..255 behandelt.
Zugelassen sind auch ARRAYs of ARRAYs, ARRAYs of SETs, RECORDs of SETs u.s.w.
Zwei Array-Typen werden nur dann als aequivalent betrachtet, wenn ihre Definition in ein und derselben Benutzung des reservierten Wertes ARRAY erfolgt. Folgende Typen sind somit nicht aequivalent:

```
TYPE
      tablea =  ARRAY[1..100] OF INTEGER;
      tableb =  ARRAY[1..100] OF INTEGER;
```

Eine Variable vom Typ tablea kann also nicht einer Variablen vom Typ tableb zugewiesen werden. Das ermoeglicht das Ermitteln von Fehlern, wenn die beiden Tabellen verschiedene Daten darstellen. Die o.g. Einschraenkung gilt nicht fuer den Spezialfall ARRAYs vom Stringtyp, da diese immer zur Darstellung von aehnlichen Daten verwendet werden.

2.8. Zeiger

KC-PASCAL erlaubt die Erzeugung dynamischer Variablen durch die Benutzung der Standardprozedur NEW. Im Gegensatz zur statischen Variablen, der ein Speicherplatz ueberall in dem Block zugewiesen ist, in der sie deklariert ist, kann auf die dynamische Variable nicht direkt mittels eines Namens Bezug genommen werden. Diese Zeigervariable ist eine statische Variable und enthaelt die Adresse der dynamischen Variablen. Auf diese wird mittels eines '^' hinter der Zeigervariablen zugegriffen.
Im KC-PASCAL gibt es folgende Einschraenkungen in der Benutzung von Zeigern:
Zeiger auf Typen, die nicht vereinbart wurden, sind nicht erlaubt. Das verhindert nicht die Erzeugung von verketteten Listen, da Typ-Definitionen Zeiger auf sich selbst enthalten duerfen, z.B.:

```
TYPE
   item = RECORD
          value:INTEGER;
          next:^item
          END;
```

```
link = ^item;
```

Zeiger auf Zeiger sind nicht erlaubt.
Zeiger auf den gleichen Typ werden als aequivalent betrachtet, z.B.:

```
VAR
   first:link;
   current:^item;
```

Die Variablen first und current sind aequivalent (d.h. Strukturaequivalenz wird benutzt) und duerfen einander zugewiesen oder verglichen werden.
Die vordefinierte Konstante NIL ist zugelassen. Wenn sie einer Zeigervariable zugewiesen wird, wird sie als keine Adressen enthaltend betrachtet.

2.9. **Records**

Die Implementierung von RECORDs, strukturierten Variablen, die sich aus einer festen Anzahl Bestandteilen, sogenannten Feldern (fields), zusammensetzen, entspricht dem Standard-PASCAL bis auf die Einschraenkung, dasz ein VARIANT-Teil der Feldliste nicht zugelassen ist.
Zwei RECORD-Typen werden nur als aequivalent betrachtet, wenn ihre Deklaration zum gleichen Zeitpunkt unter dem gleichen reservierten Wort RECORD erfolgt.
Die WITH-Anweisung kann verwendet werden, um den Zugriff zu den einzelnen Feldern innerhalb eines RECORDs in kompakterer Form zu ermoeglichen.

2.10. **Feldliste**

Wird im Zusammenhang mit RECORDs verwendet.

2.11. **Variable**

In KC-PASCAL sind sowohl statische als auch dynamische Variable zugelassen. Statische Variable werden explizit durch VAR deklariert, wobei fuer sie im gesamten Block, in dem sie deklariert sind, Speicherplatz zugewiesen wird.
Dynamische Variable werden jedoch waehrend der Programmierung durch die Prozedur NEW dynamisch erzeugt. Sie werden nicht explizit deklariert und koennen nicht durch einen Namen angesprochen werden. Auf sie wird indirekt mittels einer statischen Variablen vom Pointertyp, die die Adresse der dynamischen Variablen enthaelt, Bezug genommen.
Beim Beschreiben von Elementen mehrdimensionaler Arrays musz der Programmierer nicht die gleiche Art der Index-Beschreibung in der Bezugnahme verwenden, die in der Deklaration benutzt wurde.
z.B.: Wenn Var a als ARRAY[1..10] OF ARRAY[1..10] OF INTEGER deklariert wurde, kann sowohl mit a[1] [1] als auch mit a[1,1] auf das Element (1,1) des Arrays zugegriffen werden.

2.12. **Faktor**

Siehe Ausdruecke und Funktionen.

2.13. **Term**

Die Untergrenze einer Menge (SET) ist immer Null und die Mengengroesze ist immer das Maximum des Grundtyps des SETs. Somit belegt ein SET OF CHAR immer 32 Bytes (256 Elemente sind moeglich - ein Bit fuer jedes Element). Ebenso ist ein SET OF 0..10 aequivalent zu einem SET OF 0..255.

2.14. **Einfache Ausdrucke**

s. Term

2.15. **Ausdruck**

Bei der Benutzung von IN umfasst das SET immer den vollen Bereich des entsprechenden Grundtyps, nur bei INTEGER-Argumenten umfaszt der Grundtyp immer den Bereich [0..255].
Die obige Syntax wird beim Vergleich von Strings gleicher Laenge, Zeiger und allen skalaren Typen verwendet. SETs koennen durch >=, <=, <> oder = verglichen werden, Zeiger nur durch = oder <>.
Achtung:

> Vergleiche zwischen REAL- und INTEGER-Ausdruecken muessen so notiert werden, dasz der REAL-Ausdruck auf der linken Seite des Vergleiches steht.

2.16. **Parameterliste**

Auf den Doppelpunkt musz ein Typname folgen, sonst tritt 'FEHLER 44' auf.
Sowohl variable Parameter als auch Festwertparameter sind ohne Einschraenkungen zugelassen.
Nicht zugelassen sind Funktionen und Prozeduren.

2.17. **Anweisung**

CASE-Anweisung :

> Eine leere CASE-Liste ist nicht gestattet, d.h.

CASE OF END : ruft 'FEHLER 13' hervor.

ELSE (welches eine Alternative zu END darstellt) wird ausgefuehrt, wenn die geforderte CASE-Marke nicht existiert. Wenn mit END abgeschlossen und die geforderte CASE-Marke nicht gefunden wurde, setzt das Programm mit der auf END folgenden Anweisung fort.

2.18. **FOR - Anweisung**

Die Steuervariable einer FOR-Anweisung darf nur eine unstrukturierte Variable, kein Parameter sein.

2.19. **GOTO - Anweisung**

Es ist nur ein GOTO zu einer Marke moeglich, die sich im selben Block und in derselben Ebene befindet.
Marken muessen durch das reservierte Wort 'LABEL' in dem Block, in dem sie benutzt werden sollen, deklariert werden. Eine Marke besteht aus 1 bis 4 Zeichen. Wenn eine Anweisung markiert werden soll, musz die Marke vor der Anweisung stehen und mit einem Doppelpunkt ':' abgeschlossen werden.

2.20. **Vorwaerts - Bezuege**

Wie im Pascal User Manual and Report beschrieben, koennen Prozeduren und Funktionen verwendet werden, bevor sie deklariert wurden. Das ist moeglich durch das reservierte Wort FORWARD.

```
Beispiel:
PROCEDURE a(y:t) ; FORWARD; (Procedure a vorwaerts deklariert)
PROCEDURE b(x:t);           (Procedure b)
 BEGIN
 ....
 a(p);                      (Bezug auf Procedure a)
 ....
 END;
PROCEDURE a;                (aktuelle Deklaration Procedure a)
 BEGIN
 ....
 b(q);
 ....
 END;
```

Beachten Sie, dasz die Parameter und der Typ des Ergebnisses einer Prozedur zusammen mit FORWARD deklariert werden und in der eigentlichen Deklaration der Prozedur nicht wiederholt werden .

2.21. **Programme**

Da keine Files implementiert sind, existieren auch INPUT und OUTPUT nicht.

2.22. **Konstanten**

```
PI           – die Zahle Pi = 3.14159E+00 vom Type REAL
MAXINT       - die groeszte verfuegbare ganze Zahl, d.h. 32767
TRUE, FALSE  - die Konstanten vom Typ BOOLEAN
```

2.23. **Typen**

```
INTEGER
REAL
CHAR
BOOLEAN
```

3. **Prozeduren und Funktionen**

3.1. **Ein- und Ausgabe-Prozeduren**

WRITE

Die Prozedur WRITE wird verwendet, um Daten auf dem Bildschirm auszugeben.
Wenn der auszugebende Ausdruck vom CHARACTER-Typ ist, dann gibt WRITE (e) den 8-bit-Wert, der durch den Wert des Ausdruckes e dargestellt wird, am Bildschirm aus.

Beachte!
CHR (n) ergibt das Steuerzeichen n,

die moeglichen Steuerzeichen entnehmen Sie
bitte den KC85-Programmierhandbuechern

Allgemein gilt:

WRITE (P1, P2, ... Pn); entspricht:
BEGIN WRITE(P1);WRITE(P2);.....;WRITE(Pn) END;
Die Parameter P1,P2,...Pn koennen eine der folgenden Formen haben:

(e) oder (e:m) oder (e:m:n) oder (e:m:H)

wobei e, m, n Ausdruecke und H der unmittelbare Buchstabe ist.

5 Faelle sind zu betrachten:

1) e ist von INTEGER-Typ und (e) oder (e:m) wird benutzt:

Der Wert von e wird in einen Zeichenstring mit abschlieszendem Leerzeichen umgewandelt. Eine Verlaengerung des Strings mittels fuehrender Leerzeichen kann durch Angabe von m, welches die Gesamtlaenge des Strings angibt, erreicht werden. Wenn m nicht ausreichend ist, um e auszugeben oder m nicht vorhanden ist, dann wird e vollstaendig mit abschlieszendem Leerzeichen ausgegeben und m wird ignoriert. Wenn die durch m festgelegte Laenge der Laenge von e ohne nachfolgenden Leerzeichen entspricht, wird kein abschlieszendes Leerzeichen ausgegeben.

2) e ist vom INTEGER-Typ und (e:m:H) wird benutzt:

In diesem Fall erfolgt die Ausgabe hexadezimal. Falls m=1 oder m=2 ist, wird der Wert (e MOD 16^m) ausgegeben, d.h. die m hoechstwertigen Hex-Ziffern ausgegeben. Wenn m>4 ist, werden fuehrende Leerzeichen hinzugefuehrt. Fuehrende Nullen werden angefuegt, wo es notwendig ist.

Beispiel:

```
WRITE (1025:m:H);

m=1 outputs: 1
m=2 outputs: 01
m=3 outputs: 0401
m=4 outputs: 0401
m=5 outputs:  0401
```

3) e ist vom REAL-Typ und (e), (e:m) oder (e:m:n) wird benutzt

Der Wert von e wird in einen Zeichenstring, der eine reelle Zahl darstellt, umgewandelt. Das Format der Darstellung wird durch n festgelegt. Falls n nicht vorhanden ist, wird die Zahl in wissenschaftlicher Zahlendarstellung mit Mantisse und Exponent ausgegeben. Wenn die Zahl negativ ist, wird ein Minuszeichen vor der Mantisse, anderenfalls ein Leerzeichen ausgegeben. Die Zahl wird immer mit mindestens einer Nachkommastelle und mit maximal 5 Nachkommastellen ausgegeben. Der Exponent wird immer mit Vorzeichen notiert.
Daraus folgt, dasz die minimale Laenge der wissenschaftlichen Darstellung 8 Zeichen betraegt. Wenn m < 8 ist, wird die

vollstaendige Darstellung von 12 Zeichen genommen. Wenn 8 <= m <= 12 ist, werden mehr oder weniger Dezimalstellen ausgegeben. Ist m > 12, werden fuehrende Leerzeichen angefuegt :

Beispiel:

```
WRITE(-1.23E 10:m);
m=7            ergibt: -1.23000E+10
m=8            ergibt: -1.2E+10
m=10           ergibt: -1.230E+10
m=12           ergibt: -1.23000E+10
m=13           ergibt:  -1.23000E+10
```

Wird die Form (e:m:n) benutzt,so wird die Zahl e in Festkommadarstellung ausgegeben, wobei n die Zahl der Nachkommastellen angibt. Solange die Laenge m nicht ausreichend grosz ist, werden keine fuehrenden Leerzeichen ausgegeben. Wenn n=0 ist, ist die Ausgabe eine ganze Zahl. Falls e zu grosz ist, um in dem angegebenen Feld dargestellt zu werden, erfolgt die Ausgabe im wissenschaftlichen Format (siehe oben).

Beispiel:

```
WRITE(1E2:6:2)    ergibt:  100.00
WRITE(1E2:8:2)    ergibt:   100.00
WRITE(23.455:6:1) ergibt:   23.5
WRITE(23.4554:2)  ergibt:   2.34550E+01
WRITE(23.455:4:0) ergibt:   23
```

4) e ist vom CHARACTER- oder STRING-Typ:

Sowohl (e) als auch (e:m) koennen verwendet werden. Das Zeichen oder der String werden mit einer minimalen Laenge von 1 (bei Zeichen) oder Laenge der Strings (bei STRING-Typen) ausgegeben. Fuehrende Leerzeichen werden angefuegt, wenn m ausreichend grosz ist.

5) e ist von BOOLEAN-Typ:

(e) und (e:m) koennen verwendet werden. 'TRUE' oder 'FALSE' werden in Abhaengigkeit vom boolschen Wert e ausgegeben, wobei eine minimale Laenge von 4 bzw. 5 verwendet wird.

WRITELN

Ausgaben mittels WRITELN schlieszen mit Zeilenvorschub/Wagen-ruecklauf ab, d.h. mit einem WRITE (CHR(13)).

```
WRITELN(P1,P2......,P3); entspricht
BEGIN  WRITE(P1,P2,...,P3);WRITELN END;
```

READ

Die Prozedur READ liest Daten von der Tastatur. Dies erfolgt ueber einen Puffer, der sich in den Runtimes befindet. Dieser ist anfangs leer (bis auf eine Zeilenende-Markierung). Man kann sich den Zugriff auf diesen Puffer so vorstellen, dasz ein Textfenster ueber den Puffer gelegt wird, durch welches jeweils ein Zeichen sichtbar ist. Wenn dieses Textfenster ueber einer Zeilenende-Markierung liegt, wird vor dem Abschlusz der READ-Operation eine neue Textzeile von der Tastatur in den Puffer gelesen.

```
READ (V1, ....,Vn); entspricht:
BEGIN READ (V1); READ(V2);.....;READ(Vn) END;
```

wobei V1, V2, usw. vom Typ CHARACTER, STRING, INTEGER oder REAL sein muessen.
4 Faelle sind zu betrachten:

1) V ist vom CHARACTER-Typ

In diesem Fall liest READ (V) nur ein Zeichen aus dem Eingabepuffer und weist es V zu. Wenn das Textfenster ueber einer Zeilenmarkierung (CHR (13)) liegt, liefert die Funktion EOLN den Wert TRUE, und eine neue Textzeile wird von der Tastatur gelesen. Wenn anschlieszend eine READ-Operation ausgefuehrt wird, wird das Textfenster ueber den Beginn der neuen Zeile gelegt.
Achtung! Nach dem Start des Programms ist EOLN=TRUE, d.h. falls zuerst ein READ eines CHARACTER-Typ erfolgt, wird ein CHR(13) uebergeben und daraufhin eine neue Zeile von der Tastatur gelesen. Ein anschlieszendes READ eines CHARACTERs uebergibt das erste Zeichen dieser Zeile, vorausgesetzt, sie ist nicht leer. Siehe auch unter READLN.

2) V ist vom STRING-Typ

Wird ein String mittels READ gelesen, so werden soviele Zeichen eingelesen, wie bei der Stringdefinition als Laenge angegeben wurden, bzw. soviele bis EOLN=TRUE ist. Falls der String durch READ nicht gefuellt wird (d.h. falls das Zeilenende erreicht ist, bevor das Stringende erreicht ist), wird der Rest des Strings mit CHR(0) aufgefuellt. Das ermoeglicht dem Programmierer die Laenge des eingelesenen Strings zu ermitteln.
Die unter 1) gemachte "Achtung !"- Bemerkung gilt auch hier.

3) V ist vom INTEGER-Typ

In diesem Fall wird eine Reihe von Zeichen eingelesen, die eine INTEGER-Zahl darstellen. Alle vorausgehenden Leerzeichen und Zeilenende-Markierungen werden uebergangen. (Das bedeutet, dasz INTEGER-Zahlen direkt eingelesen werden koennen).
Wenn die eingelesene Zahl groeszer als MAXINT (32767) ist, wird der Runtime-Fehler 'Zahl zu grosz' ausgegeben und das Programm gestoppt. Wenn das erste eingelesene Zeichen (nachdem Leerzeichen und Zeilenende-Markierung uebersprungen wurden) keine Ziffer oder Vorzeichen ist, wird der Fehler 'Zahl erwartet' angezeigt und das Programm abgebrochen.

4) V ist vom REAL-Typ

Hierbei wird eine Zeichenfolge eingelesen, die eine REAL-Zahl darstellt. Alle fuehrenden Leerzeichen und Zeilenende-Markierungen werden uebergangen, und wie bei 3) musz das erste andere Zeichen eine Ziffer oder Vorzeichen sein. Wenn die Zahl zu grosz oder zu klein ist, wird der Fehler 'Ueberlauf' angezeigt, wenn 'E' ohne nachfolgendes Vorzeichen oder Ziffer eingelesen wird, tritt Fehler 'Exponent erwartet' auf, und wenn ein Dezimalpunkt ohne nachfolgende Ziffer gelesen wird, kommt es zum Fehler 'Zahl erwartet'.

READLN

```
READLN(V1,V2,....,Vn); entspricht:
EGIN READ(V1,V2,....,Vn); READLN END;
```

READLN liest einen neuen Pufferinhalt von der Tastatur.
Nach der Ausfuehrung von READLN wird EOLN = FALSE, es sei denn, die naechste Textzeile ist leer.
READLN kann verwendet werden, um die zu Beginn der Programmausfuehrung vorhandene leere Zeile zu ueberspringen, d.h. es wird ein neuer Puffer gelesen. Diese Masznahme ist nuetzlich, wenn zu Beginn eines Programms ein CHARACTER eingelesen werden soll, aber nicht notwendig, wenn Zahlen (da Zeilenende-Markierungen uebersprungen werden) oder Zeichen von spaeteren Zeilen eingelesen werden sollen.

PAGE

Die Prozedur PAGE entspricht einem WRITE (12); und bewirkt ein Loeschen des Bildschirmes.

EOLN

EOLN ist eine boolsche Funktion, die TRUE liefert, wenn das naechste zu lesende Zeichen eine Zeilenende-Markierung (CHR(13)) ist, sonst ist sie FALSE.

3.2. **Konvertierungsfunktion**

TRUNC (X)

Der Parameter X musz vom Typ REAL oder INTEGER sein. Der uebergebene Wert ist die groeszte INTEGER-Zahl, die kleiner gleich X ist, wenn X positiv ist bzw. die kleinste INTEGER-Zahl, die groeszer gleich X ist, wenn X negativ ist.

```
TRUNC (-1.5) liefert -1
TRUNC (1.9)  liefert  1
```

ROUND (X)

X musz vom Typ REAL sein. Die Funktion liefert die "naechstliegende" INTEGER-Zahl entsprechend den normalen Rechnungsvorschriften.

```
ROUND (-6,5) liefert -6  ROUND(11.7)liefert 12
ROUND (-6.51)  "     -7  ROUND(23.5)  "     24
```

ENTIER (X)

X musz REAL oder INTEGER sein. ENTIER liefert die groeszte INTEGER-Zahl die kleiner gleich X ist.

```
ENTIER(-6.5) liefert -7   ENTIER(11.7) liefert 11
```

ORD (X)

X kann jeder skalare Typ auszer REAL sein. Der uebergebene Wert ist eine INTEGER-Zahl - die Ordnungszahl von X innerhalb der Menge (SET), die den Typ von X festlegt.
Wenn X vom INTEGER-Typ ist, ist ORD(x)=x; das sollte normalerweise vermieden werden.

```
ORD('a') liefert 97
```

CHR(X)

X musz INTEGER sein. CHR liefert einen CHARACTER, der dem ASCII-Wert von X entspricht.
CHR(49) liefert '1' CHR(91) liefert '['

3.3. Arithmetische Funktionen

In diesem Abschnitt musz der Parameter X immer vom Typ REAL oder INTEGER sein.

ABS(X) liefert den Absolutwert von X (d.h. ABS(-4.5) =4.5) Das Ergebnis ist vom gleichem Typ wie X.

SQR(X) liefert den Wert X*X, d.h. das Quadrat von X. Das Resultat ist vom gleichen Typ wie X.

SQRT(X) liefert die Wurzel aus X. Das Resultat ist immer vom Typ REAL.
Ein 'mathematischer Fehler' tritt auf, wenn X negativ ist.

FRAC(X) liefert den gebrochenen Anteil von X.
FRAC(X) = X - ENTIER(X).
FRAC(1.5) = 0.5 FRAC(-12.56) = 0.44

SIN(X) liefert den Sinus von X. X wird im Bogenmasz (radian) angegeben. Das Resultat ist immer REAL.

COS(X) liefert den Cosinus von X, weiter wie SIN

TAN(X) liefert den Tangens von X, weiter wie SIN

ARCTAN(X) liefert den Winkel im Bogenmasz, dessen Tangens gleich X ist. Das Ergebnis ist REAL.

EXP(X) liefert den Wert e X (mit e=2.71828). Das Resultat ist REAL.

LN(X) liefert den natuerlichen Logarithmus von X. Das Resultat ist REAL. Wenn X kleiner gleich Null ist, kommt es zum Fehler 'mathematischer Fehler'.

3.4. Weitere vordefinierte Prozeduren

NEW(p)

Mittels NEW(p) wird dynamischen Variablen Speicherplatz zugewiesen. p ist eine Zeiger-Variable, und nachdem NEW(p) ausgefuehrt wurde, enthaelt p die Adresse der neu zugewiesenen dynamischen Variablen. Der Typ der dynamischen Variablen ist der gleiche Typ wie der der Zeiger-Variablen p, und dies kann jeder Typ sein.
Um auf die dynamische Variable zuzugreifen, wird p benutzt. Um den Speicherplatz wieder freizugeben, werden die Prozeduren MARK & RELEASE benutzt (s. unten).

MARK(vl)

Die Prozedur sichert den Zustand der Menge ("heap") der dynamischen Variablen in der Zeigervariablen vl. Der Zustand der Menge, wie sie zum Zeitpunkt der Ausfuehrung von MARK vorlag, kann durch die Prozedur RELEASE (s. unter) wieder hergestellt werden. Der Typ der Variablen, auf die v1 zeigt, ist gleichgueltig, da vl nur mit MARK & RELEASE verwendet werden darf, niemals mit NEW.

RELEASE(vl)

Mittels dieser Prozedur wird vom Ende des Bereichs der dynamischen Variablen her Speicherplatz fuer dynamische Variable freigegeben. Es wird wieder der Zustand hergestellt, der zum Zeitpunkt der Ausfuehrung von MARK(vl) bestand, d.h. alle spaeter erzeugten dynamischen Variablen werden zerstoert.

INLINE(C1,C2,C3,.....)

Diese Prozedur erlaubt das Einfuegen von Z-80 Maschinencode in das Pascal-Programm. Die Werte (C1 MOD 256, C2 MOD 256,.) werden an dieser Stelle (Adresse, an der sich der Compiler gerade befindet) in das Objekt-Programm einfuegt. C1, C2, C3, usw. sind INTEGER-Konstanten.

USER(V)

V ist ein INTEGER-Argument. Die Prozedur bewirkt den Aufruf eines Maschinenprogrammes an der Adresse V. Da KC-PASCAL INTEGER-Zahlen in Zweierkomplement-Form bearbeitet, muessen Adressen, die groeszer als 7FFFh (32767) sind, als negative Zahlen eingegeben werden. Z.B.: 0C000H ist -16384 und somit bewirkt USER(-16384) einen Aufruf an der Speicherstelle 0C000H.
Wenn Konstanten verwendet werden, um Aufrufe zu erzeugen, ist es also einfacher, hexadezimale Zahlen zu verwenden. Die aufgerufene Routine musz mit einem Z-80 RET-Befehl (0C9H) enden und darf den Wert im IX-Register nicht veraendern.

HALT

Die Prozedur stoppt die Programmausfuehrung mit der Meldung 'Halt bei PC=XXXX', wobei xxxx die hexadezimale Adresse darstellt, an der das HALT ausgegeben wurde. Zusammen mit dem Compilerlisting kann HALT verwendet werden, um festzustellen, welcher von zwei oder mehr Programmzweigen durchlaufen wurde, es wird also normalerweise bei der Fehlersuche benutzt.

POKE(X,V)

POKE schreibt den Ausdruck V angefangen von der Adresse X in den Speicher. X ist vom INTEGER-Typ und V kann von jedem Typ auszer SET sein.

Beispiele:

```
POKE(#6000,'A') schreibt #46 an Adresse #6000
POKE (-16384,3.6E3) schreibt 00 0B 80 70 (in Hex)
ab Adresse #C000
```

TOUT(NAME,START,SIZE)

Mit TOUT werden Variablen auf Band gespeichert. Der NAME ist vom Typ ARRAY[1...8] OF CHAR. Es werden SIZE (=Anzahl) Bytes aus dem Speicher, angefangen von Adresse START, abgespeichert. Diese beiden Parameter sind vom Typ INTEGER.
z.B.: Um die Variable V unter dem Namen 'VAR ' zu speichern, ist zu schreiben:

```
TOUT('VAR',ADDR(V),SIZE(V))
```

Die Benutzung von konkreten Speicheradressen ermoeglicht dem Programmierer mehr Flexibilitaet als nur die Moeglichkeiten, Arrays zu speichern. Wenn ein System beispielsweise mit mehreren Bildschirminhalten (memory mapped screen) arbeitet, koennen ganze Bildschirminhalte unmittelbar gesichert werden.

TIN(NAME,START)

Mit dieser Prozedur koennen durch TOUT gesicherte Variablen usw. wieder geladen werden. Fuer die Typen von NAME und START gilt das oben gesagte. Das Band wird nach dem File mit dem Namen NAME abgesucht ('?' fuer beliebige Zeichen ist zugelassen), welches dann ab der Adresse START eingelesen wird. Die Anzahl der zu ladenden Bytes wird vom Band entnommen (entspricht den gesicherten).
Z.B.: Um die nmit TOUT gesicherte Variable zu laden
schreibt man:

```
TIN('VAR',ADDR(V))
```

Da Quelltexte (source files), die durch den Editor gesichert wurden, das gleiche Aufzeichnungsformat haben, wie es auch von TOUT benutzt wird, koennen mit TIN Quelltexte in ARRAYs of CHAR geladen werden, um sie weiter zu verarbeiten.

OUT(P,C)

Diese Prozedur spricht unmittelbar den Z-80-Output-Port an, ohne dasz INLINE benutzt werden musz. Der Wert des INTEGER-Parameters P wird in das BC-Register und der CHARACTER-Parameter C in das A-Register geladen, dann wird der Assembler-Befehl OUT(C), A ausgefuehrt.
z.B.: OUT(l,'A') gibt das Zeichen 'A' an den Z-80 Port l aus.

3.5. Manipulation von INTEGER-Zahlen

HI(I)
Das Ergebnis der Funktion liefert den Wert des hoeherwertigen Byte der INTEGER-Zahl I (Ergebnistyp ist INTEGER).

LO(I)

Das Ergebnis dieser Funktion liefert den Wert des niederwertigen Byte der INTEGER-Zahl I (Ergebnistyp ist INTEGER)

SWAP(I)

Mit dieser Funktion werden High-Byte und Low-Byte der INTEGER-Zahl I

vertauscht

SHL(I,N)

Die INTEGER-Zahl I wird um N (Typ INTEGER) Bitstellen bitweise nach links verschoben. Das Ergebnis ist vom Typ INTEGER.

SHR(I,N)

Die INTEGER-Zahl I wird um N (Typ INTEGER) Bitstellen bitweise nach rechts verschoben. Das Ergebnis ist vom Typ INTEGER.

BAND(A,B)

Die INTEGER-Zahl A wird mit der INTEGER-Zahl B bitweise UND-verknuepft. Das Ergebnis ist vom Typ INTEGER.

BOR(A,B)

Die INTEGER-Zahl A wird mit der INTEGER-Zahl B bitweise ODER-verknuepft. Das Ergebnis ist vom Typ INTEGER.

BIOR(A,B)

Die INTEGER-Zahl A wird mit der INTEGER-Zahl B EXCLUSIV-ODER-verknuepft. Das Ergebnis ist vom Typ INTEGER.

3.6. **Graphikprozeduren**

GOTO(X,Y)

X und Y sind INTEGER-Variablen. Der Cursor (fuer WRITE und WRITELN) wird auf die Spalte X und die Zeile T eingestellt.

SETC(V,W) [nur KC85/2 und KC85/3]

V und W sind INTEGER-Variablen. V bestimmt die Vordergrundfarbe (Schrift, Punkte usw.) von 0..15 und W die Hintergrundfarbe von 0..7 (siehe Handbuecher KC85/2 und KC85/3).

GETC(X,Y) [nur KC85/2 und KC85/3]

X und Y sind INTEGER-Variablen. Mit dieser Funktion kann die Farbe im Punkt X,Y des Pixel-Bildspeicher bestimmt werden. Das Ergebnis der Funktion ist vom Typ INTEGER.

PLOT(X,Y) [nur KC85/2 und KC85/3]

X und Y sind INTEGER-Variablen. Der Bildpunkt an der Position X,Y wird entsprechend der voreingestllten Farbinformation gesetzt.

CLRPLOT(X,Y) [nur KC85/2 und KC85/3]

X und Y sind INTEGER-Variablen. Der Bildpunkt an der Position X,Y wird geloescht (auf aktuelle Hintergrundfarbe gesetzt).

LINEPLOT(X1,T1,X2,T2) [nur KC85/3]

X1, X2, T1, und T2 sind INTEGER-Variablen. Es wird eine Linie von

Punkt X1,T1 zum Punkt X2,Y2 mit der zuletzt eingestelten Farbe (s. SETC) gezogen

CIRCLE(XM,YM,R) [nur KC85/3]

XM, YM und R sind INTEGER-Variablen. Um den Punkt XM, YM wird ein Kreis mit dem Radius R gezogen (die zuletzt eingestellte Farbe wird verwendet).

3.7. Weitere vordefinierte Funktionen

RANDOM

Das Ergebnis der Funktion ist eine INTEGER-Pseudo-Zufallszahl von 0..255.

SUCC(X)

X kann jeder skalare Typ auszer REAL sein. SUCC(X) liefert den Nachfolger von X.

SUCC('A') ergibt 'B' SUCC('5') ergibt '6'

PRED(X)

PRED liefert den Vorgaenger von X.

PRED('j') ergibt 'i' PRED(TRUE) ergibt FALSE

ODD(X)

X musz vom INTEGER-Typ sein. ODD liefert das boolsche Resultat TRUE, wenn X ungerade ist und FALSE, wenn X gerade ist.

ADDR(V)

V ist eine Variablen-Name beliebigen Typs. Die Funktion liefert die Speicheraddresse der Variablen V als INTEGER-Zahl.

PEEK(X,T)

X stellt eine Speicheradresse als INTEGER-Zahl dar. T ist der Typ, den die Funktion liefern soll (jeder Typ ist moeglich). PEEK wird benutzt, um Daten aus dem Speicher zu lesen. Bei allen PEEK- und POKE-Operationen werden Daten in der KC-PASCAL Darstellung gelesen bzw. geschrieben.
Z. B.: Wenn der Speicher ab der Adresse #6000 die HEX-Bytes 50 61 73 63 61 6C enthaelt, liefern verschiedene Peeks:

```
WRITE(PEEK(#6000,ARRAY[1..6] OF CHAR)) ergibt: 'PASCAL'
WRITE(PEEK(#6000,CHAR))                ergibt: 'P'
WRITE(PEEK(#6000,INTEGER))             ergibt: 24912
WRITE(PEEK(#6000,REAL))     ergibt: 2.46227E+29
```

SIZE(V)

V ist eine Variable. Das INTEGER-Resultat stellt die Anzahl der Bytes dar, die durch diese Variable belegt werden.

INP(P)

Mittels INP laeszt sich der Z80-Input-Port (U880) direkt (ohne Benutzung von INLINE) ansprechen. Der Wert von P wird in's BC-Register geladen. Das CHARACTER-Resultat wird dann durch Ausfuehrung des Assembler-Befehls IN A,(C) erhalten.

KEYPRESSED

Als Ergebnis wird eine Variable vom Typ BOOLEAN uebergeben. Wenn eine Taste gedrueckt wurde, ist das Ergebnis TRUE, sonst FALSE

READKBD

Diese Funktion liefert ein Zeichen vom Typ CHAR von der Tastatur (wartet auf Tastenbetaetigung).

4. Kommentare und Compiler-Steuerzeichen

4.1. Kommentare

Ein Kommentar kann ueberall zwischen zwei reservierten Worten, Zahlen, Namen oder Spezialsymbolen eingefuegt werden. Ein Kommentar beginnt mit dem Zeichen '{' oder dem Zeichenpaar '(*'. Auszer, wenn das naechste Zeichen ein '¤' ist, werden alle Zeichen bis zum naechsten '}' oder '*)' ignoriert.
Wird ein '¤' gefunden, erwartet der Compiler eine Folge von Compiler-Steuerzeichen (s. unten). Die auf diese Steuerzeichen folgenden Zeichen werden uebersprungen bis '}' oder '*)' gefunden wird.

4.2. Compiler-Steuerzeichen

Es gibt folgende Moeglichkeiten:

Steuerzeichen L:

Steuert das Listen von Programmtext und Objektcode-Adressen beim Compilieren.

Bei L+ wird ein vollstaendiges Listing ausgegeben.
Bei L- werden nur die fehlerhaften Zeilen gelistet.
Vorgabewert ist L+

Steuerzeichen O:

Legt fest, ob bestimmte Ueberlauftests gemacht werden. INTEGER-Multiplikationen und Divisionen und alle arithmetischen REAL-Operationen werden immer auf Ueberlauf getestet.
Bei O+ werden auszerdem die INTEGER-Addition und -Subtraktion getestet.
Bei O- nicht.
Vorgabewert ist O+.

Steuerzeichen C:

Legt fest, ob waehrend der Ausfuehrung des Object-Codes Tastaturueberpruefungen durchgefuehrt werden. Bei C+ wird die Programmausfuehrung nach Betaetigen von EDIT mit der Meldung 'Halt'

angehalten. Dieser Test wird am Beginn aller Schleifen, Prozeduren und Funktionen gemacht. Damit kann der Benutzer bei der Fehlersuche ueberpruefen, welche Schleifen usw. nicht ordnungsgemaesz beendet werden (Endlosschleife).
Bei C- werden obige Tests nicht durchgefuehrt und das Programm laeuft somit schneller.
Vorgabewert ist C+

Steuerzeichen P:

Mit diesem Steuerzeichen wird der Drucker parallel zum Bildschirm geschalten. Wird das Steuerzeichen wiederholt verwendet, so wird der Drucker wieder ausgeschalten. Nach dem Compilieren wird der Drucker immer deaktiviert.

Steuerzeichen S:

Legt fest, ob Stack-Tests durchgefuehrt werden.
Bei S+ wird am Beginn jeder Prozedur und jedes Funktionsrufes getestet, ob der Stack in diesem Block vermutlich ueberlaufen wird. Wenn der Stack droht, den Bereich der dynamischen Variablen oder das Programm zu zerstoeren, wird die Meldung '>> RAM bei PC=XXXX' ausgegeben und das Programm abgebrochen. Wenn innerhalb einer Prozedur ein groszer Stackbereich benoetigt wird, kann das Programm abstuerzen. Benoetigt andererseits eine Funktion sehr wenig Stack, waehrend sie rekursiv arbeitet, ist es u.U. unnoetig, diesen Test durchzufuehren.
Bei S- wird kein Stack-Test durchgefuehrt.
Vorgabewert ist S+

Steuerzeichen A:

Legt fest, ob getestet wird, dasz Feldindizes in dem durch die Felddeklaration festgelegten Bereich liegen. Bei A+ wird bie Bereichsueberschreitung die Meldung 'Index zu hoch' oder 'Index zu niedrig' ausgegeben und das Programm gestoppt.
Bei A- wird dieser Test nicht durchgefuehrt.
Vorgabewert ist A+

Steuerzeichen I:

Bei der Benutzung der 16 bit-Zweierkomplement-INTEGER-Arithmetik entsteht ein Ueberlauf bei der Ausfuehrung von >, <, >= oder <=, wenn sich die Argumente um mehr als MAXINT (32767) unterscheiden. In diesem Fall waere das Ergebnis des Vergleichs falsch. Sollen solche Zahlen nicht verglichen werden, hat das keine weiteren Folgen. I+ sichert aber auch bei einem solchen Vergleich ein richtiges Ergebnis. Eine aehnliche Situation kann auch beim Vergleich von REAL-Zahlen entstehen, wenn diese groeszer, mehr als etwa 3.4E38, werden, d.h. ein 'Ueberlauf'-Fehler wird ausgegeben.
Bei I- wird obige Korrektur nicht durchgefuehrt.
Vorgabewert I-

Alle Compiler-Steuerungen koennen selektiv eingesetzt, d.h. im Programm ein- und ausgeschalten werden. Somit koennen fehlerfreie Programmteile beschleunigt und zusammengedraengt werden, indem die entsprechenden Tests in unverschachtelten Programmteilen abgeschaltet werden.

5. Der eingebaute Editor

5.1. Einfuehrung

Der mitgelieferte Editor ist ein einfacher zeilenorientierter Editor, der ein einfaches, schnelles und wirksamens Editieren der Programme erlaubt.
Eine Eingabezeile kann maximal 80 Zeichen umfassen, weitere Zeichen werden ignoriert. Mit ENTER wird der editierte Text in einer kompakten Form im Quelltextspeicher abgelegt. Die Zahl der fuehrenden Leerzeichen einer Zeile wird in einem Byte am Anfang dieser Zeile abgespeichert. Alle reservierten KC-PASCAL-Woerter werden als Token ebenfalls in einem Byte abgespeichert. Der Editor wird nach dem Laden von KC-PASCAL automatisch aufgerufen und zeigt die Meldung:

```
***** PASCAL V4.4 *****
Bearb. VON +++ AM87 +++
   (VERSION XXXXXX)
```

gefolgt vom Zeichen '+' (Editor-Prompt) an.
Dabei ist XXXXXX die jeweilige Versionsbezeichnung je nach Geraetetyp
Im folgenden ist die Eingabe einer Kommandozeile des Formates:

```
              C N1,N2,S1,S2
```

gefolgt von 'ENTER' moeglich, wobei

```
C             das auszufuehrende Kommando
N1 und N2     Zahlen im Bereich 1 bis 32767 (Ausnahme 'Z')
S1 und S2     STRINGs OF CHAR mit einer maximalen Laenge
              von 20 darstellen
```

Das Komma wird benutzt, um die einzelnen Argumente zu trennen (Kann durch das S-Kommando veraendert werden.) Auszer innerhalb der Strings werden Leerzeichen ignoriert. Keines der Argumente ist obligatorisch, aber einige Kommandos (z.B. das 'D'elete-Kommando) werden nicht ausgefuehrt, ohne dasz N1 und N2 spezifiziert werden. Wird ein Argument nicht eingegeben, nimmt der Editor das zuletzt an dieser Stelle eingegebene. Die Werte von N1 und N2 werden anfangs auf 10 gesetzt, die Strings sind leer. Wird eine unzulaessige Zeile, wie F-1,100,HELLO eingeben, wird die Zeile ignoriert und 'Pardon?' ausgegeben. Diese Fehlermeldung wird auch ausgegeben, wenn die Laenge von S2 20 uebersteigt. Wenn S1 laenger als 20 ist, werden die ueberzaehligen Zeichen ignoriert. Editier-Kommandos koennen als Grosz- oder Kleinbuchstaben eingegeben werden.

5.2. Editierkommandos

5.2.1. Text einfuegen

Text kann in das Textfile eingefuegt werden, indem man eine Zeilennummer, gefolgt von mindestens einem Leerzeichen und dem gewuenschten Text eingibt, oder indem man das I-Kommando benutzt. (Wenn eine Zeilennummer gefolgt von 'ENTER', d.h. ohne nachfolgenden Text eingegeben wird, wird sie aus dem Textfile geloescht, falls sie existiert).

Beim Eingeben von Text koennen die Eingangs beschriebenen Steuertasten verwendet werden. Der Text wird immer erst in einen Eingabepuffer eingegeben. Ist dieser voll, wird das Eingeben weiterer Zeichen verhindert.

Kommando I n,m

I ruft den automatischen Einfuege-Modus auf. Die Zeilennummer werden, beginnend mit n und mit einem Inkrement von m, automatisch erzeugt und angezeigt. Der gewuenschte Text kann unter Benutzung der Steuerkommandos hinter der angezeigten Zeilennummer eingegeben und mit ENTER abgeschlossen werden.
Durch BREAK kann man den Einfuege-Modus wieder verlassen. Wenn eine eingegebene Zeile die gleiche Zeilennummer wie eine bereits existierende besitzt, wird die alte (nach ENTER) geloescht und durch die neu eingegebene ersetzt.
Wenn die automatische Inkrementierung eine Zeilennummer groeszer als 32767 erzeugen wuerde, wird der Einfuege-Modus automatisch verlassen. Wenn beim Eingeben von Text das Ende einer Bildschirmzeile erreicht wird, ohne dasz der Textpuffer voll ist (80 Zeichen), wird der Bildschirm gerollt und auf der folgenden Zeile kann weiterer Text eingegeben werden. Der Text wird automatisch eingerueckt, so dasz die Zeilennummer vom Text abgehoben sind.

5.2.2. **Text Listen**

Der Text kann mittels des L-Kommandos durchgesehen werden. Die Anzahl der auf einmal gelisteten Zeilen kann durch das K-Kommando festgelegt werden.

Kommando L n,m (List)

Listet den Text von Zeilennummer n bis m. Der Vorgabewert fuer n ist immer 1 und der fuer m immer 32767, d.h. er wird nicht von vorher eingegebenen Argumenten uebernommen. Um den gesamten Textspeicher zu listen, kann also einfach 'L' benutzt werden. Die Zeilen werden auf dem Bildschirm mit einem linken Rand ausgegeben, so dasz die Zeilennummer deutlich erkennbar ist. Wenn die durch das K-Kommando bestimmte Zeilenzahl ausgegeben und m noch nicht erreicht ist, wird das Listen unterbrochen. BREAK bewirkt die Rueckkehr zur Editor-Hauptschleife, jede andere Taste setzt das Listen fort.

Kommando K n

Legt die Zahl der Zeilen fest, bis das Listen anhaelt. Der Wert (n MOD 256) wird berechnet und gespeichert.
Vorgabewert ist n=15.

5.2.3. **Text Editieren**

Verschiedene Kommandos ermoeglichen es, einen erzeugten Text zu veraendern, zu loeschen, zu verschieben und neu zu nummerieren.

Kommando D n,m (Delete)
Alle Zeilen von n bis m werden aus dem Textfile geloescht. Wenn m<n ist oder weniger als 2 Argumente eingegeben wurden, wird nichts unternommen, der Text also vor versehentlich falschen Eingaben geschuetzt. Eine einzelne Zeile kann durch m=n geloescht werden. Das kann aber auch durch einfaches Eingeben der Zeilennummer gefolgt von ENTER erreicht werden.

Kommando M n,m (Move)
Die Textzeile n wird auf die Textzeile m kopiert und loescht evtl. dort stehenden Text. Die alte Zeile n wird nicht geloescht! Somit

kann eine Textzeile an eine andere Position des Textfiles gesetzt werden. Falls eine Zeilennummer n nicht existiert, geschieht nichts.

Kommando N n,m (Renumber)
Durch N wird das Textfile neu durchnummeriert. Die neue erste Zeilennummer ist n, der Abstand m. Sowohl m als auch n muessen angegeben werden. Wenn die Neunummerierung eine Zeilennummer groeszer als 32767 erzeugen mueszte, wird die alte Nummerierung beibehalten.

Kommando F n,m,f,s (Find)
Der Text im Zeilenbereich n<x<m wird nach dem String f durchsucht. Wird f gefunden, so wird die entsprechende Zeile angezeigt und der Edit-Modus (s. unten) aufgerufen. Weitere Kommandos im Edit-Modus koennen verwendet werden, um nach einen weiteren Vorkommen von f im definierten Zeilenbereich zu suchen oder um den gerade gefundenen String f durch den String s zu ersetzen und dann nach einem weiteren Auftreten von f zu suchen. ((s. auch unten)) Beachten Sie, dasz der Zeilenbereich und die Strings durch vorhergehende andere Kommandos gesetzt sein koennen, so dasz die Suche u. U. nur durch 'F' initiert werden kann.

Kommando E n (Edit)

Die Zeile mit der Nummer n wird editiert. Falls n nicht existiert, geschieht nichts. Anderenfalls wird die Zeile in einen Puffer kopiert und auf dem Bildschirm angezeigt. Die Zeilennummer wird darunter nochmals angezeigt und der Edit-Modus aufgerufen. Alle nachfolgenden Handlungen finden im Puffer, nicht im eigentlichen Text statt. Somit kann der urspruengliche Zustand jederzeit wieder hergestellt werden. In diesem Modus stellt man sich einen Zeiger vor, der sich, beginnend vom ersten Zeichen, durch die Zeile bewegt.

Folgende Sub-Kommandos werden zum Editieren der Zeile benutzt:

->	- bewegt den Text-Zeiger ein Zeichen weiter
<-	- bewegt den Text-Zeiger ein Zeichen zurueck, (hoechstens bis zum ersten Zeichen der Zeile)
^	- bewegt den Text-Zeiger zur naechsten TAB-Position, (aber hoechstens bis zum Zeilenende)
ENTER	- die editierte Zeile wird so uebernommen
Q	- (QUIT) bricht das Editieren ab, d.h. belaeszt die Zeile in ihrem alten Zustand (vor dem Aufruf des Edit-Modus)
R	- ("RELOAD") laedt den Textpuffer nochmals mit der alten Version der Zeile, d.h. macht alle Aenderungen rueckgaengig
L	- ("LIST") listet den Rest der editierten Zeile, d.h. die Zeile rechts von der gegenwaertigen Zeigerposition. Der Edit-Modus bleibt bestehen und der Zeiger wird wieder an den Beginn der Zeile gesetzt.
K	- ("KILL") loescht das Zeichen an der aktuellen Zeigerposition
Z	- loescht alle Zeichen ab (und einschlieszlich) der aktuellen Zeigerposition bis zum Ende der Zeile
F	- ("FIND") sucht das naechste Auftreten des vorher mit dem Kommanod 'F' definierten Suchstring f (s. oben). Dieses Subkommando schlieszt

gleichzeitig das Editieren der gerade aufgerufenen Zeile ab (unter Beibehaltung aller Aenderungen), falls der String f in dieser Zeile nicht noch einmal auftritt. Wenn (im definierten Zeilenbereich), wird diese Zeile zum Editieren in den Puffer geladen und der Text-Zeiger auf den Beginn der Zeile gesetzt.

S - ("SUBSTITUTE") ersetzt den eben gefundenen String f durch den vorher definierten Austauschstring s und fuehrt dann automatisch das Sub-Kommando 'F' aus, d.h. sucht nach dem naechsten Auftreten von f. 'S' wird also zusammen mit 'F' benutzt, um sich durch Textfile hindurchzuarbeiten und jeden String f nach Beliegen gegen String s auszutauschen (weiter mit 'S') oder auch nicht (weiter mit 'F').

Achtung
Das Sub-Kommando 'S' darf nur unmittelbar nach einem 'F'-Sub-Kommando oder 'S'-Sub-Kommando benutzt werden.

I - ("INSERT") fuegt Zeichen an der gegenwaertigen Zeigerposition ein. Dieser Sub-Modus wird beibehalten, bis man mit ENTER in den normalen Edit-Modus zurueckgekehrt. Der Zeiger weist dann auf die Position hinter dem letzten eingefuegten Zeichen. In diesem Sub-Modus loescht DELETE das Zeichen vor dem Zeiger aus dem Textpuffer, waehrend '^' den Zeiger bis zur naechsten TAB-Position vorruecken laeszt, wobei Leerzeichen eingefuegt werden.

X - setzt den Zeiger auf das Ende der Zeile und ruft automatisch den I-Sub-Modus auf (s. oben)

C - ("CHANGE") Das Zeichen an der aktuellen Zeigerposition wird durch das eingegebene ueberschrieben und der Zeiger eine Stelle weitergerueckt. Auch dieser Sub-Modus wird beibehalten, bis man durch Druecken von ENTER in den normalen EDIT-Modus zurueckkehrt. Der Zeiger weist dann auf die Position hinter dem letzten geaenderten Zeichen. DELETE bewegt den Zeiger in diesem Sub-Modus einfach ein Zeichen nach links. '->' hat keine Wirkung.

5.2.4. **Tonband-Kommandos**

Der Text kann mittels der Kommandos 'P' und 'G' auf Band gesichert bzw. vom Band geladen werden.

Kommando P n,m,s (Put)

Der Zeilenbereich n<x<m wird im KC-PASCAL-Format unter dem Filename s auf Band gesichert. Beachten Sie, dasz die Argumente durch vorhergehende Kommandos gesetzt sein koennen (s. "V)". Vor Abschlusz dieses Kommandos schalten Sie bitte den Kassettenrecorder, auf Aufnahme.

Kommando G ,,s (Get)

Das Band wird nach einem File in KC-PASCAL-Format mit dem Namen s

abgesucht. Ist der richtige Filename gefunden, wird der Text in den Speicher geladen. Gefundene Files werden angezeigt. Fragezeichen im gesuchten Filenamen fuehren zum Ueberlesen des jeweiligen Zeichens im gesuchten File. Wird '????????' eingegeben, wird das naechste File eingelesen, gleich welchen Namen es besitzt (Typ 'PAS' ist erforderlich). Das Einlesen kann mittels BREAK abgebrochen werden. Wenn bereits ein Textfile im Speicher vorhanden ist, wird das neu geladene File an das existierende Textfile angehangen und danach der gesamte Text, beginnend mit Zeilennummer 1 in 1er Schritten, neu durchnummeriert.

5.2.5 **Comilieren und Probelauf**

Kommando C n (Compilieren)

Der Text wird, beginnend mit Zeilennummer n, compilert. Wird keine Zeilennummer angegeben, beginnt die Compilierung mit der ersten existierenden Zeile. Bei fehlerfreier Uebersetzung erfolgt die Abfrage 'Lauf?'. Bei Beantwortung mit 'J' wird das Programm ausgefuehrt.

Kommando R (Run)

Der eben erzeugte Objekt-Code wird gestartet, allerdings nur, wenn der Quelltext mittlerweile nicht erweitert wurde.

Kommando T n (Translate)

Der Quelltext wird wiederrum, beginnend mit Zeile n (bzw. ab der ersten existierenden Zeile), compiliert. War die Uebersetzung erfolgreich, wird die Frage 'Ok?' ausgegeben. Antwortet man mit 'J', wird der erzeugten Objekt-Code an das Ende der Runtimes verschoben und zusammen mit diesen auf Band gespeichert (Dabei wird der Compiler zerstoert !). Schalten Sie vor Beantworten der Frage mit 'J' Ihr Magnetbandgeraet auf Aufnahme. Als Filename wird der letzte definierte Suchstring verwendet. Wird dieses File zu einem spaeteren Zeitpunkt in den Rechner geladen, kann es als wie ueblich unter dem Namen des zuletzt verwendeten String S1 wieder gestartet werden.
Beachten Sie, dasz sich der Objekt-Code nach der Antwort 'J' am Ende des Runtimes befindet, d.h. dasz der Compiler nach Ausfuehrung von 'T' neu geladen werden musz. Das bringt jedoch keine Probleme, da Sie ja wahrscheinlich nur ein vollstaendig ausgetestetes, fehlerfreies Programm mit 'T' uebersetzen. Antwortet man auf die Frage 'Ok?' nicht mit 'J', sondern mit einer anderen Taste, wird der Objekt-Code nicht verschoben und die Steuerung an den Editor zurueckgegeben.

5.2.6. **Systemzellen**

Kommando V (Editorvariablen)

Mit diesem Kommando werden die Editorvariablen in der Reihenfolge N1,N2,S1,S2 angezeigt. Als Trennzeichen wird das aktuell vereinbarte verwendet. Damit kommen die aktuellen Zeilennummern und Suchstrings (auch fuer Tape-E/A) zur Anzeige.

Kommando X (Systemzellen)

Es werden die Systemzellen in folgender Reihenfolge angezeigt:

wwww xxxx yyyy zzzz
Dabei ist wwww die Anfangsadresse des PASCAL-Quelltextes, xxxx die Endeadresse des Quelltextes, yyyy die Endeadresse des Arbeitsspeichers und zzzz die Endeadresse des Arbeitsspeichers fuer uebersetzte Programme nach dem Kommando 'T'.

Kommando Z n,m (Systemzellen stellen)

Mit diesem Kommando koennen Sie die Endeadresse des Arbeitsspeichers n (s. xxxx oben) und die Endeadresse des Arbeitsspeichers m fuer uebersetzte Programm neu waehlen. Damit ist es moeglich, Speicherbereich fuer andere Aufgaben zu reservieren, aber auch die uebersetzten Programme in Rechnersystemen mit weniger RAM laufen zu lassen. Die Adresseingabe erfolgt **dezimal**. Nach dem Kommando erfolgt ein Neustart des KC-PASCAL.

ACHTUNG!
Liegen die eingegebenen Werte nicht im RAM, im Compiler usw., hat das einen unweigerlichen Systemabsturz zur Folge.
Nach Ausfuehren des Kommando 'Z' wird ein Kaltstart ausgefuehrt und vorhandener Quelltext geloescht. Benutzen Sie dieses Kommando nur nach dem Kaltstart oder nachdem Sie Ihr Programm abgespeichert haben.

5.2.7. **Weitere Kommandos**

Kommando B (Betriebssystem)

- bewirkt eine Rueckkehr zum Betriebssystem.

Kommando O n,m (Tokenisieren)

KC-PASCAL speichert alle reservierten Worte in tokenisierter Form (1 Byte) und fuehrende Leerzeichen einer Zeile in einem weiteren Byte. Wenn Sie jedoch Text in vollstaendiger, ausgeschriebener Form vorliegen haben (vielleicht von einem anderen Editor oder z.B. TURBO-PASCAL-Quellprogramme), kann dieser mit dem Kommando O tokenisiert werden. Der Text wird in der Quellform in einen Puffer gelesen und danach in tokenisierten Form zurueckgeschrieben. Das dauert u.U. eine gewisse Zeit. Dabei ist n die erste neue Zeilennummer, die aus dem File mit Namen s gebildet wird. Ist m = 1, werden alle Buchstaben des File s in Groszbuchstaben gewandelt. Ist m = 2, bleibt die urspruengliche Schreibweise erhalten.
Beachten Sie bitte, dasz die Programmzeilen durch CR/LF (0dh,0ah) getrennt sein muessen und die Datei mit EOF (01ah) enden musz (n – Mode bei WordStar bzw. TP). Weiterhin musz die Datei einen Vorblock (080h Byte) analog dem KC-Kassettenverfahren besitzen.
Zur Tokinisierung von TURBO-PASCAL-Quellen sollten Sie die CP/M-Version von KC-PASCAL benutzen, da dabei beim Einlesen von Diskette automatisch eine Unterscheidung zwischen Quelltext und tokonisierter Form erfolgt und der Vorblock nicht benoetigt wird. Die Laenge des Quelltextes darf 16 Kbyte nicht uebersteigen.

Kommando S,,d (Seperator-Delemiter)

Dieses Kommando ermoeglicht eine Abaenderung des Trennzeichens, welches zur Abgrenzung der Argumente in der Kommandozeile benutzt wird. Als neues Trennzeichen wird das erste Zeichen des Strings d benutzt. Nach dem Kaltstart wird das Komma benutzt. Beachten Sie,

dasz ein definiertes Trennzeichen solange benutzt werden musz (auch im 'S'-Kommando), bis ein neues definiert wurde.
Ein Leerzeichen darf nicht als Trennzeichen definiert werden.

6. Fehlermeldungen

6.1. Compiler-Fehlermeldungen

1. Zahl zu grosz
2. Semicolon wird erwartet
3. Nichtdeklarierter Name
4. Name wird erwartet
5. In der Deklaration einer Konstanten musz '=' und nicht ':=' verwendet werden
6. '=' wird erwartet
7. Eine Anweisung darf nicht mit diesem Namen beginnen
8. ':=' wird erwartet
9. ')' wird erwartet
10. Falscher Typ
11. '.' wird erwartet
12. Ein Faktor wird erwartet
13. Eine Konstante wird erwartet
14. Dieser Name ist keine Konstante
15. 'THEN' wird erwartet
16. 'DO' wird erwartet
17. 'TO' oder 'DOWNTO wird erwartet
18. '(' wird erwartet
19. Dieser Ausdrucktyp kann nicht geschrieben werden
20. 'OF' wird erwartet
21. ',' wird erwartet
22. ':' wird erwartet
23. 'PROGRAM' wird erwartet
24. Eine Variable wird erwartet, da der Parameter ein variabler Parameter ist
25. 'BEGIN' wird erwartet
26. Bei diesem READ-Aufruf wird eine Variable erwartet
27. Ausdruecke dieses Typs koennen nicht verglichen werden
28. Eine INTEGER- oder REAL-Zahl wird erwartet
29. Dieser Variablentyp kann nicht gelesen werden
30. Dieser Name ist kein Typ
31. Bei reellen Zahlen wird ein Exponent verlangt
32. Ein skalarer Ausdruck (kein numerischer) wird erwartet
33. Leerstrings sind nicht erlaubt (CHR (0) benutzen
34. '[' wird erwartet
35. ']' wird erwartet
36. Ein Arrayindex musz vom Typ SCALAR sein
37. '..' wird erwartet
38. In der Deklaration eines ARRAYS wird ']' oder ',' verlangt
39. Obergrenze ist groeszer als die untere
40. Das Set ist zu grosz (groeszer als die 256 moeglichen Elemente)
41. Das Ergebnis der Funktion musz vom Typ Name sein
42. Im Set wird ',' oder ' ' erwartet
43. Im Set wird '..' oder ']' erwartet
44. Der Parameter musz vom Typ Name sein
45. Ein leeres Set kann nicht erster Faktor in einer Nicht-Zuweisungs-Anweisung sein
46. Ein Skalartyp (einschlieszlich reelen Zahlen) wird

erwartet
47. Ein Skalartyp (auszer reelen Zahlen) wird erwartet
48. Die Sets sind nicht vertraeglich
49. Sets koennen nicht mit '<' und '>' verglichen werden
50. 'FORWARD', 'LABEL', 'CONST', 'VAR', 'TYPE' oder 'BEGIN' wird erwartet
51. Hexadezimalziffer wird erwartet
52. Sets koennen nicht gePOKEd werden
53. Das Array ist zu grosz
54. Bei der Definition von RECORDS wird 'END' oder '+' erwartet
55. Ein Feldname wird erwartet
56. Nach 'WITH' wird eine Variable erwartet
57. Die Variable bei WITH musz vom RECORD-Typ sein
58. Der Feldname wurde nicht mit der WITH-Anweisung in Verbindung gebracht
59. Nach 'LABEL' wird eine vorzeichenlose Integerzahl erwartet
60. Nach 'GOTO' wird eine vorzeichenlose Integerzahl erwartet
61. Diese Marke ist in einer falschen Programmebene
62. Marke ist nicht vereinbart
63. Der Parameter von SIZE musz eine Variable sein
64. Auf Zeiger kann nur der Gleicheitstest angewandt werden
67. Der einzig zulaessige WRITE-Parameter fuer Integerzahlen mit zwei Doppelpunkten ist e:m:H
68. Strings duerfen keine Zeilenende-Zeichen enthalten
69. Der Parameter von NEW, MARK oder RELEASE musz eine Zeigervariable sein
70. Der Parameter von ADDR musz eine Variable sein

6.2. **Runtime-Fehlermeldungen**

Tritt waehrend des Programmlaufs ein Fehler auf, so wird eine der folgenden Meldungen gefolgt von 'bei PC=XXXX' ausgegeben, wobei XXXX die Speicheradresse darstellt, an der der Fehler auftrat. Meist wird die Fehlerquelle offensichtlich sein, falls jedoch nicht, kann das Compiler-Listing zu Rate gezogen werden, um festzustellen, an welcher Stelle des Programms der Fehler auftrat. XXXX dient dann als Crossreferenz. Gelegentlich liefert das jedoch NICHT das richtige Resultat (Folgefehler).
1. Halt
2. Ueberlauf
3. >> RAM (auszerhalb des RAM)
4. / durch Null (Division durch Null; kann auch bei DIV auftreten)
5. Index zu niedrig
6. Index zu hoch
7. mathematischer Fehler (Fehler beim Aufruf einer mathematischen Routine)
8. Zahl zu grosz
9. Zahl erwartet
10. Exponent erwartet

Laufzeit-Fehler stoppen die Programmausfuehrung.

7. **Reservierte Worte und vordefinierte Namen**

7.1. **Reservierte Worte**

AND ARRAY BEGIN CASE CONST DIV DO

```
DOWNTO   ELSE        END      FORWARD  FUNCTION  GOTO IF
IN       LABEL       MOD      NIL      NOT       OF   OR
PACKED   PROCEDURE   PROGRAM  RECORD   REPEAT    SET  THEN
TO       TYPE        UNTIL    VAR      WHILE     WITH
```

7.2. Spezialsymbole

Die folgenden Symbole werden in KC-PASCAL benutzt und haben eine bestimmte Bedeutung:

```
+        -          *        /
=        <>         <        <=        >=        >
(        )          [        ]
{        }          (*       *)
^        :=         .        ,         ;         :
```

7.3. Vordefinierte Namen

Die folgenden Namen kann man sich als in einem das gesamte Programm umschlieszenden Block definiert denken. Demzufolge sind sie im gesamten Programm verfuegbar, es sei denn, sie wurden durch den Programmierer innerhalb eines Block umdefiniert.

```
CONST          MAXINT=32767; (INTEGER)
               PI=3.14159E+00; (REAL)
               TRUE; (BOOLEAN)
               FALSE; (BOOLEAN)

TYPE           BOOLEAN=(FALSE,TRUE);
               CHAR (Erweiterter ASCII-Zeichensatz)
               INTEGER=-MAXINT..MAXINT;
               REAL (Eine Untermenge der Reelen Zahlen)

PROCEDURE      WRITE; WRITELN; READ; READELN; PAGE; HALT;
               USER; POKE; INLINE; OUT; NEW; MARK; RELEASE;
               TIN; TOUT; GOTOXY
Nur KC85/2/3   SETC; GETC; PLOT; CLRPLOT
Nur KC85/3     LINEPLOT, CIRCLE

FUNCTION       ABS; SQR; ODD; RANDOM; ORD; SUCC; PRED; EOLN;
               PEEK; CHR; SQRT; ENTIER; ROUND; TRUNC;  FRAC;
               SIN; COS; TAN; ARCTAN; EXP; LN; ADDR;  SIZE;
               INP; KEYPRESSED; READKBD; SWAP;  HI;  LO; SHL;
               SHR; BOR; BAND; BXOR
```

8. Beispiel

```
 10 PROGRAM DEMO;
 20 VAR
 30 I:INTEGER;
 40 BEGIN
 50 I:=1;
 60  WHILE I < 10 DO
 70   BEGIN
 80    WRITE('Der Wert ist: ',I);
 90    WRITELN(' und I*I ist:',I*I);
100   I:=I+1;
110  END;
120  WRITELN('Programmende');
130 END.
```